ML/CGAN: Network Attack Analysis Using CGAN as Meta-Learning

Meejoung Kim

***Abstract*— This letter proposes a deep learning (DL) technique that uses Conditional GAN (CGAN) as meta-learning to overcome the network traffic data shortage in attack analysis. The technique consists of four parts: 1) Implement CGAN and MLP regression in IDS/Traffic classifier. 2) Preprocess network traffic data and divide it into training and test sets. 3) Generate data including labels by the trained CGAN and concatenate it to the training set. 4) Train MLP regression using the expanded dataset. The experiment is executed with the traffics provided by CAIDA and CIDDS-001. Numerical results show that the proposed technique enables us to overcome data shortage and outperforms MLP regression in measurements used to analyze network traffic. This study also suggests several discussions for network traffic analysis using GANs, such as the relationship between GAN generator loss and its performance.**

***Index Terms*— Meta-learning, conditional generative adversarial network, multi-layer perceptron, network attack detection, traffic prediction.**

I. Introduction

THE rapid increase of network traffic due to the evolution of smart devices makes network traffic analysis and predictive modeling more important than ever. Accurate predictions of network traffic are required in several areas, including congestion control and network management, for fast and secure communication [1]. The occurrence of network traffic can be considered as time series, and time series models such as autoregressive integrated moving average (ARIMA) and generalized autoregressive conditional heteroscedasticity (GARCH) have been traditionally used to analyze network traffic [2]–[3]. Recently, several neural network (NN) models such as multi-layer perceptron (MLP) and long short-term memory recurrent neural network (LSTM RNN) have been used to analyze and predict network traffic [4]–[8].

Despite the advantages of NN models including obtaining high accuracy and being able to handle complex models, they suffer from drawbacks such as the networks' black-box nature and requiring big data for acceptable performance. To compensate, the meta-learning (ML), known as 'learning to learn' technique, was designed for the network models. Detailed explanations and research trends of meta-learning can be found in [3], [9]. The generative adversarial network (GAN) is one of the models drawing attention in the field of deep learning (DL). As GANs generate sample data based on the distribution learning of a given data, its implementation can vary. Mode collapse and mode drop are well-known problems in GANs [10]. Recently, GANs have been applied to network traffic analysis to overcome data shortage [11], especially in detecting network attacks [12]–[15]. Several GANs are considered in the detection of attacks such as Convolutional Neural Network GAN (CNN GAN), Least Squares GAN (LSGAN), and a Gradient Penalty Wasserstein GAN (GP-WGAN) that considers alleviation of mode collapse. The approaches of using meta-learning and GANs have also developed for time series in [16], [17] and [11], [18], respectively.

In this letter, a DL approach is proposed using GAN as meta-learning in network attacks analysis. The motivation of this study consists of two parts: 1) Generate data that include labels and are similar to the given network traffic. 2) Find methods that increase the accuracy of attack detection rate using the generated data. The applied GAN is a Conditional GAN (CGAN) that considers labels in the general GAN. DL process consists of four parts: labeling and preprocessing a dataset, meta-learning process via a CGAN (ML/CGAN), training process, and testing process. Min-max transformation and MLP regression are used in the preprocessing and the DL processes, respectively.

To the best of our knowledge, this is the first study to analyze network attacks using CGAN as meta-learning in the DL process. The difference of this study with existing studies is as follows: i) Generate data including labels, ii) Use different distance metrics for data and label in loss functions to obtain accurate loss and alleviate mode collapse/mode drop. In addition to improving performance, this study is relevant in suggesting several discussions on the use of GAN in network attack analysis: i) Relationship between the generator loss of a GAN and the performance, ii) Reasons for non-convergence of generator loss, possibly due to the mode collapse/mode drop appearing in GANs or characteristics of real network traffic including attacks or both, iii) Methods for the convergence of loss functions.

