



Adaptive windowing based recurrent neural network for drift adaption in non-stationary environment

Shubhangi Suryawanshi^{1,2} · Anurag Goswami¹ · Pramod Patil² · Vipul Mishra¹

Received: 24 November 2021 / Accepted: 6 June 2022 / Published online: 30 June 2022
© The Author(s), under exclusive licence to Springer-Verlag GmbH Germany, part of Springer Nature 2022

Abstract

In today's digital era, many applications generate massive data streams that must be sequenced and processed immediately. Therefore, storing large amounts of data for analysis is impractical. Now, this infinite amount of evolving data confronts concept drifts in data stream classification. Concept drift is a phenomenon in which the distribution of input data or the relationship between input data and target label changes over time. If the drifts are not addressed, the learning model's performance suffers. Non-stationary data streams must be processed as they arrive, and neural networks' built-in capabilities aid in the processing of huge non-stationary data streams. We proposed an adaptive windowing approach based on a gated recurrent unit, a variant of the recurrent neural network incrementally trained on incoming data (for the real-world airline and synthetic Streaming Ensemble Algorithm (SEA) datasets), and employed elastic weight consolidation with the Fisher information matrix to prevent forgetting. Unlike the traditional fixed window methodology, the proposed model dynamically increases the window size if the prediction is correct and reduces it if drifts occur. As a result, an adaptive recurrent neural network model can adapt to changes in the non-stationary data stream and provide consistent performance. Moreover, the findings revealed that on the airline and the SEA dataset, the proposed model outperforms state-of-the-art methods by achieving 67.74% and 91.70% accuracy, respectively. Further, the results demonstrated that the proposed model has a better accuracy of 3.6% and 1.6% for the SEA and the airline dataset, respectively.

Keywords Non-stationary data streams · Gated recurrent unit · Fisher information matrix · Concept drift · Catastrophic forgetting · Adaptive window

1 Introduction

In the modern world of computing, a variety of applications generate a vast volume of data streams at fast speeds. A data stream is an unbounded flow of temporally sequenced data that arrives in real-time (Charu 2015). Massive streaming data are generated by real-time applications such as spam

detection, weather forecasting, intrusion detection, internet of things, telecommunications, and e-commerce websites. Owing to such data's requirements of vast memory, quick processing, and real-time decision support, traditional data mining algorithms cannot be applied (Bifet and Kirkby 2009; Hoens and Polikar 2012; Sheu et al. 2017). Moreover, prediction in real-world data stream applications is confronted by concept drift. A change in the distribution of input data or the relationship between the target label is referred to as a "concept drift". (The target label is referred to as the concept that is predicted by the class.) (Wares et al. 2019)

The concept drift is categorized into two types: virtual and real.

In data stream classification, $X \in \{x_1, x_2, \dots, x_n\}$ is a feature space and $Y \in \{Y_1, Y_2, \dots, Y_n\}$ is the target or class label. Classification in real-world data stream applications is confronted by concept drift. The prior probability of a given target class is represented as $P(Y)$, $P(Y|X)$ represents the posterior distributions of the class, and the data distribution

✉ Shubhangi Suryawanshi
ss5683@bennett.edu.in

Anurag Goswami
anurag.goswami@bennett.edu.in

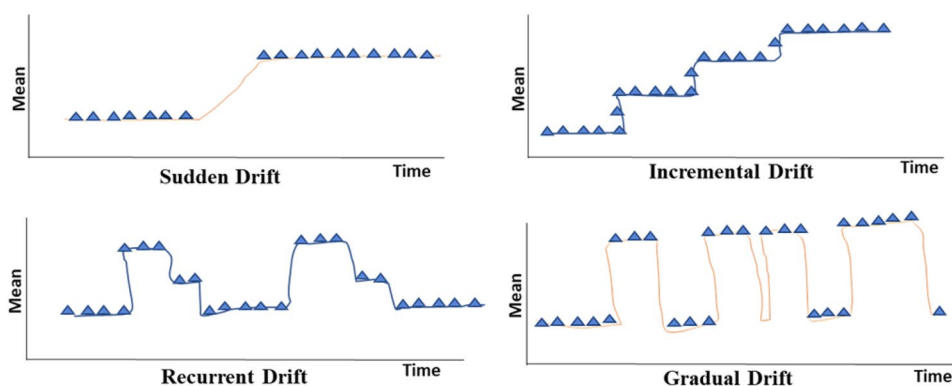
Pramod Patil
pdpatiljune@gmail.com

Vipul Mishra
vipul.mishra@bennett.edu.in

¹ Bennett University, Greater Noida, India

² Dr. D. Y. Patil Institute of Technology, Pimpri, Pune, India

Fig. 1 Categories of concept drifts



of the classes is represented as $P(X|Y)$ (Lu Jie and Zhang 2018).

In a virtual concept drift, the data distribution $P(X)$ varies and, consequently, $P(X|Y)$ changes, although the posterior probability remains constant. The boundaries of the class remain unaltered. In a real drift, a change in $P(Y|X)$ changes the decision boundaries or the target label. The posterior probability varies with or without the data distribution in a real concept drift. After considering the new information, the revised probability that occurs because of new information is known as posterior probability (Wares et al. 2019; Lu Jie and Zhang 2018).

Concept drift presents itself in several ways, as seen in Fig. 1, which shows different patterns—sudden, incremental, gradual, and recurrent drifts (Lu Jie and Zhang 2018). The data distribution changes abruptly in sudden drift; for example, in the current COVID-19 scenario, due to the lockdown, the users' shopping patterns have changed abruptly. Conversely, gradual drift demonstrates a slow transition from one data distribution to the other; for example, owing to the lockdown imposed due to the COVID-19 pandemic, inflation was influenced gradually over the months. Moreover, data distributions exhibit sequential or step-wise changes and take a longer time to appear in incremental drift; for example, following the pandemic, people's travel habits changed incrementally after the lockdown was lifted. In recurrent drift, the previous concept or target label reappears with the same data distribution after some time (Lu Jie and Zhang 2018), such as when consumers' purchasing habits change every year during the festive season. Overall, regardless of the type, a model's performance will suffer substantially if the drift is not addressed.

Traditional decision-making becomes ineffective in non-stationary data stream environments due to the lack of prior knowledge of data streaming patterns (Bifet and Kirkby 2009; Hoens and Polikar 2012; Sheu et al. 2017); thus, artificial intelligence models become prominent.

Many machine learning approaches have been used to handle various types of concept drift detection and adaptation (Charu 2015; Shikha Mehta 2017; Ayad 2014;

Nishimura et al. 2008). Machine learning models, which follow a traditional approach, are based on the assumption of an underlying pattern and assume that in data streams, the concepts (i.e., target labels) do not change. Moreover, a machine learning model takes into account the data's stationary properties, which contradicts the reality of concept drifts and impacts the classification model's performance. Thus, since prior knowledge of pattern changes in the data is required, the conventional machine learning methods are ineffective in dealing with concept drift (Ksieniewicz et al. 2019; Nishimura 2008; Charu 2015; Shikha Mehta 2017).

Further, deep learning models do an excellent work of identifying the intricate structure in massive data (Lemos Neto et al. 2020; Ksieniewicz et al. 2019; Jun Gao and Murphey 2020). However, the deep learning model faces two major challenges when processing data streams. First, in streaming data not all the data is available at the time of training. It's difficult to handle a continuously arriving data stream with a static neural network structure for faster convergence and drift adaptation. Second, the evolving neural network is confronted with the problem of catastrophic forgetting. Previously learned knowledge is replaced with new knowledge during catastrophic forgetting. When a evolving neural network learns a new data and forgets the old one, its performance deteriorates (Alvaro et al. 2020; Ksieniewicz et al. 2019; Baier et al. 2021).

Therefore, it is crucial to design a learning model that can adapt to concept drifts while learning a continuously arriving data stream. However, several challenges are imposed by data stream classification as it is not a trivial task and differs from standard classification scenarios. Data Stream classification works on an infinite amount of partially sequenced data and requires immediate processing. Furthermore, the designed algorithm should incrementally learn new data and have an evolving nature while preventing catastrophic forgetting, as the underlying concept of the new data changes over time.

Therefore, to address the above discussed challenges, we proposed a new concept drift adaptation model for

non-stationary environments, that is incrementally trained on incoming data using an adaptive recurrent neural network based on adaptive windowing with a forgetting mechanism.

