Can Network Flow Traffic Attacks Fool Machine Learning Models?: A Systematic Approach

Ming Liu
Department of Computer Science
Central Michigan University
Mount Pleasant, MI
liu3m@cmich.edu

Mahima Katuwal
Department of Computer Science
Central Michigan University
Mount Pleasant, MI
katuw1m@cmich.edu

***Abstract*—Machine learning (ML) is a subfield of Artificial Intelligence (AI) in which "learning" is involved. ML can be an essential element of a powerful security platform. Nowadays, cybersecurity relies more and more on ML models, especially Deep Learning (DL) algorithms, for network anomaly detection, network attack detection and network intrusion detection. While ML models are increasingly prominent for network security, adversarial ML, on the other hand, attempts to defeat ML models through malicious input. As security within ML has not received much attention, this project is the first of its kind to experiment with adversarial scenarios using malwares from the labeled data of Network Flow Traffic to check if the new modified Network flow traffic attacks may or may not fool ML models. Thus, the experiment focuses on a slight modification of some features such as Frame Time, Source IP, Source Port, Total Length of Packets, Header Length and Flow Bytes/Second, and URG Flag Count in network flow traffic which makes the modified network traffic flow satisfy the needs of, and ultimately bypass, the ML detector. We use labeled packet data from Canadian Institute for Cybersecurity (CIC-IDS2017), preprocess them, and focus on the Deep Learning algorithm called Convoluted Neural Network (CNN) for modeling a network attack detector. Our comprehensive experimental results show how the modification of source IP from the raw network flow traffic data fool the ML algorithm and bypass the ML detector.**

***Index Terms*—Machine Learning, Deep Learning, Network Flow Traffic, Network Flow Traffic security, Network Attack, Data modification, Machine Learning Security**

I. Introduction

The numbers of networks and network users significantly increased, especially in 2020 when coronavirus ravaged the globe, and most events moved online. With the significant increase in network users, the traditional detection methods fell behind and became a serious threat to network security [1]–[3]. Meanwhile, the fast development of hard drive, availability of large datasets and advanced algorithms allow Machine Learning (ML) algorithms to become the most popular ones for application of network security [1]–[3].

ML can be looked at as a principle to allow machines to learn the rules and perform certain tasks like classification, predictions etc. [4]. ML has been booming since the early age because of the cheap computations using ML algorithms and high chances of precise results. The driven advancement of ML is because of large dataset, powerful and advance computers, and developed ML and DL algorithms [4]. And to safeguard such advancements of ML, the security of ML is important [5]. Therefore, this project focuses on studying how modified network flow traffic satisfy the needs and at last, bypass the machine learning detector/algorithm.

In this project, data preprocessing and data mining were done using two DL algorithms, CNN and Long-Short Term Memory (LSTM) in the dataset— CIC-IDS2017 from Canadian Institute for Cybersecurity. Then the dataset was randomly divided into training, validation, and testing sets. To develop a network attack detector, ML algorithms— Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) were applied for data modeling [6], [7]. And, after the network attack detector was modeled, investigators focused solely on CNN to develop an algorithm to slightly modify the original data. This is because both CNN and LSTM gave high accuracy rates for training and validation data. The investigator's modification process focused on attributes such as Frame Time, Source IP, Source Port, Total Length of Packets, Header Length and Flow Bytes/Second from the network flow dataset to check which ones were adversarial examples. After the modification, investigators tested the security of network attack detector by feeding the detector with modified network flow data (Figure 1). This work tested the security of network attack detector of CNN and/or LSTM classifiers and, as a result, showed only Source IP to be an adversarial example.

The framework of this paper is organized as follows. Section 2 reviews related literatures/publications on network attack classification and its security. Section 3 gives a detailed description of the dataset used and the network attack classifier modeling. Section 4 provides the algorithm to modify network flow traffic and talks about the experimental results and key findings and finally, section 5 concludes the paper.

II. Related Works

The applications of machine learning algorithms have been increasing gradually. ML has been proved a powerful tool that it can be used as a defense mechanism against serious attack models like DDoS, DoS, Botnets, email spam etc. [8]–[10]. Thus, ML is important for network security, especially for DL. DL is a relatively new approach of ML which was inspired by [11] and it has been quickly developing since then. Two of the practically used DL algorithms are CNN and RNN [2].

