
Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2020 SCPE. Volume 21, Issue 2, pp. 173–188, DOI 10.12694:/scpe.v21i2.1655

BIRD SWARM OPTIMIZATION-BASED STACKED AUTOENCODER DEEP LEARNING FOR UMPIRE DETECTION AND CLASSIFICATION

SUVARNA NANDYAL* AND SURVANA LAXMIKANT KATTIMANI†

Abstract. One of the most-watched and a played sport is cricket, especially in South Asian countries. In cricket, the umpire has the power to make significant decisions about events in the field. With the growing increase in the utilization of technology in sports, this paper presents the umpire detection and classification by proposing an optimization algorithm. The overall procedure of the proposed approach involves three steps, like segmentation, feature extraction, and classification. At first, the video frames are extracted from the input cricket video, and the segmentation is performed based on the Viola-Jones algorithm. Once the segmentation is done, the feature extraction is carried out using Histogram of Oriented Gradients (HOG), and Fuzzy Local Gradient Patterns (Fuzzy LGP). Finally, the extracted features are given to the classification step. Here, the classification is done using the proposed Bird Swarm Optimization-based stacked autoencoder deep learning classifier (BSO-Stacked Autoencoders), that categories into umpire or others. The performance of the umpire detection and classification based on BSO-Stacked Autoencoders is evaluated based on sensitivity, specificity, and accuracy. The proposed BSO-Stacked Autoencoder method achieves the maximal accuracy of 96.562%, the maximal sensitivity of 91.884%, and the maximal specificity of 99%, which indicates its superiority.

Key words: Umpire classification, Viola-Jones algorithm, Bird Swarm Optimization, Stacked autoencoders deep learning, Histogram of Oriented Gradients, Fuzzy Local Gradient Patterns

AMS subject classifications. 68T05

1. Introduction.

Cricket is one of the popular games after soccer in the worldwide. Nowadays, matches are viewed and shared internationally through live satellite broadcasting with the highest viewership rating [1, 9]. Some of the cricket playing nations are Australia, New Zealand, England, India, South Africa, Pakistan, Zimbabwe, Sri Lanka, and Bangladesh. Television broadcasters, such as star sports, and ESPN consists of huge repositories of cricket videos. The analysis of cricket video has been gained more attention in digital video processing. The cricket video analysis is a challenging task, due to its complexities [10, 11]. Existing approaches of cricket videos are classified into genre-independent or genre-specific [17]. Cricket video is chosen as the initial application for entertainment purposes. In the cricket videos, the video contents are edited or recorded using various style formats [12, 14]. Some works have been done in the field that specially targets on cricket videos. One of the flourishing attempts for extracting semantic events from cricket video was based on a multi-level hierarchical framework [10, 16]. This framework employed audio-visual features for categorizing video segments [7]. On the other hand, it has been revealed that camera motion parameters are utilized for classifying limited events in cricket video [13].

Several object categories, such as objects of uninterested, and interest regions present in the video are to be segmented. The video object that is separated from its context is incomplete. These systems must be enforced with the context information. In cricket, the umpire is one of the people with high power for making decisions about events in the field. The umpire used gestures, hand signals, and poses [1]. Object extraction is one of the most crucial components in the framework, since the objects are used as the input for the event extraction process. Object detection from a frame or a video sequence has attracted the attention of many engineers working in the field. In current time object detection technology is well recognized [6, 8]. The motion of the ball and the players is important for understanding any game. The player, as well as the object identification,

*Department of Computer Science and Engineering, Poojya Doddappa Appa College of Engineering (Affiliated to Visvesvaraya Technological University, Belagavi-590018), Kalaburgi, Karnataka 585102, India (suvarna_nandyal@yahoo.co.in).

†Department of Computer Science and Engineering, B.L.D.E.A's V.P.Dr.P.G.Halakatti College of Engineering and Technology (Affiliated to Visvesvaraya Technological University, Belagavi-590018), Vijayapur, Karnataka 586103, India (suvarnaky1977@gmail.com).

is performed based on contextual color-based segmentation. Then, the location or tracking information is possible by simply finding a similar object in explicit tracking or successive frames [19]. Various algorithms are implemented for tracking and segmenting the video objects, like Continuously Adaptive Mean Shift algorithm (CAMSHIFT), partial list square analysis, Threshold decision, and diffusion distance, etc. [20]. From that CAMSHIFT [2] is a popular method for visual tracking with a minimal computational cost.

There are many techniques utilized to the task for umpire detection, from the classic methods of Hidden Markov Models (HMMs), and Gaussian Mixture Models (GMMs) to machine learning methods [35] of multilayer perceptron (MLP) and static var compensator (SVC), then further moves to the deep neural learning methods [31] of Convolutional neural network (CNN) and LSTM [12]. The k-nearest neighbor algorithm (KNN) is also employed for classification. Motion is an important feature for representing video sequences. Motion texture is the feature, which is derived from the motion field between video frames, like motion vector field or optical flow field [7]. These features are used in conjunction for devising a set of multicategory classifiers with support vector machines (SVMs) [18]. The CNN [1] is also outperformed in object detection and image classification [34], which is the integration of classifier and feature extractor. The convolutional layers of CNN are the feature extractors. They learn the representations from the input data automatically. The previous layers in CNN learn more generic features, like colour blobs, edges, and shapes. Deep learning methods are used in multidimensional applications [28], and online applications [29, 30]. The final fully connected layers of CNN employed these features and divided the data into one of the classes [18].

In this paper, an umpire detection and classification is developed based on BSO-Stacked Autoencoders. The overall procedure of the proposed method involves segmentation, feature extraction, and umpire classification. At first, the video frames are extracted from the input cricket video, and then, the segmentation is done based on the Viola-Jones algorithm. After that, the feature extraction is performed using HOG, and Fuzzy LGP. Once the feature extraction is done, the classification is performed using the proposed BSO-Stacked Autoencoders, which categories into umpire or others.

The main contribution of this paper: Developing umpire detection and classification approach using the proposed BSO-Stacked Autoencoders, in which the Stacked Autoencoder is trained using BSO for effective classification.

The rest of the paper is organized as follows: Section 2 describes the literature survey of eight existing techniques. Section 3 discusses with umpire detection and classification using BSO-Stacked Autoencoders, and section 4 discusses the results of the proposed BSO-StackedAutoencoders. Finally, section 5 concludes the paper.

2. Literature Survey. This section reveals the literature review of several methods related to umpire pose detection, pitch frame classification, batting shots recognition, event, and activity detection are described, and analyzed as follows: Aravind Ravi et al.[1] developed an approach for umpire pose detection using transfer learning to generate cricket highlights. The features were extracted from pre-trained networks. Then, the linear SVM is employed to detect the pose of the umpire. This method has improved training overhead, but did not consider other classification methods for better performance. A. Sasithradevi et al.[2] presented an approach for content-based video retrieval. Histogram of Fourier Coefficients (HFC) was introduced for indicating the objects into video frames. Additionally, the performance of a video retrieval system is validated using Extreme Learning Machine, and Random Forest. The method did not include the temporal information and trajectory information for representing objects in videos. M. Ravinder and T. Venugopal [3] developed the Bag-of-visual-words method to detect, and classify pitch frames in the cricket video. Local Binary Patterns (LBP), Color+Texture+Edge (CTE), and Scale-Invariant Feature Transform (SIFT) are the three features was employed for classification. The main drawback of this method is required more iteration. Muhammad Zeeshan Khan et al.[4] employed a deep convolutional neural network (Deep CNN) for cricket batting shots recognition. For batting shot recognition, several actions, like bowling styles, number of scores, locations of players in the field, and batting shots are analyzed. The method does not consider video to automatic textual commentary generation, information retrieval, and decision making. Sujoy Paul et al. [5] developed weakly-supervised activity localization and classification (W-TALC). Initially, a weakly-supervised module, and Two Stream-based feature extractor network were employed for extracting features. After that, Multiple Instance Learning Loss (MILL) and Co-Activity Similarity Loss (CASL) were established to learn the weights of the

network only the video-level labels of training videos. The method failed to consider CASL to other issues in computer vision.

