Pattern Recognition 59 (2016) 199–212

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Human action recognition using genetic algorithms and convolutional neural networks

Earnest Paul Ijjina *, Krishna Mohan Chalavadi

Visual Learning and Intelligence Group (VIGIL), Department of Computer Science & Engineering, Indian Institute of Technology Hyderabad, Telangana 502285, India

ARTICLE INFO

Article history:
Received 30 August 2015
Received in revised form
13 January 2016
Accepted 13 January 2016
Available online 23 January 2016

Keywords:
Convolutional neural network (CNN)
Genetic algorithms (GA)
Human action recognition
Action bank features

ABSTRACT

In this paper, an approach for human action recognition using genetic algorithms (GA) and deep convolutional neural networks (CNN) is proposed. We demonstrate that initializing the weights of a convolutional neural network (CNN) classifier based on solutions generated by genetic algorithms (GA) minimizes the classification error. A gradient descent algorithm is used to train the CNN classifiers (to find a local minimum) during fitness evaluations of GA chromosomes. The global search capabilities of genetic algorithms and the local search ability of gradient descent algorithm are exploited to find a solution that is closer to global-optimum. We show that combining the evidences of classifiers generated using genetic algorithms helps to improve the performance. We demonstrate the efficacy of the proposed classification system for human action recognition on UCF50 dataset.

1. Introduction

Inspired by biological neural networks, artificial neural networks were proposed for function approximation. Shortly after their introduction, the failure of shallow neural network models to classify non-linearly separable data resulted in the emergence of deep neural networks that contain more than two hidden layers but lacked an effective training algorithm due to vanishing gradient problem [1]. In the last decade, the advancements in computational capabilities and the introduction of effective approaches to train deep neural network architectures has lead to their wide usage to address various computer vision challenges. Some of the well-known machine learning tasks addressed by deep neural network models include MNIST handwritten digit recognition [2], ILSVRC object recognition [3] and facial expression recognition in the wild [4]. A convolutional neural network is the most popular approach among deep neural network model that generally consists of an alternating sequence of convolution and sub-sampling layers.

In the recent years, human action recognition in videos has become a major domain of research due to its applications in video retrieval, sports analysis, health monitoring, human computer interaction and video surveillance. Several surveys papers were published in the literature, each one emphasizing a particular characteristic of recognition. The various methodologies for recognizing actions performed by a single person are covered in [5,6] focuses on the approaches to classify full body motions by categorizing them into spatial and temporal structures. Approaches for multi-view 2D and 3D human action recognition are discussed in [7]. Several human action recognition datasets were proposed in the literature [8] to address different types of problems like recognition of realistic activities, interaction and multi-view analysis from varying sources. Most of the action recognition techniques rely on some extracted features or descriptors for discriminative information for classification. Some of the most commonly used features/descriptors for human action recognition are bag-of-visual-words (BoVW) [9], histograms oriented gradient (HOG) [10], histograms of optical flow (HOF) [10], motion boundary histograms (MBH) [11], action bank features [12] and dense trajectories [13]. Xiaodan Liang et al. [14] proposed a hierarchical human action recognition system by modeling each observation as an ensemble of spatio-temporal compositions. The latent structure of actions is represented by spatio-temporal and-or graphs with the leaf-nodes containing the spatial and temporal contextual interactions. The inability of these approaches to scale across multiple datasets has lead to the research on learning from data. In the recent years, deep learning gained a lot of focus due to its ability to learn features from data [15]. The effectiveness of convolutional neural networks for object recognition was demonstrated in ILSVRC [3] (IMAGENET large scale visual recognition challenge) [16,17] after which it was used to address various other visual recognition tasks like face recognition [18], facial expression

* Corresponding author.
E-mail addresses: cs12p1002@iith.ac.in (E.P. Ijjina), ckm@iith.ac.in (K.M. Chalavadi).

http://dx.doi.org/10.1016/j.patcog.2016.01.012

recognition [19,4], video quality assessment [20] and action recognition [21–23].

The convolutional neural network (CNN) introduced by LeCun et al. in [24,25] is the most popular deep neural network model in use for computer vision problems. One of the major initial attempts to use CNN for action recognition was by Baccouche et al. in [26]. In this work, a 3D convolutional neural network is trained to assign a vector of features to a small number of consecutive frames. The spatio-temporal evolution of these features is used by a recurrent neural network for classification. In [21], Shuiwang Ji et al. extracted gray, gradient and optical-flow information along x and y directions from video frames and used them as input to a 3D CNN model for human action recognition in surveillance videos. Keze Wang et al. proposed a deep learning model for human activity recognition in [27] by extending a CNN to incorporate structure alternatives by using latent variables in convolutional layers to manipulate the activation of neurons. The variation in temporal composition of activities during recognition in handled through partial activation of network configuration. A spatio-temporal CNN is used by Liang Lin et al. in [28] to decompose videos into temporal segments of sub-activities. The model is iteratively optimized by a learning algorithm with radius-margin regularization for human action recognition in RGBD videos. Guilhem Chéron et al. proposed a pose-based CNN descriptor for human action recognition in [29], that extracts and aggregates appearance and flow information at characteristic positions obtained from human pose. A differential recurrent neural network to model the temporal evolution of state dynamics is proposed by Vivek Veeriah et al. in [30] for action recognition. A differential gating scheme emphasizing the information gain caused by salient motions between successive frames is used to learn spatio-temporal dynamics associated with salient motion patters. Simonyan et al. proposed a two-stream convolutional network for action recognition [31] that uses appearance from still frames (spatial information) and motion between frames (temporal information) as separate recognition streams. The softmax scores of the two streams are combined using late fusion for classification.

Deep learning aims to learn multiple levels of representation with an intent to discover high-level abstractions for discrimination. In spite of the expressive power of deep architectures [32], learning in deep architectures [33] is still a challenge. Since 2006, several deep learning algorithms like greedy layer-wise training of deep networks [34] that initializes weights by greedy layer-wise unsupervised training, a fast learning algorithm for deep belief nets [35] and strategies for training deep neural networks [36] were proposed. There has been studies on the difficulty of training a deep feed-forward neural network [37] and techniques to improve generalization like: (1) early stopping [38] to avoid overfitting, (2) dropout [39] to avoid co-adaptation by randomly dropping neural units during training, (3) use of rectified linear units [40] whose activation function has linear response in a short range, (4) unsupervised pre-training for effective initialization of weights in deep neural networks [41], and (5) the importance of a well-designed initialization of network in deep learning [42]. There are even studies confirming that randomly chosen trails may be more effective than grid search and manual search as they effectively search a larger and less promising configuration space for hyper-parameter optimization [43].

To address these challenges in training deep neural networks, we explore the use of evolutionary algorithms (genetic algorithms in particular) for optimization of weights of neural network. In the literature, genetic algorithms (GA) were used to optimize neural network systems by feature selection [44,45], topology selection [46,47], weight selection [48,49]. GA is also used to optimize both weights and topology simultaneously [50–53]. Most of the existing evolutionary neural networks [54–56,49] are shallow and a straightforward optimization of a deep neural network weights could be computationally expensive. Some of the approaches using GA for training deep neural networks includes the one proposed by David et al. [57] to optimize a sparse autoencoder by learning the weights using GA assisted back-propagation. In [58], Oullette et al. used genetic algorithm to train the weights of a CNN without getting trapped in a local minimum. The trained classifier is used for crack detection and was evaluated on a dataset of 100 images. In [59], Fedorovici et al. proposed the use of evolutionary optimization techniques like gravitational search algorithm [60] and particle swarm optimization [61] to find the optimum weights of a convolutional neural network. The weights of CNN are further optimized using back-propagation algorithm for optical character recognition. Koutník et al. proposed an online evolutionary training algorithm [62] for driving a race car in TORCS racing simulator using recurrent neural network controller and max-pooling convolutional neural network for feature extraction. The controller and CNN are simultaneously optimized using CoSyNE [63] using the images generated due to the turn and speed predictions of the controller.

In this work, we propose a hybrid search approach for training the weights of a convolutional neural network classifier exploiting the efficient global and local search abilities of evolutionary and classical optimization algorithms for the prediction of human actions in unconstrained videos. The novelty of the proposed approach lies in: (1) modeling a convolutional neural network classifier as a GA-chromosome and its use in improving classification performance, (2) the use of genetic algorithms to explore different basins (weight initializations) in the parameter space and steepest-descent algorithm to expedite the search for finding the local optimum in a given basin, and (3) combining evidences from classifiers (that are generated by GA-framework) to overcome the limitation of individual classifiers. The reminder of the paper is organized as follows: Section 2 describes the proposed approach and its rationale. The details of the experimental set up and performance analysis are discussed in Section 3. Finally, the conclusions and future work are presented in Section 4.

