A Data-centric Analysis to Evaluate Suitable Machine-Learning-based Network-Attack Classification Schemes

Truong Thu Huong[†], Ta Phuong Bac[††], Bui Doan Thang[†], Dao Minh Long[†], Le Anh Quang[†], Nguyen Minh Dan[†], Nguyen Viet Hoang[†]

[†]Hanoi University of Science and Technology, Hanoi, Vietnam
[††]Soongsil University, Seoul 06978, Korea

Summary
Since machine learning was invented, there have been many different machine learning-based algorithms, from shallow learning to deep learning models, that provide solutions to the classification tasks. But then it poses a problem in choosing a suitable classification algorithm that can improve the classification/detection efficiency for a certain network context. With that comes whether an algorithm provides good performance, why it works in some problems and not in others. In this paper, we present a data-centric analysis to provide a way for selecting a suitable classification algorithm. This data-centric approach is a new viewpoint in exploring relationships between classification performance and facts and figures of data sets.

Key words:
Machine learning, deep learning, shallow learning, datasets.

1. Introduction

Data and model are essential components that play a significant role in building an artificial intelligence (AI) system applied in many different fields such as telecommunication, cyber-security, recommendation systems, web, finance, ecology, biology, etc. Conventional machine learning algorithms learn from training data and make predictions by optimizing model parameters, which is an iterative process. The training data is processed multiple times, and the model parameters update iteratively. In this iterative process, there are two primary directions to achieve a good AI solution: the model-centric and data-centric approach. A model-centric view involves designing empirical tests around the model to improve performance and finding the exemplary model architecture and training process in an ample space of possibilities.

In contrast, the data-centric approach involves systematically changing or enhancing the quality of the training data sets to improve AI system performance. In the AI community, most research efforts focus on model-centric AI by developing new algorithms to replace existing algorithms or using a heuristic approach from empirical knowledge to choose a suitable algorithm [1] . In paper [1] , the author proposed a method to select the best algorithms from candidate algorithms using a heuristic approach, automatically performed by theoretically analyzing the applicability of these classification algorithms. However, such theory-based methods require more background knowledge from a more specific field. Furthermore, not all possible applications of classification algorithms are possibly practically feasible.

In the same direction, Brazdil et al. [2] combine statistics and information theory measures to extract meta-features for different data sets. These features serve the execution process on many algorithms to determine the algorithm's applicability. The final decision is made based on a rule-based model with a set of rules.

Meta-learning [3] is a framework developed in the field of supervised machine learning with the aim of supporting solving important problems in the field of machine learning as classifier selection [4] . This method will provide a meta-learning framework to map an existing problem or task to one or several algorithms that are best suited to solve the problem.

However, the fact that there are many available classification algorithms creates a challenge for users in choosing and using an appropriate algorithm for their attack classification task. Therefore, the challenge is to explore the relationship of the performance of the algorithms with the characteristics of the training datasets and then choose an algorithm that best fits the data. This is a direction that has received the attention of many researchers many years ago [5] [6] [7] . However, these methods need to be updated and adapted to the wide popularity and development of artificial intelligence-based technologies and the increasing number of machine learning algorithms and the number of forms.

According to the “No Free Lunch” theory [8] , in the machine learning context, that has depicted that there is no single algorithm applicable to all different data sets. This theory implies that the performances of all algorithms are equally good if tested over a sufficiently large number of datasets. In other words, there is not a magic algorithm that is the best in every situation, and no general recommendations can be made for arbitrary data. Data characteristics or dataset meta-features always affect the actual performance of a classifier.

In this paper, we do not propose a set of classifiers to choose the best algorithm for the classification problem. Instead, the content of the study provides readers with a new

Manuscript received June 5, 2021
Manuscript revised June 20, 2021
https://doi.org/10.22937/IJCSNS.2021.21.6.23

data-centric analysis to support choosing a classification algorithm suitable for a certain type of training data being used. We review some of the available classification algorithms that can be applied in cybersecurity; and the results of analysis, evaluation, and conclusions are made on the dataset of actual cyber-attacks.