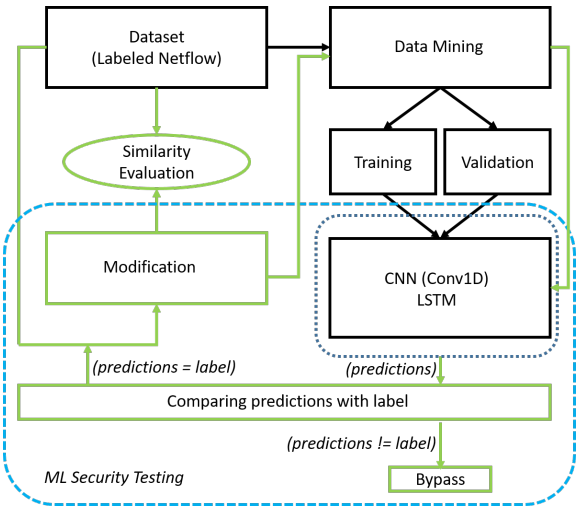

Fig. 1. Two steps in general, step 1 is data modeling (black) and step 2 is ML security testing (green).

Likewise, LSTM was developed as a variant of RNN to meet the requirement of long-term memory that traditional RNN could not achieve [12].

To detect malicious attackers in the network flow traffic, classification of the packets is a must. In 2017, [13] proposed an improved classification algorithm based on CNN. They have used a min-max scaler to process the traffic data and mapped them to grey image, which is then used as an input data for CNN for feature learning [13]. Likewise, using LSTM to predict events with time-series has shown best results [14]. For accurate prediction of data, adding features on multiple time scales help improve LSTM classification in network flow traffic [14]. In [15], researchers proposed an experiment to compare the performance of CNN [16] alone, RNN [17] especially LSTM [17], [18] alone, and the combination of both CNN and RNN. They have stated that even though CNN and RNN alone can be very good classifiers, the combination of CNN and RNN, by far, gives the best detection performance [15].

On the other hand, researchers have also suggested that network attacks are able to easily bypass DL models. In 2019, [19] did a survey on the threat of adversarial attacks in ML for network security. These authors suggested that realistic adversarial network attacks will be carried out on network flow in transit, not like adversarial data attack that is most likely to be carried out on data at rest [19]. They also documented that adversarial attacks affect only two tasks in network security. One of these two tasks is classifier and most of the network attack detector are classification models [19].

DL models are more at risk to adversarial examples which are generated by intentional perturbations on original inputs which ultimately lead ML models or DL models to misclassify [20], [21]. Indeed, in 2018, [19] suggested that DL is vulnerable to adversarial attacks where changes in the inputs which are too small to be even noticeable, could completely fool DL models. Similarly, [22] also mentioned that some minor modification in inputs, can deceive classifiers. This causes classifier such as CNN to provide incorrect predictions from input data [22]. In 2019, [19] also discussed how an attacker can modify an input data which might lead to wrong prediction when it is bypassed through a ML model.

Even though there are a lot of research papers on security of ML or DL, as per our best knowledge, we could not find any publications that talks about the modification of Network Flow Traffic attacks that may or may not fool ML models.

III. Methodology

This section provides information about the dataset used, data preprocessing methods, and description of ML/DL models.

A. Selected Dataset

For this project we used CIC-IDS2017 dataset from Canadian Institute for Cybersecurity (CIC). CIC is based at the University of New Brunswick in Fredericton. CIC is a training, and research institute for cybersecurity which provides consultancy services, training, and career-related education to students. Their program focuses on the implementation of novel approaches in the field of network security and big data.. Moreover, we used a new dataset for the modification process. This new dataset is also part of CIC-IDS2017 dataset that network attack detector had not seen during training, validation, and testing phases.

CIC used proposed B-Profile system to profile the abstract behavior of human interactions and generated naturalistic benign background traffic [23].The data in CIC-IDS2017 dataset were captured from 9 am, Monday, July 3, 2017 through 5 pm on Friday July 7, 2017, for a total of 5 days. The dataset contains both benign and the recent common attack models like DDoS, Botnet, ARES, Port Scan, Brute Force etc. After last update on September 10, 2019, the dataset has 2,371,775 rows and 85 features.

There are some other advantages of using this dataset for our project. For example, CIC-IDS2017 dataset covers all of the 11 criteria which are necessary for building a reliable benchmark dataset [24]. This dataset has also been analyzed by [25] which helps us understand the data better and allows us to preprocess the data well to meet the requirements of this project.

