Journal of Integrated
SCIENCE & TECHNOLOGY

Enhancing security in vehicular Ad hoc networks: A novel approach using DSFLA, SACVAEGAN, and OAEF

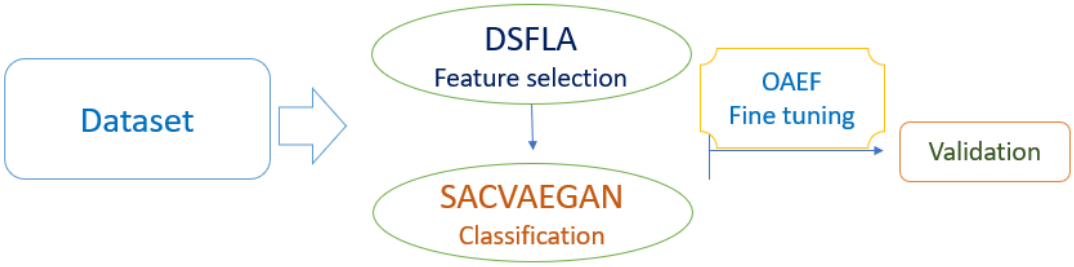
Venkata Subbaiah Desanamukula,[1] B. Gunapriya,[2] M. Janardhan,[3] Venkateswarlu Gundu,[4] Syed Ziaur Rahman,[5] R.J. Anandhi,[2] Ramesh Vatambeti[6*]

[1] Department of Computer Science and Engineering, Lakireddy Bali Reddy College of Engineering, Mylavaram 521230, India. [2]New Horizon College of Engineering, Ring Road, Bellandur Post, Bengaluru, India. [3]Department of Computer Science and Engineering, G. Pullaiah College of Engineering and Technology, Kurnool, India. [4]Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram 522302, India. [5]Faculty of Information Technology, Majan University College (Affiliated to University of Bedfordshire, United Kingdom), Muscat- 710, Oman. [6]School of Computer Science and Engineering, VIT-AP University, Vijayawada, India.

Received on: 28-Dec-2023, Accepted and Published on: 28-May-2024

Article

ABSTRACT



Vehicular Ad hoc Networks (VANETs), a crucial type of Mobile Ad hoc Networks (MANETs), consist of wirelessly connected vehicles. Ensuring the safety of VANETs is critical, as a single security failure can result in significant loss of life. Traditional Intrusion Detection Systems (IDS) often fail to keep pace with the increasing sophistication of pattern-based attacks. To address this, we propose a novel optimal feature-selection method called the Differentiated Shuffled Frog-Leaping Algorithm (DSFLA). Traditional IDSs struggle with complex attacks. We propose DSFLA for feature selection, SACVAEGAN for data enhancement, and OAEF for parameter optimization. Testing on the Car Hacking dataset shows superior performance, promising enhanced VANET security Additionally, we introduce a Self-Attention-Based Conditional Variational Autoencoder Generative Adversarial Network (SACVAEGAN) to generate virtual samples and enhance training data. To optimize hyper-parameters, we employ an enhanced artificial electric field (AEF) technique known as Opposition-Based AEF (OAEF). Experimental results on the Car Hacking dataset demonstrate that our approach not only effectively detects intrusions but also significantly outperforms current state-of-the-art deep learning models in classification tasks. The designed methodology enhances IDS efficiency, offering robust security solutions for VANETs. VANETs are susceptible to various sophisticated attacks due to their dynamic and decentralized nature. The designed approach strengthens the network's resilience by providing advanced mechanisms for detecting and responding to malicious activities, thereby reducing the risk of successful cyber-attacks.

Keywords: Vehicular Ad hoc Networks; Differentiated shuffled frog-leaping algorithm; Conditional Variational Autoencoder Generative Adversarial Network; Artificial electric field; Intrusion Detection Systems.

INTRODUCTION

Embedded updates and networking devices have all contributed to the expansion of the Internet of Vehicles (IoV). Privacy concerns are just some of the threats that remain in the IoV. The increasing prevalence of intelligent services, remote access, and routine network updates has given rise to several privacy and security problems.[1] Therefore, there is a lot of worry about security flaws in IoV data transit. Malicious attacks and data manipulation, as well as system outages, can put people and their possessions in danger.[2]

The first threat is V2V communication, in which an attacker might potentially harm drivers by tampering with their data. Simultaneously, the Vehicle-to-Infrastructure (V2I) communication situation might provide an additional security concern. Multiple attack models for intelligent cars have created

*Corresponding Author: Dr. Ramesh Vatambeti
Email: v2ramesh634@gmail.com

Cite as: *J. Integr. Sci. Technol., 2024, 12(6), 828.*
URN:NBN:sciencein.jist.2024.v12.828
DOI: 10.62110/sciencein.jist.2024.v12.828

numerous privacy and security issues for intelligent transportation networks.[3] The VANET communication network has major security concerns due to the possibility of signal jamming and spoofing by cyber attackers. Because of this, the integrity of the message delivered may be compromised, and the intended goals of the V2X system may not be achieved.[4]

An increasingly significant category of Mobile Ad hoc Networks (MANETs) is comprised of VANET, referring to a network of automobiles that are wirelessly connected to one another. Vehicle-to-Infrastructure (V2I) communication, also abbreviated as V2V and V2I, respectively, performs the bulk of the communication in VANETs.[5] The term "V2V communication" describes interactions between cars, whereas the term "V2I communication" describes interactions between vehicles and infrastructure nodes like traffic lights and petrol stations. To include the WAVE protocol, the IEEE 802.11 standard was updated to become IEEE 802.11p. Wireless Access in a Mobile Environment (WAVE) is the acronym for this concept. This protocol facilitates the deployment of 5.9 GHz-band Dedicated Short-Range Communications (DSRC).[6] Integrity attacks, confidentiality attacks, attacks, and authentication attacks are the five main types of VANET attacks.[7] In a VANET, IDS can be deployed both on the network segments and on the nodes themselves. While local IDSs work for the node on which they are deployed, global IDSs monitor the VANET segment where they are responsible for the group of cars in their segment and detect the segment. Local intrusion detection systems track all incoming and outgoing network traffic.[8]

A cyberattack is any unauthorized attempt to get information by breaking into a computer system or the internet and using that information for malicious purposes. Potential outcomes[9] include cyber terrorism, conflict, and threats. Because of its widespread availability, low price, and quick development, the internet plays a pivotal role in modern society. However, trends like remote work, inadequate and ineffective intrusion detection systems are broadening the scope of assaults alongside this rapid progress. Denial intruder assaults are just some of the cyberattacks that might happen.[10] Cybersecurity experts are responding by focusing their research on new and improved ways to prevent and detect assaults. Since attackers may now use a wide variety of sophisticated technological methods, cybersecurity has become an important subject of study in the previous decade.[11] As internet use grows and cybercrime becomes more common and damaging, the topic of cybersecurity has gained prominence throughout the world. Recent high-profile cybercrimes have shown how easily cyberattacks may spread internationally, wreaking havoc on enterprises, damaging critical data, stealing information, gaining unauthorized access, and so on.[12]

Machine learning (ML) has recently become crucial in cybersecurity[13] because of its rapid development and outstanding performance. Developing a reasonable ML model requires proper data, which means a significant quantity of data and sufficient sample categories. The Car Hacking dataset is widely used as a benchmark for intrusion detection systems. Various ML techniques, including generative adversarial networks (GANs), deep neural networks (NNs), and ensemble NNs, have been used to construct a wide variety of IDS models for VANETs.[14] Instead of addressing the inconsistencies in the datasets, the time difficulty, and the efficiency of the techniques, the IDS research for VANETs has concentrated on improving accuracy with a standardized dataset in general.[15] Since millions of packets flow across VANETs every day, relying on accuracy measures alone is not feasible. This is why even a 0.1% loss in accuracy can be problematic for the network.

