UDC 004.896
DOI: 10.15587/1729-4061.2025.343635

OPTIMIZED ADAPTIVE MACHINE LEARNING FOR DYNAMIC DATA STREAMS

Aivar Sakhipov
PhD, Assistant Professor*
Aruzhan Mektepbayeva
Corresponding author
Master's Student*
E-mail: aruzhan.mektepbaevaa@gmail.com
Amangul Talgat
Senior Lector
Department of Information Technology
K. Kulazhanov Kazakh University of Technology and Business
Kaiym Mukhamedkhanov str., 37A, Astana,
Republic of Kazakhstan, 010000
Maxot Rakhmetov
PhD, Associate Professor
Department of Computer Science**
Ainagul Adiyeva
PhD, Associate Professor
Department of Mathematics and Methodical
Teaching of Mathematics**
Altynbek Seitenov
MSc, Senior Lecturer*
Nurzhan Ualiyev
Candidate of Physical and Mathematical Sciences, Senior Lecturer
Department of Information Technology and Artificial Intelligence
I. Zhansugurov Zhetysu University
I. Zhansugurov str., 187a, Taldy-Kurgan,
Republic of Kazakhstan, 040009
Shynar Yelezhanova
Candidate of Physico-Mathematical Sciences
Department of Software Engineering**
*School of Software Engineering
Astana IT University
Mangilik El ave., 55/11, Astana, Republic of Kazakhstan, 010000
**Kh. Dosmukhamedov Atyrau University
Studenchesky ave., 1, Atyrau, Republic of Kazakhstan, 060011

The object of the study is the adaptive machine learning systems that are able to process large amounts of rapidly changing streaming data in real time. The problem of maintaining prediction accuracy and computational efficiency in the presence of concept drift is treated. Concept drift refers to the overweighting of static models when stationary models are tried, and the nature of the underlying distributions changes. The adaptive architecture includes revision divergence-oriented concept drift detection, incremental model updating via hyper-dimensional statistical clustering of segments. Results from experiments using simulated and real-world datasets demonstrate that the adaptive architecture maintains predictive accuracy above 0.83 across abrupt, gradual, recurrent, and continuous drift scenarios. Compared with non-adaptive models, adaptation latency is reduced by approximately 2.6×, while unnecessary retraining operations are decreased by up to 40%. These results are possible due to the fact that proposed framework is able to retrain solutions if, and only if, distributional changes are determined to be statistically significant and meaningful to the model. This leads to the avoidance of processors being given redundant computations and providing a steady-state model during non-drift conditions. A principal contribution is that feature engineering is accomplished in a drift-aware manner, thresholding is made adaptive to the distributions indicated, and update mechanisms are employed which efficiently utilize resources in a unified high-performance streaming pipeline. The architecture performs well under abrupt, gradual, recurrent, and continuous drift and effective for real-time applications which include smart-city analytics, cyber security monitoring, roadways system works, and IoT for industrial systems

Keywords: adaptation model, computing performance, drift detection, streaming processing, rapid updating

Received 02.10.2025
Received in revised form 05.12.2025
Accepted 16.12.2025
Published 29.12.2025

How to Cite: *Sakhipov, A., Mektepbayeva, A., Talgat, A., Rakhmetov, M., Adiyeva, A., Seitenov, A., Ualiyev, N., Yelezhanova, S. (2025). Optimized adaptive machine learning for dynamic data streams. Eastern-European Journal of Enterprise Technologies, 6 (3 (138)), 15–25. https://doi.org/10.15587/1729-4061.2025.343635*

1. Introductions

In the modern era of computer science and engineering, the explosive growth of digital information has transformed the foundations of intelligent computation.

The International Data Corporation estimates that the global datasphere will approximately exceed 175 zettabytes by 2026, with almost one third of the total produced in real time from interconnected sensors, IoT devices and cloud infrastructures [1]. The resulting data streams exhibit massive scale, large dimensionality and are time varying in character, thus posing complex challenges for real-time analysis and decision making. The ability to develop adaptive, high-quality ML systems is relevant because current and future systems

that process large amounts of data are going to need to be consistently accurate and efficient regardless of the frequent changes in the characteristics of the input data. Efficient extraction of valid knowledge from this data is dependent not only on high predictive accuracy but also on computational scalability, adaptability and robustness in continuously varying conditions [2].

Traditional machine learning methods have, however, developed to deal with large volumes of well-structured data that are fixed in time and environment leading to a situation where they are no longer suitable for the analysis of these varying environments. Models such as support vector machines, Bolsus method of estimation, random forests, and classical deep networks tend to become unstable when the statistical properties of data change over time a condition referred to as concept drift [3]. In rapidly changing conditions as those exhibited by financial markets, the cyber-world, and transport systems, this normally leads to delayed and inaccurate responses.

The need for frequent full retraining significantly increases the computational load on deployments, reaching 40–60% of the processing time during stream processing. This is due, in part, to the need to continuously adapt models in response to changes in the data distribution [4]. The field of high-performance machine learning has therefore become a crucial research direction at the intersection of algorithm design, big data analytics, and high-performance computing. Modern architectures making use of GPU acceleration, distributed frameworks and adaptive optimization mechanisms can allow learning and inference in real time on massive data volumes. Online learning and incremental learning and dynamic learning approaches make it possible to maintain model accuracy and functionality without retraining extensively, thus reducing latency of solutions in addition to scaling.

However, the extent to which speed adaptability and interpretability of these methods can be balanced is still a major problem for both researchers and end-users. The importance of the problem is wide in application. In transport and smart city systems, adaptive learning models predict traffic congestion, customer demand for transport, and vehicle routing in changing and varying conditions. In the cyber-world they intercept evolving attack conditions in the analysis of large-scale network traffic. In health, energy, and industrial IoT they predictively monitor diagnose and control automatically through processing massive data flows. In all these cases the facility for continuous learning of high-velocity non-homogeneous data streams is now recognized as a mandatory prerequisite for intelligent self-adapting systems.

At the same time, of course, there has been an ever-increasing growth in research on scalable and adaptive learning systems in the scientific world. Over the past five years the number of papers published in Scopus and in IEEE Xplore indexed under the parameters of adaptive machine learning and real time big data analysis has grown by more than a factor of three [5].

Despite this advance, however, the majority of solutions provided appear now still to be limited by static learning rates, high energy usage and also the lack of autonomous feedback mechanisms which will control and adapt model parameters in the light of subsequent continuously varying data flow. It advances the understanding of how intelligent systems can maintain accuracy, speed of use, efficiency and transparency under continuously varying, dynamic and uncertain environments. Therefore, research on the development of high-performance machine learning algorithms for processing big data and dynamically changing systems is highly relevant, representing a significant step toward the next generation of intelligent computational technologies.

