Driving Signature Analysis for Auto-Theft Recovery

Adrian Bosire
Computer Science Department,
Kiriri Womens University of Science and Technology,
Kenya
bosire.adrian@kwust.ac.ke

Damian Maingi
Department of Mathematics,
Sultan Qaboos University,
Oman
dmaingi@squ.edu.om

Abstract: *Autotheft is a crime that can be mitigated using artificial intelligence as a scientific approach. In this case, we assess the drivers driving pattern using both deep neural network and swarm intelligence algorithms. From the analysis we are able to obtain the driving signature of the driver which can be associated with the vehicle. The vehicle is then tracked and monitored. Next, a deviation from the usual driving signature of the owner or assigned driver would signify a possible instance of autotheft. Subsequently, the vehicle can be traced and reclaimed by the owner. The algorithms are evaluated based on their performance in analysing the datasets bearing variable features. The variations in features enable us to verify the efficacy and accuracy levels of the various algorithms that are used in the study. The metrics used for evaluation are the Mean Squared Error and the F1 Score for precision, accuracy and recall functionality.*

Keywords: *Deep learning, swarm intelligence, driving signature, intelligent transportation system.*

Received April 5, 2022; accepted April 28, 2022
https://doi.org/10.34028/iajit/19/3A/1

1. Introduction

The transport industry is essential in the growth of an economy because it facilitates resource sharing, mainly through communication and the transfer of goods and services. The popular means of transport is motor vehicles and this is marked by the elaborate road network infrastructure in modern society. Furthermore, as technology advances so does innovation, which necessitates the advancement in the automation of automobiles with an aim of improving the transportation process. In order to reduce errors, which may lead to accidents among other problems, vehicles have been automated, which has also led to other inventions like autonomous vehicles [11, 32]. These next generation technologies are also associated with vices and crimes such as auto-theft. As such, a need arises to curb the risks with practical counter-measures. Among these mitigating options, artificial intelligence presents itself as a viable option that can be applied in the monitoring, tracking, detection and recovery of stolen vehicles [3, 24].

Concerning the suitability of the algorithms present in various categories, the performance of algorithms based on deep learning supersedes the statistical-based machine learning algorithms because of their meta-heuristic properties [9, 34]. This means that optimality, completeness, precision and speed are guaranteed via carefully adjusted parameters and hyper parameters present in the algorithms to assure confidence in the obtained results [22].

This is an extended version of the conference paper [8], in which we make use of the Deep Belief Network (DBN) algorithm in addition to the Convolutional Neural Network (CNN), the Recurrent Neural Network with Long Short-Term Memory (RNN-LSTM), Deep Boltzmann Machines (DBM), and Deep Auto Encoders (AE). Moreover, we introduce the whale optimization algorithm among the following bio-inspired algorithms: particle swarm optimization, artificial bee colony, ant colony optimization and bat algorithm. These algorithms will be evaluated using the Figure 1. Score for accuracy and precision in addition to the Mean Squared Error which was used in the initial study [8]. Furthermore, in this study both the Aggressive driving dataset [23] and the Driving Behavior dataset [38] are used along with the Vehicular Trace Dataset [30].

This article is organized in the following order: The next section covers related work in the use of deep learning for the automobile industry. The section that follows is the methodology employed in performing the experiments for this study. Thereafter, sections four and five present the deep neural networks and swarm intelligence algorithms used in the experiments. Then section six discusses the attained results followed by a conclusion in the final section.

2. Related Work

The approaches used in previous related studies primarily rely on physical features of the stolen motor vehicle during their tracking, detection and recovery. Such approaches mostly rely on the use of license plate

recognition, Global Position System (GPS) tracking devices, sensors and other telecommunication devices fitted into the vehicle. These devices are used for geolocation and transmission of the vehicles position. However, this approach is relatively reliable until such devices are disabled, tampered with or even removed [2, 35].

Additionally, novel approaches have been introduced which cover a broad spectrum from biometrics, driving patterns as well as driver behavior analysis [1, 13, 16, 17, 19, 29, 36, 37]. However, the focus has been in areas such as accident prevention, drivers' intent prediction and monitoring of the vehicle, it's path or the driver's behavior [5, 6, 25, 39, 40]. The other authors who have ventured into auto-theft detection have used some of the previous methods mentioned earlier which are reliant on the physical characteristics of the vehicle [21].

Deep analytics, which refers to the use of sophisticated data processing techniques to yield information, has also been applied in other studies relating to automobile theft. The datasets may include both unstructured and semi-structured data from multiple sources [7, 14]. In this study, deep analytics is applied on the analysis of the driver's driving behavior, consequently yielding the driver's driving signature. Monitoring of the driver's driving signature would help track a vehicle in transit and also detect a case of its theft.

3. Methodology

The objective of this study is to evaluate several deep learning algorithms and swarm intelligence algorithms, described in the next section, on the driver's driving behavior pattern. Generally, the procedure followed in the implementation of the algorithms and subsequent analysis of the datasets [23, 30, 38] is shown below.

1. Load the dataset.
2. Pre-process the data.
3. Define the algorithm.
4. Configure the hyper parameters and parameters.
5. Define the activation function.
6. Define the Loss/Cost function.
7. Train the algorithm.
8. Optimize the network obtained.
9. Test the results obtained.