II. Deep Learning Process

A. Description of Deep Learning Process

Let $\{\boldsymbol{X}_t\}_{t=1}^{N}$ be a time series dataset, where $\boldsymbol{X}_t$ is a traffic vector at time t and N is the number of data. Let Y_t be a label corresponding to $\boldsymbol{X}_t$, which is either binary (zero or one for normal and attack, respectively) or multiclass (from zero to some integer) depending on traffic state. Add Y_t to a new column of data $\boldsymbol{X}_t$ and represent the obtained data as $(\boldsymbol{X}_t, Y_t)$.

Fig. 1 illustrates the DL process including ML/CGAN. The subsequent DL procedure explains Fig. 1 in detail. From now

Manuscript received August 28, 2020; revised September 25, 2020; accepted September 30, 2020. Date of publication October 8, 2020; date of current version February 11, 2021. This work was supported by the Mid-career Research Program through the NRF Grant funded by the Korea government (MEST) (NRF-2019R1A2C1002706). The associate editor coordinating the review of this letter and approving it for publication was L. Lampe.

The author is with the Research Institute for Information and Communication Technology, Korea University, Seoul 02841, South Korea (e-mail: meejkim@korea.ac.kr).

Digital Object Identifier 10.1109/LCOMM.2020.3029580

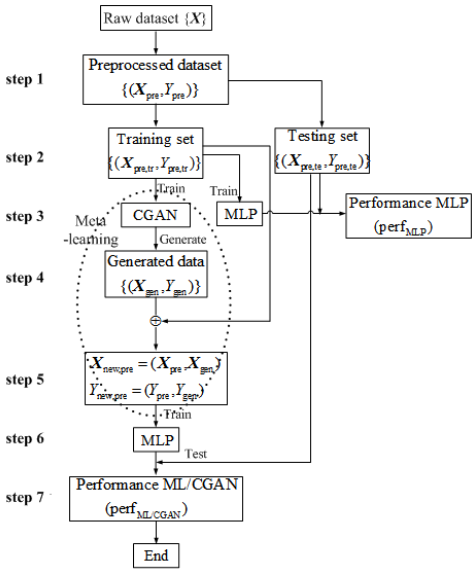

Fig. 1. Deep learning process.

on, we use $\{\boldsymbol{X}\}$ and $\{(\boldsymbol{X}, Y)\}$ instead of $\{\boldsymbol{X}_t\}_{t=1}^{N}$ and $\{(\boldsymbol{X}_t, Y_t)\}_{t=1}^{N}$, respectively, for notational simplicity.

Procedure of Deep Learning

1. **Preprocess a dataset**: Select some components of $\boldsymbol{X}$. Transform the selected components and label by the min-max transformation. Let $\{(\boldsymbol{X}_{\text{pre}}, Y_{\text{pre}})\}_{t=1}^{N}$ be the preprocessed dataset.
2. **Divide training set and testing set:** Divide $\{(\boldsymbol{X}_{\text{pre}}, Y_{\text{pre}})\}_{t=1}^{N}$ into two disjoint subsets $\{(\boldsymbol{X}_{\text{pre,tr}}, Y_{\text{pre,tr}})\}_{t=1}^{N_1}$ and $\{(\boldsymbol{X}_{\text{pre,te}}, Y_{\text{pre,te}})\}_{t=N_1+1}^{N}$, where $N_1 = N \times r$ for initially given training ratio $r \in (0, 1)$.

3-5. Meta-learning process:

3. Train CGAN and MLP by $\{(\boldsymbol{X}_{\text{pre,tr}}, Y_{\text{pre,tr}})\}_{t=1}^{N_1}$. Test $\{(\boldsymbol{X}_{\text{pre,te}}, Y_{\text{pre,te}})\}_{t=N_1+1}^{N}$ with the trained MLP. Obtain perf_{MLP}.
4. Generate fake data $\{\boldsymbol{X}_{\text{gen}}, Y_{\text{gen}}\}^{N_g}$. Y_{gen} can be a label of any classes.
5. Generate a new training set: Obtain $\{(\boldsymbol{X}_{\text{new,pre}}, Y_{\text{new,pre}})\}^{N+N_g}$ by concatenating $\{(\boldsymbol{X}_{\text{gen}}, Y_{\text{gen}})\}^{N_g}$ to $\{(\boldsymbol{X}_{\text{pre}}, Y_{\text{pre}})\}_{t=1}^{N}$.
6. **Train process**: Train MLP by $\{(\boldsymbol{X}_{\text{new,pre}}, Y_{\text{new,pre}})\}^{N+N_g}$.
7. **Test process**: Test $\{(\boldsymbol{X}_{\text{pre,te}}, Y_{\text{pre,te}})\}_{t=N_1+1}^{N}$ with the trained MLP in step 6. Obtain $\text{perf}_{\text{ML/CGAN}}$.

