IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 27, 2019, accepted October 17, 2019, date of publication October 23, 2019, date of current version November 4, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2949238

Nature Inspired Instance Selection Techniques for Support Vector Machine Speed Optimization

ANDRONICUS A. AKINYELU [1,3] AND ABSALOM E. EZUGWU [2], (Member, IEEE)

[1]Department of Computer Science and Informatics, University of the Free State, Bloemfontein 9301, South Africa
[2]School of Mathematics, Statistics, and Computer Science, University of KwaZulu-Natal, KwaZulu-Natal 3201, South Africa
[3]School of Mathematics, Statistics and Computer Science, University of KwaZulu-Natal, Durban 4000, South Africa

Corresponding author: Andronicus A. Akinyelu (akinyeluaa@ufs.ac.za)

ABSTRACT Due to the fast-growing rate of information sources, many organizations and individuals are overwhelmed with vast amounts of data. The rate of data growth is very alarming, and it is already going beyond the Exabyte limit. Hence, there is an obvious need for fast and accurate big data classification tools. Machine Learning (ML) based solutions are very useful and reliable data classification tools, however, they cannot effectively handle large-scale datasets. This paper therefore proposes two intelligent instance selection techniques for optimizing the training and classification speed of ML algorithms, with a specific focus on Support Vector Machine (SVM). Furthermore, this paper considers two different approaches to instance selection namely: filter-based and wrapper-based. Different sets of experiments are performed on 20 small-scale datasets and 10 large- or medium-scale datasets. The results show that the proposed techniques improved SVM training speed in 100% (30 out of 30) of the datasets used for evaluation and simultaneously improved SVM predictive accuracy in some cases. Furthermore, statistical analysis test is carried out and the results reveal that the training speed of the proposed techniques is statistically significantly faster than the training speed of standard SVM and some other existing instance selection techniques. In real life application, such as video surveillance and intrusion detection systems, that require a classifier to be trained quickly for speedy classification of new target concepts, the filter-based techniques provide the best solutions; while the wrapper-based techniques are better suited for applications such as email filters, that are very sensitive to slight changes in predictive accuracy.

INDEX TERMS Machine learning, SVM speed optimization, support vector machine, instance selection.

LIST OF ABBREVIATION

Abbreviation	Meaning
ACO	Ant Colony Optimization
CBD	Concept Boundary Detection
FN	False Negative
FPA	Flower Pollination Algorithm
FPISA	Flower Pollination Instance Selection Algorithm
MCIS	Multi-Class Instance Selection
ML	Machine Learning
NI	Nature Inspired
SSA	Social Spider Algorithm
SSISA	Social Spider Instance Selection Algorithm
SVM	Support Vector Machine
UCI	University of California Irvine

LIST OF ALGORITHM NOTATION

Abbreviation	Meaning
APA	Average prediction accuracy
FT	User-defined Fitness Threshold
G(x)	Fitness Function
GB	Global Best
GBV	Global Best Vibration from entire population
LF	Levy Flight.
MaxG	Maximum Generation
Min	Minimum number of selected instances
N	size of the entire training dataset
NF	Number of folds for SVM cross validation
NRuns	Number of runs for SVM cross validation
NS	Number of Selected Instances

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

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see http://creativecommons.org/licenses/by/4.0/

P	Population size
Pm	User defined probability for changing spider mask.
PS	Probability Switch
Tot	Total number of times, each spider changes its target vibration
TS	Training Subset
TV	Target Vibration
VI	Vibration Intensity

I. BACKGROUND

Since the invention of integrated circuits and computer chips, the world has experienced a global spread of information through the internet. Currently, people everywhere are surrounded by large volumes of information from different sources, including pictures, videos, emails and websites. Moreover, because such information has become essential for decision-making, there is an obvious need for fast and efficient information extraction tools for improved decision making. Machine Learning (ML) algorithms are very useful tools for such information extraction. They can extract relevant patterns from data, analyse these patterns and turn them into meaningful information for decision-making. However, ML algorithms cannot effectively handle large scale datasets; their classification speed decreases with increase in dataset size. Many speed optimization methods have been proposed in literature. The diverse approaches in their design include instance selection, feature selection and parameter optimization. The instance selection approach aims to optimize speed by removing irrelevant and superfluous instances from a dataset. Feature selection is aimed at optimizing speed by removing extraneous features from a dataset. With parameter optimization, the approach aims to optimize speed by selecting optimal parameters from a list of parameter values. Among these three techniques, the instance selection approach is regarded as one of the best techniques [1].

Instance selection techniques are used to minimize the training time of ML algorithms (such as support vector machine, SVM) by discarding superfluous and harmful instances from a training dataset. Superfluous instances are instances that make a negligible contribution to the classification accuracy of a classifier, while harmful instances are instances that lead to increased false positive and false negative rates [2]. Superfluous and harmful instances contribute little to SVM prediction process [2]; hence, discarding them does not have a negative impact on the SVM training result [3]. Many instance selection methods exist in the body of literature, with most of them being designed using a k Nearest Neighbour (kNN) classifier [1]. Some methods were designed using k-d trees [4], clustering [5], [6], tabu search [7] and sequential search [8]. Some studies have also designed novel instance selection techniques based on the nature inspired (NI) algorithms such as the evolutionary algorithm (EA) [9], [10], memetic algorithm [11], ant colony optimization (ACO) [12] and the artificial immune system (AIS) algorithm [13]. However, to the best of the authors' knowledge, no study has explored the use of the flower pollination algorithm (FPA) and social spider algorithm (SSA) for instance selection and speed optimization.

This paper proposes two intelligent instance selection techniques for optimizing the training speed, classification speed and generalization performance of ML algorithms, with a specific focus on SVM. The two proposed techniques are inspired by FPA and SSA, respectively. Moreover, this study considers two different approaches to instance selection: filter-based and wrapper-based approach. The main difference between the filter-based and wrapper-based approaches lies in their speed and method of selection. The filter-based techniques are designed to select instances based on a user-defined fitness function, while the wrapper-based techniques are designed to select instances based on the predictive accuracy of a classifier. Experimental results show that the proposed filter-based techniques perform faster than their wrapper-based counterparts, because, the fitness function computation in the filter-based techniques is faster than the fitness function computation in wrapper-based techniques. However, the wrapper-based techniques produce better classification accuracy than do the filter-based techniques. This is because the wrapper-based techniques use an ML classifier in their fitness function computation.

The filter-based techniques are suitable for applications, such as video surveillance and intrusion detection systems that require a classifier to be trained very quickly for speedy classification of new target concepts [3]. Moreover, these applications require the classifier to be trained on large training sets. Examples of such applications include video surveillance and intrusion detection. For this kind of applications, SVM training time can be excessively high, which renders SVM ineffectual. Furthermore, even in applications where training can be performed offline (such as email detection systems), if the size of the training data or number of classes is large, then SVM computational complexity will be intolerable [3]. The wrapper-based techniques are better suited for applications where sensitivity to slight changes in predictive accuracy is more important than classification speed, such as is needed for email spam or phishing classifiers. For instance, even one important email being misclassified could have huge economic implications due to lost business opportunities. The technical contributions of this paper are as follows:

A. NATURE-INSPIRED INSTANCE SELECTION TECHNIQUES

This paper presents two novel nature-inspired instance selection techniques (social spider instance selection algorithm, SSISA, and flower pollination instance selection algorithm, FPISA) for improving the training speed, storage capacity, computational complexity, and generalization performance of ML algorithms, with a specific focus on SVM. Moreover, this paper presents two variants for each of the proposed technique: the wrapper-based and filter-based variants.

B. FILTER- AND WRAPPER-BASED FITNESS FUNCTION

This paper presents two novel fitness functions suitable for nature-inspired instance selection techniques. The first fitness function is suitable for the filter-based variants of the proposed techniques, while the second fitness function is suitable for the wrapper-based variants.

The rest of the paper is organized as follows: Section II provides a review on some existing speed optimization techniques. Thereafter, Section III provides specific details on the proposed techniques and Section IV provides information on the experimental setup and datasets used for their evaluation. In addition, Section IV provides detailed information on the experimental and statistical results produced by the proposed filter-based and wrapper-based techniques. Finally, conclusions and future research directions are provided in Section V.

II. RELATED WORK

SVM is a renowned and highly efficient ML classification algorithm, proposed in 1995, by Cortes and Vapnik [14]. In data classification, SVM is an excellent choice because of its outstanding performance and good generalization capability [15]. However, SVM is still outperformed by many classifiers when applied to large-scale datasets [15] because the SVM has high computational complexity, which scales up linearly with the number of support vectors [16]. That is, large datasets produce many support vectors [16]. To offset this problem, two common speed optimization approaches may be adopted; these are the algorithmic and data processing approaches [3]. The algorithmic approach aims to speed up the SVM by using algorithms that increases the speed of the quadratic programming (QP) solver [3]. The data processing approach aims to increase speed of SVM by reducing the dataset dimension [3]. To this end, some studies have focused on feature selection, while others have focused on parameter optimization and instance selection. This section presents a review on some existing instance selection techniques.

Albelwi and Mahmood [17] proposed an instance selection technique for big datasets. The technique is a modified version of an existing instance selection technique called random mutation hill climbing (RMHC). The authors used the technique in combination with deep convolutional neural networks and tested the resultant model on large datasets. When the authors compared the new technique with standard RMHC, the results showed that it performed faster than RMHC. In [18], Blachnik evaluated the performance of an ensemble of instance selection algorithms, which they then combined with the aim of improving the quality of dataset size reduction. The algorithms combined were the edited nearest neighbour (ENN) and condensed nearest neighbour (CNN). Moreover, to ensure diversity in sub-models, the authors applied the ensemble to datasets with different feature subsets. Results from the study show the possibility of obtaining trade-offs between classification accuracy and data compression. Lee and Mangasarian [19] introduced a new technique called reduced SVM, with the aim of reducing the classification speed of SVM by generating a non-linear separating surface suitable for big data classification. The non-linear separating surface was generated by firstly decomposing the entire dataset (to be classified) into smaller linear sub-problems. Afterwards, one of the sub-problems was randomly selected and used to produce the separating surface. Lei and Govindaraju [20] proposed a new method for SVM speed optimization. Their method consists of two algorithms; namely, principal component analysis (PCA) and recursive feature elimination (RFE). The first algorithm (i.e. PCA) was used to reduce the dataset dimension, while the second algorithm (i.e. RFE) was used to select relevant features. The selection of relevant features reduced the number of irrelevant and non-discriminative features. The technique was evaluated, and result revealed that it improved the speed of SVM. Garcı*et al.* [21] introduced an EA-based instance selection technique, which they designed to obtain generalized instances from imbalanced datasets. The efficacy of the technique was evaluated on several imbalanced datasets and it produced improved results compared to other techniques.

Angiulli and Astorino [22] presented a speed optimization algorithm, based on an existing data reduction technique called Fast Condensed Nearest Neighbour (FCNN), which had been originally published by the authors in [23]. FCNN was combined with SVM, with the aim of improving SVM classification speed. The algorithm was evaluated on three big datasets, and it produced improved result. Tsai and Cheng [24] introduced an instance selection technique for bankruptcy prediction. In that study, the authors hybridized a clustering algorithm with four classifiers; namely, decision trees, SVM, artificial neural networks (ANNs) and logistic regression. The hybridized techniques were evaluated on four datasets and results showed that SVM outperforms three of the other algorithms, decision tree, ANN and logistic regression. Panda *et al.* [3] proposed a filter-based technique for speeding up SVM, called concept boundary detection (CBD). The technique is divided into two stages, namely the concept independent and concept-specific sampling stages. In the first stage, the CBD algorithm identifies K nearest neighbours for each instance (using the k-NN algorithm) and removes instances that are not support vectors. Then in the second stage, the algorithm uses a scoring function (also designed in the study) to select boundary instances. Panda *et al.* [3] noted that the function allocates more weight to instances close to the boundary between the positive and negative classes. This is because boundary instances contribute more to the generalization performance of SVM models. The technique was evaluated, and it yielded good improvement in SVM speed.