B. Preprocessing of the Dataset

For this step, we labeled benign as 1 and all other malicious data as 0. The frame time was split into two string variables, hours(h) and minutes(m). Then it was converted to an integer value of "hhmm" where hh refers to the number of hours and mm to the number of minutes. As a plain example, if h = 12 and m = 28, then the converted time frame value will be 1238. Likewise, source IP and destination IP addresses were also converted to values by shifting a certain number of bits to the left followed by logarithm transformation. Moreover, the numerical features were scaled by using MinMaxScaler and the categorical features that contained labeled values were encoded by using one-hot encoding (Figure 1). After a

random shuffle, we split a total of 1,195,960 rows into 769,560 training datasets, 190,000 validation datasets and 240,000 testing datasets with 66 features which are illustrated in Table I.

TABLE I
FEATURES SELECTED FOR MODELING NETWORK ATTACK DETECTOR BEFORE ONE-HOT ENCODING

Features	Type	Transformation
SourcePort	int64	MaxMinScaler
DestinationPort	int64	MaxMinScaer
Protocol	category	One-hot Encode
FlowDuration	int64	MaxMinScaler
TotalFwdPackets	int64	MaxMinScaler
TotalBwdPackets	int64	MaxMinScaler
TotalLengthofFwdPackets	float64	MaxMinScaler
TotalLengthofBwdPackets	float64	MaxMinScaler
FwdPacketLengthMean	float64	MaxMinScaler
BwdPacketLengthMean	float64	MacMinScaler
FlowBytesPerS	float64	MaxMinScaler
FlowPacketsPerS	float64	MaxMinScaler
FlowIATMean	float64	MaxMinScaler
FwdIATMean	float64	MaxMinScaler
BwdIATMean	float64	MaxMinScaler
FwdPSHFlags	category	One-hot Encode
BwdPSHFlags	category	One-hot Encode
FwdURGFlags	category	One-hot Encode
BwdURGFlags	category	One-hot Encode
FwdHeaderLength	int64	MaxMinScaler
BwdHeaderLength	int64	MaxMinScaler
FwdPacketsPerS	float64	MaxMinScaler
BwdPacketsPerS	float64	MaxMinScaler
PacketLengthMean	float64	MaxMinScaler
FINFlagCount	category	One-hot Encode
SYNFlagCount	category	One-hot Encode
RSTFlagCount	category	One-hot Encode
PSHFlagCount	category	One-hot Encode
ACKFlagCount	category	One-hot Encode
URGFlagCount	category	One-hot Encode
CWEFlagCount	category	One-hot Encode
ECEFlagCount	category	One-hot Encode
ECEFlagCount	category	One-hot Encode
DownToUpRatio	float64	MaxMinScaler
AveragePacketSize	float64	MaxMinScaler
AvgFwdSegmentSize	float64	MaxMinScaler
AvgBwdSegmentSize	float64	MaxMinScaler
FwdHeaderLengthPointOne	int64	MaxMinScaler
FwdAvgBytesPerBulk	int64	MaxMinScaler
FwdAvgPacketsPerBulk	int64	MaxMinScaler
FwdAvgBulkRate	int64	MaxMinScaler
BwdAvgBytesPerBulk	int64	MaxMinScaler
BwdAvgPacketsToPerBulk	int64	MaxMinScaler
BwdAvgBulkRate	int64	MaxMinScaler
SubflowFwdPackets	int64	MaxMinScaler
SubflowFwdBytes	int64	MaxMinScaler
SubflowBwdPackets	int64	MaxMinScaler
SubflowBwdBytes	int64	MaxMinScaler
InitWinbytesforward	int64	MaxMinScaler
InitWinbytesbackward	int64	MaxMinScaler
actdatapktfwd	int64	MaxMinScaler
ActivMean	float64	MaxMinScaler
IdleMean	float64	MaxMinScaler
frameTime	float64	MaxMinScaler
SourceIPValue	float64	MaxMinScaler
destinationIPValue	float64	MaxMinScaler
binaryLabel	category	Target

C. Network Flow Traffic Attack Detectors

It has been established that the combination of CNN and RNN function as the best-known model detector [15]. That is why, we have applied two of the famous and common DL models, CNN and LSTM because of its impressive accuracy rate of 98% and F1 score of 0.96 [15].After data preprocess and data partition, we further extract feature 'binaryLabel' as target, train_y, validation_y and test_y and the rest of the features as train_x, validation_x and test_x, respectively. To fit CNN, we also reshaped train_x, validation_x and test_x into two-dimension(2D) tensor. Table II and Figure 2 show the model summary of CNN and the dictionary containing configuration of LSTM. For our training, validation and testing datasets, the accuracy rates were 99.0% for both CNN and LSTM models (Figures. 3 and 4).