Due to the specific difficulties and vulnerabilities of VANETs, it is essential that a more effective and efficient IDS be developed for these networks. VANETs allow cars and infrastructure to share information, improving management. Their dynamic and decentralized nature, however, renders them vulnerable to security concerns including data manipulation, malicious attacks, and unauthorized access. Existing VANET IDSs have difficulty meeting these challenges head-on.[16] They may have a high false positive rate, disrupting traffic unnecessarily, or they may be unable to keep up with the ever-changing patterns of sophisticated attacks. Therefore, it is crucial to improve IDSs for VANETs to guarantee the safety and dependability of vehicular communication.

The purpose of this research is to create a brand-new IDS that places a premium on accurate case classification and assault category categorization. Predicting whether data is normal or attack-related is done with the use of a classifier called self-attention based conditional VAEGAN, and DSFLA is used as a feature selection model. In order to learn attack characteristics from subsets of the original dataset, we present a method for properly preparing the training set.

RELATED WORK

To better categorize attack instances with little information, Dhar et al.[17] present a new IDS for VANETs that uses principal component analysis. In the first stage, the designed Cascaded ML framework distinguishes among attack and regular situations. In the second stage, the attack data are classified. The framework stresses the need not to place an assault in the "normal" category. Finally, the suggested framework is implemented using the most widely used ML model, an artificial neural network, and tested on the Car Hacking dataset. This research not only examines the efficacy of the suggested framework but also evaluates the efficiency of common categorization tasks. The suggested technique has been shown to be an efficient IDS, with experimental findings on the Car Hacking dataset showing that it outperforms the current state-of-the-art ML representations.

An expert system for preventing and detecting intrusions in Vehicular Ad Hoc Networks has been developed by Sontakke and Chopade.[18] The data nodes originate from many web resources. Once the node data has been collected, the autoencoder model may be used to extract useful information. The Beetle-Whale Swarm Optimization was used in the feature selection phase after feature extraction to choose the best features. When it comes to detecting network intrusion, a technique is used to make use of the selected correct attributes. To stop intruders, the Trust-based routing protocol uses the same B-WSO to pick the most efficient routing path around the malicious node. The suggested approach is evaluated experimentally using standard methods to ensure its efficacy.

Registration, key creation, data decoding are the five steps in the method suggested by Kumar Pulligilla et al.[19] Data packets are used for OBU authentication. The optimized model is then used to perform the intrusion detection. Log files are analyzed with Renyi entropy to choose characteristics to analyze. For intrusion detection, we use a neural network based on Rider's optimization algorithm (ROA) and Sea Lion Optimization (SLnO) called Rider-based Sea Lion Optimization (RBSLO). With improved accuracy of 92.5%, measures and the shortest calculation time of 135.654 s, the suggested RBSLO-based RideNN promises to enhance accomplishment.

Distributed federated learning (FL) networks are presented by Arya et al.[20] as a method for intrusion detection in smart cities. By employing the most effective method of intruder detection, it helps save time and materials. In the first step, cars create local IDS classifiers for VANET data streams using deep learning and a federated learning approach. When asked, these cars will communicate the classifiers they've learned locally, cutting down on the amount of data needed to be transmitted to other vehicles. Each vehicle then has a set of locally and remotely trained classifiers added to an ensemble of neural networks. At last, the global ensemble model is redistributed to regional nodes for their periodic refresh. Using attack detection accuracy, precision, recall, and F1 scores throughout a ToN-IoT data stream, the efficacy of the designed technique is assessed for intrusion finding in VANETs. The ID model achieves an accuracy of 0.99% in training and 0.981% in testing.

Improved intrusion detection for VANETs has been proposed by Amaouche et al.[21], which uses synthetic minority oversampling (SMOTE) to address the class imbalance problem and mutual information to choose the best characteristics on which to base an efficient model. We use Random Forest (RF) as our classifier and compare it to other machine learning (ML) methods. To ensure the model can handle missing values, imbalanced data, and categorical values, it is put through its paces on three datasets: CICIDS2017. When compared to other models, ours performed really well. At 100% on the 99.9% on both the NSL-KDD and CICIDS2017 datasets, it demonstrated excellent accuracy, precision, recall, and F1 score. The ROC analysis also showed our model's superior performance, with an AUC of 100%.

An IDS model was developed by Shams et al.[22] that can gather data from automobiles and Roadside Units (RSUs) on a network together. We used the popular Network Simulator 3 (ns-3) and simulation tools to create synthetic network data for training the core of our proposed IDS. We also created independent test data, which is not a subset of the training data. This guarantees that the erroneous performance result of an overfitted model is wiped out. We used a Network, to build a multi-class IDS. Using the gathered network traffic data, the CAFECNN model can identify both passive and aggressive assaults. The findings demonstrate that the suggested model outperforms techniques in spotting difficult-to-detect passive assaults. We also simulated the model in real time and found that network performance improved immediately, notably in terms of ratio and throughput, despite the intrusions.

Cui et al.[23] investigate federated learning on VANETs and create a CIDS model that is both effective and efficient. The technique makes use of local SDN collaboration to jointly train the CIDS model flows, making IDSs more scalable and global in scope. This research uses in a restricted multi-objective optimization framework with the goal of decreasing the model alteration between clients of the same SDN and increasing the detection accuracy. Pareto optimality is reached for fairness constraint maximization performance by the method's two-stage gradient optimization of a surrogate maximum function comprising all the goals. In addition, the training model is analyzed in this paper and compared to state-of-the-art techniques using two publicly available datasets. The experimental findings show that the suggested perfect is superior to the current methods since it protects the privacy of local data and shows excellent accuracy and efficiency in identifying assaults.

Research Gap

Anomaly detection systems are used extensively in disciplines like artificial detection, pattern recognition, and machine learning due to their ability to identify unexpected attacks. The feature extraction and feature selection processes used by conventional machine learning methods, which are widely used in IDS, are tedious and time-consuming. In addition, the prevailing classification technique relies on a very basic form of machine learning. In a practical network setting, shallow methods can reduce the detection rate by analysing high-dimensional inputs.

Finally, there are important differences between network traffic and host call sequences, the two primary types of data that IDS systems must process. The sequence of host calls is more akin to a sequence problem than it is to data from a network. The detection algorithms used in traditional approaches, notably those used in hybrid data source detection systems and state-of-the-art detection systems, are not flexible enough to adapt to new situations. This means that prior detection methods are useless.

