INTERNATIONAL JOURNAL OF RESEARCH IN COMPUTER APPLICATIONS AND ROBOTICS
ISSN 2320-7345

A NEURO-GENETIC MODEL FOR MOVING OBJECT CLASSIFICATION IN VIDEO SURVEILLANCE SYSTEMS

[1]Akintola K.G., [2]Olabode O.
Computer Science Department, Federal University of Technology Akure Ondo State Nigeria

Abstract
Moving object classification in videos is a requirement in smart visual surveillance systems as it allows the system to know the kind of object in the scene and be able to recognize the actions the object can perform. Accurate classification of moving objects is highly required because false detection can lead to poor performance of the system. This paper presents a genetically trained neural-network machine learning approach for real time object classification in videos. This is necessary to boost the classification rate of Neural Networks. Moving objects are detected using Fast Kernel Density Estimation algorithm. Distance signal features are then extracted from the silhouettes of the detected objects. The distance signals features are then normalized and fed into a neuro-genetic model to obtain optimum weights. The optimum weights are saved to be used later by a multilayer feed-forward neural network to classify the object as human or vehicle. We compared the model with a neural network trained using back-propagation algorithm on a set of objects detected from real life videos. The neuro-genetic model outperforms with recognition rate of 99.09 while the back-propagation neural network achieves the recognition rate of 98.5% .

Keywords: algorithm Kernel Density, Neural Network, Normalized, Genetic

1.0 Introduction

Object classification in videos is an important requirement in surveillance systems as it aids understanding of the intentions or actions that the object can perform. For instance, human beings can sit, walk, run, fall while cars can move, run, over-speed, crash etc. Object classification is a challenging task because of various object poses, illumination, occlusion, etc. Our desire to carry out this research is to select the best model for classification which will be used in higher layer of surveillance system for activity recognition. In such systems a higher degree of recognition accuracy and higher response time is highly required. We proposed to use parallel- based kernel density estimation (Akintola and Tavakollie, 2011) for an adaptive background subtraction which has been applied to both outdoor and indoor environments and was found very fast and robust. Unlike the previous systems in the literature that uses motion information, our system is based on shape information. It has been recognized by Zhao and Thorpe (2000) that the shape information from silhouettes extracted from the segmented region is invariant to color and texture changes. Moreover, computing the distance signals is very fast and demands small storage space which are good qualities for a surveillance system which requires fast operation and storage minimization.
After object segmentation, the feature vectors are gotten by calculating the distance from the centroid of the object to the contour outline starting at right hand side and moving in anticlockwise direction. The feature vector is then normalized by dividing it the sum of the lengths. These vectors are then used to train two neural network models

using back-propagation training algorithm and neuro-genetic training algorithm.

2.0 Related works.

Many researches have been carried out in literature on object classification using neural networks (Modi and Mehta, 2011, zhao etal., 2000, Teschioni etal., Khashman etal., 2008, Karital etal., 2012, Collins etal, 2000).
Modi et al., (2011) presents Neural Network approach for recognition of human motion using stationary camera. It is noted that the task to classify and identify objects in the video is difficult for human operator. Object is detected using background subtraction technique. The detected moving object is divided into 8x8 non-overlapping blocks. The mean of each of the blocks is calculated. All mean value is then accumulated to form a feature vector. A neural network is trained using the generated feature vectors. Experiment performed shows a good recognition rate, however the algorithm can still be improved upon.

Collins etal.(2000) classifies moving object blobs into general classes such as "humans" and "vehicles "using viewpoint-specific neural networks, trained for each camera. Each neural network is a standard three-layer network . Learning in the network is accomplished using the back propagation algorithm. Input features to the network are a mixture of image-based and scene based object parameters: image blob dispersedness (perimeter2/area (pixels)); image blob area (pixels); apparent aspect ratio of the blob bounding box; and camera zoom. There are three output classes: human; vehicle; and human group. This approach fails to discriminate object with similar dispersedness. Zhao etal., (2000) presents stereo-and neural network approach for pedestrian detection in videos. The motivation for the project is to develop a surveillance system that can avoid dangerous situations. This is achieved using a stereo-based segmentation and neural network based recognition. This system performs well in pedestrian detection especially stationary pedestrians but fails to incorporate motion cues into the system which should have enhanced the performance.