1.1 Our contribution

The main contribution of the study is that it uses an adaptive windowing-based gated recurrent unit model that is incrementally trained on incoming data in order to improve drift adaption in non-stationary environments without compromising performance. In the proposed model, the size of the window varies based on the drifts: If a drift is detected, the window will be halved, and when there are no drifts, the window will expand.

A small window is utilized to detect sudden changes quickly, whereas a larger window efficiently detects gradual changes. adaptive gated recurrent unit is used to adapt to the drift once it is recognized by the drift detection module, and a new layer is added to the gated recurrent unit to deal with drifts. To overcome the problem of catastrophic forgetting, elastic weight consolidation with a Fisher information matrix is utilized, which prevents the current weight from deviating from the previous one. Consequently, information from previous tasks is retained while learning new ones.

The proposed model comprises an initial configuration with the gated recurrent unit (GRU) layer, one hidden layer with 128 neurons, and an output layer with one neuron. The learning rate is set to 0.01, the rectified linear unit (ReLU) activation function is used for the hidden layers, and the sigmoid activation function is used for the output layer.

The key contributions of the study can be summarized in the following points:

- Enhancement in drift adaption using the adaptive windowing-based evolving gated recurrent unit without compromising performance.
- Defined an evolving gated recurrent unit with growing hidden layers on the occurrence of the drift for better drift adaption.
- Incorporated elastic weight consolidation with a Fisher information matrix in an evolving gated recurrent unit to solve the problem of catastrophic forgetting.

The rest of this paper is organized as follows: The background is described in Sect. 2. Section 3 discusses related work; Sect. 4 explains the research goal, the dataset used, and the proposed methodology followed; Sect. 5 discusses the experiment and results, and finally, conclusions and future directions are presented in Sect. 6.

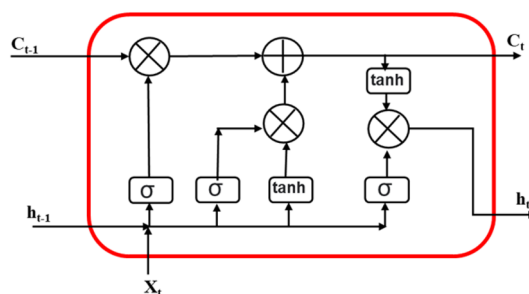


Fig. 2 Long Short Term Memory cell diagram

2 Background

This section describes the background of the terms utilized in this paper. The recurrent neural network variant used in this study is described as follows:

2.1 Long short term memory (LSTM)

One type of recurrent neural network is LSTM (Hochreiter and Schmidhuber 1997). LSTM is more effective at preserving long-term dependency and sequence modeling than other RNN variants. It consists of three gates: a forget gate, an input gate, and an output gate. Figure 2 shows the internal cell structure of LSTM. Below is an explanation of how the cell structure works and how it processes information (Hochreiter and Schmidhuber 1997).

Forget gate: It determines which information needs more attention and which may be ignored. The input data X_t and the previous hidden state h_{t-1} are passed to the activation function σ , which is a sigmoid. If older output is required, forget gate returns a value closer to 1; otherwise, forget gate ignores it. W_f represents the forget gate’s weight matrices, and b_f represents the bias. The forget gate equation Eq. 1 is shown below:

$$\Gamma_f = \sigma(W_f[h_{t-1}, X_t]) + b_f \tag{1}$$

Input gate: To update the cell state, the input gate transfers the previous h_{t-1} hidden state and the current input X_t to the next activation function for outputting of the appropriate information, with the sigmoid set to 1 or 0 (to ignore). Tanh activation is used to regulate the network by taking into account the X_t and h_{t-1} as well as the cell state C_t as shown in Eq. 3. The forget factor f_t is employed. The bias is represented by b_i , while the input gate weight matrix is represented by W_i . The input gate Eq. 2 is shown below:

$$\Gamma_i = \sigma(W_i[h_{t-1}, X_t]) + b_i \tag{2}$$

$$C_t = \tanh(W_c[h_{t-1}, X_t]) + b_c \tag{3}$$

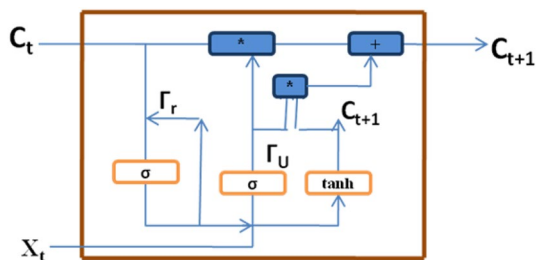


Fig. 3 Gated recurrent unit cell diagram

Output gate: The next hidden state value will be decided by the output gate. The output gate Eq. 4 is shown below:

$$\Gamma_o = \sigma(W_o[h_{t-1}, X_t] + b_o) \tag{4}$$

In Eq. 4, Γ_o represents the output gate, and σ denotes the sigmoid activation function. The bias is denoted by b_o , while the output gate weight matrix is denoted by W_o . \tanh is an activation function.

For the generation of new cell state Eq. 6 and next hidden state Eq. 5 illustrated below,

$$h_t = \Gamma_o * \tanh(C_t) \tag{5}$$

$$C_{t+1} = \Gamma_u * C_t + \Gamma_f * C_{t-1} \tag{6}$$

In Eq. 5, Γ_o represents output of the output gate, \tanh is an activation function and C_t represents the current cell state. In Eq. 6, Γ_u represents the update gate output, the cell state C_t and C_{t-1} represent the current and previous cell state, and Γ_f represents the forget gate output.

2.2 Gated recurrent unit (GRU)

GRU is a simplified and faster variant of LSTM (Junyoung Chung and Bengio 2014). GRU makes use of the update and resets gates. It efficiently preserves long-term dependency (Junyoung Chung and Bengio 2014). Figure 3 shows the internal cell structure of GRU. Below is an explanation of how the cell structure works and processes the information. In GRU, C_t presents the memory cell to preserve the past knowledge, f_{a_t} is the activation function, X_t represents the input data, and b is the bias. GRU is faster to train than other recurrent neural network types (Liu et al. 2017). In GRU, every timestamp considers the overwriting of the memory cell.

$$C_{t+1} = f_{a_t}(W_C[C_t, X_{t+1}] + b) \tag{7}$$

The memory cell C_{t+1} represents the knowledge of next input data, the X_{t+1} represents the next input data, and W_C represents the weight matrix in the Eq. 7.

Update gate: It determines how much past information needs to be preserved and passed in the future. It gives a value between 0 and 1. The sigmoid activation σ is applied to calculate the value of update gate Γ_U .

$$\Gamma_U = \sigma(W_u[C_t, X_{t+1}] + b) \tag{8}$$

The update gate memorizes the value until the value is updated. In Eq. 8, W_u represents the weight matrices for the update gate, the C_t memory cell represents the knowledge of the current input data, and X_{t+1} represents the next input data. The GRU equation is shown in Eq. 9; the equation shows that if the gate value is 0, do not update it.

$$C_{t+1} = \Gamma_U * C_{t+1} + (1 - \Gamma_U) * C_t \tag{9}$$

Reset Gate- It gives the provision to forget the past information, which is not useful and is represented as Γ_r . The Γ_r computed as follows:

$$\Gamma_r = \sigma(W_r[C_t, X_{t+1}] + b) \tag{10}$$

In Eq. 10, σ is an activation function, W_r represents the reset gate’s weight matrices, C_t represents the knowledge of current input data, and X_{t+1} represents the next input data instance. In Eq. 11 \tanh is an activation function and Γ_r represents reset gate.

$$C_{t+1} = \tanh(W_C[\Gamma_r * C_t, X_{t+1}] + b) \tag{11}$$

3 Related work

In data stream classification, concept drift handling has risen as an emerging area of research due to the rapid growth of different applications in data mining communities. Significant work has been done in machine learning to identify and deal with drifts in a variety of applications such as telecommunication (Charu 2015), weather forecasting (Shikha Mehta 2017), energy consumption, and fraud detection (Charu 2015; Lu Jie and Zhang 2018).

