

Received 3 August 2023, accepted 19 August 2023, date of publication 22 August 2023, date of current version 25 August 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3307512

RESEARCH ARTICLE

CTELC: A Constant-Time Ensemble Learning Classifier Based on KNN for Big Data

AHMAD S. TARAWNEH^{1,3}, EMAN S. ALAMRI², NAJAH NOORI AL-SAEEDI³,
MOHAMMAD ALAUTHMAN⁴, AND AHMAD B. HASSANAT³, (Senior Member, IEEE)

¹Department of Data Science and Artificial Intelligence, University of Petra, Amman 114587, Jordan

²Food Science and Nutrition Department, University of Tabuk, Tabuk 47512, Saudi Arabia

³Faculty of Information Technology, Mutah University, Kerak 61710, Jordan

⁴Department of Information Security, University of Petra, Amman 114587, Jordan

Corresponding authors: Eman S. Alamri (ialamri@ut.edu.sa) and Ahmad B. Hassanat (Hasanat@mutah.edu.jo)


ABSTRACT Big data classification is a challenging task because most known classification methods need a long time and a lot of processing resources to execute such a task and use the vast amount of available data. In this paper, we propose a novel big data classification method that leverages the power of the KNN classifier and the efficiency of the ensemble learning technique to create a new method capable of performing classification tasks on big data efficiently. The proposed method picks tiny data chunks at random from a big dataset, with each chunk including random examples of a small number of randomly selected features. A weak KNN classifier is employed on each data chunk to perform classification on new (unseen) data, and the majority voting rule is used to reach the final classification decision based on the outcomes of the weak classifiers. The proposed method has a **constant** classification time, according to the time complexity analysis. Furthermore, the proposed method was found to be more efficient on a single node than existing methods, some of which run on a large cluster of nodes. Because of its speed and enhanced performance, the proposed method can be considered an ideal classifier for handling complex data types such as Geospatial data, Big trajectory data, and Big Data in general.

INDEX TERMS Big data, geospatial data, trajectory data, classification, ensemble learning, KNN.

I. INTRODUCTION

For decades, researchers have been studying and evaluating big data methodologies and tools. This interest stems from the massive volume of data exchanged and saved by social media users, medical organizations, educational institutions, and others.

According to statistical reports, the number of users on different social media platforms has reached more than 2 billion [1]. WhatsApp, for example, has over 600 million users, more than half a billion photos, and one hundred million videos transferred and shared between users on a daily basis [2]. Also, due to the huge advance in smartphone technology, it has become easier for users to share text/images and write posts on such social media platforms. Some reports

The associate editor coordinating the review of this manuscript and approving it for publication was Ines Domingues .

show that the number of posts on Twitter in 2007 was 5K, this number became around 500 million after about 6 years, in 2013 [3], which indicates the massive amount of available data on social media in general. This amount of data is not restricted to social media, as many other platforms generate and store huge data volumes [4], [5], [6], [7].

This amount of data needs to be processed and analyzed in order to use it for building useful knowledge discovery and machine learning big data-based applications, like facial big data applications [8], [9], [10], signal big data [11] and various industry big data-based applications [12], [13], [14].

Volume (big), Variety, and Velocity are among the most distinguishing characteristics of Big data, and as a result, it is attractive to have an efficient classification/prediction system to learn from such Big data. Such applications include, but are not limited to, medical [15], [16], financial [17], [18], [19],

Security [20], [21] and image-based applications [22], [23], [24], [25].

The fundamental difficulty with big data classification is that most well-known powerful models, such as support vector machines (SVM), artificial neural networks (ANN), and decision trees, cannot be employed for big data classification because of the very long time necessary to train these models.

K-nearest neighbors (KNN) is one of the simplest and most efficient classification algorithms, which might produce performance comparable to other more popular classifiers like ANN and SVM [26], [27] for some tasks.

However, like other classifiers, KNN suffers from its time complexity which is $O(nd)$ where n is the number of examples and d is the number of features (dimensions). The typical KNN is challenging to employ for big data classification due to its time complexity, as it is called a lazy classifier because no learning model is generated, and so a test example must be compared to the entire big data set to infer its proper class.

Assuming that we have a test point x , S_x is a set of the K nearest examples, which can be expressed as $S_x \subseteq D$ s.t. $|S_x| = K$ and $\forall(x', y') \in D \setminus S_x, \text{dist}(x, x') \geq \max_{(x'', y'') \in S_x} \text{dist}(x'', y'')$, where dist is a distance metric. i.e. each example in D but not in S_x is at least as far away from x as the furthest example in S_x . The KNN model $mdl()$ is a function which provides the most frequent label in S_x : $mdl(x) = \text{mode}(\{y'' : (x'', y'') \in S_x\})$. This should be performed for all the testing examples using a distance metric. Minkowski distance for example is calculated as follows:

$$MD(x, x') = \left(\sum_{i=1}^d |x_i - x'_i|^p \right)^{\frac{1}{p}} \quad (1)$$

where x and x' are feature vectors of equal number of dimensions d .

Therefore, $O(nd)$ is the time complexity of finding the nearest neighbor to a testing example. If one wants to find the nearest example for each example in D , the time complexity grows to be $O(ndD)$, which is not practical even for medium-volume datasets.

This typical KNN can be approximated using less time complexity algorithms such as the KD tree, which has a query time complexity of $O(\log n)$. However, such approximation requires constructing a model (tree) on the training data, which is not the case in the standard KNN, and KD tree model construction consumes $O(n \log^2 n)$ [28].

Other available similar methods are more or less expensive including the work of Hassanat [29], [30], [31], [32], which is based on creating a binary tree to speed up the KNN, reducing its testing time to be logarithmic, however, we still need a linear time (at best) for creating these binary search trees.

To the best of our knowledge, all of the well-known big data classification approaches require time for testing and/or training, which is proportional to the size of the big data set. Given the huge volume and variety of big data, it would be tempting to develop a new classifier that consumes a constant amount of time regardless of the size of the big dataset.

In light of the central limit theorem, would a limited number of random sub-samples of instances suffice for training a big data set rather than using all of the big data set examples?

This paper attempts to answer this question by introducing a new ensemble classifier based on weak KNN classifiers, each of which works on a sub-sample of big data examples with replacement. This type of approximated KNN is designed to make the classification process run efficiently on a fixed (constant) number of sub-samples of big data, resulting in a constant-time classifier-the aim of this paper.

Unlike existing KNN approximation approaches, the proposed method is based solely on the constant predetermined size and number of sub-samples used, requires no training, and therefore, runs in constant time.

The rest of this paper is organized as follows, the second section reviews the literature on big data classification approaches, the third section presents the proposed method, the fourth section describes the big datasets used for evaluating the proposed classifier as well as the experimental design, and the fifth section reports the results and comparisons to the state-of-the-art big data classification methods.

II. LITERATURE REVIEW

The KNN classifier has been a temptation for academics to speed up Big data classification because it is a lazy classifier, requiring no learning phase. As a result, a plethora of big data classification approaches based on the lazy and slow KNN algorithm have been developed in the literature. For example, the furthest-pair-Based binary search tree (FPBST) [30] is an interesting method, which is based on constructing a binary search tree to speed up the classification process. The data examples are inserted into the tree depending upon their distance from the furthest pair, i.e., the examples that have the maximum distance in the dataset. The method was tested on around 20 datasets and reported good classification results. The main defect of this method is that it requires a model construction time, which reached around 50 minutes on HIGGS (11 million records) dataset [33]. If we neglect the model construction time, the model consumes a logarithmic amount of time during the testing stage.

Another method proposed in [34] used a MapReduce approach to split the big task between various workers. In order to compute the exact KNN, they used Spark in-memory computation to make the calculation of KNN fast and feasible for big data classification problems. Even though the method was running on a cluster of nodes, the time required to accomplish the classification task is high on large datasets like HIGGS. Also, the method requires a cluster of nodes, which might be costly, and for many users, this option is not always available.

Jesus Maillo and co-workers proposed a method called Fuzzy-KNN [35]. The method consists of two phases. The first one is transformation, in which the class membership degrees are added to the training set. Then, in the second phase, they apply the class membership to the test subset in order to perform the classification. They tested their

method on three popular big datasets, namely PokerHand, SUSY and HIGGS. The results they recorded illustrated that their method works well. However, the running time on the HIGGS dataset needed around 3 days to complete, which is much longer than what the approximation algorithms take to achieve almost the same accuracy.

