FreewayML: An Adaptive and Stable Streaming Learning Framework for Dynamic Data Streams

Zheng Qin[†], Zheheng Liang[‡], Lijie Xu[†§], Wentao Wu[¶], Mingchao Wu[†], Wuqiang Shen[‡], Wei Wang[†]
[†]Institute of Software, Chinese Academy of Sciences
[‡]Guangdong Power Grid Limited Liability Company
[§]ETH Zürich
[¶]Microsoft Research

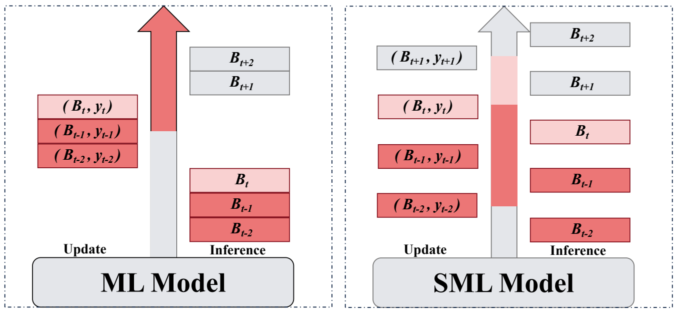
***Abstract*—Streaming (machine) learning (SML) can capture dynamic changes in real-time data and perform continuous updates. It has been widely applied in real-world scenarios such as network security, financial regulation, and energy supply. However, due to the sensitivity and lightweight nature of SML models, existing work suffers from low robustness, sudden decline, and catastrophic forgetting when facing unexpected data distribution drifts. Previous studies have attempted to enhance the stability of SML through methods such as data selection, replay, and constraints. However, these methods are typically designed for specific feature spaces and specific ML algorithms. In this paper, we introduce a shift graph based on the distances between data distributions and define three distinct data shift patterns. For these three patterns, we design three adaptive mechanisms, (a) multi-time granularity models, (b) coherent experience clustering, and (c) historical knowledge reuse, that are triggered by a strategy selector, with the goal of enhancing the accuracy and stability of SML. We implement an adaptive and stable SML framework, FreewayML, on top of PyTorch, which is suitable for most SML models. Experimental results show that FreewayML significantly outperforms existing SML systems in both stability and accuracy, with a comparable throughput and latency.**

I. Introduction

Data mining techniques have been applied in various domains to extract insights from real-time data generated by IoT sensors, web clicks, and other sources [1]–[4]. For example, users can detect cyber attacks through network traffic and connection types; power plants can supply energy based on real-time user consumption; and economists can forecast economic development trends according to financial markets [5]–[8]. Continuously arriving data are instrumental in developing robust and accurate AI-driven systems.

Existing work [9], [10] has already identified that data streams exhibit dynamic changes with non-iid distributions, attributed to the randomness of data generation, fluctuations of data quality, and the evolving trend of data distribution. As Figure 1a shows, traditional batch learning [11], [12] stores real-time data and conducts periodic training updates. This mode tends to face significant challenges when dealing with dynamically changing data, as its update frequency is relatively low and unable to promptly capture emerging patterns.

Unlike batch machine learning, streaming learning [13], [14] (SML) performs continuous incremental training on real-time data, discarding data once it is used, thus reducing memory space overhead and model update costs. Additionally, it naturally learns evolving data distribution trends, which, to a certain extent, can address the issue of dynamic changes. As Figure 1b shows, when labeled batch (B_{t-2}, y_{t-2}) flows in, SML incrementally updates the model using the mini-batch data B_{t-2}, capturing the new data distribution characteristics of the current moment. Although batch learning can also adopt the online-offline paradigm, it requires manual selection of update times and training data. Changes in data streams are often difficult to detect manually in a timely manner, and the continuous updating of SML naturally allows for quicker acquisition of new knowledge. SML frameworks, such as Flink ML [15], River [16], and Alink [17], have been developed and are widely used.



(a) Batch ML. (b) Streaming ML.

Fig. 1: Comparison between batch and streaming ML.

Given the high ingestion rates and dynamic changes of data streams, current research and implementations of SML primarily focus on sensitive and lightweight models, such as Streaming Logistic Regression and Streaming MLP based on mini-batch SGD. These models possess efficient processing and updating capabilities, meeting the practical demands of real-world scenarios. However, their sensitivity reduces model stability, and their lightweight structure diminishes generalization performance. We have identified the following shortcomings of existing methods:

(SC1) Low Robustness. For non-IID streaming data, a general challenge is how to improve model stability [18], [19] (i.e., reducing accuracy fluctuations). Slight shift in data distribution can cause decrease in accuracy. For example, since the data B_t at time t differs slightly from the previous B_{t-1} and model parameters θ_{t-1} are not fully suitable for B_t, the model's inference accuracy at the current moment can decrease.

(SC2) Sudden Decline. A major challenge in SML is how to handle the sudden severe shifts [20], [21] quickly, because the previously trained model is no longer applicable. For example, data B_{t+1} appears with a distribution that is significantly different from several previous distributions. At this point, the trained θ_t is completely unsuitable, resulting in a sharp decline of model accuracy.

(SC3) Catastrophic Forgetting. Forgetting is an unavoidable issue in SML [22]–[24] since models are updated incrementally. When old knowledge reoccurs, the model still needs to relearn, resulting in computational overhead and a decrease in accuracy. For example, if data B_{t-2} appears after B_{t+1}, the model parameters θ_{t+1} are not fully suitable for B_{t-2} .

To understand the impact of data distribution drifts better, we perform an empirical study on three real-world datasets (Section III). We find that three types of data shifts can lead to the low accuracy of current SML approaches: (a) slight shift, (b) sudden shift, and (c) reoccurring shift. Based on our findings, we propose three strategies to improve the accuracy and stability of SML, including (a) multi-time granularity models for slight shift, (b) coherent experience clustering for sudden shift, and (c) historical knowledge reuse for reoccurring shift (Section IV). We implement our approaches in a prototype framework, FreewayML (Section V). Our evaluation shows that FreewayML outperforms existing SML systems in terms of both stability and accuracy while achieving comparable throughput and latency (Section VI). For instance, we obtain improvement of accuracy by an average of **3.8%** across six datasets, with maximum improvement of **7.3%**, and FreewayML demonstrated higher stability.

We summarize our contributions in this paper as follows:

- We empirically find three common data distribution drift patterns that affect accuracy, and conduct an in-depth study including: pattern extraction (by analyzing the correlation between data distribution shifts and accuracy trends), pattern classification (by measuring the shift severity and shift range), and formal definitions of shift patterns.
- We design three adaptive mechanisms to enhance the stability and accuracy of SML in the presence of the three patterns defined, including (a) multi-time granularity models, (b) coherent experience clustering, and (c) historical knowledge reuse.
- We implement the proposed adaptive mechanisms in FreewayML. We develop a novel data structure called *adaptive streaming window* as well as other optimizations to improve the performance of FreewayML. [1]
- We conduct extensive experiments and thorough analysis that demonstrate the effectiveness of FreewayML. It outperforms existing SML systems in both stability and accuracy, with comparable throughput and latency.

[1]The source code of FreewayML are available at https://github.com/AnonymousforA/FreewayML

II. Background and related work

A. Streaming learning

1) Mini-batch: For streaming learning, the size of the mini-batch determine the frequency of inferences and model updates. If the mini-batch size is set to 1, each new piece of data that flows in will trigger inference or model update, resulting in lower latency and more sensitive updating. Conversely, setting a larger mini-batch size increases processing latency but reduces computational and communication overhead.

2) Inference and update: In streaming learning, inferences and updates occur in real time. Popular SML frameworks, such as Flink ML and River, separate the inference streams and training streams for individual processing. Once data reaches the size of a mini-batch, subsequent inferences or model updates are conducted. In terms of incremental model updates, algorithms based on mini-batch SGD are commonly used.

3) Shifts and forgetting: Due to dynamic changes of real-time data, data distribution shifts are extremely common in streaming scenarios. These shifts often accompany changes in rules of classification, posing significant challenges to both model inferences and updates. On the other hand, the problem of forgetting is a common and inevitable issue in SML. Limited generalization capability of models can lead to forgetting old knowledge because of incremental updates, which poses serious challenges to the usability of SML.

Data distribution shifts and catastrophic forgetting are two widely discussed problems in the field of streaming learning. Resolving these issues can effectively enhance the stability and accuracy of streaming learning.

B. Related work

Existing work primarily focuses on three aspects: (1) adaptive models, (2) data selection and replay, and (3) constrained learning. Adaptive models typically employ drift detection for full or incremental updates to capture characteristics of changing data streams. However, due to the inherent unpredictability, randomness, and uncertainty of changes presented in data streams, it is challenging to balance stability and responsiveness with this approach. Methods involving data selection and replay typically require specific settings that only work for certain models or parameter spaces and lack generality. Methods based on constrained learning and updating are influenced by model's generalization ability and struggle to accommodate fast-paced variation of data streams. Below, we discuss related work in more detail from these aspects.

1) Model adaptation: It has been extensively studied to improve the stability of models under dynamic changes [25]. Existing research [8], [20] has indicated that dynamic changes in data streams may have serious impact on model accuracy. Some previous work focused on designing concept drift detectors that capture potential data changes using methods based on accuracy or distribution, though these methods suffer from significant latency and uncertainty [26].

Popular model adaptation methods include [27]: (a) periodically training a new model using all available data [28],

(b) periodically fine-tuning existing models with newly added data [29], [30], and (c) continuously training models with streaming learning [15]–[17]. However, due to the inherent uncertainty of data streams, it is challenging for these approaches to balance between stability and sensitivity [31], [32].

T-SaS [33] improves accuracy on streaming data by employing a neural network architecture with a Bayesian approach to selectively activate subsets of the network. T-SaS focuses on handling reoccurring distribution shifts, which does not make targeted optimizations when facing slight shifts and new distribution. Similarly, SEED [34] selects an optimal domain network to handle specific tasks and fine-tunes this expert using data from the task. SEED represents each category using a Gaussian distribution and chooses the optimal domain network based on the similarity of these distributions.

2) Data selection and replay: Data selection [35]–[37] has been proposed to enhance stability and reduce computational overhead by selecting high-quality streaming data. The concept of data selection is often studied for specific types of ML models such as clustering [38], [39], linear regression [40], logistic regression [41], and Gaussian mixture models [42]. It is challenging to develop a general data selection method [18] that can be used for all ML algorithms.

