

“A Computationally Efficient Intrusion Detection System for VANETs Using Tabular Learning Techniques”

1st Dr. Nilesh T. Gole

Dept. Of Computer Science
& Engineering Vidarbha Institute of
Technology, Nagpur
Email: n.gole@vitnagpur.edu.in

2nd Ms Poonam Khade

Dept. Of Computer Science
& Engineering
Vidarbha Institute of Technology, Nagpur

3rd Aditya Patil

Dept. Computer Science & Engineering
Vidarbha Institute of Technology, Nagpur
Email: a.patil@vitnagpur.edu.in

Abstract— The rapid growth of the internet and the telecommunications industry has led to a huge explosion of network size and data consistency. Therefore, when VANET grows and grows more and more in terms of transmission speed, network connectivity, security, and safety with the deployment of cutting applications, there will be major changes in wireless communication. In order to prevent external communications from being hacked, this study provides an intrusion detection system (IDS) for VANET that makes use of obfuscation. In this paper we proposed intrusion detection system for VANET which used the Attentive Interpretable Tabular Learning (Tab Net) architecture of deep learning. Tab Net model has characteristics of good interpretability and Fast training speed. we examined the existing machine and Deep learning framework and application in cyber security and VANET and PyTorch-Fast.ai is selected due to it is designed for CPU usage but because of training speed is high we changed data block fetching method from Tabular panda to NumPy array. Finally, from the evaluated metrics, we have proposed the best DNN design suitable for the IDS. With an accuracy of 98.12% and a False Alarm Rate (FAR) of 0.78 %.

Keywords— *Vehicular Ad Hoc Networks (VANETs), Intrusion Detection System (IDS), Tabular Deep Learning*

1. INTRODUCTION

A Vehicular ad hoc network called VANET is a mobile network that allows vehicles to communicate with each other without installation. It aims to improve road safety by allowing vehicles to share alerts. Internal and external communication is considered the main feature of full and half cars. One of VANET's most important goals is security. As safety will decrease traffic, save lives, and reduce accidents. Other services, like as Internet connectivity, weather forecasts, and geo location data, can enhance the travel experience in addition to safety by bringing comfort, convenience, and infotainment. VANET technology is one of the emerging ones that has the potential to significantly alter business and communication practices. It indicates the best qualities of people and will soon come true. In-Vehicle Wireless Access, a technology based on the IEEE 802.11p protocol, which serves as a broadcast standard for special short-range communication technologies, has been built into many cars by manufacturers (DSRC). VANET enables wireless vehicular communication, including V2V and connectivity to vehicle infrastructure, with DSRC (V2I). In radio coverage areas, VANETs are used to represent intra-vehicle (V2V) or vehicle-to-vehicle (V2I) communication.

2. LITERATURE REVIEW

1) *In recent years, a number of literature reviews have been published on machine learning techniques for intrusion detection, some of which focus on deep learning techniques. Previous studies related to NIDS are discussed as follows: Su, Sun, Zhu, et al. [1] propose a unique bidirectional damping multiple convolution (BATMC) model with two-phase learning of bidirectional long-term memory (BLSTM) and attention to time series features for intrusion detection using the NSL-KDD dataset. The BLSTM layer connecting the forward LSTM and the backward LSTM is used to extract features on the traffic bytes of each packet. Zhang, Ruin, Yin, et al., [2] used a deep AEs model. It was compared to other deep learning methods used to evaluate intrusion detection methods on the NSL-KDD dataset. The experiment shows better performance than the previous classification model such as RF and SVM. Solanki, Gupta, and Rai [3] developed a method for data mining use in network security settings such as intrusion detection frameworks, where a hybrid feature selection algorithm (HFSA) is combined with polynomial NB techniques. They also evaluate the **Sample Heading (Third Level). Only two levels of headings should be numbered. Lower level headings remain unnumbered; they are formatted as run-in headings.***