2. Literature review and problem statement

Recent advances in artificial intelligence and in data-intensive computing, have resulted in a significant advance when developing adaptive, scalable learning architectures. However, as the complexity of environments for real-time data processing, it remains one of the outstanding problems for the community of scientists to achieve the optimum equilibrium between computational efficiency, adaptivity and interpretability.

The paper [6] presents the results of extensive survey of the use of techniques in deep learning and of the way data streams at high rates have been processed. It is shown that the techniques which are considered are convolutional neural networks (CNNs), recurrent neural networks (RNNs) and autoencoders. The study indicates clearly that the increased recognition performance attained by utilizing hierarchical feature extraction on the high volume of features. It is more conventional on large sized figures including CIC-IDS2018 and UCI streaming benchmarks. But there were unresolved issues related to the problem with dynamic overfitting of multiple machine learning models. Overfitting leads to deterioration in performance and accuracy, especially when working with conceptual frames. A way to overcome these difficulties can be approaches such as regularization, dropout, early stopping, and ensemble methods. This approach was used in [6], however it can still lead to some loss of performance in complex models, or important parameter values may be lost. All this suggests that it is advisable to conduct the study while leaving the number of examples in the training set unchanged.

The paper [7] presents the results of implementing hybrid CNN-LSTM architecture in relation to IoT sensor networks and telemetry industrial inputs data. It was shown that combining convolutional and recurrent layers enables simultaneous capture of temporal dependencies. But there were unresolved issues related to the model required frequent full retraining, increasing computational cost by over 40% due to the absence of drift detection or adaptive optimization. All this suggests that it is advisable to conduct the study on entropy learning modulation as a possible success way, but will not implement it experimentally.

The paper [8] presents the results of analysis distributed deep learning infrastructures for optimized for GPU-based training using Apache Spark and TensorFlow Distributed. It is shown that the approximate 2.3-fold increase in processing throughput, with an important acceleration in convergence times for ResNet and GRU models. But there were unresolved issues related to synchronization delay and uneven data partitioning remained important bottlenecks. All this suggests that it is advisable to conduct the study on an adaptive scheduling and load-balancing strategies presented a necessity for real scalability in dynamic big data environments.

The paper [9] proposed an online Reinforcement Learning algorithm including Q-learning and neural networks. It was used for predictive maintenance and streaming anomaly detection. But there were unresolved issues related to the absence of explicit drift detection with the model produced overfitting to temporally varying changes. All this suggests that it is advisable to conduct the study on continuous reward-oriented updates meant that adaptability in environments.

The paper [10] presents the results of entropy environmental control systems using learning-rate during training and optimization to check the speed the predictions. A timely improvement in convergence times of approximately 15%. But there were unresolved issues related to the absence of results, because there were not tested against non-stationary multivariate datasets. All this suggests that it is advisable to conduct the study on interpretation and testing against non-stationary multivariate datasets.

The paper [11] presents the results of evaluated both CNN-GRU and CNN-BiLSTM hybrid models to provide spatio and temporal classification in the cyber-physical and transportation systems. It is shown that the hybridization produces increases in generalizability against a background of volatile inputs. But there were unresolved issues related to the absence of incremental retraining pipelines and explainable AI facilities. So, it means that the model output was less than emerging practices. All this suggests that it is advisable to conduct the study on drift and aware retraining together with interpretability practices, for example Grad-CAM and SHAP are good for practical use of the models in real-time environments.

The paper [12] presents the results of the implementation of created architectures evaluated by transformer and attention models in streaming data bases. It is shown that the approach allowed the long-term dependency modelling to be improved. It also increased outputs over the LSTM model approaches in energy consumption and data from network telemetry data bases. However, it was noted that transformer architecture suffered from great computational overhead, such as memory requirements. It increased the proportionally to $O(n^2)$ and made them less than sufficient for real-time deployment. All this suggests that it is advisable to conduct the study on the areas of scalability and inference latency remain outstanding issues.

The paper [13] presents the results of examined explainable machine learning in big data analytics contexts. It incorporated both the SHAP and LIME frameworks into gradient-boosting predictive models (XGBoost, LightGBM). Improvements in interpretability were observed in the models. But there were unresolved issues related to computational layer meant total inference timings, which were qualitatively degraded by the equivalent of 30%. All this suggests that it is advisable to conduct the study on the necessity and the importance of lightweight explainability methods within adaptive opportunity architectures.

The paper [14] presents the results of using an adaptive gradient optimizer such as the Adam, AdaBelief and RMSProp approaches on dynamically evolving datasets. It was seen that whilst adaptive learning rates accelerated learning speeds. But there were unresolved issues related to random optimizers, because they had no sense of contextual awareness in any data. All this suggests that it is advisable to conduct the study on teach optimizing models themselves through data drift.

The paper [15] presents the results of conducted analysis on semi and integrative overview of online learning and incremental learning methods. Shown, Hoeffding Trees, Online SVM, and Deep Stream models. But there were unresolved issues related to incremental updates. They reduced the cost of retraining, the class of methods generally tended to compromise global consistencies. They also were remiss in that they frequently required manual hyper empirical parameter tuning. All this suggests that it is advisable to conduct the study on the consistency of learnt models and explanations of learning, which was achieved by high-performance models trained in unexplored situations.

Two areas of research dealing with the simultaneously operative open questions have also been addressed. The studies reveal that despite the observed and named overwhelming successes in deep and distributed learning research, current research work is characterized by the existence of at least three fundamental factors:

1) an insufficiency of adaptability towards dynamically changing forms of data and concept-drifts;

2) the lack of opportunity for the scalability of computations on distributive and real-time contexts;

3) totally insufficient functionality as a result of the lack of explanation-centric methods in adaptive pipelines.

The analysis of recent studies [6–15] has shown that even though there are many advances made in adaptive and scalable Machine Learning, there remains a list of unresolved interrelated issues. The first issue see is that many existing approaches to the problem of Concept Drift lack robustness due to the fact that they either use static learning strategies or need repeated full retraining of the model which results in degraded model performance and increased instability during operation within non-stationary environments. The second issue is that while larger deep learning and transformer-based models have significantly improved the ability of those models to predict accurately, they also come with a high level of computational overhead that will limit their usability within real-time and high-throughput environments. The last issue that was found is that the majority of adaptive and online learning solutions are missing a principled approach for maximizing drift detection and tightly integrated pre-processing pipelines which causes them to adapt at a late point in time, use resources inefficiently, and deliver low or inconsistent model quality.

Analysis of trends in research suggests that many factors have contributed to an identified need for continued research into the following areas. The need for a single integrated framework that supports both stable performance with concept drift, systems with real-time/high-volume data flows, and systematic updates of the models based upon detected drifts of the input variables. Therefore, further analysis on the creation of an integrated/combined, real-time, and high-performance machine learning system that meets both accuracy and stability needs to support dynamic data stream scenarios is warranted.