Mohammad Ashraf Russo et al.[6] employed the combination of CNN, and recurrent neural network for classifying sports videos. The input to the network is a sequence of RGB color frames. Then, the frames were subjected to separate convolutional layers for classification. The method failed to consider other deep learning-based techniques to adopt more weight in the specific dataset. Mahesh M. Goyani et al.[7] developed the keyframe detection-based method to minimize computation time. This framework performed top-down event classification and detection based on the hierarchical tree. At first, the keyframes were extracted using Hue Histogram difference for indexing, and then, the logo transitions were detected. After that, crowd frames were detected using edge detection. Finally, the frame categorized into the player of team A, and team B, and umpire based on skin colour. The smaller dataset is not considered in this method. Sandesh Bananki Jayanth and Gowri Srinivasa [8] developed visual content-based techniques for extracting cricket pitch frames automatically. At first, the pre-processing was performed for selecting the subset of frames. Then, the filtering was done to reduce the search space by removing the frames. Then, the subset of frames was given to the Statistical Modelling of the grayscale (SMoG). After that, the Component Quantization-based region of interest extraction (CQRE) was introduced for pitch frames extraction. Other classifiers were not considered to recognize players and detect events (such as a goal). Hari R and Wilscy M [32] developed a method for detect the presence of the Umpire by the thresholding based color segmentation algorithm. This method produced good output for stored cricket video and also be used in real time with a dedicated hardware support. Anyhow, complex object identification, and object tracking methods were the difficult tasks in this method. Mahesh M. Goyani et al. [33] developed a keyframe detection based approach, which was highly accurate with less computational time. Anyhow, it did not overcome the illumination related problems.

2.1. Challenges.

- The analysis of cricket video is a more challenging task due to the complexities of the game in itself, like various formats of matches, day and night matches (cause illumination related problem), dynamic playing conditions (pitches, field area and so on), and duration [15]. In the proposed system, Fuzzy LGP is used to enhance the distinguishing capability, and to reduce the complexities of the game.
- In [5], the W-TALC framework is developed for temporal activity localization and classification. Here, the accuracy was found better, but the W-TALC framework never been examined on smaller datasets for umpire classification. In this work, the classification is done by the proposed BSA-based stack autoencoder, which is applicable for any size of dataset.

3. Proposed Umpire Detection and Classification Using BSO-Based Stacked Auto Encoder.

This section presents the proposed umpire classification approach using a BSO-based stacked autoencoder. Figure 3.1 deliberates the schematic diagram of the proposed BSO-based stacked autoencoder for umpire detection and classification. At first, the input cricket video is converted into video frames. Subsequently, the segmentation is performed using the Viola Jones algorithm. Once the segmentation is done, the feature extraction is carried out using HOG [16], and Fuzzy LGP [24]. At last, the classification is performed based on extracted features using the proposed BSO [22]-based stacked autoencoder deep learning classifier [21] (BSO-based stacked autoencoder) that categories into umpire or others.

Let us consider G be the input video sequence and it comprises of M number of frames and is denoted as $G = \{L_a; 1 \leq a \leq M\}$. Thus, the frames extracted from is expressed as, $L_a = \{L_1, L_2.....L_M\}$.

3.1. Segmentation of humans in each frame using Viola Jones. This section presents the segmentation of face region from the extracted video frames. Different types of algorithms are used for face detection in previous works. The traditional approaches used for face detection are binary classification techniques, posing high computational complexity issues. The Viola-jones face detection algorithm is utilized for detecting face region from the extracted frames. Compared to other face detection algorithm, the computational complexity of the Viola-Jones [23] is limited that makes it suitable for various applications, like image databases, user interfaces, teleconferencing and so on. The steps involved in the Viola-Jones face detection algorithm are as follows: the creation of an integral image, feature selection by Haar-like a feature, cascading classifiers, detection of the face region, and Ada-boosting classifier based feature selection. After classification and Ada-boost

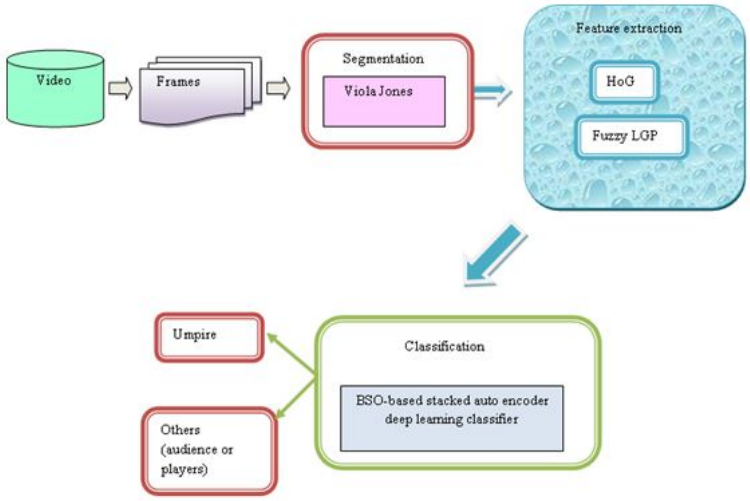

FIG. 3.1. *Block diagram of the proposed approach for umpire classification*

classifier based feature selection task is performed, the face regions are detected by the Viola-Jones algorithm and is expressed as,

$$J = VJ(L_a) \tag{3.1}$$

VJ indicates the function representing the Viola-Jones algorithm for face detection. The segmented region is expressed as J.

3.2. Feature Extraction. After segmentation, feature extraction is performed using fuzzy HoG, and Fuzzy-LGP. The feature extraction step carried out in this paper is explained as follows:

a) Fuzzy LGP: Fuzzy LGP [24] is the integration of LGP with fuzzy logic, which is used to enhance the distinguishing capability, and to reduce the noise effects. The crisp form of LGP utilizes the neighbourhood pixel properties to explain each pixel. It is more resistant, efficient, and simple to make changes in gray-level using lighting variations. The fine texture properties are effectively captured by the LBP patterns. However, LBP utilizes hard thresholding to compute the code, and has minimum discrimination power, and is sensitive against noise. The Fuzzy-LGP descriptor is an advanced form of LGP that carries highly discriminative features. In Fuzzy-LGP, every pixel is regarded as any number of LBP codes, which contributes to the Fuzzy-LGP bin histogram. The membership function is calculated as,

$$h_0(d) = \left\{ \begin{array}{ll} 0; & g_d \geq g_{center} + E \\ \frac{E - g_d + g_c enter}{2.E} & g_{center} - E < g_d < g_{center} + E \\ 1; & g_d \geq g_{center} - E \end{array} \right\} \tag{3.2}$$

$$h_1(d) = 1 - h_0(d) \tag{3.3}$$

where, $h_1()$ and $h_0()$ represents the membership functions, E denotes the threshold parameter, which is used to control the fuzziness degree, g_d denotes the neighboring pixel, and d is denoted as the total number of pixels. The membership function determines the contribution of LBP code into the FLBP histogram. The LGP code contribution is defined as

$$F(X) = \Pi_{d=0}^{8} h_{c_d}(d) \tag{3.4}$$

where, $c_d \in \{0, 1\}$. The sum of neighbourhood contribution is equal to unity, and is expressed as,

$$\sum_{X}^{255} F(X) = 1 \tag{3.5}$$

Crisp LGP histogram has zero value bins, whereas FLGP histogram has non-zero value bins. Hence FLGP is more informative than crisp LGP.