2. Proposed approach

In this work, we present a hybrid approach to train a CNN classifier by effective utilization of global and local search capabilities of genetic and steepest-descent algorithms, respectively. Training a neural network using gradient-descent algorithm may result in finding a solution that is stuck in a local minimum. As the performance of a trained neural network classifier depends on its initial weights, we explore different sets of initial weights to find the optimum weight initialization using genetic algorithms. The weights of masks in convolution layers (that act as feature detectors) and the seed value used by the random number generator to initialize the fullyconnected neural network are considered as the GA chromosome, as shown in Fig. 1. The proposed approach begins with the initialization of GA population, followed by the fitness evaluation step in GA framework. During fitness evaluation, the fitness score of each chromosome in the GA population is computed by decoding the chromosome to initialize the weights of a CNN classifier, as illustrated in step 2 of Fig. 1. The classification accuracy of the CNN classifier, after being trained for

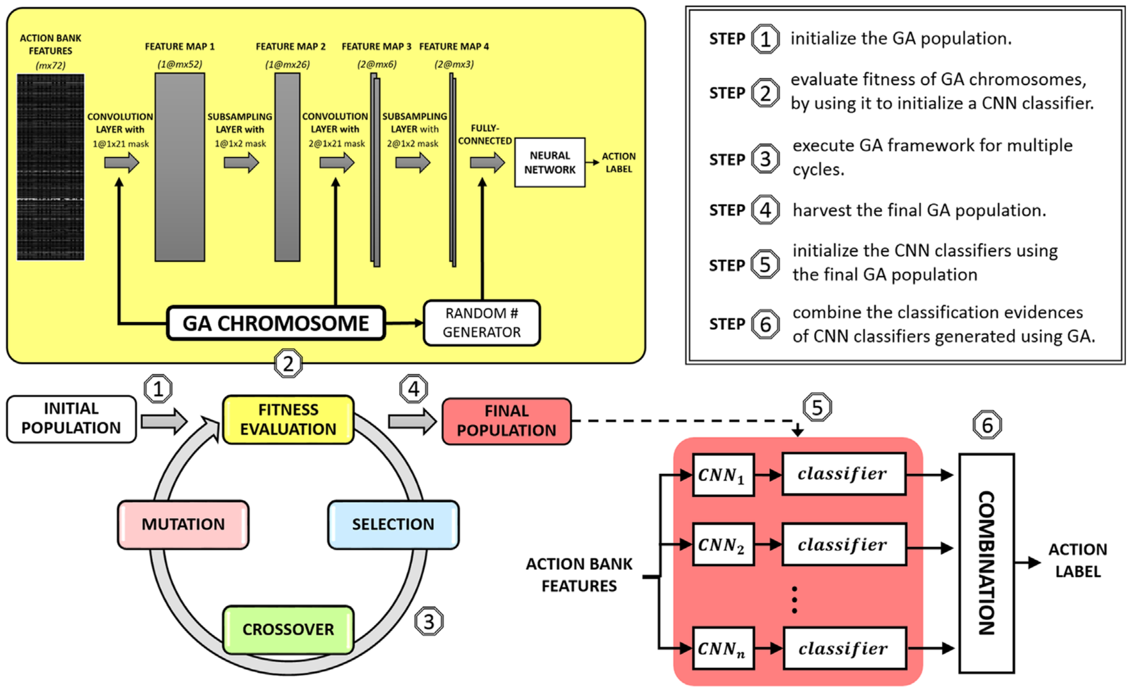


Fig. 1. Overview of the proposed classification system. The various steps involved in the proposed approach are numbered and a short description of each step is given on the top right corner. Best viewed in color.

p_1 epochs using steepest descent algorithm, is considered as the fitness value of the corresponding GA chromosome. Using GA, several local basins were identified and the steepest-descent algorithm is used to expedite the search to find the local optimum in a given basin. After executing the GA framework for several cycles with a population size of n, the final GA-population is harvested to obtain n sets of initial weights. These n sets of initial weights are used to initialize the convolutional neural network (CNN) classifiers as shown in step 5 of Fig. 1. The classification evidences of these n convolutional neural network classifiers is combined to improve the performance. The next subsection introduces genetic algorithms and explains how the fitness of a chromosome (quality of solution) improves over GA cycles.

2.1. Genetic algorithms

Genetic algorithm is an adaptive heuristic search method based on the evolutionary ideas of natural selection and genetics proposed by Holland [64]. Inspired by the Darwin's Theory of evolution (survival of the fittest) [65], this approach considers a population of GA chromosomes (candidate solutions) that go through a series of changes due to selection, crossover and mutation (operations) resulting in a modified set of chromosomes at the end of each GA cycle. Assuming that the GA-chromosome captures the key characteristics of the system being modeled, the average fitness of the population is expected to improve over generations due to the use of fitness measure (quality of the solution) of GA chromosome in GA-operations. Refer [66] for a comprehensive overview of genetic algorithms. The next section describes the fusion of evidences from multiple classifiers for performance evaluation.

2.2. Combining evidences from multiple classifiers

If $o_1, o_2, \ldots, o_c$ are the binary decoded outputs of a classifier, then the classifier is trained to output $o_p = 1$ and $o_j = 0$, for all $j \neq p$ and $1 \leq j \leq c$ for an observation of class p, where c represents the number of classes. During testing, an observation will be labeled as class p if $o_p > o_j$, for all $j \neq p$ and $1 \leq j \leq c$. The fusion (combination) of evidences across n classifiers involves the use of a fusion function like *Max*-rule, across the same index of classifier outputs to find the binary decoded output of the combined model. If $o_{1i}, o_{2i}, \ldots, o_{ci}$ are the outputs of the ith classifier in the combined model, the jth output of the combined model is defined as $f_j = \max\{o_{j1}, o_{j2}, \ldots, o_{jn}\}$. An observation will be labeled as class p by the combined model if $f_p > f_j$, for all $j \neq p$ and $1 \leq j \leq c$. Combining evidences across classifiers would generally result in a classifier that correctly labels the observations which are misclassified by some classifiers (the limitation of a single classifier). An overview of ensemble methods is given in [67]. The next subsection introduces the representation of videos as action bank features and describes the architecture of CNN classifier used for human action recognition.

2.3. CNN classifier for human action recognition

In this section, we describe the underlying principles in the computation of action bank features for a video. We will later explain some of the characteristics and advantages of action bank features that motivated us in their use as input features. Finally, the design of the convolutional neural network classifier for human action recognition from action bank features is explained in detail.

2.3.1. Input features

Introduced by Sadanand et al. in [12], the action bank representation of videos is a high level representation used for activity recognition. An action bank is a collection of multiple action detectors covering a broad semantic and viewpoint space. An action detector is a template video of an action. Some of the action detectors in the action bank are shown in Fig. 2 with columns depicting different types of actions and rows indicating different examples for the corresponding action.

Fig. 2. A screen-shot of 36 videos in the standard action bank with 205 elements. Best viewed in color. (Fig. Fig. 2 in [12]). (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

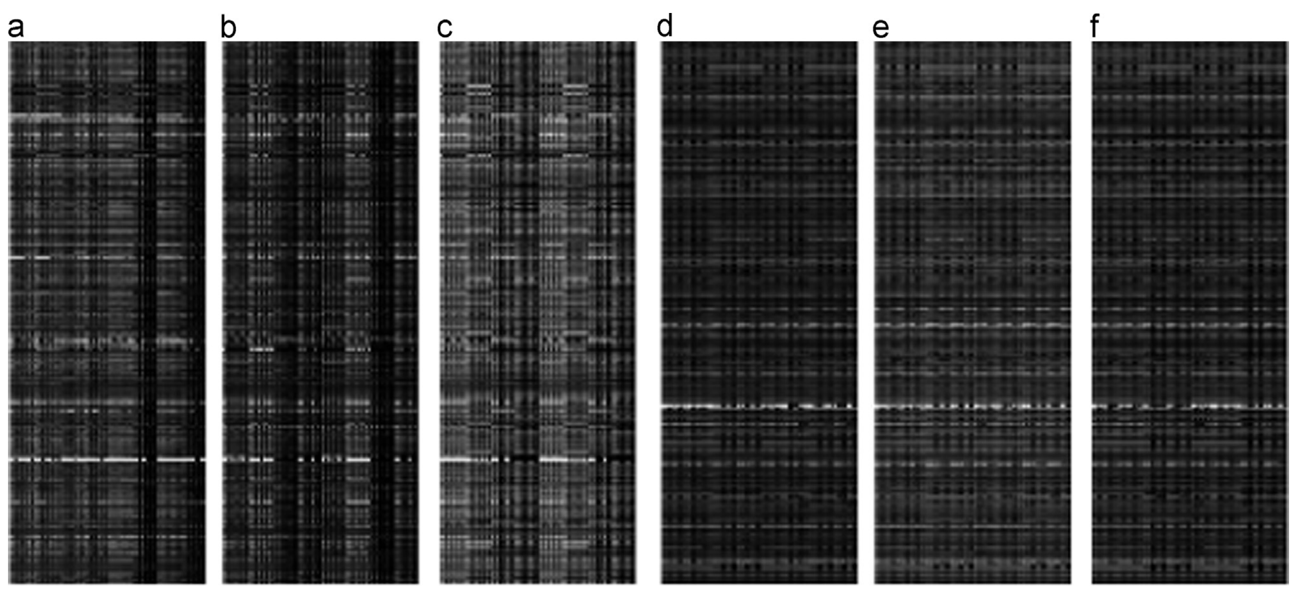


Fig. 3. Action bank representation of boxing and running videos in KTH dataset: (a)–(c) are for boxing and (d)–(f) are for running action.

To generate action bank features for a video, the correlation video volume of each action detector is transformed into a 73-dimensional response vector by volumetric-max-pooling. Thus, if an action bank of size m is used for computing action bank features of a video, the generated action bank features will be of size $m \times 73$. Since, an action detector may have similar response vector for multiple instance of the same action, their action bank representation may also have similar local patterns. The action bank representation of boxing and running videos from KTH dataset is shown in Fig. 3.