DESIGNED SYSTEM

The suggested IDS is a hierarchical structure that uses a subclass division paradigm. Furthermore, a designed model has been

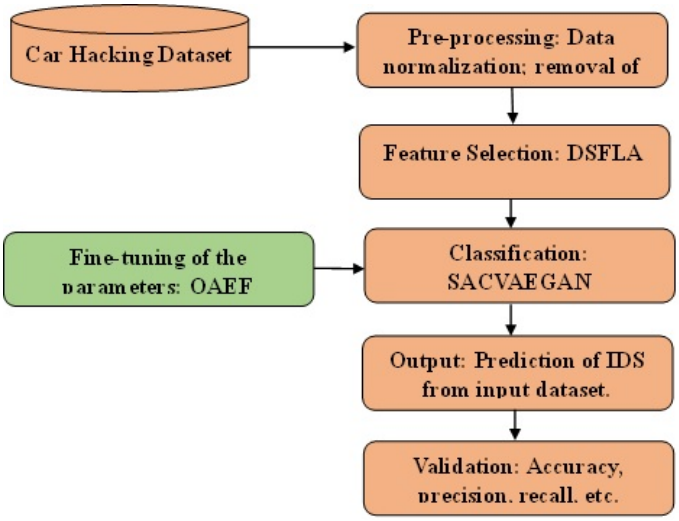


Figure 1: Workflow of the Work

reviewed to prove the effectiveness of this framework. Pioneering and more contemporary efforts have been compared to the suggested model to measure its performance. The following sections detail the dataset, the experimental design, the outcomes of the experiments, and a comparison of the results. The workflow of the designed model is given in Figure 1.

Dataset Description and Preprocessing

Eleven features/attributes and four forms of attacks are included in the Car Hacking dataset.[24] Modern VANET models, like CAN, include high levels of network connection. In order to compile the dataset, we first classified CAN traffic including malicious messages put into actual automobiles. Every 30–40 minutes of CAN traffic was disrupted for 3–5 seconds. The characteristics of Normal and attack classes are summarized in Table 1.

Table 1. Characteristics and Sum of Trials in the Car Hacking Dataset

Attack Names	Timestamp	CAN ID	Sum of Statistics Bytes	Data [0–7]	Output Class	Quantity of Samples
DoS	Every 0.3 milliseconds	Dominant value "0000"	0–8 (but altogether are 8)	Data degree	R	587,521
Normal	--	Chance hex value	0–8 (but altogether are 8)	Data degree	T	988,872
Spoofing	Every 1 millisecond	Chance hex value	0–8 (but altogether are 8)	Data degree	R	654,897
Gear attack	Every 1 millisecond	Chance hex value	0–8 (but altogether are 8)	Data degree	R	597,252

For each category in the dataset, there are 5.csv files. The Normal class's.csv file contains just Normal class data, whereas the attack class's.csv files contain data for both the attack class (labelled "R") and the Normal class (labelled "T"). Data for Normal, DoS, Gear, RPM, and Fuzzy totaled 988,872, 587,521, 597,252, 654,897, and 491,847 samples, respectively, after preprocessing. This was achieved by taking the entire.csv file containing Normal data and only the "R" labelled data from the remaining four attack.csv files. There were only numeric values for attributes. Scaling was done on numeric properties to make them fit within the range of 0–1. One-hot encoding was utilised for the transformation of output levels in both the main and secondary class models. In this method, the obtainable samples were divided into training and test sets to guarantee fair datasets. For classes with large sample sizes, about 80% of the data was used as a training set, while 20% was set aside as test samples. The diversity of attacks within the dataset was preserved, as can be seen in the next description of the confusion matrix.

Specifically, Data [0-7] in the 8-byte data representation. For examples of the DoS, Gear, and RPM classes, these 8 bytes are always assigned to the same value. Variations may be found in both Fuzzy and Normal data samples, with minimum and maximum values of 0 and 255, respectively. All 8-byte characteristics in the Fuzzy data tend to average around 127, but in the Normal data they tend to average out to different numbers.

According to the data, classes fixed values in the 8-byte characteristics, suggesting that they are free of noise and variance. Neither the Fuzzy nor the Normal data are free from noise, although the allowed range of values (0–255) is rather large. Comparatively, the fluctuating averages in Normal data may show noise, whereas the stable average of 127 across all characteristics in Fuzzy data implies coherence. The deviations appear to fall within expected thresholds and display patterns rather than being completely at random, giving the impression that the dataset is rather clean at first glance.

Data Preprocessing

Data preprocessing is a crucial step in preparing raw data for analysis or machine learning tasks. It involves several steps aimed at cleaning, transforming, and organizing the data to make it suitable for further analysis or modeling. The dataset is making it more suitable for a classifier.

Removal of Socket Material

To prevent overfitting training towards socket information, it is required to remove the IP addresses of the source and dataset. The classifier should be trained on the properties of the packets themselves, rather than the socket data, to ensure that any host with matching packet characteristics is blocked.

Remove White Spaces

White space can be used when making labels for many classes. These blanks cause new classes to form since the actual value is the same class.

Label Encoding

The dataset's multi-class labels, which contain the names of assaults, are all labelled with string values. It is required to numerically encode these values to train the classifier as to which class each tuple belongs to. Since the zero-one establishment necessary for this operation, the multi-class labels are utilised instead.

Data Normalization

Training the classifier might be difficult due to the dataset's high degree of numerical variation. This implies that zero and one should be used for the values of each feature. The classifier benefits from the more consistent values, and the relevance of each attribute's values is preserved.

Feature Selection using DSFLA

Introduction to SFLA

In SFLA, each potential answer corresponds to the location of a digital frog, and the population of answers is represented by a collection of these frogs. Once the starting populace P has been generated, the subsequent steps—are continued indefinitely or until some limiting disorder is reached.

$$x'_w = x_w + rand \times (x_b - x_w) \quad (1)$$

$$x'_w = x_w + rand \times (x_g - x_w) \quad (2)$$

where rand is a accidental amount subsequent in [0.1].

By mixing up the order of all the developed memeplexes, a new population P is built. As mentioned above, the search technique and

parameters used to generate memeplexes are generally the same,[25] and the concept of differentiated search within memeplexes is seldom taken into account. Introducing unique search operators and parameters strengthens the search capability and allows for the efficient avoidance of local optima, considerably enhancing the search efficiency. In this research, DSFLA is introduced as a means of selecting relevant characteristics from raw data. The diversified search is incorporated into DSFLA's second phase.

Initialization, Population Separation, and the First Stage

In this study, a solution of the problematic is characterized as $[M_{\theta_1}, M_{\theta_2}, \ldots., M_{\theta_n}]$ and a string $[q_1, q_2, \cdots, q_n]$, where M_{θ_j} is the allocated features for job $J_j, j = 1, 2, \cdots, n$, and q_{l} is agrees to J_{l}. Each of these two strings functions separately. What follows is a breakdown of the decoding procedure. Each job's machine is selected first using the machine Mk, all jobs are run simultaneously. $J_i, J_{i+1}, \cdots, J_j$ allocated on M_{k}—that is, $M_{\theta_i} = M_{\theta_{i+1}}, \cdots, = M_{\theta_j} = M_k$. The dispensation order of $q_{l,}\, l \in [i, j], i < j$, and M_{k} sequentially.

Following the random generation of the preliminary population P, the population is divided as follows. Select the top s solutions from set P and arrange them from best to worst according to their effectiveness. Then, the memeplexes are given a portion of the original response to work with. We'll call the first answer M_1, the second M_2, and so on. Assigning other solutions to memeplexes is then done using binary tournament selection, where two solutions are chosen at random and compared to see which is superior. Then, we incorporate x_i (x_j) into M_1. If there are multiple keys with the same goal, then randomly select one of them and include it into M_1. Unselected options are returned to population P. The same procedure for settling on a solution for $\mathcal{M}_2, \mathcal{M}_3, \ldots., \mathcal{M}_s$ and then recurrence the above way pending all keys are allocated. Obviously, $N = s \times \theta$, where θ symbolizes memeplex.