b) HOG: HOG [16] is a type of feature descriptor for extracting umpires. This technique utilizes gradients for localizing the image. The HOG features are extracted using the magnitude and orientation. Gradients (H_u, H_v) are calculated in both vertical and horizontal directions for all pixels in the frame. Gradient magnitudes and directions are estimated using the below equations,

$$h = \sqrt{H_u^2 + H_v^2} \tag{3.6}$$

$$\theta = \tan^{-} 1(\frac{H_v}{H_u}) \tag{3.7}$$

After the extraction of features from every face region, the concatenated feature is expressed as,

$$J^D = \{F||H\} \tag{3.8}$$

The extracted features are denoted as J^D with dimension $1X(M \times S)$. where, M signifies the total number of bins,S refers to the total number of extracted dimensions of the HOG features.

3.3. Umpire classification using BSO-based stack autoencoder. Once the features are extracted using fuzzy LGP, and HOG, the extracted features are given to the proposed BSA-based stack autoencoder for umpire classification. For the effective classification, stack autoencoder deep learning classifier is used, and the weights-biases are determined using the proposed algorithm optimally.

i) Architecture of stacked autoencoder deep learning. One of the building blocks of Deep Neural Network (DNN) is autoencoder. Stacked autoencoder [21] is the most commonly employed deep learning techniques. Figure 3.2 shows the architecture of the stacked autoencoder. Here, the autoencoders are stacked hierarchically. The architecture of autoencoder possesses three units, input visible units, output visible units, and hidden units, denoted as S, K and N. An autoencoder is utilized for encoding the input vector into an advanced stage hidden representation. In the encoder phase, the deterministic mapping R_ϕ , transforms the input vector into the hidden vector, and is expressed as,

$$B = fun(Q_1 P_+ A_1 P) \tag{3.9}$$

where, Q_1indicate the weight matrix, and A_1 refers to the bias vector. The term Prefers to reconstruction. Then the hidden representation is decoded and back to reconstruction factor is given as follows,

$$P = fun(Q_2 K_+ A_2 K) \tag{3.10}$$

where, Q_2 denote weight matrix, A_2 represent the bias vectors, and K hidden units.

ii) Training phase. a) Fitness Evaluation: The fitness function is computed for the individual solution for better results. The output has the limited value of the error, which considered as optimal output. The best solution is determined at the previous iteration as each solution craves to obtain a better position. To mitigate the cost function, and the squared reconstruction error, the autoencoder is trained using backpropagation algorithm expressed by

$$Jac(Q, A) = \frac{1}{2Y} \sum_{j=1}^{Y} ||P_j - P_{target}||^2 \tag{3.11}$$

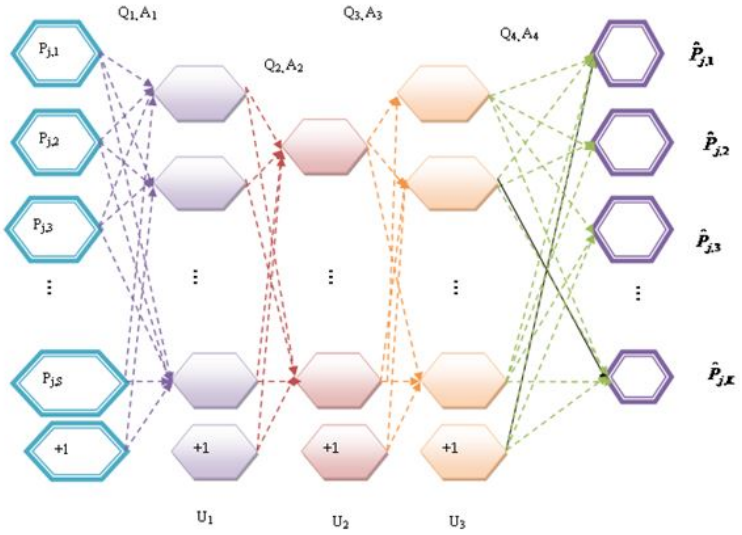

FIG. 3.2. *Architecture of stacked autoencoder classifier*

where P_j represents the estimated output, and the target output is denoted as P_{target} . The term Y denotes the total number of training samples.

b) Algorithmic description of the proposed BSO-Stacked autoencoder BSO [22] is duly based on the social behaviors of birds that follow some idealistic rules that follow: The individual bird switches between the vigilance behavior and foraging behaviour of birds. When foraging is in progress, individual bird records and updates the previous experience and their position, and also, the best experience of the swarms is updated, which is regarding the location of food. In case of vigilance behaviour, the individual birds move towards the center of swarms. The vigilance behaviour gets affected when there is any possibility of interference. At the same time, the bird switches between scrounging and producing when birds are trying to fly from one site to another. On the other hand, producers engage in active search for food. In addition, there is a perfect balance between exploration and exploitation in BSA. The Algorithmic steps of the proposed BSO-Stacked Autoencoders are illustrated below:

***a) Initialization*:** In the first step, the parameters of optimization including the population are initialized, which includes: $\{X_{i,j}; 1 \leq 1 \leq y; 1 \leq j \leq z$ where, the population size is denoted as y , τ_{max} denotes the maximal iteration, Pro represents the probability of foraging food, and the frequency of flight behaviour of birds is denoted as fr . Here,$X \in Q_1, Q_2, A_1, A_2$.

b) Evaluation of objective function: The selection of the optimal location of the bird is performed based on the minimization problem. The minimal value of the objective function describes the better solution, and therefore, the solution with the limited value of error is selected as the optimal solution. The error is estimated using equation (11).

c) Location update of the birds: For updating their positions, the birds have three stages, which are decided based on the probability. Whenever the random number , then the update is based on the foraging behaviour or else, the vigilance behaviour commences. On the other hand, swarm splits as scroungers and producers, which is modelled as flight behaviors. Finally, the feasibility of the solutions is verified and the best solution is retrieved.

d) Foraging behavior of birds: The individual bird searches the food based on their own experience, and the behaviour of the swarm, which is given below. The standard equation of the foraging behavior of birds is given as follows,

$$X_{i,j}^{\tau+1} = X_{i,j}^{\tau} + (P_{i,j} - X_{i,j}^{\tau}) \times G \times random(0,1) + [C_j - X_{i,j}^{\tau}] \times random(0,1) \tag{3.12}$$

where $X_{i,j}^{\tau+1}$ and $X_{i,j}^{\tau}$ denotes the location of i^{th} bird in j^{th} dimension at $\tau+1$ and τ .$P_{i,j}$ refers to the previous best position of the i^{th} bird, $random(0,1)$ is the independent uniformly distributed numbers, and .C_j indicates the best previous location shared by the bird swarm. The positive numbers are denoted as G and V , and $P_{i,j}$ denotes the personal best solution. C_j represents the global best solution.