Motion method uses the object's motion characteristics to distinguish the object. Zhou and Aggrawal (2006) use the variance of compactness of the object to classify target object as single person, people group, or vehicle. It is noted in the paper that vehicles are more consistent in their motion because they are rigid objects wheras humans shift some parts forward to maintain balance. So the variance of motion direction is employed to measure motion consistency. These features are got from the optical flow of the motion. The computation of optical flow can be computationally expensive.

All these research works use the back-propagation as their training algorithm. Recently genetic algorithm is being used to train a neural network for weight optimization. These works can be improved upon using the hybrid of Genetic-Neural system.

3.0 System Description

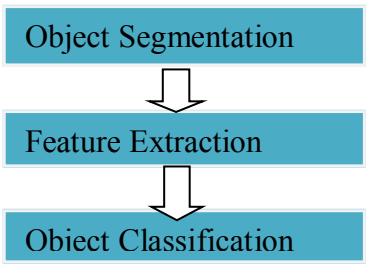

Figure 1.System Block Diagram

3.1 Object Segmentation

Kernel density estimation (KDE) is the most used and studied nonparametric density estimation method. The model is the reference dataset, containing the reference points indexed natural numbered. In addition, assume a local kernel function centered upon each reference point, and its scale parameter (the bandwidth). The common choices for kernels include the Gaussian: and the Epanechnikov kernel (Elgammal etal., 1991)

The Gaussian Kernel is given by:

$$K_N = (2\pi)^{-\frac{d}{2}} \exp\left(-\tfrac{1}{2}\|x\|^2\right)$$

(1)

The Epanechnikov kernel is given by:

$$K_E = \begin{cases} -\frac{1}{2}c_d^{-1}(d+2)(1-||x||^2 & if\ ||x|| < 1 \\ 0 & otherwise \end{cases} \quad (2)$$

Let $x_1, x_2, \ldots\ldots x_{n,}$ be a random sample taken from a continuous, univariate density *f*. The kernel density estimator is given by,

$$\hat{f}(x;h) = \frac{1}{nh}\sum_{i=1}^{n} k\left(\frac{x-x_i}{h}\right) \quad (3)$$

K is the function satisfying $\int k(x)dx = 1$ which is refered to as the Kernel, h is a positive number, usually called the bandwidth or window width.

Kernel Density Estimation (KDE) for background modeling involves using a number of frames (training frames) to build the probability density of each pixel location. A histogram of each pixel location of the image is then constructed. After the histogram is built, we find the adaptive threshold of each pixel.

For every pixel observation, classification involves determining if it belongs to the background or the foreground. The first few initial frames in the video sequence (called learning frames) are used to build histogram of distributions of the pixel color. No classification is done for these learning frames. Classification is done for the subsequent frames using the process given below. Typically, in a video sequence involving moving objects, at a particular spatial pixel position a majority of the pixel observations would correspond to the background. Therefore, background clusters would typically account for much more observations than the foreground clusters. This means that the probability of any background pixel would be higher than that of a foreground pixel. The pixels are ordered based on their corresponding value of the histogram bin based on the adaptive threshold in the previous stage. The pixel intensity values for the subsequent frames are estimated. The corresponding histogram bin is evaluated and the bin value corresponding to this intensity is determined.

If the value is < threshold

Classify as foreground

Else

Classify as background.

