Applied Intelligence (2022) 52:14164–14177
https://doi.org/10.1007/s10489-022-03308-7

Parallelized extreme learning machine for online data classification

Vidhya M[1] · **Aji S**[1]

Accepted: 26 January 2022 / Published online: 3 March 2022

Abstract
The challenges raised by the massive data are being managed by the community through the advancements of infrastructure and algorithms, and now the processing of fast data is becoming a new hurdle to the researchers. Extreme Learning Machine (ELM) is a single-layer learning model with reliable performances and it is computationally simpler than the new generation deep architectures. ELM process the data in batches and the model has to be rerun while updates happening in the datasets. In the theoretical background of ELM, the past knowledge cannot be reused for improving the performance in online learning where the data set will be updated with mini-batches. In this paper, we have introduced a knowledge base to deal with the remembrance of knowledge in ELM. The architecture of the proposed model is designed to process mini-batches of any size to speed up the processing of the data on its arrival. A group of data sets with different properties such as sparse and feature dimensions is used in the experiments to evaluate our method. The performance of the algorithm is compared with a set of benchmarked classifiers and stream classifiers in the scikit-learn public platform. It is observed that our method could perform better in most of the experiments. It clear in the results that the Parallel ELM model outperformed the other methods in the training time across all the datasets. The consistent performance of our method shows the significance of parallel algorithms of ELM that can remember past knowledge.

Keywords Extreme learning machine · Online machine learning · Stream classifiers · Fast data processing

1 Introduction

The volume of the data is increasing exponentially with the wide acceptance of IoT and as a result, the decision-making entities are modifying their strategies to accommodate the massive growth data for their business needs. The volume of data arriving at every instance of time is to be processed at the arrival itself for structured storage and future use. Handling the fast occurrence of massive data is the present challenge faced by the research and industry community dealing with such kinds of online applications. The future computational algorithms will focus more on the fast processing of data, and the near real-time solutions for very fast data streams are becoming the research target of many groups. The potential of such fast algorithms will range from the security systems used in army operations to the business plans in a consumer shop.

Online data, the live data, processing have a larger potential in smart application those are going to lead us in the coming decade. Autonomous cars [27] are one of the interesting areas where the hardware, sensors and algorithms work together efficiently and faster than the human. The online processing in the earlier computational era is carried out through the batch processing of data, where the huge data volume will be divided into smaller compared to the initial. Those batches will be fed into the computational system for processing on a sequential basis, where the batches will be processed one by one in a pre-determined order. The size of batches and execution time of the process will decide the future of such an application, and such kind of architecture is not preferred in the later stages as well. The advancements in the computing infrastructures and algorithms helped the researchers to use the online data analysis in many applications areas that has published recently, like remote sensing [22, 26], hostile activity analysis [23], cyber forensic [24, 25]. The advancements in machine learning improved the capability of artificial intelligence applications to a great extent. The biologically-inspired strategies, especially the Artificial Neural Network are used in

✉ Vidhya M
vidhya.ela@keralauniversity.ac.in

Aji S
aji@keralauniversity.ac.in

[1] Department of Computer Science, University of Kerala, Thiruvananthapuram, Kerala, India

many computer vision and allied areas. The arrival of deep architecture in learning models has provided a drastic shift in the computational aspects, especially in computer vision. The industry and researchers were deeply impressed by the capability of the model to extract high-level and quality features from the input data using multiple layers. Most of the transformations of data that happened between these layers were complex and required more computational cost in terms of time and infrastructure. Hence the deep architecture is rarely used in applications where these constraints are strictly restricted.

The real-time processing was expensive and complex in its earlier stages, but later the researchers in this area made it lighter and simple. Extreme Learning Machine (ELM) is one of the simplest machine learning model that consists of only one hidden layer, hence the computational complexity is that much reduced. The generalization capability of ELM is theoretically proven and the dimension of output weight in ELM is proportional to the number of neurons in the hidden layer. ELM working with batches of input data and hence there is a single pass computation in the model. The training parameters in the ELM are taken randomly and training output is derived from the connections between the hidden layer and output layer. The performance of ELM is competing and the time taken for the execution is remarkable, hence it is exercised in many machine learning applications particularly in near real-time environments. A recent work came for real-time COVID19 diagnosis [34] using ELM is an example for the same.

The resources utilization in terms of memory and training time of most new-generation algorithms are high, and the real-time or online data processing strategies should have emphasized those aspects. Generally, ELM takes a bundle of data in a single pass for training, and it can be modified for a sequential model where a fixed batch size of data will be fed into the model for processing. In this case, the model must wait for the next batch to be filled, creating a lag in the execution. This work intends to tap the theoretical foundation of ELM and design a model that can effectively manage input batches of any size.

The proposed model is a distributed version of batch processing where the data is divided into mini-batches and given for processing in a different ELM process. Any number of batches can be processed through this architecture at a time within a defined time of execution. Instead of waiting for the slot, a particular batch can directly give for processing on its arrival itself. The output of the parallel application is fixed according to the number of the hidden nodes so that a batch with any size can be given to a process and the integration of the results is a comparatively simple task. The knowledge updating mechanism used in our method helped to fine-tune the knowledge of ELM hence it could improve the results on every arrival of the batches. Datasets from different public sources were used in the experiments of this work. The datasets consist of sparse datasets and other formats with categorical and numerical data. The datasets are also of different dimensions, in their feature sets and number of instances, to check the constancy of the results with various combinations. The performance of the proposed parallel ELM is compared with benchmarked methods in the scikit-learn [12] implementation and stream classifiers in scikit-multiflow [18] framework. Various evaluation metrics like accuracy, f1 score and precision are used to validate the reliability of the proposal. The training time of the proposed method is also compared with all the methods used in the study. Performance of the method with some newly arrived works in stream data classification methods [39–43] is also compiled at the end of the experiment section. It is observed in the experiments that our method outperformed the other methods in terms of the performance metrics. The training time of the proposal is remarkable and shows the relevance of our architecture in the live data classification.

The rest of the paper is organized as follows: Section 2 gives an abstract view of the related works and the other methods used for the study. Section 3 is dedicated to explaining the proposed method and the theoretical model behind the proposal. The experimental setup, results and analysis are included in Section 4. The conclusion of the work is given in Section 5.

2 Related works

The machine learning methods are extensively used in many real-time applications [14, 20, 35, 36, 38] and their improvements helped those applications to become smarter. The general acceptance of Artificial Neural Networks gives confidence to the researchers to do more experiments with ANN and to derive new algorithms. There are a lot of works in the deep architecture of ANN which is found more efficient but the same is computationally complex and time-consuming. The ELM is a single-layer feed forward network derived by Huang et al. [1]. The Moore-Penrose Generalized Inverse and Minimum Norm Least Squares Solution [1] of Learner Systems are the two basic principles behind the working of ELM. The ELM became reputed among the community because of its reliable performance through the Universal approximation [8, 29] principle. Instead of instance-by-instance processing, the ELM process a batch in a single step, and hence the model has to be rerun while appending the datasets. Huang et al. [30] derived another extension of ELM called incremental ELM, where the optimum number of hidden nodes in ELM has derived analytically [31]. Online Sequential ELM (OS-ELM) [32] is emerged to address the drawback of ELM while processing

the batch data. A dual objective method is introduced by them to deal with the data that come as batches in a periodic interval. It is noted in their results that the method is performed better than the other algorithms in the domain. This work is recently updated by Hualong Yu et al. [33] by modifying it to accommodate issues of increasing classes in the upcoming batches. The work of Mirza and Lin [28], named weighted online sequential extreme learning machine, is found suitable for imbalanced class problems.

In the state of art methods, there are several proven strategies in the classification domain which are generally used in online applications. The Stochastic Gradient Descent (SGD) [4] is one among them, in which the internal learning parameters are updated in every iteration from the random mini-batches of the actual dataset. In Radial Basis Function kernel SVM [5], the classical SVM will become powerful with the help of the Radial Basis Kernel that uses the exponent polynomial for the discrimination of classes. Gaussian Process Classifier [6] is a probabilistic classifier that generalizes the Gaussian probability distribution. Random Forest Classifier [7] is an ensemble of uncorrelated decision trees and the net performance of the classifier will be always better. Multi-layer Perceptron is the classical model in learning algorithms and the basis of the new generation classifiers. AdaBoost classifier [9] is an ensemble of a couple of classifiers and it will boost performance through an iterative ensemble method. Gaussian Naive Bayes Classifier [10] is based on the Naive Bayes methods where much importance is given to the conditional independence of the feature sets. Quadratic Discriminant Analysis [11] is used to classify the nonlinear distributions which is an extension of Linear Discriminant Analysis.