In step 1, we analyze a dataset and extract some components that characterize attacks from the dataset. Let m be the number of components of $\boldsymbol{X}_{\text{pre}}$ for each t. Then $\boldsymbol{X}_{\text{pre}}$ is expressed by $(X_{\text{pre, 1},t}, \cdots, X_{\text{pre},m,t}) \equiv (X_{\text{pre},1}, \cdots, X_{\text{pre},m})$. Preprocess each component of $\boldsymbol{X}_{\text{pre}}$, $X_{\text{pre},i}$, and the corresponding label Y by min-max transformation (MMT), which is given by

$$MMT(p_i) = \frac{p_i - \min(\boldsymbol{P})}{\max(\boldsymbol{P}) - \min(\boldsymbol{P})}, \tag{1}$$

where $\boldsymbol{P}$ can be $\{X_{\text{pre},i}\}_{t=1}^{N}, i = 1, \cdots, m$, and $\{Y\}_{t=1}^{N}$. Note that any classes of data can be generated, and the data contains its label.

B. Details of ML/CGAN in the Deep Learning Process

Fig. 2 illustrates the structure of CGAN used in the meta-learning process briefly. Eqs. (2)–(3), Table I, and the CGAN process provides detailed information for CGAN architecture.

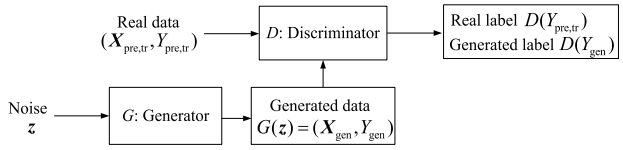

Fig. 2. Structure of CGAN used in the meta-learning process.

Procedure of CGAN in the Meta-Learning

1. **Generate n noise samples** $\boldsymbol{z} = \{\boldsymbol{z}_i\}_{i=1}^{n}$. Each $\boldsymbol{z}_i$ is a vector given by $\boldsymbol{z}_i = (\boldsymbol{z}_{\text{data},i}, z_{l,i})$, where $\boldsymbol{z}_{\text{data},i} = (z_{1,i}, \cdots, z_{m,i})$ and $z_{l,i}$ is a label corresponding to $z_{\text{data},i}$. m is the dimension of X. Each component of z_i is randomly generated uniformly in the interval $[-1,1]$ with pdf $p_{\boldsymbol{z}}$.
2. **Generate fake data** $G(z)$: $G(z)$ is generated through the generator G based on z. Generator loss is computed in this step.
3. **Take** n **samples** $\boldsymbol{x} = \{\boldsymbol{x}_i\}_{i=1}^{n}$ from real data with pdf p_{data}.
4. **Generate** $D(G(z))$ **and** $D(\boldsymbol{x})$**:** $D(G(z))$ and $D(\boldsymbol{x})$ are generated through the discriminator D based on $G(z)$ and $\boldsymbol{x}$, respectively. Discriminator losses are computed in this step.