3. The aim and objectives of the study

The aim of the study is to construct and optimize effective machine learning algorithms for the processing of large-scale, high-speed and dynamic data streams under real-time constraints. This will make it possible to creation of adaptive computational technology that is able to guarantee the operation of stable and accurate models in non-stationary environments with changing distributions associated with concept drift and changing sources of input data.

To achieve this aim, the following objectives were accomplished:

– to design and validate the unified adaptive learning architecture integrating ingestion, preprocessing, drift detection and incremental model updating for dynamic data streams;

– to develop and evaluate preprocessing, sliding-window and adaptive learning mechanisms capable of ensuring stable performance under heterogeneous, imbalanced and drifting conditions;

– to assess the effectiveness of the proposed framework under controlled non-stationary environments through accuracy, drift-detection quality, F1-score and adaptation latency.

4. Materials and methods

4. 1. The object and hypothesis of the study

The object of the study is the adaptive machine learning systems that are able to process large amounts of rapidly changing streaming data in real time.

The primary hypothesis of this research is that by using drift detection and correction along with incremental training, then can create an integrated system with superior stability, reduced latency and the minimum need to completely re-train the model under non-stationary conditions.

This study assumes that:

– there is a measurable drift in the distribution of incoming data streams over time;

– it is possible to detect drift in incoming data streams using statistically derived divergence metrics without having to wait for complete access to future data;

– an incremental model can approximate the same level of performance as a fully-trained model when the extent of drift is moderate;

– computational constraints for real-time processing are caused more by the need for timely results rather than by the limitations of available storage.

The following simplifications were made when conducting this study: instead of using real-world non-stationarity, synthetic types of drift created to simulate the effects; experiments have concentrated only on representative examples of the adaptive models; rather than calculating total costs of energy used for training a new model, only adaptation latency and retraining frequency were taken together to evaluate cost; and parallel computing and distributing resource scheduling are not explicitly addressed in the model.

4. 2. General workflow of research

The research is inherently multi-stage including: continuous data ingestion; data preprocessing; adaptive learning; drift monitoring; and incremental updates to models. Fig. 1 presents a diagram of the general workflow, showing the functioning of the system in a dynamic big-data environment.

All numerical features were normalized. This was executed by subtracting the first feature valid minimum, then dividing the results by the range. In summary, each of the numerical variables was then normalized to a start of [0,1], and this normalization was consistent across the datasets.

As seen in Fig. 1, heterogeneous sources of data are ingested, normalized and mapped to structured feature representations. These features are fed into the adaptive machine learning model, which drifting detection mechanism continuously controls for any change in the distribution of data to update models accordingly. This flow diagram is used for tracking data throughout the lifecycle of a data warehouse's transformation to a production system.

4. 3. Software tools and computational environment

The research was conducted in an industrial-grade environment for model training and data processing. All preprocessing tasks and classical machine learning models were developed in Python 3.10 using NumPy, Pandas and Scikit-learn. A moving-window estimator was utilized which determined the window average over recent observations, and thus contained particular values in a discrete rolling window of fixed size.

Neural-streaming models such as LSTM and GRU were implemented using TensorFlow 2.x or PyTorch 2.x. Streaming data ingestion or pseudo-real-time ingestion of data was performed using Apache Kafka, and distributed micro-batch processing was performed using Apache Spark Structured Streaming. Experimentation tracking and reproducibility was performed using MLflow, and a Docker container was used to isolate and deploy the software compute environment.

The hardware configuration consisted of an Intel Xeon 12-core CPU, NVIDIA RTX A5000 GPU, 118 GB of RAM, and hosting Ubuntu 22.04 Operating System. A Spark cluster of six executors facilitated distributed processing. Both the software and hardware configurations utilized realistic high-loads over a prolonged duration both for realistic performance, and thus, to reflect typical implementation conditions and typical a typical operational of a real-time big-data systems.

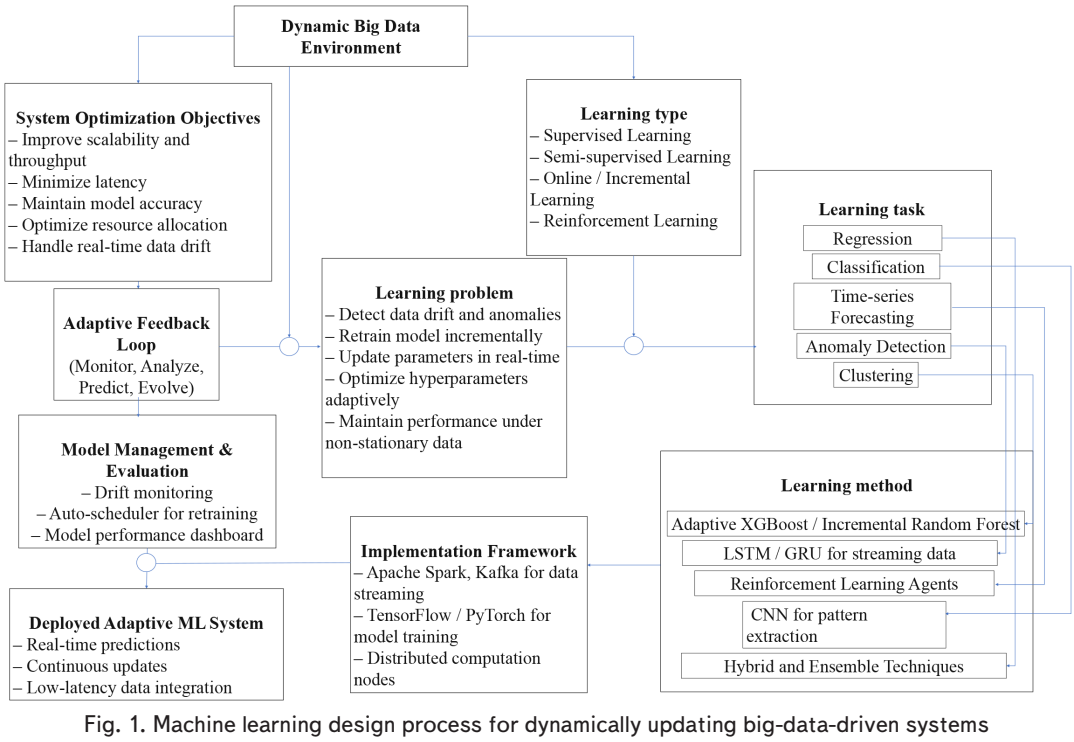

Fig. 1. Machine learning design process for dynamically updating big-data-driven systems

4. 4. A view of the study from an algorithmic perspective

The adaptive part of the system consists of three main mechanisms. The first is incremental learning, where the model itself updates based on small batches of new data. This allows the model to adapt without a full retrain.

The second is drift detection where distributions of incoming data create a divergence score. This score captures how different the current window is from the reference distribution, and the larger the score the greater the change in the pattern of the data.