The Dynamic Weighted Majority Classifier [13] can be used as a stream classifier which is an ensemble of a couple of learning methods and the classification happens according to the weighted-majority vote mechanism. Online Boosting classifier [15] is a variant of ensemble model in online machine learning methods, where the capability of weak learning algorithms is increased in batch processing. Oza Bagging classifier [16, 17] is a stream classifier which is an extended version of ensemble-based boosting algorithms. In this algorithm, an additional parameter is used to denote the importance of each batch in the online process and it will be updated over time.

3 Proposed method

ELM is a single pass SLFN network that converts the input data into ELM feature space. As explained in the proceeding part of this section, the theory and working of ELM shows that it is computationally less expensive than the other learning models. Figure 1 represents the general architecture of ELM where function f represents the relation 7 and f' is carried out as per relation 5.

There are four main components in the proposed method as shown in Fig. 2– Parallel ELM, Weight Synthesiser, Knowledge Base and Evaluator. The data, either from a live source or available data store, for the training is partitioned into batches and passes to the parallel ELM block. The ELM training modules are tiny in terms of execution complexity and coding, which takes the data chunk and produces the output weight β_i. The outputs of the parallelly executing ELMs are processed in the weight synthesizer and produce the actual output of the run. The corresponding weights are stored, updated, in the knowledge base for the incoming datasets. The evaluator module will evaluate the testing dataset with the help of output weight. The algorithm of the strategy is given in Algorithm 1.

The ELM is formulated on top of two competes [1] - Moore-Penrose Generalized Inverse and Minimum Norm Least Squares Solution of Learner Systems. Any system that can be modelled with a linear relation $Ax = y$ can be effectively solved with the help of Moore-Penrose Generalized Inverse. The matrix $A^{\dagger}_{(q\times p)}$ will be the Moore-Penrose Generalized Inverse of the matrix $A_{(p\times q)}$ under the following situations.

$$\begin{aligned} AA^{\dagger}A &= A, A^{\dagger}AA^{\dagger} = A^{\dagger}, (AA^{\dagger})^T \\ &= AA^{\dagger}, (A^{\dagger}A)^T = A^{\dagger}A \end{aligned} \tag{1}$$

$\hat{x}$ will be a the Least Mean Squares Solution of $Ax = y$ when

$$\|A\hat{x} - y\| = \min_x \|Ax - y\| \tag{2}$$

in Euclidean norm $\|.\|$. In another way, $A^{\dagger}$ will be a minimum norm least squares solution of $Ax = y$. A simple learning system with N samples - $(x_i, t_i), x_i = [x_{i1}, x_{i2}...x_{in}]^T$ and $t_i = [t_{i1}, t_{i2}...t_{im}]$ can be modelled as

$$\sum_{i=1}^{M} \beta_i g(w_i.x_j + b_i) = y_j, j = 1, 2, ...N \tag{3}$$

where M hidden neurons,w_i the initial weight β is the weight connecting to the output layer, b_i is the threshold applied to the corresponding neuron, the $g(x)$ will be the activation that produces the prediction y by satisfying $\sum_{j=a}^{N} \|y_i - t_i\| = 0$ and the relation 3 will become

$$\sum_{i=1}^{M} \beta_i g(w_i.x_j + b_i) = t_j, j = 1, 2, ...N \tag{4}$$

And it can be represented as in the form of leaner system as

$$H\beta = T \tag{5}$$

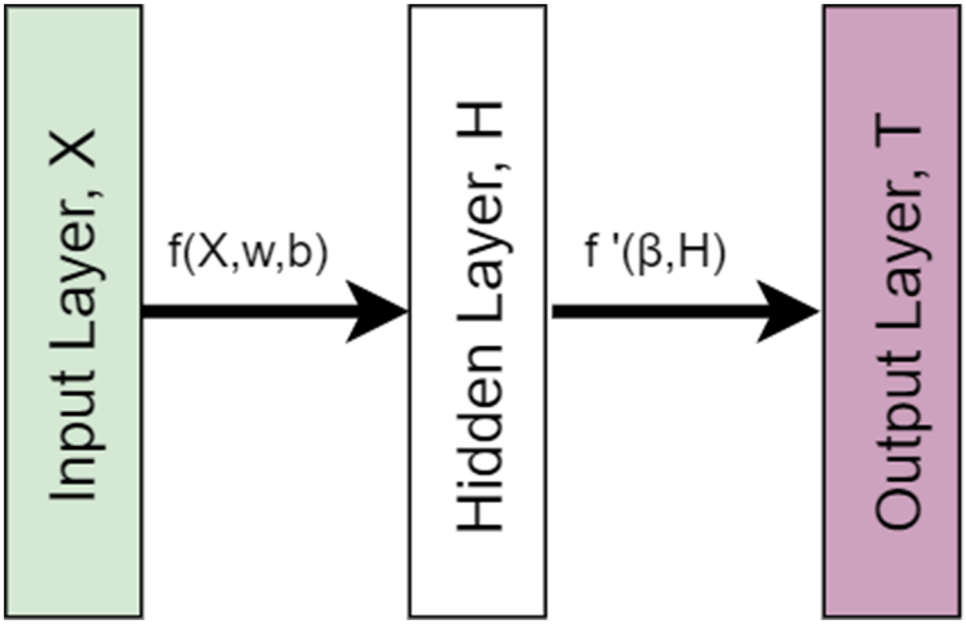

Fig. 1 General Architecture of ELM

As per the minimum norm least squares solution, the output weight β of a single feed forward learning network can be derived as

$$\beta = H^{\dagger} T \tag{6}$$

where H is the Hidden Layer vectors of the input matrix, and it is represented as

$$H = \begin{bmatrix} g(w_1.x_1 + b_1) & \cdots & g(w_N.x_1 + b_N) \\ \vdots & \ddots & \vdots \\ g(w_1.x_N + b_1) & \cdots & g(w_N.x_N + b_N) \end{bmatrix} \tag{7}$$

and

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_N^T \end{bmatrix} and \ T = \begin{bmatrix} T_1^T \\ \vdots \\ T_N^T \end{bmatrix} \tag{8}$$

The relation (6) is used to find the output weight of ELM in the Parallel ELM block of the proposed architecture. The initial value of weight and bias strongly affects the performance of the ELM. In the general implementation of ELM the $\begin{bmatrix} w \\ b \end{bmatrix}$ is assigned randomly. In our method, the primary values of these parameters are derived from the Singular Value Decomposition of the data sample.

$$SVD(d_i) = USV^T \tag{9}$$

where V is an orthogonal matrix, S is a diagonal matrix in which the first r diagonal values are non-zero values and U is a semi-unitary matrix. By the low rank approximation, the first r values of the matrices are enough to represent the initial matrix as

$$d_i \Leftarrow U_r S_r V_r^T \tag{10}$$

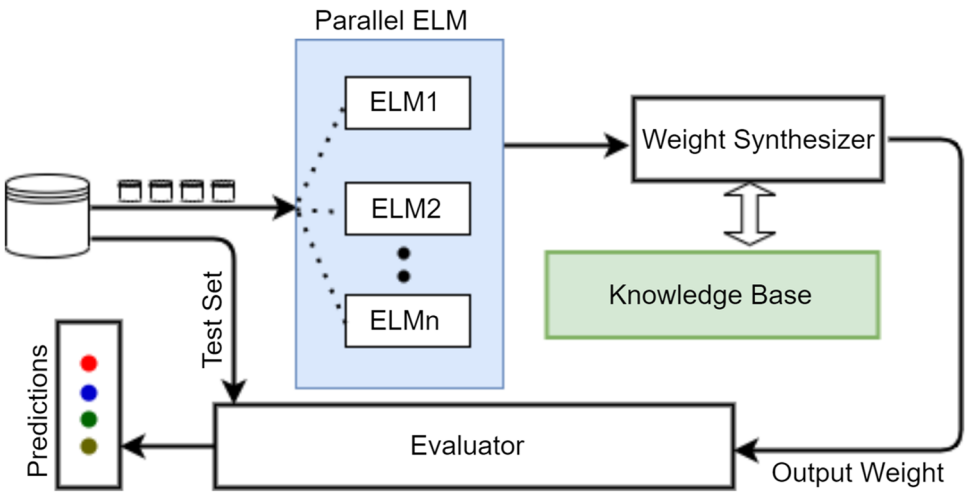

Fig. 2 The Architectural Diagram of proposed method

where U_r and V_r consists of the first r column of the matrices in (8) and S_r be a $r \times r$ square matrix. Hence the initial values of the weight and bias are can be deduced

$$\begin{bmatrix} w \\ b \end{bmatrix} = V_r \tag{11}$$

Because of the inherent representation capability of the matrix V_r, the ELM could perform better than the traditional method.