A well-prepared dataset will enhance the likelihood of obtaining better accuracy. The other configurations of the algorithm may be done stepwise as the overall reaction is observed so as to correctly analyze the dataset and obtain optimal results.

This research proposes the analysis of the driver's driving style in conjunction with the GPS data as a signature to monitor and track a vehicle and also detect its loss and recovery. The analysis of the driver's behavior will be done by the deep neural networks and swarm intelligence algorithms whose performance will be assessed based on the Mean Squared Error obtained from the execution of benchmark functions: Ackley function, Restringing function, Rosen rock function, Sphere function, Schaffer function and Himmelblau's function. The results obtained from these algorithms will be compared in order to instill the confidence needed to guarantee the quality in terms of accuracy.

4. Deep Neural Network Algorithms

The deep learning algorithms under consideration are the CNN, the RNN-LSTM, DBM, AE and DBN.

The CNN is actually popular in the analysis of images and by extension computer vision. In this case, this algorithm is used to systematically evaluate time series data. The algorithm follows the procedure in which the data's features are extracted using feature detectors to create feature maps which are then reduced through sum pooling. This is followed by flattening of the pooled feature maps which acts as the input layer for the artificial neural network with several fully connected hidden layers. The algorithm for the CNN is as follows [20, 34]:

Algorithm 1: Convolutional Neural Network

Input:
Initialize weights to a small randomly generated value, set learning rate to a small positive value, training period and the number of layers.
Output:
Begin with iteration n = 1
For n < MaxIteration, do
Forward propagate through convolution, pooling and then fully-connected layers
Derive Loss Function value for the data
Calculate the error term with respect to the weights for each type of layer
Backpropagate the error generated and calculate the change in gradient for both the weights $\nabla w_k^{(layer)}$ *and bias* $\nabla b_k^{(layer)}$ *respectively*
Update the weights and bias respectively
End For.

The RNN-LSTM algorithm can process the current input with respect to previously memorized input. This information flow through the network is controlled using the input gate *i*, forget gate *f* and output gate *o* These three gates constitute the memory cell which determines how much information to propagate through the network and which data should be discarded. The algorithm of the RNN-LSTM is shown in Algorithm 2 below [33].

Algorithm 2: Recurrent Neural Network with LSTM

Input:
Input sequence x_i*, training period T, learning rate* ϵ*, hidden layer* h *and output sequence y.*
Output:
Calculate the hidden vector sequence
For $\forall t = 1: T$ *do,*
At the input gate, calculate the effective data using $i_t = \sigma (w_{xi}x_t + w_{hi}h_{t-1} + w_{ci}c_{t-1} + b_i)$ *where* σ *is the logistic sigmoid function*

At the forget gate, determine the data to retain using $f_t = \sigma\,(w_{xf}x_t + w_{hf}h_{t-1} + w_{cf}c_{t-1} + b_f)$
Calculate the memory cell state using $c_t = \sigma c_{t-1} + i_c \tan h\,(w_{xc}x_t + w_{hc}h_{t-1} + b_c)$
Determine the output to be passed out the memory cell using $O_t = f_t\,(w_{xo}x_t + w_{ho}h_{t-1} + w_{co}c_{t-1} + b_o)$
Then calculate the effective output of the hidden layer using $h_t = O_t \tan h\,(c_t)$
End for
Calculate the predicted output using $y_t = f_t\,(w_{hy}h_t + b_y)$

The DBM algorithm utilizes the global optimization capability of simulated annealing. The visible input neurons are clamped onto specific states while the visible output neurons and the hidden neurons operate freely. Boltzmann machines learn their weights using simulated annealing. The correlation-based learning procedure of the DBM is presented in algorithm 3 below [12].

Algorithm 3: Deep Boltzmann Machine

Input:
Initialize w_{ij} *as uniform random values in* $[-a_0, a_0]$, *where* $a_0 = 0.5$ or 1.
Set the initial temperature T_0 *and the final temperatures* T_f
Output:
At the clamping phase, present the pattern and for each example x_i, *perform simulated annealing until* T_f *is reached.*
At each T, relax the network by the Boltzmann distribution for a length of time through updating the states of the unclamped (hidden) units

$$x_i = \begin{cases} +1, \text{ with probability } P_i \\ -1, \text{ with probability } 1\text{-}P_i \end{cases}$$

Where P_i *is calculated using;*

$$P_i = \frac{1}{1 + e^{-\frac{net_i}{T}}} \text{ and } net_i = \sum_{j=1,\, j \neq i}^{J} w_i\, x$$

Then update T *by the annealing schedule and at* T_f *estimate the correlation in the clamped condition using;*

$$P_{ij}^{+} = E[x_i\, x_j] \text{ where i, j=1,2,…,J; } i \neq j$$

At the free-running phase, only the input neurons are clamped and the output neurons are free-running hence at T_f *estimate the correlation in the free-running condition using;*

$$P_{ij}^{-} = E[x_i\, x_j] \text{ where i, j = 1,2,…, J; } i \neq j$$

Perform the weight update using;

$$\Delta w_{ij} = \eta\left(P_{ij}^{+} - P_{ij}^{-}\right) \text{ where i, j = 1,2,…, J; } i \neq j$$

Where $\eta = \frac{\varepsilon}{T}$, *with* ε being a small positive constant
Repeat the steps above for next epoch until there is no change in w_{ij}, $\forall i, j$.