A run-time comparison is done in [36]. The authors compared the run times of KNN in the Hadoop and Spark ecosystems. They tested the work on the HIGGS dataset. To show the difference between these two systems, they performed classification using clusters with a different number of nodes ranging from 2 to 6. The run time they observed was 10 minutes for Spark and slightly above 15 minutes for Spark when 6 nodes are used. When only two nodes were used, the run time was about 1.6 hours for Hadoop and around 16 minutes for Spark. This shows that finding the exact KNN is very costly, even when the algorithm runs on a cluster of nodes. Therefore, more innovative approaches to approximation are needed to overcome the huge time complexity associated with the exact KNN.

Another MapReduce-based implementation of KNN was done in [37]. The method was tested using various numbers of Map and Reduce operations. The run time on datasets like SUSY, with 5 Million examples, required about 3.4 hours [30].

In [38] the authors proposed a new KNN-based method to improve the performance of KNN on big datasets. The method works by aggregating the most relevant samples together using a mini-classifier, extreme learning machine (ELM), then applying the exact KNN of the groups in order to find the nearest neighbor. In this method, they need to train mini-classifier(s) in order to prepare the training data for the KNN, which requires more time. They tested their method on popular big datasets such as HIGGS and SUSY. The average time their algorithm consumed, according to the reported results, on a dataset like HIGGS was about 5K seconds.

Norm-based binary tree (NBT) and Minimum/Maximum norms-based binary tree (MNBT) are two methods proposed in [32]. In NBT the examples are stored recursively in a binary search tree based on their norm values. In MNBT, the examples are stored in a binary search tree based on their distance from the examples of the minimum and maximum 2-norms. Although these two algorithms are fast in performing predictions, they require a tree construction time, which is $sO(nd + n \log n)$. In addition, the fastest algorithm among these two, NBT, recorded inferior average accuracy on several tested datasets, 0.354, which indicates a considerable trade-off between time and accuracy.

Hassanat and co-workers [39] proposed an innovative rapid classification approach based on the well-known magnetic force (MF). This classifier calculates the magnetic force at each discrete point in the feature space based on the number of points belonging to a certain class/magnet. The magnetic forces recorded in the training model by various magnets/classes are used to classify unseen examples. According

to their experimental results using 28 distinct datasets, the MF classifier achieves equivalent classification accuracy when compared to existing classifiers. More crucially, it is demonstrated that the MF classifier classifies an unseen example in a constant time while creating its trained model in a linear time.

MULTI-Layer heterogeneous Ensemble System (MULES) was introduced in [40]. The work presented a new classification framework that consists of several heterogeneous classifiers. Also, an evolutionary feature selection algorithm was introduced in order to improve the classification results. Their results show that the model performs better than several state-of-the-art methods.

The field of ensemble learning is wide and there are many other methods were developed for various purposes including but not limited to [41], [42], and [43].

Other Big Data classification-related works are also worth mentioning, such as [44], [45], [46], [47], [48], [49], [50], [51], [52], [53], [54], [55], [56], [57], [58], [59], and [60] etc.

According to the literature on big data classification methods, the majority of them require two types of time, training and testing, and both timeframes are proportional to the size of the dataset. To the best of our knowledge, no classifier, regardless of the size of the big data set being classified, works in constant time, with the exception of the MF classifier in the testing phase, which requires a linear time at best in the training phase. Consequently, The proposed constant-time ensemble learning classifier for Big Data (CTELC) fills the gap.

Nevertheless, the most similar work found in the literature to the proposed method is the work of Louppe and Geurts [61], who proposed a supervised learning ensemble framework for classification tasks based on the concept of Random Patches (RP). This ensemble is built from random subsets of instances and features from the entire training dataset. Each individual model in the ensemble is generated using a decision tree-based estimator. For classification tasks, the voting rule is employed, whereas for regression tasks, the average of all model outputs is used. The RP classifier was tested on 29 datasets and compared to various popular ensemble approaches, delivering comparable accuracy but drastically decreasing memory requirements, especially in cases with severe memory restrictions.

However, The number of examples and features for each random patch were tuned on the validation set using a grid-search procedure, using the grid 0.01, 0.1, . . . , 0.9, 1.0 [61], which means that at some point some random patches may be formed by all examples and all features, i.e. the size of each patch is proportional to the size of the training dataset, which renders the proof of this method to be a constant time invalid regardless of the classifier used, particularly, when using decision trees, as the time complexity of building a decision tree is typically $O(nd \log n)$ [62]. Therefore, the main significant distinction between the proposed method and other approaches found in the literature is its constant time

characteristic. Thus, the key distinctiveness of the proposed CTELC is the endeavor to attain a constant time complexity classifier, which distinguishes it from earlier techniques in the literature. We have found this to be an essential contribution, especially when dealing with big datasets.

III. THE PROPOSED CTELC

Given a big training dataset D which has n observations and d features. The proposed CTELC selects a random number of subsets with replacement $\lambda \in \mathbb{N}$ and each subset $S_\lambda \subset D$. Each one of the created subsets has $\alpha \ll d$ number of features and $\beta \ll n$ number of observations. The subsets in S might have intersected samples and features. On each $S_\lambda \in S$ we create a KNN, this KNN is a weak classifier as it uses a small number (α) of features from D . It is worth mentioning here that every S_λ has an equal number β of random observations from each class γ in D . In the testing phase of a vector \vec{x} , we use all the created weak KNN classifiers to give the final prediction y using the majority voting rule. The majority voting is applied using $mode(R)$ where R is a list containing the predictions, one from each weak KNN classifier. Algorithm 1 illustrates the procedure of the proposed CTELC (split stage), while Algorithm 2 shows the procedure of obtaining prediction (prediction stage) using the weak KNNs created in Algorithm 1. Also, Figure 1 illustrates the proposed CTELC.

It is worth noting that each weak KNN classifier has its own set of related features that were allocated at random during the split stage. We use these features during testing to pass the appropriate features from the test set to the corresponding KNN. Furthermore, as shown in Algorithm 2, the final prediction is made by taking the most frequent label value from all of the weak KNN classifiers.

Algorithm 1 Pseudocode of the Split Stage of the Proposed CTELC. Here, η Is the Number of Estimators (η Is Equal to λ as We Create an Estimator for Each Chunk of the Data)

Require: D_{train} , α , β , η

Ensure: List of weak KNN classifiers (cls), List of random features for each subset(F)

- 1: **while** $idx \leq \eta$ **do** for the number of estimators η (weak KNNs), $idx = 1$
 - 2: $RF \leftarrow$ array of α random features ($indices \in [0-d]$)
 - 3: $RS \leftarrow$ array of β random samples ($indices \in [0-n]$)
 - 4: $S[idx] \leftarrow D[RS, RF]$ \triangleright add a subset of D to S .
Note that every subset in S has equal number of random samples from each class in D
 - 5: $F[idx] \leftarrow RF$ \triangleright add RF to F to be used in the classification phase
 - 6: $cls[idx] \leftarrow KNN(S[idx])$ \triangleright create weak using the subset $S[idx]$
 - 7: $idx = idx + 1$
 - 8: **end while**
-

We are undertaking simple bagging on short chunks of data, as indicated by the algorithms 1 and 2. Because the

Algorithm 2 Pseudocode of the Prediction Stage of the Proposed CTELC

Require: \vec{x}_{test} , F , cls , η

Ensure: \hat{y} : prediction

- 1: **while** $idx \leq \eta$ **do** for the number of estimators η (weak KNNs), $idx = 1$
 - 2: $RF \leftarrow F[idx]$
 - 3: $subvector \leftarrow \vec{x}_{test}[RF]$
 - 4: $Votes[idx] \leftarrow cls[idx].predict(subvector)$
 - 5: $idx = idx + 1$
 - 6: **end while**
 - 7: $\hat{y} \leftarrow mode(Votes)$
-

samples in one category belong to the same class (subject), we can take fewer of these examples than the original and yet get a decent approximation of the entire subset. Multi-weak KNNs aid in obtaining patterns that a single weak KNN cannot. As a result, the algorithm is highly fast because it runs for a constant time for each given testing example. Note here in Algorithm 1-step 6 the KNN fit does not take time as fitting KNN does not require any tuning or parameters optimization, rather, it stores the data to be used for the prediction stage. Therefore, the time required to build KNN on each chunk can be considered zero.