TABLE II
MODEL SUMMARY OF CNN.

Layer (type)	Output Shape	Param Num
conv1d (Covid1D)	(None, 65, 64)	256
max_pooling1d (MaxPooling1D)	(None, 32, 64)	0
flatten (Flatten)	(None, 2048)	0
dese_2 (Dense)	(None, 128)	262272
dropout (Dropout)	(None, 128	0
dense_3 (Dense)	(None, 1)	129
Total params: 262,657		
Trainable params, 262,657		
Non-trainable params, 0		

{'name': 'sequential_1', 'layers': [{'class_name': 'InputLayer', 'config': {'batch_input_shape': (None, None, 65), 'dtype': 'float32', 'sparse': False, 'ragged': False, 'name': 'lstm_input'}}, {'class_name': 'LSTM', 'config': {'name': 'lstm', 'trainable': True, 'batch_input_shape': (None, None, 65), 'dtype': 'float32', 'return_sequences': False, 'return_state': False, 'go_backwards': False, 'stateful': False, 'unroll': False, 'time_major': False, 'units': 4, 'activation': 'tanh', 'recurrent_activation': 'sigmoid', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'recurrent_initializer': {'class_name': 'Orthogonal', 'config': {'gain': 1.0, 'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'unit_forget_bias': True, 'kernel_regularizer': None, 'recurrent_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'recurrent_constraint': None, 'bias_constraint': None, 'dropout': 0.0, 'recurrent_dropout': 0.0, 'implementation': 2}}, {'class_name': 'Dropout', 'config': {'name': 'dropout', 'trainable': True, 'dtype': 'float32', 'rate': 0.1, 'noise_shape': None, 'seed': None}}, {'class_name': 'Dense', 'config': {'name': 'dense_2', 'trainable': True, 'dtype': 'float32', 'units': 1, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}}, {'class_name': 'Activation', 'config': {'name': 'activation', 'trainable': True, 'dtype': 'float32', 'activation': 'sigmoid'}}]}

Fig. 2. The dictionary containing configuration of LSTM model.

IV. DATA MODIFICATION AND EXPERIMENT

To create adversarial scenarios in our experiment, data used for modification is a new dataset that network attack detector had not seen during training, validation and testing in section 3. This dataset is also a part of CIC-IDS2017 dataset. It has the same structure as the dataset used for modeling attack classifier and has been preprocessed using similar methodology in section 3. After preprocessing the new dataset, we have 267,511 attacks and we use this non-modified

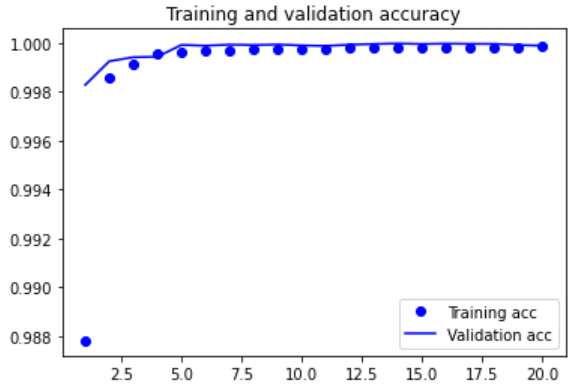

Fig. 3. The accuracy of training and validation of CNN model.

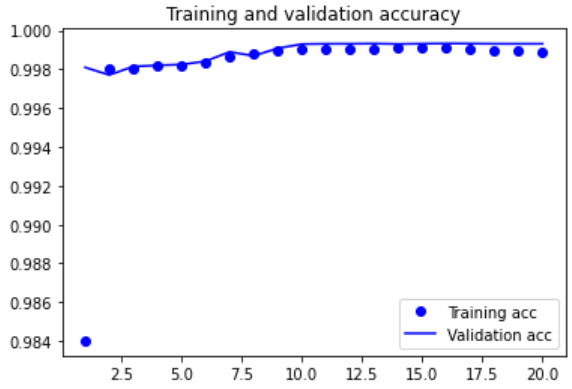

Fig. 4. The accuracy of training and validation of LSTM model.

new dataset to test our network attack detector modeled in section III (Table III).