In the CGAN process, select the noise dimension and the number of generated noise as same as those of input data. That is, m is the number of the extracted components from data in the DL procedure and n is N_1 in step 1. Train generator to minimize the generator loss L_G given in Eq. (2) in step 2. As a result, $G(z)$=$\{\boldsymbol{X}_{\text{gen}}, Y_{\text{gen}}\}$ is generated. Train discriminator to maximize the two discriminator losses, which are computed with $\{\{Y_{\text{pre,tr}}\}_{t=1}^{N_1}, \{D(Y_{\text{pre,tr}})\}_{t=1}^{N_1}\}$ and $\{\{Y_{\text{gen}}\}_{i=1}^{N_1}, \{D(Y_{\text{gen}})\}_{i=1}^{N_1}\}$ in step 4. In the loss functions of CGAN, we use different distance metrics for data and labels, mean square error (MSE) and mean absolute error (MAE), for more accurate loss. Then the generator loss and discriminator loss are computed by

$$\begin{aligned}\min_G L_G = \min_G [&MSE\{G(\boldsymbol{z}_{\text{data}}), \boldsymbol{X}_{\text{pre.tr}}\} \\ &+ \mathbb{E}\{|S(G(z_l)) - Y_{\text{pre,tr}}|\}],\end{aligned} \tag{2}$$

and

$$\begin{aligned}\max_D L_D = \max_D [&\mathbb{E}\{|D(Y_{\text{pre,tr}}) - Y_{\text{pre,tr}}|\} \\ &+ \mathbb{E}\{|D(Y_{\text{gen}}) - Y_{\text{gen}}|\}],\end{aligned} \tag{3}$$

respectively, where $S(G(\text{z}_l))$ is a value that the sigmoid function is applied to $G(\text{z}_l))$.

III. Datasets and Performance Measures

A. Datasets and Feature Vectors

We use the datasets provided by CAIDA and CIDDS-001 in the experiment [19], [20]. The CAIDA dataset contains normal (N) and DDoS attacks (A). It is approximately 2 h traffic of size 23 GB covering from 20:50:08 UTC (Coordinated Universal Time) to 21:56:16 UTC (actual date and time: 20070804, 13:49:36–14:54:36). The attacks are identified around 21:13 UTC with the rapid increase in network load within a few minutes, from 200 Kbits/sec to 90 MB/sec. On the other hand, the CIDDS-001 dataset contains four weeks of external and internal network traffic. We only use

external-week3 traffic that is 1 h data with 153,025 traffic information containing victim (V), normal (N), unknown (U), suspicious (S), and attacks (A). BruteForce and portScan are shown in the dataset.

Based on traffic analysis, we extracted the values that could be evidence of attacks from the raw data in the preprocessing process. The extracted values are four (the numbers of IPs, flows, accumulated packets, and the label) and six (Duration, Src Pt, Dst Pt, Packets, Bytes, and the label) components of the CAIDA and CIDDS-001, respectively. Details for the two datasets, including the selection of feature components, can be found in [7, 3.2] and [13, 4.1], respectively.

B. Performance Measures

To measure the accuracy of prediction, we consider the normalized mean absolute error (NMAE), precision (prec), recall, accuracy, and F1-score based on the confusion matrix for multiclass. Those are defined by

$$\begin{aligned} NMAE &= \sum_{j=N_1+1}^{N} \left\{(X_j - \widehat{X}_j)/X_j\right\} \Big/ (N - N_1), \\ \text{prec}_i &= c_{ii} \Big/ \sum_j c_{ji}, \text{recall}_i = c_{ii} \Big/ \sum_j c_{ij}, \\ \text{accuracy} &= \sum_i c_{ii} \Big/ \sum_{i,j} c_{ij}, \\ \text{and F1-score}_i &= 2(\text{prec}_i \times \text{recall}_i)/(\text{prec}_i \times \text{recall}_i), \end{aligned} \tag{4}$$

where c_{ij} is the *ij*-component of a confusion matrix C and the subscript i in the measures implies i class.

IV. Experimental Results

The experiment was executed by using Python 3.6 and Tensorflow v.1.13.1 on Intel Core i7, 16 GB RAM. The CAIDA dataset was divided into 10-s periods; consequently, 396 subsets were used in the DL process. For CIDDS-001, [V, N] and [U, S, A] are considered as normal and attack, respectively, for binary, while each of V, N, U, S, and A is considered as one class for multiclass. The noise dimension in CGAN was set to the same as the data dimension. Table I describes the used parameters. In the Tables II–V, 'R' and 'L' in ML/CGAN represent the generated label taking a real number in [0, 1] and assigned integer corresponding to the real number, respectively. That is, assign 0 and 1 (N and A) for the generated values in [0,1/2) and [1/2,1], respectively, for binary, while assign integers in [0, 4] (V, N, U, S, and A) for the generated values in [0,1/5), [1/5,2/5), [2/5,3/5), [3/5,4/5), and [4/5,1], respectively, for multiclass. Tables are obtained with $r = 0.7$ and 500 epochs that result in acceptable performance.