Classification performance in terms of classification accuracy, precision and recall, all aspects of existing recognition systems. This preserves the implementation of the distribution, which also significantly reduces computation time and costs. To establish a safe network, Mohan, Jain, Suyal, et al [4]. Applied data mining classification algorithms for intrusion detection in the following study. For classification, the Random Tree (RT), NB, J48, and RF ML classifiers are employed. Comparison of strategies based on factors such as data accuracy, detecting interesting patterns, and extracting usable information. Shah, Clachar, Minimair, et al [5]. study presents multi-class classification baselines for identifying normal network traffic from direct and disguised network attacks using various machine learning methods and neural networks. The baselines for this study are derived from the Advanced Security network Metrics and Tunneling Obfuscation (ASNMT0) dataset. On chosen vulnerable network services, the dataset collected both valid and disguised malicious TCP/IP traffic. With up to 95% accuracy, the multi-class classification Anomaly-Based Intrusion

Detection System (AIDS) can differentiate between disguised and direct network intrusion [6]. Kumar and Singh study proposes the creation of a Distributed intrusion detection system (DIDS) on a reliable platform such as cloud infrastructure using new and promising technologies such as Block chain. It also discusses the architecture of the DIDS utilizing Cloud Computing Infrastructure and Block chain in this research. DIDS server's performance with various load of data are shown. There is other more aspects to consider, like Communication lag, block-chain overhead, and the expense of implementation, and so on, which must be considered and analyzed [7]. Grey Wolf Optimizer Algorithm based on Particle Swarm Optimization (PSOGWO) algorithm used by Chen, Song, Bo, et al[8]. Optimizes parameters of SVM to improve the overall performance of intrusion detection based on SVM. Results show that the method has better performance for network intrusion detection. The "optimal detection model" of an SVM classifier is determined by the fusion of PSOGWO and SVM algorithms in [9]. Dutt, Borah, and Maitra [10] study offers Statically Modeling based Anomaly detection (SMAD) as the initial layer of an IDS. It acts as the innate natural Immune System (IS) interface, capturing the initial network data to determine the first-hand vulnerability. The second layer, Adaptive Immune-based Anomaly Detection (AIAD) has been studied for determining the characteristics of suspicious network packets for anomaly detection. It mimics the adaptive immune system by taking T-cell and B-cell activation. It extracts pertinent information from the header and payload parts for efficient intrusion detection. For intrusion detection, experiments were carried out on both real-time network traffic and the standard datasets KDD 99 and University of New South Wales (UNSW)-NB15. Using real-time traffic and standard data sets, the SMAD model achieves a true positive rate of up to 96.04 % and a true positive rate of about 97%. The AIAD model is used to test for vulnerabilities in highly suspicious traffic found in the SMAD model. The results reveal a strong true positive rate, approaching almost 99 percent accuracy in detecting file-based and user-based abnormalities in both real-time traffic and standard data sets [11]. Chen, Kuang, Xu, et al [12]. Present a unique NIDS system based on a convolutional in this study. Using both extracted characteristics and actual network traffic, we build deep-learning-based detection models. We run extensive tests on well-known benchmark datasets. The findings validate the usefulness of our technique and show that the model trained using raw traffic outperforms the model trained using extracted features in terms of accuracy [13].

3. REVIEW OF AI AND DEEP LEARNING COMPONENTS

Deep learning (DL), Li et al. Using back propagation, DL can learn complex data, and it shows that the machine changes its parameters at each layer. Deep Belief Network (DBN), Auto Encoder (AE), Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) are some DL algorithms in regular use. A more sophisticated form of machine learning that uses multiple levels of data abstraction across different layers, deep learning can discover complex patterns in data sets by using the process recovery, which shows how the

system changes the internal processes of each layer. . Deep learning methods are using various DBNs, AEs, CNNs and RNNs [15]. Back-propagation, which shows how the system changes the internal parameters of each layer, allows Deep Learning to learn different patterns in the data. Al-Gardai and others [15]. ANN To store and identify which inputs are important to the output, ANNs use hidden algorithms. The secret process has a relationship between the importance of communication and information about the impact of ideas. [16].