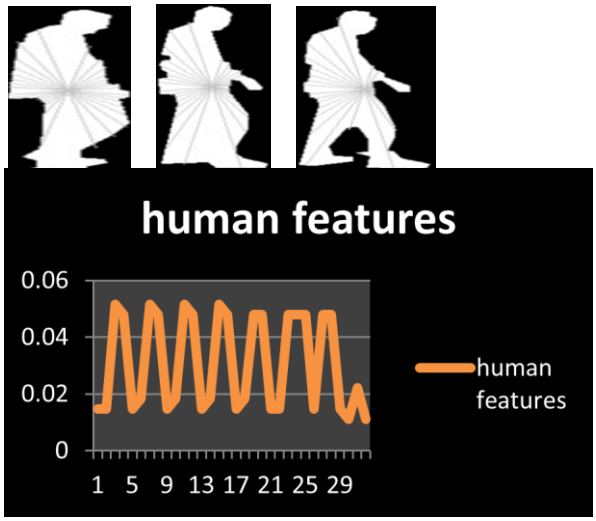

Figure 2a A typiocal 1D feature vector generated from pedestrian silhouete

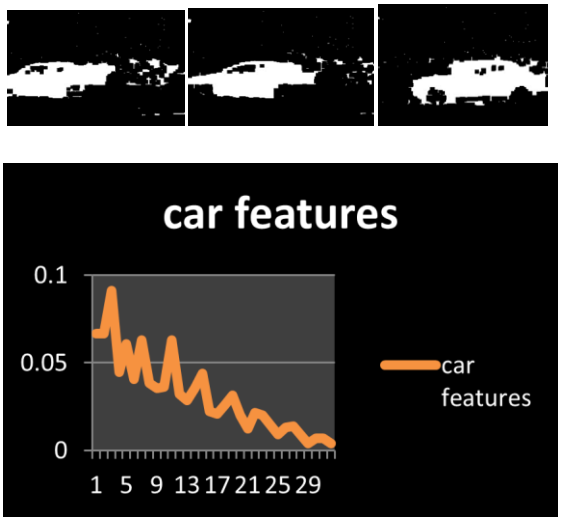

Figure 2b A typical 1D feature vector generated from vehicle silhouete

3.2 Feature Extraction.

After extracting the contour, the centroid of the contour is calculated using equation (4). From that centre a pre-defined number of axes are projected outwards at specified regular angles to the nearest edges of the contour in an anti- clockwise direction see Figure 4.

$$(c_x, c_y) = \frac{1}{n}\left(\sum_{t=1}^{n} xt, \sum_{t=1}^{n} xt\right). \ldots \quad (4)$$

The distance from the contour's centre to its nearest edge along that a predefined angle is then stored. This is done for all the other angles to for a set of values called a vector. The dimension of the vector equals to the number of axes being projected from the centre. The vector is then normalized. This ensures that the vector is scale invariant as the largest value in the vector will at this point be 1.0, which will be the longest of the axes projected.

Let S be the segmented object region within the frame. Let line i be a line projected at from the centroid to the object boundary at angle Θ to the horizontal line passing through the centroid of the object. The length of each line to the contour boundary of the object is given by

$$l_i = \sum_{k=c}^{w} \delta(p(k,l)) \quad (5)$$

Where k and l are the co-ordinates in the x and y directions respectively.

L is given by ktan(θ)

$\delta()$ is a binary function that returns 0 or 1

$$\delta(p(k,l)) = \begin{cases} 1 & if\ p(k,l)\ \epsilon\ S \\ 0 & otherwise \end{cases} \quad (6)$$

p(k,l) =is the pixel value of the object

The number of lines in each image containing an object as well as the number of neurons in the input layer is j where j ={1,2,3,….,n) and n is given by N=360/ θmin where θmin is the smallest of the angles. Angle size of 10 degrees interval have been used in this research e.g (10, 20, 30,…., 360) will divide the object into 36 regions. Out of these 36 lines we only used 32 of them as feature vectors.

3.3 Object Classification

3.3.1 Artificial Neural Network

Artificial neural network (ANN) is a widely used pattern-recognition methodology for machine learning. ANN is an emulation of biological neural network, which is composed of many interconnected neurons. However, it only utilized a very limited set of concepts from its biological counterpart. An ANN could have one or more layer of neurons as shown in Figure 4. They could be fully or partially connected. Each connection between two nodes has a weight, which encapsulate the "knowledge" of the system. By processing existing cases with inputs and expected outputs, these weights would be adjusted based on differences between actual and expected outputs. Because of the nonlinear fashion of ANN, they could be used in a lot of engineering applications.

The general ANN learning process has three continuous steps:

a. Compute temporary outputs
b. Compare outputs with desired targets
c. Adjust the weights and repeat the process

We develop a neural network model to shown in Figure 4. The architecture of the neural network consists of 32 input nodes, one hidden layer with 32 neurons and one output layer. The shape features extracted in section 3.2 were used as input to the network.