Furthermore, in a different study, Chen *et al.* [25] proposed a filter-based data reduction technique for selecting boundary instances, called multi-class instance selection (MCIS). In the study, firstly, Chen et al. used a clustering algorithm to select cluster centers of positive class instances. Furthermore, they used the cluster centers to select boundary instances. Chen *et al.* based the design of their MCIS algorithm on

two theories. The first theory states that negative instances close to cluster centers of a positive class are also close to the boundary. Secondly, positive instances far from the cluster centers of a positive class are close to the boundary. Therefore, positive instances that are close to a boundary and negative instances that are far from a boundary do not contribute meaningfully to the generalization performance of a classifier. By contrast, positive instances that are far from a boundary and negative instances that are close to a boundary contribute more to the prediction accuracy of a classifier. Chen et al. evaluated their MCIS on some datasets, and it produced improved results. In another study, Anwar *et al.* [12] introduced an extended version of an existing wrapper-based instance selection technique called ADR-miner, with an emphasis on improving the prediction accuracy of a classifier. The algorithm is divided into two stages and it uses two different classifiers at each stage. In the first stage, the ADR-Miner uses the ant colony optimization (ACO) algorithm and a classifier to select the best reduced subset. Specifically, the algorithm use ACO to select different subsets and then uses the classifier to evaluate the quality of each subset. In the second stage, the best reduced subset is used to build the final prediction model. As mentioned above, ADR-miner is designed to use a different classifier at each stage; that is one classifier is used to evaluate the quality of each dataset and then another classifier is used to build the final classification model. The technique was evaluated on 37 datasets and it produced improved prediction accuracy.

Of the publications surveyed, a few explored using FPA or SSA for either feature selection or other optimization problems rather than for instance selection. To elaborate, the authors in [26], [27] introduced FPA-based techniques for feature selection. They evaluated the techniques on classification and regression datasets, showed satisfactory results. In another study, AbdEl-Fattah *et al.* [28] introduced a hybrid approach for feature selection. They designed a wrapper-based technique using FPA and the clonal selection algorithm (CSA). In addition, they used the optimum path forest (OPF) classifier to evaluate the objective function of the technique and it produced good results. In addition, Zawbaa *et al.* [29] introduced a multi-objective fitness function that combines the capabilities of both wrapper-based and filter-based fitness functions. The fitness function uses the FPA to search for optimal features. The technique was tested on eight datasets, and it produced competitive results. SSAs have also been used to solve different real world problems of economic load dispatch problems [30], [31], clustering [32], base station switching problem [33], etc. However, it has not be fully used to solve the feature selection or instance selection problems. Most of the spider-inspired data reduction techniques in the literature have used the Social Spider Optimization Algorithm (SSOA) [34]–[36] and not SSA.

As shown in the survey, bio-inspired algorithms can be used to solve optimization problems, such as instance selection, feature selection, and parameter optimization. However, to the best of our knowledge, no study used SSA or FPA to handle instance selection or SVM speed optimization. FPA and SSA are efficient bio-inspired algorithms. FPA is unique in addressing long-distance pollination and flower consistency. Long-distance pollination makes FPA capable of escaping a local search space and exploring a larger search space. In addition, flower consistency gives FPA the capacity to choose similar solutions more frequently and thus guarantee fast convergence. The interaction between long-distance pollination and flower consistency, and the selection of the best solution makes FPA very attractive and efficient [37]. Besides, FPA has the capacity to adaptively search a large space with many local optima [27]. On the other hand, SSA has a unique searching pattern and foraging model [38]. Moreover, SSA can search for optimal solutions in some complex optimization problems, and it has an underlying social foraging strategy. All these unique features make FPA and SSA very applicable for solving real world optimization problems involving instance selection. This paper therefore introduces SSA-based and FPA-based intelligent instance selection techniques for optimizing the training speed, classification speed, storage capacity and generalization performance of ML algorithms, with a particular focus on SVM.

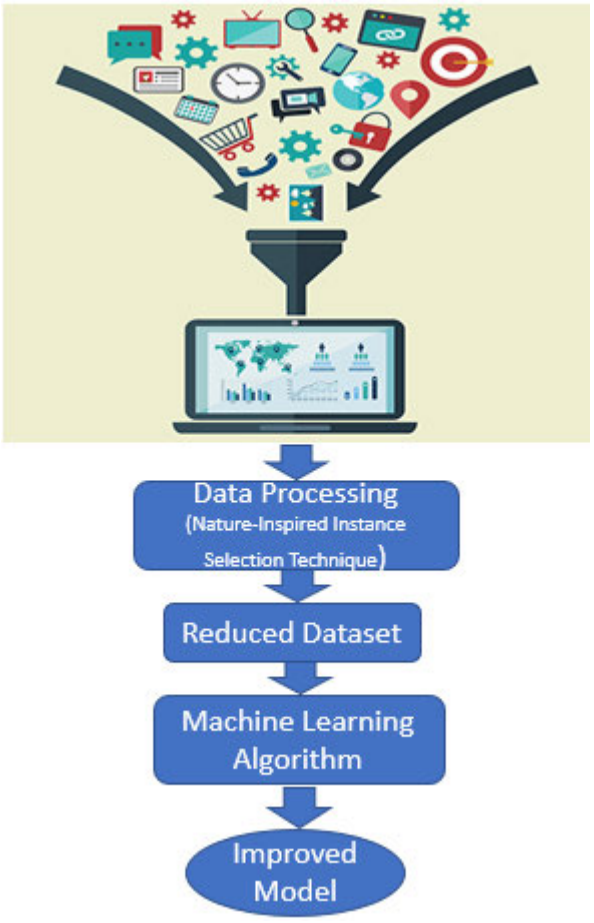

FIGURE 1. Pictorial description of the hybrid Nature-Inspired instance selection techniques for SVM speed optimization.

III. PROPOSED TECHNIQUES

This paper proposes two filter-based and two wrapper-based instance selection techniques for the optimization of SVM training speed and predictive accuracy. The primary objective of the proposed filter-based techniques is to improve SVM training speed, while the primary objective of the proposed wrapper-based techniques is to improve SVM predictive accuracy. The proposed techniques aim to select a classification model that will produce the best predictive accuracy. The pictorial description of the hybrid Nature-Inspired instance selection techniques for SVM speed optimization is shown in Figure 1. More details about the techniques are provided in Sections III.A and III.B respectively.

A. FLOWER POLLINATION-INSPIRED TECHNIQUE

One of the instance selection techniques proposed in this study is inspired by the flower pollination algorithm (FPA) [37]. More details of the technique are presented next.

1) FLOWER POLLINATION INSTANCE SELECTION ALGORITHM

Our flower pollination instance selection algorithm (FPISA) is inspired by the process of pollinating flowering plants. Each flower (or solution) consists of N number of instances, where N is user-defined. Pseudocode for the FPISA is shown in Algorithm 1. The FPISA begins by initializing each pollen solution (line 3) and defining a probability that controls the switch between global and local pollination (line 7). Moreover, all the initialized solutions are evaluated, and the best solution is retained (line 15). Then the FPISA continues searching the solution space by performing global or local pollination. Local pollen solutions are generated (using Equation (1)) [37], if a user-defined probability switch is less than a randomly generated number (lines 19 to 23). Otherwise, global pollen solutions are generated using Equation (2) [37]. Next, the new solutions are evaluated, and the global best solution is updated if a better solution has been found. This process is repeated until a user-defined maximum is reached. The algorithm is also terminated if it converges on a solution (lines 28 to 30). After termination, the solution selected by the best flower is used to build the SVM model (line 39). Prior to training, if the solution size is less than a user-defined threshold, further instances, which have not been previously selected, are selected from the training subset and added to the solution space. This is to ensure that the total number of training instances is not less than the minimum pre-defined value (lines 34 to 36). Also, for the FPISA, the rounding function is used to convert continuous values to binary values and the parameters recommended in the report by Yang [37] are used in the experiments.

$$x_i^{t+1} = x_i^t + \in \left(x_j^t - x_k^t\right), \tag{1}$$

where x_j^t refers to pollen j at iteration t, x_k^t refers to pollen k at iteration t. They refer to pollen grains from different flowers. $\in$ is a constant, drawn from the range [0, 1].

$$x_i^{t+1} = x_i^t + L\left(x_i^t - g_*\right), \tag{2}$$

where X_i^t refers to vector x_i at different iteration t, and g_* refers to the current best solution in iteration t. Also, L refers to Levy flight, which can be drawn from a levy distribution given in equation (3) [37].

$$L \sim \frac{\lambda\Gamma(\lambda)\sin\left(\pi^{\lambda}/_{2}\right)}{\pi}\frac{1}{S^{1+\lambda}}, \quad (s \gg s_0 > 0), \tag{3}$$

$\Gamma(\lambda)$ is a standard gamma function, valid for huge steps, $s > 0$.

FPA was designed to handle continuous problem, however, in this research, the rounding-off approach, shown in equation (4) is used to convert each flower position to a binary value.

$$\mathrm{X_i^t} = \begin{cases} 1 & \text{if } \mathrm{V_i^t} > 0.5, \\ 0 & \text{otherwise}, \end{cases} \tag{4}$$

where x_i^t and v_i^t refers to different flower position and velocity at different iterations. t refers to dimension.

B. SOCIAL SPIDER-INSPIRED TECHNIQUE

The social spider algorithm (SSA) is a recent NI-based swarm intelligence algorithm, proposed by James and Li [38]. In this paper, a social spider-based instance selection technique (called SSISA) is designed for improving SVM predictive accuracy and training speed. This section presents a description of the social spider-based technique.