Fig. 3 shows the generator and discriminator losses of the datasets for varying epochs. It shows that the two losses for CIDDS-001 and the discriminator loss for CAIDA tend to converge, while the generator loss for CAIDA does not converge up to 1000 epochs. Possible reasons for convergence failure could be the characteristics of the traffic, a small amount of data, and mode collapse/mode drop occurring in CGAN. For example, a big difference in feature vector values between N and A for CAIDA may lead to similar differences in feature vector values between the generated data and real data, which can cause the convergence failure and even affect when mode collapse/mode drop occurred. The differences for CISDDS-001 are observed relatively small compared to those of CAIDA, which seems to cause the convergence of generator loss. The generator loss is best for multiclass of CIDDS-001, while the discriminator losses are almost the same in all cases. Table II compares NMAE. Two epochs were considered to investigate the effect of epochs on performance. As shown in the table, the epoch 500 performs better in most cases.

Tables III–IV present the confusion matrices (A) and the performance measures based on the confusion matrices (B) for binary of the two datasets and multiclass of CIDDS-001, respectively. N/A in Table III (B) means 'not available' caused by zero denominators of precision and recall. The performance with CAIDA seems due to the small size of data rather than the superiority of training techniques. As shown in the confusion matrix in Table IV (A), N, U, and S are predicted accurately, which gives F1-score one for the classes. 2738 of V are misclassified as A, which seems to owe little differences in feature vector values between V and A in bruteForce and portScan [20]. The results in Tables III–IV imply ML/CGAN is superior to MLP for classification. Based on the generator loss of CAIDA in Fig. 3 and the results in Table III, we can infer that the large value and convergence failure of generator loss does not imply unacceptable performance, which is noteworthy. Even though [12]–[15] considered attack

TABLE I
Parameters Used in the Network

Network / Parameter	MLP	ML/CGAN	Network / Parameter	MLP & ML/CGAN
Metric in loss	MSE	MSE(data), MAE(label)	Activation (Hidden layer)	ReLU
Hidden layers	6	G: 6 D: 3	Activation (Output layer)	Sigmoid
Nodes in a layer	20		Optimizer	Adam
Epochs	100, 500		Learning rate	0.001
Batch size	200		r	0.7

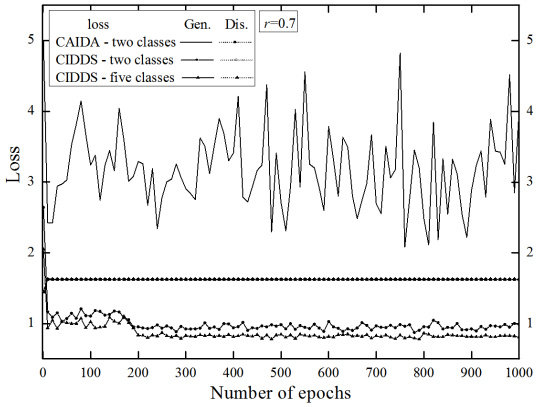

Fig. 3. Comparison of generator and discriminator losses.

TABLE II
Comparison of Obtained NMAE

Training(Ep) / Traffic	MLP (100/500)	ML/CGAN(R) (100/500)	ML/CGAN(L) (100/500)
CAIDA(two classes)	45.90/102.37	3.23/0.99	**0/0**
CIDDS(two classes)	68.89/68.72	26.78/15.86	27.41/**8.97**
CIDDS(five classes)	105.69/95.62	37.58/37.53	**22.11**/23.85

TABLE III
RESULTS FOR TWO DATASETS (TWO CLASSES)