In Algorithm 1, the steps with a line number 2 are executed by the parallel processes. There are four major steps in it, 2.a. to 2.d., in which the SVD initialization of internal parameters is the first phase. The Hidden Layer matrix and output weights are calculated for each set of batches as in the case of the first phase. The output weight of the particular batch is also stored by the concerned parallel process.

Algorithm 1 Parallel ELM.

Input:
DTrain – The set of input batches of data with fixed length, $DTrain = d_1, d_2, ...d_n$ and $d_i = x_1, x_2...x_N$
TTrain – Target labels of the batches in D,TTrain
$TTrain = t_1, t_2, ...t_n$, $t_i = c_1, c_2...c_p$, t_i can be in p classes
DTest – Set of samples for evaluating the learner, TTrain
$DTest = d_1, d_2, ...d_r$ and $d_i = x_1, x_2...x_N$
TTest – Target labels of testing set $DTest, TTest = t_1, t_2, ...t_r$
$t_i = c_1, c_2...c_p$, t_i can be any one in p classes
g(.) – Activation function
M– Number of hidden neurons
L– length of Knowledge Base
1. Create parallel process for n batches and assign the data
2.a. Initialize the initial weight w_i and bias b_i
2.b. Calculate Hidden Layer matrix, H as per the relation (6)
2.c. Find the output-Weight, β_k with the help of (5)
2.d. Store β_k in Knowledge Base
3. Find the output weight β from KB
4. Update Knowledge Base
5. Find the Hidden Layer matrix for $DTest$ with β (6)
6. Find the predicted Output of Test Samples with the help of (5)

Remembrance of the past knowledge for future improvement is one of the key ideas of major learning models. In the case of ELM, such kind of feedback is technically not possible in its learning activity. We have introduced the additional structure Knowledge Base (KB) for attempting this drawback of ELM. The KB is a fixed-length structure that consists of L weight vectors. The weight vectors of the individual ELM in the parallel processes will be stored in the KB according to the eligibility criteria. The weight that has been derived at each time of evaluation will be fed back to the KB to strengthen the performance of the methodology. The output weight that is indifferent from the existing collection is discarded through mechanisms such as distance measure from the mean of the entire collection and the degree of reliability that the testing data set gives. Hence the vectors that can contribute significantly to processing the incoming data stream are stored in KB. A mean model-based centrality measure is used to find the output weight of the model at an instance of evaluation. Being a simple measure, the weight synthesizing process will not create much computational burden to the model without sacrificing the performance.

The theoretical model reveals that the ELM is computationally efficient for the processing of data on the fly. There is no need to fine-tune the internal parameters, and the major computation is happening with a few steps. By introducing the parallel algorithm architecture, the proposed method can handle any number of batches at a time. Hence the proposed method can further improve the computational cost, comparing with sequential processing. The SVD initialization of the parameters, Hidden Layer matrix and output weight calculations are the steps in the method that can consume computational power but it is negligible in comparison with the new generation learning models.

4 Experiment and results

We have conducted a different set of experiments to evaluate the performance of the proposed strategy. The experiments are categorized into three - i) detecting the influence of batch size in the overall performance, ii) performance comparison with a bunch of benchmarked strategies in classification, iii) a comparative study with the performance of some methods in the domain. The F1 score, Accuracy and Precision are used to quantify the efficiency of the method, while milliseconds are used to measure the execution time of the learner algorithm. A desktop PC with an Intel i5 processor with 6 cores, 2.90GHz speed and an internal memory of 32 GB is used for both coding and execution of the work. We have conducted experiments with hidden nodes of $\min\{BatchSize, FeatureDimension\}$. The model is designed to incorporate any number of parallel processes at a time. In the experiments, we have used two or three parallel processes at a time.

We have used a basket of publically available real-world datasets in these experiments and the details are abstracted in Table 1.

The datasets are selected from different public depositories - the Covtype, Electricity, Airlines and Poker are taken from Massive Online Analysis(MOA) [2]]; Adult, Nomao, RCV1 and Shuttle are from UCI machine learning

Table 1 Datasets used in the experiments

Dataset	Type	Feature Type	Nos.Features	Class	Instances
Covtype	Dense	Categorical, Numeric	54	10	581012
Electricity	Dense	Numeric	8	2	45312
KDD99	Dense	Numeric	41	23	494021
Shuttle	Dense	Numeric	9	7	58000
SEAa	Dense	Numeric	3	2	1000000
AGRa	Dense	Numeric	9	2	1000000
Airlines	Dense	Numeric	7	2	539383
Weather	Dense	Numeric	9	2	18160
Adult	Dense	Categorical, Numeric	14	2	48842
Nomao	Dense	Numeric	120	2	34465
RCV1	Sparse	Numeric	47236	2	111740
Real-sim	Sparse	Numeric	20958	2	72309

repository [3]; Weather and Real-sim are downloaded from OpenML [19]; The SEAa and AGRa [21] are synthetic datasets. Two sparse datasets - RCV1 and Real-sim are also used in the preliminary experiments. We have selected comparatively larger datasets, ranges from 34465 to Ten Lakhs instances, to simulate the online arrival of data by dividing it into batches. The number of feature sets was also one of the criteria for selecting the data sets, and we have a very good range of features with a minimum of 3 and a maximum of 47236. Most of the datasets are binary classes and some of them have more than 10 classes as well. The attributes of the two datasets have both categorical and numerical data and we have converted those categorical data to numerical as a preprocessing phase. As seen in Table 1, we have chosen the datasets such a way to give participation to the maximum parameters that can directly affect the performance of the classification task.

4.1 Batch size and performance of proposed method

The number of samples used in the training activity will generally affect the quality of the performance of the classifiers. The objective of this set of experiments is to verify the same in our proposed method. All the datasets listed in Table 1 are used for the experiments. We have created a stream of batches in different sizes, ranges from 200 to 2000, for the experiment. The datasets are divided into an 80-20 ratio for training and testing purposes.

The classification accuracies of the experiments with 12 datasets are shown in Fig. 3. The training part of the datasets is partitioned into batches of 200, 300, 400, 500, 1000 and 2000 sizes, and conducted the experiments separately for each set. The testing dataset will be fed into the model in a single stretch, and there were samples with more than lakhs of instances in some datasets. The accuracies obtained in all the experiments for the 200 samples were the lowest among all. It is clear in the results that the accuracy was increasing along with the size of the samples. The Airlines dataset got the lowest accuracy of 0.65 among all the datasets. Three datasets- Electricity, KDD99 and Shuttle – were top in the performance and the Electricity marked the best accuracy of 0.998. The other datasets performed in the range of 75% to 95% accuracy.

It is also observed in Fig. 4 that the performance of the model becoming consistent after the batch size of 500. Only two datasets- SEAa and Weather show a simple negative trend after the 1000 batch size. While studying the spread and deviation of the accuracy within every batch size, we have noted that the deviation of the results was very minute and ranges from 0.0251 to 0.00021. The interesting thing is that the standard deviation of the results in the batch sizes 500, 1000 and 2000 were again reducing significantly and coming to the range 6.4x10-3 to 0.18x10-3. For these batches, the negligible inconstancy of the results shown the earlier batches have further reduced more than 50% in seven datasets in the collection. All these statistics show that the method is consistently performing with almost all datasets and batches taken for the study, especially from 500 to 2000.

The proposed method is also compared with a couple of ELM variants-UFROS-ELM [44], EOS-ELM [45] and IDS-ELM [46]. The results obtained are pictured in Fig. 5 – (a) shows the accuracy obtained with these methods against the datasets Electricity, KDD99, and Shuttle. Figure 5(b) gives the training time of these methods with another set of data.

It is noted in Fig. 5(a) that our method performed well with the other variants of ELM; there is a slight upper hand only for the others in some experiments. These competitive results show the importance of our method, where the training process happens with mini-batches. The knowledge base that has been updated in every stage, with new mini-batches, has a significant role in the results produced.