On the other hand, data replay [43]–[45] is widely used to mitigate catastrophic forgetting. The core idea is to periodically retrain the current model with past data, thereby preserving previous knowledge. Early methods used random sampling for periodic replay, which can preserve memory to some extent. However, this approach can lead to decrease in the accuracy of SML, as such data often acts as noise. Recent work [46], [47] maintained a replay buffer for data augmentation, which enhances data quality by sampling real-time data and selecting more similar data from the buffer for data replay. On the other hand, our proposed framework in this work, FreewayML, selectively employs appropriate optimization mechanisms based on the degree of data distribution shifts, which is more suitable for a wide range of SML models.

3) Constrained learning: When data streams change dynamically, constraint-based methods [48], [49] retain past experience while learning new knowledge by imposing certain constraints on parameter updates. These methods can also mitigate performance fluctuations caused by low-quality data to some extent. EWC [23] introduced a parameter-constraint method that incorporates an additional regularization loss related to the parameters. GEM [50] updated only the parameters for new tasks without interfering with those of old tasks. GEM modified the gradient update direction for new tasks using inequality constraints, aiming for minimizing the loss of new tasks without increasing the loss of old tasks. A-GEM [51] reduced the computational burden associated with GEM, making it more practical for scenarios with a large number of tasks or when computational resource is restricted. Due to limited generalization capability of ML models, such methods often encounter performance bottlenecks. Moreover, because parameter updates are constrained, their ability to capture dynamic changes also diminishes.

III. Study on Data Distribution shifts

Since streaming data is non-IID, streaming ML (SML) models tend to suffer from unstable or low accuracy as data distribution shifts. In this Section, to identify which types of shifts significantly affect the model accuracy, we use a graph-based visualization to analyze the correlation between distribution shifts and accuracy trends. According to the shift severity and shift range, we identify three primary shift patterns associated with accuracy fluctuations and drops.

A. Experimental setups

Existing SML frameworks, such as Flink ML, Spark Streaming, and River, predominantly employ mini-batch SGD for incremental updates of ML models. We therefore use the commonly used *mini-batch StreamingMLP* with the batch size of 1024 and test its accuracy in a single-machine environment.

We conduct our experimental evaluation on three representative real-world datasets: (1) *electricity load*, (2) *stock price trend*, and (3) *solar irradiance*.

We use real-time accuracy (acc) as our evaluation metric [52]. When each batch of data flows in, we first use the existing model to make predictions, and the prediction accuracy for that batch is referred to as the real-time accuracy (acc). In detail, The real-time accuracy (acc) of Batch j can be calculated as Equation 1 shows, where y_{ji} is the actual label of the i-th sample in the batch, and $\hat{y}_{ji}$ is the prediction.

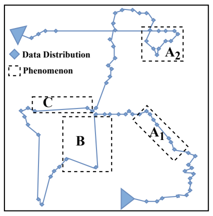
$$acc_j = \frac{\sum_{i=1}^{k} \mathrm{I}(y_{ji} = \hat{y}_{ji})}{k} \tag{1}$$

B. Correlation between accuracy and distribution

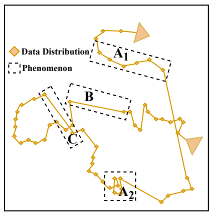
Streaming data is dynamic and non-iid distributed. Existing research has indicated that changes in data distribution can impact the accuracy of streaming learning models. To more intuitively explore the correlation, we constructed a data shift graph with PCA. We reduce each batch from 3 real-world datasets to a point, and Figure 2a, 2b, and 2c presents the results. We connect the two-dimensional points in chronological order, and the line connecting two points represents a data shift. Through these steps, we obtain a *shift graph* that represents the changes in data distribution.

In Figure 2, the greater the linear distance between two points is, the larger the shift in data distribution is. We first observe that streaming data exhibits uncertain patterns of change; at times, the changes are minor, while at other times, they are significant. Consequently, we seek to examine how the accuracy of streaming learning models varies with different magnitudes of changes.

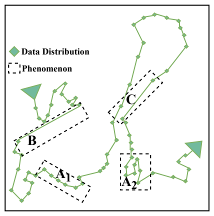
Accordingly, we catch several typical phenomena: *(Phenomenon A)* slight data distribution shifts, *(Phenomenon B)* severe data distribution shifts, and *(Phenomenon C)* severe shifts where the distribution has previously occurred, which is a special case. For *phenomenon A*, there are two subcategories: (1) a directional slight shift, referred to as A_1, and (2) a localized slight shift, referred to as A_2.



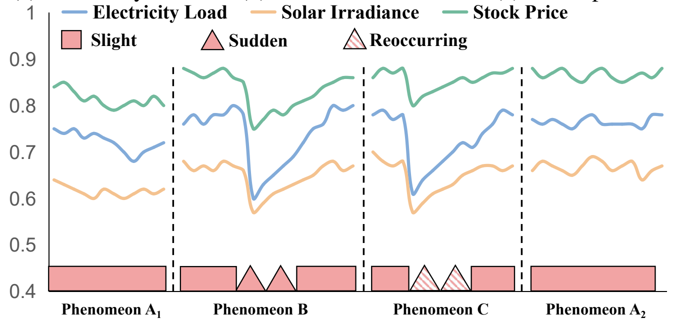
(a) Electricity load.



(b) Solar irradiance.



(c) Stock price.



(d) Accuracy of streaming MLP under shifts.

Fig. 2: Visualization of data distribution shifts and model accuracy under typical shifts.

Experimental results show a strong correlation between the degree of accuracy changes and the magnitude of shifts as Figure 2d shows. Simply put, when data shifts are minor, the decrease in model accuracy is also small; when data shifts are significant, the model accuracy decreases drastically, which is consistent with the assumptions made in existing research.

C. Patterns of data distribution shifts

Pattern Classification Rationale with Examples. Furthermore, we aim to extract useful pattern information from these phenomena to guide more stable streaming learning. We classify shift patterns based on two key factors: *shift severity* and *shift range*. We first identify two primary patterns by shift severity: Pattern A (slight shifts from *Phenomenon A*) and Pattern B (severe shifts from *Phenomenon B* and *C*). Pattern A generally causes accuracy fluctuations, whereas Pattern B often leads to significant accuracy drops. Within each category, we further differentiate patterns by shift range.

1) **Slight Shifts**

- **Pattern A$_1$ (Directional Shift):** Data distribution gradually shifts in a new direction over time. This pattern is common in applications such as traffic flow, where the number or direction of vehicles increases or decreases in real-time. Pattern A$_1$ reflects evolving trends, requiring frequent SML model updates to track directional changes. Therefore, a short-time granularity model is preferred to adapt to these shifts.
- **Pattern A$_2$ (Localized Shift):** The data distribution shifts within a small, stable range, preserving existing patterns rather than evolving into new ones. Examples include slight daily temperature variations, minor fluctuations in energy consumption, or gradual changes in customer engagement metrics during regular business cycles. For Pattern A$_2$, a long-time granularity model is preferred to stabilize the accuracy within this narrow range of variation. Here comes our Insight A:

Insight A: Slight shifts in data distribution affect accuracy, so we consider combining models with both long-time and short-time granularity.

2) **Severe Shifts**

- **Pattern B (Sudden Shift):** The data distribution abruptly transitions to a new distribution, as observed in events like Black Friday, where transaction volumes can surge unexpectedly. Pattern B is challenging for pre-trained model due to the unpredictability and magnitude of the shift. Here comes our Insight B:

Insight B: In the presence of sudden shifts, unsupervised clustering may demonstrate better performance compared to SML methods that require pre-training.

- **Pattern C (Reoccurring Shift):** The data distribution shifts back to a previous pattern. This may occur in scenarios like network security, where different types of attack patterns may alternate over time. For Pattern C, historical models or prior knowledge can be utilized to improve accuracy, allowing the model to anticipate and respond effectively to familiar shifts. Here comes our Insight C:

Insight C: When data distributions are similar, the previously trained model demonstrates high accuracy.

This pattern extraction and classification inspire us to design new and adaptive model update strategies, such as the multi-time (short and long) granularity models, the coherent experience clustering, and historical knowledge reuse.

Quantitative Pattern Definition. To formally define the patterns, we use Equations 2-10 to quantify data distribution shifts. We begin by warming up a Principal Component Analysis (PCA) model to reduce the dimensionality of the streaming data (Equations 2-5). We then calculate the current shift, i.e., the data distribution distance between the current batch and the previous batch (Equations 6-7). Finally, we evaluate the shift severity by comparing the current shift with the previous shifts (Equations 8-10). For instance, if the current shift is statistically significant (an outlier in statistical terms), it is classified as Pattern B. More details about the equations are as follows. We also add Figure 3 to illustrate these equations.

(1) Dimension reduction (Equations 2-5): To accelerate the shift calculation and focus on the most relevant data features, we employ Principal Component Analysis (PCA) for dimension reduction. Initially, we train a PCA model using a set of n initial data points. Subsequently, we form the component matrix P_d and apply this to reduce the dimension of incoming streaming data.

$$\mu = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i, \tag{2}$$

$$\Sigma = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - \mu)(\mathbf{x}_i - \mu)^T, \tag{3}$$

$$\Sigma = VDV^T, \tag{4}$$

$$P_d = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d]. \tag{5}$$

(2) Shift distance calculation (Equations 6-7): To track changes in data distribution after dimension reduction, we employ a simple yet effective approach: calculating the Euclidean distance between consecutive batches. For each incoming batch, we use the average value, $\bar{y}_t$, as a representation of its data distribution, which is a commonly used approach for simplicity and computational efficiency [53], [54]. The current shift distance, d_t, is thus computed as the Euclidean distance between the current batch at time t and the previous batch at time $t-1$. In future work, we plan to explore more statistical metrics, such as standard deviation, to improve the representation of data distribution and the accuracy of shift distance calculations.

$$\bar{y}_t = P_d^T(\mu_t - \mu), \tag{6}$$

$$d_t = \|\bar{y}_t - \bar{y}_{t-1}\|. \tag{7}$$

(3) Shift severity evaluation (Equations 8-10): To evaluate the shift severity, we compare the current shift distance to the (statistical distribution of) previous/historical shift distances. We first compute the weighted mean μ_d and standard deviation σ_d of previous shift distances, assigning higher weights w_i to more recent batches. We then classify the current shift as sudden (Pattern B) if its distance magnitude M exceeds a statistical threshold α, typically three standard deviations above the weighted mean (i.e., $\alpha = 1.96$).

$$\mu_d = \frac{\sum_{i=1}^{k} w_i d_{t-i}}{\sum_{i=1}^{k} w_i}, \tag{8}$$

$$\sigma_d = \sqrt{\frac{\sum_{i=1}^{k}(d_{t-i} - \mu_d)^2}{k}}, \tag{9}$$

$$M = \frac{d_t - \mu_d}{\sigma_d}. \tag{10}$$

To further determine if Pattern C can be applied, we calculate the nearest distance between the current batch and previous batches, denoted as d_h, and compare it with current shift distance d_t. If $d_h < d_t$, it indicates that the current shift is moving towards a previously observed data distribution.