Global search is only employed in the initial stage because of its superior exploratory capabilities. Differentiated search procedures based on memeplex quality assessments are employed in the second stage.

The Second Phase

In SFLA, assessing memeplex quality is rarely taken into account. The quality of memeplexes is measured by how well they solve problems and how well they evolve. Meant Memeplexly $\mathcal{M}_l$, its quality M_{eql} is defined by

$$M_{\mathrm{eql}} = a_1 \times \frac{msq_{max} - ms_{q1}}{msq_{max} - msq_{min}} + a_2 \times \frac{mvq_1 - mvq_{min}}{mvq_{max} - mvq_{min}} \quad (3)$$

where a_1, a_2 are real number, msq_{l} and mvq_{l} indicate solution quality of $\mathcal{M}_l$, respectively, $msq_{max} = \max_{l = 1,2,\cdots,s} \{msq_l\}, msq_{min} = \min_{l = 1,2,\cdots,s} \{msq_l\}$, mvq_{max} and mvq_{min} represent all memeplexes, separately.

After entirely solutions in $\mathcal{M}_l$ are organized in the make span, let H_1 indicate primary $\theta/2$ solutions except x_{b} and H_2 is the set of the endured $\theta/2$ keys in $\mathcal{M}_l$,

$$msq_l = C_{max}(x_b) + \beta_1 \times \bar{C}_{max}(H_1) + \beta_2 \times \bar{C}_{max}(H_2) \quad (4)$$

where $\bar{C}_{max}(H_i)$ is the regular makespan of all keys in H_{i}, i = 1, 2, $\beta_i . i = 1, 2$ is a real number. Solutions of H_1 are better than those of H_2; consequently, we set $\beta_1 > \beta_2$ to reflect this feature. $\beta_1 = 0.4$ and $\beta_2 = 0.1$ are gotten by trials.

Let Im_{x} designate the improved sum of x group. When $x \in M_l$ is selected x_{w}, in general SFLA, if than x, then $Im_x = Im_x + 1. Se_{\mathrm{x}}$ is the total primary generation.

$$mvq_1 = \sum_{x \in M_l} Im_x / \sum_{x \in M_l} Se_{\mathrm{x}} \quad (5)$$

For solution x_{i}, its act_{x_i} is used to assess is figured by

$$act_{x_i} = Im_{x_i} / Se_{x_i} \quad (6)$$

The second phase is exposed as shadows.

(1) Perform populace separation, calculate Meq_{l} for completely in descending order of Meq_{l}, and construct set $\Theta = \{M_l | meq_l > \overline{Meq}, l \leq \eta \times \theta\}$.

(2) For correspondingly memeplex $M_l, M_l \notin \Theta$, recurrence the subsequent steps R_1 aeras if $|\tau| > 0$,
execute global search among xb and chosen y 2 T ; else achieve global
search among x_{b} and a key $y \mid \in M_l$ with $act_y \geq act_x$ for all $x \in M_l$.

(3) For each memeplex $M_l \in \Theta$,
   1. sort all keys in M_l in the suppose $C_{max}(x_1) \leq C_{max}(x_2) \leq \cdots \leq C_{max}(x_\theta)$, and hypothesis a set $\varphi = \{x_i | dist_{x_i} < \overline{dist}, i \leq \theta/2\}$.
   2. Recurrence the subsequent ladders R_2 times, key $x_i \in M_l/\varphi$ if $act_{x_i} > 0.5$, then select a solution $y \in \varphi$ by roulette assortment based on Pr_{y}, execute global search among x_{i} and y, and inform memory T ; else search among x_{i} and a solution z with $act_z \geq act_{x_i}$ for all $x_i \in M_l$ and T .

(4) Execute hunts on each key $x \in \varphi$.

(5) Perform novel populace shuffling.

where $dist_{x_i} = |C_{max}(x_i) - C_{max}(x_b)|$ is distinct for each key $x_i \in M_l$ and $\overline{dist}$ is the
regular value of all $dist_{\mathrm{x_i}}$ in M_l. η is a real sum and set to be 0.4 by trials, $\overline{Meq}$
designates the average excellence, Θ is the set of Pr_{y}
is a probability and distinct by

$$Pr_{\mathrm{y}} = \frac{|\varphi| - rank_y}{|\varphi|} \times \frac{Im_y}{\sum_{x \in \varphi} Im_x} \quad (7)$$

where $rank_y$ is an numeral and obvious by ranking rendering to initial phase of step (3) in the above Procedure.

In the second phase, after altogether in the descendant order of Meql,
suppose $Meq_1 \geq Meq_2 \geq \cdots Meq_s$.

Memory T is used to store keys. The maximum extent $|T|_{\mathrm{max}}$ is given in early payment. We set $|T|_{\mathrm{max}}$ to be 200 by trials. When the sum of keys exceeds $|T|_{\mathrm{max}}$, a key x can be supplementary into better than one.

Six neighborhood constructions are used. N_1 is exposed below. Arbitrarily first-class a job from the machine M_{k} with the largest C^k_{max} and machine M_{g} with the smallest C^g_{max}, where C^k_{max} and C^g_{max} are last treated job on M_{k} and M_{g}, individually. N_2 is achieved in the subsequent way. Decide on a machine M_{k} with the major C^k_{max}

and a job J_i with the major processing time p_{ki} on M_k, arbitrarily pick a machine M_g, g≠k and a job J_j with the largest p_{gj} and conversation J_i and J_j among M_k and Mg.

N_3 is described as shadows. Arbitrarily choice two machines M_k and M_g and talk a job J_i with the largest p_{ki} and a job J_j with the major pgj among these two machines. N_1, N_2, N_3 only act on the string.

N_4, N_5, N_6 are operations on a string whereby two genes are exchanged, one gene is inserted into a new location that is also decided at random, and the genes are inverted amid two positions $k_1, k_2, k_1 < k_2$.

Multiple key x, let $u = 1$, recurrence the subsequent ladders V times: yield a key $z \in N_u(x)$, u=u+1, let $u = 1$ if $u = 7$, and $Im_x = Im_x + 1$.

The procedure for the second phase of the global search is identical to the first.

Using the s developed memeplexes, the current SFLA[25] builds a new population P. In this research, we reshuffle the population in the following ways: The most successful memeplexes from both the original population (T) and the new population (P) are included into the new population. Using experimental methods, we establish γ = 0.1 ×$|T|_{max}$.

In other words, memeplex search or shuffling can be used to enhance some of P's worse solutions.

A global search of optimisation object x is applied in accordance with act_x, and then manifold neighbourhood search is performed on the keys in to find a good memeplex, which is the focus of the second phase. For other memeplexes, just a global search is performed; additionally, several parameters, $R_1, R_2, R_1 \neq R_2$, are used, and, as a consequence, distinguished search is applied.

Algorithm Explanation

The comprehensive stepladders of DSFLA are exposed underneath.

1. Initiation, 1. Let T start out empty and randomly generate N solutions for P.
2. Divvying up the people, number two. carry out the search procedure inside of every memeplex.
3. Three, reorganise the people.
4. (If the first phase's termination condition is not fulfilled, proceed to the second stage.
5. Carry on with the second stage until the termination disorder is reached.

The computational difficulty is $O(N \times R_1 \times L)$, where L is the recurrent sum of phases 2–3.