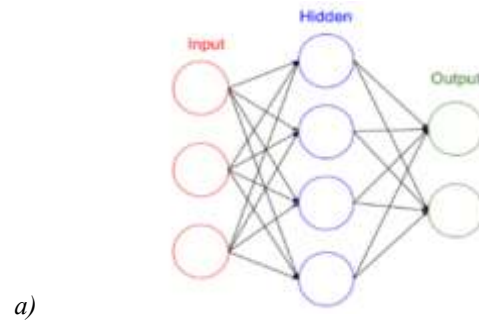


Figure 1: Artificial neural network

DNNs, on the other hand, use an ANN component, as shown in Figure 2. It is claimed to work well for updating/updating models. Because each hidden layer node creates connections and evaluates inputs to determine outputs. So why not stack more and more to get the most out of the hidden system? Therefore, the deep web has many hidden levels [16]. The most important/important features Tab Net improves learning by using color combinations to select which features to consider at each stage [16]. Since the given option is pattern specific, it may vary depending on the strategy. Sharing hyper parameters, weights and biases between layers for optimal performance is another important aspect of multi-layer Tab Nets. Figure 4 shows a fog view of several positions, including the FT and AT positions 6 and 7.

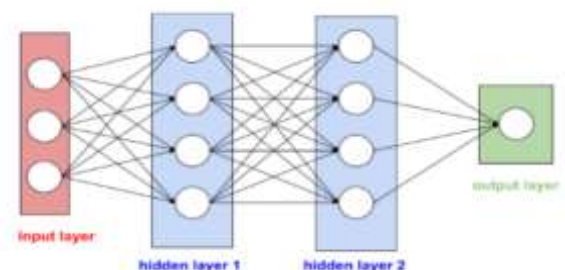


Figure 2: Deep neural network

The topic at hand: improving performance. Deep meshes improve the accuracy of model performance. They allow the model to process multiple inputs and provide an output. It's as simple as copying and pasting a line of code for each layer to use a deep network. It doesn't matter if you use an ML platform; Telling a model to use two or 2,000 nodes per layer is as easy as entering the number 2 or 2000. However, fitting these deep networks raises a question: How do these models make their decision? The explanatory power of the model is

greatly reduced when these basic techniques are used. As we saw from Figure 3, the Deep Net allows the model to generalize itself and then store these generalizations in a hidden layer, known as a black box. The dark box is difficult to study. Although the values in the black box are known, they do not exist in a logical framework [17].

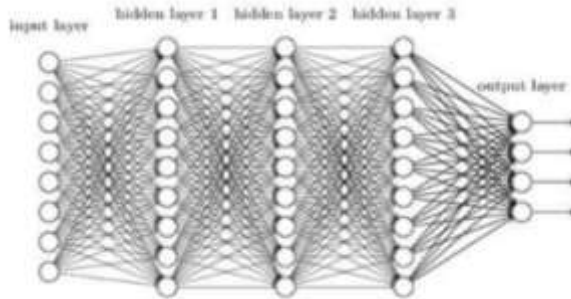


Figure 3: Complex deep neural network

4. Tab Net DNN

In order to benefit from the interpretive and less specific selection of decision tree models, TabNet, a DNN for tabular data, was developed to learn the same.

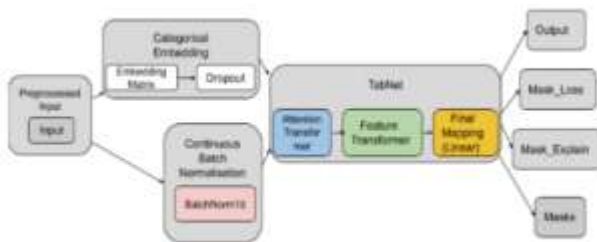


Figure 4: Tab net architecture visualization (Pochetti, 2020)