1) SOCIAL SPIDER INSTANCE SELECTION ALGORITHM

Our social spider instance selection algorithm (SSISA) is inspired by the foraging behaviour of social spiders. As shown in Algorithm 2, the SSISA begins by initializing all parameters and generating an initial solution of N spiders, where each spider consists of d instances (lines 4 and 8). The vibration intensity for each spider is also initialized (line 7). In addition, the fitness score for each spider is calculated and the spider with the best fitness value is stored (line 9). Then new solutions are generated by moving each spider to different positions on the web (lines 11 to 35). Each spider movement causes a vibration, as calculated in Equation (5) [38]. Typically, spiders capture prey based on propagated vibrations, and they attack the prey direction (or source of vibration) if the vibration is within a defined frequency range [38], [39]. In SSISA, if the vibration generated by the current solution is greater than a pre-defined target vibration, then the target vibration is updated with the best vibration. In addition, a random number is generated, and the instance mask is updated if the random number is greater than a pre-defined threshold (line 20 to 22). Further, the position of each spider is generated (line 24) and the fitness value for the newly generated solutions are computed, and the current best solution is compared to the global best solution (line 31). If it is better, it is retained, otherwise it is

Algorithm 1 Flower Pollination Instance Selection Algorithm

Input: *MaxG*, *N*, *NRuns*, *Min*, *FT*
Output: GB

1 *TrainingSubset* ← *RandomSelect(TrainingDataset) /*randomly select training subset from trainingDataset*/*
2 *FPISA (TrainingSubset) /*Start instance selection using training subset*/*
3 Initialize Parameters */*initialize all the algorithm parameters*/*
4 Define $G(x)$ for flowers */*define fitness function for both filter and wrapper-based FPISA*/*
5 Define PS, $PS \in [0, 1]$ */*define probability switch for flowers*/*
6 **for** $a = 1$ *to N*
7 Initialize solution for $spider_a$ */*initialize the solution for all flowers in the solution space*/*
8 **end** for
9 Evaluate $G(x)$, and select *CB /*Evaluate the objective function for all solutions & select the current best*/*
10 *GB* ← *CB /*Retain the current best solution*/*
11 **while** ($p <$ *MaxG*) */*Start searching for new pollen solutions */*
12 **for** $k = 1$ *to N*
13 *R* ← *RandomNumber() /*generate random number R, whereR* $\in [0, 1]$**/*
14 **if** $R > PS$ */*if this is true, perform global pollination*/*
15 **for** $l = 1$ *toDim /*define levy flight factor for global pollinators*/*
16 Randomly generate LF vector for each dimension
17 **end** *l*
18 Perform global pollination using $x_i^{t+1} = x_i^t + LF\left(x_i^t - g_*\right)$,
19 **else** */*perform local pollination*/*
20 *R* ← *RandomNumber() /*generate random number R, whereR* $\in [0, 1]$**/*
21 Randomly select two solutions, x_j^t, x_k^t from population
22 Perform local pollination using $x_i^{t+1} = x_i^t + \in \left(x_j^t - x_k^t\right)$
23 **end** if
24 Convert solutions to binary using equation (4)
25 **end** k
26 Evaluate $G(x)$ */*evaluate the fitness value for all the new solutions*/*
27 *GB* ← *CB /*Update the global best with the current best solution*/*
28 **if** $GB > FT$
29 **end while***/*Stop algorithm if global best is greater than a pre-defined fitness threshold*/*
30 **end** if
31 p++
32 **end** while
33 *NS* ← *GB*
34 **if** $NS <$ *Min /*Add more instances if the number of instances is less than a user-defined threshold*/*
35 *AddInstances(GB) /*Add (Min-NS) instances to the instances selected by the global best*/*
36 **end** if
37 Output *GB /*Output the global best solution*/*
38 **end**
39 Perform Parameter Optimization and *TrainSVM(GB) /*Train SVM on the instances selected by FPISA*/*

discarded. The process is performed repeatedly until a stop criterion is reached. Finally, after the algorithm terminates, the instances selected by the best spider solution are used to train SVM and the predictive accuracy is displayed. Before training, the number of instances selected by the best solution are checked to ensure that they are not less than the minimum threshold (lines 36 to 38). Pseudocode for SSISA is shown in Algorithm 2. SSA is designed to work in a continuous space, hence SSISA uses the sigmoid function (defined in Equation (6)) to convert continuous values to binary form.

$$I\left(P_s, P_s, t\right) = log\left(\frac{1}{f\left(P_s\right) - C} + 1\right), \tag{5}$$

where P_s, refers to spider position at time t, $I\left(P_s, P_s, t\right)$ refers to intensity of vibration generated by spider at source position, $f\left(P_s\right)$ refers to fitness of spider at time t, and C is a

Algorithm 2 Social Spider Instance Selection Algorithm

Input: NF, $MaxG$, N, NRuns, Pm, FT
Output: GB

TrainingSubset ← *RandomSelect(TrainingDataset)* /**randomly select training subset from TrainingDataset**/
SSISA (TrainingSubset) /**start instance selection**/
  Define $G(x)$for spiders /**pass the selected training subset to FPISA for processing**/
  Initialize Parameters /**initialize all the algorithm parameters**/
  for $a = 1$ *to* N
    Initialize solution for $spider_a$ /**initialize the solution for all spiders in the solution space**/
    Initialize vibration (TV) for $spider_a$ /**generate initial vibration for each spider**/
  end for
  Evaluate$G(x)$, /**evaluate the fitness of the initial solution**/
  $GB \leftarrow CB$ /**if current best is greater than global best, update the global best solution**/
  while ($p < MaxG$) /**start search for more solution**/
    for $k = 1$ *to* N
      Calculate VI generated by all spiders and select GBV /*select the best bat vibration*/
      if $GBV > TV_k$/**if the best vibration is greater than a user defined target vibration**/
        $TV_k = G$BV /**update the target vibration**/
      end if
      Update Tot_k /**keep track of frequency of vibration change**/
      for $a = 1$ *to* D
        Generate Random Number, R where $R \in [0, 1)$
        If $R1 > P$
          Update $dimension_a$ for $spider_k$ /**update dimension mask */*
        end if
      end a
      Generate new position for $spider_k$
      Do Random Walk, and handle violated boundary constraints
      Convert $spider_k$ to binary using sigmoid
    end k
    Evaluate G(x) for new $solution_a$ and generate vibration for $spider_a$
    Convert $spider_a$ to binary using sigmoid function
    Evaluate $G(x)$ for new solutions, and update GB accordingly
    if $GB > FT$
      end while/**Stop algorithm if global best is greater than a pre-defined fitness threshold**/
    end if
    p++
  end while
  if $NS < Min$
    update GB by adding (Min - NS) instances to GB
  end if
  Output GB
end SSISA
Perform Parameter Optimization and Train SVM model on instances selected by GB

constant value, where all $f(P_s) > C$.

$$S\left(V_i^t\right) = \frac{1}{1 + e^{-V_i^t}}, \tag{6}$$

Each spider position is updated by equation (7):

$$X_i^t = \begin{cases} 1 & if\ rand() \leq S\left(V_i^t\right), \\ 0 & otherwise, \end{cases} \tag{7}$$

where $rand()$ is a random number selected from the range, [0, 1].

C. FITNESS FUNCTION

The fitness function used by the filter- and wrapper-based techniques is discussed in this section. The fitness functions are designed for both FPISA and SSISA, and the results obtained show that they can be used by other nature-inspired algorithms.

1) FILTER-BASED FITNESS FUNCTION

The fitness function for the two proposed filter-based techniques is shown in Equation (8). It is designed according to the criteria of percentage reduction. The fitness function allocates more weight to solutions with good percentage reduction. Evaluation of the fitness function starts by computing the total number of instances of each agent (δ) in the solution space. The evaluation continues by calculating the number of instances selected by each agent (τ). For each agent, each instance position is either 0 or 1, where 1 indicates that an instance is selected by an agent and 0 indicates otherwise. Hence, the total number of instances selected by an agent is computed by totalling all the ''ones'' in the instance mask of the agent. Finally, the fitness value is then calculated using Equation (8).

$$fitness_i = \left(100 * \frac{\delta - \tau}{\delta}\right) \tag{8}$$

where δ = total number of instances in an agent and τ = number of selected instances in an agent.

To avoid a scenario where total number of selected instance (τ) is zero, the proposed techniques are designed to enforce selection of N instances, where N is user defined. Hence, for example, if B instances are selected by an agent, and B is less than N, then (N - B) additional instances, which have not been previously selected, will be selected and added to the number of instances already selected by the agent.

2) WRAPPER-BASED FITNESS FUNCTION

The fitness function used for the two proposed wrapper-based techniques, as defined in Equation (9), is different from the fitness function for the filter-based techniques. The main goal of the wrapper-based techniques is to select reduced instances with optimized predictive accuracy. Therefore, the wrapper-based fitness function is computed by firstly calculating the predictive accuracy of all potential solutions in the solution space. That is, for each candidate solution in the solution space (i.e. for each reduced subset), a classification model is built by training the reduced subset on the SVM classifier. Subsequently, the model is tested by evaluating it on a test dataset, and the resulting predictive accuracy is utilized as the fitness score of the solution. The solution with the highest fitness score is the solution that produce the best predictive accuracy. Finally, the solution with the highest fitness score (that is, the solution that produces the best predictive accuracy) is selected and used to construct the final classification model.

$$fitness_i = \alpha_i \tag{9}$$

where α_i is the predictive accuracy for each solution.

IV. EXPERIMENTAL STUDY AND DATASET

The proposed techniques were evaluated on 20 small-scale datasets and 10 medium or large-scale datasets. All the small-scale datasets were obtained from the well-known University of California, Irvine (UCI) dataset repository [40]. The UCI machine learning (ML) repository consists of many widely used datasets, provided for experimental evaluation of ML algorithms. The medium and large-scale datasets were also obtained from UCI, with the exception of USPS, which can be obtained from the LIBSVM data repository [41] – the SVM library used in this study. Furthermore, due to the high dimensionality of USPS and Isolet, in a similar way to Chen *et al.* [25], the number of features in Isolet and USPS were reduced by principal component analysis (PCA) to 150 and 80 features, respectively. For each dataset, all the features were converted to the input format required by LIBSVM [42].

Given N training instances, using the entire training set for training is time consuming. Brighton and Mellish [2] noted that training a classifier on a reduced subset (void of superfluous or harmful instances) will not significantly affect the classification accuracy of a classifier. Rather, it can lead to similar or improved classification accuracy. On this basis, all the proposed filter-based techniques were designed to use only a subset of the entire training set for instance selection. That is, for all experiments, n instances were passed to the instance selection techniques for processing, where n refers to the subset size. Besides, for all the experiments, different subset sizes and number of particles (NP) were evaluated, and the results for the best subset sizes and NP are reported. The subset sizes used for the experiments are reported in Table 1. Unlike the filter-based techniques, the wrapper-based techniques are designed to search the entire training set for relevant instances. Other parameters used for FPISA and SSISA are reported in Tables 2 and 3, respectively. For all the experiments performed on the small-scale datasets, 10-times, 10-fold cross validation was used, while for the large-scale datasets, each experiment was performed 10 times. Cross validation was not performed for the large-scale datasets because, all the datasets were originally divided into training and test parts, hence performing cross validation was not necessary.

A. EXPERIMENTAL SETUP

All the experiments performed on the small-scale datasets were performed on a computer operating on Windows 7, 64 bits. The computer had 8 GB RAM and runs on a processor speed of 3.10 GHz. For the medium or large-scale datasets, the experiments were performed on a computer with the following specifications: Windows 10, Core i5, 1.7 GHz, 64 bits operating system and 8.00 GB RAM. The parameters used for FPA and SSA are reported in Tables 2 and 3; these are the same parameters as used by James [43] and Yang [44]. In addition, the radial basic function (RBF) kernel was used for all experiments; it requires selection of two parameters, C and gamma. Hsu *et al.* [45] suggested using exponentially growing sequence of C and gamma. In this study, different parameter pairs were evaluated for each dataset, and the best parameter pair (as reported in Table 1) was used for training.

TABLE 1. Summary of dataset.