We can now define data distribution shift patterns as follows:

- **Pattern A** Slight shift. Condition: $M < \alpha$;
- **Pattern B** Sudden shift. Condition: $M > \alpha$;
- **Pattern C** Reoccurring shift. Condition: $M > \alpha$ and $d_h < d_t$.

The three patterns of data distribution shifts defined above cover the entire lifecycle of data streams, and they allow for a more abstract and simplified definition of streaming trends. In the data pattern phase, FreewayML calculates the shift distance for each batch and compares it with previous shifts, as shown in Equations 2-10. The time complexity for shift distance calculation is $O(nd)$, where n is the number of data points in each batch, and d is the feature dimension. The comparison with the past k batches (Equations 8-10) has time complexity of $O(k)$. Space complexity is minimal, requiring storage of statistical values such as $\bar{y}_t$ and the matrix P_d, totaling $O(d^2)$

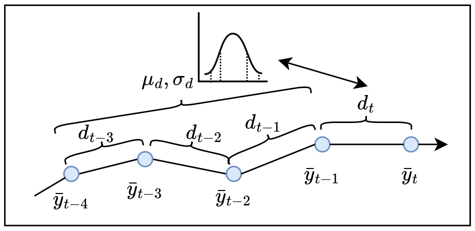


Fig. 3: The measurement of data distribution shift.

IV. The Design of FreewayML

In Section III, we introduced the shift graph and observed that accuracy of SML suffers from fluctuations under three different shift patterns. We have designed specific optimization mechanisms for different patterns, and in this section we propose FreewayML, an adaptive and stable SML framework. We start by an overview of FreewayML, which includes three different optimization mechanisms as Figure 4 shows. Subsequently, we detail the design of each optimization mechanism.

A. Overview of FreewayML

1) Multi-time granularity models: We design a multi-time granularity model to enhance the stability and accuracy of streaming learning when facing slight shifts, including directional shifts (i.e., Pattern A_1) and localized shifts (i.e., Pattern A_2). For A_1, models with short-time granularity can learn new distribution more sensitively and quickly. Conversely, for A_2, models with long-time granularity are less affected by fluctuations and exhibit better stability.

Therefore, we design a mechanism with multi-time granularity models, where each model possesses different update granularity and data selection methods. During update, short-time granularity model utilizes the entire batch and updates at a fixed frequency. For long-time granularity model, we develop a new data structure named *adaptive streaming window* (ASW) for maintaining data with a balance of low overhead and high value. Specifically, we innovatively defined the *disorder* of this adaptive streaming window to facilitate decay maintenance, effectively reducing unnecessary overhead. During inference, FreewayML integrates multiple models based on the distribution distance between models and data, and outputs the results accordingly. We include the detailed design in Section IV-B.

2) Coherent experience clustering: We design a coherent experience clustering (CEC) to temporarily replace deployed SML models when sudden distribution shifts occur and pretrained models are not suitable. Unsupervised clustering can make predictions based on the current data distribution without the need for model warm-up. However, it lacks specific label information to map clusters into labels. Streaming data inherently possesses continuity, not only in terms of time but also in its distribution. Here, we have made the following Hypothesis:

<u>**(Hypothesis)**</u> Data that is continuous over time also exhibits continuity in its distribution.

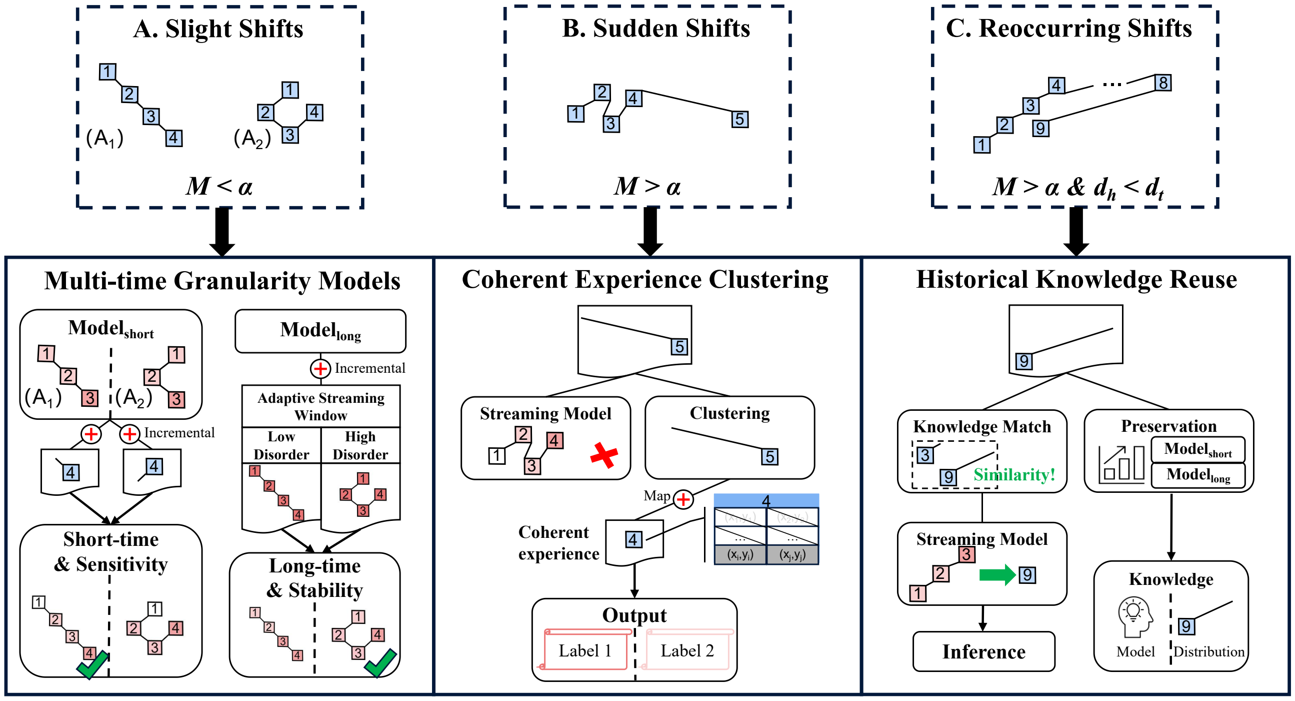


Fig. 4: Overview of FreewayML.

Our innovation leverages adjacent labeled data as *coherent experience*, grounded in the continuity of streaming data distribution. This approach guides the clustering of current batch data and maps unlabeled clusters to labeled categories, enhancing model accuracy. We include the detailed design in Section IV-C.

3) Historical knowledge reuse: We propose historical knowledge reuse that includes knowledge preservation and knowledge match to handle reoccurring shifts. Based on the disorder in the ASW, we selectively preserve more stable knowledge (e.g., long-time granularity mode) to balance knowledge coverage and knowledge quality. Furthermore, when an old data distribution reappears, we introduce a distance measurement to match the available historical knowledge. We include the detailed design in Section IV-D.

B. Details of multi-time granularity models

Multi-time granularity models are proposed to achieve better prediction accuracy under the slight shifts. Each model possesses different update granularity. We aim to establish more stable models with minimal computational overhead, which presents the following technical challenges: (1) How to determine the update granularity for different models? (2) How to ensemble multiple models to enhance overall stability? We propose (1) a model construction method based on ASW and (2) a model ensemble method based on distance to address these two technical challenges.

ASW-based model construction: For short-time granularity model, update granularity tends to be fixed (e.g., one minute or one hour), as developers aim for swift capture of potential changes. To reach a balance between sensitivity and stability for long-time granularity model, we propose a new data structure, *adaptive streaming window* (ASW) for training data management, as illustrated in Figure 5. ASW can adjust the update granularity in an *adaptive* manner based on the real-time distribution shift pattern.

In ASW, long-time granularity model starts an update when the window reaches the maximum number of batches or window size. Before it reaches the preset batches or size, when a new batch flows in, existing batches in the window have to be decayed.

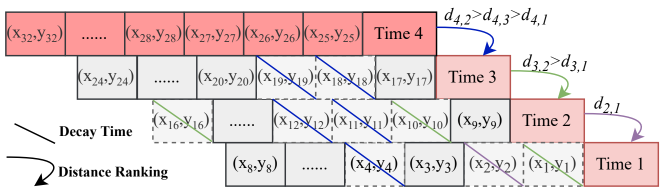

(x_{32},y_{32}) ...... (x_{28},y_{28}) (x_{27},y_{27}) (x_{26},y_{26}) (x_{25},y_{25}) Time 4
$d_{4,2}>d_{4,3}>d_{4,1}$
(x_{24},y_{24}) ...... (x_{20},y_{20}) (x_{19},y_{19}) (x_{18},y_{18}) (x_{17},y_{17}) Time 3
$d_{3,2}>d_{3,1}$
(x_{16},y_{16}) ...... (x_{12},y_{12}) (x_{11},y_{11}) (x_{10},y_{10}) (x_9,y_9) Time 2
$d_{2,1}$
Decay Time
Distance Ranking
(x_8,y_8) ...... (x_4,y_4) (x_3,y_3) (x_2,y_2) (x_1,y_1) Time 1

Fig. 5: Adaptive streaming window based on shift.

The decay takes into account not only the time but also the shift range of the data within the window. Shift range represents the changes in data distribution over a period of time, it imposes different requirements on the model's updates. Therefore, we defined disorder value $order$ using the following Equation 11 to assist in quantitatively calculating the shift range. Here, τ represents the distance ranking at the current moment. Figure 7 is a typical example, when the disorder is low, the model may be in Pattern A_1, whereas when the disorder is high, the model may be in Pattern A_2.

$$\text{order}(\tau) = |\{(i,j) : i < j \text{ and } \tau_i > \tau_j\}|. \tag{11}$$

This reduces unnecessary computational and space overhead. Data management in ASW adheres to the following two principles:

1) When a new batch flows in, its shift distances from existing batches are calculated and ranked; and the smaller the distance is, the less the decay rate is. Hence, by using a sorting decay based on shift severity, the window can align with the current data distribution more closely.
2) If the distance ranking shows a high disorder, the decay rate will be increased, because data may be localized and thus model update is not urgent. On the other hand, if the disorder is low, the data may be experiencing a directional shift, necessitating a faster update to be aligned with the new data distribution.