e) Vigilance Behaviour of Birds: The birds move towards the center during which the birds compete with each other and the vigilance behaviour of birds is modelled as,

$$X_{i,j}^{\tau+1} = X_{i,j}^{\tau} + B_1(\mu_j - X_{i,j}^{\tau}) \times random(0,1) + B_2[M_{kj} - X_{i,j}^{\tau}] \times random(-1,1) \tag{3.13}$$

$$B_1 = w_1 \times exp\left(\frac{-Ff(M)_i}{\sum Ff + \delta} \times y\right) \tag{3.14}$$

$$B_1 = w_1 \times exp\left[\left(\frac{Ff(M)_i - Ff(M)_\tau}{|Ff(M)_\tau - Ff(M)_i - | + \delta}\right)\frac{y \times Ff(m)_\tau}{\sum Ff + \delta} \times y\right] \tag{3.15}$$

where y represents the number of birds, w_1 and w_2 are the positive constants lying in the range of $[0,2]$, $Ff(M)_i$ represents the optimal fitness value of i^{th} bird, and $\sum Ff$ corresponds to the addition of the best fitness values of the swarm. δ be the constant that keeps optimization away from zero-division error. T refers to the positive integer $(j \neq i)$. Whenever the bird approaches the center of the swarm, there is a tendency to compete with each other. The average fitness $\sum Ff$ value of the swarm corresponds to the indirect effect caused by surroundings when the swarm approaches the surroundings. The mean μ_h refers to the h^{th} element of the average position of the swarm.

f) Flight behaviour: This behaviour of the bird's progress when the birds fly to another site in case of any threatening events and foraging mechanisms. When the birds reach a new site, they search for food. Few birds in the group try acting as producers and few as scroungers. The behaviour is modelled as

$$X_{i,j}^{\tau+1} = X_{i,j}^{\tau} + \text{random } r(0.1) \times X_{i,j}^{\tau} \tag{3.16}$$

$$X_{i,j}^{\tau+1} = X_{i,j}^{\tau} + (X_{\gamma,j}^{\tau} - X_{i,j}^{\tau}) \times fl \times Rr(0,1) \tag{3.17}$$

where $Rr(0,1)$ refer to the Gaussian distributed random number with zero mean and standard deviation, $X_{i,j}^{\tau+1}$ and $X_{i,j}^{\tau}$ denotes the location of i^{th} bird in j^{th} dimension at $(\tau+1)$ and τ.

g) Checking the feasibility of solution: The feasibility of the solution is computed based on the objective function. If the newly generated solution is best than the previous one, then it is changed by the new solution.

h) Termination: Repeat the steps for the maximal iterations until the global optimal best solutions are determined. Thus, the optimization algorithm discussed in this section aims at determining the optimal multipath for enabling secure communication in the network. Algorithm 1 depicts the pseudo-code of the proposed BSO-Stacked Autoencoders, which demonstrates the step-wise description of the algorithm.

The flowchart for the proposed BSO-Stacked Autoencoders is given in Figure 3.3.

4. Discussion of Results. The results obtained by the proposed BSO-Stacked Autoencoders are described in this section. The proposed BSO-Stacked Autoencoders is analyzed based on the performance using three measures, which include accuracy, sensitivity, and specificity.

4.1. Experimental setup. The experimentation of the developed approach is performed using manually collected database, and the implementation is done in the MATLAB tool. Here, the 20 videos are collected from YouTube.

Algorithm 1: Pseudocode for the proposed BSO-Stacked Autoencoders

```
Data: Bird swarm population {X_i, j; 1 ≤ i ≤ y; 1 ≤ j ≤ z;}
Result: Best solution
Procedure:
Begin
Population initiation: {X_i, j; 1 ≤ i ≤ y; 1 ≤ j ≤ z;}
Read the parameters y -population size; τ_max maximal iteration, Pro -probability of foraging food, fr -frequency of
  flight behaviour of birds
Determine the fitness of the solutions
while τ < τ_max do
    for g = 1 : y do
        if R(0, 1) < Pro then
            Foraging behaviour using equation (3.12)
        else
            Vigilance behaviour using equation (3.15)
    Split the swarm as scroungers and producers
    for g = 1 : y do
        if g is a producer then
            Update using equation (3.16)
        else
            Update using equation (3.17)
    Check the feasibility of the solutions
    Return the best solution
    τ = τ + 1
Optimal solution is obtained
End
```

4.2. Performance metrics. The performance of the developed BSO-Stacked Autoencoders is evaluated using the metrics, namely accuracy, sensitivity, and specificity. a) Accuracy: Accuracy is defined as the measure of accurateness based on the BSO-Stacked Autoencoders, and is expressed as,

$$Accuracy = \frac{X + Y}{X + P + Y + Q} \tag{4.1}$$

where, denotes the true positives, and is the true negatives. denotes the false positives and is the false negatives. b) Sensitivity: Sensitivity is also called as a True positive Rate (TPR), which defines the measure of positiveness and is computed using the below expression.

$$Sensitivity = \frac{Y}{P + Y} \tag{4.2}$$

c) Specificity: Specificity is otherwise called as a True Negative Rate (TNR), which is the measure of false negatives. Specificity is represented as,

$$Specificity = \frac{X}{X + Q} \tag{4.3}$$

4.3. Experimental results. The experimental results obtained by the proposed BSO-Stacked Autoencoders are given in this section. Figure 4.1 depicts the sample results of the umpire. Figure 4.1.a) shows the input image 23, figure 4.1.b) depicts the input image 42, figure 4.1.c) shows the input image 69, and figure 4.1.d) depicts the input image 60. Figure 4.1.e) shows the predicted output of input image 23, figure 4.1.f) depicts the predicted output of input image 42, figure 4.1.g), and figure 4.1.h) demonstrate the predicted output of input image 69, and 60.

The sample results of non-umpire are depicted in figure 4.2 depicts the sample results. Figure 4.2.a) shows the input image 1, figure 4.2.b) depicts the input image 13, figure 4.2.c) shows the input image 15, and figure 4.2.d) depicts the input image 61. Figure 4.2.e) shows the predicted output of input image 1, figure 4.2.f) depicts

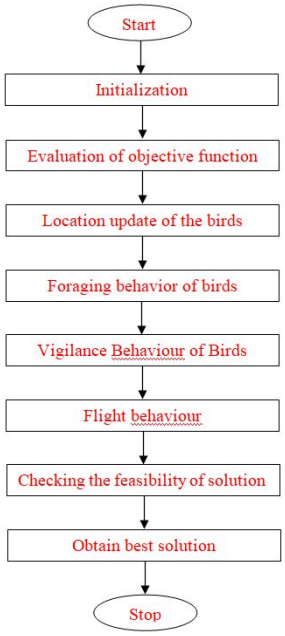

FIG. 3.3. *Flowchart of the proposed BSO-Stacked Autoencoders*

FIG. 4.1. *Sample results of umpire a) Input image 23, b) Input image 42, c) Input image 69, d) Input image 60, e) predicted output of input image 23, f) predicted output of input image 42, g) predicted output of input image 69, h) predicted output of input image 60*

the predicted output of input image 13, figure 4.2.g), and figure 4.2.h) demonstrate the predicted output of input image 15, and 61.