The rest of our paper is organized as follows: Background on data characteristics that elaborates data parameters used in our study are provided in Section 2. In Section 3, we elaborate how we set up our experiments in terms of a machine learning algorithms and training data sets. Then the facts and figures on data characteristics as well as detection performance are given and analyzed. Finally, some discussion is given in Section 4.

2. Background on data characteristics

To better understanding why an algorithm has better performance on a dataset over other algorithms, we need to examine the characteristics of a dataset. This paper introduces three types of data characteristics: Statistical measures, Discriminated measures and Information-theoretic measures.

2.1. Statistical measures

Skewness is the measure of the asymmetry of a probability distribution. A variable with normal distribution will have 0 skewness. On the other hand, a positive or negative skewness indicates the most probable value will be on the left or right-hand side, respectively. Skewness is calculated as followed:

$$Skew = \sum_{i=1}^{N} \frac{|x_i - m|^3}{N * \delta^3}$$

Kurtosis is the measure of whether a variable has heavy-tailed or light-tailed distribution. Larger kurtosis means heavier tail and less concentrated central. A normal distribution variable has zero kurtosis. Kurtosis is calculated as:

$$Kurt = \sum_{i=1}^{N} \frac{(x_i - m)^4}{N * \delta^4} - 3$$

Where:

- N is the sample size
- x_i is the sample value of a variable
- m is the average value of that variable
- δ is the standard deviation

Mean absolute correlation coefficient: The last statistical measure is the mean absolute correlation coefficient. The correlation is a metric calculated between two variables and represents the linearity of their relationship. The closer the absolute value is to 1, the stronger the correlation between the two variables. The mean absolute correlation coefficient is the absolute value of the correlations averaged over all pairs of features and all classes. In this research, we use Pearson's Correlation Coefficient, which can be computed as:

$$PCC = \frac{\sum_{i=1}^{N}(x_i - \mu_x)(y_i - \mu_y)}{\sqrt{\sum_{i=1}^{N}(x_i - \mu_x)^2} * \sqrt{\sum_{i=1}^{N}(y_i - \mu_y)^2}}$$

Where x_i and y_i are individual datapoints of the two variables and μ_x and μ_y are their mean, respectively.

2.2. Discriminant measures

The discriminant measures are based on Linear Discriminant Analysis, in which the classifier tries to transform the data to another dimension where it would be easier to classify the data by minimizing within-between variance ratio. The three measures are:

Within-class variance

$$s_W = trace(\boldsymbol{W}^T \boldsymbol{S}_W \boldsymbol{W})$$

Between class variance

$$s_B = trace(\boldsymbol{W}^T \boldsymbol{S}_B \boldsymbol{W})$$

Within/between class variance ratio

$$R = \frac{s_W}{s_B}$$

Where:

- $\boldsymbol{S}_W = \sum_{k=1}^{C} \sum_{n \epsilon \mathcal{C}_k} (\boldsymbol{x}_n - \boldsymbol{m}_k)(\boldsymbol{x}_n - \boldsymbol{m}_k)^T$,
- $\boldsymbol{S}_B = \sum_{k=1}^{C} N_k (\boldsymbol{m}_k - \boldsymbol{m})(\boldsymbol{m}_k - \boldsymbol{m})^T$,
- $\mathbf{W} = arg\max_{\mathbf{W}} R$
- C : the number of classes.
- $\mathcal{C}_k$: the indexes of all the datapoints in class k.
- N_k : the number of datapoints in class k.
- $\boldsymbol{x}_n$: a datapoint with index n.
- $\boldsymbol{m}$: the mean vector of all data.
- $\boldsymbol{m}_k$: the mean vector of all data in class k.

2.3. Information theoretic measures

The following information theoretic measures [19] are normally used for categorical or discreate value. To use the

following characteristics for continuous cases, we can divide the variable value into a number of partitions and treat the partition as a discreate value.