As illustrated in Figure 5, different colored lines represent decayed data at different time points. At time point 4, the data exhibits the greatest shift from time point 2, while it shows less shift from time points 3 and 1. This results in the most significant decay of data from time point 2. In addition, when focusing on data from time points 1 and 3, current data shifts less from time point 1, which does not indicate a directed drift but potentially indicates an localized one. A higher decay rate in such cases can reduce unnecessary update overhead. The specific calculation process is summarized in Algorithm 1.

Algorithm 1 Adaptive Streaming Window Processing

```
 1: ASW: window with max_batches and max_items
 2: procedure PROCESSING($batch\_now$, $window$)
 3:     if $window$.batches $\geq$ max_batches OR $window$.items $\geq$ max_items then
 4:         Use $window$ for updating
 5:     end if
 6:     $shifts \leftarrow []$
 7:     for each $batch_i$ in $window$ do
 8:         $shift \leftarrow$ SHIFTSEVERITY($batch\_now, batch_i$)
 9:         $shifts$.append($shift$)
10:     end for
11:     $sorted\_shifts \leftarrow$ SORT($shifts$)
12:     $order \leftarrow$ DISORDER($sorted\_shifts$)
13:     for each $batch_i$ in $window$ do
14:         $decay\_rate \leftarrow f(rank_i, order)$
15:         DECAY($batch\_i$, $decay\_rate$)
16:     end for
17: end procedure
```

Distance-based adaptive ensemble: For model ensemble, we further extend *shift distance* to *model shift distance* D as Equations 12 and 13 below show. D represents the shift between the existing model and the real-time data. Intuitively, the smaller the value of D is, the higher the match between the model and the current data is.

$$D_{Short} = \|\bar{y}_n - \bar{y}_{n-1}\|, \tag{12}$$

$$D_{Long} = \|\bar{y}_n - \bar{y}_{ASW}\|. \tag{13}$$

We can calculate the distance from the last batch for short-time granularity models, while long-time granularity model requires computing the distance from data in their ASW. Since D represents the degree of match between the model and the current data, we incorporate it as a weighting factor in model ensemble with Gaussian kernel, as Equation 14 shows:

$$y = \frac{K(D_{Short}, \sigma) \cdot y_{Short} + K(D_{Long}, \sigma) \cdot y_{Long}}{K(D_{Short}, \sigma) + K(D_{Long}, \sigma)}. \tag{14}$$

By default, FreewayML employs two models for constructing multi-time granularity models. However, user can customize the number of models without extra implementation effort.

FreewayML includes both short and long-time granularity models, with the help of ASW. The time complexity of the ASW is $O(w \log w)$, which includes calculating distances, sorting the w batches of n data points with d dimensions within the window, and decaying the data. The space complexity of ASW is $O(wnd)$ for storing w batches. As model updates for different time granularities can be parallelized in FreewayML, the time complexity of model update is similar to traditional SML.

C. Details of coherent experience clustering

Unsupervised clustering may offer unexpected advantages as they focus more on the features of the current data itself, without requirement for prior training. Therefore, when a sudden shift occurs, we replace the original model with K-means clustering and output the corresponding clusters.

However, a challenge arises here: streaming clustering methods can only output unlabeled data as distinct clusters, and there is no direct correspondence between clusters and labels. Therefore, such clusters cannot be directly presented as results to user.

However, streaming data inherently possesses continuity, not only in terms of time but also in its distribution. Specifically, it is impossible to perfectly segment different data distributions with each batch. Therefore, when we detect a sudden shift, this distribution often has already occurred at the end of the previous batch. Based on this hypothesis, we can use a small subset of labeled data that is closest to the current batch to map clusters to labels, assuming that it shares a similar distribution.

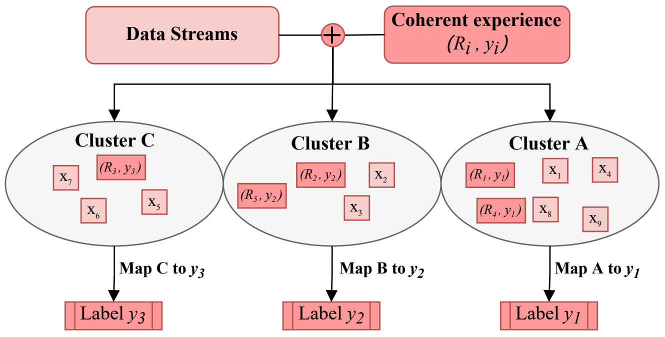


Fig. 6: Example of coherent experience clustering.

For this sake, we introduce *coherent experience clustering (CEC)* to map clusters to labels. Specifically, it includes the most recent m labeled data points as additional guidance. We cluster the current batch of data together with these m data points and map the clusters to their most probable labels based on the clustering results of the labeled data. As Figure 6 shows, although the current data has been divided into three unlabeled clusters A, B, and C, when we introduce some labeled data (R_i, y_i) the label information within each cluster can help map the clusters to the labels.

CEC groups m data points with d dimensions from the previous batch and n data points from the current batch into c clusters (where c is the number of labels), with time complexity of $O((m+n)cd)$. The corresponding space complexity is $O(md)$, as only m additional data points are stored.

D. Details of historical knowledge reuse

If historical knowledge can be preserved and effectively reused when a previous distribution reoccurs, it can then significantly enhance the accuracy of the model. However, this still faces two challenges: (1) Which knowledge and when should it be preserved? (2) Which historical knowledge should we use when a distribution reoccurs?

1) Knowledge preservation: For knowledge preservation, there are two main issues: (a) what to save and (b) when to save. The solution to (a) is relatively straightforward. We can construct a *mapping relationship* between data distributions and model parameters. As a result, the knowledge i to be preserved is stored in the form of (d_i, k_i), where d_i represents the distribution and k_i represents the reusable model information.

The solution to (b) is not so obvious and is worth discussing. Clearly, it is impractical to preserve knowledge from all moments, which would incur considerable space overhead and computational cost during knowledge selection. Moreover, not all models from each moment are effective enough and therefore are worth preserving. Therefore, we aim to select and preserve relatively stable models, (e.g., long-time granularity models). However, the long-time granularity models sometimes miss distribution information of certain batches within the ASW.

The disorder of a window is an effective indicator for distinguishing different situations, because it reflects the shift ranges in ASW. When disorder is high, as Figure 7 shows, it indicates a high frequency of localized data. Under these conditions, long-time granularity models tend to be more stable, whereas short-time granularity models are not necessarily stable. Conversely, when disorder is low, it suggests that the data stream may be undergoing an orderly directional shift which means that the stabilized data distribution after the shift is more valuable. In this scenario, while long-time granularity models may lose some information, short-time granularity models with information lacked are therefore worth preserving as well.

Therefore, we set a threshold β for the disorder. At the end of ASW i, if the disorder exceeds β, then the distribution of ASW and the parameters of the long-time granularity model are preserved in the form of (d_i, k_i). On the other hand, if the disorder is less than β, then the parameters of the short-time granularity model and its data distribution at the current moment are also saved.

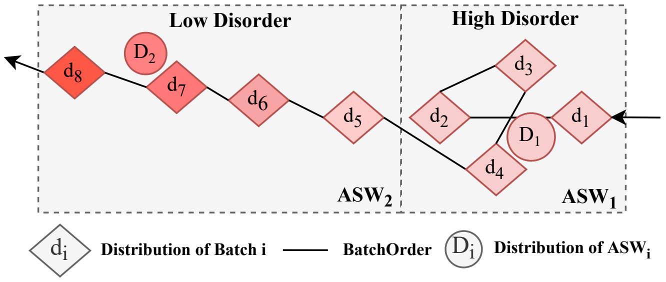


Fig. 7: Streams with different disorders in ASW.

Knowledge match: We have stored knowledge pairs in the form of (d_i, k_i), where d_i represents the distribution corresponding to the knowledge k_i. Therefore, when severe shift occurs, we calculate the shift between the current inference data and the historical distributions, and we then select the d_i with the smallest distance. This shift is then compared with the shift distance between the current data and the last batch. If the d_i is smaller, the knowledge k_i is reused.

With h historical stable models, the storage complexity is $O(hs)$, where s represents the number of model parameters. Regarding time complexity, identifying the nearest data distribution is performed during the pattern detection phase, which has a time complexity of $O(k)$. Retrieving the corresponding historical model from the key-value store is efficient, with a time complexity of $O(1)$ due to hashing.

V. Implementation

We implement FreewayML based on PyTorch. Our implementation provides a simple interface for users to invoke FreewayML, with the following template:

```
SML = Learner (Model = model, ModelNum = 2,
    MiniBatch = 1024, KdgBuffer = 20, ExpBuffer
    = 10, α = 1.96)
```

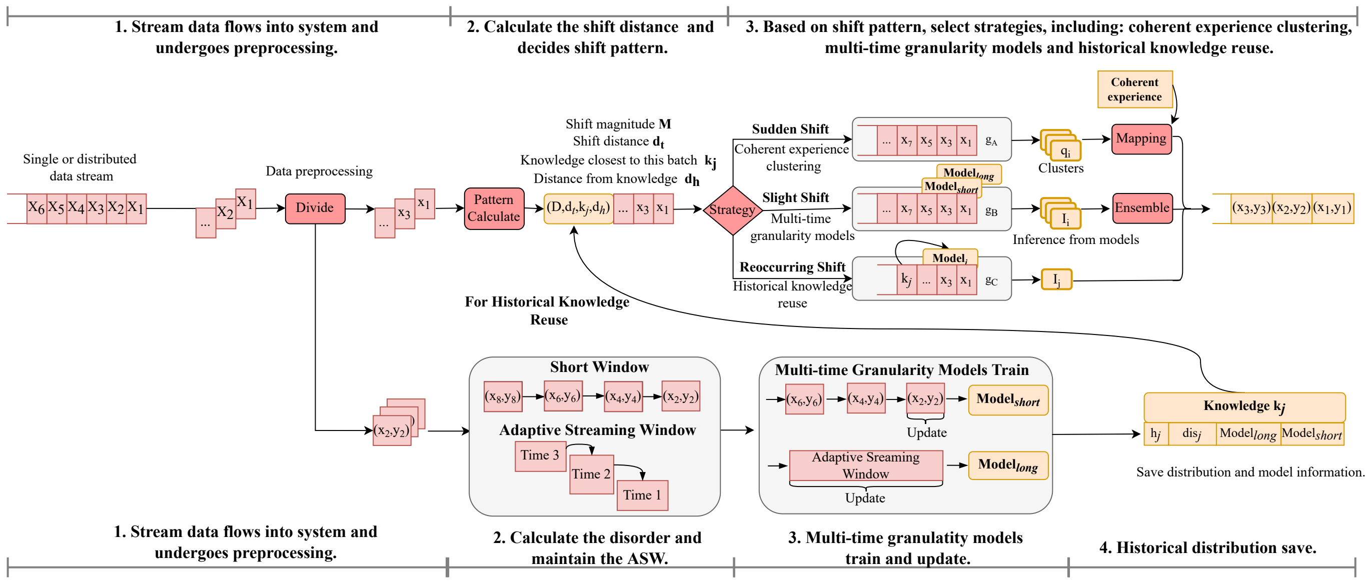
This interface is similar to that offered by existing Streaming ML frameworks such as River and Flink ML. FreewayML outputs various metrics after each epoch, such as training loss, accuracy, and execution time.