The deep autoencoder makes use of several stacked hidden layers that force the data to converge hence retain only the optimal data that is suitable for predictions. Furthermore, the non-linear hidden layers allow the network to learn more complex encoding functions and enhance higher precision. The following is the deep autoencoder algorithm [10].

Algorithm 4: Deep Autoencoder Neural Network

Input:
Input matrix $A \in \{0, 1\}^{m \times n}$, *where rows and columns correspond to vector values for input features*
Output:
Fix a number h *of hidden units* ($h \in N$, $h<m$), *and a number* d *of hidden layers* ($d \in \{1, \ldots, maxhl\}$)
Training: for each driver profile a_i *of* A, *where* $i \in [1, m]$*: for each* d *hidden layer compute hidden activation* h_i *from the input vectors*
Compute reconstructed output $\hat{a}_i$ *from hidden activation* h_i
Perform the stochastic gradient descent and compute the MSE between a_i *and* $\hat{a}_i$
Back-propagate error gradient to update weight parameters
Testing: for each driver profile a_i *of A, where* $i \in [1, m]$*: autoencode* a_i *and produce* $\hat{a}_i$
Set $\hat{a}_i$ *as* i^{th} *row of the output matrix* $\tilde{A}$.

The DBN is a combination of Restricted Boltzmann Machines (RBM) that are stacked. The initial RBM in the stack contains undirected connections but the others have directed connections that enable them to act as feature detectors. RBMs are special DBM composed of dual layer networks that allow for both intra-layer and inter-layer connections. Therefore, DBN presents us with a means of extracting and exploring knowledge present in the hierarchical abstraction of the dataset. The procedure of the DBN is presented in algorithm 5 below [18, 31].

Algorithm 5: Deep Belief Network

Input:
Set the initial temperature T_0, *minimum temperature* T_{min}, *intra-layer iteration limit* D_{max}, *network overall iteration limit* G_{max}, *objective function threshold* R_{end}, *initial network depth* $D = 2$ *(input layer and output layer), and memory matrix I.*
Output:

i. *For i = 1:* D_{max} *align all the symbols correctly*

   D = D + 1, T = T0

   a. *Generate* N_i, *form current network structure C based on* N_i, *and calculate the reconstruction error R of C.*
   b. *For j = 1:* G_{max} *The new number of neurons N' is randomly generated as the undetermined solution, the DBN structure C' formed by N' is the candidate DBN structure, and the reconstruction error R' corresponding to C' is calculated.*
      i. *If* $\Delta R = R' - R < 0$ *or* $\exp(-\Delta R/T) >$ *rand* $C = C'$, $j = 1: G_{max}$
      ii. If $j \geq G_{max}$[1] or $T \leq T_{min}$ or $R \leq R_{end}$ find C_{best} in matrix I and search the adjacent domain of C_{best} to obtain C_{final}
   c. *If* $D \geq Dmax$ *or* $R \leq Rend$ *return the optimal network structure*

5. Swarm Intelligence Algorithms

This study uses the following swarm intelligence algorithms; Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC), Ant Colony Optimization (ACO), Bat Algorithm (BA) and Whale Optimization Algorithm (WOA).

The PSO algorithm mimics bird flocking and predation. Thus, the particles represent birds in a search space. The particles will fly in restricted directions of the bounded search space which the group perceives to be ideal. Their velocities are dynamically adjusted based on

both their individual experience and that of the other particles in the population. Adjustment of the inertia weight facilitates both the global and local search capability of the algorithm during its implementation [15].

Algorithm 6: Particle Swarm Optimization

Initialization:
For each particle $i = 1, \ldots, N_p$ *, do*
- i. *Initialize the particle's position with a uniformly distribution as* $P_i(0) \sim U(LB, UB)$*, where LB and UB represent the lower*
- ii. *and upper bounds of the search space*
- iii. *Initialize* $pbest$ *to its initial position:* $pbest(i, 0) = P_i(0)$.
- iv. *Initialize* $gbest$ *to the minimal value of the swarm:* $gbest(0) = argminf[P_i(0)]$.
- v. *Initialize velocity:* $V_i \sim U(-|UB-LB|, |UB-LB|)$.

Termination:
Repeat until a termination criterion is met
For each particle $i = 1, \ldots, N_p$*, do*
- i. *Pick random numbers:* $r_i, r_2 \sim U(0, 1)$
- ii. *Update particle's velocity using* $V_i(t+1) = \omega V_i(t) + c_1 r_1 \left(pbest(i, t) - p_i(t)\right) + c_2 r_2 (gbest(t) - Pi(t))$
- iii. *Update particle's position using* $P_i(t+1) = P_i(t) + V_i(t+1)$.
- iv. *If* $f[P_i(t)] < f[pbest(i, t)]$*, do*
  - a. *Update the best-known position of particle* $i: pbest(i, t) = p_i(t)$.
  - b. *If* $f[P_i(t)] < f[gbest(i, t)$*, update the swarm's best-known position:* $gbest(t) = p_i(t)$.
- v. $t \leftarrow (t+1)$;

Output gbest(t) *that holds the optimal solution.*

The ABC algorithm is dependent on the foraging patterns of the honey bees in nature. The bees are split into groups based on their responsibilities. The unemployed onlooker bees will wait for the employed bees who will use the waggle dance to communicate to them about the viability of the food source based on the quality and quantity of nectar. These bees will then be recruited to that food source or alternatively start scouting for potential food sources. The food sources in this case are the plausible solutions available in the search space. The quantity and quality of these solutions is measured using fitness and probabilistic functions respectively as shown in the algorithm steps below [17].