***Entropy of attributes**:*

$$H_X = -\sum q_i * \log(q_i)$$

Where: q_i is the probability of the ith possible value

***Entropy of class**:*

$$H_C = -\sum c_i * \log(c_i)$$

Where: c_i is the probability of the ith class.

***Joint entropy of class and attribute**:*

$$H_{C,X} = -\sum_{i,j} p_{ij} * \log(p_{ij})$$

Where p_{ij} is the probability that attribute X has ith value and belong in *j*th class.

Mutual information of class and attribute:

$$M_{C,X} = H_C + H_X - H_{C,X}$$

Equivalent number of attributes:

$$EqAttr = \frac{H_c}{\bar{M}_{C,X}}$$

Noisiness of attributes:

$$Noisiness = \frac{\bar{H}_X}{\bar{M}_{C,X}} - 1$$

3. Performance of anomaly detection schemes

3.1 Experimental setup on machine learning-based detection algorithms

To evaluate the effects of various characteristics to the classification process, we run 13 Machine-learning algorithms on top of those 25 data sets (the datasets will be elaborated in the following subsection). The 13 algorithms include both shallow-model algorithms and deep-model algorithms, as follows:

- Shallow model: Decision tree, Linear Discriminant Analysis, K-Nearest Neighbors, Support Vector Machine, Shallow Neural Network, Naïve Bayes, Logistic Regression
- Deep model: Bidirectional-recurrent neural network, Gated Recurrent Units, Long Short-term Memory, Deep Neural Network, Convolutional Neural Network, Deep Belief Network.

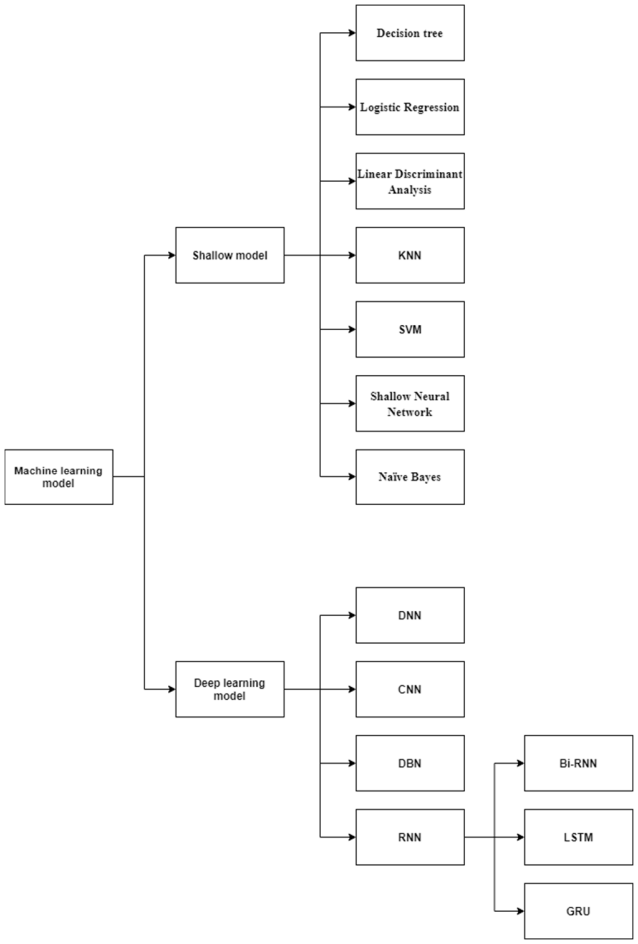


Fig. 1: Taxonomy of the ML algorithms used in our analysis

3.2 Experimental setup on Data sets

To evaluate and assess detection performance of the 13 machine-learning-based detection schemes, originally, 7 different datasets of network traffic collected from IoT devices are selected as follows:

– IoTID20 [9] : a dataset stems from a IoT environment of two typical smart home devices being attacked by other devices in the network. This csv-formatted dataset includes 83 network features along with label feature of normal and eight different attack types, ranging from Denial of Service, Mirai, Man in the Middle to Scan.