Training	Traffic type		Predicted class			
			CAIDA		CIDDS-001	
			N	A	N	A
MLP	Actual class	N	0	0	14359	0
		A	18	101	31549	0
ML/CGAN (L)	Actual class	N	0	0	30531	3130
		A	0	119	1018	11256

(A) CONFUSION MATRIX

Dataset	Measure	MLP			ML/CGAN(L)		
		N	A	Ave	N	A	Ave
CAIDA	Precision	N/A	1	1	N/A	1	**1**
	Recall	N/A	0.85	0.85	N/A	1	**1**
	F1-score	N/A	0.92	0.92	N/A	1	**1**
	Accuracy	0.85			**1**		
CIDDS -001	Precision	0.31	0	0.1	0.92	0.91	**0.92**
	Recall	1	0	0.31	0.78	0.97	**0.91**
	F1-score	0.48	0	0.15	0.86	0.94	**0.91**
	Accuracy	0.31			**0.91**		

(B) MEASURES

TABLE IV
RESULTS FOR CIDDS-001 (FIVE CLASSES)

Training	Actual class	Predicted class				
		V	N	U	S	A
MLP	V	0	0	0	0	0
	N	1044	208	1410	388	3052
	U	3489	1416	2530	184	9985
	S	1511	844	1773	2657	16507
	A	0	0	0	0	0
ML/CGAN (L)	V	1377	0	0	0	2738
	N	0	580	0	0	0
	U	0	0	1521	0	0
	S	0	0	0	39692	0
	A	0	0	0	0	0

(A) CONFUSION MATRIX

Training	Measure	Class					
		V	N	U	S	A	Ave
MLP	Precision	0	0.08	0.44	0.82	0	0.59
	Recall	0	0.03	0.15	0.11	0	0.12
	F1-score	0	0.05	0.23	0.20	0	0.19
	Accuracy	0.12					
ML/CGAN (L)	Precision	1	1	1	1	0	**1**
	Recall	0.33	1	1	1	0	**0.94**
	F1-score	0.5	1	1	1	0	**0.96**
	Accuracy	0.94					

(B) MEASURES

analyses based on GAN, the performance measures used in the studies are different from ours except [14]. Table V compares our results with those in [14]. The multiclass performance of CIDDS-001 can also be compared with the results in [8], where k-nearest neighborhood classification and k-means clustering algorithms were considered for the dataset. The accuracy, precision, and F1-score in [8] are 0.993∼0.996, 0.982∼1, and 0.99∼1, respectively, depending on k.

V. CONCLUSION

A DL technique that includes ML/CGAN was proposed to overcome the network traffic data shortage. The study generated data including labels and differentiated the distance metric used in CGAN depending on the characteristics of data. In addition to improving performance, this study suggests several discussions on the use of GAN in network attack analysis, such as the relationship between the generator loss of a GAN and its performance. Further study shall follow to find the optimal number of concatenations and theoretical verification.

TABLE V
COMPARISON WITH EXISTING RESULTS

Comparison	Data	Model	Precision	Recall	F1-score
this study	CAIDA	MLP	1	0.85	0.92
		ML/CGAN(L)	**1**	**1**	**1**
	CIDDS-001	MLP	0.59	0.12	0.19
		ML/CGAN(L)	**1**	**0.94**	**0.96**
[14] (2019)	KDDCUP99	MAD-GAN (chosen by best precision)	0.94	0.19	0.32
	SWaT		0.99	0.54	0.70
	WADI		0.46	0.24	0.32

REFERENCES

[1] M. Amiri and L. Mohammad-Khanli, "Survey on prediction models of applications for resources provisioning in cloud," *J. Netw. Comput. Appl.*, vol. 82, pp. 93–113, Mar. 2017.

[2] M. Laner, P. Svoboda, and M. Rupp, "Parsimonious fitting of long-range dependent network traffic using ARMA models," *IEEE Commun. Lett.*, vol. 17, no. 12, pp. 2368–2371, Dec. 2013.

[3] M. Kim, "Network traffic prediction based on INGARCH model," *Wireless Netw.*, vol. 26, no. 8, pp. 6189–6202, Nov. 2020.