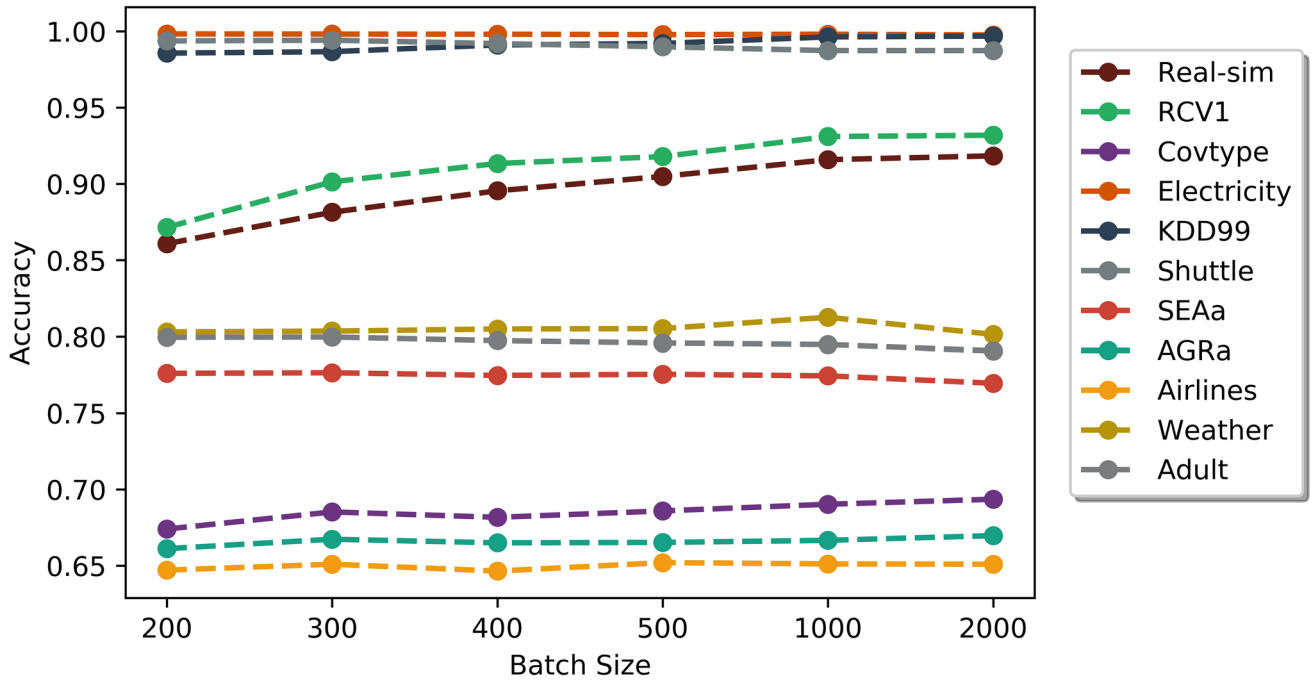

Fig. 3 Performance of proposed method in different datasets and batches

In other ELM variants, the entire training data is taken in a single stretch, but they could not capture a substantial upper hand in the accuracy.

It is evident in Fig. 5(b) that the time taken by the proposed method is significantly low compared with the other ELM variants used for the study. Since the proposed method used mini-batches for the training, the training time recorded the lowest in the group, an essential criterion for the stream data analysis.

4.2 Performance comparison with benchmarked methods

The experiments conducted in this section are to compare the performance of our method with a bunch of benchmarked algorithms in classification. We have chosen the Stochastic Gradient Descent (SGD) [4], Radial Basis Function kernel SVM (RBF SVM)[5], Gaussian Process Classifier (GPC) [6], Random Forest Classifier (RFC)[7],

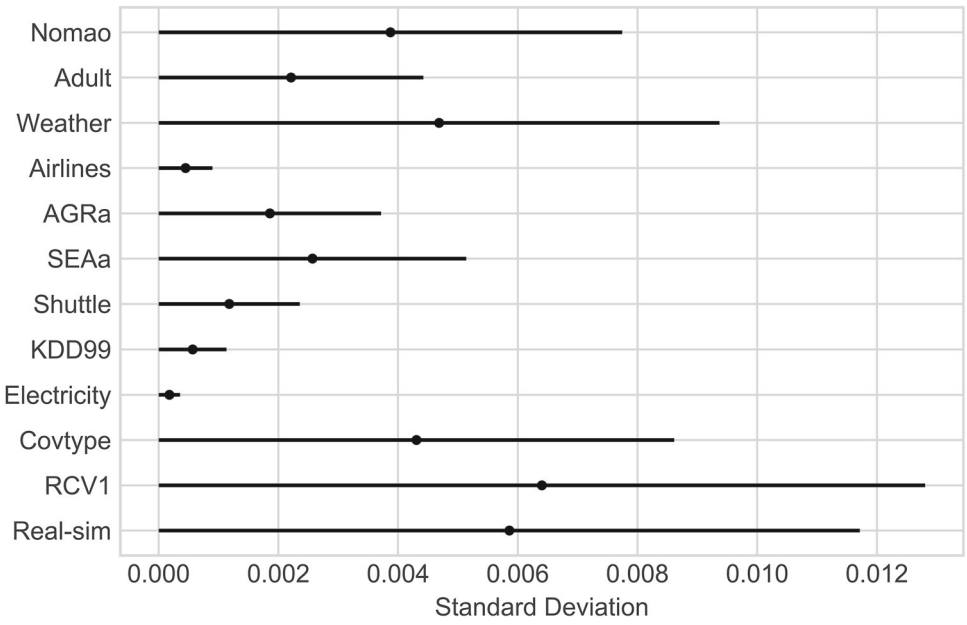

Fig. 4 Standard Deviation of the results for the batch size 500 to 2000

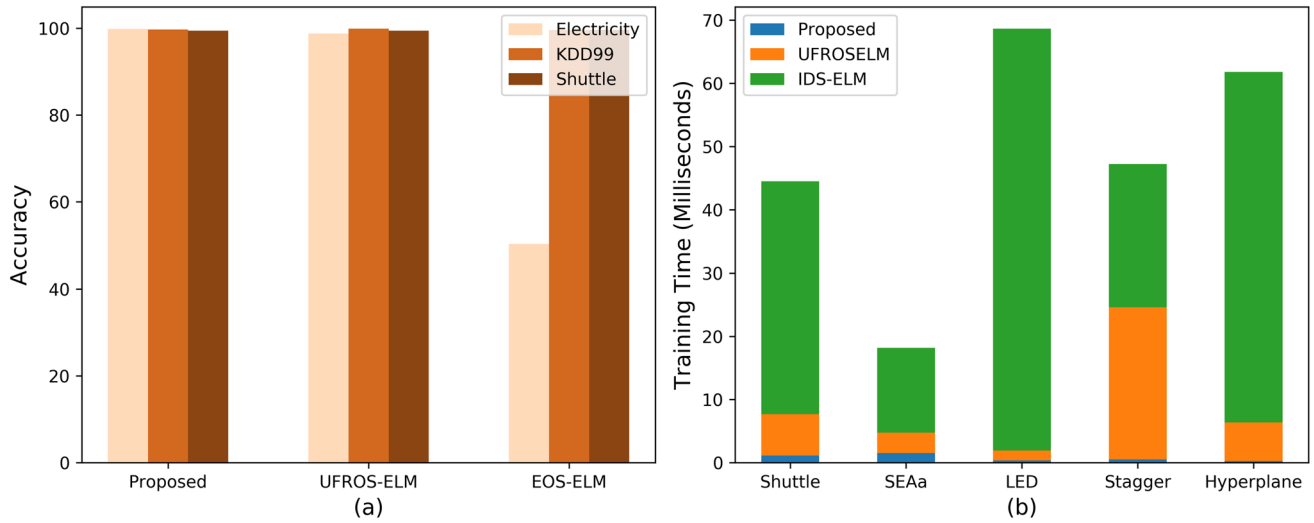

Fig. 5 Performance comparisons of the proposed method with ELM-Variants. (a) Accuracy of the methods (b) Training time of the methods

Multi-layer Perceptron (MLP), AdaBoost classifier (ABC)[9], Gaussian Naive Bayes Classifier (GNB)[10], Quadratic Discriminant Analysis(QDA)[11], and ELM for the study. The scikit-learn [12] implementation of these classifiers is used though out these experiments. Some of the methods are not directly supporting the sparse datasets and hence the RCV1 and Real-sim are not analysed anywhere in this section. The presentation of the results in this section is done in three stages – analyse the effect of mini-batch size in the performance of the classifiers, comparative analysis of the accuracy and F1 score of the methods and the comparative analysis of training time of the classifiers with the proposed method.