- N-BaIoT [10] : a dataset about a network constitutes IoT devices in enterprise context. In this dataset, a total of 115 features about traffic statistics over 5 temporal windows were extracted when two common IoT botnets, namely BASHLITE and Mirai, were injected into the network to execute multiple attacks.

- Kitsune [11] : a NIDS dataset with 115 statistical features over different time windows containing traffic of two networks. A surveillance network consists of two separate four-camera clusters, nine different attack scenarios of type Recon, Man in the Middle, Denial of Service and Botnet Malware were recorded. In the second network, this is an IoT network of 9 IoT devices and 3 personal computers in which Mirai botnet malware attack was conducted.

- MQTTset [12] : an IoT cyber-security dataset with 60 features that focuses on communications using the Message Queue Telemetry Transport (MQTT) protocol. The network consists of 8 sensors, each of which directly communicates with the MQTT broker to simulate a smart house scenario. MQTTset includes legitimate network data and five categories of MQTT-based cyber-attacks.

- MQTT-IoT-IDS2020 [13]: an IoT dataset simulating a realistic MQTT IoT network, which includes 12 sensors, an MQTT broker, a camera feed server, and an attacker machine. MQTT-IoT-IDS2020 consists of 1 normal scenario and 4 attack scenarios were collected separately with the normal scenario in the background.

- TON_IoT [14] : a telemetry dataset in the context of Industrial IoT, derived from a medium-scale IoT testbed with seven IoT and Industrial IoT sensors. The dataset includes normal network data and 9 types of cyber-attacks, which were executed against various sensors across the network.

- UNSW-NB15 [15] : a dataset created for Network Intrusion Detection Systems, addressing the problems of the existing datasets such as duplicated samples, unrepresentative training sets and test sets, and the lack of low-footprint cyberattacks. 100 GBs of hybrid (normal and attack scenarios) network data was generated and passed through feature extraction tools to create the dataset.

- CIC-IDS2017[16] : a NIDS dataset generated from and captured in a comprehensive dataset infrastructure, which was separated into two distinct networks: the Victim-Network and the Attack-Network. The B-Profile system was utilized to emulate realistic benign traffic in the background while one of six types of cyberattacks was being executed against the Victim-Network.

- CIC-DDoS2019 [17] : a dataset generated from an analogous two-network testbed as CIC-IDS2017 with differences of Victim-Network consisting of one server, one firewall, two switches and four PCs while Attack-Network deployed by third party tools executing a disparate group of 12 DDoS attacks. The subsequent dataset comprises 80 different flow-based features extracted using the CICFlowMeter tool [18] .

To investigate the relationship between the characteristics of a data set and the detection performance (e.g accuracy, F1-score, Precision, Recall, AUC) of the 13 popular ML-based detection algorithms aforementioned. We increase the number of datasets and diversify the data characteristics in our experiments by splitting the 9 original datasets into 25 smaller ones. Each dataset has its own characteristics in basic measures such as the number of samples (from 8125 to 242787 samples), the number of classes (3-8 classes), and in other measures (such as statistic measures, discriminant measures, etc.)

3.3. Experimental setup on detection parameters

In this subsection, we describe a set of metrics used in our evaluation of machine learning algorithms' classification performance. The first metric is training time, which is the duration a specific algorithm uses for the model building phase, measured in minutes.

The second parameter is Accuracy, which is the fraction of total samples correctly classified by classifier model:

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions}$$

The third metric used is macro-averaged precision calculating the average precision, in which each class is treated equally to others:

$$Macro\ Precision = \frac{1}{N}\sum_{i=0}^{N-1} Precision_i$$

where N is the total number of classes and i is class index.

Likewise, macro-averaged recall and macro-averaged F1-score are computed by averaging recall and F1-score of each individual class, respectively:

$$Macro\ Recall = \frac{1}{N}\sum_{i=0}^{N-1} Recall_i$$

$$Macro\ F1\ Score = \frac{1}{N}\sum_{i=0}^{N-1} F1\ Score_i$$

where *N* is the total number of classes and *i* is class index.