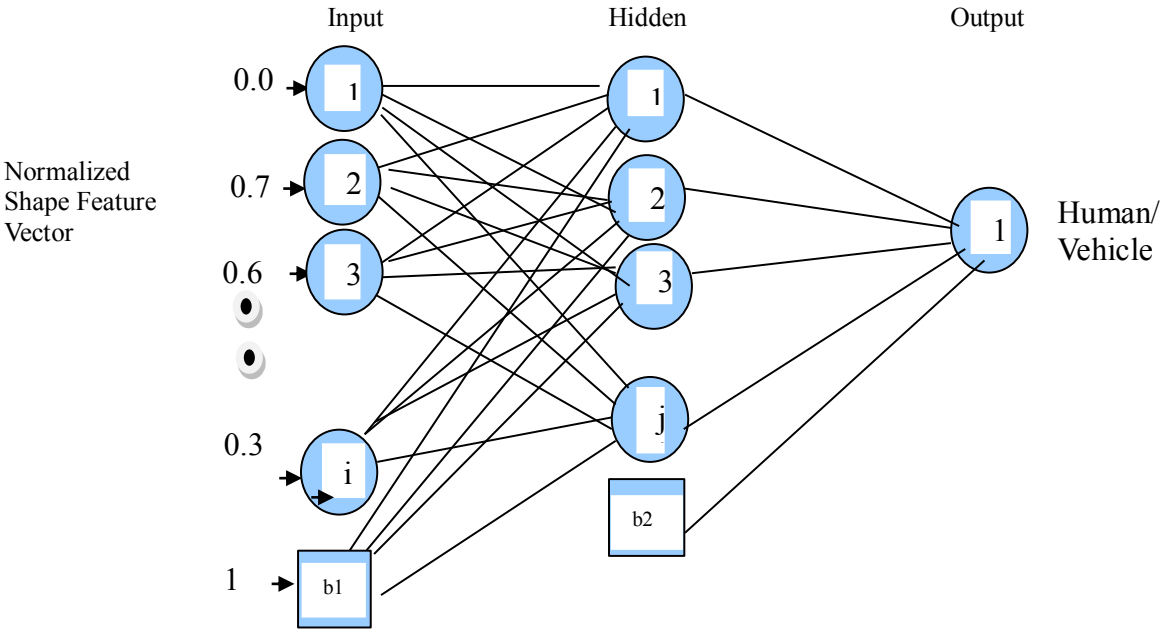

Figure 4 The Neural Network model for the pedestrian/car detection.

The network is then trained using back-propagation and Genetic Algorithm as follow.

3.3.1.1 The Backpropagation Training Algorithm

The algorithm for back-propagation is as follow:

1. The architecture of the network is first determined. This involves the following:
    a. How many input and output neurons are needed
    b. How many Hidden neurons and layers are required
2. Initialize all weights and biases to small random values.
3. Repeat until termination criterion satisfied:
    a. Present a training example into the input node and propagate the results forward until the output node is reached [forward Pass].
    b. Calculate the actual output
    c. Adapt the weights starting from the output layer and working backwards until first hidden layer [backward Pass]. This can be done by the following:
        a. Output Neuron: $\delta_i^p = (d_k^p - o_k^p).\, o_k^p.(1 - o_k^p)$
        b. $w_{xy}{}^{(t+1)} = w_{xy}{}^{(t)} + \Delta w_{xy}$ Where $w_{xy}{}^{(t)}$ is the weight from node x to node y at time t
        c. $\Delta w_{xy} = \eta.\delta_x.o_y$ is the weight change
        d. $\delta_i = (d_i - o_i).\, o_i.(1 - o_i)$ for output neuron
        e. $\delta_j = o_j.(1 - o_j).\, \sum_i w_{ji}.o_l^p$
        f. Hidden Neuron: $\delta_j{}^p = o_k^p.(1 - o_k^p).\, w_{ji}.\delta_i$ for hidden neuron j (the sum is over the i
        g. nodes in the layer above the node i)

The stopping criterion is checked at the end of each epoch. These stopping conditions may be any of the following:

a. The error (mean absolute or mean square) at the end of an epoch is below a threshold.
    i. All training examples are propagated and the mean (absolute or square) error is calculated.
    ii. The threshold is determined heuristically e.g. 0.3
b. Maximum number of epoch is reached
c. Early stopping using a validation set (TTS)

If any of the conditions are fulfilled, then training stops, otherwise we repeat the training in the next epoch. It typically takes hundreds or thousands of epochs for a neural network to converge.