As in the case of previous experiments, we have conducted a series of experiments with different mini-batch sizes ranging from 200 to 2000. The first set of experiments in this section is to analyse the influence of the batch size in different classifiers. It is noted that different methods recorded their maximum in various batch sizes.

It is noted that all the batches got represented in Fig. 6. There were ten datasets and ten classifiers; hence a total of 100 combinations were plotted in the scatter plot Fig. 6. It is

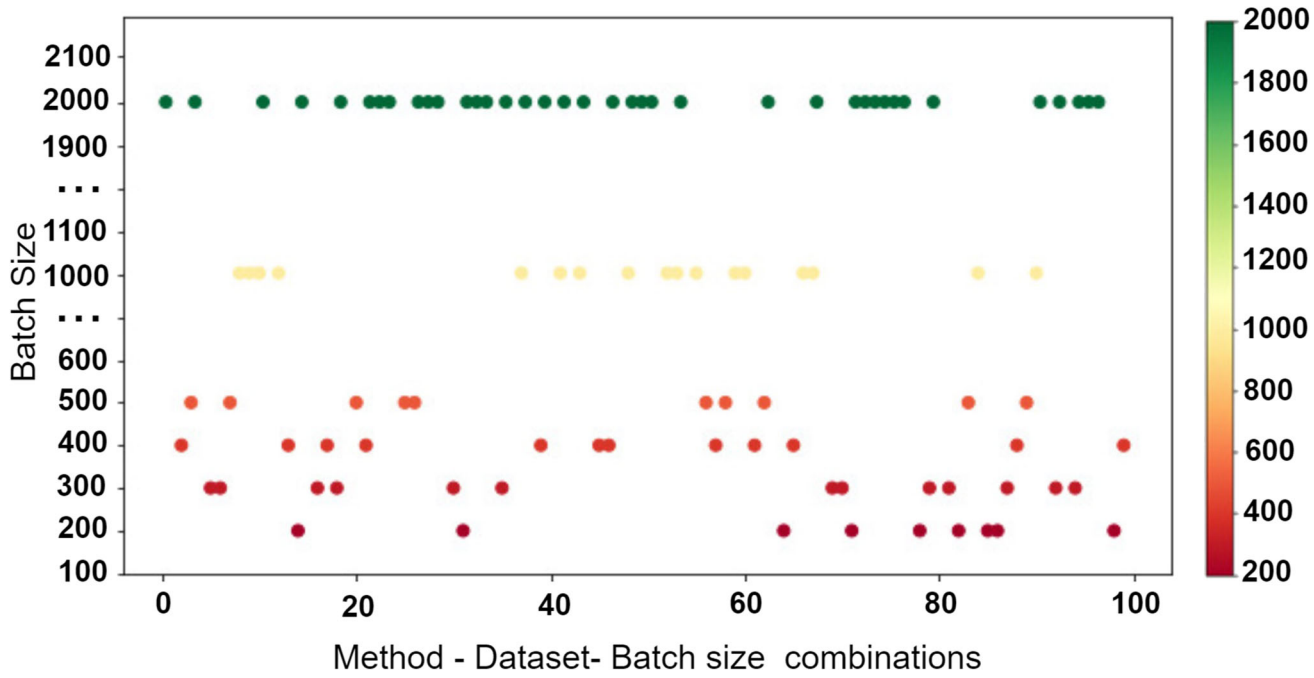

Fig. 6 The size of the top wining batches of the classifiers with various dataset-method combinations

Table 2 Comparison of Accuracy with benchmarked methods

	SGD	RBFSVM	GPC	RFC	MLP	ABC	GNB	QDA	ELM	Proposed
Covtyge	49.96	50.24	67.73	56.16	49.55	55.57	43.88	43.92	57.67	69.36
Electricity	99.3	99.78	99.88	96.49	99.25	99.75	79.01	99.79	99.49	99.82
KDD99	98.06	98.15	98.63	99.19	98.07	79.41	98.8	98.8	97.92	99.68
Shuttle	92.45	95.98	99.68	98.1	92.03	91.31	91.3	95.02	91.91	99.41
SEAa	75.32	75.51	75.57	75.17	75.59	74.45	74.98	75.25	75.37	77.65
AGRa	54.55	57.92	60.7	57.63	58.31	58.47	56.51	61.35	56.99	66.97
Airlines	56.08	55.76	56.37	56.08	55.43	55.76	56.28	57.01	56.35	65.20
Weather	67.81	67.81	79.82	74.2	67.81	76.21	63.82	78.61	78.8	81.26
Adult	79.58	77.83	79.61	80.42	76.32	80.94	79.75	80	79.66	79.97
Nomao	93.28	93.92	94.89	88.95	91.57	94.02	85.09	84.62	85.32	94.21
Avg.Rank	7.15	5.95	2.8	5.45	7.1	6.05	8.05	4.85	6	1.6

noted that some dataset-method combinations for maximum results in the lesser batch sizes. The frequency of top winning combinations in the 200 to 500 batch size is comparatively lower than 1000 and 2000. Even though the mini-batch 2000 has received a higher density of winning hits, it is noted that the best results will not be deeply affected by the batch sizes, and it shows the importance of parallelized strategy using mini-batches.

The average ranking of the methods is also given in Table 2. It is noted that Covtype and Electricity got the lowest and highest accuracies respectively in the entire experiments. In the other methods, the GPC performed well and they are top in two datasets – Electricity and Nomao. NBC was the poor performer in the list and they ranked the least in the four data sets- Covtype, Electricity, Shuttle and Weather. Our method received the best ranking compared with the other nine methods. We could achieve the top position in six datasets and the second position in the other three. The highest accuracy received for our method was with Electricity even though we are in the second position. The lowest accuracy of 65.2 is marked with Airlines, but we could win the top position in that experiment. Our proposed method received an accuracy above 90% with four datasets and only three got below 70%. It is also observed that the performance of our method with the winning method in the concerned experiment is only below 1%. It is happy to view that our method performed better than the initial version of ELM. All these facts show that our method performed better compared with all other benchmarked methods in most of the datasets. The knowledgebase updated in every process of validation helped our model to fine-tune the knowledge for better classification.

It is in Fig. 7 noted that most of the datasets except Covtype, AGRa and AGRa, scored more than 80% precision in at least one of the experiments. Three datasets- Electricity, KDD99 and Shuttle – got more than 99% precision and Electricity got the highest score of 99.88. Our parallel method scored the best precision in four datasets and reached the second potion with another two. The RFC received the highest precisions for Electricity and Nomao datasets, and hence they could reach the second position. It is observed that most of the methods performed well with the datasets Electricity, KDD99 and Shuttle. As seen in the previous set of experiments the proposed methods outperformed in the precision score as well. The consistent and better results of some datasets with most of the methods show the rich features or patterns underlined in those datasets.

It is noted in Fig. 8 that the training time is increasing along with the mini-batch size. There are two distinct groups of graphs in the figure where the training time of the sparse dataset, Real-sim and RCV1, are entirely different from others. The minimum training time for these datasets was 15.01 and 26.12 respectively which is much above the other groups. The other nine datasets in the list have taken only a minimal time for the training that comes in the range from 0.284 to 14.6 milliseconds. While considering the average training time of all the datasets, the Weather performed the best and the RCV1 reached the worst. Considering the earlier group, the variation of time across the batch size dimension is very less in this group. It is cleared in the experiment that both the batch size and feature size influence the training time of our proposed method. The consistent performance of our method in a range of 1.09 to 6.12 milliseconds once again proves the significance of our strategy.