4.4. Performance analysis. The performance analysis of the developed BSO-Stacked Autoencoder method is carried out with respect to the sensitivity, specificity, and accuracy. These parameters are necessary to be improved to enhance the performance of the technique.

a) Analysis based on training percentage Figure 4.3 shows the performance analysis of the proposed BSO-Stacked Autoencoder method. Figure 4.3.a) shows the performance analysis in terms of accuracy by varying the number of iteration. In this figure, the accuracy value has its limit ranging from 0 to 90 and the training percentage value varies between 40 and 90. For 40% of training data, the accuracy values measured by the BSO-Stacked Autoencoders for the number of iterations 50, 100, 150, 200, and 250 are 61.182%, 62.756%, 63.014%, 68.125%, and 78.191%, respectively. Similarly, when the training data percentage is 90, the accuracy value measured by BSO-Stacked Autoencoders with iteration 50 is 71.074%, BSO-Stacked Autoencoders with

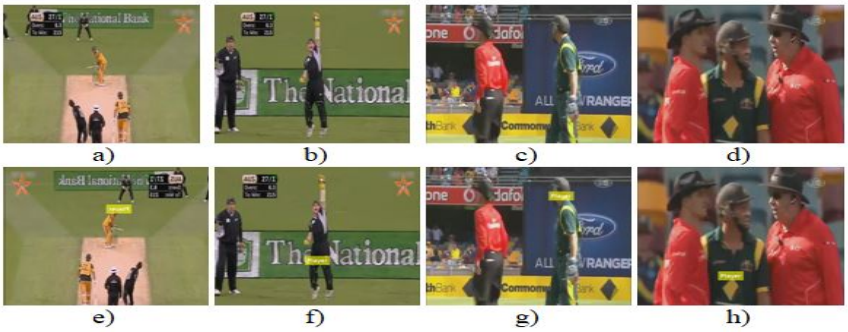

FIG. 4.2. *Sample results of non-umpire a) Input image 1, b) Input image 13, c) Input image 15, d) Input image 61, e) predicted output of input image 1, f) predicted output of input image 13, g) predicted output of input image 15, h) predicted output of input image 61*

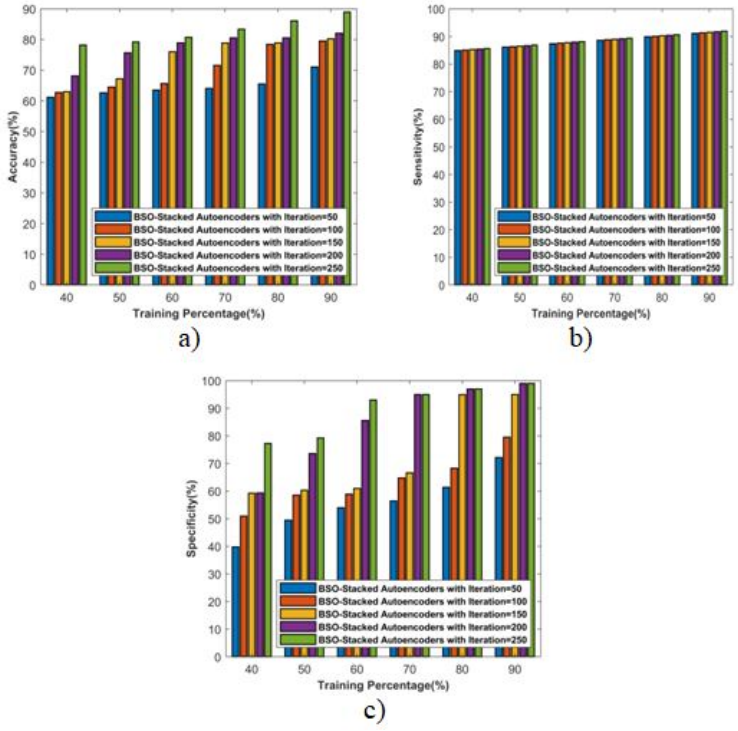

FIG. 4.3. *Performance analysis of the proposed BSO-Stacked Autoencoders (a) Accuracy, (b) Sensitivity, and (c) Specificity*

iteration 100 is 79.502%, BSO-Stacked Autoencoders with iteration 150 is 80.246%, BSO-Stacked Autoencoders with iteration 200 is 82.055%, and BSO-Stacked Autoencoders with iteration 250 is 88.941%. From the above figure, it is clearly shown the maximum accuracy measure of 88.941% is attained with the iteration 250 for the training data percentage 90.

Figure 4.3.b) depicts the performance analysis in terms of sensitivity. When the training data percentage is 50, the sensitivity values measured by the proposed BSO-Stacked Autoencoders with iteration 50, 100, 150, 200, and 250 are 86.152%, 86.33%, 86.508%, 86.686%, and 86.864%, respectively. Likewise, when the training data percentage is 80, the sensitivity value measured by BSO-Stacked Autoencoders with iteration 50 is 89.884%, BSO-Stacked Autoencoders with iteration 100 is 90.068%, BSO-Stacked Autoencoders with iteration

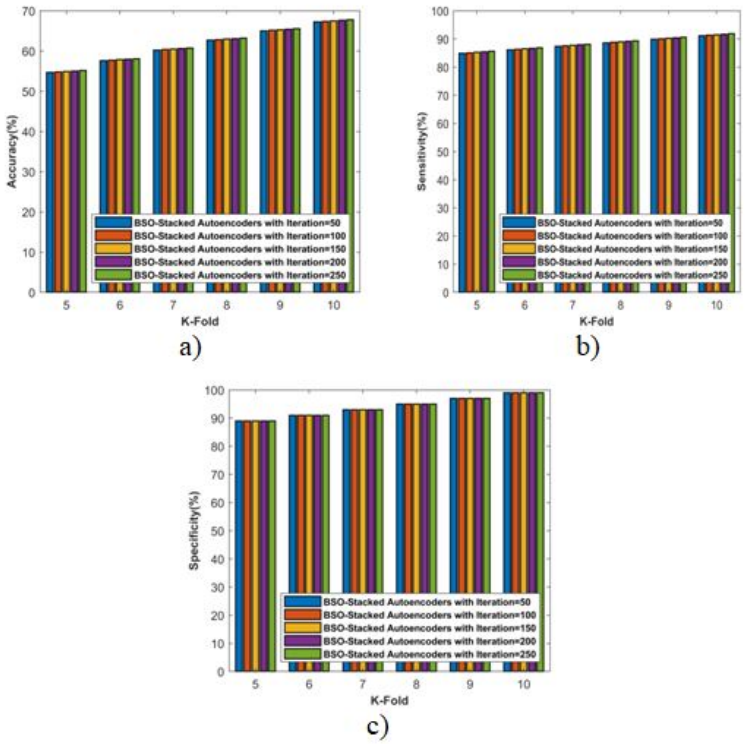

FIG. 4.4. *Performance analysis of the proposed BSO-Stacked Autoencoders (a) Accuracy, (b) Sensitivity, and (c) Specificity*

150 is 90.252%, BSO-Stacked Autoencoders with iteration 200 is 90.436%, and BSO-Stacked Autoencoders with iteration 250 is 90.62%. From the figure, it is concluded that when the training data percentage is 90, with the iteration 250, the proposed method acquired better performance.

Figure 4.3.c) depicts the performance analysis in terms of specificity. When the training data percentage is 60, the specificity values measured by the proposed BSO-Stacked Autoencoders with iteration 50, 100, 150, 200, and 250 are 53.975%, 58.910%, 60.983%, 85.592%, and 93%, respectively. Likewise, for 90% training data, the specificity value measured by BSO-Stacked Autoencoders with iteration 50 is 72.174%, BSO-Stacked Autoencoders with iteration 60 is 79.529%, BSO-Stacked Autoencoders with iteration 70 is 95%, BSO-Stacked Autoencoders with iteration 80 is 99%, and BSO-Stacked Autoencoders with iteration 90 is 99%. From the figure, it is clearly shown that when the training data percentage is 90 with iteration 200, and 250, the proposed method acquired better performance.

b) Analysis based on K-Fold Figure 4.4 depicts the performance analysis of the proposed BSO-Stacked Autoencoder method. Figure 4.4.a) shows the analysis based on accuracy by varying the K-Fold. For K-Fold=7, the accuracy values measured by the BSO-Stacked Autoencoder method for the number of iterations 50, 100, 150, 200, and 250 are 60.250%, 60.374%, 60.499%, 60.623%, and 60.748%. Similarly, when K-Fold is 10, the accuracy value measured by BSO-Stacked Autoencoders with iteration 50 is 67.235%, BSO-Stacked Autoencoders with iteration 100 is 63.374%, BSO-Stacked Autoencoders with iteration 150 is 67.512%, BSO-Stacked Autoencoders with iteration 200 is 67.650%, and BSO-Stacked Autoencoders with iteration 250 is 67.788%.