A. Pipeline of FreewayML

Figure 8 illustrates the pipeline of FreewayML. FreewayML makes no assumptions about data streams. It supports the input of both inference and training data, with different modules processing them accordingly. When FreewayML is deployed and connected to real-time data, the data comes in a single stream and is divided into *inference* and *training* streams based on whether it is labeled.

When training stream flows in, multiple models are incrementally updated according to their granularity. The model is saved periodically based on the degree of shift disorder.

When inference stream flows in, the shift distance is calculated to determine the shift pattern. For slight shifts, the more stable multi-time granularity models are used for prediction;



1. Stream data flows into system and undergoes preprocessing.
2. Calculate the shift distance and decides shift pattern.
3. Based on shift pattern, select strategies, including: coherent experience clustering, multi-time granularity models and historical knowledge reuse.
Single or distributed data stream
Data preprocessing
Divide
Pattern Calculate
Shift magnitude M
Shift distance d_t
Knowledge closest to this batch k_j
Distance from knowledge d_h
Strategy
Sudden Shift
Coherent experience clustering
Slight Shift
Multi-time granularity models
Reoccurring Shift
Historical knowledge reuse
Coherent experience
Clusters
Mapping
Ensemble
Inference from models
For Historical Knowledge Reuse
Short Window
Adaptive Streaming Window
Time 3
Time 2
Time 1
Multi-time Granularity Models Train
Update
Adaptive Streaming Window
Knowledge k_j
Save distribution and model information.
1. Stream data flows into system and undergoes preprocessing.
2. Calculate the disorder and maintain the ASW.
3. Multi-time granularity models train and update.
4. Historical distribution save.

Fig. 8: Pipeline of FreewayML.

for sudden shifts, coherent experience clustering is utilized; and for reoccurring patterns, historical knowledge reuse is employed.

It is important to note that, for the training stream, the updating of multi-time granularity model is always enabled, which uses different granularities for incremental training. However, for the inference stream, only **one** strategy is selected and executed based on the shift pattern of the current batch. We develop a *strategy selector* based on the pattern classifier introduced in Section III.

1) Multi-time granularity models: The management of window and synchronization of models can significantly impact the performance and accuracy of multi-time granularity models. We have developed an *Adaptive Window* operator leveraging PyTorch's DataLoader. This operator sorts incoming batches by distance and computes disorder after each batch, enabling automated decay. Once the window reaches a specified number of batches or items, it triggers an update of the long-time granularity model. FreewayML employs a multi-process architecture with independently managed processes and asynchronous updates, ensuring non-blocking inference execution. By utilizing inter-process locks, it maintains update atomicity, enhancing concurrency and reducing latency.

2) Coherent experience clustering: Additional data and labels need to be stored for coherent experience clustering. In FreewayML, we offer the *ExpBuffer* interface, which allows users to customize the amount they wish to retain. FreewayML systematically saves the data in memory w.r.t. batch order. In addition, the *expiration time* is set to remove outdated experiences.

3) Historical knowledge reuse: As historical knowledge accumulates over time, FreewayML offers the *KdgBuffer* interface to customize the maximum number of knowledge maintained. When knowledge reaches the size, FreewayML saves the first half to local storage and clears it from memory.

B. Optimization

Pre-computing window mechanism. To reduce the computational latency of updating, we have designed a pre-computing mechanism within the window. Our method involves incrementally calculating the gradient for subsets of data and accumulating them, rather than computing the gradient for the entire batch in the end. When using pre-computing, we divide the original window data into n subsets, where the gradient calculated from the i_{th} subsets is denoted as $\nabla\theta_i$. For model updating, it is only necessary to calculate the gradient for the last subset and aggregate it with the previously computed gradients. This method efficiently computes and stores corresponding gradients while waiting for data, effectively reducing processing latency.

Rate-aware Adjuster. Inference and training can compete for resources, particularly during high-speed data streams. To mitigate this, we have optimized the inference and training frequencies as follows.

Inference frequency controller adaptively adjusts the inference frequency based on real-time data flow rate and window pressure. When the data flow rate is low and the window pressure is minimal, we increase the inference frequency to quickly consume the pending data. Update frequency adjustment increases the decay rate of the training window (adaptive streaming window) when the data flow rate exceeds a certain threshold. This reduces the frequency of model updates to lower resource competition.

VI. Experiemental Evaluation

We conduct extensive experiments to evaluate FreewayML. We aim to answer the following questions:

(RQ1) How good is the accuracy and stability of FreewayML when compared with existing methods?

(RQ2) How does FreewayML perform with its three mechanisms based on the data distribution shift patterns?

TABLE I: Accuracy and stability of different streaming learning frameworks on six datasets.

Model	Frameworks	Hyperplane		SEA		Airlines		Covertype		NSL-KDD		Electricity	
		G_{acc}	SI	G_{acc}	SI	G_{acc}	SI	G_{acc}	SI	G_{acc}	SI	G_{acc}	SI
StreamingLR	Flink ML	81.51%	0.932	82.54%	0.926	62.17%	0.861	58.78%	0.843	80.72%	0.908	77.54%	0.894
	Spark MLlib	81.74%	0.928	82.68%	0.929	62.30%	0.858	59.12%	0.850	80.98%	0.906	77.27%	0.892
	Alink	83.25%	0.935	82.02%	0.924	63.08%	0.862	58.61%	0.838	82.04%	0.912	78.12%	0.898
	FreewayML	**88.69%**	**0.948**	**84.62%**	**0.935**	**65.12%**	**0.914**	**62.46%**	**0.892**	**86.52%**	**0.926**	**81.38%**	**0.915**
StreamingMLP	River	81.14%	0.935	82.06%	0.928	62.28%	0.858	62.57%	0.882	81.43%	0.910	83.11%	0.906
	Camel	84.82%	0.938	82.14%	0.929	64.37%	0.872	63.85%	0.894	82.21%	0.913	83.57%	0.906
	A-GEM	84.60%	0.934	81.98%	0.917	64.54%	0.890	62.14%	0.891	82.38%	0.920	83.33%	0.902
	FreewayML	**88.47%**	**0.946**	**84.80%**	**0.930**	**66.70%**	**0.918**	**65.78%**	**0.905**	**87.10%**	**0.937**	**85.91%**	**0.921**

(RQ3) What is the processing performance of FreewayML compared with existing methods?

A. Datasets and baselines

We conducted our experiments on six datasets, which include two synthetic datasets, **Hyperplane** [55] and **SEA** [56], as well as four real-world datasets **Airlines** [57], **CoverType** [58], **NSL-KDD** [59], and **Electricity** [60]. These datasets cover typical streaming learning scenarios in practical applications such as network security, power scheduling, and flight forecasting.

To validate the effectiveness of FreewayML, we conduct comprehensive comparisons against a range of competitive baselines, including **Flink ML** [15], **Spark MLlib** [14], **Alink** [17], **River** [16], **Camel** [18] and **A-GEM** [51].

Given the ML models supported by existing work, we select two models that are commonly support and are representatives of linear and nonlinear ML models with batch size of 1024: (1) Streaming Logistic Regression and (2) Streaming MLP. For FreewayML, we set $\alpha = 1.96$ for the sake of classifying different shift patterns.

B. Metrics

Prequential evaluation is widely adopted when evaluating streaming learning. In this paper, we employ two types of accuracy: real-time accuracy (acc) introduced in Section III and global average accuracy (G_acc) as Equation 15 shows. Both of them are widely used in streaming learning to evaluate the model's real-time response/fluctuations and overall performance. To measure accuracy fluctuations, we employ a Stability Index (SI) derived from statistical theory. This Stability Index (SI) is defined as the ratio of the standard deviation of batch accuracies σ_{acc}, to the mean of the batch accuracies μ_{acc}. For interpretability, we apply an exponential scaling to normalize this value to a [0, 1] range, as shown in Equation 16.

$$G\_acc = \frac{1}{m}\sum_{j=1}^{m}\left(\frac{\sum_{i=1}^{k}\mathrm{I}(y_{ji} = \hat{y}_{ji})}{k}\right) \tag{15}$$

$$SI = \exp\left(-\frac{\sigma_{acc}}{\mu_{acc}}\right) \tag{16}$$

In our study, we also place significant emphasis on the performance of FreewayML, in particular its throughput and latency metrics when processing various data streams. At the same time, we calculated the additional space overhead resulting from the historical knowledge of FreewayML.

C. Accuracy and stability comparison

To answer RQ1, we evaluate the two models across the six datasets. Table I summarizes the results, where the best-performing methods have been highlighted.

We observe that FreewayML outperforms existing methods in terms of both accuracy and stability across all six datasets. When compared against **Camel**, one of the most robust baselines on top of the datasets evaluated, FreewayML achieves an average relative improvement of **3.9%** in accuracy. In StreamingLR, our approach registers a relative improvement of **5.8%** over the widely utilized **Flink ML**.