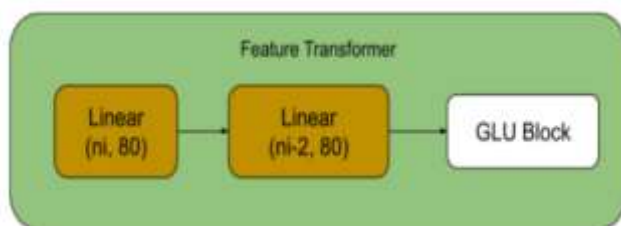


Figure 5: Feature transformation component

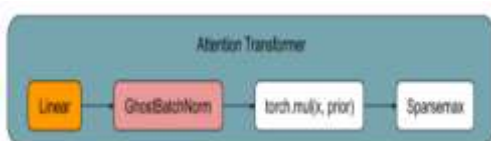


Figure 6: Attentive transformation component

Feature Transformer (FT)

The GLU layer Pochetti, 2020[17] is the main element of the second building block of Tab Net, the FT. This block consists of 4 GLU layers: the first two are shared across the entire network, i.e. the weights of the basic FCs are initialized only once, and future FT blocks share the same. The last 2 are different for each FT, which makes modeling easier. As shown in Figure 7, the main view this block is the interconnected GLU process in ResNet style after multiplying by a constant parameter

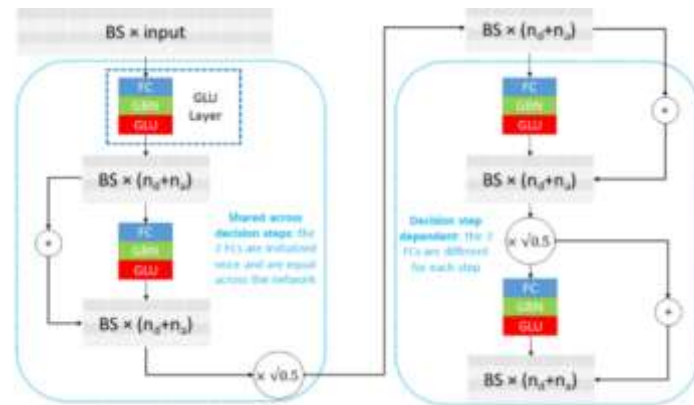


Figure 7: Feature transformer (Pochetti, 2020)

Attentive transformer (AT)

AT is the next brick in TabNet's foundation. By imposing sparsity on the feature set and teaching the network to focus exclusively on certain variables, this block aims to provide an attention mechanism for the network. An FC layer handles the learning component by extracting a size tensor as input. Instead, the Sparse ax, a variant of SoftMax that can reduce all probabilities below a certain threshold to zero, provides the rare bit. This operation resets certain features of the final resulting tensor. This operation resets certain features of the final resulting tensor. The output of the AT block and the pre tensor are used to distribute the attention mechanism in the model. The Prior, which is initially set to 1 second, is updated and multiplied both inside and outside the AT.

Decision Step (DS)

After defining the characteristics and attentive transformers, we can start to assemble the components of the decision step (DS). We will see that stacking the following DSs one after the other is the basic concept of Tab Net, so let's start by examining one. Figure 8 shows the initial link of the chain. Inside the blue dashed rectangle is the architecture, or what is repeated, and outside is either the initialization of the matrices (above) or the preparation of the raw data (once at the beginning).

1. There is no need to normalize unprocessed tabular data (x). That is handled by the first batch normalization layer, which results in $x1$.

6. As you can see from the figure, the next DS receives 4 elements: `x1`, which must be hidden by the following `m`, `previous`, `res`, and `att`.



Page 4

Traffic Attack Type	Number of samples
SSH-BruteForce	187,589
Infiltration	92,403
FTP-BruteForce	193,354
DoS-Slowloris	10,990
DoS-GoldenEye	41,508
DoS-SlowHTTPTest	139,890
BruteForce-Web	611
DoS-Hulk	461,912
SQL-Injection	87
Bot Attack	286,191
Benign	5,177,646
Total Attack	1,414,766

Figure 11: DL frameworks

We tested with numerous cutting-edge DL frameworks, which are presented in above. Two techniques are frequently employed to evaluate ML models: n-fold cross-validation (typically 10-fold) and train-test split (typically 70-30 or 80-20). Train-test split is used when the dataset has a sizable number of samples in each category, but n-fold cross-validation is used when the number of samples in some categories is small or unequal. When appropriate, we used both strategies. In order to test binary-class and multi-class classification, we combined all the original files and produced two more datasets. Furthermore, for binary-class categorization, we assigned the values 1 and 0, respectively, to all attack traffic and all benign traffic. We either left the labels as-is (textual) or converted them into one-hot (also known as one-of-K scheme) encoding for multi-class categorization, depending on the framework requirements.