Figure 4.4.b) shows the performance analysis based on sensitivity by varying the K-Fold. In this figure, the sensitivity value is plotted against the K-Fold varies between 5 and 10. For K-Fold=8, the sensitivity values measured by the BSO-Stacked Autoencoder method for the number of iterations 50, 100, 150, 200, and 250 are 88.634%, 88.816%, 88.998%, 89.18%, and 89.362%. Similarly, when the K-Fold value is 10, the sensitivity value measured by the BSO-Stacked Autoencoders with iteration 50 is 91.14%, BSO-Stacked Autoencoders with

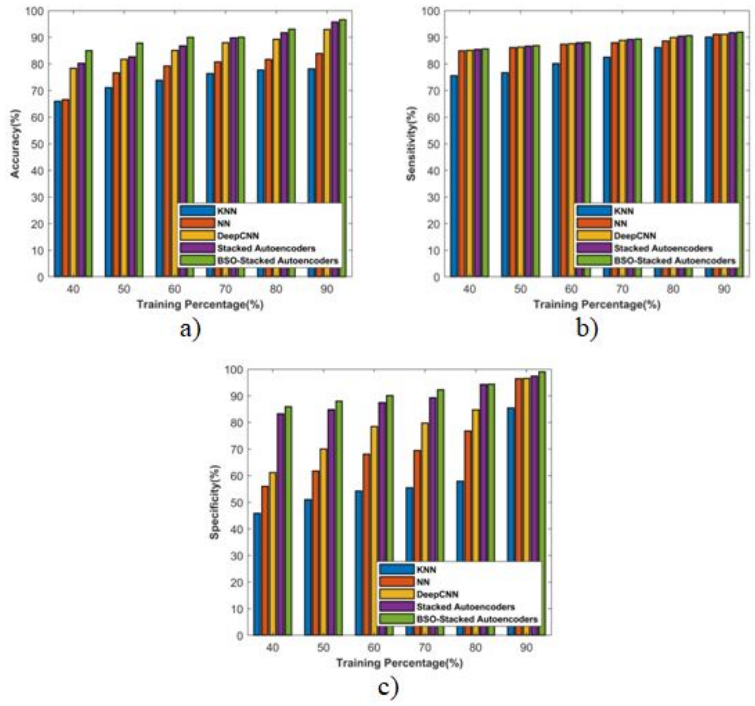

FIG. 4.5. *Comparative analysis by varying the training percentage a) Accuracy, b) Sensitivity, and c) Specificity*

iteration is 91.326%, BSO-Stacked Autoencoders with iteration 150 is 91.512%, BSO-Stacked Autoencoders with iteration 200 is 91.698%, and BSO-Stacked Autoencoders with iteration 250 is 91.884%. The maximum sensitivity measure of 91.884% is attained when the iteration is 250, with 10% of K-Fold.

Figure 4.4.c) depicts the performance analysis in terms of specificity. When the K-Fold is 5, the specificity values measured by BSO-Stacked Autoencoders with iterations 50, 100, 150, 200, and 250 are found to be 89%, respectively. Likewise, when the K-Fold is 9, the specificity value measured by BSO-Stacked Autoencoders with iterations 50, 100, 150, 200, and 250 are found to be 89%.

4.5. Comparative analysis. The comparative analysis of the proposed BSO-Stacked Autoencoder method by evaluating the performance of other comparative techniques is elaborated in this section. The comparative analysis is performed by varying the training data percentage, and K-fold, and the results are evaluated based on the metrics, like accuracy, specificity, and sensitivity.

4.6. Competing methods. The methods, such as KNN [26], Neural Network (NN) [27], DCNN [25], and Stacked Autoencoders, are used for the comparison with the proposed BSO-Stacked Autoencoder method for the analysis.

***a) Comparative analysis based on training data percentage*:** The analysis of the comparative methods based on accuracy, sensitivity, and specificity is depicted in figure 4.5. Figure 4.5.a) shows the analysis based on accuracy by varying the percentage of training data. For the training data=40%, the existing techniques, such as KNN, NN, DCNN, and Stacked Autoencoders, possesses the accuracy of 65.937%, 66.593%, 78.345%, and 80.262%, respectively, which is comparatively lower than the proposed BSO-Stacked Autoencoders. For the same training data, the proposed BSO-Stacked Autoencoders acquired the accuracy of 84.99%. Similarly, when the training percentage increased to 90%, the methods, KNN, NN, DCNN, and Stacked Autoencoders, attained the accuracy of 78.114%, 83.903%, 92.936%, and 95.717%, respectively, whereas the accuracy of the proposed method is 96.56%. From the above interpretation, it is seen that the proposed BSO-Stacked Autoencoders achieved improved accuracy of 96.56% at 90% training data.

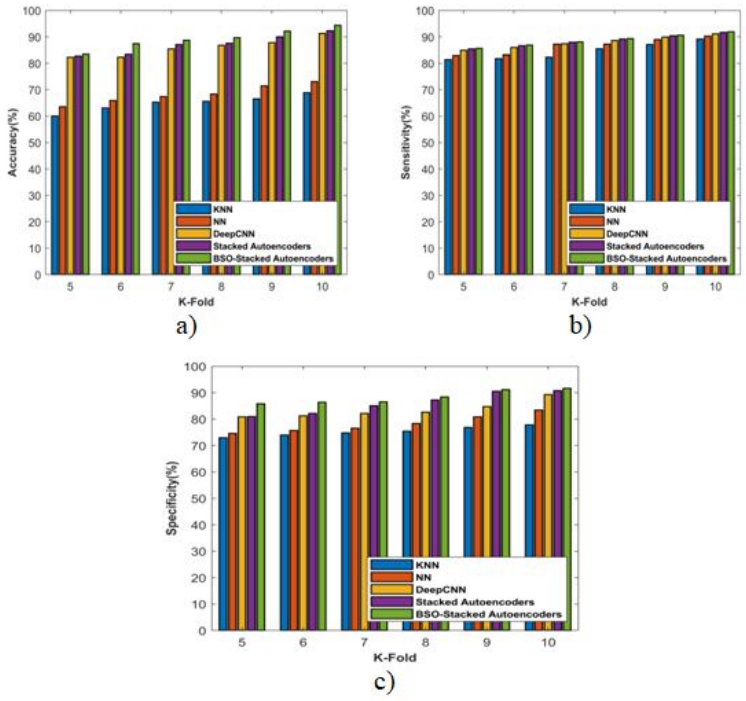

FIG. 4.6. *Comparative analysis by varying the K-Fold a) accuracy b) sensitivity and c) specificity*

The comparative analysis based on the sensitivity metric is depicted in figure 4.5.b). When 60% of training data is considered, the existing methods, like KNN, NN, DCNN, and Stacked Autoencoders, acquire the sensitivity value of 80.134%, 87.39%, 87.57%, and 87.93%, respectively. Meanwhile, the proposed BSO-Stacked Autoencoders obtained a sensitivity value of 88.11%. When 80% of training data is considered, the sensitivity of the existing methods, like KNN, NN, DCNN, and Stacked Autoencoders, is 86.159%, 88.634%, 89.884%, and 90.436%, respectively, whereas the proposed BSO-Stacked Autoencoders attained the sensitivity of 90.62%.