The distance in each weak KNN is calculated for α features instead of d . Therefore, equation 1 becomes as follows:

$$MD(x, x') = \left(\sum_{i=1}^{\alpha} |q_i - q'_i|^p \right)^{\frac{1}{p}} \quad (2)$$

where q and q' are the feature vectors the first obtained from the training set, and the second obtained from the testing set, both having length = α and their values obtained from the same sub-features from the original feature space. The power value $p = 2$ is used in the conducted experiments.

A. COMPLEXITY ANALYSIS

As can be seen in Figure 1, and Algorithms 1 and 2: Given a big dataset with n examples, d features, and γ classes. When testing an example \vec{x} , the proposed CTELC applies the KNN only on the selected random number of subsets with replacement λ , each of which has only α features, which is $\ll d$, and β examples for each class γ , which is $\ll n$. Therefore, the time complexity of testing an example \vec{x} in the worst case will be: $O(\lambda \beta \alpha \gamma)$ and since λ is a predefined constant number, $\alpha \ll d$, $\beta \ll n$, and $\gamma \ll n$, then all of the four values can be considered as constants, hence the testing time complexity becomes: $O(4c)$ where c is a constant. The time of the voting process, which is finding the mode of an array of size λ is also a constant time. this makes the overall time complexity of the proposed CTELC = $O(5c) \approx O(c)$, and can be asymptotically approximated to $O(1)$.

As a result, regardless of the size of the training data, the proposed CTELC can achieve classification/prediction in a constant time.

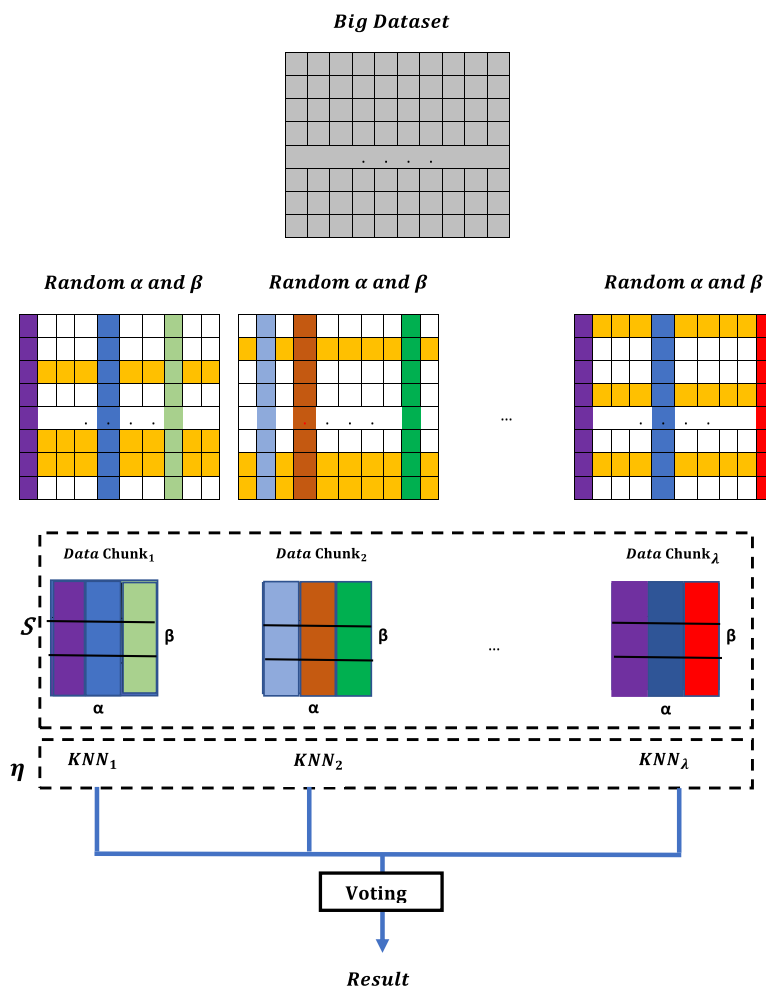


FIGURE 1. Typical diagram of the proposed CTELC. The selected features for each data chunk are presented in different colors.

Similarly, space complexity can be demonstrated to be a constant because the memory allocated by the big dataset classified can be released while keeping only the selected subsets of total size = $S(\lambda \beta \alpha \gamma) \approx S(c)$. Such a low space complexity is critical for mobile applications and other scenarios where memory use is a concern.

IV. DATASETS AND EXPERIMENTAL DESIGN

A. DATASETS

In this work, we used a variety of datasets ranging from medium size to big size. Table 1 shows the datasets used in our experiments and comparisons.

Although most researchers in this area have used two datasets, HIGGS and SUSY, we want to evaluate the proposed CTELC on a variety of datasets to demonstrate performance in light of varying data volumes. As one can see from Table 1, SUSY and HIGGS are the biggest datasets with 5 million and 11 million samples, respectively. Also, MiniBooNE and Mnist digits dataset are considered medium-size datasets with 130K and 70K samples, respectively. The other datasets are

small-size datasets. However, we include these small-size datasets just to present comparative results to illustrate how the algorithm works on such small datasets.

B. EXPERIMENTS DESIGN

In our experiments, there is no specific validation approach as we are comparing with several algorithms, each of which used a different approach for validation. However, the main used approaches are 5-fold and 10-fold cross-validations. For the performance evaluation, we used accuracy, following the rest of the research articles in this domain. Table 2 gives information about the computer’s specifications on which the experiments are conducted.

V. RESULTS AND DISCUSSION

The proposed CTELC, as previously described, includes several hyper-parameters, α , β , and η . Such hyper-parameters can be adjusted independently for each problem. However, as mentioned in [63], selecting a large number of features for a classification problem introduces geometric distortions

TABLE 1. A summary of the datasets used to evaluate the proposed CTELC classifier and the ratio of the data used by CTELC to the original data volume.

Dataset	# Size	# Classes	# Dimensions	Type	Used samples	Used Ratio
Covtype	581,012	7	54	Integers	35,000	0.06
HIGGS	11,000,000	2	28	Real	10,000	0.0009
MiniBooNE	130,064	2	50	Real	10,000	0.076
Homus	15,199	32	1600	Integers	160,000	10.53
Letter	20,000	26	16	Real	100,000	5
Mnist	70,000	10	784	Integers	50,000	0.71
Nist	44,951	26	1024	Integers	130,000	2.89
Pendigits	10,992	10	16	Integers	50,000	4.55
SUSY	5,000,000	2	18	Real	10,000	0.002
Usps	9,298	10	256	Real	50,000	5.38
Gisette	7,000	2	5000	Real	10,000	1.43

* The used samples are calculated as $\beta * \gamma * \eta$. In our case, $\beta = 250$ and $\eta = 20$

TABLE 2. The specification of the computer used for the experiments.

Specification	Value
Processor	Intel(R) Core(TM) i5-4590 CPU @ 3.30GHz, 3301 Mhz, 4 Core(s), 4 Logical Processor(s)
Physical Memory (RAM)	8.00 GB
Total Physical Memory	7.80 GB
Total Virtual Memory	9.05 GB

number, we choose a constant number of features to preserve the proposed CTELC classifier’s constant time advantage.

To approximately optimize β and η for the proposed algorithm, we perform a grid search on two different volume datasets and consider the best values for the rest of the datasets. The datasets used for grid search are MiniBooNE, around 130K samples, and Usps datasets, about 9K samples. Figure 2 shows the grid search results of selecting a different number of samples and various numbers of estimators, on these two datasets, in addition to the average on these two datasets.

As one might anticipate, as the number of examples and estimators increases, the accuracy goes up. Figure 2(a) shows that we record high accuracy values, almost 0.86, even when the number of samples is small, 250, compared to the whole available population, 130K. That is, according to the central limit theorem [66], if one takes “enough” random samples from a population, it would give results as if the whole population is considered. The aim here is to define “enough”, i.e. to find the best possible β and η values by which we achieve acceptable accuracy within a reasonable time delay. For the rest of the experiments, we set β and η values to be 250 and 20, respectively.