6. Evaluation of the Deep Learning (DL) Framework

It uses recorded performance parameters like accuracy, loss, time, etc. and DL frames. To develop a better deep learning classifier to act as a base classifier, he experimented with several parameters of each framework. Furthermore, attempts were made to normalize the numerical results, but none of the studies found sufficient performance change to warrant normalization of the numerical values. Results from the Tensor-Flow-Keras model

Table 1: Performance results for Tensor-Flow-Keras.

DLframework	Class	Accuracy (%)	Loss(%)	Time(min)
Tensor-Flow-Keras	Binary-class	94.4	18.54	105.90
	Multi-Class	94.73	17.58	632.35

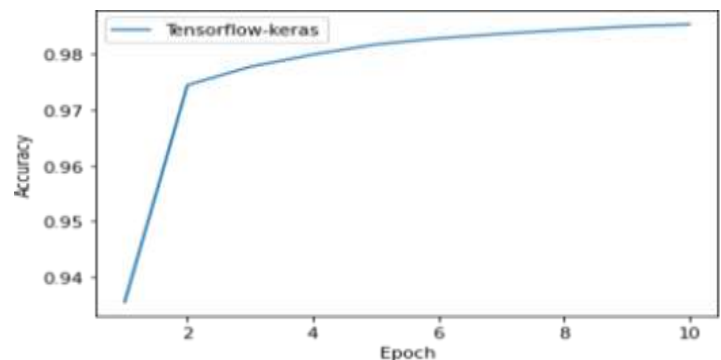


Figure 12: Accuracy results over 10 epochs.

Table 2: Performance results for Theano-Keras.

DLframework	Class	Accuracy(%)	Loss(%)	Time(min)
Theano-Keras	Binary-class	92.01	18.52	105.90
	Multi-Class	95.105	17.43	632.02

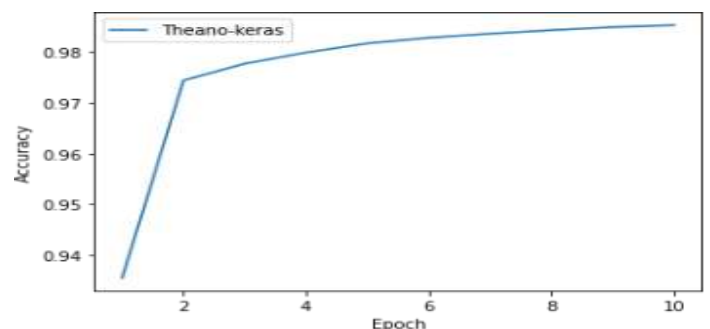


Fig13: Accuracy results on over 10 epochs.

Performance Results of PyTorch-Fast.ai

DL framework	Class	Accuracy (%)	Loss (%)	Time(min)
PyTorch-Fast.ai	Binary-class	96.68	0.3769	675.98
	Multi-Class	98.31	7.0616	683.12

Table 3: Performance results for PyTorch-Fast.ai

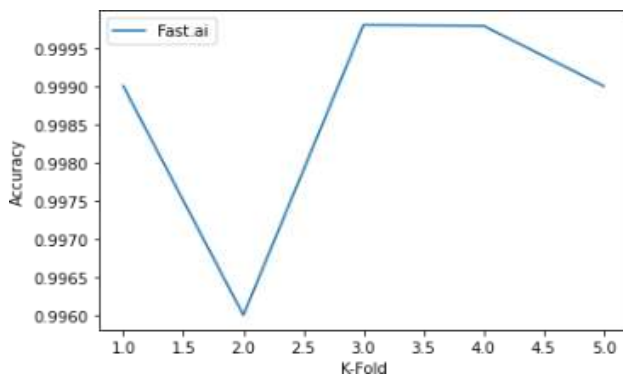


Figure 14: Accuracy results over 5- folds.