	Dataset	Size	#Features	#Class	#Tra/#Test	C	gamma	*k*	*SubSize(CBD)*	*SubSize(filter)*
					Small-scale Datasets					
1	Abalone	4177	8	29	-	1024	0.015625	300	500	700
2	Balance Scale	625	4	3	-	1024	0.015625	200	350	350
3	Breast Tissue	106	10	6	-	1024	0.015625	30	60	40
4	Bupa	345	7	2	-	1024	0.015625	100	200	300
5	Credit-g	1000	20	2	-	1024	0.015625	200	400	900
6	Cleaveland	303	13	5	-	8	0.001953	50	100	200
7	Ecoli	336	8	6	-	8	0.03125	50	100	200
8	Glass	214	10	7	-	8	0.03125	50	100	200
9	Hungarian	294	13	5	-	8	0.001953	100	200	200
10	Iris	150	4	4	-	8	0.125	30	60	200
11	Liver	345	7	2	-	8	0.03125	100	150	300
12	Pima Indians	768	8	2	-	8	0.001953	150	300	600
13	Post Operative	87	8	3	-	0.5	0.0078125	20	40	40
14	Transfusion	748	5	2	-	2	0.5	150	300	600
15	Vertebral-3c	310	6	3	-	8	0.03125	70	130	250
16	Voting	435	16	2	-	8	0.001953	80	150	300
17	Waveform	5000	21	3	-	8	0.001953	300	500	700
18	Wine	178	13	3	-	8	0.001953	40	70	150
19	Yeast	1484	8	10	-	2	0.125	200	450	900
20	Zoo	101	17	8	-	8	0.03125	20	40	100
					Large- or Medium-scale Datasets					
21	Pendigit	10,992	16	26	7494/3498	8	0.125	700	1000	3000
22	Letter	20,000	16	26	16000/4000	8	0.125	800	1200	11000
23	OptDigits	5620	64	10	3823/1797	8	0.03125	700	1000	3000
24	USPS	9298	256	10	7291/2007	8	0.001953	700	1000	3000
25	Isolet	7797	617	26	6238/1559	8	0.001953	700	1000	5000
26	Mushroom	8124	22	2	6500/1624	8	0.125	300	500	2000
27	Page-blocks	5473	10	5	4379/1094	8	0.03125	300	500	2000
28	Shuttle	58000	9	7	43500/14500	8	0.125	700	1000	4000
29	Twitter	140707	77	2	112566/28141	8	0.001953	1500	2000	10000
30	Landstat	6435	36	7	4435/2000	8	0.125	700	1000	3000

Key:#Tra/#Test – size of training/test dataset, *k*,, *SubSize(CBD)* – *k* values, subset size used for CBD algorithm, SubSize(filter) – subset size used for the filter-based FPISA and SSISA

TABLE 2. Parameter used for FPISA.

Population Size (Filter)	Population Size (Wrapper)	Probability Switch	N_g(Filter)	N_g(Wrapper)
5	5	0.8	5	3

Key: N_g = *Number of generations*

TABLE 3. Parameter used for SSISA.

Population Size (Filter)	Population Size (Wrapper)	Attenuation Rate	Probability Change	Assigning Probability	N_g(Filter)	N_g(Wrapper)
10	5	1	0.7	0.1	5	3

Key: N_g = *Number of generations*

B. EXPERIMENTAL RESULTS AND DISCUSSION

This section presents and discusses the experimental results produced by FPISA, SSISA, and the compared techniques. Firstly, the results of standard SVM, MCIS [25] and CBD [3], are discussed. In the first subsection (Section IV.C), the results of the proposed filter-based techniques using the small dataset are discussed and compared to the results of the standard SVM, and two filter-based instance selection techniques

TABLE 4. Filter-based techniques on small-scale datasets – accuracy & storage.

Dataset	Average Prediction Accuracy (%)					Average Storage Reduction Percentage (%)				
	SSISA	FPISA	SVM	CBD	MCIS	SSISA	FPISA	SVM	CBD	MCIS
Abalone	54.156	54.259	56.283	54.856	55.748	8.747	8.675	100	13.32	41.674
Balanced Scale	94.323	93.79	98.548	96.387	93.742	28.39	28.534	100	53.71	41.611
Breast Tissue	65.8	62.2	64	61.9	61.1	44.168	44.168	100	66.25	42.167
Bupa	69.324	69.588	70.294	67.588	66.206	43.903	44.55	100	65.254	41.859
Credig-g	68.27	68.71	69.7	69.98	68.59	47.173	47.301	100	44.444	41.667
Cleaveland	60.31	61.62	63.793	58.759	61.724	33.262	33.956	100	38.214	41.765
Ecoli	85.668	85.576	87.576	85.333	84.303	29.856	30.131	100	33.603	41.767
Glass	63.762	64.857	68.095	65.476	62.714	43.29	44.464	100	52.8	41.764
Hungarian	64.31	64.655	67.241	62.448	63.759	33.386	34.144	100	38.256	41.737
Iris	96	95.333	95.333	95.2	95.467	41.919	29.763	100	44.444	42.222
Liver	68.912	69.029	72.941	68.441	64.824	44.083	44.591	100	48.941	41.859
Puma Indian	76.645	76.224	77.105	75.671	75.895	40.628	40.683	100	43.809	41.676
Post Operatives	72.5	72.5	72.5	72.5	72.5	55.063	55.063	100	55.063	41.677
Transfusion	73.514	73.514	78.919	77.135	78.243	42.197	42.266	100	44.992	41.752
Vertebral-3c	84	84.516	86.452	82.161	84.065	39.62	40.172	100	46.595	41.935
Voting	91.628	91.629	95.814	95.442	95.372	35.149	35.283	100	38.71	41.858
Waveform	84.582	84.608	87.12	85.334	86.412	7.3	7.247	100	11.111	41.667
Wine	97.235	96.941	98.824	97.647	96	41.531	42.142	100	58.531	41.871
Yeast	58.164	58.055	60	58.094	56.5	36.993	36.68	100	39.063	41.667
Zoo	92.3	92.9	95	90.7	90.2	44.984	45.15	100	44.396	42.176

(CBD and MCIS). Standard SVM refers to SVM without data reduction. MCIS and CBD are existing filter-based instance selection techniques used here primarily for comparison. Then in Section IV.D, the results of the proposed wrapper-based techniques for the small dataset are presented and compared to the standard SVM and a wrapper-based instance selection technique (ADRMiner [12]). This is followed by the proposed techniques being evaluated on medium and large datasets, with results being reported in Section IV.E. Finally, the result is summarized in Section IV.F and statistical analysis is reported in Section IV.G.

As shown in Table 1, for each dataset, the following information are reported: name of dataset (Dataset), size of dataset (Size), number of attributes (#Attributes), number of classes (#Classes), number of ham instances (#Ham), number of training and testing samples (#Tra/#Tes) per dataset, k values and subset size used for the CBD algorithm (SubSize(CBD)) and the subset size used for the filter- and wrapper-based FPISA and SSISA (SubSize(filter)). Subset size represents the number of training instances selected by the algorithms, while K refers to the number of nearest neighbours used by CBD algorithm for training. In addition, for each dataset, the grid algorithm is used to perform parameter optimization, and the C and gamma pair that produced the best result for each dataset is used for training. The best C and gamma pair for each dataset is reported in Table 1. As shown, the proposed techniques were evaluated using the following criteria: average prediction accuracy (APA), storage reduction percentage (Storage), algorithm time (alg-time), and training time (tr-time). The storage reduction percentage is the ratio of instances selected (in percentage) by each algorithm. Moreover, the proposed techniques are compared to two filter-based instance selection techniques (CBD [21] and MCIS [25]) and one wrapper-based technique (ADR-Miner [12]). Different subset sizes and K values were tested for CBD algorithm, and the values that produced the best results are reported in this study.

Attenuation rate, probability change and assigning probability are user-defined parameters. Attenuation rate controls the reduction rate of a spider vibration intensity over distance. Probability change describes the probability of changing a spider mask. Assigning probability controls the probability of assigning zero or one to each bit of a spider mask. The three aforementioned parameters are all user-defined. Attenuation rate is defined in $(0, \infty)$. Probability change and assigning probability are both defined in (0, 1).

C. EXPERIMENT 1: FILTER-BASED TECHNIQUES ON SMALL-SCALE DATASETS

Tables 4 and 5 show the average prediction accuracy, storage percentage, training time (in seconds) and algorithm time (in seconds) achieved by FPISA, SSISA, standard SVM, CBD and MCIS. As shown in Table 5, Figures 2 and 3,

TABLE 5. Filter-based proposed techniques on small-scale datasets – average time (in seconds).

Dataset	Average Training Time					Average Algorithm Time			
	SSISA	FPISA	SVM	CBD	MCIS	SSISA	FPISA	CBD	MCIS
Abalone	0.2963	0.324	18.293	0.3922	3.836	0.653	0.66	69.03654	0.119
Balanced Scale	0.0537	0.0535	0.589	0.0667	0.057	0.0681	0.03706	22.24896	0.011
Breast Tissue	0.0108	0.0102	0.022	0.0114	0.008	0.0038	0.00212	0.00194	0.002
Bupa	0.1182	0.0899	0.4	0.1206	0.089	0.0375	0.0113	4.01105	0.007
Credig-g	0.3281	0.3759	2.481	0.1695	0.169	0.0683	0.02272	31.31051	0.14
Cleaveland	0.02	0.0119	0.062	0.0081	0.031	0.0288	0.01126	1.29427	0.012
Ecoli	0.0089	0.0116	0.045	0.0064	0.022	0.026	0.01528	1.4711	0.01
Glass	0.0239	0.0121	0.043	0.0098	0.017	0.0298	0.00815	0.88106	0.005
Hungarian	0.0096	0.0104	0.049	0.0087	0.02	0.0218	0.00984	1.26314	0.008
Iris	0.0053	0.0055	0.01	0.0072	0.007	0.0127	0.00429	0.27169	0.003
Liver	0.0141	0.012	0.062	0.0097	0.016	0.0262	0.0092	2.28548	0.009
Puma Indian	0.0367	0.0295	0.143	0.0212	0.037	0.0745	0.027	16.18597	0.021
Post Operatives	0.0083	0.0061	0.016	0.0062	0.009	0.0074	0.00147	0.07244	0.003
Transfusion	0.0065	0.0058	0.234	0.0234	0.059	0.0668	0.0257	16.42126	0.019
Vertebral-3c	0.0093	0.0132	0.036	0.0086	0.013	0.0261	0.01812	2.06013	0.008
Voting	0.0088	0.01	0.06	0.0064	0.018	0.0395	0.01468	2.68905	0.018
Waveform	0.0414	0.0566	3.364	0.0561	0.883	0.7828	0.91688	78.78151	0.239
Wine	0.007	0.0102	0.02	0.0047	0.012	0.0166	0.00769	0.56892	0.006
Yeast	0.1528	0.1621	0.873	0.0886	0.2	0.1562	0.12267	79.01742	0.028
Zoo	0.0082	0.01	0.033	90.80	0.012	0.009	0.00356	0.1	0.009

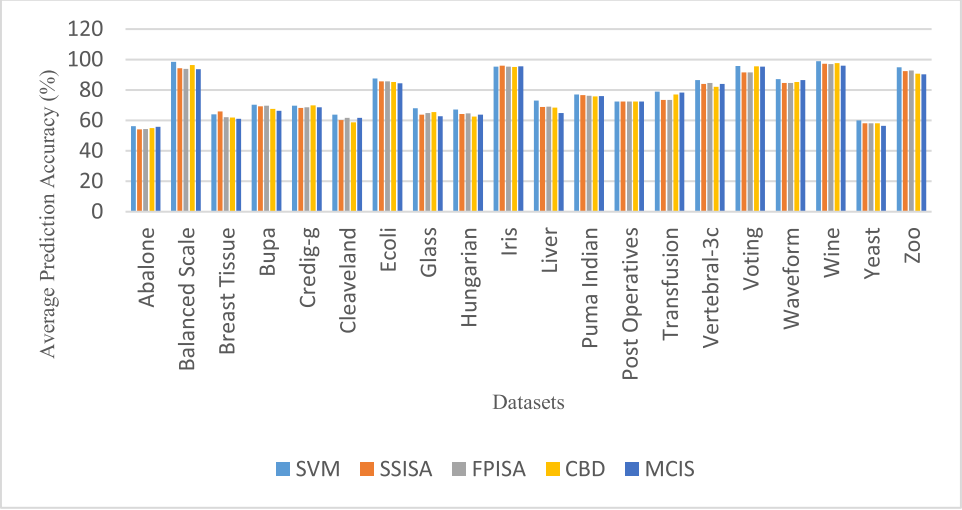

FIGURE 2. Filter-based techniques on small-scale datasets – average prediction accuracy.

the proposed filter-based techniques consistently improved the training speed of SVM in all the datasets (20 out of 20) used for evaluation, without significantly affecting SVM prediction accuracy. Specifically, FPISA and SSISA improved the training speed of SVM by an average of 76% and 75%, respectively. The training speed improvement is calculated using $\left((\alpha - \beta)\,/\alpha\right)*100$, where α *and* β refers to the training speed produced by the standard and hybrid models, respectively. Furthermore, the two filter-based techniques achieved an average data reduction of 37%, implying that only 37% of the dataset is required to achieve improved speed-accuracy trade-off. Moreover, FPISA and SSISA require very little time to perform data reduction. As shown in Table 5, the two algorithms require an average of less than 1 second to select relevant instances from small datasets. This shows the improved speed and storage reduction capacity of FPISA and SSISA.