Algorithm 7: Artificial Bee Colony

Initialization:
For each bee $i = 1, \ldots, n$*, do*
- i. *Initialize the positions of bees* $(x_i = 1, \ldots, SN)$
- ii. *Randomly initialize the food sources within the search space using* $x_{ij} = x_j^{min} + rand(0, 1)(x_j^{max} - x_j^{min})$ *where* x_{ij} *represents the parameter for* i_{th} *employed bee on* j_{th} *dimension*
- iii. *Evaluate fitness* (fit_i) *of the population*

Termination:
Repeat until $(iter \leq MaxCycles)$
- i. *Generate new positions which represent new solutions* v_{ij} *by the employed bees and calculate the fitness value* (fit_i) *on the new population*
- ii. *Apply greedy selection process between the initial solutions* (x_{ij}) *and the resultant solutions* (v_{ij}).
- iii. *Calculate the probability values* (P_i) *for the solutions* (x_i)
- iv. *Generate other new solutions* (v_i) *for the onlookers from the solutions* (x_i) *selected depending on their* (P_i) *values and evaluate the nectar quality of new positions using* (fit_i).
- v. *Apply greedy selection process to solutions found by onlooker bees.*
- vi. *If there is an abandoned solution for the scout then replace it with a new random solution* (x_i).
- vii. *Memorize the best solution so far* $(x_{best} \leftarrow x_i \,||\, v_i)$.
- viii. *Increment iteration* $(iter + 1)$

Output (x_{best}) *the best memorized solution.*

The ACO algorithm is based on the foraging behavior of ants which start by randomly exploring the area surrounding their nest. The ant that finds a worthwhile food source will evaluate the quality and quantity of the food which determines the rate of pheromone it will deposit on its trail back to the nest. The rest of the ants evaluate the viability and shortest route to the source of the food using the pheromones deposited on the trail as a means of communication. Therefore, the essence of the ACO approach is the pheromone model which uses a probability function to evaluate the probable solutions in each iteration measured by pheromones deposited and further quantified using the evaporation rate. The algorithm is presented below [26].

Algorithm 8: Ant Colony Optimization Algorithm

Initialization:
For each ant $j = 1, \ldots, n_a$ *do*
- i. *Initialize pheromones trail* (T_0)
- ii. *Assign best solution* (S^{bs}) *at any time in each iteration* t
- iii. *Assign the minimum* T_{min} *and maximum* T_{max} *value for the pheromone trail*
- iv. *Store the best solution in each iteration* (S^{ib})

Termination:
Repeat until a termination criterion is met
For $j = 1, \ldots, n_a$ *do*
- i. *Construct a solution* (S^{ib}) *using a probabilistic function*
- ii. *If* S^{ib} *is a valid solution then optionally perform a local search to find the best solution* S^{bs}
- iii. *If* $\left(f(S^{is}, t) < f(S^{bs}, t) \text{ or } S^{bs} = NULL\right)$ *then* $S^{bs} \leftarrow S^{ib}$
- iv. *Update the pheromone bounds* (T_{min}, T_{max}) *using* $T_{ij} \leftarrow (1 - \rho)\, T_{ij}, \forall(i, j)$ *where* ρ *is the evaporation rate of the pheromones which satisfies* $0 < \rho \leq 1$
- v. *If (stagnation behavior detected) then initialize pheromones trail* (t_{max})
- vi. *Update the iteration* $t \leftarrow (t + 1)$;

Output S^{bs} *which is considered the optimal solution*

The bat algorithm is another met heuristic optimization algorithm based on the echolocation technique of microbats which they use for detecting prey, avoiding obstacles and locating their roosting crevices. These bats emit short bursts of sound pulses whose echo is detected by the bats and together with the time delay, detection difference between the ears and pitch of the echo to build a three dimensional view of their surroundings. Therefore, the bat algorithm relies on the way bats fly

randomly from their crevices at a given velocity and emit sound pulses at an adjustable frequency to locate their food. The bat algorithm is presented below [4].

Algorithm 9: Bat Algorithm

Initialization:
For each bat i = 1, …, n, do
- i. *Initialize the bat positions x_i (i = 1, …, n) and velocity v_i*
- ii. *Define the pulse frequency f_i at x_i*
- iii. *Initialize pulse rates r and the loudness A*

Termination:
Repeat until ($t < t_{max}$)
- i. *Generate new solutions by;*
  - a. *adjusting frequency $f_i = f_{min} + (f_{max} - f_{min})\beta$, where β is a random vector in the range of [0,1]*
  - b. *updating velocities $v_i^t = v_i^{t-1} + (x_i^{t-1} - x^{best})f_i$*
  - c. *update locations $x_i^t = x_i^{t-1} + v_i^t$*
- ii. *If (rand > r), do*
  - a. *Select a solution among the best solutions*
  - b. *Generate a local solution around the selected best solution*
- iii. *Generate a new solution by flying randomly*
- iv. *If (rand < A & $f(x_i) < f(x^{best})$) then accept the new solutions*
- v. *Rank the bats and find the current best x^{best}*

Output the optimal solution (x^{best}).