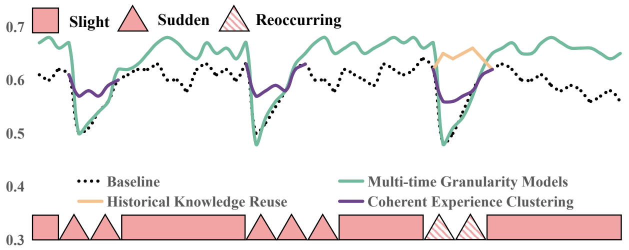
In experiments conducted on top of the **NSL-KDD** dataset, while existing methods have already demonstrated high accuracy, FreewayML still manages to achieve notable improvements. Detailed analysis of the dataset reveals that the data distribution shifts with the types of current network attacks, often leading to significant class imbalances. Our method significantly enhances the classification performance of the minority classes, which, as a result, improves the overall accuracy.

D. Effectiveness of mechanisms

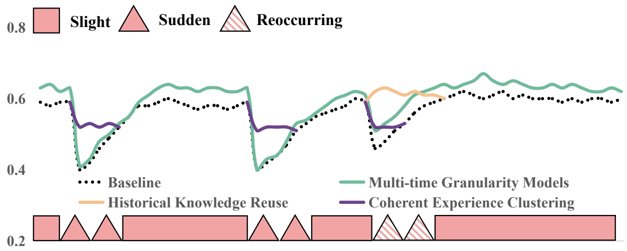
To answer RQ2, we investigate the effectiveness of our proposed mechanisms across the three data distribution shift patterns. To evaluate the improvements brought by FreewayML, we list the improvement in accuracy of each shift pattern compared with original Streaming MLP, as shown in Table II. This underscores the effectiveness of our mechanisms. For most of the time, data streams meet slight shifts, and multi-time granularity models enhance stability. The coherent experience clustering and historical knowledge reuse mechanisms improve accuracy in specific data scenarios and mitigate potential severe issues.

TABLE II: Accuracy improvement compared with original Streaming MLP under 3 patterns.

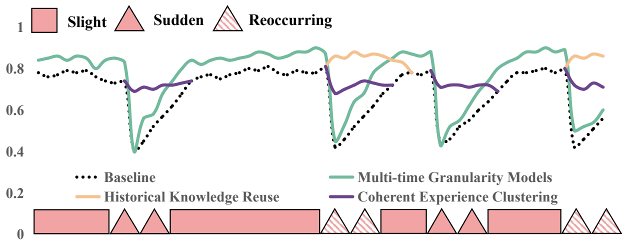
Dataset	Slight Shifts	Sudden Shifts	Reoccurring Shifts
Hyperplane	5.7%	34.1%	59.3%
SEA	2.1%	10.5%	57.8%
Airlines	3.8%	8.6%	17.7%
Covertype	2.6%	11.7%	16.8%
NSL-KDD	4.4%	28.5%	41.0%
Electricity	2.3%	5.1%	18.9%



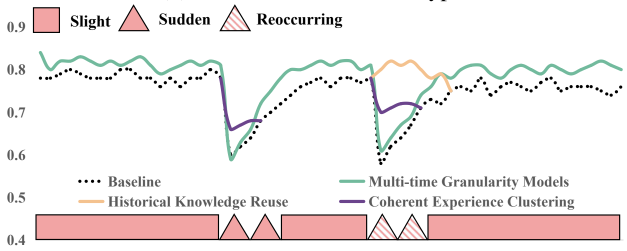
(a) Mechanisms on Airlines.



(b) Mechanisms on CoverType.



(c) Mechanisms on NSL-KDD.



(d) Mechanisms on Electricity.

Fig. 9: Comparative accuracy(%) analysis of FreewayML mechanisms under distribution shift patterns.

To further visually analyze the effectiveness of the FreewayML, we selected four real datasets, each characterized by distinct shift patterns, to compare the accuracy of FreewayML with Streaming MLP without the optimization mechanism. As shown in Figure 9, the dashed line represents the baseline version of the Streaming MLP, and the three solid lines with different colors represent the corresponding optimization mechanisms. Among them, multi-time granularity models enhance overall stability and accuracy for the majority of instances, while coherent experience clustering and historical knowledge reuse also exhibit significant improvements of accuracy in the presence of sudden shifts and reoccurring shifts. This demonstrates that, based on the three effective mechanisms, our strategy selector can accurately identify shift patterns and apply the appropriate mechanisms.

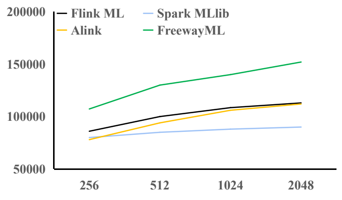
When compared with existing methods, as shown in Figure 11, FreewayML shows improvements across all three patterns, in particular under sudden shifts and reoccurring shifts. This demonstrates that our three mechanisms indeed perform well across the distribution shift patterns.

E. Performance comparison

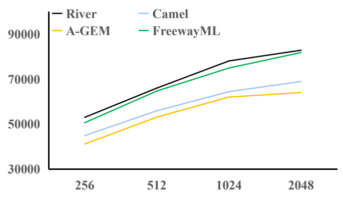
To answer RQ3, we need to compare the performance of FreewayML with existing methods. From a performance perspective, our primary concerns with streaming learning frameworks are throughput and latency metrics. In addition, FreewayML's historical knowledge requires storing information, and we also focus on the extra space overhead.

To evaluate throughput and latency, we employ the same model to first infer and then train on the simulated dataset, **Hyperplane**. The most critical performance-relevant parameter identified is the batch size. Consequently, we investigate how performance varies with different batch sizes.

Regarding throughput, we vary the batch size from 256 to 2,048 Figure 10 shows. For LR models, it is evident that FreewayML substantially outperforms other frameworks. Notably,



(a) Throughput of StreamingLR



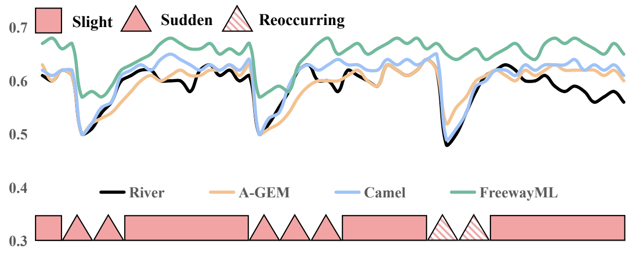
(b) Throughput of StreamingMLP

Fig. 10: Throughput of FreewayML compared to existing methods.

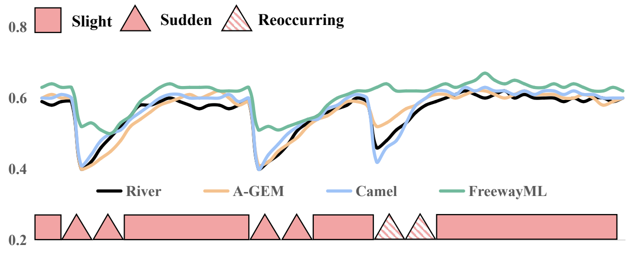
the throughput of FreewayML exceeds that of **Flink ML** by more than **1.4**× with batch size 2,048, highlighting its superior efficiency in processing data streams. For Streaming MLP, on the other hand, FreewayML demonstrates a clear performance advantage over **Camel**, which is based on data selection, and **A-GEM**, which is based on constrained learning. This superior performance is attributed to our design of adaptive streaming window and disorder-based knowledge preservation, which take performance overhead into account. Finally, the throughput of FreewayML is comparable to the basic streaming learning framework **River**, though the prediction accuracy of FreewayML is significantly higher, as Table I shows.

Regarding latency, we divide the computing tasks into inference and update phases, which are characterized by distinct computational complexities and communication costs. We further categorize the models into Streaming Logistic Regression and Streaming MLP, as detailed in Table III. The batch size is set to ranging from 512 to 4,096, as smaller batch size typically yields lower inference latency, which is challenging to measure accurately due to fluctuations in network communication speed.

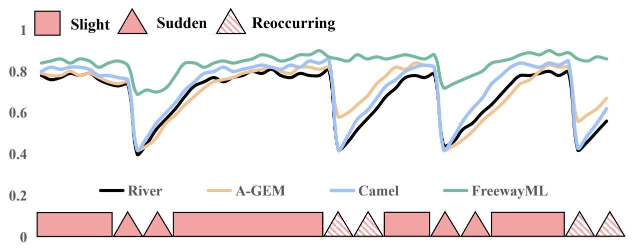
We observe that FreewayML achieves the lowest inference latency of only **390** μs when using linear model with a batch size of 512. For nonlinear model, FreewayML provides quicker feedback compared to **Camel** and **A-GEM**, with prediction latency comparable to that of **River**. Notably, with



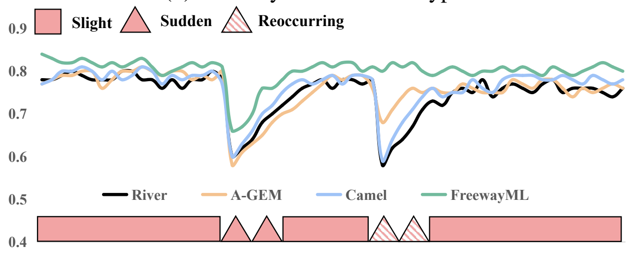
(a) FreewayML on Airlines.



(b) FreewayML on CoverType.



(c) FreewayML on NSL-KDD.



(d) FreewayML on Electricty.

Fig. 11: Accuracy(%) of FreewayML compared to existing methods.

a batch size of 4,096, the prediction latency of FreewayML is nearly equal to that of **River**, which indicates that FreewayML retains efficiency even under increased computational demands.

TABLE III: Latency(μs) of FreewayML compared to existing methods with various batch sizes.

Size of batch		512	1024	2048	4096
LR_{update}	Flink ML	4500	8350	15960	31340
	Spark MLlib	5320	10230	20060	39610
	Alink	4800	8540	16100	31500
	FreewayML	3460	6460	11880	22530
MLP_{update}	River	5110	8690	16500	31780
	Camel	6040	10580	19600	37640
	A-GEM	6360	10910	21190	42270
	FreewayML	5210	9010	16540	31850
LR_{infer}	Flink ML	510	920	1780	3470
	Spark MLlib	590	1140	2230	4410
	Alink	530	950	1800	3510
	FreewayML	390	720	1320	2500
MLP_{infer}	River	2560	4320	8170	15780
	Camel	3030	5290	9890	18900
	A-GEM	3180	5440	10550	20640
	FreewayML	2610	4510	8290	15920

In addition to this, FreewayML saves valuable historical knowledge to improve accuracy when facing reoccurring shifts. However, this also incurs additional space overhead, which is closely related to the number of number of saved knowledge k. Therefore, we tested the space occupied by historical knowledge when k ranges from 1 to 100, as shown in Table IV below. Overall, since the structure of streaming learning models is relatively simple, even when k reaches 100, the space occupied remains less than 2 MB. Given the rapid data and the accuracy improvements provided, we consider this overhead to be acceptable.