The proposed techniques are further compared to CBD and MCIS. As shown in Table 4, in terms of prediction accuracy, FPISA and SSISA outperformed CBD in 11 out

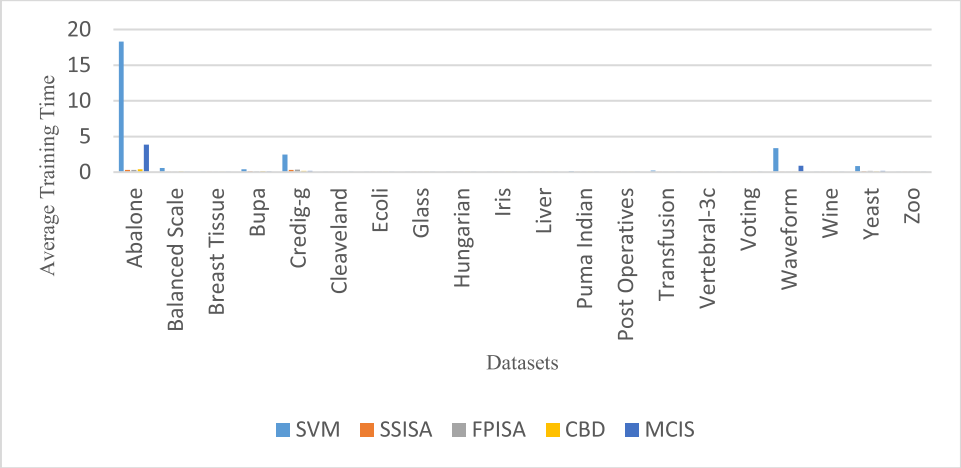

FIGURE 3. Filter-based techniques on small-scale datasets – average training time (in seconds).

of 20 datasets. Also, in terms of storage reduction, FPISA and SSISA outperformed CBD in 17 out of 20 datasets. Besides, in terms of prediction accuracy, FPISA and SSISA outperformed MCIS in 15 out of 20 datasets and simultaneously produced better storage reduction percentage in 10 out of 20 datasets. In addition, as shown in Table 5, FPISA and SSISA achieved better algorithm speed compared to CBD and achieved algorithm speed comparable with that of MCIS. Overall, the filter-based FPSIA and SSISA achieved better results compared to CBD and MCIS, showing that the two hybrid algorithms are competitive and better alternatives to the standard ones.

D. EXPERIMENT 2: WRAPPER-BASED TECHNIQUES ON SMALL-SCALE DATASETS

The wrapper-based FPISA and SSISA were validated on 20 small-scale datasets. As mentioned above, unlike the filter-based techniques, which search through a subset of the dataset, the wrapper-based techniques are designed to search through the entire training dataset for relevant instances. Tables 6 and 7, and Figures 4 and 5 show the predictive accuracy, training speed and storage reduction percentage produced by FPISA, SSISA, standard SVM, CBD and MCIS. As shown in Table 6, the proposed techniques reduced the storage size of the evaluated datasets by an average of 50% and, in most cases, simultaneously improved SVM prediction accuracy. This shows that only 50% of the training dataset is required to achieve accuracy preservation, as achieved in most cases by FPISA and SSISA. In addition, FPISA and SSISA achieved an average training speed of less than 0.4 second and improved the training speed of SVM by an average of approximately 75% (SSISA) and 77% (FPISA). This implies that they require very little training time to produce fast and improved models. In addition, the techniques are very fast, as they achieved an average algorithm speed of approximately 18 seconds (SSISA) and 2 seconds (FPISA). This makes them effective and fast approaches for improving the speed and predictive performance of SVM.

Furthermore, the wrapper-based techniques were compared to CBD and MCIS. As shown in Table 6, they produced better prediction accuracy than CBD and MCIS in 15 out of the 20 datasets. Besides, they also outperformed CBD in algorithm speed. In terms of storage reduction percentage and training speed, all the algorithms produced similar results; none of them consistently outperform each other. This shows the efficiency and robustness of the wrapper-based FPISA and SSISA; they compare well with filter-based techniques. In addition, the proposed wrapper-based techniques were compared to an existing state-of-the-art wrapper-based instance selection technique, called ADR-Miner [12]. ADR-Miner was designed to use two classification algorithms for evaluation. One classification algorithm was used to evaluate the quality of each candidate solution and the second classification algorithm is used to build the final model. To ensure a fair comparison, we compare FPISA and SSISA to the algorithm combination that used SVM at both the instance-selection and model-construction stage. This is because FPISA and SSISA also used SVM at both stages. In Table 8, for each dataset, the best predictive accuracy is underlined. It can be seen that the proposed wrapper-based techniques outperform ADR-Miner in 7 out of 10 datasets. This shows their superiority in preserving SVM prediction accuracy.

E. EXPERIMENT 3: FILTER-AND WRAPPER BASED TECHNIQUES ON MEDIUM OR LARGE-SCALE DATASETS

The proposed filter- and wrapper-based techniques were evaluated on 10 medium or large-scale datasets; specifically, the Pendigit, Letter, OptDigits, USPS, Isolet, Mushroom, Page-blocks, Shuttle, Twitter, and Landstat datasets. All of these had been previously divided into training and test sets by their providers, except for the Page-block, Mushroom and Twitter datasets. These three were divided into training and test subsets by ourselves, using 80% of the datasets for training and 20% for testing. Also, due the limited processing

TABLE 6. Wrapper-based techniques on small-scale datasets – accuracy & storage.

Dataset	Average Prediction Accuracy (%)					Average Storage Reduction Percentage (%)				
	SSISA	FPISA	SVM	CBD	MCIS	SSISA	FPISA	SVM	CBD	MCIS
Abalone	57.573	55.645	56.283	54.856	55.748	53.14	49.109	100	13.32	41.674
Balanced Scale	96.613	95.823	98.548	96.387	93.742	52.862	47.871	100	53.71	41.611
Breast Tissue	73.8	61.4	64	61.9	61.1	53.103	45.797	100	66.25	42.167
Bupa	75.765	68.588	70.294	67.588	66.206	52.704	46.918	100	65.254	41.859
Credig-g	72.18	68.17	69.7	69.98	68.59	52.367	48.304	100	44.444	41.667
Cleaveland	62.69	61.276	63.793	58.759	61.724	52.477	46.632	100	38.214	41.765
Ecoli	88.848	86.545	87.576	85.333	84.303	52.244	47.426	100	33.603	41.767
Glass	71.81	64.762	68.095	65.476	62.714	52.671	46.241	100	52.8	41.764
Hungarian	65.207	64.759	67.241	62.448	63.759	52.295	47.042	100	38.256	41.737
Iris	98.733	96.133	95.333	95.2	95.467	51.2	45.178	100	44.444	42.222
Liver	73.912	69.735	72.941	68.441	64.824	51.963	47.32	100	48.941	41.859
Puma Indian	76.776	76.776	77.105	75.671	75.895	52.886	48.053	100	43.809	41.676
Post Operatives	72.5	72.5	72.5	72.5	72.5	57.052	55.903	100	55.063	41.677
Transfusion	73.514	73.514	78.919	77.135	78.243	50.552	48.368	100	44.992	41.752
Vertebral-3c	87.29	84.742	86.452	82.161	84.065	51.355	46.599	100	46.595	41.935
Voting	87.442	87.442	95.814	95.442	95.372	50.616	47.953	100	38.71	41.858
Waveform	86.78	86.572	87.12	85.334	86.412	52.954	49.104	100	11.111	41.667
Wine	98.118	97.235	98.824	97.647	96	50.016	45.551	100	58.531	41.871
Yeast	61.438	57.891	60	58.094	56.5	53.393	48.277	100	39.063	41.667
Zoo	97.3	92	95	90.8	90.2	51.889	46.514	100	44.396	42.176

TABLE 7. Wrapper-based techniques on small-scale datasets – average time (in seconds).

Dataset	Average Training Time					Average Algorithm Time			
	SSISA	FPISA	SVM	CBD	MCIS	SSISA	FPISA	CBD	MCIS
Abalone	5.12	4.02	18.293	0.3922	3.836	116.81	12.669	69.03654	0.119
Balanced Scale	0.073	0.074	0.589	0.0667	0.057	8.169	1.329	22.24896	0.011
Breast Tissue	0.008	0.008	0.022	0.0114	0.008	1.712	0.296	0.00194	0.002
Bupa	0.114	0.087	0.4	0.1206	0.089	5.006	0.609	4.01105	0.007
Credig-g	0.411	0.366	2.481	0.1695	0.169	19.639	2.572	31.31051	0.14
Cleaveland	0.013	0.014	0.062	0.0081	0.031	4.257	0.556	1.29427	0.012
Ecoli	0.013	0.013	0.045	0.0064	0.022	4.113	0.557	1.4711	0.01
Glass	0.01	0.008	0.043	0.0098	0.017	2.703	0.344	0.88106	0.005
Hungarian	0.017	0.014	0.049	0.0087	0.02	4.532	0.598	1.26314	0.008
Iris	0.004	0.003	0.01	0.0072	0.007	1.029	0.239	0.27169	0.003
Liver	0.017	0.012	0.062	0.0097	0.016	5.074	0.667	2.28548	0.009
Puma Indian	0.041	0.037	0.143	0.0212	0.037	10.783	1.571	16.18597	0.021
Post Operatives	0.004	0.006	0.016	0.0062	0.009	1.315	0.286	0.07244	0.003
Transfusion	0.004	0.004	0.234	0.0234	0.059	9.081	1.015	16.42126	0.019
Vertebral-3c	0.008	0.009	0.036	0.0086	0.013	3.667	0.527	2.06013	0.008
Voting	0.003	0.004	0.06	0.0064	0.018	4.831	0.632	2.68905	0.018
Waveform	1.249	1.094	3.364	0.0561	0.883	141.42	16.314	78.78151	0.239
Wine	0.007	0.005	0.02	0.0047	0.012	1.346	0.355	0.56892	0.006
Yeast	0.202	0.155	0.873	0.0886	0.2	23.252	2.886	79.01742	0.028
Zoo	0.007	0.007	0.033	0.158	0.012	0.898	0.228	0.1	0.009

speed of the computer used for experiments, we only used 50% of the Twitter training subset for training. Furthermore, due to the stochastic nature of FPISA and SSISA, for all the datasets, ten different runs were performed using the training subset to build the model and the test the subset so as to evaluate the performance of the model. For each