The third mechanism is dynamic learning-rate change, which increases when the model is uncertain and decreases when the model is confident, allowing the model to be able to respond during drift, and slowly and steadily during normal times. In the case of the regression tasks, the optimization was based on mean squared error, which is the average of squared differences between predictions and true values.

Fig. 2 outlines how the mechanisms work together. The system has a continuous data ingestion mechanism that is constantly calculating drift scores, checking thresholds, and updating the model when needed. This is done to support adaptive processing.

All the steps of the method are straightforward and come right after one another. The system receives the data, partitions it into windows, predicts drift scores, checks the thresholds, fire the partial updates, and stores the results.

4. 5. Datasets and data preprocessing

The robustness of the system was evaluated using a variety of datasets. These datasets incorporate a range of concept drift: abrupt, gradual, continuous, recurrent and cases with imbalanced distribution. This provides a means to evaluate model behavior under drift concept.

The model uncertainty was assessed using prediction entropy. Entropy is defined as the negative summation of class probabilities weighted by their logarithm. When entropy is high, the model is uncertain. In these higher entropy situations, learning rate is increased, which helps with faster adaptation. This principle is based on a prior experiment [11]. The datasets consist of synthetic benchmarks, "real" data streams, and datasets from the THU Concept Drift benchmark suite. Synthetic datasets are used to assess the behavior over controlled drift scenarios. Real-world datasets provide insights into the model's behavior under natural drift. Finally, the imbalanced datasets are used to assess sensitivity to rare events. To implement the proposed framework, different types of datasets were summarized, and the list is demonstrated in Table 1.

All datasets were preprocessed with a uniform preprocessing pipeline. This guaranteed that all models were trained with data in similar formats according to the flowchart in Table 1. The pipeline actions were identical and trivial.

The first step was to normalize numerical values. The second step was to encode the categorical values. Subsequently, the entire dataset was segmented into sliding windows, in such a way as to preserve the temporal semantics in the dataset. Drift-related features were also enhanced to bring attention the notable changes in the data. Lastly, data were streamed through Kafka using controlled micro-batches. These steps established stable and comparable conditions for all experiment replications.

4. 6. Setup for experimentation and conditions for evaluation

Experiments were organized in order to determine if the components that adapted followed reliably when tested with streaming conditions, etc. based on feasibility, along with stability, and computational efficiency, to sample how the components were evaluated.

Drift was determined using the Kullback-Leibler divergence, and was determined as the summed product of each component of the new distribution multiplied by the natural logarithm of the ratio of the normalized distribution to the reference distribution, which allowed the reader to see how strong the incoming data changed.

Streaming intensity varied from a rate of 2,000 events to 20,000 events within a second relative to the new distribution, where window size rates changed from 5 seconds to 60 seconds. Each experiment was repeated at minimum, 10 times to verify consistent, and valid results.

```
Input:
    {X_t}       - streaming data batches from multiple heterogeneous sources
    M = {M1,…,Mn} - initial set of base models (e.g., XGBoost, LSTM, GRU)
    D(·)        - drift detector (e.g., KL-divergence, PSI, Page-Hinkley)
    η_t         - adaptive learning-rate controller
    R           - resource monitor (CPU/GPU utilisation, latency thresholds)

Output:
    M*          - adapted model ensemble for dynamic big data streams

Initialise distributed environment E (e.g., Spark cluster, GPUs)
Draw initial reference window W_ref ← sample(X_0)
Train initial ensemble M ← { train(M_i, W_ref)  for each M_i ∈ M }
while stream({X_t}) is active do
      x_t ← parallel_ingest(X_t, E)
      f_t ← feature_extraction(x_t)
      ŷ_t ← aggregate_predictions(M, f_t)          ▷ ensemble output
      s_t ← D(f_t, W_ref)                          ▷ drift score
      if s_t > τ_drift  and  R.is_available() then
            η_t ← η_t · (1 + α · entropy(ŷ_t))     ▷ entropy-aware LR
            ΔM ← partial_update(M, f_t, y_t, η_t)  ▷ incremental training
            M  ← merge(M, ΔM)                      ▷ update ensemble
            update(W_ref, f_t)                     ▷ refresh reference window
       end if
       log_metrics({accuracy, F1, latency, throughput, energy_usage})
       auto_scheduler ← check(trigger_rules, R)
       if auto_scheduler = True then
            redistribute_load(E, priority = "GPU") ▷ maintain low latency
       end if
 end while
 return M*
```

Fig. 2. Drift-aware incremental learning process

Table 1

List of all datasets used in the study to evaluate concept drift

Dataset	Name	Drift type	Description
Synthetic	SEA concepts	Drift abrupt	This dataset consists of three numerical features. Although these features are defined at an initial time, drift occurs when the decisions are modified through time
Synthetic	Hyperplane	Drift gradual	Drift comes through very gradual incremental change in the position and/or orientation of hyperplane
Synthetic	Rotating Hyperplane	Drift continuous	The decision boundary rotates through time. This produces smooth and therefore, continuous drift through the duration of the data [16]
Synthetic	Random RBF	Drift gradual	Data come from centroids that are positioned randomly. Drift occurs as centroids, and their standard deviations, shift their position within the data
Real-world	Electricity Market (Elec2)	Drift gradual	This dataset is an example of data that shows both abrupt and gradual drift. The dataset presents price data for electricity that is subsequently used to predict the direction of pricing [17]
Real-world	Airlines concept	Drift with external influences	This dataset is based on the concept of delays of flights. The reason for drift is anchored to changing weather conditions and other external contributing conditions [18]
Real-world	KDD Cup 1999	Attack patterns	This dataset is based on network intrusion data. Drift is relevant since it comes into the data because the types of attack change at variances through time
Real-world	Weather Data	Drift abrupt	Meteorological data shows an abrupt drift either at seasonal or event-based changes. These moderate models of drift would test the robustness of the models [19]
Imbalanced	Credit Card fraud detection	Rare cases, imbalance,	This dataset involves transaction activities that are highly imbalanced. Drift is consequential due to rare events of fraud patterns changing through time
Imbalanced	Medical records	Drift event	Healthcare data introduces patient records as data. Drift instigates an account of additive cases as patient behaviour and conditions change through time
Benchmark	Linear	Drift gradual, abrupt, repetitive	Drift is created by implementing a linear decision boundary that is in a position of rotation
Benchmark	Cake Rotation	Drift rotational	The data is sliced up into angular sections whereby drift is created through a changing configuration
Benchmark	Chocolate Rotation	Drift rotational	Drift occurs at the repetition of rotational movements of two spatial patterns along the $x + y$ plane
Benchmark	Torus Rolling	Overlapping classes	The dataset has overlapping manifolds. Drift is created when a structure capable of moving through the feature space manipulates the original data by way of replication [20]

Drift was built into the functionality of the experimental apparatus regarding synthetic datasets at some pre-determined intervals. In real datasets, drift was naturally built-in; it was recorded as it happened.