The WOA is based on the hunting behavior of the humpback whales. These whales may start by randomly chasing the prey in order to encircle and capture it and later on employ the bubble net strategy. This feeding strategy enables the whales to survive in their environment. The algorithm starts by assigning the whale population with random solutions. This assumes either a minimum or maximum value for the optimal solution. Afterwards, the objective function is calculated for each search agent which must update its location on each iteration depending on their best solution. This is then repeated until an optimal solution is found. The whale optimization algorithm is presented below [27, 28].

Algorithm 10. Whale Optimization Algorithm

Initialization:
For each whale i = 1, …, n, do
- i. *Randomly initialize the whale population X_i (i = 1, …, n)*
- ii. *Calculate the fitness of each search agent*

Termination:
Repeat until ($t < t_{maxiterations}$)
- i. *For each search agent;*
  - a. *Update a, A, C, l, and p where A and C are coefficient vectors and l is the logarithmic spiral*
  - b. *If ($p < 0.5$), and ($|A| < 1$)*
    - i. *Update the current search agent position by $X_{(t+1)} = X^*_{(t)} - A.D$*
    - ii. *But if ($|A| > 1$) select a random search agent X_{rand} and update the current search agent by $X_{(t+1)} = X_{rand} - A.D$*
  - c. *Otherwise, If ($p \geq 0.5$),*
    - i. *Update the position of the current search agent by $X_{(t+1)} = D.e^{bl}.cos(2\pi l) + X^*_{(t)}$*
  - d. *Check if any search agent goes beyond the search space and amend it.*
  - e. *Calculate the fitness of each search agent.*
- ii. *Update X* if there is a better solution t = t+1*

Output the optimal solution X = the best search agent*

6. Experiments and Results

The deep neural network and swarm intelligence algorithms are implemented in python programming language. The parameters and hyper parameters of the algorithms are configured and consistently adjusted on several runs based on performance. In this study, there were three datasets used as shown in Table 1.

Table 1. The datasets.

No.	Dataset	Short Name
1.	Vehicular Trace Dataset [30]	Dataset 1
2.	Aggressive driving dataset [23]	Dataset 2
3.	Driving Behaviour dataset [38]	Dataset 3

The performance of the algorithms was evaluated by using the Mean Squared Error (MSE) defined in (1) below.

$$\text{MSE}=\frac{\sum_{i=1}^{N}(y_i-\hat{y}_i)^2}{N} \quad (1)$$

Where $\hat{y}_i$ is the predicted value and y_i is the actual value. The MSE is the average squared difference between the predicted values and the actual values.

The datasets were split into three divisions during the experiments; 70% for training, 20% for validation and the other 10% for testing. The idea behind splitting the datasets is to compare the results obtained with the original values. This makes it easy to verify the accuracy of the respective algorithms in terms of the obtained results as shown in the MSE values of the three datasets in Tables 2, 3, 4.

Table 2. The Mean Squared Error (MSE) obtained when performing driver behavior analysis on Dataset 1.

	Loss/Cost Function	CNN	RNN-LSTM	DBM	AE	DBN	PSO	ABC	ACO	BA	WOA
F_1	Ackley Function	1.7329	0.4578	0.4101	1.7305	1.6791	0.7741	1.5038	1.2596	2.1733	0.8176
F_2	Rastrigin Function	1.6521	1.5673	1.02947	0.6958	1.1738	2.4205	1.7396	2.6681	2.6908	2.0107
F_3	Rosenbrock Function	1.7952	0.1203	1.0592	1.3942	0.7235	1.7103	1.8529	3.7305	2.5842	1.9958
F_4	Sphere Function	2.0178	1.2995	2.3061	3.9148	2.7945	2.3059	0.4099	0.9426	3.2227	2.5132
F_5	Schaffer Function	1.1262	1.4189	2.2979	2.7427	2.0046	1.9904.	1.0733	0.4471	2.6932	1.7661
F_6	Himmelblau's Function	2.4911	2.2194	3.9647	3.4031	2.4395	1.5390	2.5412	4.7841	4.6117	1.8326

Table 3. The MSE obtained when performing driver behavior analysis on dataset 2.

	Loss/Cost Function	CNN	RNN-LSTM	DBM	AE	DBN	PSO	ABC	ACO	BA	WOA
F_1	Ackley Function	2.9472	1.3925	1.5650	1.0385	0.3221	1.4599	1.0251	1.0498	1.3490	1.4491
F_2	Rastrigin Function	1.4482	1.9937	1.7932	1.7943	1.0629	2.4033	1.9245	1.1719	1.4962	2.1803
F_3	Rosenbrock Function	2.1193	0.2847	1.9471	0.9221	1.3401	1.4042	1.2485	0.3421	2.5691	1.7601
F_4	Sphere Function	2.6201	1.6005	2.4310	1.3339	1.2005	2.4309	1.3021	1.9032	1.9837	2.5152
F_5	Schaffer Function	1.3492	1.2289	2.0122	1.7066	1.2054	2.4309	1.2294	1.4502	3.6501	2.3640
F_6	Himmelblau's Function	2.4503	1.5460	2.1132	1.9011	2.1201	2.2231	2.4025	1.5049	3.4402	2.3231

Table 4. The MSE obtained when performing driver behavior analysis on dataset 3.