The analysis based on specificity by varying the training data percentage is depicted in figure 4.5.c). Here, for the 70% training data, the existing techniques, like KNN, NN, DCNN, and Stacked Autoencoders achieved the specificity of 55.454%, 69.450%, 79.771%, and 89.270%, but the proposed BSO-Stacked Autoencoders acquired the specificity of 92.245%. While considering 80% of training data, the existing techniques, like KNN, NN, DCNN, and Stacked Autoencoders, achieved the specificity of 57.953%, 76.818%, 84.736%, and 94.284%, respectively. Meanwhile, the proposed BSO-Stacked Autoencoders attained the specificity of 94.381%. From figure 8 c), the proposed BSO-Stacked Autoencoders is found to possess the maximum specificity of 99% at 90% training data.

b) Comparative analysis based on K-Fold: The analysis of the comparative methods based on accuracy, sensitivity, and specificity is depicted in figure 4.6. Figure 4.6.a) shows the analysis based on accuracy by varying the percentage of training data. For the K-Fold =5, the existing techniques, such as KNN, NN, DCNN, and Stacked Autoencoders, possesses the accuracy of 60.063%, 63.530%, 82.231%, and 82.717%, respectively, which is comparatively lower than the proposed BSO-Stacked Autoencoders. For the same training data, the proposed BSO-Stacked Autoencoders achieved an accuracy value of 84.49%. Similarly, when the K-Fold is increased to 9, the methods, KNN, NN, DCNN, and Stacked Autoencoders, attained the accuracy of 66.500%, 71.447%, 87.798%, and 90.001%, respectively, whereas the accuracy of the proposed BSO-Stacked Autoencoders is 92.135%. From the above interpretation, it is seen that the proposed method achieved an improved accuracy value of 94.356% at 10% of K-Fold.

The comparative analysis in terms of the sensitivity metric is depicted in figure 4.6.b). When K-Fold=6 is

TABLE 4.1
Comparative analysis based on training data percentage

Methods	Accuracy (%)	Sensitivity (%)	Specificity (%)
KNN	78.114	90.065	85.444
NN	83.903	91.14	96.426
DCNN	92.936	91.151	96.525
Stacked Autoencoders	95.717	91.698	97.381
Proposed BSO-Stacked Autoencoders	96.562	91.884	99

TABLE 4.2
Comparative analysis based on K-Fold

Methods	Accuracy (%)	Sensitivity (%)	Specificity (%)
KNN	68.811	89.177	77.810
NN	73.068	90.255	83.448
DCNN	91.305	91.14	89.287
Stacked Autoencoders	92.294	91.698	90.792
Proposed BSO-Stacked Autoencoders	94.356	91.884	91.663

considered, the existing methods, like KNN, NN, DCNN, and Stacked Autoencoders acquire the sensitivity of 81.707%, 83.246%, 85.993%, and 86.686%, respectively. Meanwhile, the proposed BSO-Stacked Autoencoders acquired the sensitivity value of 86.864%. For K-Fold=10, the sensitivity of the comparative methods, like KNN, NN, DCNN, and Stacked Autoencoders is 89.177%, 90.255%, 91.14%, and 91.698%, respectively, whereas the proposed BSO-Stacked Autoencoders attained the sensitivity of 91.884%.

The analysis in terms of specificity metric is depicted in figure 4.6.c). Here, for K-Fold=7, the existing techniques, like KNN, NN, DCNN, and Stacked Autoencoders, achieved the specificity of 74.806%, 76.551%, 82.167%, and 85.055%, respectively, but the BSO-Stacked Autoencoders acquired the specificity of 86.530%. While considering K-Fold=10, the existing techniques, such as KNN, NN, DCNN, and Stacked Autoencoders, achieved the specificity of 77.810%, 83.448%, 89.287%, and 90.792%, respectively. Meanwhile, the proposed BSO-Stacked Autoencoders attained the specificity value of 91.663%. From figure 9 c), the BSO-Stacked Autoencoders is found to possess the maximum specificity value of 91.663% at 10% K-Fold.

4.7. Comparative discussion. Table 4.1 depicts the comparative discussion of the existing KNN, NN, DCNN, and Stacked Autoencoders and the proposed BSO-Stacked Autoencoders in terms of sensitivity, specificity, and accuracy parameters by varying the training data percentage. The maximum performance measured by proposed BSO-Stacked Autoencoders in terms of accuracy parameter is 96.562%, whereas the accuracy values of existing KNN, NN, DCNN, and Stacked Autoencoders are 78.114%, 83.903%, 92.936%, and 95.717%, respectively. The maximal sensitivity is computed by proposed BSO-Stacked Autoencoders with a value of 91.884%, whereas the existing KNN, NN, DCNN, and Stacked Autoencoders acquired the sensitivity of 90.065%, 91.14%, 91.151%, and 91.698%, respectively. The specificity value computed by proposed BSO-Stacked Autoencoders is 99%, whereas the existing KNN, NN, DCNN, and Stacked Autoencoders methods acquired the specificity of 85.444%, 96.426%, 96.525%, and 97.381%, respectively.

Table 4.2 depicts the comparative discussion of the existing KNN, NN, DCNN, and Stacked Autoencoders and proposed BSO-Stacked Autoencoders in terms of accuracy, sensitivity, and specificity parameters based on K-Fold. The maximum performance measured by proposed BSO-Stacked Autoencoders in terms of accuracy parameter is 94.356%, whereas the accuracy values of existing KNN, NN, DCNN, and Stacked Autoencoders are 68.811%, 73.068%, 91.305%, and 92.294%, respectively. The maximal sensitivity is computed by proposed BSO-Stacked Autoencoders with a value of 91.884%, whereas the existing KNN, NN, DCNN, and Stacked Autoencoders acquired the sensitivity of 89.177%, 90.255%, 91.14%, and 91.698%, respectively. The specificity value computed by proposed BSO-Stacked Autoencoders is 91.663%, whereas the existing KNN, NN, DCNN, and Stacked Autoencoders methods acquired the specificity of 77.810%, 83.448%, 89.287%, and 90.792%, respectively.

5. Conclusion. This paper presents an approach for umpire detection and classification by proposing an optimization algorithm. The proposed model undergoes three steps for the umpire classification and detection, namely segmentation, feature extraction, and classification. At first, the cricket video is converted into frames, and the segmentation is done using the Viola-Jones algorithm. After the segmentation, the feature extraction is performed by extracting features, like HOG, and Fuzzy LGP. Finally, the umpire classification is carried out using the proposed BSO-Stacked Autoencoders deep learning classifier. Experimentation is carried out using a manually collected dataset. The performance of the BSO-Stacked Autoencoders is evaluated using accuracy, sensitivity, and specificity by varying the training percentage and K-Fold. The proposed method produces the maximal accuracy of 96.562%, maximal sensitivity of 91.884%, and the maximal specificity of 99%, which indicates the superiority of the proposed method. In future, the proposed BSO-Stacked Autoencoders will be further improved with a hybrid optimization approach for better results.

REFERENCES

[1] A. Ravi, H. Venugopal, S. Paul, H. R. Tizhoosh, *A Dataset and Preliminary Results for Umpire Pose Detection Using SVM Classification of Deep Features*, In proceedings of IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1396-1402, 2018.

[2] A. Sasithradevi, S. M. M. Roomi, and G. Maragatham, *Content Based Video Retrieval via Object Based Approach*, In proceedings of 10th IEEE conference, pp.5-8, 2017.