It can be observed that videos of same action will have similar local patterns corresponding to some action detectors, depending on their nature and extent of similarity. Therefore, it is possible to discriminate actions by using a pattern recognition approach that can learn local patterns associated with each action. In this work, a convolutional neural network classifier capable of recognizing local patterns with some degree of noise is used to recognize human actions from action bank features.

2.3.2. *Configuration of CNN classifier*

A convolutional neural network (CNN) classifier comprises of a convolutional neural network for feature extraction and a classifier in the last step for classification. The architecture of CNN classifier used for human action recognition from action bank features is shown in Fig. 4. To avoid padding during computation, the first 72 elements of action bank features are considered, resulting in an input of size $m \times 72$. Here, m represents the size of action bank used for generating action bank features. During training, the convolution masks are learned to recognize the necessary discriminative local patterns for classification. As the local patterns in action bank features are horizontal and independent of its vertical neighbors, only linear (horizontal) convolution masks are used in the CNN classifier. A single convolution mask is considered in the

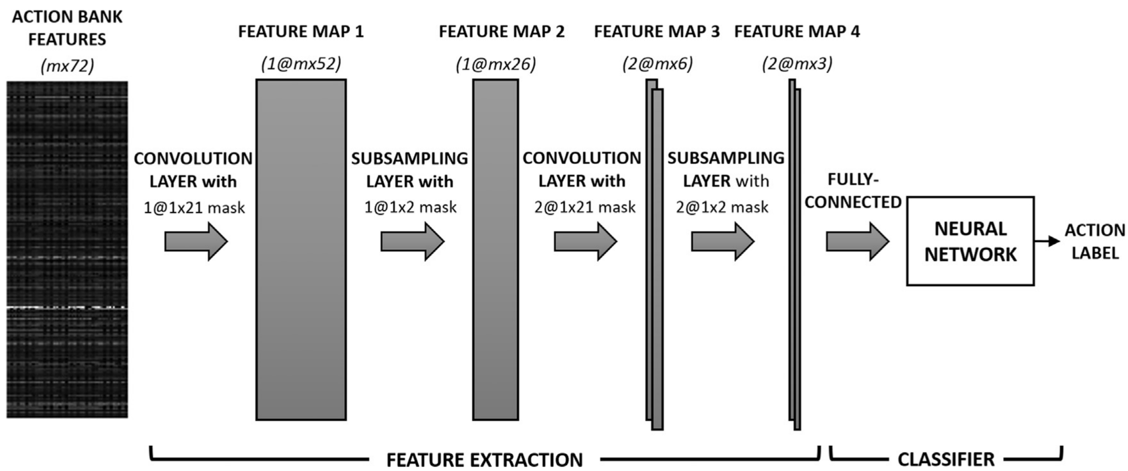


Fig. 4. Architecture CNN classifier for human action recognition.

first convolution layer due to the simplicity of the pattern being recognized (a white line) and to minimize the computational complexity. We doubled the number of masks in the respective succeeding layers and used two convolution masks in the second convolution layer. In addition, to use the same mask size in both convolution layers, we chose a mask size of 1×21. The sub-sampling masks of size 1×2 are used to minimize the loss of data during sub-sampling. The deep convolutional features extracted by CNN are given as input to a fully connected, single layer neural network for classification. The action labels are determined from the binary decoded outputs of the classifier.

As the convolution masks in CNN classifier act as feature detectors, optimal initialization of these kernels is crucial for the design of an effective CNN classifier. The next subsection explains the initialization and training of this CNN classifier.

2.4. Training a CNN classifier using genetic algorithms and back-propagation algorithm

One of the major limitation of training a neural network using steepest-descent algorithm is the possibility of solution getting stuck in a local optimum. To overcome this problem, we use genetic algorithms whose solutions evolve over generations. In this work, we explore the use of genetic algorithms to identify the optimum weight initialization of the CNN classifier discussed in the previous section. A genetic algorithm (GA) chromosome of 64 real numbers is used to represent the weights of CNN classifier, in which the first 63 real numbers are used to encode the three convolution masks of size 1×21. The last real number is for the seed value of the random number generator that initializes the fully connected neural network classifier shown in Fig. 1. The classification error of the CNN classifier initialized using a GA chromosome after training with back-propagation algorithm for p_1 epochs is used as the fitness value of the GA chromosome. The CNN classifier is trained using back-propagation algorithm for a small number of epochs (p_1) to avoid over-fitting the data. As the performance of a gradient-descent algorithm depends on the initial (starting) weights of the neural network, the use of genetic algorithm to explore different weight initializations may result in finding weight initializations that would lead to a better solution than random initialization. Thus, by exploring different basins (weight initializations) using GA, we aim to find a solution that is closer to global-optimum. The use of steepest descent algorithm, to quickly find the local optimum in a given basin, reduces the number of candidate solutions (initial weights) to be explored by GA to find local optimum of basins. The next section discusses the experimental results.

3. Experimental results

The proposed CNN classifier approach is implemented by customizing the deep learning toolbox [68] to use linear masks and using the native GA functionality available in Matlab. The range of weights in convolution masks is in between -100 and 100. The range of seed value is 0 to 5000. The GA with a population size of 20 (n in Fig. 1) is run for 5 generations considering a cross-over probability of 0.8 and mutation probability of 0.01. Low mutation probability is used in the GA-framework as GA relies on the construction capability of crossover operator rather than on the disruptive power of mutation operator. The optimum range of these parameters and GA configuration is determined empirically. By expediting local-search using steepest-descent algorithm, we aim to find an optimal solution even with a small number of GA-generations and population size. The experimental results on UCF50 dataset are discussed below.

3.1. UCF50 dataset

The proposed approach is evaluated on UCF50 dataset [69], that consists of unconstrained realistic videos for 50 action categories taken from Youtube. UCF50 dataset is selected due to its high number of action categories and the availability of pre-computed action bank features [70]. The use of pre-computed action bank features in this work, facilitates the comparative study with existing approaches. The evaluation is done using 5-fold cross-validation. Here, k-fold cross-validation refers to splitting the dataset into k splits say $S_1, S_2, \ldots, S_k$ followed by using split S_i for testing and the remaining (k–1) splits for training in *Fold-i*. This process is repeated k times as i is varied from 1 to k. During fitness computation of GA-chromosomes, the initialized CNN classifier is trained using back-propagation algorithm in batch mode for 50 (p_1) epochs. A batch-size of 10 is used for the first four folds and 8 for the fifth fold. The best and mean fitness value (indicating the classification error in %) of population in each GA generation, for the 5-folds of UCF50 dataset (on training data) is given in Table 1.

Table 1
Best and mean fitness (classification error in %) of GA population across generations for the 5 folds in UCF50 dataset.

Generation	*Fold-1*		*Fold-2*		*Fold-3*		*Fold-4*		*Fold-5*	
	Best	Mean	Best	Mean	Best	Mean	Best	Mean	Best	Mean
1	9.21	59.57	3.18	45.7	8.67	64.9	6.15	44.5	5.79	41.6
2	5.57	38.78	3.10	23.8	3.47	51.2	3.04	11.6	5.79	32.5
3	3.49	33.19	2.95	5.8	3.47	35.2	1.82	4.2	3.73	8.0
4	2.67	5.66	2.87	3.6	2.18	18.4	1.52	3.1	3.04	3.8
5	2.45	3.45	2.80	3.4	1.81	2.4	1.44	2.8	3.04	3.5

Table 2
Performance of candidate solutions (in %) generated from final GA population on test data using neural network classifier and extreme learning machine (ELM) classifier for the 5-folds of UCF50 dataset. (Here Avg represents the average performance across all candidate solutions).

Sol.	Neural network (NN) classifier					Extreme learning machine (ELM) classifier				
#	*Fold-1*	*Fold-2*	*Fold-3*	*Fold-4*	*Fold-5*	*Fold-1*	*Fold-2*	*Fold-3*	*Fold-4*	*Fold-5*
1	97.40	97.20	98.19	98.48	96.95	100.00	99.70	100.00	100.00	100.00
2	96.36	95.98	98.19	84.18	96.42	100.00	99.77	100.00	97.87	100.00
3	96.88	96.97	96.60	98.02	96.65	100.00	99.85	100.00	100.00	100.00
4	97.03	96.82	97.81	98.17	96.11	100.00	99.85	100.00	100.00	100.00
5	97.55	96.59	96.91	98.33	96.65	100.00	99.85	100.00	100.00	100.00
6	97.40	96.97	98.19	98.33	96.49	100.00	99.77	100.00	100.00	100.00
7	97.17	96.52	97.74	96.88	96.65	100.00	99.92	100.00	99.77	100.00
8	96.88	96.89	97.74	98.17	96.34	100.00	99.77	100.00	100.00	100.00
9	97.40	96.52	98.04	98.02	96.95	100.00	99.85	100.00	100.00	100.00
10	90.93	97.05	98.04	97.72	96.80	100.00	99.77	100.00	99.85	100.00
11	97.10	96.89	97.89	97.57	96.80	100.00	100.00	100.00	100.00	100.00
12	96.95	96.74	97.06	97.03	96.49	100.00	99.77	100.00	99.92	100.00
13	92.86	96.14	98.11	98.25	96.49	100.00	99.70	100.00	100.00	100.00
14	96.58	96.74	97.96	98.40	96.57	100.00	99.92	100.00	100.00	100.00
15	96.80	95.91	98.11	98.48	96.65	100.00	99.77	100.00	100.00	100.00
16	97.17	97.12	97.43	98.56	96.42	100.00	99.70	100.00	100.00	100.00
17	96.51	96.06	96.60	96.05	96.27	100.00	99.77	100.00	100.00	100.00
18	97.40	95.91	97.81	98.33	96.57	100.00	99.17	100.00	100.00	100.00
19	97.10	96.67	95.09	98.25	95.96	100.00	99.85	98.94	100.00	100.00
20	97.40	96.52	97.89	96.43	96.49	100.00	99.85	100.00	99.77	100.00
Avg	96.50	96.60	97.56	97.18	96.53	100.00	99.78	99.94	99.85	100.00

The consistent decrease in mean and best fitness value of the population over generations for the 5-folds of UCF50 dataset indicates the proper selection of GA parameters. This also confirms the proper balance between exploration (due to mutation) and exploitation (due to crossover). This completes step 3 of Fig. 1 and produces 20 (n) candidate classifier initializations for each fold.