F. Limitations of FreewayML

Although experimental results indicate that coherent experience clustering effectively improves model accuracy under sudden shifts, in some scenarios shown in Figure 9, the increase in accuracy does not maintain the same high level as before the shift. This is due to the increased challenges faced by unsupervised clustering algorithms following a large shift, as the data distribution itself becomes more uncertain. As future work, we plan to further optimize FreewayML by improving data quality and using data segmentation to enhance accuracy under sudden shifts.

TABLE IV: Space Overhead of Historical Knowledge for Different k

k	Space Overhead(KB)	
	LR	MLP
1	1.3	8.4
5	7.4	49.6
10	15.4	110.9
40	70.2	490.6
100	196.3	1360.8

VII. Conclusion

In this paper, we studied the problem of robust streaming learning in the presence of dynamic data change. We conducted an empirical study that highlights three data distribution shift patterns, namely, (1) slight shift, (2) sudden shift, and (3) reoccurring shift, which occur throughout the lifecycle of a data stream. We further developed FreewayML, an adaptive and stable streaming learning framework based on optimization mechanisms designed for the distribution shift patterns, namely, (1) multi-time granularity models, (2) coherent experience clustering, and (3) historical knowledge reuse. Our experimental evaluation results on top of synthetic and real-world datasets demonstrate the effectiveness of FreewayML. Compared to existing SML systems, FreewayML achieves much higher prediction accuracy while maintaining similar or better throughput and latency. As future work, we plan to optimize the scalability of FreewayML and enhance its performance in distributed computing environments.

References

[1] Y. Zhang, F. Zhang, H. Li, S. Zhang, and X. Du, "Compressstreamdb: Fine-grained adaptive stream processing without decompression," in *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 2023, pp. 408–422.

[2] S. S. Bhowmick, E. C. Dragut, and W. Meng, "Boosting entity mention detection for targetted twitter streams with global contextual embeddings," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2022, pp. 1085–1097.

[3] X. Zeng and S. Zhang, "Parallelizing stream compression for iot applications on asymmetric multicores," in *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 2023, pp. 950–964.

[4] Y. Zhang, Y. Liu, H. Xiong, Y. Liu, F. Yu, W. He, Y. Xu, L. Cui, and C. Miao, "Cross-domain disentangled learning for e-commerce live streaming recommendation," in *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 2023, pp. 2955–2968.

[5] E. Mousavinejad, F. Yang, Q.-L. Han, and L. Vlacic, "A novel cyber attack detection method in networked control systems," *IEEE transactions on cybernetics*, vol. 48, no. 11, pp. 3254–3264, 2018.

[6] W. W. Ng, J. Zhang, C. S. Lai, W. Pedrycz, L. L. Lai, and X. Wang, "Cost-sensitive weighting and imbalance-reversed bagging for streaming imbalanced and concept drifting in electricity pricing classification," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 3, pp. 1588–1597, 2018.

[7] Y. Luopan, R. Han, Q. Zhang, C. H. Liu, G. Wang, and L. Y. Chen, "Fedknow: Federated continual learning with signature task knowledge integration at edge," in *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 2023, pp. 341–354.

[8] B. Kim, K. Koo, J. Kim, and B. Moon, "Disc: Density-based incremental clustering by striding over streaming data," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 828–839.

[9] L. Xu, X. Ye, K. Kang, T. Guo, W. Dou, W. Wang, and J. Wei, "Diststream: an order-aware distributed framework for online-offline stream clustering algorithms," in *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2020, pp. 842–852.

[10] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE transactions on knowledge and data engineering*, vol. 31, no. 12, pp. 2346–2363, 2018.

[11] K. S. Adewole, T. T. Salau-Ibrahim, A. L. Imoize, I. D. Oladipo, M. AbdulRaheem, J. B. Awotunde, A. O. Balogun, R. M. Isiaka, and T. O. Aro, "Empirical analysis of data streaming and batch learning models for network intrusion detection," *Electronics*, vol. 11, no. 19, p. 3109, 2022.

[12] T.-H. Chang, M. Hong, H.-T. Wai, X. Zhang, and S. Lu, "Distributed learning in the nonconvex world: From batch data to streaming and beyond," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 26–38, 2020.

[13] H. M. Gomes, J. Read, A. Bifet, J. P. Barddal, and J. Gama, "Machine learning for streaming data: state of the art, challenges, and opportunities," *ACM SIGKDD Explorations Newsletter*, vol. 21, no. 2, pp. 6–22, 2019.

[14] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen *et al.*, "Mllib: Machine learning in apache spark," *Journal of Machine Learning Research*, vol. 17, no. 34, pp. 1–7, 2016.

[15] P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi, and K. Tzoumas, "Apache flink: Stream and batch processing in a single engine," *The Bulletin of the Technical Committee on Data Engineering*, vol. 38, no. 4, 2015.

[16] J. Montiel, M. Halford, S. M. Mastelini, G. Bolmier, R. Sourty, R. Vaysse, A. Zouitine, H. M. Gomes, J. Read, T. Abdessalem *et al.*, "River: machine learning for streaming data in python," *Journal of Machine Learning Research*, vol. 22, no. 110, pp. 1–8, 2021.

[17] A. Group, "Alink," 2024, gitHub repository. [Online]. Available: https://github.com/alibaba/Alink

[18] Y. Li, Y. Shen, and L. Chen, "Camel: Managing data for efficient stream learning," in *Proceedings of the 2022 International Conference on Management of Data*, 2022, pp. 1271–1285.

[19] A. Alsaedi, N. Sohrabi, R. Mahmud, and Z. Tari, "Radar: Reactive concept drift management with robust variational inference for evolving iot data streams," in *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 2023, pp. 1995–2007.

[20] F. Bayram, B. S. Ahmed, and A. Kassler, "From concept drift to model degradation: An overview on performance-aware drift detectors," *Knowledge-Based Systems*, vol. 245, p. 108632, 2022.

[21] G. I. Webb, R. Hyde, H. Cao, H. L. Nguyen, and F. Petitjean, "Characterizing concept drift," *Data Mining and Knowledge Discovery*, vol. 30, no. 4, pp. 964–994, 2016.

[22] M. Riemer, I. Cases, R. Ajemian, M. Liu, I. Rish, Y. Tu, and G. Tesauro, "Learning to learn without forgetting by maximizing transfer and minimizing interference," in *International Conference on Learning Representations*, 2018.

[23] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[24] Z. Lin, Y. Wang, and H. Lin, "Continual contrastive learning for image classification," in *2022 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2022, pp. 1–6.

[25] M. U. Togbe, Y. Chabchoub, A. Boly, M. Barry, R. Chiky, and M. Bahri, "Anomalies detection using isolation in concept-drifting data streams," *Computers*, vol. 10, no. 1, p. 13, 2021.

[26] D. Brzezinski and J. Stefanowski, "Reacting to different types of concept drift: The accuracy updated ensemble algorithm," *IEEE transactions on neural networks and learning systems*, vol. 25, no. 1, pp. 81–94, 2013.

[27] S. C. Hoi, D. Sahoo, J. Lu, and P. Zhao, "Online learning: A comprehensive survey," *Neurocomputing*, vol. 459, pp. 249–289, 2021.

[28] N. C. Oza and S. Russell, "Experimental comparisons of online and batch versions of bagging and boosting," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, 2001, pp. 359–364.

[29] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, and Y. Fu, "Large scale incremental learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 374–382.

[30] Y. Luo, L. Yin, W. Bai, and K. Mao, "An appraisal of incremental learning methods," *Entropy*, vol. 22, no. 11, p. 1190, 2020.

[31] J. Gasteiger, C. Qian, and S. Günnemann, "Influence-based mini-batching for graph neural networks," in *Learning on Graphs Conference*. PMLR, 2022, pp. 9–1.

[32] K. Dahiya, N. Gupta, D. Saini, A. Soni, Y. Wang, K. Dave, J. Jiao, G. K, P. Dey, A. Singh *et al.*, "Ngame: Negative mining-aware mini-batching for extreme classification," in *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, 2023, pp. 258–266.

[33] W. Ren, T. Zhao, W. Qin, and K. Liu, "T-sas: Toward shift-aware dynamic adaptation for streaming data," in *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 2023, pp. 4244–4248.

[34] G. Rypeść, S. Cygert, V. Khan, T. Trzcinski, B. M. Zieliński, and B. Twardowski, "Divide and not forget: Ensemble of selectively trained experts in continual learning," in *The Twelfth International Conference on Learning Representations*.

[35] J. Gui, Y. Song, Z. Wang, C. He, and Q. Huang, "Sk-gradient: Efficient communication for distributed machine learning with data sketch," in *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 2023, pp. 2372–2385.

[36] Z. Fan, J. Guo, X. Li, T. Yang, Y. Zhao, Y. Wu, B. Cui, Y. Xu, S. Uhlig, and G. Zhang, "Finding simplex items in data streams," in *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 2023, pp. 1953–1966.

[37] Z. Liu, C. Kong, K. Yang, T. Yang, R. Miao, Q. Chen, Y. Zhao, Y. Tu, and B. Cui, "Hypercalm sketch: One-pass mining periodic batches in data streams," in *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 2023, pp. 14–26.

[38] D. Feldman and M. Langberg, "A unified framework for approximating and clustering data," in *Proceedings of the forty-third annual ACM symposium on Theory of computing*, 2011, pp. 569–578.

[39] D. Feldman, M. Schmidt, and C. Sohler, "Turning big data into tiny data: Constant-size coresets for k-means, pca, and projective clustering," *SIAM Journal on Computing*, vol. 49, no. 3, pp. 601–657, 2020.

[40] C. Boutsidis, P. Drineas, and M. Magdon-Ismail, "Near-optimal coresets for least-squares regression," *IEEE transactions on information theory*, vol. 59, no. 10, pp. 6880–6892, 2013.

[41] A. Munteanu, C. Schwiegelshohn, C. Sohler, and D. Woodruff, "On coresets for logistic regression," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[42] M. Lucic, M. Faulkner, A. Krause, and D. Feldman, "Training gaussian mixture models at scale via coresets," *Journal of Machine Learning Research*, vol. 18, no. 160, pp. 1–25, 2018.