1. Proposed Modification on Fast.ai Tabular Training

Fast.ai has introduced Tabular-Pandas, a powerful tool to assist with processing your data while working with tabular data. The ability to have everything in one location, encode and decode all of your tables at once, and have very little memory use on top of your Pandas data-frame make it quite helpful and practical. I wanted to maintain Tabular-Pandas in its current form because it's a fantastic pre-processing tool while also speeding up everything! As an alternative, we'll develop a new Dataset class and turn our Tabular-Pandas object into a NumPy array there. Why is that crucial? In comparison to Python, NumPy is a considerably faster library that has been hyper-optimized to use as much C code as possible.

The first modification is converting the Tabular-Pandas dataset of fast.ai with NumPy Dataset. NumPy dataset performs the same operation with fast.ai Tabular-Pandas the only difference is that NumPy Dataset calls to NumPy () on the stored Data-Frame (cats, counts, and ys) to maintain as

it is. We'll assign our cats as np.long and our counts as np.float32 to deal with categorical versus continuous variables, respectively (we also have our ys as np.int8 but this is because we're doing classification).

8. RESULTS AND DISCUSSION

Recorded process uses parameters such as accuracy, loss, time, etc. and DL staff. To develop a better deep learning classifier that could serve as a base classifier, he experimented with several parameters of each structure. In addition, attempts have been made to generalize the numerical results, but none of the studies have shown sufficient change in performance to justify generalizing the numerical values. The results from the Tensor-Flow-Keras model

Deep learning (DL) Framework evaluation

We use deep learning techniques to measure performance metrics such as accuracy, loss, time, etc. for analysis. This table summarizes the CPU performance results for each data set. For each file, Theano's results are similar to Keras-TensorFlow, so we don't think Keras-Theano will provide better results than fast.ai, for example. Fastai is actually another layer built on top of PyTorch that offers many new features for neural networks, including data visualization tools, data entry and classification capabilities, and retrieving the number of classes in a dataset. providing and extending training capabilities known as "callbacks" (which Keras has but PyTorch does not).

DL framework	Class	Accuracy (%)	Loss (%)	Time (min)
Theano-Keras	Binary-class	92.01	18.52	105.90
	Multi-Class	95.105	17.43	632.02
Tensor-Flow-Keras	Binary-class	94.4	18.54	105.97
	Multi-Class	94.73	17.58	632.35
PyTorch-Fast.ai	Binary-class	96.68	0.376	675.98
	Multi-Class	98.31	7.061	683.12

Table 4: Comparison results of DL framework.

Modified PyTorch-Fast.ai Evaluation

We ran tests on quick fix.ai to see how fast it is and how much it uses. Tips for success, follow all the details. Site complexity (memory usage) has been tested on network-level datasets to confirm that optimized consumes less memory than basic, fast. one Keep in mind that small devices have limited resources, should have simple and light systems. So in agreement, by thinking everything from Pandas to NumPy, we can reduce the time it takes to fetch a bunch of data. Then, to complete what we've seen, our documentation includes a Loader that can handle NumPy arrays. Your training on tabular data should be reapplied now that you have a better understanding of how to integrate internal data with NumPy. Tips for fixing file loaders go much further than the basics, as you can see from the table below.

Fast.ai	Devices	First Batch		Per batch		Per	
		Train (sec)	Valid(Sec)	Train (sec)	Valid(Sec)	Epoch(sec)	Ten epoch(sec)
Baseline Fast.ai	CPU	878.8 ms	8.65 ms	20.3 s	4.85 s		
	GPU	877 ms	10 ms	20.3 s	4.86 s	96 s	960 s
Modified Fast.ai	CPU	24.8 ms	3.56 ms	1.3 s	326 ms		
	GPU	25.4 ms	3.79 s	2.3 s	567 ms	69.6 s	696s

Figure15: comparison performance based on time/speed

As we have seen from below table, the proposed system has the highest speed on customized data loaders than baseline for CPU base sensor node. But not for GPU based because NumPy is designed for latest CPU architecture.