	Loss/Cost Function	CNN	RNN-LSTM	DBM	AE	DBN	PSO	ABC	ACO	BA	WOA
F_1	Ackley Function	2.0119	0.1039	1.4933	1.8873	1.2945	1.2997	1.0405	1.2143	2.2190	1.4578
F_2	Rastrigin Function	1.2941	1.2833	1.9244	1.0095	2.5450	1.0037	1.6307	1.4552	2.3901	1.1365
F_3	Rosenbrock Function	1.3921	0.0239	1.9930	1.4550	1.9745	1.7003	0.9017	1.2071	1.3081	1.9033
F_4	Sphere Function	2.0143	1.3049	2.0078	2.3993	1.1067	1.9975	0.0630	1.5600	1.4501	2.0105
F_5	Schaffer Function	1.3032	1.5822	1.4057	1.6770	1.2011	1.4367	1.3002	2.6722	1.2844	1.9274
F_6	Himmelblau's Function	1.9910	1.9925	2.4011	1.4401	2.3851	1.2829	1.8541	2.4901	2.0012	1.3966

Additionally, the accuracy *(a)*, precision *(p)* recall (*r*) and F1-score measures are used to verify the results through assessing the predictability performance of the algorithms. Hence, we are able to determine the number of correct predictions against predictions made. These measures are calculated as shown below.

$$a = \frac{TP + TN}{TP + FP + TN + FN} \quad (2)$$

$$p = \frac{TP}{TII + FP} \quad (3)$$

$$r = \frac{TP}{TII + FN} \quad (4)$$

$$F_1 = \frac{2pr}{\pi + r} \quad (5)$$

Where the True Positive *(TP)*, is the True Negative *(TN)*, is the False Positive (*FP)* and is the False Negative (*FN).*

In order to make use of the performance measures, we generate a confusion or matching matrix representing the frequency of the actual values against the predicted values. Then, we calculate and tabulate the values as shown in Table 5, from these values we are able to verify the performance of the algorithms on the three datasets. The results show that the RNN-LSTM algorithm has the best overall F1 score of 97.78% in all the three datasets followed by the ABC algorithm that performed better amongst the swarm intelligence algorithms with a maximum 96.5% *F1* score. The CNN algorithm also performed well on dataset III particularly on accuracy (90.78%) and precision (94.16%). Moreover, the CNN and PSO algorithms were second in performance to the RNN-LSTM and ABC algorithms based on the *F1* Scores. On average, the DNN performed better than the swarm intelligence algorithms on all the three datasets.

Table 5. The accuracy, precision, recall and F1. Score values obtained for dataset 1, dataset 2 and dataset 3.

Dataset	Measure	CNN	RNN-LSTM	DBM	AE	DBN	PSO	ABC	ACO	BA	WOA
1	**Accuracy (a)**	73.67%	95.02%	77.62%	89.05%	91.44%	91.69%	96.71%	87.90%	92.56%	93.83%
	Precision (p)	97.97%	98.73%	88.72%	91.23%	94.31%	93.07%	97.21%	80.03%	86.55%	89.48%
	Recall (r)	88.33%	96.84%	88.01%	92.52%	91.38%	91.71%	95.82%	83.07%	94.57%	93.21%
	F_1 Score	92.90%	97.78%	88.36%	91.87%	92.82%	92.38%	96.51%	81.52%	90.38%	91.31%
2	**Accuracy (a)**	85.84%	95.21%	91.38%	90.77%	79.52%	90.47%	92.46%	75.67%	92.45%	84.58%
	Precision (p)	92.27%	97.04%	90.01%	88.39%	91.75%	89.16%	96.43%	85.61%	93.21%	93.18%
	Recall (r)	92.15%	95.23%	88.68%	94.27%	86.38%	93.14%	92.32%	88.01%	92.18%	94.52%
	F_1 Score	92.21%	96.13%	89.34%	91.24%	88.98%	91.11%	94.33%	86.79%	92.69%	93.85%
3	**Accuracy (a)**	90.78%	91.26%	78.72%	90.11%	89.04%	90.14%	88.43%	85.45%	88.63%	79.46%
	Precision (p)	94.16%	96.73%	80.46%	86.51%	93.48%	92.74%	93.16%	91.79%	84.72%	90.18%
	Recall (r)	90.86%	93.77%	85.68%	90.94%	91.22%	90.68%	92.23%	80.77%	82.71%	91.25%
	F_1 Score	92.48%	95.23%	82.99%	88.67%	92.34%	91.70%	92.69%	85.93%	83.70%	90.71%

During the parameter tuning, results also show that the depth of the deep neural networks in terms of the number of hidden layers had a relatively minimal influence on the predictability accuracy. This is because increasing the number of hidden layers past 50 slowed down the network and this had a minimal increase in relation to the accurate predicted values.

Furthermore, the breadth of the network in terms of the number of neurons per layer had a correlation with the number of features that were being observed in the dataset. This is because increasing the number of neurons past 20 did not make a significant change in the MSE. The RNN-LSTM performed relatively better than the other algorithms.

Generally, increasing the breadth and depth of the network had a relatively small impact on the performance in terms of classification and matching of the drivers with their driving pattern. However, there was some dependence on the features being observed. This means that observing fewer features for a larger dataset

increased the confidence levels for the predictability compared to many features for the same large dataset.

The ABC algorithm and the PSO algorithm also made good predictions although the bio-inspired algorithms made significant changes when the parameters were slightly changed. This is unlike the deep neural networks which were dependent on the hyper parameter changes in order to lower the errors in predictions.