DSFLA differs from the original SFLA in the ways listed below. (1) Memeplexes are sorted into two groups, good and other, based on an evaluation of their quality that takes into account both their solution quality and their evolution quality. (2) The distinguished search is put into action by employing various search algorithms and limits for two types of memeplexes, which increases exploration capabilities and drastically reduces the likelihood of settling into local optimums.

Classification using Designed Methodology

Here, we provide an overview of the SA-CVAE procedure that has been proposed. The network architecture is composed of a discriminator, a value-added extractor, and a classifier. If the input tasters are genuine or virtual, the discriminator's module can tell you. The encoder and generator make up the VAE module. To create synthetic data, the generator takes actual samples and utilises the encoder's latent vectors in combination with random latent vectors. Input actual and synthetic samples are sorted by the classifier module.

Discriminator

It is the discriminator's module, as seen in Figure 2. There are four convolutional layers in total. The kernel size is 3 3 for each layer. The self-attention layer is used after the initial two convolution layers have already been applied. The information is transformed into a feature vector after the last layer. Reference[26] suggests that the model's stability can be improved by feeding the label information into the discriminator. The label is transformed into a vector and then appended to the feature vector through a complete connection layer. Then, the dimension is shrunk by applying a complete connecting layer. The function is then used to the data to verify their veracity.

Here is the loss discriminator D:

$$L_D = L_{D_{GP}} + L_{D_{G(z_{random}|y)}} \quad (8)$$

Loss in WGAN-GP among x and G(z|y) can improve perfect stability, and loss in the discriminator determines whether a feature is retained or dropped. $G(z_{random}|y)$ is false.

Among them:

$$L_{D_{GP}} = E_{z\sim P_z}[D(G(z|y)|y)] - E_{x\sim P(x_{real})}[D(x|y)] + \lambda E_{z\sim P_z}\left[\left(\left\|\nabla_{G(z|y)} D(G(z|y)|y)\right\|_2 - 1\right)^2\right] \quad (9)$$

$$L_{D_{xz_{random}}} = -E_{z\sim P_{z_{random}}}[logD(G(z_{random}|y)|y)] \quad (10)$$

where z characterizes the latent vector produced by the encoder, z_{random} characterizes the arbitrarily, x_{real} embodies real tasters, G(z|y) characterizes the produced by the generator rendering consistent label, $G(z_{random}|y)$ characterizes the virtual taster produced by the producer rendering to z_{random} and the consistent label, and y characterizes the tag.

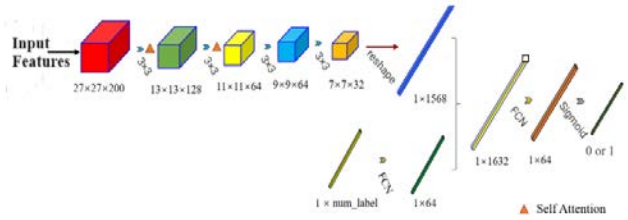


Figure 2 The assembly of the discriminator D.

Variational Auto-Encoder

The encoder and generator are the two main components of the VAE module. After the genuine samples have been encoded into generator G may utilise that information to create a synthetic sample. To vector m and the covariance e of the space, the encoder E is split into extraction networks. The network architecture is the

same for all feature extraction networks. It is made up of networks for extracting features both in space and in time. The network for extracting spectral features has four 1-dimensional convolution layers, each with a 5 by 1 kernel. Both the initial and secondary levels incorporate self-awareness. There are four 2-D convolution layers in the spatial feature network, each with in the first two layers. Following a network for extracting both spectral and spatial features, we combine the two into a single unified feature and apply a complete connection layer for dimensionality reduction. Using the vector e, we can calculate the latent vector l using the following equation.:

$$z = \mu + r * \exp(\epsilon) \quad (11)$$

The purpose of generator G is to simulate data distributions based on learned models. Specifically, G is made up of layers, and a final hidden layer. The latent vector and its associated label are first obtained, and then the vector is reshaped using two complete connection layers. The vector is then transformed into a cube of three-dimensional data and sent along to the transposed convolution layers. The transposed convolution layer has a kernel size of 3 x 3. At long last, a digital sample is within reach.

The loss function of VAE is as shadows:

$$L_{VAE} = L_{kl} + L_G + L_{vae_D} + L_{vae_C} \quad (12)$$

KL divergence is the initial term, and it is used to close the gap between the observed and expected latent vector distributions. Renovation loss among x and G(z|y) is l_2, thus that's the second term. The third term is the total of the loss in determining whether or not G(z_random |y)|y is true, as well as the loss in matching among x and G(z|y) in the discriminator D. The last term is the addition of the loss from the classification result of classifier C and the loss among x and G(z|y)..

Among them:

$$L_{kl} = \frac{1}{2}(\mu^T\mu + sum(\exp(\epsilon) - \epsilon - 1)) \quad (13)$$

$$L_G = \frac{1}{2}(\|x - G(z|y)\|_2^2) \quad (14)$$

$$L_{vae_D} = \|f_D(x) - f_D(G(z|y))\|_2^2 + E[\log(1 - D(G(z_{random}|y)))] \quad (15)$$

$$L_{vae_C} = \|f_C(x) - f_C(G(z|y))\|_2^2 - E[\log P(y|G(z_{random}|y)))] \quad (16)$$

where μ and ϵ characterize encoder, respectively; f_D characterizes the features discriminator D; f_C represents the topographies of samples is embodied as x_{real}; the rendering to the latent vector produced by the encoder is $G(z|y)$; the produced vector is $G(z_{random}|y)$; and y characterizes the label.

Classifier

The outputs of classifier C are used for this purpose. Also, spectral-spatial feature extraction networks are part of classifier C. Five 1 layers with a 1 5 kernel make up network, while five 2-D layers with a 3 3 kernel make up the spatial chin extraction network. In the end, we combine the spectral and spatial information and feed them into two complete connection layers. Here is how the LC loss function is calculated: The unit exposed in Figure 3.

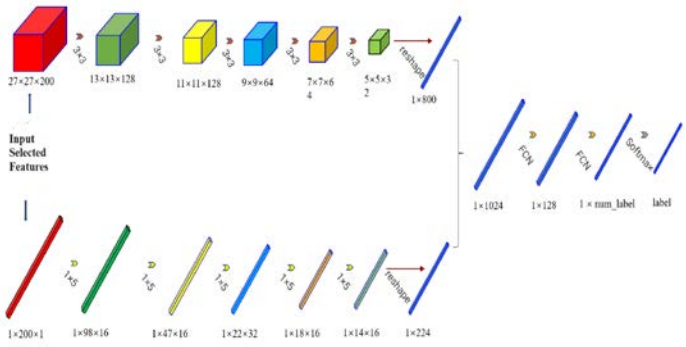


Figure 3: Structure of classifier

$$L_C = L_{C_{x_{real}}} + \lambda_1 L_{C_{x_z}} + \lambda_1 L_{C_{x_{z_{random}}}} \quad (17)$$

where the first term represents a reduction in the classification of x's outcome. The second tenure is the total loss from matching features among x and G(z|y). The final word represents a failure to preserve the outcome of classifying G(zrandom|y). the relative strengths of l1 and l2 in $L_{C_{x_z}}$ and $L_{C_{x_{z_{random}}}}$ loss, separately.

$$L_{C_{x_{real}}} = -E[logP(c|x_{real})] \quad (18)$$

$$L_{C_{x_z}} = \|f_C(x_{real}) - f_C(x_z)\|_2^2 \quad (19)$$

$$L_{C_{z_{random}}} = -E\left[logP(y|x_{z_{random}})\right] \quad (20)$$

where f_C represents the features of classifier, x_{real} characterizes the real tasters, x_z characterizes the virtual dormant vector z produced by, $x_{z_{random}}$ characterizes the virtual taster generated by entering the arbitrarily vector z_{random} into the producer, and y embodies the label. The parameters are tuned by using improved AEF algorithm that is described as follows.