In each class, the corresponding metrics are calculated as follows:

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

$$F1Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Where:

- *TruePositive*: number of outcomes correctly predicted as positive.
- *FalsePositive*: number of wrong predictions of actual negative as positive.
- *FalseNegative*: number of wrong predictions of actual positive as negative.

The final metric used for this performance analysis is Area Under the Receiver Operating Characteristic Curve (i.e. AUC), computed in both micro-averaged and macro-averaged manner. While micro-averaged is computed by treating each element of the label indicator matrix as a binary prediction, macro-averaged gives equal weight to the classification of each label as aforementioned calculations. These metrics computation will help us to assess the overall performance of each machine algorithm over a collection of datasets in the following sections.

3.4 Performance Analysis

3.4.1 Findings on dataset characteristics

Table 1 shows the characteristics of the statistic measures (e.g. Skewness, Kurtosis, Correlation Coefficient) and discriminant measures (Within-class variance, Between Class variance, BW class variance ratio) of the 25 datasets in our experiment.

While Table 2 shows the findings on the Information theoretic measures (e.g. Entropy of classes, Entropy of attributes, Mutual information of classes and attributes, Equivalent number of attributes, Noisiness of attribute) and basic measures (e.g. Number of samples, Number of attributes, Number of classes) of those datasets.

Table 1: Statistic measures and discriminant measures

	Statistic Measures			Discriminant Measures		
Dataset	Mean Absolute Skewness	Mean Kurtosis	Mean Absolute Correlation Coefficient	Within Class Variance	Between Class Variance	BW Class Variance Ratio
1	14.1568	2375.93	0.383013	102.028	8.372598	0.082062
2	3.470468	169.5141	0.34325	83.65785	8.244138	0.098546
3	13.10017	552.5776	0.249309	159645.5	16121.32	0.100982
4	43.2335	17777.06	0.312516	2.69661	0.255143	0.094616
5	33.82218	5370.224	0.242386	277812.5	27568.15	0.099233
6	9.656811	1064.526	0.377135	78.26708	6.406469	0.081854
7	2.900592	160.436	0.353527	21.6827	2.17989	0.100536
8	12.30063	423.8788	0.272488	51359.67	5204.035	0.101325
9	5.46019	536.5678	0.328734	1.25E-10	1.18E-11	0.09484
10	21.92411	1940.541	0.258087	61195.58	6072.069	0.099224
11	31.12854	5703.112	0.252651	289149.2	28714.34	0.099306
12	4.47841	63.14469	0.329684	17555.54	1351.986	0.077012
13	78.96164	15762.92	0.14306	25950.51	2379.746	0.091703
14	80.41875	16583.36	0.155178	18545.02	1747.006	0.094203
15	46.64629	8920.242	0.468853	1.34E+10	1.08E+09	0.08033
16	22.22354	1882.079	0.458282	1.91E+09	1.77E+08	0.092579
17	16.77525	1395.422	0.228164	538095.5	50158.75	0.093215

18	13.98218	613.0596	0.263987	96426.79	7051.234	0.073125
19	32.14089	4032.785	0.462566	43194.88	5426.19	0.125621
20	25.56158	3918.489	0.373687	1.33E+10	1657462	0.000124
21	15.45524	657.5736	0.441197	826870.8	4546.162	0.005498
22	33.27938	6326.76	0.275952	364848.9	35669.1	0.097764
23	35.74576	7981.264	0.245051	885407.2	59120.89	0.066773
24	32.50158	4801.343	0.247711	7360406	994298.8	0.135087
25	33.92826	4693.048	0.208794	281955.3	16164.94	0.057332