3.3.1.2 The Neuro-Genetic Training Algorithm

A Neuro-genetic training algorithm is also adopted. The schematic diagram of the model is shown in Figure 3

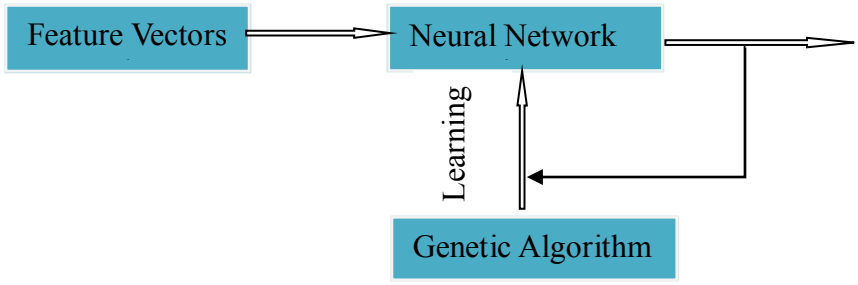

Figure 3 Neuro-Gentic Training model

In using Genetic Algorithm to solving this problem, the following three phases are used:

a. Initialization
b. Evaluation
c. Application of Genetic operators

a. Initialization. Genetic algorithms operate on a set of strings. This set of strings is known as a population and is through the process of evolution to produce new set of strings. To start with, the initial population could be made up of chromosomes chosen at random or based on heuristically selected strings. The initial population is a wide varieties of structures. Population size affects the efficiency of the performance of a GA.

The first step in GA implementation is the determination of a genetic encoding scheme, that is, to denote each possible point in the problem's search space as a characteristic string of defined length. This is in order to be sure that GA will not only optimize network configuration but, in the meantime, genetic training will proceed on weight values. Figure 5.0 shows a typical chromosome generated from the network above. The genes are gotten from the weights at each of the nodes in the network. These genes are concatenated to form a chromosome. Several of these chromosomes are randomly generated from the network architecture to form the initial population of the solution space.

01011100000	1001010101	1111000010	100011110	10101010101	1111110000	.....	10101010101

Figure 5 Sample Chromosome Encoding of ANN using strings.

In this research work, weight values between each layer of the multi-layer feed-forward neural network are simultaneously coded as one gene. Each of the gene which represents the weights of the neural network is coded using randomized binary numbers of length 10. The minimum value is zero while the maximum value is one. The number of bits to represent the value of a gene (weight) is calculated as

$$l_i \geq log_2(\frac{x_i^{max} - x_i^{max}}{\Delta x} + 1)$$

where l_i is the string length, x_i^{min} is the minimum value of the gene, x_i^{max} is the maximum value of the gene and Δx is the error tolerance.

A chromosome represents one neural network structure and weight value of the neuron network is mapped to this very chromosome. The length of chromosome is obtained by concatenating the bits representing each gene.

b. **Definition of Evaluation Function.** For this, the suitability of the solutions to the problem is determined. When GA is applied to solve a problem, the definition of the evaluation function to evaluate the problem-solving ability of a chromosome is important. Since Neural Network works by error minimization, therefore the objective function adopted in this work is the m*ean-square error* (*MSE*) between the Neural Network output and the desired output. The reciprocal of the MSE is then used as the evaluation function. Its computational formula is as follows:
First, the initial strings are randomly generated. The generated string is converted into real number by (7) and 8

$$w_n = min + \frac{max-min}{2^{l-1}} y_n \quad (7)$$

$$yn = \sum_{j=0}^{l-1} 2^j bj \quad (8)$$

where l_i is the string length $w_n$is the real weight, min is the minimum value of the gene, in this case 0 is used, max is the maximum value of the gene, in this case 1 is used and yn is the equivalent decimal value of the gene given and bj is the bit value.