As mentioned in steps 5 and 6 of Fig. 1, the candidate solutions are used to initialize the CNN classifiers and their classification evidences are combined to assign the class labels. The performance of the n CNN classifiers using neural network and extreme learning machine (ELM) [71] classifiers is given in Table 2. From the average accuracy given in the last row of this table, it can be observed that extreme learning machine (ELM) classifier gives better performance than neural network classifier. This could be due to the better generalization capability of ELM over gradient-based training algorithms.

As discussed in step 6 of Fig. 1, the n classifiers generated at the end of step 5 are used as base classifiers in an ensemble model and various fusion functions are considered to combine their classification evidences. The performance on UCF50 dataset for various folds with different fusion functions using ELM classifier is given in Table 3. It can be observed that the performance remains the same irrespective of the fusion-rule. This may be due to the small deviation in performance of the classifiers used in the ensemble.

Table 3
Performance of the proposed classification system (in terms of # of misclassified observations) using ELM classifier with various fusion functions for 5-fold cross-validation of UCF50 dataset.

Data fold	Number of observations	Fusion function					*Majority voting*
		Min	*Max*	*Avg*	*Prod*	*Median*	
Fold-1	1345	0	0	0	0	0	0
Fold-2	1320	1	1	1	1	1	1
Fold-3	1325	0	0	0	0	0	0
Fold-4	1315	0	0	0	0	0	0
Fold-5	1312	0	0	0	0	0	0
Total	6617	1	1	1	1	1	1
Accuracy (in %)=		99.98	99.98	99.98	99.98	99.98	99.98

From Table 3, it can be observed that one observation gets misclassified irrespective of the fusion rule. Thus, a classification accuracy of 99.98% is achieved by the proposed approach for 5-fold cross-validation of UCF50 dataset. The confusion matrix of the proposed approach for UCF50 dataset is shown in Table 4. The labels on the vertical axis indicate the true class labels and the labels on the horizontal axis indicate the predicted class labels.

Table 4
Confusion matrix of the proposed approach for 5-fold cross validation of UCF50 dataset.

	BaseballPitch	Basketball	BenchPress	Biking	Billiards	BreastStroke	CleanAndJerk	Diving	Drumming	Fencing	GolfSwing	HighJump	HorseRace	HorseRiding	HulaHoop	JavelinThrow	JugglingBalls	JumpRope	JumpingJack	Kayaking	Lunges	MilitaryParade	Mixing	Nunchucks	PizzaTossing	PlayingGuitar	PlayingPiano	PlayingTabla	PlayingViolin	PoleVault	PommelHorse	PullUps	Punch	PushUps	RockClimbingIndoor	RopeClimbing	Rowing	SalsaSpin	SkateBoarding	Skiing	Skijet	SoccerJuggling	Swing	TaiChi	TennisSwing	ThrowDiscus	TrampolineJumping	VolleyballSpiking	WalkingWithDog	YoYo
BaseballPitch	150																																																	
Basketball		137																																																
BenchPress			160																																															
Biking				145																																														
Billiards					150																																													
BreastStroke						101																																												
CleanAndJerk							112																																											
Diving								153																																										
Drumming									161																																									
Fencing										111																																								
GolfSwing											142																																							
HighJump												123																																						
HorseRace													127																																					
HorseRiding														196																																				
HulaHoop															125																																			
JavelinThrow																117																																		
JugglingBalls																	122																																	
JumpRope																		144																																
JumpingJack																			123																															
Kayaking																				157																														
Lunges																					141																													
MilitaryParade																						127																												
Mixing																							141																											
Nunchucks																								132																										
PizzaTossing																									114																									
PlayingGuitar																										160																								
PlayingPiano																											105																							
PlayingTabla																												111																						
PlayingViolin																													100																					
PoleVault																														160																				
PommelHorse																															123																			
PullUps																																100																		
Punch																																	160																	
PushUps																																		102																
RockClimbingIndoor																																			144															
RopeClimbing																																				130														
Rowing																																					140													
SalsaSpin																																						133												
SkateBoarding																																							120											
Skiing																																								144										
Skijet																																									100									
SoccerJuggling																																										156								
Swing																																											137							
TaiChi																																												100						
TennisSwing																																													167					
ThrowDiscus																																														131				
TrampolineJumping																																															119			
VolleyballSpiking																																																116		
WalkingWithDog																																			1														122	
YoYo																																																		128

The diagonal elements represent the correctly predicted test cases and the non-diagonal elements represent the misclassified test cases. It can be observed that one test instance of *WalkingWithDog* is misclassified as *RockClimbingIndoor* by the proposed approach.

The performance of the CNN classifier when trained using back-propagation algorithm (BPA), genetic algorithms (GA) and both is given in Table 5. It can be observed that CNN classifiers whose weights are initialized using GA and trained using back-propagation algorithm gives better performance than the rest of the approaches. As a set of solutions gets generated when GA is used for training, the average performance of the resulting classifiers is reported in the table. Here, experiments for training using only GA were conducted with population a size of 200 for 5 generations. Thus, training the CNN classifier initialized by GA with BPA finds an optimal solution in less number of generations even with a small population size. The performance of the proposed classification framework using neural network (NN) and extreme learning machine (ELM) classifiers is shown in Table 6. From the table, it can be concluded that better performance can be achieved using ELM classifier compared to NN classifier. From Table 3, it can be concluded that the performance of the proposed approach using an ensemble of CNN classifiers employing ELM classification is 99.98% as one observation among 6617 test cases got misclassified. The performance of the proposed approach against exiting techniques for 5-fold cross-validation on UCF50 dataset is given in Table 7.

It can be observed from Table 7 that an accuracy of 94.1% is achieved by Nicolas Ballas et al. in [77] by building an action model from salient regions using spatio-temporal context and weighted SVM. In the proposed approach, a classification accuracy of 99.98% is achieved using action bank features. The improvement in classification performance using the proposed classification system is indicative of the effectiveness of the proposed hybrid search approach to find optimum initial weights of the CNN classifier. The next section analyzes the results presented in this section.

3.2. Analysis

The two important issues in training a neural network using back-propagation are: (1) over-fitting of training data, and (2) the possibility of solution getting stuck in a local minimum. When the neural network is over-fit due to excessive training, the error on training set will be very low but on testing set will be high. In this section, we analyze the classification error of CNN classifiers immediately after weight initialization using GA, and also after training the CNN classifier using back-propagation for 50 (p_1) epochs. The weights of a CNN classifier is represented by a circle in a 2D plane with the *x*-axis representing the percentage of classification error on training data and *y*-axis representing the percentage of classification error on testing data.

The graph in Fig. 5(a) shows the classification error of weight initializations explored by GA for the first fold of UCF50 dataset.

Table 5
Performance of CNN classifier (in %) using back propagation algorithm (BPA), genetic algorithms (GA) and both for 5-fold cross-validation of UCF50 dataset.

Training approach	*Fold-1*	*Fold-2*	*Fold-3*	*Fold-4*	*Fold-5*	Average
CNN classifier with *only GA* (i.e., initialized using GA)	2.22	1.87	2.18	1.98	1.87	2.02
CNN classifier *without GA* (i.e., trained using BPA)	86.02	87.50	18.26	80.30	23.39	59.19
CNN classifier *with GA* (using GA and BPA)	96.54	96.60	97.56	97.18	96.53	96.88

Table 6
Performance of the proposed classification framework (in %) using neural network (NN) and extreme learning machine (ELM) classifiers for 5-fold cross-validation of UCF50 dataset.

Classification methodology	*Fold-1*	*Fold-2*	*Fold-3*	*Fold-4*	*Fold-5*	Average
Proposed framework using *NN classifier*	96.54	96.60	97.56	97.18	96.53	96.88
Proposed framework using *ELM classifier*	100	99.78	99.94	99.85	100	99.91

Table 7
Performance comparison of the proposed approach with existing techniques for 5-fold cross-validation on UCF50 dataset.