It is cleared in Table 3 that seven methods out of ten need much time for the training in all the datasets where SGD is the only one exceptional case other than ELM and the proposed method. The basic theoretical background of ELM and proposed methods are the same and our method is doing some additional computations as explained in Algorithm 1. Hence the proposed strategy got a slightly higher learning

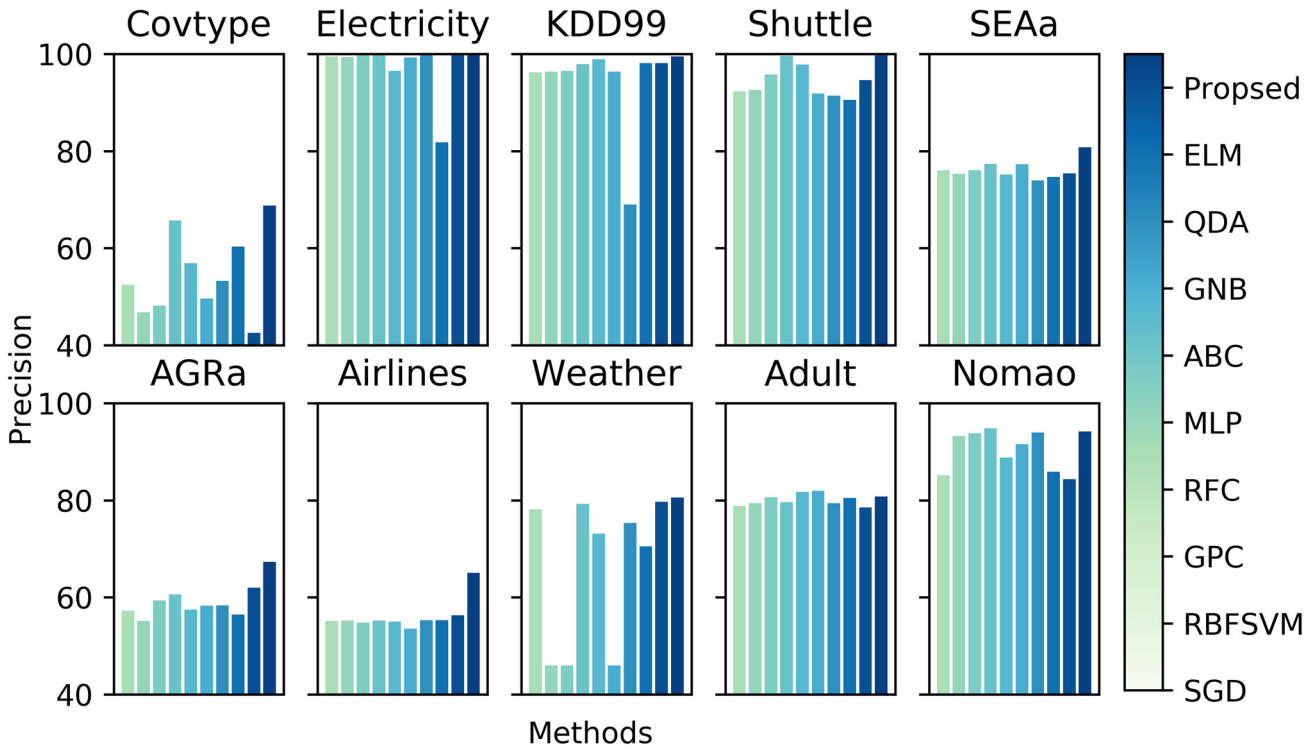

Fig. 7 Precision obtained by the methods for different datasets

time than the ELM which is below 0.1 milliseconds in most cases. The other methods except SGD need more than 100 times the learning time than the ELM versions. The proposed method required below 1 millisecond for completing the training in seven datasets and others have taken only 2.3 seconds and below.

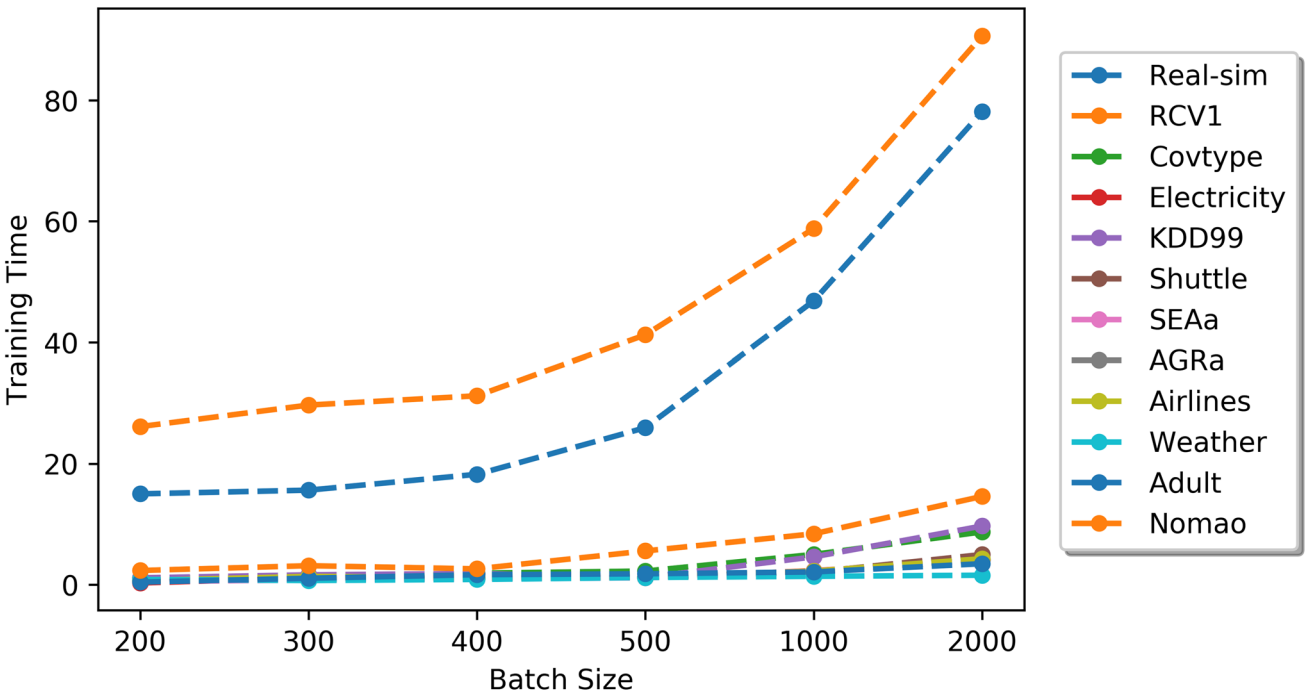

Fig. 8 The time taken (milliseconds) for training for different datasets and batch sizes

Table 3 Training Time (milliseconds) of benchmarked and proposed method

	Covtype	Electricity	KDD99	Shuttle	SEAa	AGRa	Airlines	Weather	Adult	Nomao	Av.Rank
SGD	3.000	2.167	2.667	2.000	2.333	3.000	3.167	2.000	2.333	3.167	3.20
RBFSVM	26.117	1.667	19.133	7.500	3.167	3.333	3.167	3.667	3.667	7.167	4.05
GPC	268.300	26.900	6.100	31.900	120.70	175.5	4.000	2.000	84.80	74.800	5.45
RFC	23441	424.200	57640	36827	375	14662	3361	25503	7139	29601	9.80
MLP	32.900	34.900	34.900	38.900	39.90	43.90	33.90	36.90	37.90	35.900	5.50
ABC	913.600	824.100	1029.7	1342.8	455.80	237.30	423.8	130.7	196.5	1047.2	8.90
GNB	226.400	244.200	164.60	163.60	192.50	231.40	221.4	235.4	252.4	1044.2	7.90
QDA	211.233	189.283	193.98	181.65	170.88	180.03	183.07	180.85	188.7	463.61	7.20
ELM	1.032	0.267	1.039	0.633	0.480	0.615	0.667	0.675	0.105	0.234	1.10
Proposed	1.076	0.284	1.134	0.691	0.530	0.620	0.690	0.692	0.105	2.345	1.90

4.3 Performance comparison with stream classifiers

We have extensively used the scikit-multiflow [18], a dedicated and open source framework for stream data analysis, in the experiments discussed in this section. The performance of the proposed method is compared with four ensemble methods for stream classifiers [37] - Dynamic Weighted Majority Classifier (DWMC)[13], Accuracy Weighted Ensemble Classifier (AWEC), Online Boosting classifier (OBC) [15], Oza Bagging classifier (OzBC)[17]. All the datasets except the sparse datasets are used in the experiments. The accuracy and F1 score of the methods are compared to evaluate the reliability of the proposed method. The training time of the method is also studied in the experiments.