Overall, there was a high dependence of both hyper parameter and parameter tuning to feature selection to achieve accurate prediction by the algorithms.

7. Conclusions

The ultimate objective of this study was to use the results as a means of specifically identifying a driver based on their driving pattern and using this information to detect and possibly recover a stolen vehicle. The results obtained show that this is possible and presents possibilities for further research.

The analysis of driving behavior provides insight to the vehicles design, safety of the occupants, quality and maintenance. Therefore, there are several aspects of any driver's behavior that can be studied in order to help us understand how we can improve the quality of driving and assure safety.

Furthermore, the driver's driving signature is dependent upon their experience and skill as exhibited by their reactions over time. This is dynamic owing to the fact that their reactions may change, more so, when other extenuating factors are involved such as the weather, road condition or the kind of occupants present in the vehicle.

Acknowledgment

This work was supported by Kiriri Women's University of Science and Technology.

References

[1] Aishwarya K. and Manjesh R., "A Novel Technique for Vehicle Theft Detection System Using MQTT on IoT," *in proceeding of International Conference on Communication, Computing and Electronics Systems*, Singapore, pp. 725-733, 2020.

[2] Alhussein M., Aurangzeb K., and Haider S., "Vehicle License Plate Detection and Perspective Rectification," *Elektronika Ir Elektrotechnika*, vol. 25 no. 5, pp. 47-56, 2019.

[3] Alsrehin N., Klaib A., and Magableh A., "Intelligent Transportation and Control Systems Using Data Mining and Machine Learning Techniques: A Comprehensive Study," *IEEE Access*, vol. 7, pp. 49830-49857, 2019.

[4] Bangyal W., Ahmad J., Rauf H., and Pervaiz S., "An Improved Bat Algorithm Based on Novel Initialization Technique for Global Optimization Problem," *International Journal of Advanced Computer Science and Applications*, vol. 9 no. 7, pp. 158-166, 2018.

[5] Bernardi M., Cimitile M., Martinelli F., and Mercaldo F., "Driver and Path Detection Through Time-Series Classification," *Journal of Advanced Transportation*, pp. 1-20, 2018.

[6] Bhalerao J., Kadam A., Shinde A., Mugalikar V., and Bhan H., "Proposed Design on Driver Behavioral Analysis," *International Journal of Engineering Research and Technology*, vol. 9, no. 5, pp. 554-557, 2020.

[7] Bhuyan H. and Pani S., *Intelligent Data Analytics for Terror Threat Prediction: Architectures, Methodologies, Techniques and Applications*, Wiley Online Library, 2021.

[8] Bosire A. and Maingi D., "Using Deep Analysis of Driver Behavior for Vehicle Theft Detection and Recovery," *in proceedind of International Arab Conference on Information Technology*, Oman, pp. 1-6, 2021.

[9] Chalapathy R. and Chawla S., "Deep Learning for Anomaly Detection: A Survey," *arXiv:1901.03407v2 [cs.LG]*, 2019.

[10] Chicco D., Sadowski P., and Baldi P., "Deep Autoencoder Neural Networks for Gene Ontology Annotation Predictions," *in proceeding of The 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics*, New York, pp. 533-540, 2014.

[11] Dong Z., Shi W., Tong G., and Yang K., "Collaborative Autonomous Driving: Vision and Challenges," *in proceeding of International Conference on Connected and Autonomous Driving*, USA, pp. 17-26, 2020.

[12] Du K. and Swamy M., *Neural Networks and Statistical Learning*, Springer, 2019.

[13] Elngar A. and Kayed M., "Vehicle Security Systems Using Face Recognition Based on Internet of Things," *Open Computer Science*, vol. 10, no. 1, pp. 17-29, 2020.

[14] Feng M., Zhen J., Ren J., Hussein A., Li X., Xi Y., and Liu, Q., "Big Data Analytics and Mining for Effective Visualization and Trends Forecasting of Crime Data," *IEEE Access*, vol. 7, pp. 106111-106123, 2019.

[15] Gao D., Li X., and Chen H., "Application of Improved Particle Swarm Optimization," *Mathematical Problems in Engineering*, vol. 2019, pp. 1-10, 2019.

[16] Girma A., Yan X., and Homaifar A., "Driver Identification Based on Vehicle Telematics Data using LSTM-Recurrent Neural Network," *in proceedind of the IEEE 31st International Conference on Tools with Artificial Intelligence*, USA, pp. 894-902, 2019.

[17] Hakli H. and Kiran M., "An improved artificial bee colony algorithm for balancing local and global search behaviors in continuous optimization," *International Journal of Machine Learning and Cybernetics*, vol. 11, n. 9, pp. 2051-2076, 2020.

[18] Jiang J., Zhang J., Zhang L., Ran X., Jiang J., and Wu Y., "DBN Structure Design Algorithm for Different Datasets Based on Information Entropy and Reconstruction Error," *Entropy*, vol. 20, no. 12, pp. 1-18, 2018.

[19] Kang Y., Park K., and Kim H., "Automobile Theft Detection by Clustering Owner Driver Data," *in Procceding of the "17th Escar Europe : Embedded Security in Cars, Ruhr-Universität Bochum*, pp. 185-199, 2019.

[20] Khan A., Sohail A., Zahoora U., and Qureshi A., "A Survey of the Recent Architectures of Deep Convolutional Neural Networks," *Artificial Intelligence Review*, vol. 53, no. 8, pp. 5455-5516, 2020.