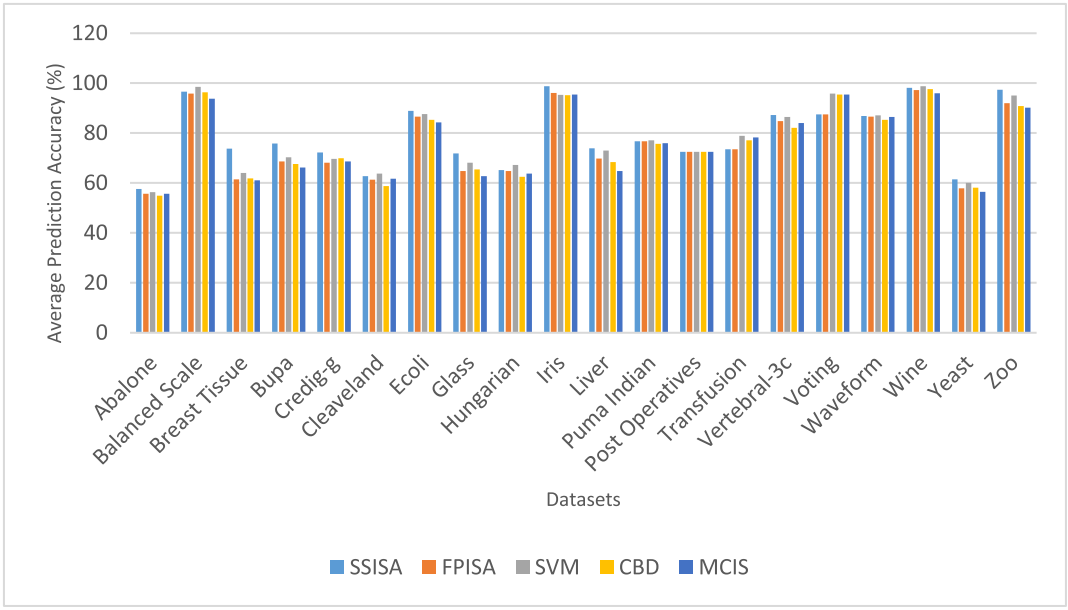

FIGURE 4. Wrapper-based techniques on small-scale datasets – average prediction accuracy.

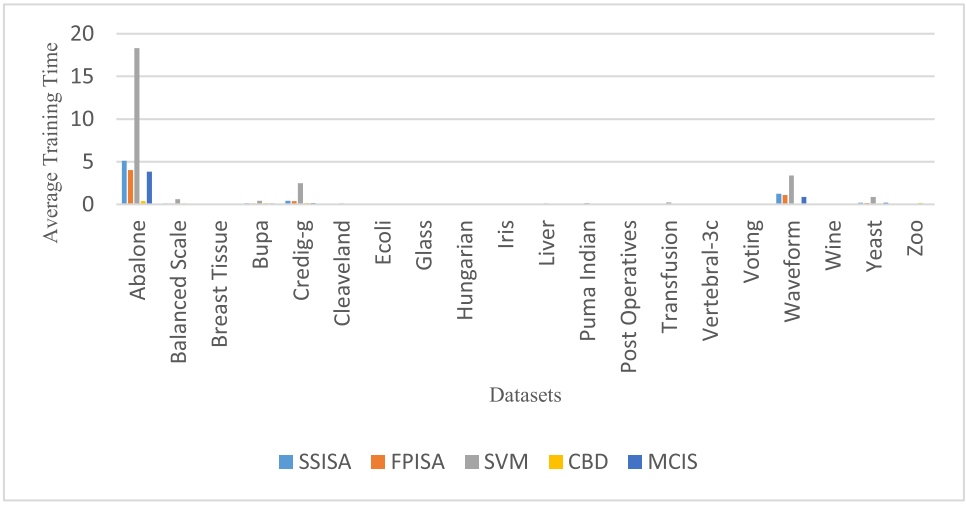

FIGURE 5. Wrapper-based techniques on small-scale datasets – average training time (in seconds).

TABLE 8. Wrapper-based proposed techniques vs ADR-Miner.

Dataset	FPISA		SSISA		ADR-Miner	
	Acc	Sto	Acc	Sto	Acc	Sto
Breast T	61.40	50.88	73.8	53.103	60.64	23.98
Credit-g	68.17	48.304	72.18	52.367	74.1	19.31
Ecoli	86.545	47.426	88.848	52.244	81.34	21.33
Glass	64.762	46.241	71.81	52.671	69.64	31.4
Iris	96.133	45.178	98.733	51.2	92.56	42.08
Liver	69.735	47.32	73.912	51.963	58.56	17.55
Transfusion	73.514	48.368	73.514	50.552	72.31	21.88
Vertebral-3c	84.742	46.599	87.29	51.355	83.55	23.30
Voting	87.442	47.953	87.442	50.616	95.46	12
Zoo	97.3	46.514	92	51.889	98.75	52.78

Key: Acc – Accuracy (%), Sto – Storage (%)

experiment, the average prediction accuracy, storage capacity, training speed, and algorithm speed are reported here. In addition, for each dataset, different experiments were performed to determine the best parameters (C, gamma, number of particles and number of generation), and the parameters that produced the best result was used for all the experiments.

1) FILTER-BASED TECHNIQUES ON MEDIUM OR LARGE-SCALE DATASETS

As shown in Table 9, the filter-based SSISA and FPISA algorithms reduced the storage size of big datasets meaningfully, by an average of 23.39% and 21.28%, respectively, without significantly affecting the accuracy of their resulting models. This shows their efficiency in data reduction and usefulness for big data processing, as they required less than 24% of big datasets to achieve competitive results. Besides, the result reveal that the proposed filter-based techniques achieved better training speed compared to standard SVM on all 10 of the datasets. Specifically, SSISA and

TABLE 9. Filter-based techniques on large-scale datasets (average accuracy & storage).

Dataset	Average Prediction Accuracy (%)					Average Storage Reduction Percentage (%)				
	SSISA	FPISA	SVM	CBD	MCIS	SSISA	FPISA	SVM	CBD	MCIS
Pentdigit	97.421	97.527	98.056	96.827	97.644	19.406	19.255	100	13.344	41.66
Letter	94.513	94.268	97.475	83.963	94.523	33.916	33.769	100	7.5	41.669
Isolet	93.611	93.515	95.959	77.524	92.848	39.301	19.907	100	16.031	41.68
USPS	93.154	93.333	95.167	91.973	94.31	19.907	19.838	100	13.716	41.668
Optdigit	95.454	95.62	96.717	94.725	95.754	38.368	37.497	100	26.157	41.669
Mushroom	99.587	99.593	100	55.111	98.153	14.909	14.843	100	7.692	41.677
Page-block	96.481	96.536	97.075	94.826	96.746	22.101	21.921	100	11.418	41.699
Shuttle	99.592	99.676	99.862	99.363	99.782	4.488	4.448	100	2.299	41.667
Twitter	94.988	95.035	96.535	95.635	96.39	8.737	8.705	100	3.553	41.666
Landstat	89.04	89.245	91.6	62.395	89.08	32.794	32.607	100	22.548	41.669

TABLE 10. Filter-based techniques on large-scale datasets - average time (in seconds).

Dataset	Average Training Time					Average Algorithm Time			
	SSISA	FPISA	SVM	CBD	MCIS	SSISA	FPISA	CBD	MCIS
Pentdigit	0.403	0.418	3.98	0.225	1.303	2.274	1.828	492.243	0.336
Letter	10.938	9.321	44.952	1.003	14.162	11.08	9.379	719.413	0.954
Isolet	9.886	1.696	34.354	1.423	11.692	1.696	1.569	590.435	1.518
USPS	2.642	2.564	19.605	1.106	7.78	2.012	1.802	488.164	2.784
Optdigit	1.77	1.343	7.287	1.014	1.688	0.8	0.666	495.149	0.685
Mushroom	0.304	0.265	4.671	0.036	0.709	1.402	1.045	67.872	0.682
Page-block	0.138	0.127	1.516	0.016	0.342	1.089	0.734	127.556	0.144
Shuttle	0.241	0.189	20.957	0.042	4.589	67.198	50.072	2172.681	2.877
Twitter	5.428	4.24	332.039	0.586	74.752	88.91	66.949	3015.985	7.581
Landstat	0.85	1.041	5.213	0.27	1.102	1.102	1.182	501.528	0.261

FPISA achieved an average training speed improvement of 86.42% and 89.61%, respectively. This implies that the proposed techniques require only 23.39% and 21.28% of the large-scale datasets to achieve comparable classification accuracy and improved training speed. Although, the standard SVM produced slightly better prediction accuracy compared to the filter based FPISA and SSISA, the latter two produced both better training speed and storage reduction percentage. Moreover, as shown in Table 10, the filter-based FPISA and SSISA techniques required only a small amount of time to process and select useful instances from big datasets, which demonstrates their relevance to big dataset processing.

The results achieved by FPISA and SSISA were also compared to those from CBD and MCIS. As shown in Tables 9 and 10, in all cases, both FPISA and SSISA achieved better prediction accuracy than CBD and produced comparable storage reduction percentage. Moreover, in all cases, FPISA achieved better storage reduction percentage than MCIS and achieved comparable prediction accuracy. Besides these advantages, FPISA and SSISA also outperform MCIS in terms of training time and outperform CBD in terms of algorithm speed.

2) WRAPPER-BASED TECHNIQUES ON MEDIUM OR LARGE-SCALE DATASETS

Tables 11 and 12 show results for the performance of the wrapper-based techniques on medium or large-scale datasets. As shown by the results, the wrapper-based FPISA and SSISA algorithms achieved average training speeds of 17.94 seconds and 17.77 seconds, respectively, and thus improved on SVM training speed by an average of 61.22% (SSISA) and 63.08% (FPISA). Moreover, the wrapper-based SSISA and FPISA reduced the storage size of the evaluated big datasets by averages of 52.57% and 49.52%, respectively. This shows that they removed an average of 50% of the irrelevant instances from the datasets without significantly affecting their quality. Moreover, in most cases, the wrapper-based FPISA and SSISA achieved prediction accuracy that is comparable with standard SVM. Overall, the wrapper-based FPISA and SSISA achieved very good prediction accuracy, training speed, storage percentage reduction, and algorithm time.

The performance of the wrapper-based FPISA and SSISA are further demonstrated by comparing this to the performance of CBD and MCIS, as reported in Tables 11 and 12 for prediction accuracy, storage capacity, training speed and

TABLE 11. Wrapper-based techniques on large-scale datasets (average accuracy & storage).

Dataset	Average Prediction Accuracy (%)					Average Storage Reduction Percentage (%)				
	SSISA	FPISA	SVM	CBD	MCIS	SSISA	FPISA	SVM	CBD	MCIS
Pentdigit	98.087	97.759	98.056	96.827	97.644	51.348	49.572	100	13.344	41.66
Letter	96.1	95.855	97.475	83.963	94.523	54.05	49.585	100	7.5	41.669
Isolet	94.567	94.099	95.959	77.524	92.848	53.679	49.58	100	16.031	41.68
USPS	94.738	94.554	95.167	91.973	94.31	51.527	49.503	100	13.716	41.668
Optdigit	96.155	95.982	96.717	94.725	95.754	54.886	49.124	100	26.157	41.669
Mushroom	99.982	100	100	55.111	98.153	50.192	49.543	100	7.692	41.677
Page-block	96.947	96.645	97.075	94.826	96.746	52.674	49.233	100	11.418	41.699
Shuttle	99.835	99.824	99.862	99.363	99.782	51.113	49.789	100	2.299	41.667
Twitter	96.494	96.455	96.535	95.635	96.39	53.913	49.76	100	3.553	41.666
Landstat	90.605	90.625	91.6	62.395	89.08	52.334	49.475	100	22.548	41.669

TABLE 12. Wrapper-based techniques on large-scale datasets - average time (in seconds).