Table 2: Information theoretic measures and basic measures

	Information Theoretic Measures					**Basic Measures**		
Dataset	**Entropy of Classes**	**Entropy of Attributes**	**Mutual Information of Classes and Attributes**	**Equivalent Number of Attributes**	**Noisiness of Attribute**	**Number of Samples**	**Number of Attributes**	**Number of Classes**
1	1.604684	1.784163	0.460912	3.48154	2.87094	217573	115	5
2	2.107886	1.812993	0.784671	2.686332	1.310515	203095	115	5
3	2.121286	0.287287	1.248589	1.698946	-0.76991	200532	115	5
4	2.232657	0.184393	1.248475	1.788308	-0.85231	200130	115	5
5	2.321928	0.290485	1.014079	2.289691	-0.71355	200000	79	5
6	1.612421	1.946768	0.554126	2.909845	2.513222	54392	115	5
7	2.297753	1.772714	0.83368	2.756158	1.126372	51216	115	5
8	2.122432	0.288038	1.2495	1.698625	-0.76948	50062	115	5
9	2.230321	0.18498	1.24831	1.786672	-0.85182	50002	115	5
10	2.321928	0.290527	1.013625	2.290716	-0.71338	50000	79	5
11	2.321928	0.305334	1.043036	2.226125	-0.70726	200000	79	5
12	2.255648	0.232037	0.143471	15.72202	0.617314	74627	41	5
13	2.321928	0.180679	0.207006	11.2167	-0.12718	100000	85	5
14	2.321928	0.15053	0.18767	12.37239	-0.1979	100000	85	5
15	2.204004	0.224583	1.446916	1.523243	-0.84479	111001	29	5
16	2.263019	0.241172	1.594354	1.419395	-0.84873	169339	16	5
17	2.207921	0.423122	1.200392	1.839333	-0.64751	185124	41	5
18	1.852082	0.427514	0.946719	1.956317	-0.54843	65346	42	5
19	1.378527	0.382633	0.681698	2.022197	-0.43871	33820	78	3
20	1.875601	0.433882	0.982421	1.909162	-0.55835	88577	78	6
21	1.126466	0.391177	0.477067	2.361232	-0.18004	8215	78	5
22	1.467821	0.374083	0.786108	1.867201	-0.52413	142125	78	4
23	2.713882	0.163252	0.984771	2.755851	-0.83422	242787	81	8
24	1.2827	0.11735	0.479144	2.677065	-0.75508	60053	82	3
25	1.727957	0.117116	0.60713	2.846105	-0.8071	58418	81	4