To test the security of network attack detector, we randomly chose several attributes/features for modification, including Frame Time, Source IP, Source Port, Total Length of Packets, Header Length, Flow Bytes/Second, and URG Flag Count (Table III). In the experiments of testing security of classifier, we focus on network attack detector modeled by CNN algorithm because both CNN and LSTM models showed similar performance (Figures. 2 and 3).

A. Frame Time Modification and test

As descripted in Figure 5, time frame attribute is modified by switching the hours from morning to afternoon and vice versa. We modified a total of 267,511 attacks and all these attacks were detected by our network attack classifier (Table III).

B. Source IP Modification and the test

We simply modified source IP by replacing all source IPs with an IP that we were aware of. We found that this modified source IP led the detector to misclassify all attacks as benign (Table III).

Algorithm 1 Frame time Modification

```
procedure framtime_modification(data)
    for i ← 0 to number of rows of data do
        hour ← data.Timestamp.values[i].splite(' ')[1].split(':')[0]
        minute ← data.Timestamp.values[i].splite(' ')[1].split(':')[1]
        if hour >= 8
            data.Timestamp.values[i] ← int(string(int(hour) -7) + minute)
        if hour <= 5
            data.Timestamp.values[i] ← int(string(int(hour) + 7) + minute)
```

Fig. 5. Algorithm for frame time modification.

C. Source Port Modification and the test

All source ports were modified by replacing 8157. Here, the detector worked well to successfully detect all attacks accurately (Table III).

D. Total Length of Packet and Length of Header Modification and the test

We randomly increased or decreased total length of packet and the length of header. Our modified network attacks did not fool the ML detector (Table III).

E. Flow Bytes Per Second Modification and the test

We randomly increased or decreased flow bytes per second for network attacks and our ML detector detected them as attacks (Table III).

F. URG Flag Count Modification and the test

We simply removed all URG flags from network attacks or assigned 1 URG flag to all network attacks. The modified network attacks did not bypass the ML detector (Table III).

V. Conclusion

This project provides a proof that an non-noticeable modification will fool ML/DL model, and eventually bypass network attack detector. The objective of this project was to test the security of network attack detector modeled by ML/DL algorithms. To fulfill our objective, we first modeled a network attack detector by applying both CNN (conv1d) and LSTM algorithms, and secondly, we conducted several security tests by modifying raw network flow traffic and feeding this modified data into detector. As a result, we found that the source IP modification in network flow traffic dataset bypasses network detector (Table III).

It is very important to focus on modeling an appropriate and accurate network attack detector in this project. If, in section III, the detector we modeled fails to perform, then the security test conducted in section IV is futile. Talking about the

TABLE III
RESULTS OF DETECTOR SECURITY EXPERIMENTS AMONG MODIFIED AND NON-MODIFIED DATA.

Data Description		Benign	Attack
non-modified Attacks	Predict	0	267511
	Actual	0	267511
modified Frame Time Attacks	Predict	0	267511
	Actual	0	267511
modified Source IP Attacks	Predict	267511	0
	Actual	0	267511
modified Source Port Attacks	Predict	0	267511
	Actual	0	267511
modified Total Length Attacks	Predict	0	267511
	Actual	0	267511
modified Header Length Attacks	Predict	0	267511
	Actual	0	267511
modified Flow Bytes/Second	Predict	0	267511
	Actual	0	267511
modified URG Flag Count	Predict	0	267511
	Actual	0	267511

classifiers used in this project, CNN (conv1D) can only learn local patterns but not the order of local patterns. Whereas, RNN or LSTM can learn the sequence of patterns(the order of the local patterns). Although, both CNN and LSTM models showed very high accuracy in this project, LSTM also has its own significant drawbacks and does not work well for modeling a network attack detector [26]. Thus, in the future, we will do a mixed approach of CNN (conv1D) with LSTM and/or GRU to learn not only the patterns but also the orders of patterns detected and finalize our network attack detector before experiments.

[27] documented that an appropriate dataset is one of the most important factors for a successful experiment. However, it is very difficult to obtain a real or an appropriate artificial network attack packets dataset. Therefore, in our future work, we will further explore in the details about CIC-IDS2017 dataset and appropriately select our data to balance both classes— benign and malicious attack. We will also explore the general patterns of the whole dataset especially for malicious attacks, which will help us to understand the network flow traffic dataset and CIC-IDS2017 dataset. Eventually, an important lesson that we learned while working on this paper is that a well-curated dataset helps us to develop algorithms to find adversarial examples and conduct security testing more accurately and efficiently. Moreover, as a second approach, we could capture our own network flow traffic data to model the network attack detector and test the security of ML/DL as our experiment.