Dataset	Average Training Time					Average Algorithm Time			
	SSISA	FPISA	SVM	CBD	MCIS	SSISA	FPISA	CBD	MCIS
Pentdigit	1.739	1.653	3.98	0.225	1.303	56.786	14.635	492.243	0.336
Letter	18.852	15.534	44.952	1.003	14.162	428.242	99.194	719.413	0.954
Isolet	14.995	12.752	34.354	1.423	11.692	318.237	68.838	590.435	1.518
USPS	8.643	9.009	19.605	1.106	7.78	206.157	53.753	488.164	2.784
Optdigit	2.875	2.293	7.287	1.014	1.688	71.955	15.336	495.149	0.685
Mushroom	1.685	1.525	4.671	0.036	0.709	24.577	10.128	67.872	0.682
Page-block	0.559	0.558	1.516	0.016	0.342	19.699	4.826	127.556	0.144
Shuttle	6.49	7.9	20.957	0.042	4.589	208.458	91.406	2172.681	2.877
Twitter	120.03	126.48	332.039	0.586	74.752	2710.133	745.969	3015.985	7.581
Landstat	1.822	1.736	5.213	0.27	1.102	50.17	13.107	501.528	0.261

TABLE 13. Average rank from friedman's non-parametric test for filter-based FPISA.

Letter ($\chi^2 = 30$, $p < 00001$)			Twitter ($\chi^2 = 30$. $p < 00001$)			Shuttle ($\chi^2 = 30$, $p < 00001$)		
Algorithm	Ranking	S.Dev	Algorithm	Ranking	S.Dev	Algorithm	Ranking	S.Dev
FPISA	2.00	1.571	FPISA	2.00	0.427	FPISA	2.00	0.028
SVM	4.00	44.929	SVM	4.00	8.962	SVM	4.00	2.137
MCIS	3.00	14.162	MCIS	3.00	6.410	MCIS	3.00	0.247
CBD	1.00	0.312	CBD	1.00	0.058	CBD	1.00	0.312

algorithm time produced by CBD and MCIS. These results indicate that the wrapper-based FPISA and SSISA outperform CBD and MCIS in prediction accuracy. Moreover, they outperform MCIS in storage reduction percentage.

F. RESULT SUMMARY

In summary, as shown in all the results, for small and big datasets, both the proposed filter-and wrapper-based SSISA and FPISA produced competitive results and can be used as better alternatives. Moreover, they significantly improved SVM training speed and prediction accuracy. In addition, the results reveal that the wrapper-based variants of FPISA and SSSIA are slower than their filter-based variants; however, they still outperform CBD (a filter-based technique) in algorithm speed and also produced similar algorithm and training compared to MCIS. This demonstrates their efficiency and usefulness in big data processing and ML speed optimization.

Furthermore, comparing SSISA to FPISA, SSISA outperforms FPISA in classification accuracy, while FPISA outperform SSISA in training speed, storage percentage and algorithm time. Moreover, comparing the filter-and wrapper-based techniques, the filter-based techniques performed

TABLE 14. Average rank from friedman's non-parametric test for filter-based SSISA.

Letter ($\chi^2 = 28.92, p < 00001$)			Twitter ($\chi^2 = 30, p < 00001$)			Shuttle ($\chi^2 = 30, p < 00001$)		
Algorithm	**Ranking**	**S.Dev**	**Algorithm**	**Ranking**	**S.Dev**	**Algorithm**	**Ranking**	**S.Dev**
SSISA	2.10	10.938	SSISA	2.00	2.901	SSISA	2.00	0.076
SVM	4.00	44.952	SVM	4.00	9.135	SVM	4.00	2.137
MCIS	2.90	14.162	MCIS	3.00	1.810	MCIS	3.00	0.247
CBD	1.00	0.312	CBD	1.00	2.501	CBD	1.00	0.013

TABLE 15. Average rank from friedman's non-parametric test for wrapper-based FPISA.

Letter ($\chi^2 = 30, p < 00001$)			Twitter ($\chi^2 = 30, p < 00001$)			Shuttle ($\chi^2 = 30, p < 00001$)		
Algorithm	**Ranking**	**S.Dev**	**Algorithm**	**Ranking**	**S.Dev**	**Algorithm**	**Ranking**	**S.Dev**
FPISA	3.00	0.671	FPISA	3.00	7.221	FPISA	3.00	1.003
SVM	4.00	4.929	SVM	4.00	8.962	SVM	4.00	2.137
MCIS	2.00	1.222	MCIS	2.00	6.410	MCIS	2.00	0.247
CBD	1.00	0.054	CBD	1.00	0.057	CBD	1.00	0.013

TABLE 16. Average rank from friedman's non-parametric test for wrapper-based SSISA.

Letter ($\chi^2 = 28.920, p < 00001$)			Twitter ($\chi^2 = 30, p < 00001$)			Shuttle ($\chi^2 = 30, p < 00001$)		
Algorithm	**Ranking**	**S.Dev**	**Algorithm**	**Ranking**	**S.Dev**	**Algorithm**	**Ranking**	**S.Dev**
SSISA	2.90	2.763	SSISA	3.00	14.941	SSISA	3.00	0.758
SVM	4.00	4.929	SVM	4.00	8.962	SVM	4.00	2.137
MCIS	2.10	1.222	MCIS	2.00	6.410	MCIS	2.00	0.247
CBD	1.00	0.312	CBD	1.00	0.058	CBD	1.00	0.013

better than the wrapper-based techniques in training speed, storage reduction percentage, and algorithm speed, while the wrapper-based techniques outperform the filter-based techniques in prediction accuracy.

G. STATISTICAL ANALYSIS

In this study, two different statistical tests; namely, Friedman non-parametric and Wilcoxon signed rank tests, are conducted with the aim of showing that the training speeds of FPISA and SSISA are statistically significantly better than the training speed of standard SVM. We deliberately conducted the statistical analysis on large-scale datasets because, SVM training complexity increases as the problem size and number of classes increases [25]. Therefore, the training speed improvement would be more obvious in datasets with many instances. SVM training time complexity is $O(n^2)$, where n refers to the number of training instances [46]. First, a Friedman non-parametric analysis test for multiple comparison was carried out on three large-scale datasets (Twitter, Letter and Shuttle), and the test show that a change in dataset size, or dimensionality, significantly alters SVM training speed. As shown in Tables 13 to 16, all the p-values are less than the 0.05. Furthermore, the Wilcoxon signed ranks test revealed that the training speed of SVM was significantly improved compared to the training speed achieved by the filter- and wrapper-based FPISA and SSISA. For example, as shown in Table 10, the analysis on the Twitter dataset reveals that SVM training speed was significantly improved when using SSISA and FPISA (5.43 seconds and 4.24 seconds, respectively) compared to 332.04 seconds needed for SVM. The test showed that the p-values for all the comparisons are less than the adjusted p value ($p < 0.0083$). Therefore, it can be concluded (with 95%-degree confidence level) that the filter- and wrapper-based FPISA and SSISA techniques significantly improved the training speed of SVM.

Moreover, as shown in the results presented in Tables 13 and 14, the filter-based FPISA and SSISA are significantly faster (in terms of training speed) than MCIS, but not CBD. However, FPISA and SSISA outperformed CBD in prediction accuracy and algorithm time. For example, as shown in Tables 9 and 10, it took CBD about 2172.68 seconds to select 2.3% instances from the Shuttle dataset, while it took FPISA 50.1 seconds to select 4.4% instances from the same dataset. This shows that FPISA requires less time to perform data reduction on large datasets than does CBD, which implies that FPISA is faster than CBD. Furthermore, as shown in Tables 15 and 16, MCIS and CBD achieved better training speeds than did the wrapper-based FPISA and SSISA. This is because MCIS and CBD are filter-based techniques, which are generally faster than wrapper-based techniques. However, the result show that the wrapper-based FPISA and SSISA still outperformed CBD, in algorithm

speed and prediction accuracy. They also outperform MCIS in prediction accuracy and storage reduction capacity. This shows that the wrapper-based techniques are more efficient in preserving prediction accuracy and reducing big datasets, which makes them competitive and preferable.

To summarise the above evaluations, it has been shown in all the results presented in Tables 4 to 16 that the proposed techniques are very good SVM speed optimizers and instance selection techniques, which demonstrates the usefulness of these NI-algorithms in instance selection and speed optimization. Such NI-algorithms can be used in combination with standard SVM and other ML algorithms to produce efficient and fast classification models. In real life applications, such as video surveillance and intrusion detection systems, that require a classifier to be trained very quickly for speedy classification of new target concepts, the filter-based techniques provide the best solutions. By contrast, the wrapper-based techniques are better suited for applications, such as email filters, that are very sensitive to slight changes in predictive accuracy.

V. CONCLUSION

In the world of growing information overload and complexities in decision-making, ML-based solutions are becoming very useful tools for many businesses. ML algorithms are known for their robustness [47], accurate data mining and classification proficiency [47], [48]. They are also known for dynamic problem solving [47]. SVM is a well-known ML algorithm that has been widely used to tackle many real-world problems, with good results. However, SVM suffers from high computational complexity, which is become most noticeable with massive datasets. This research therefore proposes two intelligent speed optimization techniques (called FPISA and SSISA), for improving SVM training speed, computational complexity, and generalization performance, without significantly affecting the SVM prediction accuracy. Accordingly, this study introduces two variants of each technique; filter-based and wrapper-based variants.

Different set of experiments were performed to evaluate the efficacy of the proposed techniques on 30 datasets. The first set of experiments was performed on 20 small-scale datasets, while the last set of experiments was performed on 10 medium- or large-scale datasets. Furthermore, the performance of the proposed techniques was compared to that of the standard SVM and three existing instance selection techniques.

Experimental results show that, in all cases, the filter-based techniques considerably improved SVM training speed, and simultaneously achieved comparable prediction accuracy for both small and medium- or large-scaled datasets. Furthermore, the results show that, in some cases, the wrapper-based techniques improved SVM training speed and simultaneously improved SVM predictive accuracy. In addition, the experimental results show that the proposed techniques produced excellent storage reduction and speed-accuracy trade-offs. Furthermore, two different statistical tests were conducted with the aim of evaluating the training speed difference between the proposed techniques and standard SVM. As shown in the test results, it can be concluded with a 95% confidence level, that the proposed techniques are significantly faster (in terms of training speed) than the standard SVM and some existing instance-selection techniques.

In addition, SSISA produced better prediction accuracy compared to FPISA, while FPISA performed better than SSISA in training speed, storage reduction percentage and algorithm speed. In addition, the filter-based techniques outperform the wrapper-based techniques in training speed, storage reduction percentage and algorithm speed, while the wrapper-based techniques outperform their filter-based counterpart in prediction accuracy.

Overall, the results achieved by FPISA and SSISA established them to be global search algorithms with a promising approach for SVM speed optimization (and other ML algorithms). The results also show that they are very useful and efficient for both small and big data processing.

Future research could focus on designing improved and faster hybrid techniques, by trying other bio-inspired algorithms. In addition, the methods considered in this study are iterative in structure, so future research could therefore explore the possible implementation of non-iterative approaches. Future research could also consider exploring other ML algorithms.