[43] Y. Balaji, M. Farajtabar, D. Yin, A. Mott, and A. Li, "The effectiveness of memory replay in large scale continual learning," *arXiv preprint arXiv:2010.02418*, 2020.

[44] K. Holstein, E. Harpstead, R. Gulotta, and J. Forlizzi, "Replay enactments: Exploring possible futures through historical data," in *Proceedings of the 2020 ACM Designing Interactive Systems Conference*, 2020, pp. 1607–1618.

[45] S. Channappayya, B. R. Tamma *et al.*, "Augmented memory replay-based continual learning approaches for network intrusion detection," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[46] X. Jin, A. Sadhu, J. Du, and X. Ren, "Gradient-based editing of memory examples for online task-free continual learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 29 193–29 205, 2021.

[47] D. Brignac, N. Lobo, and A. Mahalanobis, "Improving replay sample selection and storage for less forgetting in continual learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 3540–3549.

[48] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "icarl: Incremental classifier and representation learning," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 2001–2010.

[49] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," *arXiv preprint arXiv:1606.04671*, 2016.

[50] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning," *Advances in neural information processing systems*, vol. 30, 2017.

[51] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny, "Efficient lifelong learning with a-gem," *arXiv preprint arXiv:1812.00420*, 2018.

[52] J. Zhu, Q. Jia, G. Cai, Q. Dai, J. Li, Z. Dong, R. Tang, and R. Zhang, "Final: Factorized interaction layer for ctr prediction," in *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2023, pp. 2006–2010.

[53] A. Bifet and R. Gavalda, "Learning from time-changing data with adaptive windowing," in *Proceedings of the 2007 SIAM international conference on data mining*. SIAM, 2007, pp. 443–448.

[54] F. Bayram, B. S. Ahmed, and A. Kassler, "From concept drift to model degradation: An overview on performance-aware drift detectors," *Knowledge-Based Systems*, vol. 245, p. 108632, 2022.

[55] "Hyperplane dataset," 2021. [Online]. Available: https://riverml.xyz/0.14.0/api/datasets/synth/Hyperplane/

[56] "Sea dataset," 2021. [Online]. Available: https://riverml.xyz/0.14.0/api/datasets/synth/SEA/

[57] "Airlines dataset," 2023. [Online]. Available: https://www.kaggle.com/datasets/saadharoon27/airlines-dataset

[58] "Covertype dataset," 1998. [Online]. Available: https://archive.ics.uci.edu/dataset/31/covertype

[59] "Nsl-kdd dataset," 2009. [Online]. Available: https://www.unb.ca/cic/datasets/nsl.html

[60] "Elec2 dataset," 2021. [Online]. Available: https://riverml.xyz/dev/api/datasets/Elec2/

Appendix

A. Details of baseline

To validate the effectiveness of FreewayML, we conduct comprehensive comparisons against a range of competitive baselines, including **Flink ML** [15], **Spark MLlib** [14], **Alink** [17], **River** [16], **Camel** [18] and **A-GEM** [51]

- **Flink ML** [15], which is built on top of Apache Flink, enables efficient data stream processing and enhances accuracy through its watermark mechanism.
- **Spark MLlib** [14] processes streaming data through mini-batch windows and incrementally updates ML models based on average gradients.
- **Alink** [17] integrates FOBOS and RDA with logistic regression to enhance model stability when dealing with real-time data streams.
- **River** [16] is an efficient streaming learning framework which provides drift detectors and model integrators to combat potential changes in data distribution.
- **Camel** [18] provides effective data selection to reduce model training cost and increase data quality.
- **A-GEM** [51] constrains gradient update direction to avoid interference with previous data buffered.

B. More experiments on CNN models

To validate the effectiveness of our proposed method in the field of multimedia data, we additionally introduced CNN models for relevant experiments. To ensure the consistency of the experiment, we still first used the six benchmark datasets used in the above experiment. Furthermore, we additionally tested the performance of CNN on the image dataset.

For our 6 benchmark datasets, we constructed a three layer CNN architecture, including a convolutional layer with 32 convolutional kernels of size 3, a max pooling layer with a window size of 2, and a fully connected layer for classification.

For the image datasets, we follow the methodologies from existing work [34], utilizing the ImageNet-Subset and Flowers datasets and transforming them into image streams. These datasets involve tasks such as identifying types of animals and flowers. We constructed a five-layer CNN architecture, comprising two convolutional layers with 64 3x3 kernels, two 2x2 max pooling layers, and a fully connected layer for classification. Since image exhibits different feature representations from value-based data, we introduced a VGG-16 to extract features from the original images before performing coherent experience clustering.

First, we continue to use a fixed window size to evaluate the accuracy and stability of eight datasets through prequential evaluation, as shown in Table V below. It can be observed that compared to StreamingCNN without our mechanisms/strategies, FreewayML demonstrates superior accuracy and stability across all eight datasets. For benchmark datasets, the global average accuracy G_acc improves by an average of approximately 5.1 points. For image datasets, the results show that the global average accuracy improves by an average of approximately 4.3 points.

Furthermore, to observe the accuracy improvement of different strategies in FreewayML, we visualize the real-time accuracy (acc) of FreewayML on four real datasets and two image datasets. As shown in Figure 12, the dashed line represents the baseline, i.e., StreamingCNN. The three solid lines with different colors represent the corresponding FreewayML's strategies. We can see that the three strategies are effective to improve the baseline accuracy.

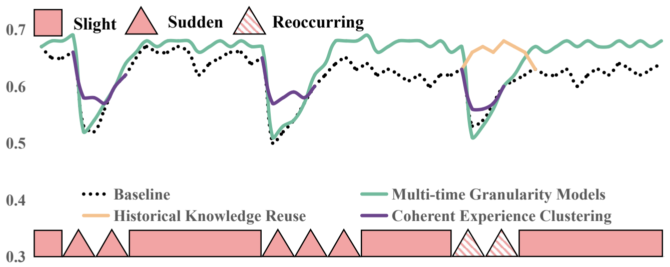
TABLE V: Accuracy comparison of StreamingCNN and FreewayML on different datasets.

Dataset	StreamingCNN		FreewayML	
	G_acc	SI	G_acc	SI
Hyperplane	80.23%	0.928	87.18%	0.941
SEA	78.69%	0.915	83.24%	0.930
Airlines	59.25%	0.853	65.13%	0.916
Covertype	65.20%	0.895	67.24%	0.924
NSL-KDD	76.42%	0.882	84.10%	0.925
Electricity	80.14%	0.902	83.71%	0.917
Animals	86.60%	0.924	90.08%	0.935
Flowers	81.55%	0.911	86.57%	0.932

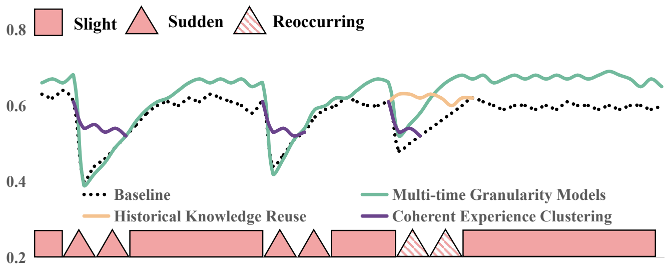
To evaluate the overhead of CNN under FreewayML's mechanisms. We used the same network structure and different batch sizes to perform inference and training on the Hyperplane dataset as Table VI shows. The overhead introduced by the optimization mechanism is less than 5%.

TABLE VI: Latency (μs) of CNN updates and inferences compared to FreewayML with various batch sizes.

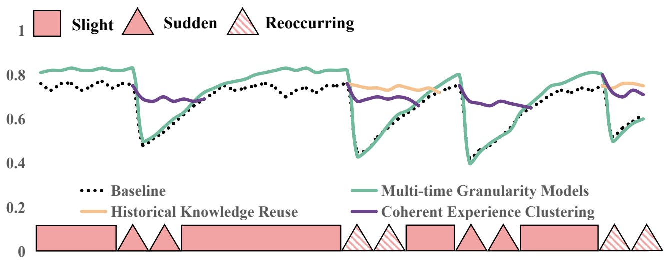
Size of batch		512	1024	2048	4096
$\boldsymbol{CNN}_{infer}$	StreamingCNN	3070	5370	10040	19690
	FreewayML	3220	5610	10380	20150
$\boldsymbol{CNN}_{update}$	StreamingCNN	6560	11550	21710	42970
	FreewayML	6890	11970	22250	43670



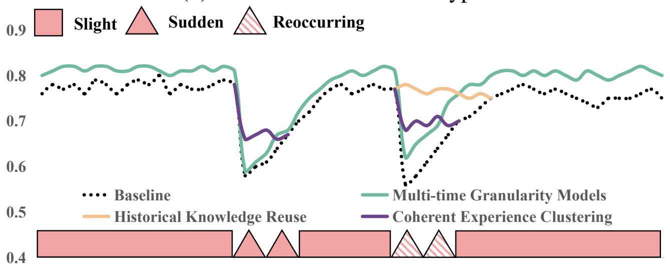
(a) Mechanisms on Airlines.



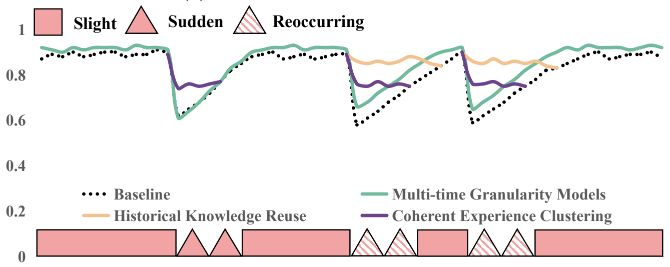
(b) Mechanisms on CoverType.



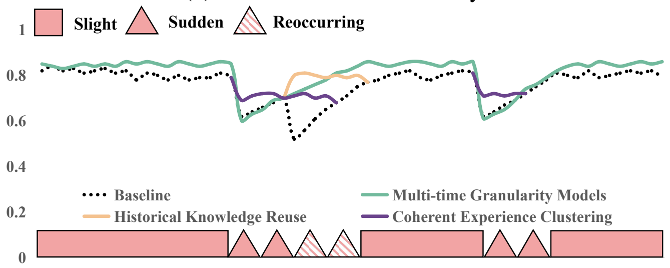
(c) Mechanisms on NSL-KDD.



(d) Mechanisms on Electricity.



(e) Mechanisms on Animals.



(f) Mechanisms on Flowers.

Fig. 12: Comparative accuracy (%) analysis of CNN with FreewayML mechanisms under distribution shift patterns.