These decoded values are then used to train the neural network using back-propagation. The error of desired output – computed output is calculated using Mean Squre Error (9).

$$MSE = \frac{1}{pm} \sum_{j=1}^{p} \sum_{j=1}^{m} (d_{pj} - y_{pj})^2 \quad (9)$$

where; m=no of output nodes; p=no of trained samples; d_{pj}=expected output of network; y_{pj}=actual output of network. The fitness function used is the reciprocal of the mean square error as given by (4.11).

$$f(MSE) = \frac{1}{MSE} \quad (10)$$

c. **Application of Genetic operators:** Three GA operators adopted in this work are:

i. **Selection.** The fitness of the new offspring is calculated and are sorted in the descending order. So chromosomes of highest fitness values are selected for the next generation.
In this research work, the roulette wheel method is adopted for selection. The probability of selection is given by 4.12.

$$P_i = \frac{f_i}{\sum_{i=1}^{N} f_i} = \frac{f_i}{f_{sum}} \quad (11)$$

in which; f_i=fitness value of individual i; f_{sum}=total fitness value of population; P_i=selective probability of individual. It's obvious that individuals with high flexibility values are more likely to be reproduced during the next generation.

ii. **Crossing.** In this research work, one-point crossing is adopted. The specific operation is to randomly set one crossing point among individual strings. When crossing is executed, partial configurations of the very point's anterior and posterior individuals are exchanged, and gave birth to two new offspring

iii. **Mutation.** As for two-value code strings, mutation operation is to reverse the gene values within a random number generated between zero and one.

Genetic control parameters dictate how the algorithm will behave. Changing these parameters can change the computational result. These parameters are population size, crossing probability , mutation probability and network termination condition. In this work population size N is 50, crossing probability P_c is 0.8, mutation probability P_m is 0.015, and network's terminative condition is MAXGEN of 100. This combination gave us an excellent empirical performance.

The BPN learns during a training epoch, you will probably go through several epochs before the network has sufficiently learnt to handle all the data you've provided it and the end result is satisfactory. A training epoch is described below: For each input entry in the training data set:

i. feed input entry data into the network (feed forward).
ii. Initialized weights
iii. check output against desired value and feedback error (back-propagate)
Where back-propagation consists of:
iv. calculate error gradients
v. update weights (in our case the weights between output and hidden layers hidden and hidden layers and input and hidden layers were updated in that order)

The network parameters for our system model consist of an input layer with 32 neurons, one hidden layer with 4 neurons, and an output layer with 1 neuron (MLP 32 : 4: 1) as shown above. The number of epochs used is 1000, with the momentum of 0.5 and learning rate of 0.3. The initial weights were randomly initialized to small random numbers less than 1 using random number generators.

For the Stopping Conditions there are various commonly used stopping conditions for neural networks, such include; desired accuracy , desired mean square error and elapsed epochs. In this project we used the 1000 elapsed epochs to train the Network.

4.0 Experiment and Result

ANN models' performances can be measured by the mean relative percentage error. It measures the accuracy of prediction through representing the degree of scatter. Eq.1 was utilized to calculate the relative error for each case in the testing set. Then, the calculated values were averaged and factored by 100 to express in percentages. (Actual)-(predicted)/actual *100%.

We trained the network using back propagation algorithm. 80 data items consisting of forty vehicles and 40 pedestrians were used to train the network. The class vehicle is assigned 1 while the class human is assigned 0. The network is then trained using back-propagation algorithm. The weights from the training are then saved. For testing, 343 data items were used to test the system. Out of this 110 were vehicle data while 223 were human data. We decided from the ROC curve to make the threshold for vehicles to be greater or equal to 0.9 while that of the human is set to less or equal to 0.1. Out of the 110 vehicles, only one was misclassified while for 223 human class data items, only four were misclassified. Error in forecast is 1.46% so the accuracy is 98.54%

We trained the same network using Genetic Algorithm using the same data set as above for training and testing. Out of the 110 vehicles, only none was misclassified while for 223 human class data items, only three were misclassified. Error in forecast is 0.009% so the accuracy is 99.01%