The study (Bifet and Kirkby 2009) presented neural networks for data stream processing, demonstrating how their simplicity successfully handles the data stream. The study suggested that to manage the data stream, a neural network with back-propagation can be used, which avoids repetitive scanning of data streams. In the non-stationary time series (A time series is a collection of data that emerges in succession across time, with statistical characteristics such as covariance, mean, and variance changing with time.), a feed-forward neural network with auto regressive procedures is used to address the concept drift problem and its influence on risk forecasting (Suicheng Gu and He 2013). However, the challenge with this technique lies in determining a drift type in the real world.

The study (Ksieniewicz et al. 2019) employed a multi-layer perceptron with a hidden layer to evaluate labeling costs in data streams and their impact on classification accuracy. Catastrophic forgetting was used in the study to avoid the problem of forgetting and to manage both abrupt and gradual drifts. The author proposed a neural network ensemble for various types of drifts. Another study (Jun Gao and Murphey 2020) applied an LSTM neural network with adjustable windowing to identify temporal dependency and obtain greater accuracy in advanced driver assistance systems in real-time. LSTM with adaptive windowing was utilized to extract the patterns from the data; the efficacy of dynamic window size adaptations aided in the selection of the most prominent features in multi-modal data.

In one of our studies (Suryawanshi et al. 2021), a one-class ensemble classifier was progressively trained on positive data samples to handle both gradual and abrupt drifts. A one-class classifier predicts its counterparts based on simply a positive sample. It is difficult to distinguish between outliers and drifts when only considering positive samples in one-class classification.

Several machine learning algorithms, including decision trees, random forests, Naive Bayes, Hoeffding trees, and support vector machines, are used to identify and handle concept drift (Ayad 2014; Yang and Fong 2012; Hemalatha and Pathak 2019; Rad and Haeri 2019). Machine learning-based techniques need decades of domain expertise to adapt to drift (Ksieniewicz et al. 2019; Nishimura et al. 2008), whereas deep learning algorithms excel in identifying and adapting the complex structure in evolving streaming data (Jun Gao and Murphey 2020).

The study presented the most widely used dynamic weighted majority (DWM) (Kolter and Maloof 2007) for drift adaptation using a prediction technique employs a voting mechanism and a combination of a base classifier and a global ensemble. It trains a new classifier if one of the ensemble's classifiers fails, penalizing the classifier, and replaces the previous classifier if the threshold value is reached. It can manage both gradual and sudden drifts. In the research study (Sidhu and Bhatia 2017) compared the recurrent dynamic weighted majority and dynamic weighted majority ensemble classifier in their research article.

The research study (Priya and Uthra 2020) employed an ensemble approach to address the issue of class imbalance and concept drift. Different drift detectors and ensemble classifiers were evaluated on real-world and synthetic datasets. The dynamic weighted majority and accuracy weighted ensemble methods outperformed traditional ensemble classifiers for both sudden and gradual drifts. For the implementation, a Massive Online Analysis was employed.

For drift detection, the study (Song et al. 2016; Pesaranghader and Viktor 2016) combined the Hoeffding

inequality and a sliding window. The proper window size is determined and utilized the Hoeffding tree and Naive Bayes as base classifiers to identify both gradual and sudden drifts. Moreover, it obtained the smallest detection latency and, more importantly, the best accuracy by employing the Naive Bayes and Hoeffding tree as base classifiers.

Many novel ensemble techniques based on diverse voting criteria handle the concept drift (Mahdi et al. 2018; Jadhav and Deshpande 2017). The Learn Non-stationary Environments (Learn NSE) (Yan Shena and Yuquan Zhub 2017) algorithm overcomes the performance problem experienced by various ensemble classifiers by assessing prediction error on the most recent batch of data example. Evaluated the online error rate of the base classifier, and if the errors rose, a new classifier was trained on the new data. This reduced the memory requirements by simply saving the current batch of data and removing the under-performing model immediately.

In an OzaBagADWIN method, an incremental ensemble is combined with the adaptive windowing drift detector. It is an adaptation of the classic bagging classifier. To monitor the change in adaptive windowing, a variable-size window is employed and cuts are applied in its window to better adapt to the change (Oza and Russell 2001; Bifet and Pfahringer 2009).

The study (Alvaro et al. 2020) presented the incremental LSTM (ILSTM) to adapt to concept drift in data streams. The model used a fixed batch size and a static structure. In this investigation, explicit drift detection was not required.

In the study (Baier et al. 2021), uncertainty drift detection (UDD) approach was proposed to detect concept drift without the necessity for true labels; for this, a deep neural network using Monte Carlo dropouts was utilized. To detect changes in the data, the adaptive windowing technique was used to estimate the data's uncertainty over time, and the impact of the input data on the model was analyzed to avoid model retraining.

The accuracy weighted ensemble (AWE) classifier (Haixun Wang et al. 2003) that employs N static classifiers that are fixed at the start and aggregate their predictions by applying function f for training in batch mode or online streaming mode. Accuracy weighted ensemble outperforms a single classifier in terms of effectiveness. It divides the input into portions and assigns a different classifier to each portion while removing an old classifier. It effortlessly handles both stationary and non-stationary data streams.

The study (Lee et al. 2017) presented elastic weight consolidation, which introduces a regularization term that causes the current network parameters to be near the network parameters provided by previous tasks. Moreover, The study (Li and Hoiem 2016) utilized the learning-without-forgetting method to preserve the knowledge of previous steps.

The study (Kirkpatrick et al. 2017) demonstrated the elastic consolidation weight technique, which overcomes the difficulty associated with incremental learning and prevents the current weight from deviating from the prior one.

4 Experiment design

In this section, the research goals, dataset, proposed methodology, and evaluation measures are detailed. The experimentation was carried out on real-world and synthetic datasets with various drifts. The proposed model used forgetting mechanism with an adaptive neural network structure to adapt to the drift and was compared to the best-performing state-of-the-art methods such as accuracy weighted ensemble, dynamic weighted majority, Learn NSE, OzaBagADWIN, Hoeffding tree, UDD, and ILSTM (Shikha Mehta 2017; Lu Jie and Zhang 2018; Agustin et al. 2015; Alvaro et al. 2020; Baier et al. 2021).

4.1 Research goal

The main goal of this study is to improve drift adaptation without compromising classification accuracy as well as to evaluate how well an adaptive windowing-based recurrent neural network model can adapt to changes in a non-stationary data stream without forgetting.

4.2 Dataset

The datasets used in the experiment are described below:

- **Airlines dataset** - There are 7 attributes and 37,900 instances in the real-world airlines' dataset, and two classes. Time, Airline, Flight, AirportFrom, AirportTo, DayOfWeek, and Length are the attributes. These are used to predict whether or not a flight will be delayed.¹
- **Streaming Ensemble Algorithm (SEA)** - SEA is a synthetic stream generator that generates a synthetic data stream dataset with sudden and gradual concept drifts. The SEA dataset is a synthetic dataset with three attributes: 50000 instances have two essential attributes, and b is a predetermined threshold. This dataset simulates both gradual and abrupt changes. There is 10% noise in the dataset.²

¹ [Airline Dataset] <https://moa.cms.waikato.ac.nz/datasets>.

² [SEA Dataset] <https://github.com/vlosing/driftDatasets/tree/artificial/>.

4.3 Evaluation measure

The following evaluation measures were used to evaluate the proposed model on a real-world and synthetic dataset:

- (a) **Accuracy** - It is determined by taking the total number of correct predictions out of all possible outcomes.
$$Accuracy = \frac{\text{Correctly predicted values}}{\text{Total no of predictions}}$$
- (b) **Recall** - It is a measure of how well the model recognizes positive values. It is the ratio of correctly predicted positive values to the total of true positive and false negative values (Goutte and Gaussier 2005).
$$Recall = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$
- (c) **Precision** - It is the ratio of true positive values to all positive values predicted by the model. Precision gives the number of data points in the relevant class predicted by our model (Goutte and Gaussier 2005).
$$Precision = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$
- (d) **F1-Score** - To calculate the F1-score, the harmonic mean of precision and recall are used. It accurately measures the precision and recall balance (Goutte and Gaussier 2005).
$$F1\text{-Score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$
- (e) **Receiver Operating Characteristics (ROC)** - It is a probability curve, and the area under the curve (AUC) represents the degree or measure of separability. It demonstrates the model's ability to distinguish among classes.