VI. ACKNOWLEDGMENT

The authors would like to thank and appreciate Dr. Qi Liao, Priya Darshani, and Salin Shakya for all the supports to make this research paper successful. Dr. Liao has been the advisor of this project, Ms. Darshani has shared her ideas about dataset collection and Mr. Shakya has helped to proofread this research paper.

REFERENCES

[1] H. Wang and B. Raj, "On the origin of deep learning," 03 2017.
[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," vol. 521, pp. 436–444, 05 2015.
[3] M. M. Najafabadi, "Machine learning algorithms for the analysis and detection of network attacks," 2017.
[4] D. Gündüz, P. de Kerret, N. D. Sidiropoulos, D. Gesbert, C. R.Murthy, and M. V. der Schaar, "Machine learning in the air," vol. 37, no. 10, pp. 2184–2199, 2019.
[5] M. S. Elsayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut, "Ddosnet: A deep-learning model for detecting network attacks," 2020.
[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.
[7] S. Hochreiter and J. Schmidhuber, "Long short-term memory," pp. 1735–1780, 1997.
[8] I. Firdaus, C. Lim, A. Erwin, and A. S. Nugroho, "Analysis of machine learning techniques used in behavior-based malware detection." 2010.
[9] C. Livadas, R. Walsh, D. Lapsley, and W. T. Strayer, "Usilng machine learning technliques to identify botnet traffic," in *Proceedings. 2006 31st IEEE Conference on Local Computer Networks*, 2006, pp. 967–974.
[10] B. Kuchipudi, R. T. Nannapaneni, and Q. Liao, "Adversarial machine learning for spam filters," 2020.
[11] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," 1998.
[12] P. Khandelwal, J. Konar, and B. Brahma, "Training rnn and it's variants using sliding window technique," in *2020 IEEE International Students' Conference on Electrical,Electronics and Computer Science (SCEECS)*, 2020, pp. 1–5.
[13] H. Zhou, Y. Wang, X. Lei, and Y. Liu, "A method of improved cnn traffic classification," in *2017 13th International Conference on Computational Intelligence and Security (CIS)*, 2017, pp. 177–181.
[14] B. Yang, S. Sun, J. Li, X. Lin, and Y. Tian, "Traffic flow prediction using lstm with feature enhancement," vol. 332, pp. 320–327, 2018.
[15] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Network traffic classifier with convolutional and recurrent neural networks for internet of things," *IEEE Access*, vol. 5, pp. 18 042–18 050, 2017.
[16] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, long short-term memory, fully connected deep neural network," *IEEE International Conference on Acoustic, Speech and Signal Process (ICASSP)*, vol. 5, pp. 4580–4584, 2015.
[17] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A critical review of recurrent neural networks for sequence learning," 2015.
[18] J. Kim, J. Kim, H. L. Thi Thu, and H. Kim, "Long short term memory recurrent neural network classifier for intrusion detection," in *2016 International Conference on Platform Technology and Service (PlatCon)*, 2016, pp. 1–5.
[19] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *IEEE Access*, vol. 6, pp. 14 410–14 430, 2018.
[20] K. Yang, J. Liu, C. Zhang, and Y. Fang, "Adversarial examples against the deep learning based network intrusion detection systems," in *MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM)*, 2018, pp. 559–564.
[21] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," 2014.
[22] N. Martins, J. M. Cruz, T. Cruz, and P. Henriques Abreu, "Adversarial machine learning applied to intrusion and malware scenarios: A systematic review," *IEEE Access*, vol. 8, pp. 35 403–35 419, 2020.
[23] I. Sharafaldin, A. Gharib, A. H. Lashkari, and A. A. Ghorbani, "Towards a reliable intrusion detection benchmark dataset," 2017.
[24] A. Gharib, I. Sharafadin, A. H. Lashkari, and A. A. Ghorbani, "An evaluation framework for intrusion detection," 2016.
[25] R. Panigrahi and S. Borah, "A detailed analysis of cicids2017 dataset for designing intrusion detection systems," vol. 7, pp. 479–482, 2018.
[26] E. Culurciello, "The fall of rnn / lstm," Jan 2019.
[27] Y. Xin, L. Kong, Z. Liu, Y. Chen, Y. Li, H. Zhu, M. Gao, H. Hou, and C. Wang, "Machine learning and deep learning methods for cybersecurity," *IEEE Access*, vol. 6, pp. 35 365–35 381, 2018.