Hyper-parameter tuning using Opposition-based optimization approach

Slow finest key are problems for the fundamental artificial electric field. There may be superior alternatives that are far from the present solution, however upgrading certain solutions towards the local best solution causes these drawbacks. By looking at other approaches, the OAEF sidesteps these problems. When the search capabilities of the normal version of AEF are combined with those of OBL, the search space may be explored more efficiently. Since the addition of OBL has no effect on the AEF setup, the suggested method requires less parameters to be set, and its best solution is more accurate when compared to other methods. Since OAEF can search a larger space, a smaller initial population may be used, which improves optimum solution convergence.

The suggested technique improves AEF in two phases. First, OBL is used to seed the population, which searches the whole search space for solutions to boost convergence speed and avoid becoming stuck on the local best one. Second, it's put to use when determining if a solution that goes in the other way improves upon the present population solution. Both procedures are elaborated about below.

A). Initialization stage

A random populace of X (of size N) is first generated so that the site vector of the preliminary solution may be written as $X_i = [X_i^1, X_i^2, \ldots, X_i^D]$where $i = 1,2, \ldots, N$ and $d = 1,2, \ldots, D$. After that,

OBL computes the conflicting populace of X is produced. The best N sum of keys is designated based on X and $\bar{X}$.

- Keys of X populace are arbitrarily prepared.
- The opposite populace $\bar{X}(\bar{X}_i^d = u^d + l^d - X_i^d, i = 1,2,..,N$ and d=1,2,..,D) is intended l is search space, X_i^d and $\bar{X}_i^d$, denote the locations of the d-th and i-th parameters of the i-th and x-th solutions, respectively, in populace X.
- The top N answers derived from all of these variables are: $X \cup \bar{X}$ are selected to make a new populace.

B). Updating stage

The optimal solution X_best is found after selecting the top N solutions from a fresh population. In order to determine the fitness functions of the updated solutions in the X population, AEF is applied. Additionally, fitness functions for the OBL-obtained X population and its counterpart are determined.

For the purpose of evaluating OAEF's efficacy, we used the Sphere, Ackley, Egg Crate, and Easom benchmark functions. The definitions of adopted test functions are summarised in Table 2.[27] We settled on 50 for the population size and 500 for the maximum sum of iterations. Twenty iterations of the OAEF procedure proposal were tested.

Table 2: Approximately benchmark purposes.

Test Function	Function Description	Dimension	Range	Global Optimum
Sphere	$F_1(X) = \sum_{i=1}^{D} X_i^2$	D=30	-100≤ x_i ≤ 100	0
Rosenbrock	$F_2(X) = \sum_{i=1}^{D-1} [100(X_{i+1} - X_i^2) + (X_i - 1)^2]$	D=30	-30≤ x_i ≤ 30	0
Schwefel	$F_3(X) = -\sum_{i=1}^{D} X_i sin\left(\sqrt{\lvert X_i \rvert}\right)$	D=30	-50≤ x_i ≤ 500	-12569.487
Ackley	$F_4(X) = X_1^2 + X_2^2 + 25[sin^2(x_1) + sin^2(x_2)]$	D=30	-32≤ x_i ≤ 32	0
Egg Crate	$F_5(X) = X_1^2 + X_2^2 + 25[sin^2(x_1) + sin^2(x_2)]$	D=2	-5≤ x_i ≤ 5	0
Easom	$F_6(X) = -\cos(x_1)\cos(x_2)exp(-(x_1 - \pi)^2 - (x_2 - \pi)^2)$	D=2	-100≤ x_i ≤ 100	-1

RESULTS AND DISCUSSION

During the trial, Python and the Anaconda integrated development environment were utilized. The experimental computer used the following specifications: NVIDIA GeForce GTX 1070Ti, 8 GB GPU, with Intel® CoreTM i5-7400 CPU, 3.50 GHz. Each model was trained for 300 iterations using the suggested optimizer.

Performance Metrics

For the determination of measuring the efficacy of transfer learning with learning, we employed the following metrics to assess the CNN models used in the process:

$$accuracy = \frac{TP+TN}{TP+FP+TN+FN} \quad (21)$$

$$Precision = \frac{TP}{TP+FP} \quad (22)$$

$$recall = \frac{TP}{TP+FN} \quad (23)$$

$$Specificity = \frac{TN}{TN+FP} \quad (24)$$

$$negative\ predictive\ value = \frac{TN}{TN+FN} \quad (25)$$

$$F_1 score = 2\frac{precision \times recall}{precision+recall} \quad (26)$$

$False\ positives\ (FP)\ and\ false\ negatives\ (FN)$ are the inverse of the $true\ positives\ (TP)$ and true negatives (TN). The AUC and ROC curve were also determined.

Table 3: Comparative analysis of Feature Selection

Metrics/Models	SFLA	GWO	ACO	DSFLA
Recall (%)	93.98	87.86	89.33	96.24
F-measure (%)	93.79	86.55	88.74	95.91
Accuracy (%)	93.88	88.35	90.61	96.72
Precision (%)	93.97	85.27	87.92	95.85

In above Table 3 characterise that the Comparative investigation of Feature Selection. In the analysis of Accuracy (%) of GWO model attained as 88.35 and ACO model attained the 90.61and SFLA model attained the value as 93.88 and lastly DFLA model attained the value as 96.72 correspondingly. Then the Precision (%) of GWO model attained as 85.27 and ACO model attained the 87.92 and SFLA model attained the value as 93.97 and lastly DFLA model attained the value as 95.85 correspondingly. Then the Recall

(%) of GWO model attained as 87.86 and ACO model attained the 89.33 and SFLA model attained the value as 93.98 and lastly DFLA model attained the value as 96.24 correspondingly. Then the F-measure (%) of GWO model attained as 86.55 and ACO model attained the 88.74 and SFLA model attained the value as 93.79 and lastly DFLA model attained the value as 95.91 correspondingly. Figure 4 presents the visual analysis of the designed model.