4.4 Methodology

Figure 4 depicts the proposed model's workflow: importing and pre-processing, creating the adaptive window, drift detection, forgetting mechanism, and implementing the adaptive GRU model. Finally, the proposed model was assessed in terms of F1-score, accuracy, precision, recall, ROC curve, and confusion matrix. Python and the Sklearn library were used for the implementation. The workflow of the proposed study is described in this section. The summary of notations used to describe the proposed model and their description is summarized in Table 1.

Step 1: Pre-processing

The min-max scaler is used to scale values from 0 to 1 when the dataset is imported. The record with the missing values is removed, and label values ranging from 0 to 1 are encoded using the label encoder. The dataset is divided into equal-sized chunks to incrementally train the model on chunks.

Step 2: Adaptive Window

The adaptive window utilized in this study works in the same way as the sliding window. The difference in the size of the window varies based on whether the drift occurs or not. The data samples are separated into sub-batches using

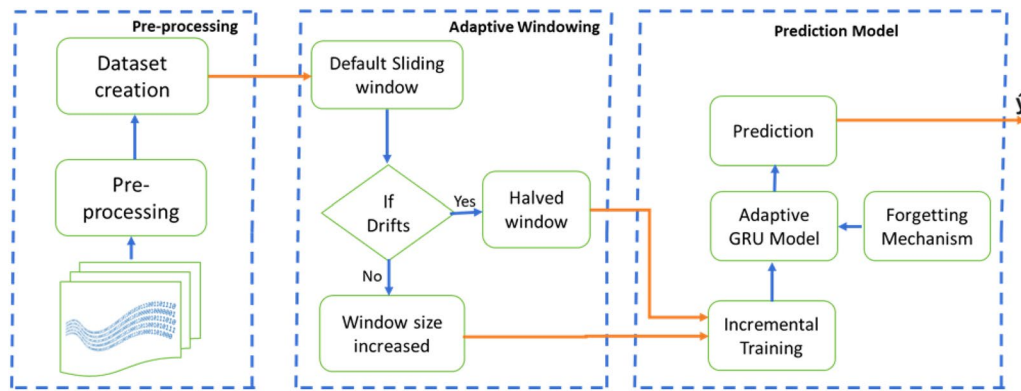


Fig. 4 Adaptive windowing based recurrent neural network for drift adaption in non-stationary environments

Table 1 Summary of notations used to describe the proposed model

Notations	Meaning
w	Window size
D_c	The cumulative difference between the mean of the current window and prior values
D_m	The minimum difference between the mean of the current window and prior values
λ	Threshold value
δ	The magnitude of the changes that are allowed
N	Number of elements in a window
X	Input features dimensions
Y	Target Label
e	Number of epochs
t	Task
θ_{c-1}	Parameters for the previous task
θ_c	Parameters for Current task
F_i	Fisher Information Matrix
$p(Y_i X_i; \theta_c)$	The log-likelihood of posterior probability for the current task
C	Constant Parameter
$L(\theta_{c-1})$	The loss from the previous task
$L(\theta_c)$	The loss of the current task

a window after they are received. If the drift is identified, the window is considered as its half, and the model training is done on the smaller window size.

The data elements arrive in a continuous stream, and only the most recent N elements are used in window strategies. If the $N*2$ predictions are correct, the size of the window is extended to its original size. The experiments were performed with window size $w = 10, 100, 200, 300, 400,$ and 500 . Several experiments were conducted to determine the optimal window size, which was chosen based on the model’s performance.

Step 3: Drift Detection

The drift detector module is called if the classifier misclassifies an instance. To track changes, the Page-Hinckley statistical test is employed (Sebastião and Fernandes 2017). It senses the change by incrementally calculating the mean of input data.

The test calculates the cumulative difference D_c between the mean of the current window and prior values as well as the minimum difference D_m and determines the threshold value λ . If the condition presented in Eq. 12 is met, an alert is raised.

$$\lambda > D_c - D_m \tag{12}$$

If the drift is spotted, the window size is halved, and the model is then trained using that small amount of data. The window size is expanded to its original size if no drift is detected and after $N*2$ correct predictions.

Step 4: Forgetting Mechanism

In data stream, incremental learning causes a learning problem known as catastrophic forgetting, wherein the model abruptly forgets previously learned knowledge while learning new information. When a model learns a new data

and forgets the old one, its performance deteriorates. The task/chunk borders are determined using the change detection approach. When a change is observed, it is marked as a new task boundary. To overcome this difficulty, the divergence of the current weights from their prior ones may be prevented.

We used elastic weight consolidation with the Fisher information matrix to decrease the effect of catastrophic forgetting.

In the provided dataset $D(X_i, Y_i)$, X_i contains the $X_i \in \{X_1, X_2, \dots, X_n\}$ data samples and $Y_i \in \{Y_1, Y_2, \dots, Y_n\}$ labels. The ideal set of parameters learned from the previous task t_1 is θ_{c-1} . To determine the optimal parameters θ_c for the current task, the elastic weight consolidation factorizes the preceding model using Bayesian estimation.

$$\theta_{c-1} = p(\theta_c | \theta_{c-1}) \quad (13)$$

Eq. 13 gives the probability of θ_{c-1} , which represents the previous model's optimal parameters, and θ_c , which indicates the current parameters. The previous batch's/task's optimum parameters θ_{c-1} are fed into the Fisher information matrix F_{c-1} shown in Eq. 14, which defines the relevance of each current parameter θ_c and confines the previous batch parameters θ_{c-1} while training on task t . The Laplace approximation is used to choose the optimal parameters θ_c for the current task. $\log(p(Y_i | X_i; \theta_c))$ represents the log-likelihood of posterior probability distribution function $p(Y_i | X_i; \theta_c)$. The constant parameters C can be ignored during training shown in Eq. 14. The probability of parameter θ_c gives an i^{th} batch target label and input data $Y_i | X_i$.

$$\log p_c = \log(p(Y_i | X_i; \theta_c)) - \lambda/2(\theta - \theta_{c-1})F_{c-1}(\theta - \theta_{c-1})^2 + C \quad (14)$$

$$L(\theta_c) = L(\theta_{c-1}) + \sum_i \lambda/2F_i(\theta_i - \theta_{c-i})^2 \quad (15)$$

The penalization regularization term is shown in Eq. 15. The Fisher information matrix F_i represents the i^{th} diagonal element. The Fisher information quantifies how much information the parameters provide about the data. After computing first-order derivatives, the Fisher information matrix is utilized to estimate the importance of each connection or parameter. $L(\theta_{c-1})$ is the loss from the previous task, and $L(\theta_c)$ is the loss from the current task.

In Eq. 14, the log-likelihood gradient is computed with the model performance, and the gradient squared mean is also computed. In Eq. 15, the loss till the previous task parameters is calculated, and the symbol λ reflects how significant the previous stage parameter is in comparison to the new one. It is evident that the Fisher regularization will attempt to maintain the essential parameters near to the previously learned values. The elastic weighted computation

maintains the current parameter's weights close to the previous parameter's values.

Step 5: Adaptive Neural Network for Drift Adaption

The proposed model employs a GRU, which is a simplified variant of a recurrent neural network. A GRU layer, a hidden layer, and an output layer are the only layers in the initial structure of an adaptive neural network.

When a drift is detected, a new layer is added to the existing model, while the other layers are kept frozen, and the new layer is initialized with the preceding layer's optimal weights. For the GRU, the ReLU activation function is used. ReLU will output the input straight if the input is positive; otherwise, it will output 0. The memory cell captures the drifts automatically.

The sigmoid activation was used for the output layer. Selecting the optimized hyper-parameters is crucial, as it influences the model's performance. In addition, optimized hyper-parameters, such as a learning rate of 0.01, the Adam optimizer, and a binary cross entropy loss function were utilized.

The standard test-then-train (prequential) procedure was utilized to evaluate the data stream. The procedure is carried out by first testing and then training. In the procedure, the dataset is processed incrementally in a sequence of batches. In incremental learning, the catastrophic forgetting issue is also considered.

Step 6: Prediction Result

The final prediction is determined by dividing the total prediction by the number of correct predictions. F1-score, precision, recall, ROC curve, and confusion matrix were also computed to evaluate the performance of the proposed model.

5 Results and discussion

The state-of-the-art methods such as accuracy weighted ensemble(AWE), dynamic weighted majority(DWM), Learn NSE, OzaBagADWIN, hoeffding tree, UDD, and ILSTM were compared with the proposed model adaptive windowing-based recurrent neural network for drift adaptation, and the comparative analysis is provided below.