[4] R. Li, Z. Zhao, J. Zheng, C. Mei, Y. Cai, and H. Zhang, "The learning and prediction of application-level traffic data in cellular networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 6, pp. 3899–3912, Jun. 2017.

[5] C. Qiu, Y. Zhang, Z. Feng, P. Zhang, and S. Cui, "Spatio-temporal wireless traffic prediction with recurrent neural network," *IEEE Wireless Commun. Lett.*, vol. 7, no. 4, pp. 554–557, Aug. 2018.

[6] I. Alawe, A. Ksentini, Y. Hadjadj-Aoul, and P. Bertin, "Improving traffic forecasting for 5G core network scalability: A machine learning approach," *IEEE Netw.*, vol. 32, no. 6, pp. 42–49, Nov. 2018.

[7] M. Kim, "Supervised learning-based DDoS attacks detection: Tuning hyperparameters," *ETRI J.*, vol. 41, no. 5, pp. 560–573, Sep. 2019.

[8] A. Verma and V. Ranga, "Statistical analysis of CIDDS-001 dataset for network intrusion detection systems using distance-based machine learning," *Procedia Comput. Sci.*, vol. 125, pp. 709–716, Dec. 2018.

[9] J. Vanschoren, "Meta-learning: A survey," 2018, *arXiv:1810.03548*. [Online]. Available: http://arxiv.org/abs/1810.03548

[10] C.-I. Kim, M. Kim, S. Jung, and E. Hwang, "Simplified Fréchet distance for generative adversarial Nets," *Sensors*, vol. 20, p. 1548, pp. 1–27, Mar. 2020.

[11] A. Cheng, "PAC-GAN: Packet generation of network traffic using generative adversarial networks," in *Proc. IEEE 10th Annu. Inf. Technol., Electron. Mobile Commun. Conf. (IEMCON)*, Oct. 2019, pp. 728–734.

[12] D. Sun, K. Yang, Z. Shi, and C. Chen, "A new mimicking attack by LSGAN," in *Proc. IEEE 29th Int. Conf. Tools with Artif. Intell. (ICTAI)*, Nov. 2017, pp. 441–447.

[13] M. Ring, D. Schlör, D. Landes, and A. Hotho, "Flow-based network traffic generation using generative adversarial networks," 2018, *arXiv:1810.07795*. [Online]. Available: http://arxiv.org/abs/1810.07795

[14] D. Li, D. Chen, L. Shi, B. Jin, J. Goh, and S.-K. Ng, "MAD-GAN: Multivariate anomaly detection for time series data with generative adversarial networks," 2019, *arXiv:1901.04997*. [Online]. Available: http://arxiv.org/abs/1901.04997

[15] J. Charlier, A. Singh, G. Ormazabal, R. State, and H. Schulzrinne, "SynGAN: Towards generating synthetic network attacks using GANs," 2019, *arXiv:1908.09899*. [Online]. Available: http://arxiv.org/abs/1908.09899

[16] C. Lemke and B. Gabrys, "Meta-learning for time series forecasting and forecast combination," *Neurocomputing*, vol. 73, nos. 10–12, pp. 2006–2016, Jun. 2010.

[17] D. Afolabi, S.-U. Guan, K. Man, P. Wong, and X. Zhao, "Hierarchical meta-learning in time series forecasting for improved interference-less machine learning," *Symmetry*, vol. 9, no. 11, p. 283, Nov. 2017.

[18] M. Naruse, T. Matsubara, N. Chauvet, K. Kanno, T. Yang, and A. Uchida, "Generative adversarial network based on chaotic time series," *Sci. Rep.*, vol. 9, no. 1, Sep. 2019, Art. no. 012963.

[19] *The CAIDA UCSD Anonymized Internet Traces*. Accessed: Sep. 25, 2020. [Online]. Available: http://caida.org/data/passive/passive_dataset.xml

[20] *CIDDS-001 Dataset*. (Aug. 2017). [Online]. Available: https://www.hs-coburg.de/forschung/forschungsprojekte-oeffentlich/informationstechnologie/cidds-coburg-intrusion-detection-data-sets.html