Approach	Accuracy (in %)
Sadanand and J. Corso [12]	57.9
Kliper-Gross et al. [72]	68.51
Shi Feng et al. [73]	71.7
LiMin Wang et al. [74]	71.7
H. Wang et al. [13]	75.7
Qiang Zhou et al. [75]	80.2
Ijjina Earnest et al. [76]	94.02
Nicolas Ballas et al. [77]	94.1
Proposed approach	**99.98**

Fig. 5(b) depicts the solutions in Fig. 5(a) trained with back-propagation algorithm for 50 epochs. Each circle in these graphs represents a CNN classifier whose weights are initialized using GA. The location of the circle is determined by the classification error of the CNN classifier for train and test data. The color of the circle indicates when (the time) the weight initialization is explored during the GA-cycles. As shown by the scale in the right-hand of these figures, blue color is assigned to solutions (weight initializations) explored in the first generation and yellow color to the solutions explored in the last generation. The same convention is used to represent the solutions for *Fold-2*, *Fold-3*, *Fold-4* and *Fold-5* in Fig. 6. From the sub-figures in Figs. 5 and 6, it can be observed that: (1) the proposed initialization of CNN classifiers using GA and post-training using back-propagation algorithm significantly improves the performance of classification system, (2) the location of circles closer to 45° diagonal line indicates the existence of similar local-patterns for actions in both training and testing data. (This may be due to the use of k-fold cross-validation), (3) the high concentration of circles in the top-right corner in graphs depicting the solutions initialized using GA indicates the use of GA to identify optimum initial weights of the classifier rather than the final weights used for fitness computation, and (4) the high concentration of yellow circles closer in the bottom left corner (area with low classification error) in graphs with solutions trained using back-propagation algorithm demonstrates the improvement of solutions generated by GA over generations. The most likely reasons for misclassification of *WalkingWithDog* observation in Fig. 7 by the proposed approach are the large variation in illumination conditions, change in scale and the existence of camera shake. The predicted top-5 class labels for this observation using the proposed approach with various fusion rules is given in Table 8. It can be observed that 100% prediction accuracy is achieved by the proposed approach if top-2 predictions are used for performance evaluation. The feasibility to extend this approach to solve problems in other domains, is demonstrated by evaluating this approach for handwritten character recognition on MNIST dataset.

3.3. MNIST dataset

The recognition of hand-written characters using computer vision algorithms is a challenging task with practical applications. The MNIST dataset [25] is one of the standard benchmark used to compare performance of different approaches. The proposed approach is evaluated on MNIST dataset, using GA with a population size of 10 (n) for 3 generation. The optimum size of convolution and sub-sampling masks is empirically determined to be 5×5 and 1×1, respectively. The CNN classifier is trained using back propagation algorithm in batch mode with a batch size of 10 for 10 (p_1) epochs. The average fitness value of GA population decreases from 79.45 in the first generation to 14.56 in the last generation, suggests the convergence of GA.

The performance of the CNN classifier trained using back propagation algorithm (BPA), genetic algorithms (GA) and both is given in Table 9. The table shows the performance of CNN classifier *without GA* (i.e., trained using BPA) against the average performance of solutions generated *with GA* (i.e., using GA and BPA). As the best performance and standard deviation of solutions generated by *with GA* training approach are 96.92% and 22.91, respectively, it can be concluded that CNN classifiers initialized by genetic algorithms and trained with back propagation algorithm gives better performance than the rest of the approaches. The performance of the proposed classification framework using neural network (NN) and extreme learning machine (ELM)

Table 8
Top 5 class labels predicted by the proposed approach using various fusion rules, for the misclassified *WalkingWithDog* observation. (Here, RCI denotes *RockClimbingIndoor*, WWD represents *WalkingWithDog*, P denotes *Punch*, L denotes *Lunges*, D represents *Diving*, K denotes *Kayaking* and HR denotes *HorseRiding* action).

Top	Fusion-rule				
#	*Min*	*Max*	*Avg*	*Prod*	*Median*
1	RCI	RCI	RCI	RCI	RCI
2	*WWD*	*WWD*	*WWD*	*WWD*	*WWD*
3	P	L	P	P	P
4	L	P	L	L	L
5	D	K	HR	HR	HR

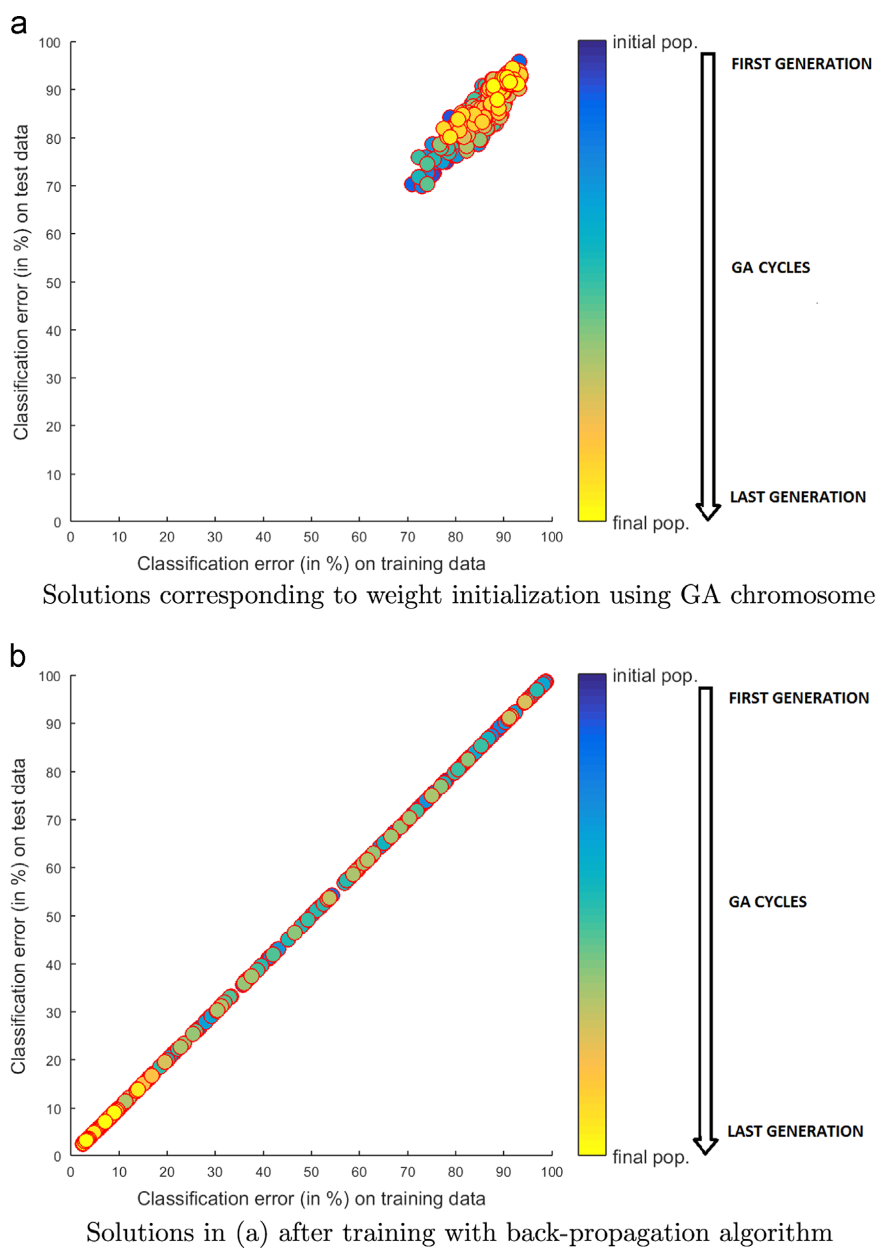


Fig. 5. Solutions explored by the proposed approach for *Fold-1* of UCF50 dataset: (a) after initialization using GA chromosomes and (b) after training the classifier using back-propagation algorithm for p_1 epochs. Best viewed in color. (a) Solutions corresponding to weight initialization using GA chromosome, and (b) Solutions in (a) after training with back-propagation algorithm.