5.1 Analytical results for proposed model

Sklearn and the Scikit-Multiflow were utilized in the implementation, while the Matplotlib was used to plot the graph (Hunter 2007). The experimental setup was as follows:

To avoid catastrophic forgetting, elastic weight consolidation with a Fisher information matrix was used, which prevents the deviation of the current weight from the previous one. As a result, it retains information from previous tasks while learning new ones. Data samples were handled

in chunks, and the model was trained using incremental learning and evaluated using prequential evaluation. This approach was repeated until the models had used all of the data instances available. Adaptive windowing was combined with drift detection, and an adaptive neural network was used once the drift was discovered. The accuracy, F1-score, precision, recall, confusion matrix, and ROC curve were used to assess the model's performance.

The layers in the initial configuration of a neural network were one GRU layer, one hidden middle layer with 128 neurons, and the output layer with one neuron with sigmoid activation. The learning rate was set to 0.01, the Adam optimizer was used, and binary cross-entropy was used as the loss function.

Experimentation was carried out to determine the best window size by taking into account various window sizes. Several experiments were run with $w = 10, 100, 200, 300, 400,$ and 500 to discover the optimal window size w , which was chosen based on the model's performance and the results.

The Page-Hinckley statistical test was used for the drift detection. The parameters λ and δ in the Page-Hinckley statistical test were set to 50 and 0.005, respectively. λ is the threshold of the change detection and the forgetting factor α , which is used to weigh the observed value and the mean. The threshold value λ is determined by considering the admissible false alarm rate. While increasing threshold value λ reduces false alarms, it also increases the risk of missing or delaying changes. Change detection is a crucial factor in this study; thus, the threshold value is set to 50 to balance the false alarm rate and the risk of missing or delaying changes.

The model was trained with 1, 5, 10, and 20 epochs, and its performance was assessed. If the drift was discovered, the window size was halved. A new layer was added to the existing model, and the model was incrementally trained on the lesser number of data samples from the halved window size. A small window was utilized to detect sudden changes quickly, whereas a larger window efficiently detected gradual changes.

Adaptive windowing based on LSTM recurrent neural network was employed with the aforementioned hyperparameter values, and the results were compared to the adaptive GRU model using synthetic and real-world datasets. The comparative study of adaptive windowing-based recurrent neural networks based on GRU and LSTM is presented in Table 2 and 3.

The results of an experiment on airline and SEA datasets using the proposed model based on adaptive GRU and LSTM were compared and evaluated. Table 2 shows the comparison for the synthetic dataset, and Table 3 shows the comparison of the classifier on the real-world dataset.

The results show that the adaptive windowing-based GRU model outperforms the LSTM model.

The proposed model with one pass performs poorly on synthetic and real-world datasets, as shown in Tables 2 and 3. This is because it scans the data in a single pass and is unable to discover the data's underlying structure, resulting in performance degradation.

The accuracy, F1-score, precision, and recall of the SEA dataset improve as the number of epochs increase until $e = 10$. The difference between $e = 10$ and $e = 20$ is the smallest, as indicated in Table 2.

Table 3 depicts that the accuracy, F1-score, precision, and recall of the airline dataset improve until $e = 10$, after which they remain constant or show the least improvement.

The proposed model's confusion matrix for the SEA and the airline dataset is shown in Figs. 5 and 6, respectively. A confusion matrix is used to demonstrate a classifier's performance. It represents the count of predicted and actual values. True positive values indicate that the classifier correctly classified the positive samples, while true negative values indicate that the classifier correctly classified the negative samples. False negative values indicate that the classifier misclassified actual positive samples as negative, whereas false positive values indicate that the classifier misclassified actual negative samples as positive. The proposed method has better true positive and true negative values as well as lower false negative and false positive rates, as shown by the confusion matrix. The proposed model performs better for airlines and the SEA dataset, as demonstrated by the confusion matrix.

Figures 7 and 8 demonstrate the ROC curves for the proposed model using the SEA dataset and the airline dataset, respectively. Figures 7 and 8 show that the proposed model gained proficient performance, with AUC values of 0.86 and 0.69, respectively. The GRU model has a larger AUC than the LSTM model, indicating that it is better for the SEA and airline datasets.

5.2 Comparative results with other methods

Tables 4 and 5 present the summary of the comparison of different methods applied to the SEA and airline datasets with the proposed model. The tables show that the proposed model outperforms existing state-of-the-art methods in terms of accuracy, F1-score, precision, and recall.

In most real-world applications, data distribution is unbalanced; therefore, F1-score, recall, and precision measures for assessment are more accurate matrices than accuracy.

The accuracy of the classifier on synthetic and real-world datasets was compared and is depicted in Fig. 9. In 9, methods on different datasets are displayed along the x-axis, represented by different colors, while the y-axis represents accuracy. The graph indicates that the proposed model based on GRU can adapt to the different types of drifts more

Table 2 Comparison of the classifier on SEA synthetic dataset

Model		Accuracy	F1-Score	Recall	Precision
GRU					
w = 10	Proposed model with single pass	81.07	83.25	82.94	86.24
	Proposed model with e = 5	82.08	82.79	82.49	86.49
	Proposed model with e = 10	84.41	83.08	82.77	83.37
	Proposed model with e = 20	89.69	88.42	89.27	88.02
w = 100	Proposed model with single pass	76.75	88.07	89.34	87.49
	Proposed model with e = 5	89.76	88.28	89.36	87.49
	Proposed model with e = 10	91.70	88.57	89.68	87.81
	Proposed model with e = 20	91.59	88.49	89.63	87.68
w = 200	Proposed model with single pass	84.05	88.39	89.60	87.62
	Proposed model with e = 5	88.21	88.51	89.65	87.74
	Proposed model with e = 10	90.11	88.52	88.77	87.75
	Proposed model with e = 20	91.09	88.49	88.63	87.68
w = 300	Proposed model with single pass	85.28	88.25	89.48	87.46
	Proposed model with e = 5	88.71	88.33	89.49	87.64
	Proposed model with e = 10	87.32	88.41	89.32	87.89
	Proposed model with e = 20	91.00	88.04	89.54	88.11
w = 400	Proposed model with single pass	83.42	88.39	89.51	87.68
	Proposed model with e = 5	89.66	88.54	89.60	87.79
	Proposed model with e = 10	86.4	88.12	89.4	87.82
	Proposed model with e = 20	91.41	88.45	89.55	87.79
w = 500	Proposed model with single pass	81.90	88.11	89.31	87.28
	Proposed model with e = 5	87.68	87.62	88.85	87.80
	Proposed model with e = 10	91.12	88.15	89.14	87.99
	Proposed model with e = 20	89.78	88.08	89.64	87.91
LSTM					
w = 10	Proposed model with single pass	71.31	88.19	89.31	87.48
	Proposed model with e = 5	82.09	88.42	89.31	87.68
	Proposed model with e = 10	83.15	88.22	89.14	87.94
	Proposed model with e = 20	85.87	88.54	89.57	88.00
w = 100	Proposed model with single pass	86.59	88.08	89.28	87.39
	Proposed model with e = 5	85.01	88.51	89.54	87.88
	Proposed model with e = 10	90.85	88.14	89.24	89.94
	Proposed model with e = 20	87.52	88.51	89.51	87.84
w = 200	Proposed model with single pass	83.55	88.25	89.42	87.37
	Proposed model with e = 5	87.32	88.31	89.32	87.89
	Proposed model with e = 10	88.25	88.11	89.85	87.85
	Proposed model with e = 20	88.58	88.17	89.11	87.94
w = 300	Proposed model with single pass	86.71	88.25	89.51	87.41
	Proposed model with e = 5	89.02	88.12	89.65	87.71
	Proposed model with e = 10	90.03	88.38	89.02	87.88
	Proposed model with e = 20	91.20	87.82	88.82	86.82
w = 400	Proposed model with single pass	84.21	88.14	89.28	87.37
	Proposed model with e = 5	87.63	88.25	89.31	87.85
	Proposed model with e = 10	91.01	87.77	89.94	88.02
	Proposed model with e = 20	91.12	87.82	88.89	88.01
w = 500	Proposed model with single pass	83.74	88.28	89.45	87.51
	Proposed model with e = 5	89.22	88.48	89.44	87.68
	Proposed model with e = 10	89.34	88.12	89.25	87.97
	Proposed model with e = 20	89.54	88.09	89.01	87.97