The maximum accuracy in Table 4 of this section, 99.82%, is obtained for the Electricity with parallel ELM. The least accuracy of 50.3 is recorded by the Accuracy Weighted Ensemble Classifier for the dataset Adult. It is observed that the proposed method got the highest accuracy for six datasets and in the meantime, it has placed in the last position for the other two. Our method received more than 90% accuracy for four datasets and reached the first position in six datasets. It is noted in the results that all the methods performed consistently with four datasets in the list. Among the other stream classifiers, DWC and Online Boosting classifier performed almost the same pattern and shared the second position in the average ranking. As in the case of other experiments, the overall performance of our parallel ELM performed better compared with the stream-based classifiers as well.

Table 4 Comparison of Accuracy with stream methods

	DWMC	AWEC	OBC	OzBC	Proposed
Covtype	81.75	64.35	85.5	88.67	69.36
Electricity	83.45	77.2	77.85	76.9	99.82
KDD99	98.6	56.15	99.7	97.97	99.68
Shuttle	92.1	94.9	97	96.37	99.41
SEAa	87.5	87.85	79.95	86.17	77.65
AGRa	79	82.15	71.35	74.43	66.97
Airlines	63	59.65	60.3	64.53	65.2
Weather	71.35	71.4	75.3	77.3	81.26
Adult	75.15	50.3	72.95	75.05	79.97
Nomao	79.95	88.5	91.05	86.9	94.21
Avg. Rank	2.857	3.571	2.857	3.000	2.714

The F1 score achieved by the different streaming classifiers and the proposed methods for the ten datasets is abstracted in Fig. 9. A score of 90% and more has been recorded in six experiments in the pool and four of them were scored by the proposed method. As the extension of the indication of Table 4, the parallel ELM outperformed the stream classifiers for the F1 matrix.

It is seen in Table 5 that the parallel ELM could complete the training earlier than the other stream-based classifiers for all the data sets. The DWMC performed comparatively better than the other three methods and in most of the data sets, it has recorded the training time near to the proposed strategy. A proportional variation of training time with respect to the number of features and batch size is visible throughout our method. Such kind of pattern is not visible in most of the classifiers in this category. The comparative study in the table strengthens the inferences derived in the previous sections and experiments.

Our proposed method obtained maximum accuracy of 99.82%, 99.68% in Electricity and KDD99 datasets. The Pokerhand, Stagger, LED datasets obtained third and the fourth position. Hyperplane obtained the fifth position. Covtype, Airline datasets got the lowest accuracy of all. It is clear from the table, the training time of our proposed method is comparatively low than the other existing methods.

Table 6 compares the proposed method with some ensemble-based learning algorithms that deal with stream data classification. Datasets that have some reported outputs with the selected methods are taken for the compilation of this

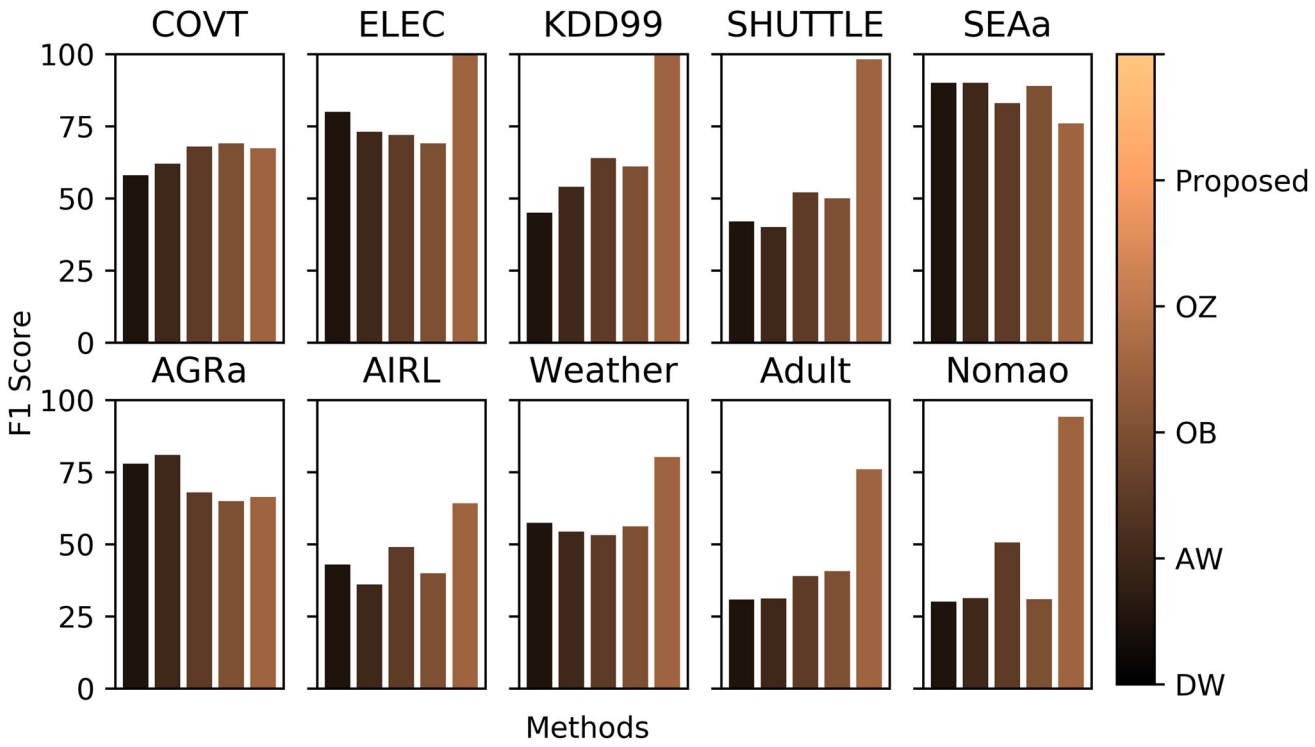

Fig. 9 The F1 Score of obtained for the datasets by the stream classifiers and proposed method

Table 5 Training Time (milliseconds) of stream classifiers and proposed method with 2000 batch mini size

	DWMC	AWEC	OBC	OzBC	Proposed
Covtype	9.500	11.845	742.129	142.912	1.076
Electricity	1.010	0.903	574.437	26.473	0.284
KDD99	4.127	8.088	1643.593	260.920	1.134
Shuttle	1.841	1.548	586.249	31.928	0.691
SEAa	0.586	0.565	580.006	14.878	0.530
AGRa	0.646	1.059	690.716	30.980	0.620
Airlines	1.558	0.836	595.261	24.049	0.690
Weather	1.115	0.962	619.228	25.671	0.692
Adult	1.557	1.575	751.515	48.316	0.105
Nomao	4.505	8.750	1348.254	265.967	2.345

Table 6 A comparative analysis with existing methods in stream data classification. The training Time(TT) is given in milliseconds

	CALMID[39]		DUE[40]		AWDOB[41]		OzaNB[42]		Proposed	
	Acc.	TT	Acc.	TT	Acc.	TT	Acc.	TT	Acc.	TT
KDD99	76.38	28.14	81.36	30.53	70.36	26.68	94.6	1650	99.68	1.134
SEAa	64.47	24.05	72.43	20.03	66.45	38.45	85.4	243	77.65	1.53
Covtype	75.29	13.48	72.49	26.48	64.56	36.17	87.1	9532	69.36	1.076
Electricity	–	–	–	–	–	–	73.4	15.8	99.82	0.284
Airlines	73.29	60.48	76.35	82.47	70.32	84.26	64.4	132	65.2	0.69
LED	72.37	15.68	66.49	13.68	73.54	17.98	69.8	478	74.12	0.374
Stagger	66.58	33.45	70.33	22.26	62.48	9.16	60.2	86.1	70.36	0.541
Hyperplane	72.53	23.45	63.24	57.23	66.32	35.46	–	–	73.69	0.254
Pokerhand	62.42	28.35	71.37	16.67	64.56	30.23	–	–	80.58	0.356

table. The time and accuracy of each dataset are compared and found that our method came first in six among the nine datasets. In the case of training time, our method outperformed all these stream-based classifies. These results show the significance of the single-pass, knowledge-based parallel methods in online data classification.

The results of all the experiments reveal that the strategy followed in the proposed methods is efficient in terms of training time and reliable performance in most of the data sets. The underlined principle of ELM and the mean model-based knowledge base are helped the model for producing a consistent result. Since the learning strategy is almost stable in different batch sizes, the performance of parallel implementation was steady and it could improve the execution time effectively.