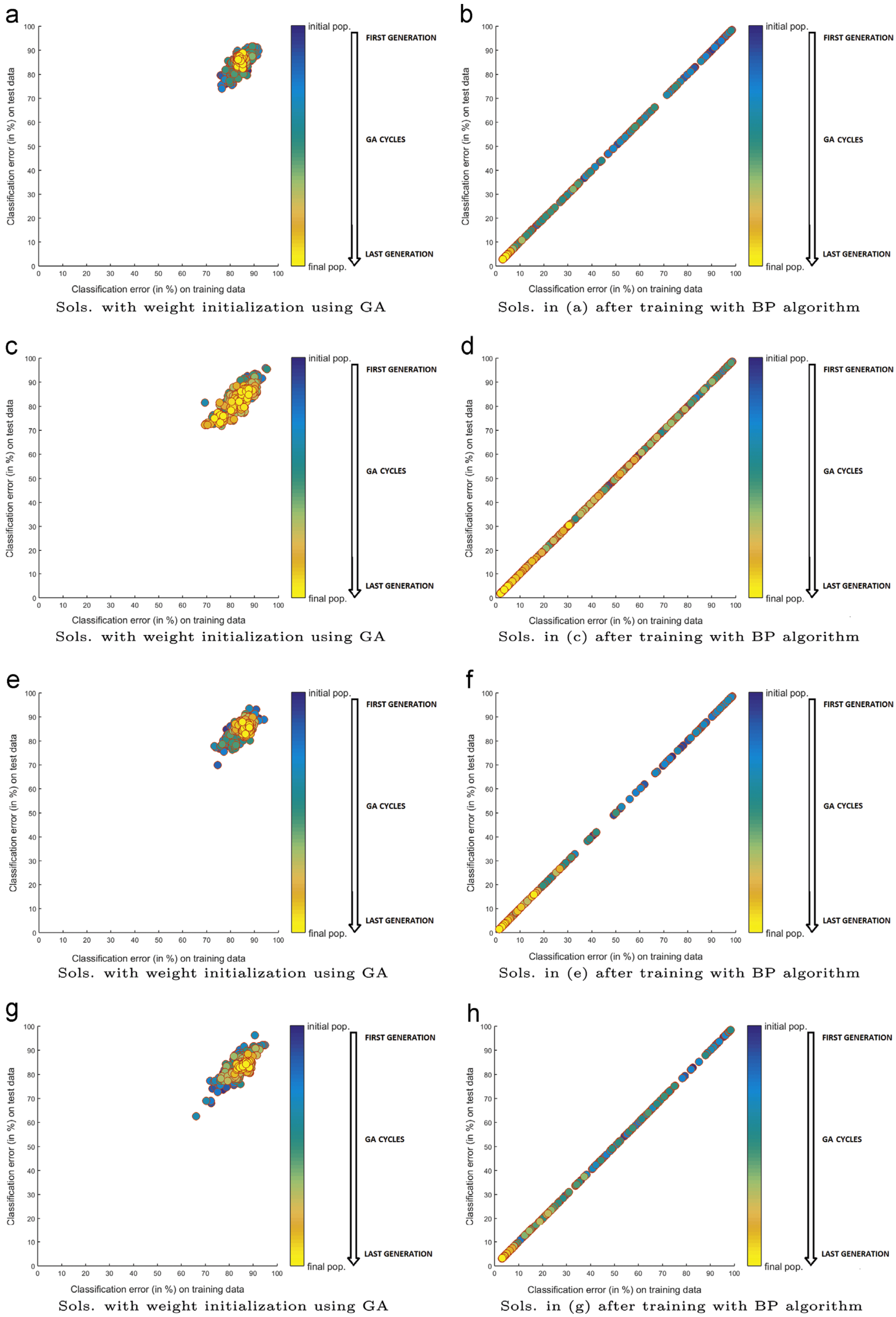


Fig. 6. Solutions explored by the proposed approach for *Fold-2*, *Fold-3*, *Fold-4* and *Fold-5* of UCF50 dataset. The sub-figures (a), (b) correspond to *Fold-2*; (c), (d) are for *Fold-3*; (e), (f) correspond to *Fold-4* and (g), (h) are for *Fold-5*. (Here, BP represents back-propagation algorithm and Sols represent solutions). Best viewed in color. (a) Sols. with weight initialization using GA, (b) Sols. in (a) after training with BP algorithm, (c) Sols. with weight initialization using GA, (d) Sols. in (c) after training with BP algorithm, (e) Sols. with weight initialization using GA, (f) Sols. in (e) after training with BP algorithm, (g) Sols. with weight initialization using GA, (h) Sols. in (g) after training with BP algorithm.

Fig. 7. Misclassified UCF50 *WalkingWithDog* observation. (Frames of 'v_WalkingWithDog_g08_c02.avi' in UCF50 dataset [69]).

Table 9
Performance of CNN classifiers using back propagation algorithm (BPA), genetic algorithms (GA) and both for MNIST dataset.

Training approach	Performance (in %)
CNN classifier with *only GA* (i.e., initialized using GA)	12.58
CNN classifier *without GA* (i.e., trained using BPA)	91.0
CNN classifier *with GA* (using GA and BPA)	87.85

Table 10
Performance of the proposed classification framework using neural network (NN) and extreme learning machine (ELM) classifiers for MNIST dataset.

Classification methodology	Performance (in %)
Proposed framework using NN classifier	87.85
Proposed framework using ELM classifier	96.74
Proposed approach with ensemble of classifiers	97.9

Table 11
Performance comparison of the proposed approach with existing techniques on MNIST dataset.

Approach	Masks *count*, *size*	Layers	Accuracy (in %)
Convolutional net LeNet-5 [25]	22, 5×5	7	99.0
Proposed approach	3, 5×5	3	**97.9**

classifiers is given in Table 10. The performance of ensemble of CNN classifiers using ELM classification is also shown in the last row of this table. The performance of the proposed and existing approaches for character recognition on MNIST dataset is given in Table 11. The table also shows the number of layers with trainable weights, the size and count of masks in the CNN architecture. From the table, it can be observed that the proposed approach uses less number of layers, masks and training epochs to achieve comparable performance with the existing approach. The performance can be further improved by considering deeper architectures with more number of masks. The next section analyzes the solutions explored by the proposed approach for MNIST dataset.

3.4. Analysis

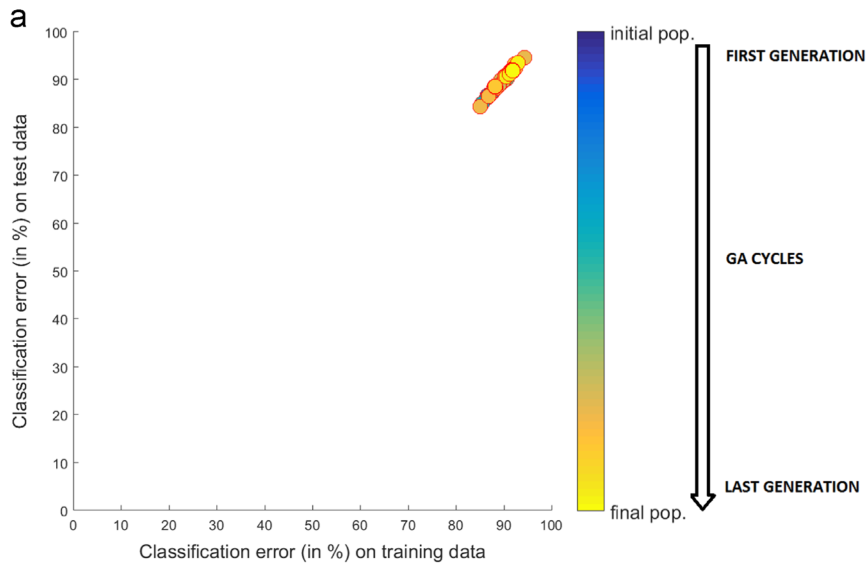
We visualize the performance of solutions explored by the proposed approach during the GA cycles, to validate the improvement of candidate solutions (GA population) over generations. The solutions explored by the proposed approach by initializing the weights using GA and training the generated classifiers using back-propagation algorithm for p_1 epochs for MNIST dataset are shown in Fig. 8(a) and (b), respectively. Each circle in these graphs correspond to a CNN classifier, with the error for training and testing data used as x and y coordinates of the circle and the time at which the solution is generated during the GA cycles determines the color of the circle. From Fig. 8(b), it can be observed that the classification error of the solutions initialized using GA and trained using back propagation algorithms decreases significantly with generations. The next section discusses the time complexity of this approach and its suitability for use in real-time applications.

3.5. Computational complexity

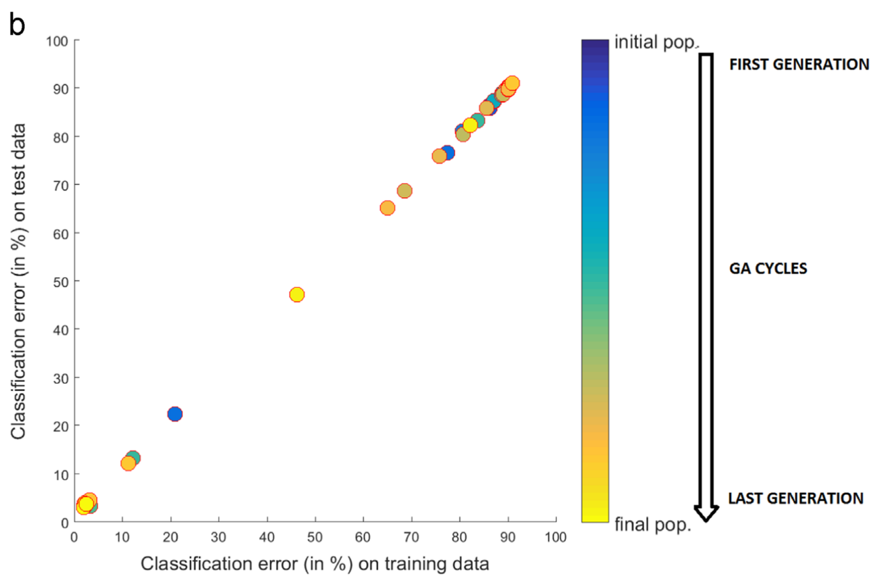
The existing approach uses genetic algorithms and training of convolutional neural network (CNN) classifier using back propagation algorithm, which can be parallelized. By parallel evaluation of candidate solutions (population) in genetic algorithms and use of efficient GPU-based CNN implementation (like cuDNN [78]) to train CNN classifiers for p_1 epochs results in a significant reduction in computation time. In this work, the CNN classifiers are trained for a small number of epochs (p_1) i.e., 50 epochs for UCF50 and 10 epochs for MNIST dataset. Several efficient multi-GPU implementations of CNN were proposed in the last few years like Berkeley's Caffe, Torch and Theano. Several browser-based user-friendly platforms like NVIDIA's DIGITS, Google's TensorFlow and Microsoft's Azure are proposed to aid the design and deployment of CNN classifiers for real-time applications. As inferencing is less expensive than training a deep neural network, trained CNN classifiers are used in many online systems like mobile applications for speech processing, image recognition, etc. Thus, the proposed approach generates a set of optimized CNN classifiers, which could then be deployed for real-time online application. The computational complexity of action back features restrict the feasibility to use this approach for real-time human action recognition. The next section gives the conclusions of this work.