As a result, we can estimate how much data the proposed CTELC uses from the actual dataset. Table 1 gives information about the ratio of data used by CTELC to the samples in the original dataset. Obviously, the ratio depends mainly on the values of β , η as well as the number of classes in a given dataset. The proposed CTELC’s major purpose is to efficiently classify enormous volumes of data. However, in the case of a small dataset with a large number of classes, we should feed the model with more samples than the original dataset, which unjustifiably raises the model’s space complexity. For example, the original Homus dataset contains around 15K samples divided into 32 classes, and when we use the proposed CTELC on this dataset, we need to feed it with 160K samples, which is a constant, but it is also a tenfold increase in data size, in order to solve a task that can be solved using the exact KNN. When using CTELC on

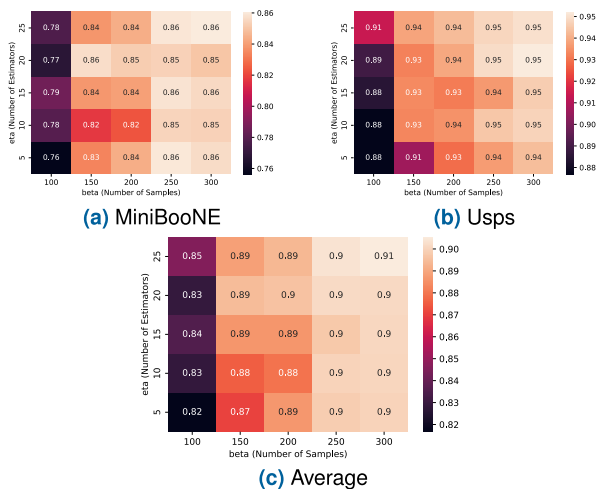


FIGURE 2. Grid search on two datasets and the average of both datasets. The results were obtained using a 5-fold cross-validation approach.

and results in high prediction error [64], [65]. The effect of high dimensional feature space will increase in our case as we are trimming the population by selecting only β number of samples instead of the whole sample space. When the number of features is too small the error as the features will not be sufficient to distinguish and detect the unique pattern from the feature vectors. Due to the previous reasons, we set the number of features to be \sqrt{d} , where d is the number of features in the original dataset. However, if d is a large

HIGGS and SUSY, nevertheless, we simply need to feed the model 0.0009 and 0.002 from the original data, respectively. However, the proposed CTELC can be utilized on smaller datasets after adjusting β and η to be appropriate for the volume of the input dataset, as partitioning has been shown to be beneficial in tackling the class imbalance problem even on relatively small datasets [67].

We maintain good accuracy while feeding the model a substantially lesser number of examples from the big datasets. The findings, accuracy, and time on the largest datasets utilized in this research, HIGGS, and SUSY, are shown in Table 3

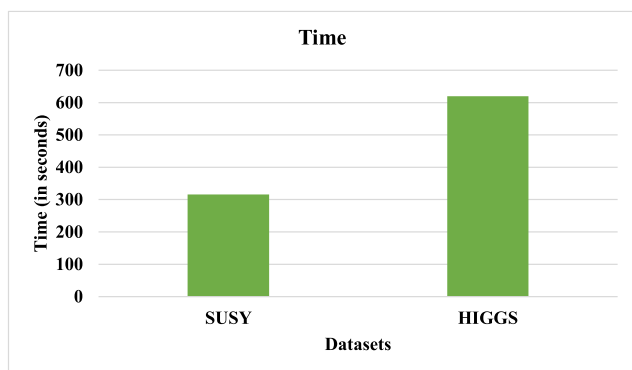


FIGURE 3. The time consumed by the proposed CTELC on HIGGS and SUSY datasets.

TABLE 3. Accuracy comparisons between the proposed CTELC and other methods on HIGGS and SUSY datasets.

Method	Reference	SUSY	HIGGS
Proposed CTELC	-	0.758	0.590
MDT1	[68]	0.729	0.581
MDT2	[68]	0.749	0.600
FPBST	[30]	0.710	0.582
MR-KNN	[37]	0.694	-
KNN-IS	[34]	0.694	-
Fuzzy	[35]	0.735	-
NBT	[32]	0.594	0.529
MNBT	[32]	0.710	0.589

A careful inspection of Table 3 demonstrates that the proposed CTELC outperforms all previous approaches evaluated on SUSY and the majority of approaches tested on HIGGS, while employing a significantly smaller subset of the original data with each estimator. Figure 3 shows the time consumed by the proposed CTELC on HIGGS and SUSY datasets. Table 4 contains the information on the proposed CTELC’s performance on other common datasets.

Although the Mnist dataset contains 70000 examples, which is greater than the Nist (44951 examples), Table 4 shows that the time consumed on Nist (44951) was 11.80S, which is longer than that consumed on Mnist (9.20S), this is true because in these experiments we used the \sqrt{d} as the

number of features for each Random patch, which depends on the data used, Knowing that the amount of features in Nist is 1024, as opposed to Mnist’s (784). Another key parameter is the number of classes, which increases the space complexity and size of the model, and hence the time consumed; in this case, the Nist has 26 classes, whereas the Mnist has only 10 classes. It is worth noting that the CTELC’s time consumption can be modified by its hyperparameters (λ , β , α , and γ), which determine the size of the final ensemble model for each dataset, but it can be asymptotically approximated to constant.

As indicated in Table 4, a similar argument can be extended to the time consumed by the CTELC when applied to the other datasets. The constant time characteristic of the CTELC is advantageous for a variety of datasets, providing efficient and reliable classification while retaining competitive accuracy levels.

As seen in Table 4, the results are primarily determined by the dataset used. In terms of accuracy, some datasets performed poorly, such as the Covtype dataset.

The fundamental reason for this is that the Covtype is extremely sparse, with many zero-valued attributes. When CTELC creates a sub-model that selects from the available features, it is quite likely that the model will select all features with zero values. When this occurs with several estimators, the entire model is influenced, and these estimators all contribute equally to the final decision.

The second problem is that the Covtype dataset is severely class imbalanced, limiting the sample pool from which the CTELC draws samples to offer generalization when new unseen data is tested.

One thing to note from Table 4 is that the accuracy of RP is better than the proposed method in most cases. However, when it comes to run time, the proposed method significantly outperforms the RP, except for Covtype. The RP is faster on Covtype because the decision tree ignores the zero-valued features due to their low gain, while the KNN does not.

TABLE 4. The results (accuracy/time) of the proposed CTELC compared to RP using 5-fold cross-validation.

Datasets	Proposed		RP	
	Acc	Time	Acc	Time
Mnist	0.86	9.2	0.95	99.5
Usps	0.94	1.6	0.94	12.9
Homus	0.55	4.6	0.59	54.2
Nist	0.3	11.8	0.82	28.4
Pendigits	0.95	0.64	0.98	0.88
Covtype	0.4	57.2	0.59	56
Letter	0.7	1.3	0.93	0.68
MiniBooNE	0.87	8.3	0.93	78.29
Gisette	0.9	0.68	0.96	40.4

The accuracy results of CTELC on some datasets, such as SUSY and HIGGS, were much better than that of most methods compared; however, for some datasets, such as

TABLE 5. The results (accuracy/time) of the proposed CTELC compared to RP (with memory constraint = 0.5).

Run	Accuracy (RP/CTELC)	Time (RP/CTELC)
CIFAR-10		
1	0.397/0.33	741.6/23.1
2	0.381/0.34	700.2/22.4
3	0.402/0.35	720.1/20.6
Mnist*		
1	0.940/0.929	76.8/30.1
2	0.947/0.928	70.8/27.6
3	0.948/0.935	73.8/32.2

Covtype and NIST, we were not optimistic due to the sparse nature of both datasets, and one more thing is the number of random patches used in these experiments, which was 20, compared to RP, which used 250 random patches; we reduced this number because our focus in this paper was on the efficiency of the method, However, this hyperparameter can be modified to include additional data in the learning process, which would logically lead to an improvement in overall accuracy outcomes, providing that the increase is not proportional to the dataset used, i.e. the overall size of all ensembles remains constant.