[21] Kommaraju R., Kommanduri R., Lingeswararao S., Sravanthi B., and Srivalli C., "IoT Based Vehicle (Car) Theft Detection," *in proceeding of International Conference on Image Processing and Capsule Networks*, pp. 620-628, 2020.

[22] Koutsoukas A., Monaghan K., Li X., and Huan J., "Deep-learning: Investigating Deep Neural Networks Hyper-Parameters and Comparison of Performance to Shallow Methods for Modeling Bioactivity Data," *Journal of Cheminformatics*, vol. 9 no. 42, pp. 1-13, 2017.

[23] Kumar V. and Veerala, K., "Aggressive driving dataset," Retrieved from https://www.kaggle.com/datasets/veeralakrishna/aggressive-driving-data, 2022.

[24] Li J., Cheng H., Guo H., and Qiu S., "Survey on Artificial Intelligence for Vehicles," *Automotive Innovation*, vol. 1, no. 1, pp. 2-14, 2018.

[25] Martinelli F., Mercaldo F., Orlando A., Nardone V., Santone A., and Sangaiah A., "Human Behavior Characterization for Driving Style Recognition in Vehicle System," *Computers and Electrical Engineering*, vol. 83, pp.102504, 2020.

[26] Mavrovouniotis M., Muller F., and Yang S., "Ant Colony Optimization With Local Search for Dynamic Traveling Salesman Problems," *IEEE Transactions on Cybernetics*, vol. 47, no. 7, pp. 1743-1756, 2016.

[27] Mirjalili S. and Lewis A., "The Whale Optimization Algorithm." *Advances in Engineering Software*, vol. 95, pp. 51-67, 2016.

[28] Mohammed H., Umar S., and Rashid T., "A Systematic and Meta-Analysis Survey of Whale Optimization Algorithm," *Computational Intelligence and Neuroscience*, vol. 2019, pp. 1-27, 2019.

[29] Ramesh M., Akruthi S., Nandhini K., Meena S., Gladwin S., and Rajavel R, "Implementation of Vehicle Security System using GPS,GSM and Biometric," *in proceeding of Women Institute of Technology Conference on Electrical and Computer Engineering*, India, pp. 71-75, 2019.

[30] Rettore P., "Vehicular Trace Data, " Retrieved from http://www.rettore.com.br/prof/vehicular-trace, 2022.

[31] Rizk Y., Hajj N., Mitri N., and Awad M., "Deep Belief Networks and Cortical Algorithms: A Comparative Study for Supervised Classification" *Applied Computing and Informatics*, vol. 15, no. 2, 81-93, 2019.

[32] Rodg J. and Jaiswal S., "Comprehensive Overview of Neural Networks and its Applications in Autonomous Vehicles," *Computational Intelligence in the Internet of Things*, pp. 159-173, 2019.

[33] Sherstinsky A., "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network," *Physica D: Nonlinear Phenomena*, vol. 404, pp. 1-43, 2020.

[34] Shrestha A. and Mahmood A., "Review of Deep Learning Algorithms and Architectures," *IEEE Access*, vol. 7, pp. 53040-53065, 2019.

[35] Singh S. and Kumar, P., "Automatic Car Theft Detection System Based on GPS and GSM Technology," *International Journal of Trend in Scientific Research and Development*, vol. 3, n. 4, pp. 689-692, 2019.

[36] Villa M., Gofman M., and Mitra S., "Survey of Biometric Techniques for Automotive Applications," *Information Technology - New Generations*, vol. 738, pp. 475-481, 2018.

[37] Wang W., Xi J., and Zhao D., "Driving Style Analysis Using Primitive Driving Patterns With Bayesian Nonparametric Approaches," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 8, pp. 2986-2998, 2018.

[38] Yüksel A. and Atmaca S., "Driving Behavior Dataset," Retrieved from https://doi.org/10.17632/jj3tw8kj6h.3, 2022.

[39] Zhang J., Wu Z., Li F., Xie C., Ren,T., Chen J., and Liu L, "A Deep Learning Framework for Driving Behavior Identification on In-Vehicle CAN-BUS Sensor Data," *Sensors*, vol. 19, no. 6, pp. 1-17, 2019.

[40] Zinebi K., Souissi N., and Tikito K., "Driver Behavior Analysis Methods: Applications oriented study," *in proceeding of The 3rd International Conference on Big Data, Cloud and Applications - BDCA'18*, Morocco, 2018.

Adrian Bosire is a PhD candidate at Jomo Kenyatta University of Agriculture and Technology, Juja, Kenya. He received his B.Sc. degree in Information Technology from the Jomo Kenyatta University of Agriculture and Technology, Juja, Kenya in 2010. He earned his M.Sc. degree in Information Technology from Preston University, Kohat, Pakistan in 2014. His research interests include artificial intelligence, deep learning, intelligent transport systems and cyber-security.

Damian Maingi is an Associate Professor at Sultan Qaboos University (SQU), Muscat, Oman. He received his PhD in Algebraic Geometry at the Universite de Nice Sophia-Antipolis in Nice, France in 2010. He got his MSc degree in Mathematics at the University of Nairobi (UoN) in 2005. He graduated with a BSc Mathematics degree in 2002 from the University of Nairobi. His current research interests are vector bundle construction and deep learning.