Significant results are indicated by bold

Table 3 Comparison of the classifier on airline real-world dataset

Model		Accuracy	F1 score	Recall	Precision
GRU					
w = 10	Proposed model with single pass	67.36	41.93	43.25	50.29
	Proposed model with e = 5	67.13	51.06	55.67	54.94
	Proposed model with e = 10	67.36	48.81	54.67	53.53
	Proposed model with e = 20	61.85	50.71	53.46	50.89
w = 100	Proposed model with single pass	67.63	53.43	56.96	56.49
	Proposed model with e = 5	67.41	57.49	62.73	58.16
	Proposed model with e = 10	67.74	59.17	64.42	59.21
	Proposed model with e = 20	62.32	54.32	56.50	54.14
w = 200	Proposed model with single pass	67.03	52.89	57.89	55.57
	Proposed model with e = 5	66.54	59.02	63.39	59.64
	Proposed model with e = 10	65.34	58.85	63.42	59.00
	Proposed model with e = 20	62.61	54.42	57.25	54.39
w = 300	Proposed model with single pass	67.60	55.07	59.89	57.32
	Proposed model with e = 5	67.0	57.92	64.35	58.82
	Proposed model with e = 10	65.8	58.35	64.25	60.42
	Proposed model with e = 20	62.95	59.06	62.35	59.21
w = 400	Proposed model with single pass	67.71	53.32	58.53	56.07
	Proposed model with e = 5	66.24	59.05	63.53	58.82
	Proposed model with e = 10	65.17	58.39	63.14	58.42
	Proposed model with e = 20	64.55	55.14	58.71	55.03
w = 500	Proposed model with single pass	67.43	56.30	60.90	57.90
	Proposed model with e = 5	66.58	58.43	62.83	59.06
	Proposed model with e = 10	65.57	58.09	62.56	58.23
	Proposed model with e = 20	64.53	57.46	58.10	58.10
LSTM					
w = 10	Proposed model with single pass	67.16	41.83	41.03	50.12
	Proposed model with e = 5	67.10	50.59	56.83	54.90
	Proposed model with e = 10	67.73	56.38	60.09	58.31
	Proposed model with e = 20	66.32	58.63	63.24	59.54
w = 100	Proposed model with single pass	67.02	51.96	58.64	55.42
	Proposed model with e = 5	67.41	58.35	63.17	58.99
	Proposed model with e = 10	65.97	58.64	63.32	58.60
	Proposed model with e = 20	66.32	58.63	63.24	59.54
w = 200	Proposed model with single pass	67.13	52.64	57.46	55.89
	Proposed model with e = 5	67.14	41.52	43.78	50.10
	Proposed model with e = 10	67.08	47.94	56.85	53.18
	Proposed model with e = 20	67.15	54.98	61.07	57.03
w = 300	Proposed model with single pass	67.67	53.42	58.17	56.39
	Proposed model with e = 5	66.8	58.75	63.25	59.53
	Proposed model with e = 10	64.61	59.07	63.21	59.42
	Proposed model with e = 20	63.59	58.85	62.57	59.64
w = 400	Proposed model with single pass	67.63	53.03	56.92	56.21
	Proposed model with e = 5	66.93	57.92	64.14	58.89
	Proposed model with e = 10	65.43	57.39	62.64	57.57
	Proposed model with e = 20	64.54	56.96	55.85	57.57
w = 500	Proposed model with single pass	67.57	52.39	60.03	55.82
	Proposed model with e = 5	66.59	58.49	64.15	60.57
	Proposed model with e = 10	65.36	58.07	64.21	59.74
	Proposed model with e = 20	64.78	59.07	63.39	59.03

Significant results are indicated by bold

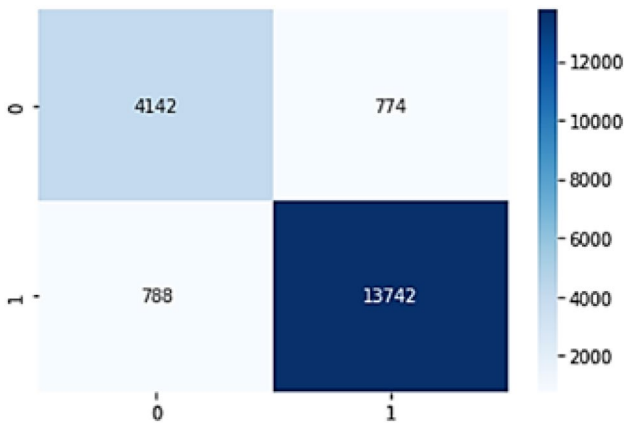


Fig. 5 Confusion matrix for SEA dataset

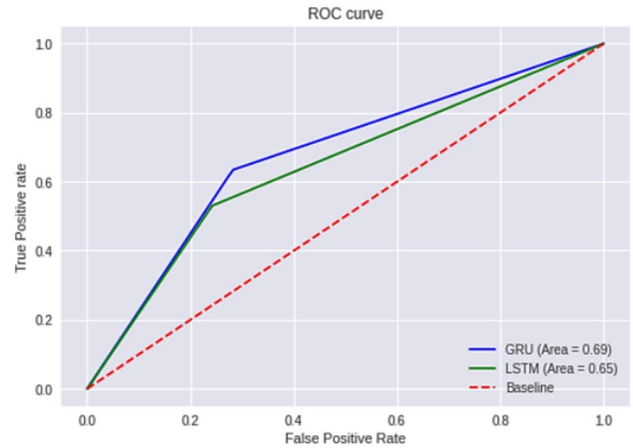


Fig. 8 ROC curve for proposed model with airline Dataset

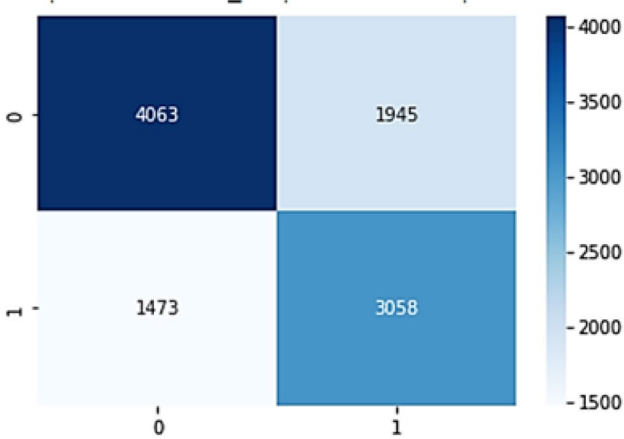


Fig. 6 Confusion matrix for airline dataset

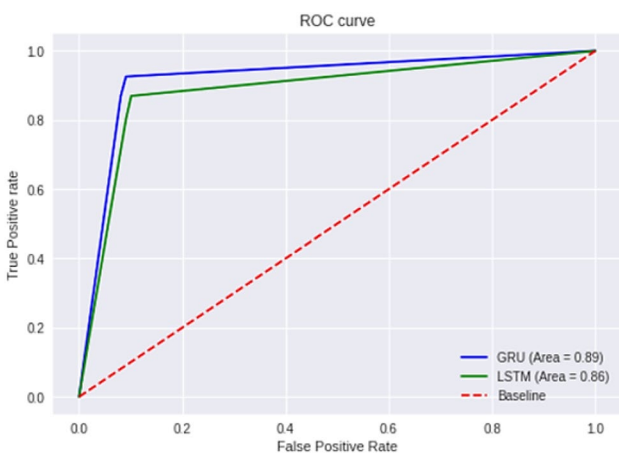


Fig. 7 ROC curve for proposed model with SEA dataset

effectively without forgetting than the other state-of-the-art methods. The proposed model outperformed the classifier by achieving 67.74% and 91.70% accuracy on the airline and the SEA dataset, respectively.

The experiments indicated that adaptive windowing efficiently adjusts the window size if there are drifts or a more volatile data distribution and expands the window size when there are no drifts. Multiple types of drifts occur simultaneously in a SEA synthetic dataset, and state-of-the-art methods fail to adjust to the different types of drift at the same time. The comparative result is provided in Table 4.