To further illustrate the significance of the proposed method over the RP¹ method proposed in [61], we used the largest two datasets mentioned in their study, CIFAR-10 and Mnist, to compare RP's performance to the proposed method. CIFAR-10 contains 60K samples and 3072 features, in total. While the Mnist has 70K samples and 784 features. For this sub-experiment, we ran each method 3 times. Also, following [61], we took 50% as training and 25% as testing and ignored the validation set as it is not strictly needed for this illustration. Furthermore, we used 0.5 memory constraint, for RP, as it is the threshold where the authors in [61] reported the best accuracy. For both methods, the number of the used estimators is set to 25.

We utilized both methods using the same training and testing sets. The performance achieved on both datasets, by RP, is around the accuracy reported by the authors [61] (the small difference in the accuracy might be due to the randomness in the sample selection). The accuracy of CTELC is slightly less than the RP method on both datasets. However, when it comes to execution time we see that CTELC provides a significant improvement. Tables 4 and 5 show the results (accuracy and time in seconds) of both RP and CTELC.

Another analysis is done using the simple additive weighting (SAW) method. SAW is one of the basic and widely used Multi-Criteria Decision Analysis (MCDA) techniques [69], [70]. The analysis is done on the comparison recorded in Table 4. In this analysis, we assumed that time and accuracy have the same importance. Based on the average of the

¹The implementation is available in the SKlearn library: <https://scikit-learn.org/stable/modules/ensemble.html>

normalized scores across all datasets, the Proposed CTELC has a higher average score, 0.814, than the RP method, 0.729. This suggests that, when considering both accuracy and time with equal importance, the Proposed method performs better on average across the given datasets.

Moreover, We used the Welch Two Sample t-test for each case to evaluate if there was a statistically significant difference between CTELC and RP for both accuracy and time results presented in Table 4:

- **Accuracy Test:** The null hypothesis (there is no meaningful difference in accuracy between the two methods). The test shows that the p-value is 0.1854, and since it is greater than the standard significance level of 0.05, we do not have enough evidence to reject the null hypothesis or to prove otherwise. However, the 95% confidence interval for the mean difference between the CTELC and RP is -0.3446 to 0.0735. And the sample means are 0.7189 and 0.8544 for CTELC and RP respectively, therefore, the accuracy test results show that there is no statistically significant difference in accuracy between CTELC and RP. Moreover, the confidence interval contains zero, indicating that there is no notable difference in accuracy between the two methods.
- **Time Test:** The null hypothesis (there is no meaningful difference in consumed time by the two methods). The p-value is 0.03481, and therefore, We have adequate evidence to reject the null hypothesis because the p-value is less than the conventional significance level of 0.05. The 95% confidence interval for the mean difference between the CTELC and RP is -58.738 to -2.58, the sample means are 10.591 and 41.2500 for CTELC and RP respectively, therefore, there is a statistically significant difference in time consumption between the CTELC and RP, according to the time test. Moreover, the confidence interval does not include zero, showing that the time consumption of the two methods differs significantly. When compared to the RP, the CTELC has a much reduced mean time consumption.

Overall, the t-tests show that, while there is no statistically significant difference in accuracy between the two methods, there is a statistically significant difference in time consumption, with the CTELC being faster than the RP. This demonstrates the novelty and efficiency of the CTELC method in terms of time consumption when compared to the RP method. However, this does not necessarily mean that there is no difference in accuracy between both methods, perhaps if we used a larger number of datasets, RP could win in terms of accuracy because it uses larger sizes of patches, while CTELC uses a constant size.

To demonstrate the effect of data sparsity on CTELC results, we run a basic variance-threshold feature selection on the dataset *Covtype* to remove the problematic zero-valued features. We noticed a considerable improvement in the performance of the CTELC, over 22%, from 0.4 to 0.64. This happens because this dataset contains many zero-value features, therefore, many weak estimators might work on

TABLE 6. The accuracy of the proposed CTELC compared to other methods using 10-fold cross-validation.

Dataset	CTELC	RC-KNN	LC-KNN	FPBST	NBT	MNBT
Mnist	0.868	0.722	0.839	0.855	0.190	0.858
Usps	0.950	0.903	0.936	0.873	0.336	0.864
Pendigits	0.968	0.945	0.972	0.965	0.254	0.966
Letter	0.700	0.789	0.950	0.789	0.327	0.802
Gisette	0.920	0.931	0.953	0.895	0.616	0.884

TABLE 7. P-values of the pairwise comparisons using Nemenyi-Wilcoxon-Wilcox all-pairs test for the accuracy results reported in Table 6.

	CTELC	FPBST	LC-KNN	MNBT	NBT
FPBST	0.88	-	-	-	-
LC-KNN	1.00	0.59	-	-	-
MNBT	0.96	1.00	0.76	-	-
NBT	0.03	0.38	0.01	0.24	-
RC-KNN	0.88	1.00	0.59	1.00	0.38

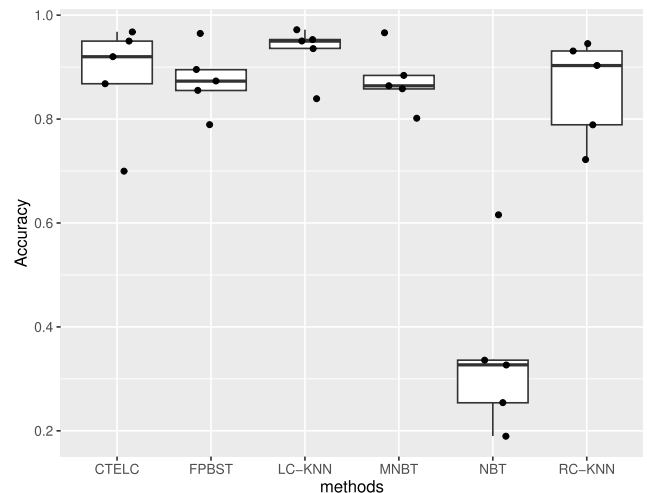
these features, which results in degradation in the overall performance. We argue that the performance would increase much more if the data was preprocessed using a more robust and efficient feature selection algorithm before being sent to the proposed CTELC.

Table 6 includes performance comparisons of CTELC with other KNN-based big data classification methods. As can be seen from this table and Figure 4 the CTELC method outperforms all methods compared on two datasets and provides comparative results on the others. In order to statistically analyze these results, we choose the Friedman test, a non-parametric statistical test that is frequently used to compare the mean ranks of the performance of a number of methods in order to ascertain whether they are substantially different [71]. The null hypothesis of the Friedman test is that all compared methods are not equally effective. After applying the Friedman test to the data obtained in Table 6 we get $\chi^2(5) = 15$, $p = 0.01036$, since the p -value < 0.05 this allows for rejecting the null hypothesis indicating that there are significant differences in accuracy obtained on datasets based on the method used.

The effect size (degree of difference) for the Friedman test can be calculated using Kendall's coefficient of concordance (also known as Kendall's W). A substantial effect size is indicated by Kendall's W, which is 0.6. Cohen's interpretation standards (0.1: little effect; 0.3: moderate effect; and > 0.5 : significant effect) provide the foundation for Kendall's W.

The Nemenyi-Wilcoxon-Wilcox all-pairs test was used to determine which methods are significantly different because the Friedman test does not reveal which methods have a significant effect. The results are displayed in Table 7.

The Nemenyi-Wilcoxon-Wilcox all-pairs test results show that the p-value for the comparison between CTELC and NBT is 0.03, which is lower than the significance level of 0.05, as can be seen from the P-values in Table 7. This

**FIGURE 4.** Visualizing the accuracy results reported in Table 6.

implies that the performance of CTELC and NBT varies statistically significantly. However, CTELC and FPBST have a P-value of 0.8821, showing that neither the performance of these two approaches nor the performance of CTELC and the other methods, differs statistically significantly from one to another.

Because each of the previous Big data classification methods was evaluated on a different machine, time comparisons with reported consumed times are invalid. As a result, assessing the speed-up rates of various methods in this field is a good way to compare their efficiency. The speed-up equation is formulated as follows:

$$speedup = \frac{Time(KNN)}{Time(Method)} \quad (3)$$

where $Time(KNN)$ is the time consumed by KNN classification, while $Time(Method)$ is the time consumed by the compared classification method on the same dataset. Figure 5, shows the speed-up rates of the proposed CTELC compared to other methods.

As shown in Figure 5, the proposed CTELC achieves comparable, if not superior, speed-up rates on several datasets. It is worth noting, however, that the recorded speed time for other algorithms is only for their run time.