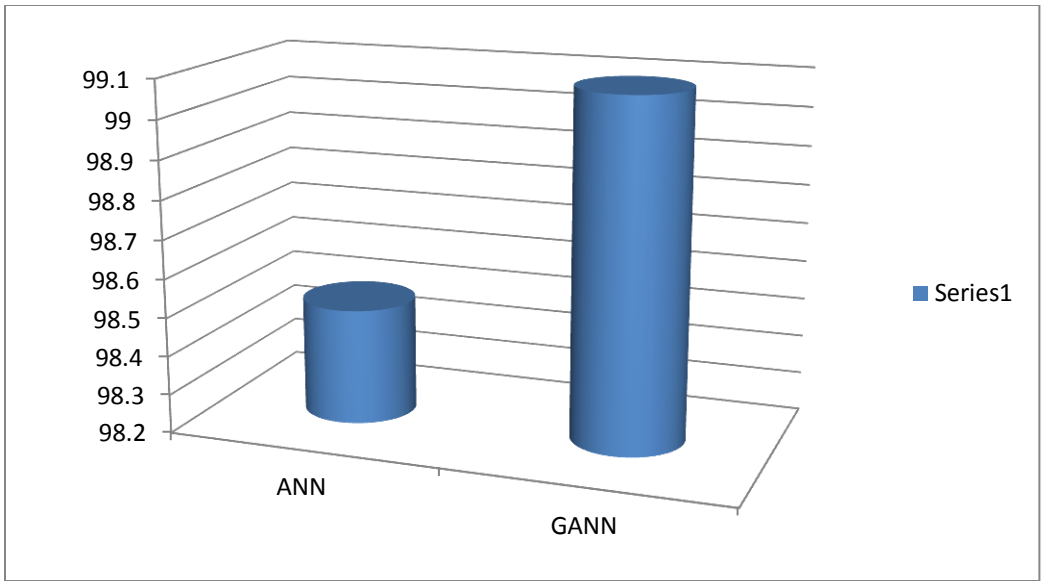

Figure 5 ANN and GANN Recognition Performance

5.0 Conclusion

In this paper, we focused on moving object classification in videos. The case is a binary classification between humans and vehicles. Training algorithms based on back-propagation and neuro-genetic algorithm are use to train a neural network. The performances of the two algorithms are contrasted. The result shows that genetically trained neural network outperform the back-propagation training algorithm. Future work can be done on this study by using different classifiers such as support vector machines.

6.0 REFERENCES

[1] Akintola K.G , Tavakollie A., (2011) , Robust Foreground detection in Videos using Adaptive Colour Histogram Thresholding and shadow removal 7th International Symposium on Visual Computing Las-Vegas, NV, September, 2011.

[2] Collins, R. T., Lipton, A. J., Fujiyoshi, H., & Kanade, T.(2000). A system for video surveillance and monitoring: VSAM final report. Technical report CMU-RI-TR-00-12, Robotics Institute, Carnegie Mellon University, May 2000.No. 0802-mds2008020001.

[3] Elgammal A., Duraiswami R, Harwood D., and Davis L.S (2002) Background and Foreground Modeling Using Nonparametric Kernel Density Estimation for Visual Surveillance, PROCEEDINGS OF THE IEEE, VOL. 90, NO. 7, JULY 2002.

[4] Kavita P. Patil S.V. (2012) Tracking and counting human in visual surveillance systems, international journal of electronics and communication engineering and technology (IJECET) vol 3 issues 3 pp 139-146 www.iame.com/ijecet.asp

[5] Modi R.V and Mehta T.B. (2011) Neural network based approach for recognition of human motion using stationary camera, International journal of computer applications (975-887) vol 25 –no 6 -July 2011

[6] Teschioni A., Oberti F., and Regazzoni C., A neural network approach for moving objects recognition in color image sequences for surveillance applications

[7] Khashman A. (2008) Automatic detection, extraction and recognition of moving objects international journal of systems applications, engineering and development issue 1, vol, 2 2008

[8] Zhao L., Thorpe C.E., (2000) Stereo and neural network based pedestrian detection IEE transactions on intelligent transportation systems vol. 1No. 3 September 2000

[9] Zhou Q., Aggrawal J.K (2006) Object tracking in an outdoor environment using fusion of features and cameras, Journal of Image and Vision Computing 24 (2006) 1244-1255