The proposed model with an adaptive GRU model with optimized hyper-parameters, forgetting mechanism, and self-adapting window adapted to several types of drifts occurring at the same time.

Figures 10 and 11 compare the proposed model to state-of-the-art methods in terms of F1-score, precision, and recall. Precision demonstrates the proposed model’s ability to correctly identify and predict positive data samples in the presence of a drift. Existing methods, on the other hand, consider the impact of current input data on the classification model, which leads to inaccurate prediction when concept drifts exist.

Recall illustrates the proposed model’s ability to produce less false negatives. In terms of recall, the proposed model exceeded previous methods by 64.42% for airlines and 89.68% for the SEA dataset. F1-score is computed and compared when false negatives and positives are essential and the class distribution in the real-world dataset is unbalanced. The proposed model outperformed existing methods by 59.17% and 88.57% F1-score for the airline and the SEA dataset, respectively.

Higher accuracy and F1-score indicate that incrementally training the adaptive GRU model by automatically selecting the optimal window size for the input data can help to adapt

Table 4 Summary of the different methods applied to the SEA dataset

Method name	Accuracy	Precision	Recall	F1-Score
Learn NSE (Shena and Yuquan Zhub 2017)	86.04	85.96	83.15	84.53
Hoeffding Tree	86.8	88.38	86.42	87.39
OzaBagAdwin (Agustin et al. 2015)	87.25	84.26	83.15	84.23
DWM (Kolter and Maloof 2007; Sidhu and Bhatia 2017; Priya and Uthra 2020)	88.09	86.46	85.30	86.85
AWE (Agustin et al. 2015)	87.68	87.66	86.71	87.46
UDD	88.53	88.79	88.11	87.93
ILSTM (Alvaro et al. 2020)	84.68	86.51	88.45	87.46
Proposed Model	91.70	87.81	89.68	88.57

Significant results are indicated by bold

Table 5 Summary of the different methods applied to the airlines dataset

Method name	Accuracy	Precision	Recall	F1-Score
Learn NSE (Agustin et al. 2015)	59.47	56.79	56.84	56.81
Hoeffding Tree (Nyati et al. 2017)	65.08	58.99	58.24	57.62
OzaBagAdwin (Nyati et al. 2017)	66.06	54.83	52.73	49.83
DWM	65.13	61.16	56.08	54.87
AWE (Nyati et al. 2017)	59.89	51.13	51.23	49.42
UDD	53.09	32.91	54.11	40.89
ILSTM (Alvaro et al. 2020)	64.78	59.08	61.39	59.01
Proposed Model	67.74	59.21	64.42	59.17

Significant results are indicated by bold

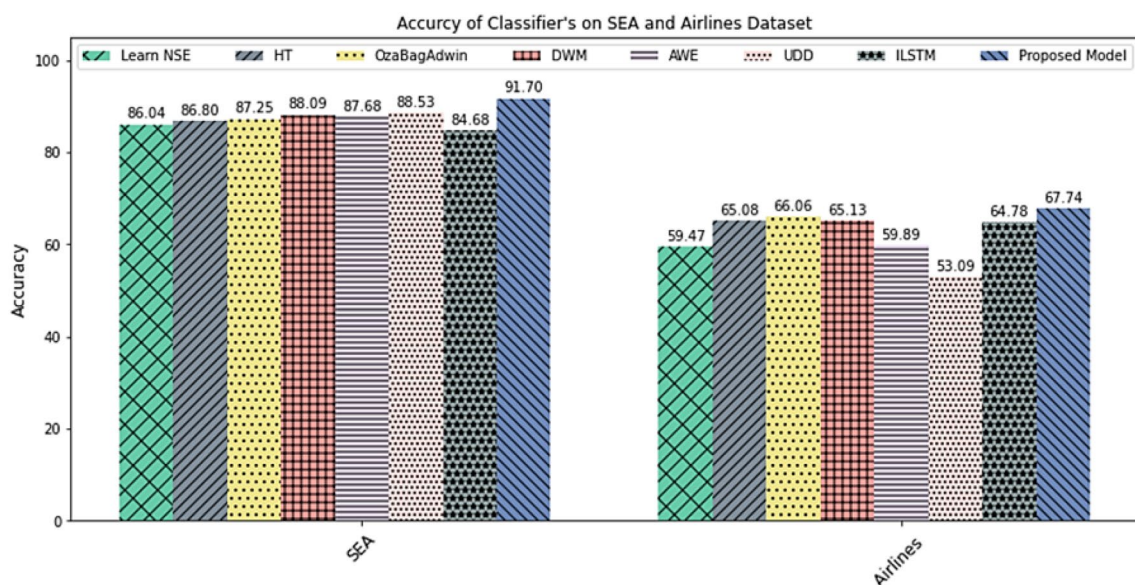
to non-stationary data stream changes and provide consistent performance even when drift occurs.

The results demonstrated that the proposed model can successfully adjust its size in response to drifts or changes and the adaptive neural network adapts to drifts efficiently by taking the catastrophic forgetting effect into account.

6 Conclusion

This study aimed to validate the effectiveness of an adaptive windowing-based recurrent neural network on various types of drifts and compare it with the state-of-the-art methods such as accuracy weighted ensemble, dynamic weighted majority, Learn NSE, OzaBagADWIN, hoeffding tree, UDD, and ILSTM.

The study proposed an adaptive windowing-based recurrent neural network model (based on the GRU model) with a forgetting mechanism to evaluate the sequential relationship and drift adaption effectively. When checking for less fluctuating or stable data distributions or drifts, a wider window

**Fig. 9** Classifier accuracy on a real-world and synthetic dataset

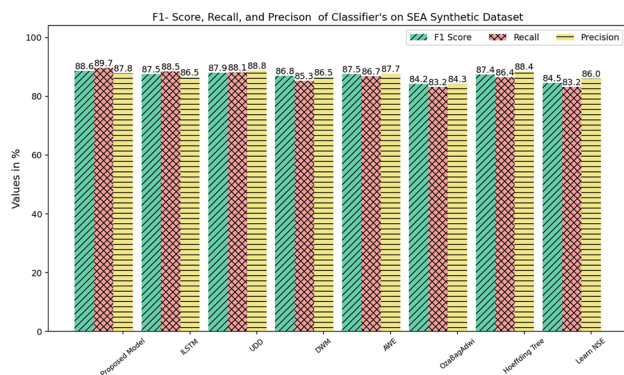


Fig. 10 Performance evaluation with F1 Score, recall and precision of model on SEA synthetic dataset

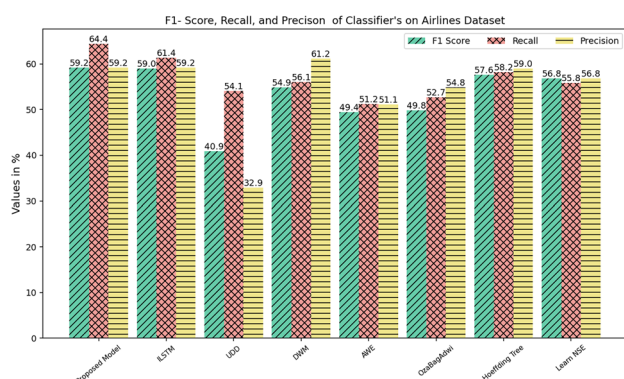


Fig. 11 Performance evaluation with F1 Score, recall and precision of model on real-world dataset

was utilized, but when the data distribution was more volatile or was evolving, a narrower window was employed.

The proposed model's performance was assessed using a real-world airline and a synthetic SEA dataset that included a simultaneous gradual and abrupt drifts. The state-of-the-art methods failed to adjust to several types of drifts occurring at the same time. The observed results revealed that on the airline and the SEA dataset, the proposed model outperforms the state-of-the-art methods by achieving 67.74% and 91.70% accuracy, respectively.

Our findings demonstrate that the adaptive window can successfully adjust its size in response to drifts or changes and the adaptive neural network adapts to the drifts efficiently by taking into account the problem of catastrophic forgetting. In the future, an adaptive ensemble of neural networks could be used to adapt to drifts by considering the unstructured data streams and avoiding forgetting.