As previously stated, each of the majority of these approaches has a construction/training time that is often significantly longer than their run time. In CTELC, such a construction/training time is 0 because the proposed method

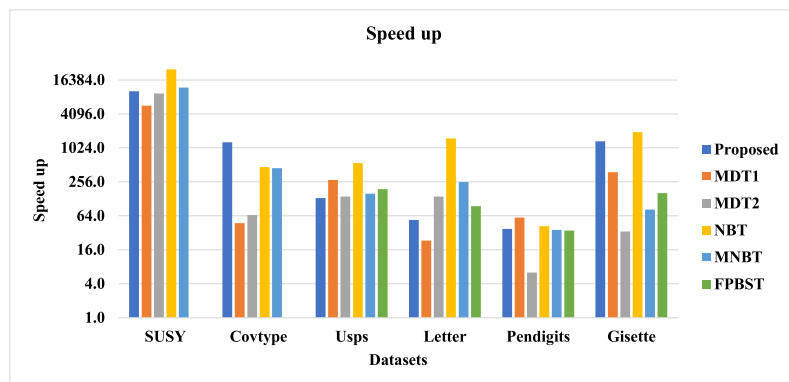


FIGURE 5. Speed up rates of the proposed CTELC compared to other methods on different datasets.

does not require time to build a model on training data. In terms of construction/training time, the proposed method would thus be superior to all existing methods, and none of these methods can compete with the proposed constant-time classifier in this regard.

VI. CONCLUSION

A new ensemble learning method for big data classification is proposed in this paper. The proposed method operates in a constant time and is unaffected by the volume of the training set, making it an ideal classifier for Big Data.

The experiments on 11 medium and big datasets, including HIGGS and SUSY, demonstrate that the proposed CTELC performs efficiently and swiftly completes the classification task. Comparing the proposed CTELC to other state-of-the-art big data classification methods reveals that it can efficiently and effectively function on big datasets, and obtain comparable if not superior results. A significant advantage of the proposed method is that it can be parallelized to set to work on a cluster of nodes, which we believe will result in faster and more accurate classification.

While the CTELC is similar to the RP ensemble classifier, it has significant differences that lead to its innovation. Unlike RP, which uses an online grid search during classifier training on each dataset, the CTELC uses an offline grid search once and forever. Furthermore, RP uses Decision Trees as its estimator, whereas CTELC makes use of the efficiency of KNN, a classifier that does not require substantial training time. This study demonstrates that, while the CTELC has slightly less accuracy than the RP, it greatly outperforms it in terms of speed. This trade-off, in which a slight loss of accuracy is exchanged for astounding speed, increases the CTELC's appeal, especially in scenarios such as Big Data classification, online applications, and resource-constrained machines. In these situations, acquiring a constant time classifier, such as that provided by CTELC, is extremely valuable and satisfies the need for quick and efficient processing.

The key limitation of the proposed CTELC is that its performance suffers when a given dataset is sparse or contains

many zero-valued features. In such circumstances, we recommend doing a feature selection before applying the proposed method to eliminate the superfluous features. However, further research is needed to critically look at and address the issue of sparsity to enhance the performance of the proposed method. Another limitation of this study is the use of only one distance metric, the Minkowski, which is not always the optimum distance to employ; other distance measures, such as Hassanat [72], [73], [74], maybe a better option.

In addition to focusing on the mentioned limitations, the future plans will include a MapReduce implementation of the proposed method to speed it up. Also, we think that adding some kind of weights to the predictions of the individual estimators based on the nearest neighbor concept may improve the classification performance. Furthermore, when a MapReduce implementation is provided for this method, the effect of the parameter K will be worth a deep discussion. We also plan to investigate the performance of the proposed method on more big datasets such as RLCP, KddCup 1999 (DOS vs. normal classes), and ECBDL14 (which has about 32 million instances).

ACKNOWLEDGMENT

Ahmad B. Hassanat would like to thank Mutah University for supporting his contribution.

REFERENCES

- [1] E. AlFaris, F. Irfan, G. Ponnampuruma, A. Jamal, C. van der Vleuten, N. Al Maflehi, S. Al-Qeas, A. Alenezi, M. Alrowaished, R. Alsaman, and A. M. A. Ahmed, "The pattern of social media use and its association with academic performance among medical students," *Med. Teacher*, vol. 40, no. sup1, pp. S77–S82, Jul. 2018.
- [2] A. D. Ahad and S. M. A. Lim, "Convenience or nuisance? The 'WhatsApp' dilemma," *Proc. Social Behav. Sci.*, vol. 155, pp. 189–196, Nov. 2014.
- [3] S. Greenwood, A. Perrin, and M. Duggan, "Social media update 2016," *Pew Res. Center*, vol. 11, no. 2, pp. 1–18, 2016.
- [4] R. Sahal, J. G. Breslin, and M. I. Ali, "Big data and stream processing platforms for Industry 4.0 requirements mapping for a predictive maintenance use case," *J. Manuf. Syst.*, vol. 54, pp. 138–151, Jan. 2020.
- [5] Y. Hajjaji, W. Boulila, I. R. Farah, I. Romdhani, and A. Hussain, "Big data and IoT-based applications in smart environments: A systematic review," *Comput. Sci. Rev.*, vol. 39, Feb. 2021, Art. no. 100318.