4. Conclusion and future work

In this paper, we proposed a deep learning algorithm inspired by hybrid search approach of evolutionary and classical algorithms. As the performance of a neural network classifier (after training) depends on its weight initialization, we aim to optimize the initial weights using a GA framework. The proposed approach finds the weights of a convolutional neural network classifier that is neither overfit for training data nor stuck in a local minimum. The fusion across models identified using GA framework aims to overcome the limitations of individual models, by combining evidences across classifiers. Experimental studies on UCF50 dataset to recognize human actions from action bank features suggests that the proposed approach achieves a recognition accuracy of 99.98%. The future work will consider other spatio-temporal features like exmoves [79].



Solutions corresponding to weight initialization using GA chromosome



Solutions in (a) after training with back-propagation algorithm

Fig. 8. Solutions explored by the proposed approach for MNIST dataset: (a) After initialization using GA chromosomes, and (b) After training the classifiers using back-propagation algorithm for p_1 epochs. Best viewed in color. (a) Solutions corresponding to weight initialization using GA chromosome, and (b) Solutions in (a) after training with back-propagation algorithm.

Conflict of interest statement

None declared.

References

[1] Y. Bengio, P. Simard, P. Frasconi, Learning long-term dependencies with gradient descent is difficult, IEEE Trans. Neural Netw. 5 (2) (1994) 157–166.
[2] Y. Lecun, C. Cortes, The MNIST database of handwritten digits. URL ⟨http://yann.lecun.com/exdb/mnist/ ⟩.
[3] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M.S. Bernstein, A.C. Berg, L. Fei-Fei, Imagenet large scale visual recognition challenge, CoRR abs/1409.0575.
[4] S.E. Kahou, C. Pal, X. Bouthillier, P. Froumenty, c. Gülçehre, R. Memisevic, P. Vincent, A. Courville, Y. Bengio, R.C. Ferrari, M. Mirza, S. Jean, P.-L. Carrier, Y. Dauphin, N. Boulanger-Lewandowski, A. Aggarwal, J. Zumer, P. Lamblin, J.-P. Raymond, G. Desjardins, R. Pascanu, D. Warde-Farley, A. Torabi, A. Sharma, E. Bengio, M. Côté, K. R. Konda, Z. Wu, Combining modality specific deep neural networks for emotion recognition in video, in: Proceedings of the 15th ACM International Conference on Multimodal Interaction, ICMI '13, ACM, New York, NY, USA, 2013, pp. 543–550. http://dx.doi.org/10.1145/2522848.2531745.
[5] J. Aggarwal, M. Ryoo, Human activity analysis: a review, ACM Comput. Surv. 43 (3) (2011) 1–43, http://dx.doi.org/10.1145/1922649.1922653.
[6] D. Weinland, R. Ronfard, E. Boyer, A survey of vision-based methods for action representation, segmentation and recognition, Comput. Vis. Image Underst. 115 (2) (2011) 224–241, http://dx.doi.org/10.1016/j.cviu.2010.10.002.
[7] M.B. Holte, C. Tran, M.M. Trivedi, T.B. Moeslund, Human action recognition using multiple views: a comparative perspective on recent developments, in: Proceedings of the 2011 Joint ACM Workshop on Human Gesture and Behavior Understanding, J-HGBU '11, ACM, New York, NY, USA, 2011, pp. 47–52. http://dx.doi.org/10.1145/2072572.2072588.
[8] J.M. Chaquet, E.J. Carmona, A. Fernández-Caballero, A survey of video datasets for human action and activity recognition, Comput. Vis. Image Underst. 117 (6) (2013) 633–659.
[9] P. Foggia, G. Percannella, A. Saggese, M. Vento, Recognizing human actions by a bag of visual words, in: Proceedings of the 2013 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2013, pp. 2910–2915. http://dx.doi.org/10.1109/SMC.2013.496.