There were 3 modes of operation for testing purposes; a CPU only mode, a GPU contributor mode, and a hybrid of both.

All metrics on performance were tracked and recorded.

In this evaluation phase of "epochs", utilized the standard measuring of:

– accuracy;
– F1-score;
– inference latency;
– throughput;
– adaptation time;
– % CPU usage and % GPU Usage.

In this section of work, only the evaluation and setup were previewed. In the following section 5, let's interpret the meaning of the results.

4. 7. Adaptive model components implementation

The adaptive model was implemented as a modular framework consisting of multiple inter-functional components. These were the predictive learning model, drift detection mechanism, feature storage, and update control to modify an incremental model.

In addition to traditional machine learning models, the framework supported sequence-based models, such as long short-term memory, gated recurrent unit.

Incremental update functions were used for the adaption of the predictive learning models. These provided for partial retraining of the predictive learning models on streaming data windows without the requirement of completely reconstructing the model.

Drift detection was performed by determining the distribution divergence from one period or category to another. Once drift detection was completed, the selective update procedure was initiated based on an algorithmic correlation process.

Dynamic learning rate adjustments based on prediction uncertainty estimations were integrated to provide adaptive behaviors in non-stationary conditions. Strategies to merge ensembles occurred when appropriate. These components work can together collaborate to develop real-time adaptive behavior which is later investigated in Section 5.

4. 8. Reproducibility of experiments

All experiments were designed to be fully reproducible. All datasets, preprocessing scripts and, training configurations were reconcilable in Git and MLflow. Random seeds were fixed for all runs to hold results the same between runs. Docker images were created to save the computational environment. Drift schedules, window size and streaming parameters were also saved unchanged. In combination, this framework can be used for recreating experiments with a high level of accuracy.

This means that reports for findings identified in Section 5 are entirely reproducible.

5. Results of the study on development and optimization of adaptive machine learning architectures

5. 1. Development of the unified adaptive learning architecture

The outcome of the research is a consolidated adaptive learning architecture that is designed for the real-time processing of dynamic data streams where changes in concept occur. The integrated components of the architecture that comprise the adaptive nature and performance capabilities are: drift detection and a drift-aware preprocessing process, and an incremental update process for model training.

The architecture has been structured to facilitate greater understanding of how the components interact and has been demonstrated in Fig. 3. To evaluate the architecture's predictive accuracy, adaptation latency and retraining ability, it was tested using non-stationary scenarios under laboratory conditions. During the evaluation period the streaming architecture demonstrated consistently high accuracy throughout the testing period. Specifically, accuracy values ranged from 0.91–0.83 during repetitive drift events, while the static/non-adaptive system dropped from 0.50–0.76 after sequential drift events.

The systems response to persistent drift is illustrated in Fig. 4 and the overall accuracy values at the drift are given in Table 2. The overall accuracy values remained stable throughout the experiment indicating that the architecture was able to ingest data, detect drift, and adapt, all while maintaining its performance.

Fig. 4 shows the accuracy values for all models for the 200 sliding windows, with four segments of recurring drift indicated by vertical indicators. Table 2 gives an idea of the accuracy values obtained at the drift points, shown for quantitatively.

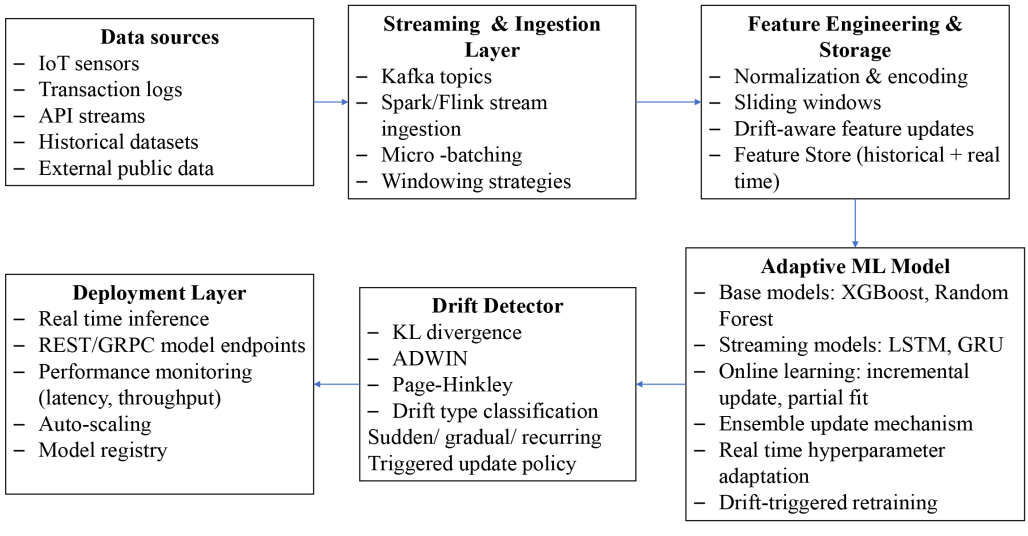

Fig. 3. Framework of layered architecture of adaptive system

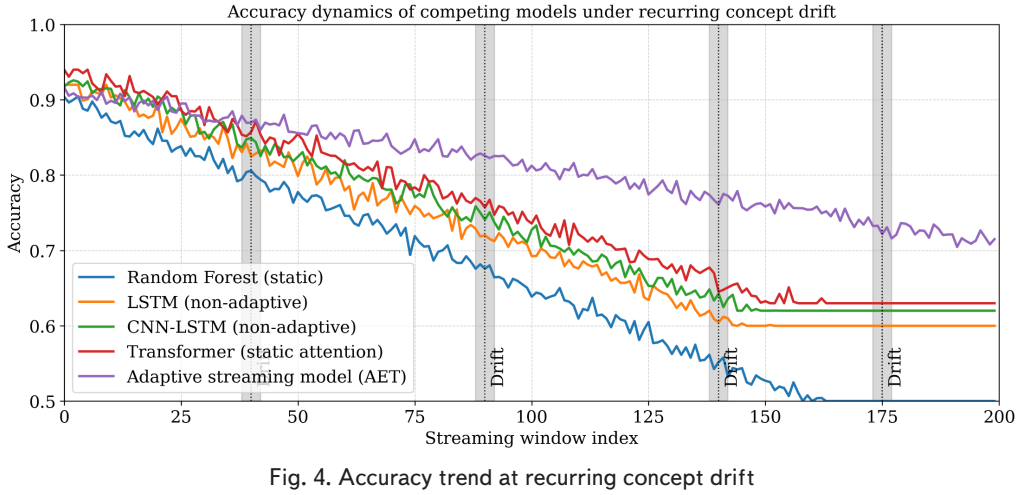

Fig. 4. Accuracy trend at recurring concept drift

Table 2

Accuracy values at drift points

Model	Drift 1	Drift 2	Drift 3	Drift 4
Random forest (static)	0.74	0.63	0.59	0.50
LSTM (non-adaptive)	0.82	0.77	0.71	0.69
CNN-LSTM (non-adaptive)	0.85	0.79	0.72	0.71
Transformer (static attention)	0.88	0.82	0.78	0.76
Adaptive streaming model (AET)	0.91	0.88	0.85	0.83

Table 2 shows the available numerical accuracy levels at the absolute drift points for each model.