- [6] S. Hu, C. Xiong, M. Yang, H. Younes, W. Luo, and L. Zhang, "A big-data driven approach to analyzing and modeling human mobility trend under non-pharmaceutical interventions during COVID-19 pandemic," *Transp. Res. C, Emerg. Technol.*, vol. 124, Mar. 2021, Art. no. 102955.
- [7] A. Ponmalar and V. Dhanakoti, "An intrusion detection approach using ensemble support vector machine based chaos game optimization algorithm in big data platform," *Appl. Soft Comput.*, vol. 116, Feb. 2022, Art. no. 108295.
- [8] C. Vernallis, D. Oore, and J. Buhler, "Facial recognition, big data, and close readings: Tracing the asset in the *Bourne* and *Snowden* trailers," *Quart. Rev. Film Video*, vol. 37, no. 5, pp. 431–455, Jun. 2020.
- [9] S. Zerdoumi, A. Q. M. Sabri, A. Kamsin, I. A. T. Hashem, A. Gani, S. Hakak, M. A. Al-Garadi, and V. Chang, "Image pattern recognition in big data: Taxonomy and open challenges: Survey," *Multimedia Tools Appl.*, vol. 77, no. 8, pp. 10091–10121, 2018.
- [10] A. S. Tarawneh, A. B. Hassanat, C. Celik, D. Chetverikov, M. S. Rahman, and C. Verma, "Deep face image retrieval: A comparative study with dictionary learning," in *Proc. 10th Int. Conf. Inf. Commun. Syst. (ICICS)*, Jun. 2019, pp. 185–192.
- [11] M. S. Hossain and G. Muhammad, "Emotion recognition using deep learning approach from audio-visual emotional big data," *Inf. Fusion*, vol. 49, pp. 69–78, Sep. 2019.
- [12] M. Mohammadpoor and F. Torabi, "Big data analytics in oil and gas industry: An emerging trend," *Petroleum*, vol. 6, no. 4, pp. 321–328, Dec. 2020.
- [13] C.-F. Lai, W.-C. Chien, L. T. Yang, and W. Qiang, "LSTM and edge computing for big data feature recognition of industrial electrical equipment," *IEEE Trans. Ind. Informat.*, vol. 15, no. 4, pp. 2469–2477, Apr. 2019.
- [14] S. Neethirajan, "The role of sensors, big data and machine learning in modern animal farming," *Sens. Bio-Sens. Res.*, vol. 29, Aug. 2020, Art. no. 100367.
- [15] S. Boyapati, S. R. Swarna, V. Dutt, and N. Vyas, "Big data approach for medical data classification: A review study," in *Proc. 3rd Int. Conf. Intell. Sustain. Syst. (ICISS)*, Dec. 2020, pp. 762–766.
- [16] C. Jiang and Y. Li, "Health big data classification using improved radial basis function neural network and nearest neighbor propagation algorithm," *IEEE Access*, vol. 7, pp. 176782–176789, 2019.
- [17] S. Rajendran, O. I. Khalaf, Y. Alotaibi, and S. Alghamdi, "MapReduce-based big data classification model using feature subset selection and hyperparameter tuned deep belief network," *Sci. Rep.*, vol. 11, no. 1, pp. 1–10, Dec. 2021.
- [18] S. Mnasri, N. Nasri, A. van den Bossche, and T. Val, "A new multi-agent particle swarm algorithm based on birds accents for the 3D indoor deployment problem," *ISA Trans.*, vol. 91, pp. 262–280, Aug. 2019.
- [19] S. Mnasri, N. Nasri, and T. Val, "The 3D indoor deployment in DL-IoT with experimental validation using a particle swarm algorithm based on the dialects of songs," in *Proc. 14th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Jun. 2018, pp. 928–933.
- [20] A. Abadleh, S. Han, S. J. Hyun, B. Lee, and M. Kim, "Construction of indoor floor plan and localization," *Wireless Netw.*, vol. 22, no. 1, pp. 175–191, Jan. 2016.
- [21] S. Mnasri, N. Nasri, M. Alrashidi, A. van den Bossche, and T. Val, "IoT networks 3D deployment using hybrid many-objective optimization algorithms," *J. Heuristics*, vol. 26, no. 5, pp. 663–709, Oct. 2020.
- [22] A. B. A. Hassanat, V. B. S. Prasath, M. Al-kasassbeh, A. S. Tarawneh, and A. J. Al-shamailh, "Magnetic energy-based feature extraction for low-quality fingerprint images," *Signal, Image Video Process.*, vol. 12, no. 8, pp. 1471–1478, Nov. 2018.
- [23] A. S. Tarawneh, C. Celik, A. B. Hassanat, and D. Chetverikov, "Detailed investigation of deep features with sparse representation and dimensionality reduction in CBIR: A comparative study," *Intell. Data Anal.*, vol. 24, no. 1, pp. 47–68, Feb. 2020.
- [24] A. B. A. Hassanat, "On identifying terrorists using their victory signs," *Data Sci. J.*, vol. 17, p. 27, Oct. 2018.
- [25] A. B. Hassanat, E. Btoush, M. A. Abbadi, B. M. Al-Mahadeen, M. Al-Awadi, K. I. Mseidein, A. M. Almaseden, A. S. Tarawneh, M. B. Alhasanat, V. S. Prasath, and F. A. Al-alem, "Victory sign biometric for terrorists identification: Preliminary results," in *Proc. 8th Int. Conf. Inf. Commun. Syst. (ICICS)*, Apr. 2017, pp. 182–187.
- [26] C. Panjaitan, A. Silaban, M. Napitupulu, and J. W. Simatupang, "Comparison K-Nearest Neighbors (K-NN) and Artificial Neural Network (ANN) in real time entrants recognition," in *Proc. Int. Seminar Res. Inf. Technol. Intell. Syst. (ISRITI)*, Nov. 2018, pp. 1–4.
- [27] M. Rezwanul, A. Ali, and A. Rahman, "Sentiment analysis on Twitter data using KNN and SVM," *Int. J. Adv. Comput. Sci. Appl.*, vol. 8, no. 6, pp. 19–25, 2017.
- [28] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, Sep. 1975.
- [29] A. Hassanat, "Furthest-pair-based decision trees: Experimental results on big data classification," *Information*, vol. 9, no. 11, p. 284, Nov. 2018.
- [30] A. B. A. Hassanat, "Furthest-pair-based binary search tree for speeding big data classification using K-nearest neighbors," *Big Data*, vol. 6, no. 3, pp. 225–235, Sep. 2018.
- [31] A. B. A. Hassanat, "Two-point-based binary search trees for accelerating big data classification using KNN," *PLoS ONE*, vol. 13, no. 11, Nov. 2018, Art. no. e0207772.
- [32] A. Hassanat, "Norm-based binary search trees for speeding up KNN big data classification," *Computers*, vol. 7, no. 4, p. 54, Oct. 2018.
- [33] P. Baldi, P. Sadowski, and D. Whiteson, "Searching for exotic particles in high-energy physics with deep learning," *Nature Commun.*, vol. 5, no. 1, pp. 1–9, Jul. 2014.
- [34] J. Maillou, S. Ramírez, I. Triguero, and F. Herrera, "kNN-IS: An iterative Spark-based design of the k-Nearest Neighbors classifier for big data," *Knowl.-Based Syst.*, vol. 117, pp. 3–15, Feb. 2017.
- [35] J. Maillou, J. Luengo, S. García, F. Herrera, and I. Triguero, "Exact fuzzy k-Nearest Neighbor classification for big datasets," in *Proc. IEEE Int. Conf. Fuzzy Syst. (FUZZ-IEEE)*, Jul. 2017, pp. 1–6.
- [36] A. Mostafaiepour, A. J. Rafsanjani, M. Ahmadi, and J. A. Dhanraj, "Investigating the performance of Hadoop and Spark platforms on machine learning algorithms," *J. Supercomput.*, vol. 77, no. 2, pp. 1273–1300, Feb. 2021.
- [37] J. Maillou, I. Triguero, and F. Herrera, "A MapReduce-based k-Nearest Neighbor approach for big data classification," in *Proc. IEEE Trust-com/BigDataSE/ISPA*, vol. 2, Aug. 2015, pp. 167–172.
- [38] A. Shokrzade, M. Ramezani, F. Akhlaghian Tab, and M. A. Mohammad, "A novel extreme learning machine based kNN classification method for dealing with big data," *Expert Syst. Appl.*, vol. 183, Nov. 2021, Art. no. 115293.
- [39] A. B. Hassanat, H. N. Ali, A. S. Tarawneh, M. Alrashidi, M. Alghamdi, G. A. Altarawneh, and M. A. Abbadi, "Magnetic force classifier: A novel method for big data classification," *IEEE Access*, vol. 10, pp. 12592–12606, 2022.
- [40] T. T. Nguyen, N. Van Pham, M. T. Dang, A. V. Luong, J. McCall, and A. W. C. Liew, "Multi-layer heterogeneous ensemble with classifier and feature selection," in *Proc. Genetic Evol. Comput. Conf.*, Jun. 2020, pp. 725–733.
- [41] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger, "Snapshot ensembles: Train 1, get M for free," 2017, *arXiv:1704.00109*.
- [42] Z.-H. Zhou and J. Feng, "Deep forest: Towards an alternative to deep neural networks," in *Proc. 26th Int. Joint Conf. Artif. Intell. (IJCAI)*, 2017, pp. 3553–3559.
- [43] A. V. Luong, T. H. Vu, P. M. Nguyen, N. Van Pham, J. McCall, A. W.-C. Liew, and T. T. Nguyen, "A homogeneous-heterogeneous ensemble of classifiers," in *Proc. 27th Int. Conf. Neural Inf. Process. (ICONIP)*. Bangkok, Thailand: Springer, Nov. 2020, pp. 251–259.
- [44] W. Alzyadat, A. AlHroob, I. H. Almkahel, M. Muhairat, M. Abdallah, and A. Althunibat, "Big data, classification, clustering and generate rules: An inevitably intertwined for prediction," in *Proc. Int. Conf. Inf. Technol. (ICIT)*, Jul. 2021, pp. 149–155.
- [45] P.-Y. Wang, "Design of distributed multidimensional big data classification system based on differential equation," in *Advanced Hybrid Information Processing (Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering LNICST)*, vol. 347. Cham, Switzerland: Springer, 2021.
- [46] Y.-Y. Gao, J. Xiang, Y.-N. Tang, M. He, and W. Li, "Research on big data classification algorithm of disease gene detection based on complex network technology," *Advanced Hybrid Information Processing (Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering LNICST)*, vol. 347. Cham, Switzerland: Springer, 2021.
- [47] L.-H. Yang, J. Liu, Y.-M. Wang, and L. Martínez, "A micro-extended belief rule-based system for big data multiclass classification problems," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 51, no. 1, pp. 420–440, Jan. 2021.