Data availability The two datasets that we used, are available online and links are provided in the article.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

- Ayad O (2014) Learning under concept drift with support vector machines. *Lect Notes Comput Sci* 8681:587–594
- Baier L, Schlör T, Schöffel J, Kühl N (2021) Detecting concept drift with neural network model uncertainty. *CoRR* abs/2107.01873. [arXiv:2107.01873](https://arxiv.org/abs/2107.01873)
- Bifet A, Kirkby R (2009) *Data stream mining a practical approach*. Kluwer Academic Publishers, London
- Bifet A, Pfahringer B (2009) Improving adaptive bagging methods for evolving data streams. In: *ACML*, vol 5828 LNAI, pp 23–37. https://doi.org/10.1007/978-3-642-05224-8_4
- Charu A (2015) Chapter 9 A survey of stream classification algorithms. Kluwer Academic Publishers, London
- del Agustín J, Bueno R (2015) Fast adapting ensemble: a new algorithm for mining data streams with concept drift. *Sci World J*. <https://doi.org/10.1155/2015/235810>
- Goutte C, Gaussier E (2005) A probabilistic interpretation of precision, recall and F-score, with implication for evaluation. In: Losada DE, Fernández-Luna JM (eds) *Advances in information retrieval. ECIR 2005*. Lecture notes in computer science, vol 3408. Springer, Berlin. https://doi.org/10.1007/978-3-540-31865-1_25
- Haixun Wang PSY, Wei F, Han J (2003) Mining concept-drifting data streams using ensemble classifiers. In: *Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining (KDD 03)*, pp 226–235
- Hemalatha CS, Pathak R (2019) Hybrid decision trees for data streams based on incremental flexible Naive Bayes prediction at leaf nodes. *Evol Intell*. <https://doi.org/10.1007/s12065-019-00252-3>
- Hochreiter S, Schmidhuber J (1997) Long short-term memory. *Neural Comput* 9(8):1735–1780
- Hoens TR, Polikar C (2012) Learning from streaming data with concept drift and imbalance: an overview. *Mach Learn Knowl Discov Databases*. <https://doi.org/10.1007/s13748-011-0008-0>
- Hunter JD (2007) Matplotlib a 2D graphics environment. *Comput Sci Eng* 9:90–95
- Jadhav A, Deshpande L (2017) An efficient approach to detect concept drifts in data streams. In: *Proceedings—7th IEEE international advanced computing conference. IACC 2017*, pp 28–32. <https://doi.org/10.1109/IACC.2017.0021>
- Jun Gao HZ, Murphey YL (2020) Adaptive window size based deep neural network for driving maneuver prediction. In: *Chinese control and decision conference (CCDC)*, IEEE. <https://doi.org/10.1109/CCDC49329.2020.9163824>
- Junyoung Chung KC, Bengio Y (2014) Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR* abs/1412.3555. [arXiv:1412.3555](https://arxiv.org/abs/1412.3555)
- Kirkpatrick J, Pascanu R, Rabinowitz N, Veness J, Desjardins G, Rusu AA, Milan K, Quan J, Ramalho T, Grabska-Barwinska A, Hassabis D, Clopath C, Kumaran D, Hadsell R (2017) Overcoming catastrophic forgetting in neural networks. *Proc Natl Acad Sci* 114(13):3521–3526. <https://doi.org/10.1073/pnas.1611835114>
- Kolter JZ, Maloof MA (2007) Dynamic weighted majority an ensemble method for drifting concepts. *J Mach Learn Res* 8:2755–2790
- Ksieniewicz P, Woźniak M, Cyganek B, Kasprzak A, Walkowiak K (2019) Data stream classification using active learned neural networks. *Neurocomputing* 353:74–82. <https://doi.org/10.1016/j.neucom.2018.05.130>

- Lee SW, Kim JH, Jun J, Ha JW, Zhang BT (2017) Overcoming catastrophic forgetting by incremental moment matching. In: Proceedings of the 31st international conference on neural information processing systems. Curran Associates Inc., Red Hook, NIPS' 17, pp 4655–4665
- Lemos Neto AC, Coelho RA, de Castro CL (2020) An incremental learning approach using long short-term memory neural networks. *Automática* 2(1):CBA2020. <https://doi.org/10.48011/asba.v2i1.1491>
- Li Z, Hoiem D (2016) Learning without forgetting. CoRR abs/1606.09282. [arXiv:1606.09282](https://arxiv.org/abs/1606.09282)
- Liu Y, Wang Y, Yang X, Zhang L (2017) Short-term travel time prediction by deep learning: a comparison of different LSTM-DNN models. In: 2017 IEEE 20th international conference on intelligent transportation systems (ITSC), pp 1–8. <https://doi.org/10.1109/ITSC.2017.8317886>
- Lu J, Liu A, Dong F, Gu F, Gama J, Zhang G (2018) Learning under concept drift: a review. *IEEE Trans Knowl Data Eng* 31:2346–2363
- Mahdi OA, Pardede E, Cao J (2018) Combination of information entropy and ensemble classification for detecting concept drift in data stream. In: ACM international conference Proceeding series, vol 10, no 1145/3167918, p 3167946
- Nishimura S, Terabe M, Hashimoto K, Mihara K (2008) Learning higher accuracy decision trees from concept drifting data streams. In: Nguyen NT, Borzowski L, Grzech A, Ali M (eds) *New frontiers in applied artificial intelligence. IEA/AIE 2008*. Lecture notes in computer science, vol 5027. Springer, Berlin, Heidelberg, pp 179–188. https://doi.org/10.1007/978-3-540-69052-8_19
- Nyati A, Bhatnagar D, Panwar A (2017) Analyzing performance of classification algorithms on concept drifted data streams. *Int J Comput Appl* 159:13–17
- Oza NC, Russell S (2001) 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, pp 359–364. <https://doi.org/10.1145/502512.502565>
- Pesaranghader A, Viktor HL (2016) Fast hoeffding drift detection method for evolving data streams. *Mach Learn Knowl Discov Databases*. https://doi.org/10.1007/978-3-319-46227-1_7
- Priya S, Uthra RA (2020) Comprehensive analysis for class imbalance data with concept drift using ensemble-based classification. *J Ambient Intell Human Comput* 12:4943–4956
- Rad RH, Haeri MA (2019) Hybrid forest: a concept drift aware data stream mining algorithm. pp 1–25. [arXiv:1902.03609](https://arxiv.org/abs/1902.03609)
- Sebastião R, Fernandes JM (2017) Supporting the page-hinkley test with empirical mode decomposition for change detection. *ISMIS, Springer, lecture notes in computer science*, vol 10352, pp 492–498
- Shena Yan B, Yuquan Zhub JD (2017) A fast learn++ NSE classification algorithm based on weighted moving average. Published by Faculty of Sciences and Mathematics. <https://doi.org/10.2298/FIL1805737S>
- Sheu J-J, Chu K-T, Li N-F, Lee C-C (2017) An efficient incremental learning mechanism for tracking concept drift in spam filtering. *PLoS One* 12(2):e0171518. <https://doi.org/10.1371/journal.pone.0171518>
- Shikha Mehta J (2017) Concept drift in streaming data classification algorithms, platforms and issues. *Procedia Comput Sci* 122:804–811
- Sidhu P, Bhatia MPS (2017) A two ensemble system to handle concept drifting data streams: recurring dynamic weighted majority. Springer, Berlin, p 12
- Song X, He H, Niu S, Gao J (2016) A data streams analysis strategy based on hoeffding tree with concept drift on hadoop system. In: 2016 international conference on advanced cloud and big data (CBD), pp 45–48. <https://doi.org/10.1109/CBD.2016.018>
- Suicheng Gu YT, He X (2013) Recentness biased learning for time series forecasting. *Inf Sci* 237:29–38
- Suryawanshi S, Goswami A, Patil P (2021) Incremental ensemble of one-class classifier for data streams with concept drift adaption. In: *Advanced computing IACC 2020 communications in computer and information science*, vol 1367. Springer, Singapore. https://doi.org/10.1007/978-981-16-0401-0_31
- Wares S, Isaacs J, Elyan E (2019) Data stream mining: methods and challenges for handling concept drift. *SN Appl Sci* 1(11):1–19. <https://doi.org/10.1007/s42452-019-1433-0>
- Yang H, Fong S (2012) Incrementally optimized decision tree for mining imperfect data streams. *Commun Comput Inf Sci* 293(PART 1):281–296. https://doi.org/10.1007/978-3-642-30507-8_25

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.