Fast.ai	Devices	First Batch		Per batch		Per	
		Train (sec)	Valid(Sec)	Train (sec)	Valid(Sec)	Epoch(sec)	Ten epoch(sec)
Baseline Fast.ai	CPU	1487.34 MB	1486.71 MB	1502.07 MB	1502.10 MB		
	GPU	1484.67 MB	1503.06 MB	1505.44 MB	1505.67 MB	4032.7 MB	40327.00 MB
Modified Fast.ai	CPU	1491.96 MB	1491.96 MB	1492.48 MB	1492.50 MB		
	GPU	3152.18 MB	3154.28 MB	3154.72 MB	3154.72 MB	4032.7 MB	40327.00 MB

Figure16: comparison performance based on memory consumption.

Comparison of the IDS classifiers' performance with current work Fig. 17 displays the findings of an additional investigation and a comparison with earlier work on intrusion detection conducted using ML and DL approaches. It can be demonstrated that our study has performed better than any earlier investigations.

Algorithm Used	Accuracy(%)	Precision(%)	FAR(%)	Recall(%)	FI-score(%)
BLSTM[8]	84.25	X	X	X	X
AE[11]	79.74	82.22	X	X	X
RF,NBRT,JB[13]	90	X	X	X	X
ANN,DT,KNN[16]	95	X	X	X	X
SMAE[17]	97.1	X	X	2.7	X
CNN[18]	96.81	X	X	X	X
LSTM,KNN[19]	92.3	X	X	X	X
Proposed TabNet [22] (All layer ID6)	98.12	98.66	0.78	98.34	98.60

Figure 17: comparison between the proposed IDS classifier and the current IDS classifier

9. Results of Feature Selection

Detailed result of feature selection As shown in Figures 18, the probability of being attacked is determined by various factors; therefore, how to better select features becomes an important factor affecting model performance. To select features, we use the Tab Net feature selection method.

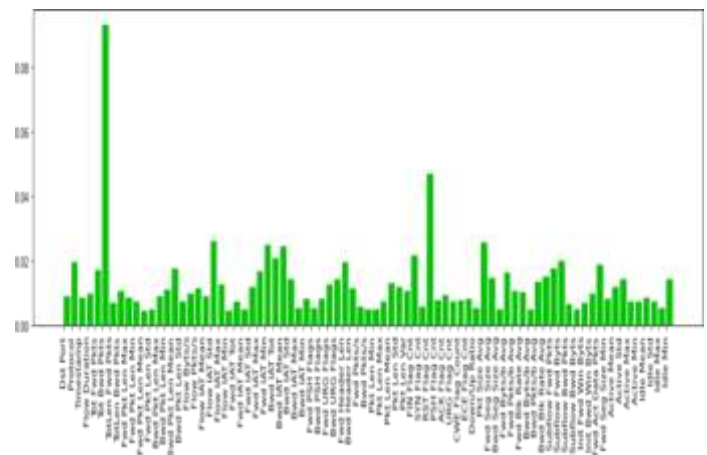


Fig18: Important Features

II. CONCLUSION

This is unique as it is the first time TabNet DL has been used for VANET security. We have fully evaluated the VANET architecture, including security and privacy aspects. As part of our research, we have narrowed the scope of this research to securing network data. We introduced a heavy-duty design for IDS in the VANET network. The TabNet model was proposed, which produced exceptional results, surpassing previous works in the literature. In addition, unlike previous studies, we used the CIC-CSE-IDS2018 dataset for the experiment. As shown in result and discussion section. All layer IDS outperforms all author known IDS classifiers in terms of accuracy and FAR, with 98.12 % and 0.78%, respectively.