[3] M. Ravinder and T. Venugopal, *Pitch Frames Classification in a Cricket Video Using Bag-of-Visual-Words*, Artificial Intelligence and Evolutionary Computations in Engineering Systems, pp. 793-801, 2016.

[4] M. Z. Khan, M. A. Hassan, and A. Farooq, *Deep CNN based Data-driven Recognition of Cricket Batting Shots*, In proceedings of international conference on Applied and engineering mathematics, 2018.

[5] S. Paul, S. Roy and A.K. Roy-Chowdhury, *W-TALC: Weakly-supervised Temporal Activity Localization and Classification*, In Proceedings of the European Conference on Computer Vision (ECCV), pp. 563-579, 2018.

[6] M. A. Russo, L. Kurnianggoro, and K.-H. Jo, *Classification of sports videos with combination of deep learning models and transfer learning*, In proceedings of International Conference on Electrical, Computer and Communication Engineering, pp.7-9, February 2019.

[7] M.M. Goyani, S. K. Dutta, and P. Raj, *Key Frame Detection Based Semantic Event Detection and Classification Using Heirarchical Approach for Cricket Sport Video Indexing*, In proceedings of International Conference on Computer Science and Information Technology, pp. 388-397, 2011.

[8] S. B. Jayanth and G. Srinivasa, *Automated Classification of Cricket Pitch Frames in Cricket Video*, Electronic Letters on Computer Vision and Image Analysis, vol.13, no.1, pp.33-49, 2014.

[9] M. H. Kolekar, S. Sengupta, *Semantic concept mining in cricket videos for automated highlight generation*, vol.47, pp.545-579, 2010.

[10] M. Goyani, S. Dutta, G. Gohil, and S. Naik, *wicket fall concept mining from cricket video using a-priori algorithm*, The International Journal of Multimedia & Its Applications (IJMA), vol.3, no.1, February 2011.

[11] M. H. Kolekar and S. Sengupta, *Event-importance based customized and automatic cricket highlight generation*, In proceedings of International Conference on Multimedia and expo, 2006.

[12] P. Xu, L. Xie, S. Chang, A. Divakaran, A. Vetro, and H. Sun, *Algorithms and system for segmentation and structure analysis in soccer video*, In IEEE ICME, 2001.

[13] N. Harikrishna, S. Satheesh, S. D. Sriram, and K.S. Easwarakumar, *Temporal Classification of Events in Cricket Videos*, In proceedings of National Conference on Communications (NCC), pp. 1-5, 2011.

[14] Vijayakumar.V, and Nedunchezhian.R, *Event detection in cricket video based on visual and acoustic features*, vol.3, no.8, August 2012.

[15] Y S. Kumar, S. K. Gupta, B R. Kiran, K R Ramakrishnan, and C. Bhattacharyya, *Automatic summarization of broadcast cricket videos*, In proceedings of 15 international symposium on consumer electronics, 2011.

[16] N. D. Lakshmi, Y. M. Latha, and A. Damodaram, *SILHOUETTE Extraction of a human body based on fusion of hog and graph-cut segmentation in dynamic backgrounds*, 2013.

[17] S. Abburu, *Context Ontology Construction For Cricket Video*, International Journal on Computer Science and Engineering, vol.2, no. 8, pp.2593-2597, 2010.

[18] C. K.Mohan, B. Yegnanarayana, *Classification of sport videos using edge-based features and autoassociative neural network models*, Signal, Image and Video Processing, vol.4, no.1, pp.61-73, 2010.

[19] A. Kokaram, N. Rea, R. Dahyot, A. M. Tekalp, P. Bouthemy, P. Gros, and I. Sezan, *Browsing Sports Video*, IEEE signal processing magazine, March 2006.

[20] P. Chaudhari, and K. S. Bhagat, *Video Frame Segmentation and Object Tracking using ANN for Surveillance System*, International Journal of Research in Advent Technology (IJRAT), April 2017.

[21] K. Jayapriya, N. A. B. Mary, *Employing a novel 2-gram subgroup intra pattern (2GSIP) with stacked auto encoder for membrane protein classification*, Molecular Biology Reports, February 2019.

[22] X.-B. Mengab, X.Z. Gaoc, L. Lude, Y. Liub, and H. Zhanga, *A new bio-inspired optimisation algorithm: Bird Swarm Algorithm*, Journal of Experimental & Theoretical Artificial Intelligence, vol.28, no.4, pp.673-687, 2016.

[23] P. Viola, and M. J. Jones, *Robust Real-Time Face Detection*, International Journal of Computer Vision, vol.57, no.2, pp.137–154, 2004.
[24] A. G. Binsaadoon and E.-Sayed M. El-Alfy, *Gait-based Recognition for Human Identification using Fuzzy Local Binary Patterns*, In ICAART, vol.2,pp. 314-321, 2016.
[25] A. Rakhlin, A. Shvets, V. Iglovikov, and A. A. Kalinin, *Deep Convolutional Neural Networks for Breast Cancer Histology Image Analysis*, In proceedings of International Conference on Image Analysis and Recognition ICIAR, Image Analysis and Recognition, pp. 737-744, 2018.
[26] S. Zhang, Z. Deng, D. Cheng, M. Zong, and X. Zhu, *Efficient kNN Classification Algorithm for Big Data*, Neurocomputing, vol.195, pp.143-148, 2016.
[27] Ch. Mamatha, P. B. Reddy, M.A. R. Kumar, S. Kumar, *Analysis of Big Data With Neural Network*, International Journal of Civil Engineering and Technology (IJCIET), vol. 8, no.12, December 2017.
[28] C. Diamantini, D. Potena, and E. Storti, *Multidimensional query reformulation with measure decomposition*, Information Systems, vol. 78, pp. 23-39, November 2018.
[29] T. Cong N. D. H. Vo, P. V. Nguyen, H. M. Nguyen, and A. T. Vo, *Derivatives market and economic growth nexus: Policy implications for emerging markets*, North American Journal of Economics and Finance, 2019.
[30] B. T. Khoa, H. M. Nguyen, *The Relationship between the Perceived Mental Benefits, Online Trust, and Personal Information Disclosure in Online Shopping*, The Journal of Asian Finance, Economics and Business, vol. 6, no. 4, pp. 261-270, November 2019.
[31] V.H. Arul, V.G. Sivakumar, R. Marimuthu, and B. Chakraborty, *An Approach for Speech Enhancement Using Deep Convolutional Neural Network*, Multimedia Research (MR), vol. 2, no. 1, pp. 37-44, 2019.
[32] R. Hari, and M. Wilscy, *Event Detection in Cricket Videos Using Intensity Projection Profile of Umpire Gestures*, In proceeding of 2014 Annual IEEE India Conference (INDICON), Pune, India, 2014.
[33] M. M. Goyani, S. K. Dutta, and P. Raj, *Key Frame Detection Based Semantic Event Detection and Classification Using Heirarchical Approach for Cricket Sport Video Indexing*, Communications in Computer and Information Science, vol.131, pp. 388-397, 2011.
[34] S. K. Nerella, K. V. Gadi, and R. S. Chaganti, *Securing Images Using Colour Visual Cryptography and Wavelets*, International Journal of Advanced Research in Computer Science and Software Engineering, vol. 2, no. 3, pp. 163-168, 2012.
[35] R. V. Boppana, R. Chaganti, and V. Vedula, *Analyzing the Vulnerabilities Introduced by DDoS Mitigation Techniques for Software-Defined Networks*, National Cyber Summit (NCS) Research Track, pp 169-184, 2019.

Edited by: P. Vijaya
Received: Dec 7, 2019
Accepted: Jun 23, 2020