- [48] P. Bedi, S. Goyal, R. Sharma, D. K. Yadav, and M. Sharma, "Smart model for big data classification using deep learning in wireless body area networks," in *Micro-Electronics and Telecommunication Engineering*. Singapore: Springer, 2021, pp. 215–224.
- [49] N. Albarrak, H. Alsanousi, I. Moulitsas, and S. Filippone, "Using big data to compare classification models for household credit rating in Kuwait," in *Proc. 6th Int. Congr. Inf. Commun. Technol.* (Lecture Notes in Networks and Systems), vol. 216, 2022, pp. 609–618.
- [50] I. Pintye, E. Kail, P. Kacsuk, and R. Lovas, "Big data and machine learning framework for clouds and its usage for text classification," *Concurrency Comput., Pract. Exp.*, vol. 33, no. 19, p. e6164, Oct. 2021.
- [51] D. García-Gil, J. Luengo, S. García, and F. Herrera, "Enabling smart data: Noise filtering in big data classification," *Inf. Sci.*, vol. 479, pp. 135–152, Apr. 2019.
- [52] N. A. Al-Thanoon, Z. Y. Algama, and O. S. Qasim, "Feature selection based on a crow search algorithm for big data classification," *Chemometrics Intell. Lab. Syst.*, vol. 212, May 2021, Art. no. 104288.
- [53] W. C. Sleeman IV and B. Krawczyk, "Multi-class imbalanced big data classification on spark," *Knowl.-Based Syst.*, vol. 212, Jan. 2021, Art. no. 106598.
- [54] S. M. Mujeeb, R. P. Sam, and K. Madhavi, "Adaptive exponential bat algorithm and deep learning for big data classification," *Sādhanā Acad. Proc. Eng. Sci.*, vol. 46, no. 1, p. 15, Dec. 2021.
- [55] M. Juez-Gil, Á. Arnaiz-González, J. J. Rodríguez, and C. García-Osorio, "Experimental evaluation of ensemble classifiers for imbalance in big data," *Appl. Soft Comput.*, vol. 108, Sep. 2021, Art. no. 107447.
- [56] J. A. R. Cedeño, C. F. Leopoldo, H. Neira-Molina, Y. J. García-López, R. Morales-Ortega, and H. Combata-Niño, "Big data classification using fuzzy logical concepts for paddy yield prediction," *Rev. Int. Geograph. Educ. Online*, vol. 11, no. 5, pp. 4483–4490, 2021.
- [57] N. P. Jayasri and R. Aruna, "Big data analytics in health care by data mining and classification techniques," *ICT Exp.*, vol. 8, no. 2, pp. 250–257, Jun. 2022. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2405959521000849>
- [58] M. Juez-Gil, Á. Arnaiz-González, J. J. Rodríguez, C. López-Nozal, and C. García-Osorio, "Approx-SMOTE: Fast SMOTE for Big Data on Apache Spark," *Neurocomputing*, vol. 464, pp. 432–437, Nov. 2021.
- [59] D. C. Asogwa, S. O. Anigbogu, I. E. Onyenwe, and F. A. Sani, "Text classification using hybrid machine learning algorithms on big data," *Int. J. Trend Res. Develop.*, vol. 6, no. 5, pp. 127–134, 2021.
- [60] R. T. Selvi and I. Muthulakshmi, "An optimal artificial neural network based big data application for heart disease diagnosis and classification model," *J. Ambient Intell. Humanized Comput.*, vol. 12, no. 6, pp. 6129–6139, Jun. 2021.
- [61] G. Louppe and P. Geurts, "Ensembles on random patches," in *Proc. Eur. Conf. Mach. Learn. Knowl. Discovery Databases (ECML PKDD)*. Bristol, U.K.: Springer, Sep. 2012, pp. 346–361.
- [62] H. M. Sani, C. Lei, and D. Neagu, "Computational complexity analysis of decision tree algorithms," in *Proc. 38th Int. Conf. Innov. Techn. Appl. Artif. Intell. (SGAI)*. Cambridge, U.K.: Springer, Dec. 2018, pp. 191–197.
- [63] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
- [64] M. Köppen, "The curse of dimensionality," in *Proc. 5th Online World Conf. Soft Comput. Ind. Appl. (WSC5)*, vol. 1, 2000, pp. 4–8.
- [65] A. C. Márquez, "The curse of dimensionality," in *Digital Maintenance Management*. Cham, Switzerland: Springer, 2022, pp. 67–86.
- [66] Y. Dodge, *The Concise Encyclopedia of Statistics*. New York, NY, USA: Springer, 2008.
- [67] A. B. Hassanat, A. S. Tarawneh, S. S. Abed, G. A. Altarawneh, M. Alrashidi, and M. Alghamdi, "RDPVR: Random data partitioning with voting rule for machine learning from class-imbalanced datasets," *Electronics*, vol. 11, no. 2, p. 228, Jan. 2022.
- [68] F. Wang, Q. Wang, F. Nie, W. Yu, and R. Wang, "Efficient tree classifiers for large scale datasets," *Neurocomputing*, vol. 284, pp. 70–79, Apr. 2018.
- [69] E. Triantaphyllou, *Multi-Criteria Decision Making Methods*. New York, NY, USA: Springer, 2000.
- [70] V. Belton and T. J. Stewart, *Multiple Criteria Decision Analysis: An Integrated Approach*. New York, NY, USA: Springer, 2002.
- [71] R. Eisinga, T. Heskes, B. Pelzer, and M. Te Grotenhuis, "Exact p -values for pairwise comparison of Friedman rank sums, with application to comparing classifiers," *BMC Bioinf.*, vol. 18, no. 1, pp. 1–18, Dec. 2017.
- [72] A. B. Hassanat, "Dimensionality invariant similarity measure," 2014, [arXiv:1409.0923](https://arxiv.org/abs/1409.0923).
- [73] H. A. A. Alfeilat, A. B. A. Hassanat, O. Lasassmeh, A. S. Tarawneh, M. B. Alhasanat, H. S. Eyal Salman, and V. B. S. Prasath, "Effects of distance measure choice on K-nearest neighbor classifier performance: A review," *Big Data*, vol. 7, no. 4, pp. 221–248, Dec. 2019.
- [74] A. Hassanat, E. Alkafaween, A. S. Tarawneh, and S. Elmougy, "Applications review of Hassanat distance metric," in *Proc. Int. Conf. Emerg. Trends Comput. Eng. Appl. (ETCEA)*, Nov. 2022, pp. 1–6.

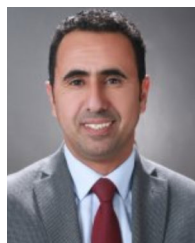


AHMAD S. TARAWNEH was born in Kerak, Jordan. He received the B.S. and M.S. degrees in computer science from Mutah University, in 2013 and 2015, respectively, and the Ph.D. degree in computer science from Eötvös Loránd University (ELTE), Hungary, Budapest. Currently, he is an Assistant Professor with the Department of Data Science and AI, University of Petra. He worked on the project EFOP (image and video processing), which was sponsored by the Hungarian Government and co-financed by the European Social Fund. His main research interests include computer vision, deep learning, machine learning, and data mining.

EMAN S. ALAMRI received the bachelor's degree in nutrition and food science from King Abdulaziz University, in 2007, the master's degree in nutrition and food science from Caledonian University, in 2013, and the Ph.D. degree in nutrition and food science from Plymouth University, in 2016. She is with the Food Science and Nutrition Department, Faculty of Science, University of Tabuk, Saudi Arabia. Her career history includes, positions such as a Teaching Assistant and an Associate Professor with the Department of Nutrition and Food Sciences, Tabuk University, since 2009 and since 2019, respectively. Her research interests include human nutrition, nutrition diseases, and food sustainability.



NAJAH NOORI AL-SAEDI was born in Iraq. He received the B.S. degree in computer science from Iraq. He is currently pursuing the M.S. degree with Mutah University, Jordan. His main research interests include machine learning, big data, and data mining.



MOHAMMAD ALAITHMAN received the Ph.D. degree in computer science, specializing in network security from Northumbria University, U.K. He is currently with the University of Petra. He possesses academic experience in teaching, course planning, and ensuring the quality of the educational process. His research interests include cybersecurity, network security, and cross-layered solutions for intrusion detection systems.



AHMAD B. HASSANAT (Senior Member, IEEE) was born in Jordan. He received the B.S. degree in computer science from Mutah University, Jordan, in 1995, the M.S. degree in computer science from Al Albayt University, Jordan, in 2004, and the Ph.D. degree in computer science from the University of Buckingham, U.K., in 2010. Since 2010, he has been a Faculty Member with the Faculty of Information Technology, Mutah University. His primary research interests include computer vision, machine learning, big data, and pattern recognition.