References:

- An Effective Machine Learning Approach for Clustering Categorical Data with High DimensionsS Umar, TD Deressa, TB Yadesa, GB Beshan... - International Conference on Artificial Intelligence and ..., 2021 [2]Zhang, C., Ruan, F., Yin, L., Chen, X., Zhai, L., & Liu, F. (2019). A deep learning approach for network intrusion detection based on nsl-kdd dataset. 2019
- [3]Solanki, S., Gupta, C., & Rai, K. (2020). A survey on machine learning based intrusion detection system on nsl-kdd dataset. *Int. J. Comput. Appl.*, 176, 36–39.
- [4]Mohan, L., Jain, S., Suyal, P., & Kumar, A. (2020). Data mining classification techniques for intrusion detection system. 2020 12th International Conference on Computational Intelligence and Communication Networks (CICN), 351–355.
- [5]Shah, A., Clachar, S., Minimair, M., & Cook, D. (2020). Building multiclass classification baselines for anomaly-based network intrusion detection systems. 2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA), 759–760. <https://doi.org/10.1109/DSAA49011.2020.00102>
- [6]LSSA: A Protective Shared Data Communication Mechanism in Cloud EnvironmentS Umar, N Gole, PG Kulurkar, TB Yadesa, P Prabhat - Advanced Informatics for Computing Research: 4th ..., 2021 [7]Mane, S., & Rao, D. (2021). Explaining network intrusion detection system using explainable ai framework. *arXiv preprint arXiv:2103.07110*.
- [8]Chen, C., Song, L., Bo, C., & Shuo, W. (2021). A support vector machine with particle swarm optimization grey wolf optimizer for network intrusion detection. 2021 International Conference on Big Data Analysis and Computer Science (BDACS), 199–204.
- [9]He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.
- [10]Dutt, I., Borah, S., & Maitra, I. K. (2020). Immune system based intrusion detection system (is-ids): A proposed model. *IEEE Access*, 8, 34929–34941. <https://doi.org/10.1109/ACCESS.2020.2973608>
- [11]SwarnaSugi, S. S., & Ratna, S. R. (2020). Investigation of machine learning techniques in intrusion detection system for iot network. 2020 3rd International Conference on Intelligent Sustainable Systems (ICISS), 1164–1167. <https://doi.org/10.1109/ICISS49785.2020.9315900>
- [12]Chen, L., Kuang, X., Xu, A., Suo, S., & Yang, Y. (2020). A novel network intrusion detection system based on cnn. 2020 Eighth International Conference on Advanced Cloud and Big Data (CBD), 243–247. <https://doi.org/10.1109/CBD51900.2020.00051>
- [13]He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.
- [14]Conneau, A., Schwenk, H., Barrault, L., & Lecun, Y. (2016). Very deep convolutional networks for text classification. *arXiv preprint arXiv:1606.01781*.
- [15]Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Cheng, Q., Chen, G., et al. (2016). Deep speech 2: End-to-end speech recognition in english and mandarin. *International conference on machine learning*, 173–182.
- [16]L., Rashid, S. A., Mohammed, A. H., Alani, S., & Mustafa, A. S. (2020). A review of applications, characteristics and challenges in vehicular ad hoc networks (VANETs). 2020 international congress on human-computer interaction, optimization and robotic applications (HORA),
- [17]Pochetti, F. (2020). Tabular Deep Learning: TabNet deep-dive. <https://francescopochetti.com/tabular-deep-learning-tabnet-deep-dive/>
- [18]Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018b). Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp*, 1, 108–116.
- [19]Polzlbauer, G., Lidy, T., & Rauber, A. (2008). Decision manifolds—a supervised learning algorithm based on self-organization. *IEEE transactions on neural networks*, 19(9), 1518–1530.
- [20]Grbovic, M., & Cheng, H. (2018). Real-time personalization using embeddings for search ranking at airbnb. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 311–320.
- [21]Gao, Y., Wu, H., Song, B., Jin, Y., Luo, X., & Zeng, X. (2019). A distributed network intrusion detection system for distributed denial of service attacks in vehicular ad hoc network. *IEEE Access*, 7, 154560–154571.
- [22]Zeng, Y., Qiu, M., Zhu, D., Xue, Z., Xiong, J., & Liu, M. (2019). Deepvcm: A deep learning based intrusion detection method in vanet. 2019 IEEE 5th intl conference on big data security on cloud (BigDataSecurity), IEEE intl conference on high performance and smart computing (HPSC) and IEEE intl conference on intelligent data and security (IDS), 288–293.