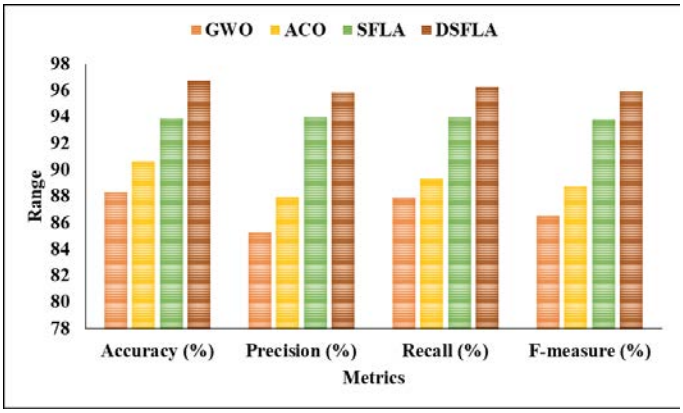


Figure 4: Graphical Description of different feature selection techniques

Table 4: Analysis of Various Classifiers on K=10

Models	F1 Score	Precision	Recall	Specificity	mpv	Accuracy	AUC Score
AE	0.8475	0.8929	0.8065	0.8824	0.7895	0.8407	0.9064
VAE	0.8475	0.8929	0.8065	0.8824	0.7895	0.8407	0.9089
SVM	0.7895	0.8654	0.7258	0.8627	0.7213	0.7876	0.9089
DBN	0.8500	0.8793	0.8226	0.8627	0.8000	0.8407	0.9203
SACVAEGAN	0.8281	0.8030	0.8548	0.7451	0.8085	0.8053	0.8786

In above Table 4 characterise that the Investigation of Various Classifiers on K=10. In the analysis of SVM model accomplished the accuracy obtained as 0.7876 then precision as 0.8654 besides the recall range of 0.7258 besides the specificity as 0.8627 and the mpy range of 0.7213 besides F1-score as 0.7895 and to conclude the AUC as 0.9089 congruently. Then the DBN model accomplished the accuracy of 0.8407 and precision accomplished as 0.8793 besides the recall range of 0.8226 and the specificity as 0.8627 and the mpy range of 0.8000 and F1-score as 0.8500 and lastly the AUC slash as 0.9203 correspondingly. Then the AE model attained the accuracy obtained as 0.8407 and precision as 0.8929 and the recall range of 0.8065 and the specificity as 0.8824 and lastly the AUC score as 0.7895 and the mpy range of 0.8475 and F1-score as 0.9064 congruently. Then the VAE model attained as 0.8929 besides the recall range of 0.8065 and the specificity as 0.8824 and lastly the AUC score as 0.7895 and the mpy range of 0.8475 and F1-score as 0.9089 correspondingly. Then the SACVAEGAN prototypical accomplished the accuracy of 0.8053 besides precision as 0.8030 then the recall range of 0.8548 then the specificity as 0.7451 formerly the mpy range of 0.8085 besides F1-score obtained as 0.8281 and lastly the AUC groove as 0.8786 consistently, where it is graphically seen in Figure 5.

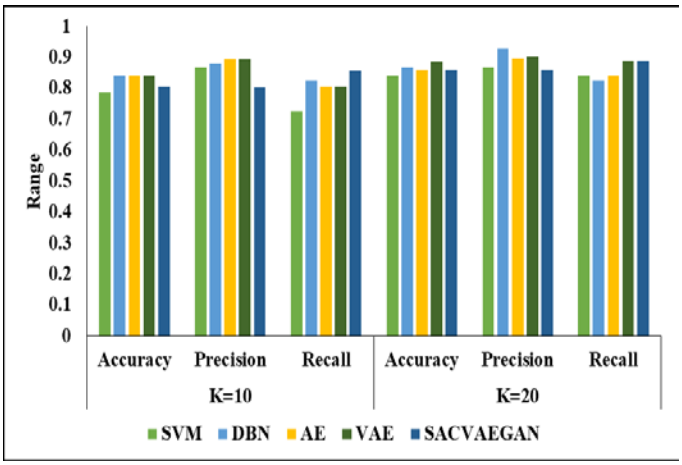


Figure 5: Graphical Description of different models on two K-fold analysis

Table 5: Comparison of different models on K=20

Models	Precision	Recall	Specificity	mpv	F1 Score	AUC Score	Accuracy
AE	0.8966	0.8387	0.8824	0.8182	0.8667	0.9330	0.8584
VAE	0.9016	0.8871	0.8824	0.8654	0.8943	0.9374	0.8850
SVM	0.8667	0.8387	0.8431	0.8113	0.8525	0.9190	0.8407
DBN	0.9273	0.8226	0.9216	0.8103	0.8718	0.9219	0.8673
SACVAEGAN	0.8594	0.8871	0.8235	0.8571	0.8730	0.9282	0.8584

Table 5 describes the comparison of various models on a K=20 basis. In the analysis of the SVM model, the accuracy was 0.8407, precision was 0.8667, recall range was 0.8387, specificity was 0.8431, mpy range was 0.8113, F1-score was 0.8525, and lastly the AUC score was 0.9190. The DBN model then achieved the corresponding accuracy of 0.8673, precision of 0.9273, specificity of 0.8226, mpy range of 0.9216, 0.8103, and 8718, and AUC score of 0.9219. The AE model then achieved the following results in that order: accuracy obtained as 0.8584, precision of 0.8966, recall range of 0.8387, specificity of 0.8824, mpy range of 0.8182, F1-score of 0.8667, and lastly the AUC notch of 0.9330. The VAE prototypical then achieved accuracy of 0.8850, precision accomplished as 0.9016, recall range of 0.8871, specificity of 0.8824, mpy range of 0.8654, F1-score obtained as 0.8943, and lastly an AUC score of 0.9374, all in that order. The SACVAEGAN model then obtained the corresponding accuracy of 0.8584, precision accomplished as 0.8594, recall range of 0.8871, specificity of 0.8235, mpy range of 0.8571, F1-score obtained as

0.8730, and lastly AUC notch of 0.9282, where Figure 6 shows the comparison of two different K values of designed model with existing techniques.

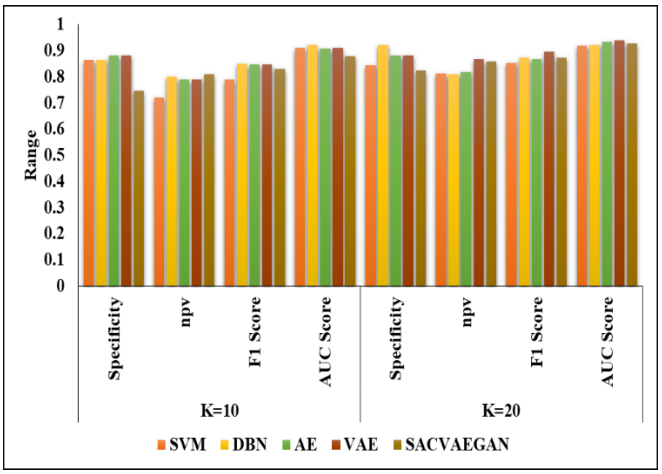


Figure 6: Analysis of various classifiers in terms of different metrics

Table 6: Validation Analysis of different Classifiers

Models	Accuracy	Train Time(s)	Test Time(s)
SVM	93.00%	6.503	0.01505
DBN	95.00%	10.083	0.01506
AE	90.70%	1.632	0.01512
VAE	82.20%	2.207	0.01702
SACVAEGAN	98.67%	16.146	0.01553

In above Table 6 signifies that the Validation Investigation of different Classifiers. In the investigation of SVM model attained accuracy rate as 93.00% and train time as 6.503 and test time as 0.01505 correspondingly. Then the DBN model attained accuracy rate as 95.00% and train time as 10.083 and test time as 0.01506 correspondingly. Then the AE model attained accuracy rate as 90.70% and train time as 1.632 and test time as 0.01512 correspondingly. Then the VAE model attained accuracy rate as 82.20% and train time as 2.207 and test time as 0.01702 correspondingly. Then the SACVAEGAN model attained accuracy rate as 98.67% and train time as 16.146 and test time as 0.01553 correspondingly. Study A utilized a qualitative approach to explore the impact of social media on adolescent mental health. Through interviews and thematic analysis, it identified key stressors and coping mechanisms.