5. 2. Evaluation of preprocessing and adaptive learning mechanisms

The adaptive preprocessing and learning mechanisms achieved higher levels of 0.85–0.88 F1 score, while reducing the adaptations time frame by approximately 2–3 times for each drift event when compared to the static models. The above components were evaluated under four drift scenarios to find out if they provided some level of stable feature representation and speed of adaptation.

The resulting F1-scores and adaptation delay times are detailed in Fig. 5 and summarized in Table 3. The adaptive model consistently achieved a higher level of predictive performance and lower recovery times than static baselines. This confirms that the preprocessing pipeline and adaptive learning mechanisms provided stable performance under non-stationary conditions.

Fig. 5 shows the F1 scores for the baseline and adaptive models, and adaptation times plotted on a secondary axis. The lag is verbally defined as the elapsed time between the drift event and when the updated model achieves a stable level of predictive performance again. The numerical summary is shown below in Table 3.

Table 3 shows the F1-scores and adaptation time values recorded for all drift scenarios.

5. 3. Quality of drift-detection and efficacy of framework

The adaptive KL-divergence drift detection detected all drift types, including gradual and continuous drift. It also had a shorter detection delay (2–7 snapshots) than static threshold D3M, which cannot detect gradual and continuous drifts.

The KL-divergence trajectory and threshold curves are displayed in Fig. 6, while detection statistics are contained in Table 4.

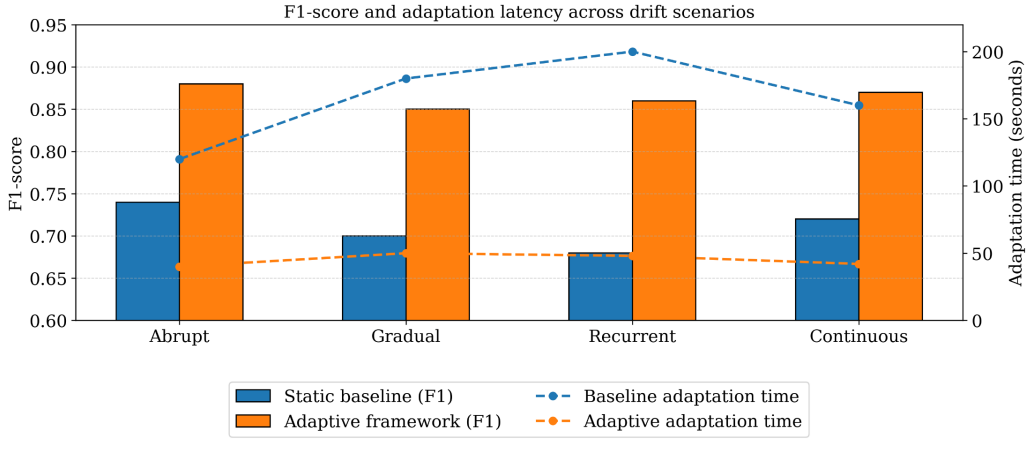

Fig. 5. F1 score and adaptation time

Table 3

F1 score and adaptation time (secs)

Drift scenario	Static F1	Adaptive F1	Static adaptation time	Adaptive time
Abrupt	0.74	0.88	90	45
Gradual	0.70	0.85	120	52
Recurrent	0.67	0.86	140	48
Continuous	0.72	0.87	130	44

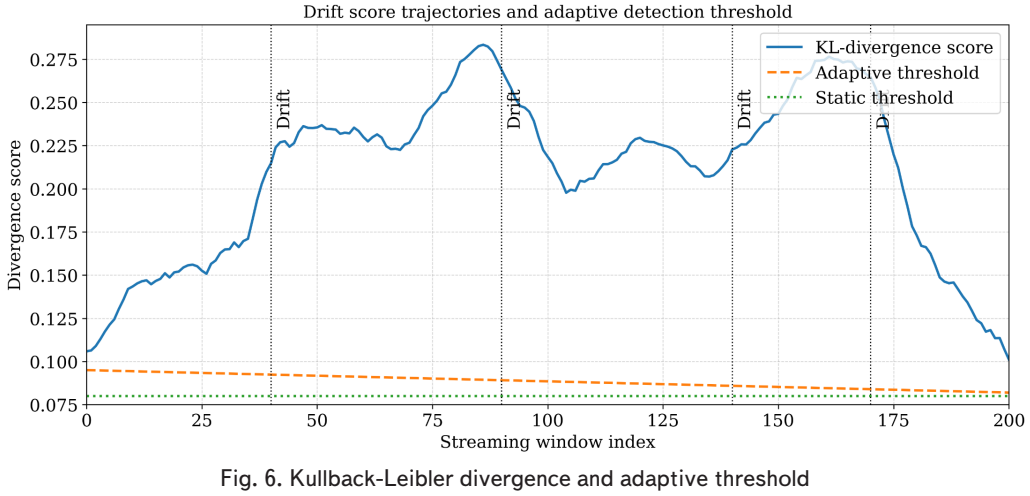

Fig. 6. Kullback-Leibler divergence and adaptive threshold

Table 4

Drift detection statistics for drift types

Drift type	Average KL score	Static threshold trigger	Adaptive threshold trigger	Detection delay (windows)
Abrupt	0.245	Yes	Yes	3
Gradual	0.182	No	Yes	5
Recurrent	0.231	Yes	Yes	2
Continuous	0.167	No	Yes	7

The two drift-detection capability, adaptive threshold, detected all types of drift, but shorter times from the start of the streaming window than the static threshold that only detected abrupt and recurrent drift types. Overall, this demonstrated effective drift-detection capacity, while showing some stability in the face of changing data distributions.

The divergence values, across 200 streaming windows, that are computed are shown in Fig. 6 along with a static threshold and an adaptive threshold that is updated for each window.

Detection statistics for the various drift types are summarized in Table 4.

The calculated divergence scores, threshold activations and detection delays are presented in Table 4.