ACKNOWLEDGMENT

The authors acknowledge the significant contributions of Professor Aderemi Adewumi to this study.

REFERENCES

[1] J. A. Olvera-López, J. A. Carrasco-Ochoa, J. F. Martínez-Trinidad, and J. Kittler, "A review of instance selection methods," *Artif. Intell. Rev.*, vol. 34, no. 2, pp. 133–143, 2010.
[2] H. Brighton and C. Mellish, "Advances in instance selection for instance-based learning algorithms," *Data Mining Knowl. Discovery*, vol. 6, no. 2, pp. 153–172, Apr. 2002.
[3] N. Panda, E. Y. Chang, and G. Wu, "Concept boundary detection for speeding up SVMs," in *Proc. 23rd Int. Conf. Mach. Learn.*, 2006, pp. 681–688.
[4] B. L. Narayan, C. A. Murthy, and S. K. Pal, "Maxdiff kd-trees for data condensation," *Pattern Recognit. Lett.*, vol. 27, no. 3, pp. 187–200, 2006.
[5] H. Liu and H. Motoda, "On issues of instance selection," *Data Mining Knowl. Discovery*, vol. 6, no. 2, pp. 115–130, 2002.
[6] J. C. Bezdek and L. I. Kuncheva, "Nearest prototype classifier designs: An experimental study," *Int. J. Intell. Syst.*, vol. 16, no. 12, pp. 1445–1473, Dec. 2001.
[7] V. Cerveron and F. J. Ferri, "Another move toward the minimum consistent subset: A tabu search approach to the condensed nearest neighbor rule," *IEEE Trans. Syst., Man, Cybern. B. Cybern.*, vol. 31, no. 3, pp. 408–413, Jun. 2001.
[8] J. A. Olvera-López, J. A. Carrasco-Ochoa, and J. F. Martínez-Trinidad, "Sequential search for decremental edition," in *Proc. Int. Conf. Intell. Data Eng. Automated Learn.*, 2005, pp. 280–285.
[9] L. I. Kuncheva, "Fitness functions in editing k-NN reference set by genetic algorithms," *Pattern Recognit.*, vol. 30, no. 6, pp. 1041–1049, 1997.
[10] J. R. Cano, F. Herrera, and M. Lozano, "Stratification for scaling up evolutionary prototype selection," *Pattern Recognit. Lett.*, vol. 26, no. 7, pp. 953–963, 2005.
[11] S. García, J. R. Cano, and F. Herrera, "A memetic algorithm for evolutionary prototype selection: A scaling up approach," *Pattern Recognit.*, vol. 41, no. 8, pp. 2693–2709, Aug. 2008.

[12] I. M. Anwar, K. M. Salama, and A. M. Abdelbar, "Instance selection with ant colony optimization," *Procedia Comput. Sci.*, vol. 53, pp. 248–256, Jan. 2015.
[13] U. Garain, "Prototype reduction using an artificial immune model," *Pattern Anal. Appl.*, vol. 11, nos. 3–4, pp. 353–363, Sep. 2008.
[14] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995.
[15] J. Fehr, K. Z. Arreola, and H. Burkhardt, "Fast support vector machine classification of very large datasets," in *Data Analysis, Machine Learning and Applications*. Berlin, Germany: Springer, 2008, pp. 11–18.
[16] K. Z. Arreola, J. Fehr, and H. Burkhardt, "Fast support vector machine classification using linear SVMs," in *Proc. 18th Int. Conf. Pattern Recognit. (ICPR)*, Aug. 2006, pp. 366–369.
[17] S. Albelwi and A. Mahmood, "Analysis of instance selection algorithms on large datasets with deep convolutional neural networks," in *Proc. IEEE Long Island Syst., Appl. Technol. Conf. (LISAT)*, Apr. 2016, pp. 1–5.
[18] M. Blachnik, "Ensembles of instance selection methods based on feature subset," *Procedia Comput. Sci.*, vol. 35, pp. 388–396, Jan. 2014.
[19] Y.-J. Lee and O. L. Mangasarian, "RSVM: Reduced support vector machines," in *Proc. SIAM Int. Conf. Data Mining*, Apr. 2001, pp. 1–17.
[20] H. Lei and V. Govindaraju, "Speeding up multi-class SVM evaluation by PCA and feature selection," in *Proc. Workshop Feature Selection Data Mining, Interfacing Mach. Learn. Statist.*, Newport Beach, CA, USA, Apr. 2005, pp. 72–79.
[21] S. García, J. Derrac, I. Triguero, C. J. Carmona, and F. Herrera, "Evolutionary-based selection of generalized instances for imbalanced classification," *Knowl-Based Syst.*, vol. 25, no. 1, pp. 3–12, Feb. 2012.
[22] F. Angiulli and A. Astorino, "Scaling up support vector machines using nearest neighbor condensation," *IEEE Trans. Neural Netw.*, vol. 21, no. 2, pp. 351–357, Feb. 2010.
[23] F. Angiulli, "Fast nearest neighbor condensation for large data sets classification," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 11, pp. 1450–1464, Nov. 2007.
[24] C.-F. Tsai and K.-C. Cheng, "Simple instance selection for bankruptcy prediction," *Knowl.-Based Syst.*, vol. 27, pp. 333–342, Mar. 2012.
[25] J. Chen, C. Zhang, X. Xue, and C.-L. Liu, "Fast instance selection for speeding up support vector machines," *Knowl.-Based Syst.*, vol. 45, pp. 1–7, Jun. 2013.
[26] D. Rodrigues, X.-S. Yang, A. N. de Souza, and J. P. Papa, "Binary flower pollination algorithm and its application to feature selection," in *Recent Advances in Swarm Intelligence and Evolutionary Computation*. Cham, Switzerland: Springer, 2015, pp. 85–100.
[27] H. M. Zawbaa and E. Emary, "Applications of flower pollination algorithm in feature selection and knapsack problems," in *Nature-Inspired Algorithms and Applied Optimization*. Cham, Switzerland: Springer, 2018, pp. 217–243.
[28] S. A.-F. Sayed, E. Nabil, and A. Badr, "A binary clonal flower pollination algorithm for feature selection," *Pattern Recognit. Lett.*, vol. 77, pp. 21–27, Jul. 2016.
[29] H. M. Zawbaa, A. E. Hassanien, E. Emary, W. Yamany, and B. Parv, "Hybrid flower pollination algorithm with rough sets for feature selection," in *Proc. 11th Int. Comput. Eng. Conf. (ICENCO)*, Dec. 2015, pp. 278–283.
[30] J. J. Q. Yu and V. O. K. Li, "A social spider algorithm for solving the non-convex economic load dispatch problem," *Neurocomputing*, vol. 171, pp. 955–965, Jan. 2016.
[31] W. T. Elsayed, Y. G. Hegazy, F. M. Bendary, and M. S. El-bages, "Modified social spider algorithm for solving the economic dispatch problem," *Eng. Sci. Technol., Int. J.*, vol. 19, no. 4, pp. 1672–1681, Dec. 2016.
[32] U. P. Shukla and S. J. Nanda, "Parallel social spider clustering algorithm for high dimensional datasets," *Eng. Appl. Artif. Intell.*, vol. 56, pp. 75–90, Nov. 2016.
[33] J. J. Q. Yu and V. O. K. Li, "Base station switching problem for green cellular networks with social spider algorithm," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2014, pp. 2338–2344.
[34] H. M. Zawbaa, E. Emary, A. E. Hassanien, and B. Parv, "A wrapper approach for feature selection based on swarm optimization algorithm inspired from the behavior of social-spiders," in *Proc. 7th Int. Conf. Soft Comput. Pattern Recognit. (SoCPaR)*, Nov. 2015, pp. 25–30.
[35] M. A. El Aziz and A. E. Hassanien, "An improved social spider optimization algorithm based on rough sets for solving minimum number attribute reduction problem," *Neural Comput. Appl.*, vol. 30, no. 8, pp. 2441–2452, 2018.
[36] A. M. Anter, A. E. Hassanien, M. A. ElSoud, and T.-H. Kim, "Feature selection approach based on social spider algorithm: Case study on abdominal CT liver tumor," in *Proc. 7th Int. Conf. Adv. Commun. Netw. (ACN)*, Jul. 2015, pp. 89–94.
[37] X.-S. Yang, "Flower pollination algorithm for global optimization," in *Proc. Int. Conf. Unconventional Comput. Natural Comput.*, 2012, pp. 240–249.
[38] J. J. Q. Yu and V. O. K. Li, "A social spider algorithm for global optimization," *Appl. Soft Comput.*, vol. 30, pp. 614–627, May 2015.
[39] F. F. Campón, "Group foraging in the colonial spider *Parawixia bistriata* (Araneidae): Effect of resource levels and prey size," *Animal Behav.*, vol. 74, no. 5, pp. 1551–1562, 2007.
[40] A. Asuncion and D. Newman. (2007). *UCI Machine Learning Repository*. Accessed: Aug. 15, 2016. [Online]. Available: http://archive.ics.uci.edu/ml/datasets.html
[41] LIBSVM. (2018). *LIBSVM Data Classification (Multi-Class)*. Accessed: Aug. 8, 2018. [Online]. Available: https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html#usps
[42] C. C. Chang and C. J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 1–27, 2011.
[43] J. Q. James. (2015). *Social Spider Algorithm*. Accessed: Sep. 20, 2016. [Online]. Available: https://github.com/James-Yu/SocialSpiderAlgorithm/blob/master/MATLAB/SSA.m
[44] X.-S. Yang. (2014). *Flower Pollination Algorithm*. Accessed: Sep. 20, 2016. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/45112-flower-pollination-algorithm/content/fpa_demo.m
[45] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, "A practical guide to support vector classification," Dept. Comput. Sci., Nat. Taiwan Univ., Taipei, Taiwan, Tech. Rep., 2003.
[46] S. Fine and K. Scheinberg, "Efficient SVM training using low-rank kernel representations," *J. Mach. Learn. Res.*, vol. 2, pp. 243–264, Mar. 2002.
[47] M. Behdad, L. Barone, M. Bennamoun, and T. French, "Nature-inspired techniques in the context of fraud detection," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 6, pp. 1273–1290, Nov. 2012.
[48] P. J. Fleming and R. C. Purshouse, "Evolutionary algorithms in control systems engineering: A survey," *Control Eng. Pract.*, vol. 10, no. 11, pp. 1223–1241, Nov. 2002.

ANDRONICUS A. AKINYELU received the B.Sc. degree (Hons.) in computer science from the Federal University of Technology, Akure, Nigeria, in 2011, and the M.Sc. and Ph.D. degrees in computer science from the University of Kwa-Zulu Natal, South Africa, in 2015 and 2017, respectively. He is currently a Postdoctoral Research Fellow with the University of the Free State, Bloemfontein, South Africa. His research interests include deep learning, machine learning, big data analytics, artificial intelligence, computer vision, and nature inspired optimization.

ABSALOM E. EZUGWU received the B.Sc. degree in mathematics and the M.Sc. and Ph.D. degrees in computer science from Ahmadu Bello University, Zaria, Nigeria. He is currently a Senior Lecturer with the School of Mathematics, Statistics, and Computer Science, University of KwaZulu-Natal, Durban, South Africa. He has published articles relevant to his research interest in internationally referred journals and edited books, conference proceedings, and local journals. His main research interests include parallel algorithms design in cloud and grid computing environments, and artificial intelligence with specific interest in computational intelligence and metaheuristic solutions to real-world global optimization problems. He is a member of IAENG and ORSSA.