CONCLUSION

This work presents a deep learning-based IDS for VANETs, efficiently dividing assaults into subclasses without incorrectly labeling any attacks as belonging to the Normal class. In this study, we introduce a SACVAEGAN for classifying VANET IDS. To further improve performance, we employ CVAEGAN, capable of producing more high-quality training data. Additionally, our suggested SACVAEGAN utilizes the self-attention mechanism to enhance feature extraction. The entire training procedure is stabilized by employing a unique loss function. On input datasets, SACVAEGAN outperformed state-of-the-art tactics, including GAN-based algorithms, in terms of classification accuracy. The suggested approach achieves the required outcomes, as shown by experiments conducted on the Car Hacking dataset. This finding opens up a number of avenues for further investigation. To begin, the suggested framework relies heavily on NNs, but alternative methods, such as the game theoretic approach, could achieve better results. Adopting this IDS for use in a real-time situation in a safety scheme is another potential direction for future development. The lack of a robust and endangered scheme in VANETs makes it interesting that the concept of statistics increases and cascaded outlines will be appropriate for similar situation belongings like handling security issues and preventing an attacker from entering the system.

FUTURE SCOPE

Future research on the deep learning-based IDS for VANETs can explore alternative methods like game theory, implement real-time applications through edge computing, and enhance robustness against adversarial attacks. Additionally, applying this IDS framework to other networks, such as IoT, and collaborating with industry for real-world deployment and standardization are promising directions.

CONFLICT OF INTEREST STATEMENT

Authors do not have any confict of interest, financial, academic or otherwise, for publication of this work in public domain.

REFERENCES

1. H. Bangui, M. Ge, B. Buhnova. A hybrid machine learning model for intrusion detection in VANET. *Computing* **2022**, 104 (3), 503–531.
2. A. Haydari, Y. Yilmaz. RSU-Based Online Intrusion Detection and Mitigation for VANET. *Sensors* **2022**, 22 (19), 7612.
3. I. Naqvi, A. Chaudhary, A. Kumar. A Systematic Review of the Intrusion Detection Techniques in VANETS. *TEM J.* **2022**, 11 (2), 900–907.
4. H. Bangui, M. Ge, B. Buhnova. A hybrid machine learning model for intrusion detection in VANET. *Computing* **2022**, 104 (3), 503–531.
5. H. Bangui, B. Buhnova. Recent advances in machine-learning driven intrusion detection in transportation: Survey. *Procedia Comput. Sci.* **2021**, 184, 877–886.
6. B. Karthiga, D. Durairaj, N. Nawaz, et al. Intelligent Intrusion Detection System for VANET Using Machine Learning and Deep Learning Approaches. *Wirel. Commun. Mob. Comput.* **2022**, 2022.
7. I. Naqvi, A. Chaudhary, A. Rana. Intrusion Detection in VANETs. In *2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions), ICRITO 2021*; IEEE, **2021**; pp 1–5.
8. A. Thirumalraj, R.J. Anandhi, V. Revathi, S. Stephe. Supply chain management using fermatean fuzzy-based decision making with ISSOA. In *Convergence of Industry 4.0 and Supply Chain Sustainability*; **2024**; pp 296–318.
9. F. Gonçalves, J. Macedo, A. Santos. Evaluation of VANET datasets in context of an intrusion detection system. In *2021 29th International Conference on Software, Telecommunications and Computer Networks, SoftCOM 2021*; IEEE, **2021**; pp 1–6.
10. A.R. Gad, A.A. Nashat, T.M. Barkat. Intrusion Detection System Using Machine Learning for Vehicular Ad Hoc Networks Based on ToN-IoT Dataset. *IEEE Access* **2021**, 9, 142206–142217.

11. Y. Yu, X. Zeng, X. Xue, J. Ma. LSTM-Based Intrusion Detection System for VANETs: A Time Series Classification Approach to False Message Detection. *IEEE Trans. Intell. Transp. Syst.* **2022**, 23 (12), 23906–23918.
12. M. Zang, Y. Yan. Machine Learning-Based Intrusion Detection System for Big Data Analytics in VANET. In *IEEE Vehicular Technology Conference*; IEEE, **2021**; Vol. 2021-April, pp 1–5.
13. A. Alsarhan, M. Alauthman, E. Alshdaifat, A.R. Al-Ghuwairi, A. Al-Dubai. Machine Learning-driven optimization for SVM-based intrusion detection system in vehicular ad hoc networks. *J. Ambient Intell. Humaniz. Comput.* **2023**, 14 (5), 6113–6122.
14. N. Ben Rabah, H. Idoudi. A Machine Learning Framework for Intrusion Detection in VANET Communications. In *Emerging Trends in Cybersecurity Applications*; Springer International Publishing, Cham, **2022**; pp 209–227.
15. S.N. Ohatkar. Heuristics for optimizing minimum interference channel allocation problem in cellular networks. *J. Integr. Sci. Technol.* **2024**, 12 (4 SE-Computer Sciences and Mathematics), 789.
16. G. Singh, N. Khare. A survey of intrusion detection from the perspective of intrusion datasets and machine learning techniques. *Int. J. Comput. Appl.* **2022**, 44 (7), 659–669.
17. A.C. Dhar, A. Roy, M.A.H. Akhand, M.A.S. Kamal. CascadMLIDS: A Cascaded Machine Learning Framework for Intrusion Detection System in VANET. *Electron.* **2023**, 12 (18), 3779.
18. P. V. Sontakke, N.B. Chopade. Hybrid DNN-BiLSTM-aided intrusion detection and trust-clustering and routing-based intrusion prevention system in VANET. *J. Control Decis.* **2023**, 1–18.
19. M. kumar Pulligilla, C. Vanmathi. An authentication approach in SDN-VANET architecture with Rider-Sea Lion optimized neural network for intrusion detection. *Internet of Things (Netherlands)* **2023**, 22, 100723.
20. M. Arya, H. Sastry, B.K. Dewangan, et al. Intruder Detection in VANET Data Streams Using Federated Learning for Smart City Environments. *Electron.* **2023**, 12 (4), 894.
21. S. Amaouche, A. Guezzaz, S. Benkirane, et al. FSCB-IDS: Feature Selection and Minority Class Balancing for Attacks Detection in VANETs. *Appl. Sci.* **2023**, 13 (13), 7488.
22. E. A. Shams, A. Rizaner, A.H. Ulusoy. Flow-based intrusion detection system in Vehicular Ad hoc Network using context-aware feature extraction. *Veh. Commun.* **2023**, 41, 100585.
23. J. Cui, H. Sun, H. Zhong, et al. Collaborative Intrusion Detection System for SDVN: A Fairness Federated Deep Learning Approach. *IEEE Trans. Parallel Distrib. Syst.* **2023**, 34 (9), 2512–2528.
24. H.M. Song, J. Woo, H.K. Kim. In-vehicle network intrusion detection using deep convolutional neural network. *Veh. Commun.* **2020**, 21, 100198.
25. D. Lei, X. Guo. A shuffled frog-leaping algorithm for hybrid flow shop scheduling with two agents. *Expert Syst. Appl.* **2015**, 42 (23), 9333–9339.
26. A.W. Burange, V.M. Deshmukh. Trust based secured Routing System for low power networks. *J. Integr. Sci. Technol.* **2023**, 11 (1), 431.
27. W. Zhao, L. Wang, Z. Zhang. Atom search optimization and its application to solve a hydrogeologic parameter estimation problem. *Knowledge-Based Systems*. 2019, pp 283–304.