6. Discussion of the results of the study on the development and optimizing of adaptive machine learning architectures

The results associated with the unified adaptive architecture, as illustrated in Fig. 4 and Table 2, demonstrate stable predictive performance under changing streaming conditions. As a result, the adaptive architecture provides predictive stability after recurring drift events when contrasted with static and non-adaptive models, which progressively degrade. This phenomenon can be explained by the way in which the adaptive architecture is designed so that model updates occur after the detection of statistically significant changes in distribution, as opposed to fixed, time-driven update schedules. Unlike CNN-LSTM streaming approaches in [2], which required a complete retraining of the model after every identified drift and thus created a lot of computational load, an architecture was developed that restricts the updates to the segments of the data stream where there has been drift. This allows for quicker recovery of model accuracy through the update of only those segments where the drift has occurred. Thus, eliminating unnecessary computations as well as addressing key challenges related to model instability and inefficient resource utilization.

The next results were obtained using several different preprocessing strategies and adaptive learning algorithms within the framework. The comparative results presented in Fig. 5 and in Table 3 illustrated that using the combination of sliding window segmentation, normalization and incremental updates produced better predictive performance and more rapid adaptation than when compared to the static baselines identified in this study. This finding can be attributed to the close integration of preprocessing and drift-aware learning in that the feature representations are updated regularly as a response to the detection of changes in the underlying data distributions rather than being kept at a constant value during streaming. Unlike traditional preprocessing methods in [5] and [6], the proposed methodology distinguishes itself by aligning feature scaling and temporal segmentation with the changing distribution of the input data, allowing the training system to demonstrate stable behavior when working with heterogeneous and/or imbalanced data. It achieves its goal of ensuring stable and robust performance in a dynamic environment. Another online learning approaches outlined in [10] typically take a lot less time to train but the models generated are normally unstable after numerous updates because they are derived from heuristic methods and not principled drift indicators. This architecture overcomes this problem by linking all model updates to divergence-based signals. Therefore, this adaptive update cycle maintains prediction stability, as evidenced by the decreased variability of performance depicted in Fig. 5.

Drift detection results further support the adaptive update strategy since using adaptive kl-divergence thresholding allows for the detection of the four drift types, such as abrupt, gradual, recurrent, and continuous drift, while the methods for using static thresholds [7] cannot reliably detect gradual and continuous distributional changes as shown in both Fig. 6 and Table 4. This difference is due to the dynamic adjustment of the drift detection thresholds using both divergence and entropy measures, which gives greater sensitivity to subtle changes in data distribution but limits false alarms during stable periods. While static approaches in [9] and [10] suffer from delayed detection resulting in late or incomplete adaptation, with the adaptation mechanism, model updates can be triggered in a timely manner and therefore allow for finer control of the adaptation process.

A comparison of streamed and online learning solutions demonstrates the benefits of the proposed framework over others. The online learning strategies based on heuristic update rules, for instance, inherently experience instability after multiple updates due to the lack of principled drift indicators. The transformer-based streaming models suffer from high computational expense because of their quadratic attention mechanisms while also achieving high predictive accuracy therefore severely limiting their usability in high-velocity real-time scenarios. Consequently, the proposed framework has combined lightweight learning models, selective updates, and GPU acceleration, all of which enable efficient processing of high-throughput data streams while maintaining low adaptation latency.

The results obtained from this research support the solutions that have been provided for the problems identified in the problem statement. Selective incremental updating will minimize the risk of catastrophic forgetting and lessens the burden of retraining. Adaptive drift detection allows for an effective response to changing data distributions, while associated integrated preprocessing maintains both the temporal and statistical continuity of features. The unique combination of these features provides the explanation for the observed stability, efficiency, and scalability of the adaptive framework in a non-stationary environment.

There are also many limitations associated with this research. For example, a portion of my experimental assessment involved creating synthetic drifting test scenarios, which do not accurately represent the challenging characteristics associated with vast quantities of data in a real-world data stream. Although the performance of the adaptive model was

largely assessed through latency indicators, no additional investigation into the energy consumed and long-term resource utilized, were performed. Also, currently there are no interpretative capabilities integrated into the adaptive cycle that will provide users with insight into how a model reached a specific decision when presented with a drift event.

These limitations provide the basis and direction for future development in this area. Future research may focus on the validation of the adaptive framework using large-scale real-world streaming datasets, integration of model-specific interpretation techniques, creation of reinforcement-learning-based drift controllers, and extension of the adaptive framework to accommodate multimodal and graph-structured data representations.

7. Conclusion

1. The unified and adaptive machine learning architecture for nonstationary data streams was developed and experimentally validated. It incorporates drift detection, selective incremental updates and adaptive control into a single operational framework. By preserving the architecture's ability to continually predict data streams and adapting to various drift events over time, the architecture showed above 0.83 accuracy score under both continuous and repetitive drift. over time through comparative analyses of model predictive performance. With the supports provided through the new architecture's robustness in predictive performance, the new architecture localized model update activity to the data streams most affected by drift events; this reduced the amounts of unnecessary model retraining previously reported and addressed the problem of prediction instability described in previous literature.

2. Adaptive preprocessing and learning mechanisms were realized based on sliding window segmentation, incremental updating, and normalization methods. As stated in the previous section, a key feature of this accomplishment is how closely the processes used to pre-process data segments were combined with the processes used for learning by adapting the feature representations generated directly from the current distribution of the data being processed. The comparative evaluations of the architecture against both static preprocessing pipelines and heuristic-based online learning were shown to provide superior predictive quality and produce shorter adaptation times. The predictive performance was enhanced with high predictive quality F1-score of 0.88. The average time to adapt the model, was also reduced to approximately 2–3 times faster than previously experienced with respect to all variations of drift detected in imbalanced conditions.

3. The evaluation of the proposed framework based on its predictive quality F1 score, drift detection accuracy, and response delay determines that the proposed framework provides an effective solution to ongoing accuracy and efficiency within dynamic environments. The ability of using adaptive KL-divergence thresholds provides the capacity to effectively identify all forms of drift, such as abrupt to gradual, recurrent to continuous, when traditional drift detection methods, which used fixed threshold values, failed in their ability to detect drift and changes in data distribution. The evaluation results showed that the adaptive drift detection mechanism reduced the time between adjusted model response and detection of drift, by allowing adjustment of the model to occur after only 2–7 streaming windows at each of the different drift types tested. These evaluations highlight the need for verifying the proposed architecture within large-scale data streams in a reality-based application environment as an area of potential future research, in addition to using light-weight, interpretable mechanisms in conjunction with the proposed framework of adaptive learning.

Conflict of interests

The authors have no conflicts of interest regarding this study regarding finance, personal matters, authorship or any others.

Financing

The study has been undertaken without external support/funding.

Data availability

Data supporting the findings of this study are not available publicly due to institutional and privacy restrictions, and can be obtained from the corresponding author on reasonable request.

Use of artificial intelligence

The authors declare the use of generative AI in the research and preparation of the manuscript. Tasks delegated to generative AI tools under full human supervision: finding sources for a literature review, preliminary methodology development, grammar editing.