Table 3: Accuracy of 13 ML-based detection algorithms over 25 datasets

Dataset	KNN	SVM	DT	SNN	LDA	NB	LR	BRNN	GRU	LSTM	CNN	DNN	DBN
1	0.947	0.9345	0.999	0.9402	0.8728	0.573	0.9315	0.9631	0.9674	0.9534	0.9667	0.9471	0.8182
2	0.9865	0.9645	0.9996	0.9904	0.9462	0.8224	0.9548	0.9666	0.9682	0.9665	0.9817	0.9479	0.8898
3	0.9993	0.9234	0.9997	0.985	0.8773	0.8766	0.9125	0.9806	0.9916	0.9871	0.989	0.9926	0.8913
4	0.9985	0.9938	0.9998	0.9998	0.983	0.7994	0.9946	0.9963	0.9969	0.996	0.9996	0.9909	0.7812
5	0.7463	0.6796	0.74	0.7549	0.6653	0.6313	0.6794	0.7697	0.7718	0.7694	0.7657	0.7699	0.6028
6	0.9521	0.894	0.9982	0.9627	0.8921	0.6202	0.9498	0.9505	0.9306	0.9407	0.9585	0.9126	0.773
7	0.9811	0.9671	0.9986	0.9641	0.944	0.8596	0.9573	0.9466	0.9602	0.9593	0.9617	0.9443	0.8661
8	0.9963	0.912	0.9989	0.9889	0.8831	0.8735	0.9055	0.987	0.9842	0.9824	0.9873	0.9848	0.8694
9	0.9923	0.9892	0.9995	0.9932	0.9834	0.7054	0.9909	0.9942	0.9947	0.9951	0.9982	0.9843	0.7753
10	0.759	0.6734	0.7637	0.7243	0.6699	0.6271	0.68	0.7497	0.7446	0.7458	0.7335	0.7546	0.4654
11	0.92	0.8493	0.919	0.9055	0.8227	0.8179	0.8535	0.9192	0.9165	0.9259	0.9252	0.9171	0.7779
12	0.815	0.669	0.7313	0.6721	0.6671	0.5904	0.6693	0.6631	0.6604	0.6626	0.6627	0.6627	0.6077
13	0.9623	0.9457	0.988	0.9548	0.9265	0.5268	0.9464	0.9601	0.9628	0.9639	0.9528	0.9541	0.9162
14	0.9809	0.8467	0.9983	0.9638	0.8214	0.6781	0.8606	0.9564	0.9659	0.9634	0.9674	0.9574	0.8169
15	0.9956	0.993	0.9977	0.9957	0.9868	0.9955	0.9914	0.9939	0.996	0.9948	0.9957	0.9955	0.9939
16	0.9967	0.9896	0.9982	0.9948	0.9894	0.9935	0.9939	0.9936	0.9938	0.9942	0.9937	0.9926	0.994
17	0.7825	0.7594	0.806	0.8047	0.7272	0.6003	0.7658	0.8063	0.8168	0.8117	0.8092	0.8085	0.7477
18	0.8321	0.7925	0.8319	0.8524	0.7471	0.4128	0.7912	0.8701	0.8704	0.8711	0.8373	0.8582	0.6841
19	0.9981	0.972	0.9998	0.9971	0.9451	0.9934	0.9658	0.9985	0.9987	0.9982	0.9991	0.9935	0.8118
20	0.9972	0.965	0.9977	0.996	0.9379	0.9189	0.9601	0.9968	0.9977	0.9976	0.9976	0.9971	0.8986
21	0.912	0.9051	0.9108	0.9201	0.8848	0.8122	0.8961	0.9836	0.9726	0.9811	0.9653	0.9757	0.7292
22	0.9956	0.9713	0.9975	0.9927	0.9637	0.8605	0.9718	0.9992	0.9991	0.9976	0.9985	0.9956	0.9543
23	0.9017	0.8416	0.9033	0.9147	0.8178	0.7912	0.8997	0.9947	0.9913	0.9951	0.9951	0.995	0.9157
24	0.9994	0.9993	0.9997	0.9996	0.9979	0.9987	0.9992	0.9998	0.9998	0.9997	0.9998	0.9998	0.9993
25	0.9975	0.9985	0.9993	0.9987	0.9345	0.8195	0.9966	0.9984	0.9991	0.9991	0.9991	0.9994	0.9835

3.4.2. Analysis on detection metrics

Accuracy

The accuracy of the 13 algorithms over the 25 datasets is shown in Table 3. In most of the cases, KNN, DT, SNN and the deep neural model algorithms have significantly better performance than the other algorithms, so we decide to divide the algorithms into 2 groups: the shallow model consisting of KNN, DT and SNN; and the deep model consisting of CNN, DNN, BRNN, GRU and LSTM. We calculate the average accuracy of detection algorithms of each group for 25 datasets, and the accuracy difference of the 2 groups for each dataset. If the accuracy difference value is greater than 0, the shallow model is better; and if it is less than 0, then the deep model is better.

Fig. 2 shows the effect of within-class variance on accuracy difference between the shallow model and deep model group of algorithms. When the within-class variance ranges from 10^1 to $10^{4.5}$, the shallow model has better accuracy by up to 8%. But when the within-class variance continues to increase to the range of $10^{7.5}$-10^9, the deep model algorithms outperform in terms of accuracy. It means, we'd better to apply a detection algorithm of the shallow model group to a dataset that has within-class variance in the range of [10^1 to $10^{4.5}$] to achieve better accuracy.