5 Conclusion

This paper exploited the theory of ELM and the working principle of parallel algorithms for modelling an efficient architecture for online data classifications. The introduction of a knowledge base and output weight processing module improved the initial version of ELM. We have divided the data sets into different mini-batches for the implementation of parallel ELM. We have conducted a set of extensive experiments to validate our architecture. The data sets with various properties are selected from different public sources for the experiments. The performance of a group of benchmarked classifiers and stream-based methods are compared and analysed in a collection of experiments. It is observed in the results that our method outperformed the other strategies in all validation matrices though out the experiments. The training time of parallel ELM is considerably reduced in all the experiments without sacrificing much in the reliability of the results. In the future, the results can be further improved by fine-tuning the knowledge updating process of the knowledge-based module.

References

1. Huang GB, Zhu QY, Siew CK (2004) Extreme learning machine: a new learning scheme of feedforward neural networks. In: 2004 IEEE international joint conference on neural networks (IEEE Cat. No. 04CH37541), vol 2. IEEE, pp 985–990
2. Bifet A, Holmes G, Pfahringer B, Kranen P, Kremer H, Jansen T, Seidl T (2010) Moa: Massive online analysis, a framework for stream classification and clustering. In: Proceedings of the first workshop on applications of pattern analysis. PMLR, pp 44–50
3. Dua D, Graff C (2017) UCI machine learning repository
4. Bottou L (2010) Large-scale machine learning with stochastic gradient descent. In: Proceedings of COMPSTAT'2010. Physica-Verlag HD, pp 177–186
5. Chang YW, Hsieh CJ, Chang KW, Ringgaard M, Lin CJ (2010) Training and testing low-degree polynomial data mappings via linear SVM. J Mach Learn Res, 11(4)
6. Yang X (2020) Introduction to stochastic calculus and its applications. Available at SSRN 3607647
7. Ho TK (1995) Random decision forests. In: Proceedings of 3rd international conference on document analysis and recognition, vol 1. IEEE, pp 278–282
8. Hastie T, Tibshirani R, Friedman J (2009) The elements of statistical learning: data mining, inference, and prediction. Springer Science & Business Media, Berlin
9. Kégl B (2013) The return of AdaBoost. MH: multi-class Hamming trees. arXiv:1312.6086
10. John GH, Langley P (2013) Estimating continuous distributions in Bayesian classifiers. arXiv:1302.4964
11. Tharwat A (2016) Linear vs. quadratic discriminant analysis classifier: a tutorial. Int J Appl Pattern Recognit 3(2):145–180
12. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B et al (2011) Scikit-learn: Machine learning in Python. J Machine Learn Res 12:2825–2830
13. Kolter JZ, Maloof MA (2007) Dynamic weighted majority: an ensemble method for drifting concepts. J Machine Learn Res 8:2755–2790
14. Wang R, Chow CY, Lyu Y, Lee VC, Kwong S, Li Y, Zeng J (2017) Taxirec: recommending road clusters to taxi drivers using ranking-based extreme learning machines. IEEE Trans Knowl Data Eng 30(3):585–598
15. Wang H, Fan W, Yu PS, 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, pp 226–235
16. Wang B, Pineau J (2016) Online bagging and boosting for imbalanced data streams. IEEE Trans Knowl Data Eng 28(12):3353–3366
17. Oza NC, Russell SJ (2001) Online bagging and boosting. In: International workshop on artificial intelligence and statistics. PMLR, pp 229–236
18. Montiel J, Read J, Bifet A, Abdessalem T (2018) Scikit-multiflow: A multi-output streaming framework. J Machine Learn Res 19(1):2915–2914
19. Vanschoren J, Van Rijn JN, Bischl B, Torgo L (2014) OpenML: networked science in machine learning. ACM SIGKDD Explorations Newsletter 15(2):49–60. https://doi.org/10.1145/2641190.2641198
20. Duan J, Ou Y, Hu J, Wang Z, Jin S, Xu C (2017) Fast and stable learning of dynamical systems based on extreme learning machine. IEEE Trans Syst Man Cybern Syst 49(6):1175–1185
21. Gomes HM, Bifet A, Read J, Barddal JP, Enembreck F, Pfharinger B, Holmes G, Abdessalem T (2017) Adaptive random forests for evolving data stream classification. Mach Learn 106(9):1469–1495
22. Kumar S, Banerjee B, Chaudhuri S (2021) Improved landcover classification using online spectral data hallucination. Neurocomputing 439:316–326
23. Dadkhah S, Shoeleh F, Yadollahi MM, Zhang X, Ghorbani AA (2021) A real-time hostile activities analyses and detection system. Applied Soft Computing 104:107175
24. Seraphim BI, Poovammal E (2021) Adversarial attack by inducing drift in streaming data. Wirel Pers Commun, 1–25
25. Li K, Luo G, Ye Y, Li W, Ji S, Cai Z (2020) Adversarial Privacy Preserving Graph Embedding against Inference Attack. IEEE Internet of Things Journal
26. Dong Y, Yang C, Zhang Y (2021) Deep metric learning with online hard mining for hyperspectral classification. Remote Sens 13(7):1368

27. Jo K, Kim J, Kim D, Jang C, Sunwoo M (2014) Development of autonomous car—Part I: Distributed system architecture and development process. IEEE Trans Ind Electron 61(12):7131–7140
28. Mirza B, Lin Z, Toh KA (2013) Weighted online sequential extreme learning machine for class imbalance learning. Neural Process Lett 38(3):465–486
29. Huang GB, Chen L, Siew CK (2006) Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Trans Neural Netw 17(4):879–892
30. Huang GB, Chen L (2008) Enhanced random search based incremental extreme learning machine. Neurocomputing 71(16-18):3460–3468
31. Huang GB, Chen L, Siew CK (2006) Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Trans Neural Netw 17(4):879–892
32. Liang NY, Huang GB, Saratchandran P, Sundararajan N (2006) A fast and accurate online sequential learning algorithm for feedforward networks. IEEE Trans Neural Netw 17(6):1411–1423
33. Yu H, Xie H, Yang X, Zou H, Gao S (2021) Online sequential extreme learning machine with the increased classes. Comput Electric Eng 90:107008
34. Wu C, Khishe M, Mohammadi M, Karim SHT, Rashid TA (2021) Evolving deep convolutional neutral network by hybrid sine–cosine and extreme learning machine for real-time COVID19 diagnosis from X-ray images. Soft Comput, 1–20
35. Rathore S, Park JH (2018) Semi-supervised learning based distributed attack detection framework for IoT. Appl Soft Comput 72:79–89
36. Zhang X, He T, Lu L, Yue S, Cheng D, Xu X (2017) Video analysis of traffic accidents based on projection extreme learning machine. In: 2017 international symposium on intelligent signal processing and communication systems (ISPACS). IEEE, pp 149–154
37. Ghomeshi H, Gaber MM, Kovalchuk Y (2020) A non-canonical hybrid metaheuristic approach to adaptive data stream classification. Futur Gener Comput Syst 102:127–139
38. Ghomeshi H, Gaber MM, Kovalchuk Y (2019) EACD: Evolutionary Adaptation to concept drifts in data streams. Data Min Knowl Disc 33(3):663–694
39. Liu W, Zhang H, Ding Z, Liu Q, Zhu C (2021) A comprehensive active learning method for multiclass imbalanced data streams with concept drift. Knowledge-Based Systems 215:106778
40. Li Z, Huang W, Xiong Y, Ren S, Zhu T (2020) Incremental learning imbalanced data streams with concept drift: The dynamic updated ensemble algorithm. Knowledge-Based Systems 195:105694
41. Baidari I, Honnikoll N (2020) Accuracy weighted diversity-based online boosting. Expert Systems with Applications 160:113723
42. Sarnovsky M, Kolarik M (2021) Classification of the drifting data streams using heterogeneous diversified dynamic class-weighted ensemble. PeerJ Computer Science 7:e459
43. Museba T, Nelwamondo F, Ouahada K, Akinola A (2021) Recurrent adaptive classifier ensemble for handling recurring concept drifts. Applied Computational Intelligence and Soft Computing, 2021
44. Aydogdu O, Ekinci M (2020) A new approach for data stream classification: unsupervised feature representational online sequential extreme learning machine. Multimed Tools Appl 79(37):27205–27227
45. Lan Y, Soh YC, Huang GB (2009) Ensemble of online sequential extreme learning machine. Neurocomputing 72(13-15):3391–3395
46. Xu S, Wang J (2016) A fast incremental extreme learning machine algorithm for data streams classification. Expert Syst Appl 65:332–344