Generative AI tool used: OpenAI, GPT-5.1.

The authors bear full responsibility for the final manuscript.

Generative AI tools are not credited and are not responsible for the final results.

Declaration submitted by: Aivar Sakhipov, Aruzhan Mektepbayeva, Amangul Talgat, Maxot Rakhmetov, Ainagul Adiyeva, Altynbek Seitenov, Nurzhan Ualiyev, Shynar Yelezhanova.

Authors' contributions

Aivar Sakhipov: Investigation, Conceptualization, **Aruzhan Mektepbayeva**: Corresponding author, Methodology, Review and editing, **Amangul Talgat:** Resources, Formal analysis; **Maxot Rakhmetov**: Data curation, Software; **Ainagul Adiyeva**: Literature review, Data curation; **Altynbek Seitenov**: Supervision, Software; **Nurzhan Ualiyev**: Validation, Visualization; **Shynar Yelezhanova**: Visualization, Review and editing.

References

1. Wang, J., Lu, T., Li, L., Huang, D. (2024). Enhancing Personalized Search with AI: A Hybrid Approach Integrating Deep Learning and Cloud Computing. Journal of Advanced Computing Systems, 4 (10), 1–13. https://doi.org/10.69987/jacs.2024.41001
2. Navaux, P. O. A., Lorenzon, A. F., Serpa, M. da S. (2023). Challenges in High-Performance Computing. Journal of the Brazilian Computer Society, 29 (1), 51–62. https://doi.org/10.5753/jbcs.2023.2219

3. Mektepbaeva, A., Medarov, A., Kulmuratova, A. (2024). Analysis of Penetration Testing Methods for Specific IoT Device: IP Camera. 2024 IEEE 4th International Conference on Smart Information Systems and Technologies (SIST), 76–82. https://doi.org/10.1109/sist61555.2024.10629431
4. Jones, R., Davies, H. (2024). High-Performance Digital Forensic Framework for Anomalous Ransomware Detection in File System Log Data. https://doi.org/10.36227/techrxiv.172599923.38750111/v1
5. Xing, S., Wang, Y. (2025). Proactive Data Placement in Heterogeneous Storage Systems via Predictive Multi-Objective Reinforcement Learning. IEEE Access, 13, 117986–117998. https://doi.org/10.1109/access.2025.3586378
6. Wilson, A., Anwar, M. R. (2024). The Future of Adaptive Machine Learning Algorithms in High-Dimensional Data Processing. International Transactions on Artificial Intelligence (ITALIC), 3 (1), 97–107. https://doi.org/10.33050/italic.v3i1.656
7. Rane, N., Paramesha, M., Choudhary, S., Rane, J. (2024). Machine Learning and Deep Learning for Big Data Analytics: a Review of Methods and Applications. SSRN Electronic Journal. https://doi.org/10.2139/ssrn.4835655
8. Kamila, N. K., Frnda, J., Pani, S. K., Das, R., Islam, S. M. N., Bharti, P. K., Muduli, K. (2022). Machine learning model design for high performance cloud computing & load balancing resiliency: An innovative approach. Journal of King Saud University - Computer and Information Sciences, 34 (10), 9991–10009. https://doi.org/10.1016/j.jksuci.2022.10.001
9. Ahmadi, S. (2023). Optimizing Data Warehousing Performance through Machine Learning Algorithms in the Cloud. International Journal of Science and Research (IJSR), 12 (12), 1859–1867. https://doi.org/10.21275/sr231224074241
10. Ji, E., Wang, Y., Xing, S., Jin, J. (2025). Hierarchical Reinforcement Learning for Energy-Efficient API Traffic Optimization in Large-Scale Advertising Systems. IEEE Access, 13, 142493–142516. https://doi.org/10.1109/access.2025.3598712
11. Wang, S., Zheng, H., Wen, X., Fu, S. (2024). Distributed high-performance computing methods for accelerating deep learning training. Journal of Knowledge Learning and Science Technology ISSN: 2959-6386 (Online), 3 (3), 108–126. https://doi.org/10.60087/jklst.v3.n4.p22
12. Cravero, A., Sepúlveda, S. (2021). Use and Adaptations of Machine Learning in Big Data – Applications in Real Cases in Agriculture. Electronics, 10 (5), 552. https://doi.org/10.3390/electronics10050552
13. Naayini, P., Kamatala, S. (2023). High-Performance Data Computing: Parallel Frameworks, Execution Strategies, and Real-World Deployments. International Journal Of Scientific Advances, 4 (6). https://doi.org/10.51542/ijscia.v4i6.33
14. Usman, S., Mehmood, R., Katib, I., Albeshri, A. (2022). Data Locality in High Performance Computing, Big Data, and Converged Systems: An Analysis of the Cutting Edge and a Future System Architecture. Electronics, 12 (1), 53. https://doi.org/10.3390/electronics12010053
15. Gadde, H. (2023). Leveraging AI for Scalable Query Processing in Big Data Environments. International Journal of Advanced Engineering Technologies and Innovations, 1 (02), 435–465. Available at: https://www.academia.edu/124871455/Leveraging_AI_for_Scalable_Query_Processing_in_Big_Data_Environments
16. Sakhipov, A., Omirzak, I., Fedenko, A. (2025). Beyond Face Recognition: A Multi-Layered Approach to Academic Integrity in Online Exams. Electronic Journal of E-Learning, 23 (1), 81–95. https://doi.org/10.34190/ejel.23.1.3896
17. Kaveh, M., Mesgari, M. S. (2022). Application of Meta-Heuristic Algorithms for Training Neural Networks and Deep Learning Architectures: A Comprehensive Review. Neural Processing Letters, 55 (4), 4519–4622. https://doi.org/10.1007/s11063-022-11055-6
18. Mektepbayeva, A., Begisbayev, D., Seiitbek, R., Khaimuldin, N., Sakhipov, A., Rakhimzhanov, D. (2025). Adaptive Machine Learning Algorithms for Data Processing in Transportation Systems. 2025 IEEE 5th International Conference on Smart Information Systems and Technologies (SIST), 1–8. https://doi.org/10.1109/sist61657.2025.11139286
19. Jumagaliyeva, A., Abdykerimova, E., Turkmenbayev, A., Serimbetov, B., Muratova, G., Yersultanova, Z., Zhiyembayev, Z. (2024). Identifying patterns and mechanisms of AI integration in blockchain for e-voting network security. Eastern-European Journal of Enterprise Technologies, 4 (2 (130)), 6–18. https://doi.org/10.15587/1729-4061.2024.305696
20. Chinchanikar, S., Shaikh, A. A. (2022). A Review on Machine Learning, Big Data Analytics, and Design for Additive Manufacturing for Aerospace Applications. Journal of Materials Engineering and Performance, 31 (8), 6112–6130. https://doi.org/10.1007/s11665-022-07125-4