[10] I. Laptev, M. Marszalek, C. Schmid, B. Rozenfeld, Learning realistic human actions from movies, In: Proceedings of the 2008 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2008, pp. 1–8.
[11] N. Dalal, B. Triggs, C. Schmid, Human detection using oriented histograms of flow and appearance, In: Proceedings of the 9th European Conference on Computer Vision - Volume Part II, ECCV 06, Springer-Verlag, Berlin, Heidelberg, 2006, pp. 428–441.
[12] S. Sadanand, J. J. Corso, Action bank: A high-level representation of activity in video, in: Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012, pp. 1234–1241.
[13] H. Wang, A. Klaser, C. Schmid, C.-L. Liu, Action recognition by dense trajectories, in: Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2011, pp. 3169–3176.
[14] X. Liang, L. Lin, L. Cao, Learning latent spatio-temporal compositional model for human action recognition, In: ACM International Conference on Multimedia (ACM MM), 2013, pp. 263–272.
[15] Y. Bengio, A. Courville, P. Vincent, Representation learning: a review and new perspectives, IEEE Trans. Pattern Anal. Mach. Intell. 35 (8) (2013) 1798–1828, http://dx.doi.org/10.1109/TPAMI.2013.50.
[16] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, In: Advances in Neural Information Processing Systems (NIPS 2012), 2012, pp. 1097–1105.
[17] R.B. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, CoRR abs/1311.2524.
[18] S. Lawrence, C. Giles, A.C. Tsoi, A. Back, Face recognition: a convolutional neural-network approach, IEEE Trans. Neural Netw. 8 (1) (1997) 98–113, http://dx.doi.org/10.1109/72.554195.
[19] M. Matsugu, K. Mori, Y. Mitari, Y. Kaneda, Subject independent facial expression recognition with robust face detection using a convolutional neural network, Neural Netw. 16 (5–6) (2003) 555–559.
[20] P. Le Callet, C. Viard-Gaudin, D. Barba, A convolutional neural network approach for objective video quality assessment, IEEE Trans. Neural Netw. 17 (5) (2006) 1316–1327, http://dx.doi.org/10.1109/TNN.2006.879766.
[21] S. Ji, W. Xu, M. Yang, K. Yu, 3d convolutional neural networks for human action recognition, IEEE Trans. Pattern Anal. Mach. Intell. (PAMI) 35 (1) (2013) 221–231.
[22] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, L. Fei-Fei, Large-scale video classification with convolutional neural networks, In: Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014, pp. 1725–1732. http://dx.doi.org/10.1109/CVPR.2014.223.
[23] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, M. Paluri, C3D: generic features for video analysis, CoRR abs/1412.0767.
[24] Y. LeCun, K. Kavukcuoglu, C. Farabet, Convolutional networks and applications in vision, in: Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS), 2010, pp. 253–256. http://dx.doi.org/10.1109/ISCAS.2010.5537907.
[25] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[26] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, A. Baskurt, Sequential deep learning for human action recognition, In: Proceedings of the Second International Conference on Human Behavior Unterstanding, HBU'11, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 29–39.
[27] K. Wang, X. Wang, L. Lin, M. Wang, W. Zuo, 3d human activity recognition with reconfigurable convolutional neural networks, in: Proceedings of the ACM International Conference on Multimedia, MM '14, ACM, New York, NY, USA, 2014, pp. 97–106. http://dx.doi.org/10.1145/2647868.2654912.
[28] L. Lin, K. Wang, W. Zuo, M. Wang, J. Luo, L. Zhang, A deep structured model with radius-margin bound for 3d human activity recognition, Int. J. Comput. Vis., 2015, pp. 1–18. http://dx.doi.org/10.1007/s11263-015-0876-z.
[29] G. Chéron, I. Laptev, C. Schmid, P-CNN: pose-based CNN features for action recognition, CoRR abs/1506.03607 URL arxiv.org/abs/1506.03607.
[30] V. Veeriah, N. Zhuang, G. Qi, Differential recurrent neural networks for action recognition, CoRR abs/1504.06678. URL arxiv.org/abs/1504.06678.
[31] K. Simonyan, A. Zisserman, Two-stream convolutional networks for action recognition in videos, CoRR abs/1406.2199. URL arxiv.org/abs/1406.2199.
[32] Y. Bengio, O. Delalleau, On the expressive power of deep architectures, in: Proceedings of the 22nd International Conference on Algorithmic Learning Theory, ALT'11, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 18–36.
[33] Y. Bengio, Learning deep architectures for ai, Found. Trends Mach. Learn. 2 (1) (2009) 1–127, http://dx.doi.org/10.1561/2200000006.
[34] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, U.D. Montral, M. Québec, Greedy layer-wise training of deep networks, In: In NIPS, MIT Press, 2007.
[35] G.E. Hinton, S. Osindero, Y.-W. Teh, A fast learning algorithm for deep belief nets, Neural Comput. 18 (7) (2006) 1527–1554.
[36] H. Larochelle, Y. Bengio, J. Louradour, P. Lamblin, Exploring strategies for training deep neural networks, J. Mach. Learn. Res. 10 (2009) 1–40.
[37] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10). Society for Artificial Intelligence and Statistics, 2010.
[38] L. Prechelt, Early stopping – but when? In: Neural Networks: Tricks of the Trade, Lecture Notes in Computer Science, vol. 1524, Springer-Verlag, 1997, pp. 55–69, Chapter 2.
[39] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.
[40] G.E. Dahl, T.N. Sainath, G.E. Hinton, Improving deep neural networks for lvcsr using rectified linear units and dropout, in: Proceedings of the 2013 International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2013, pp. 8609–8613.
[41] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, S. Bengio, Why does unsupervised pre-training help deep learning? J. Mach. Learn. Res. 11 (2010) 625–660.
[42] I. Sutskever, J. Martens, G.E. Dahl, G.E. Hinton, On the importance of initialization and momentum in deep learning, In: Proceedings of the 30th International Conference on Machine Learning (ICML-13), JMLR Proceedings, vol. 28, JMLR.org, 2013, pp. 1139–1147.
[43] J. Bergstra, Y. Bengio, Random search for hyper-parameter optimization, J. Mach. Learn. Res. 13 (2012) 281–305.
[44] E.I. Chang, R. Lippmann, Using genetic algorithms to improve pattern classification performance, in: R. Lippmann, J.E. Moody, D.S. Touretzky (Eds.), Proceedings of Advances in Neural Information Processing Systems (NIPS), November 26–29, Morgan Kaufmann, Denver, Colorado, USA, 1990, pp. 797–803.
[45] D. Decker, J. Hintz, A genetic algorithm and neural network hybrid classification scheme, in: Proceedings of 9th AIAA Computers in Aerospace Conference, AIAA, 1993, pp. 473–475.
[46] S.A. Harp, T. Samad, A. Guha, Towards the genetic synthesis of neural network, in: Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1989, pp. 360–369.
[47] J. Schaffer, R.A. Caruana, L.J. Eshelman, Using genetic search to exploit the emergent behavior of neural networks, Phys. D: Nonlinear Phenom. 42 (1–3) (1990) 244–248 http://dx.doi.org/10.1016/0167-2789(90)90078-4.
[48] D.J. Montana, L. Davis, Training feedforward neural networks using genetic algorithms, in: Proceedings of the 11th International Joint Conference on Artificial Intelligence (IJCAI'89), vol. 1, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1989, pp. 762–767.
[49] S. Ding, H. Li, C. Su, J. Yu, F. Jin, Evolutionary artificial neural networks: a review, Artif. Intell. Rev. 39 (3) (2013) 251–260, http://dx.doi.org/10.1007/s10462-011-9270-6.
[50] J.R. Koza, J.P. Rice, Genetic generation of both the weights and architecture for a neural network, in: International Joint Conference on Neural Networks (IJCNN-91), vol. ii, 1991, pp. 397–404 vol.2. http://dx.doi.org/10.1109/IJCNN.1991.155366.
[51] F. Gruau, Genetic synthesis of boolean neural networks with a cell rewriting developmental process, in: International Workshop on Combinations of Genetic Algorithms and Neural Networks (COGANN-92), 1992, pp. 55–74. http://dx.doi.org/10.1109/COGANN.1992.273948.
[52] R.J. Collins, D.R. Jefferson, An artificial neural network representation for artificial organisms, in: Parallel Problem Solving from Nature, Springer-Verlag, 1990, pp. 259–263.
[53] S. Bornholdt, D. Graudenz, General asymmetric neural networks and structure design by genetic algorithms, Neural Netw. 5 (2) (1992) 327–334 http://dx.doi.org/10.1016/S0893-6080(05)80030-9.
[54] J. Schaffer, D. Whitley, L. Eshelman, Combinations of genetic algorithms and neural networks: a survey of the state of the art, in: International Workshop on Combinations of Genetic Algorithms and Neural Networks (COGANN-92), 1992, pp. 1–37. http://dx.doi.org/10.1109/COGANN.1992.273950.
[55] X. Yao, Evolving artificial neural networks, Proc. IEEE 87 (9) (1999) 1423–1447, http://dx.doi.org/10.1109/5.784219.
[56] X. Yao, A review of evolutionary artificial neural networks, Int. J. Intell. Syst. 4 (1993) 539–567.
[57] O.E. David, I. Greental, Genetic algorithms for evolving deep neural networks, in: Proceedings of the 2014 Conference Companion on Genetic and Evolutionary Computation Companion, GECCO Comp '14, ACM, New York, NY, USA, 2014, pp. 1451–1452. http://dx.doi.org/10.1145/2598394.2602287.
[58] R. Oullette, M. Browne, K. Hirasawa, Genetic algorithm optimization of a convolutional neural network for autonomous crack detection, in: Congress on Evolutionary Computation (CEC2004), vol. 1, 2004, pp. 516–521.
[59] L.-O. Fedorovici, R.-E. Precup, F. Dragan, C. Purcaru, Evolutionary optimization-based training of convolutional neural networks for ocr applications, in: 17th International Conference on System Theory, Control and Computing (ICSTCC), 2013, pp. 207–212.
[60] E. Rashedi, H. Nezamabadi-pour, S. Saryazdi, Gsa: a gravitational search algorithm, Inf. Sci. 179 (13) (2009) 2232–2248.
[61] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: IEEE International Conference on Neural Networks, vol. 4, Perth, Australia, IEEE Service Center, Piscataway, NJ, 1995, pp. 1942–1948.
[62] J. Koutník, J. Schmidhuber, F. Gomez, Lecture Notes in Computer Science, in: A. del Pobil, E. Chinellato, E. Martinez-Martin, J. Hallam, E. Cervera, A. Morales (Eds.), From Animals to Animats 13, Lecture Notes in Computer Science, vol. 8575, Springer International Publishing, 2014, pp. 260–269.
[63] F. Gomez, J. Schmidhuber, R. Miikkulainen, Accelerated neural evolution through cooperatively coevolved synapses, J. Mach. Learn. Res. 9 (2009) 937–965.
[64] J. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, MI, USA, 1975. URL 〈http://books.google.com/books?id=YE5RAAAAMAAJ〉.
[65] J. Bascom, Darwin's theory of the origin of species, Am. Theol. Rev. 3 (1871) 349–379.
[66] D.E. Goldberg, Genetic algorithms, Pearson Education, India, 2006.

[67] Z.-H. Zhou, Ensemble Methods: Foundations and Algorithms, 1st Ed., Chapman & Hall/CRC, 2012.
[68] R.B. Palm, Prediction as a candidate for learning deep hierarchical models of data (Master's thesis). Technical University of Denmark, Asmussens Alle, Denmark, 2012.
[69] K.K. Reddy, M. Shah, Recognizing 50 human action categories of web videos, Mach. Vis. Appl. 24 (5) (2012) 971–981, http://dx.doi.org/10.1007/s00138-012-0450-4.
[70] Action bank: a high-level representation of activity in video, URL ⟨http://www.cse.buffalo.edu/~jcorso/r/actionbank/⟩, accessed on: 2015-08-08.
[71] G.-B. Huang, Q.-Y. Zhu, C.K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (1–3) (2006) 489–501.
[72] O. Kliper-Gross, Y. Gurovich, T. Hassner, L. Wolf, Motion interchange patterns for action recognition in unconstrained videos, In: Proceedings of the 12th European Conference on Computer Vision(ECCV) – volume Part VI, ECCV'12, Springer-Verlag, Berlin, Heidelberg, 2012, pp. 256–269.
[73] F. Shi, E. Petriu, R. Laganiere, Sampling strategies for real-time action recognition, In: Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013, pp. 2595–2602.
[74] L. Wang, Y. Qiao, X. Tang, Motionlets: Mid-level 3d parts for human motion recognition, in: Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013, pp. 2674–2681.
[75] Q. Zhou, G. Wang, K. Jia, Q. Zhao, Learning to share latent tasks for action recognition, in: Proceedings of the 2013 IEEE International Conference on Computer Vision (ICCV), 2013, pp. 2264 –2271.
[76] E.P. Ijjina, C. Mohan, Human action recognition based on recognition of linear patterns in action bank features using convolutional neural networks, In: Proceedings of the 13th International Conference on Machine Learning and Applications (ICMLA), 2014, pp. 178–182. http://dx.doi.org/10.1109/ICMLA.2014.33.
[77] N. Ballas, Y. Yang, Z.-Z. Lan, B. Delezoide, F. Preteux, A. Hauptmann, Space-time robust representation for action recognition, In: The IEEE International Conference on Computer Vision (ICCV), 2013.
[78] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, E. Shelhamer, cudnn: Efficient primitives for deep learning, CoRR abs/1410.0759. URL arxiv.org/abs/1410.0759.
[79] D. Tran, L. Torresani, EXMOVES: classifier-based features for scalable action recognition, CoRR abs/1312.5785.

Earnest Paul Ijjina received his Masters in Computer Science and Engineering from Indian Institute of Technology Kharagpur in 2007 and is currently pursuing his Ph.D in Computer Science and Engineering at Indian Institute of Technology, Hyderabad. Previously, he was an IT manager for four years at Morgan Stanley and an assistant professor for one year. His research interests include computer vision, video content analysis and deep learning.

Dr. C. Krishan Mohan received the Ph.D. Degree in Computer Science and Engineering from Indian Institute of Technology, Madras, India in 2007. He received the Master of Technology in system analysis and computer applications from National Institute of Technology, Surathkal, India in 2000. He received the Master of Computer Applications degree from S.J. College of Engineering, Mysore, India in 1991 and the Bachelor of Science Education (B.Sc. Ed.) degree from Regional Institute of Education in 1988. He is currently associate professor with the Department of Computer Science and Engineering, Indian Institute of Technology, Hyderabad, India. His research interests include video content analysis, pattern recognition, and neural networks.