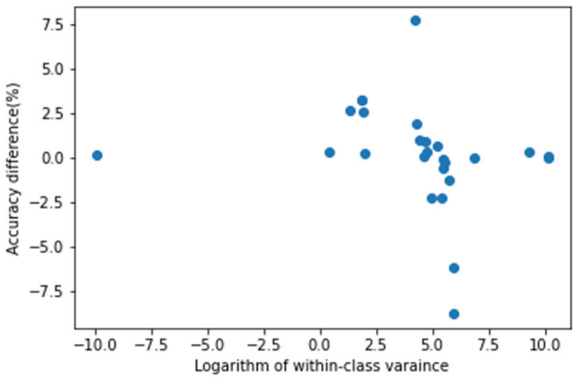


Fig. 2: Effect of within-class variance on detection accuracy

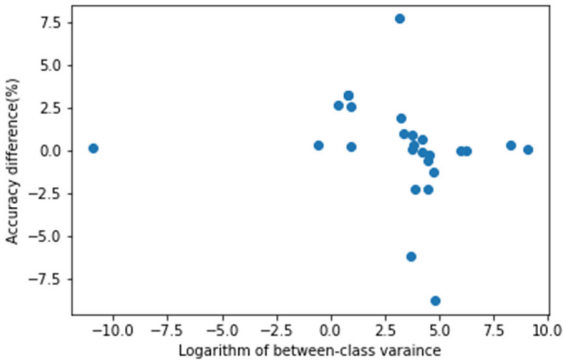


Fig. 3: Effect of between-class variance on accuracy difference

The findings above can also be applied for between-class variance attribute of the datasets. As seen in **Fig. *3***, we should apply detection algorithms of the deep model group if the training dataset has between-class variance range from 10^4 to 10^6-10^8. While the shallow model algorithms should be applied to achieve better accuracy if the dataset has the between-class variance values ranging from 10^0 to 10^4.

Precision, Recall, F1-score, AUC

In this section, we want to investigate how Precision, Recall, F1-score and AUC of different detection algorithms vary over the 25 datasets. Fig. 4 shows the mean performance metrics of 13 algorithms overall 25 datasets. From left to right are the plot of accuracy, macro precision, macro recall, macro F1-score, macro AUC and micro AUC.

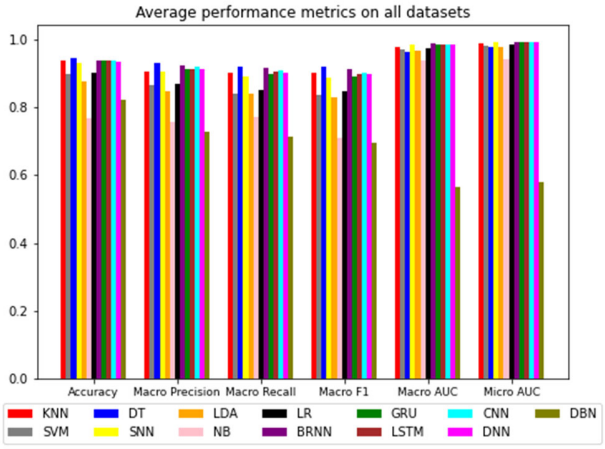


Fig. 4: Average performance of the 13 algorithms over all datasets

These performance metrics can also be described numerically in Table 4 and Table 5 that show the mean macro precision and mean macro recall values of the 13 algorithms, respectively. It can be seen that Decision tree algorithm always achieves the highest score, followed by Bi-current neural network in second place. The Deep learning model group (i.e. Long Short-Term Memory, Convolutional Neural Network, Deep Neural Network for Recall score and also Gated Recurrent Units for Precision score) in third place. K-Nearest Neighbors gets the fourth place.

Table 4: Mean macro precision

Algorithms	Mean macro precision	Standard deviation
KNN	0.906334	0.12429
SVM	0.865944	0.126722
DT	0.928545	0.114905
SNN	0.905582	0.113445
LDA	0.846959	0.120341
NB	0.754938	0.170322
LR	0.867847	0.141994
BIRNN	0.9249	0.095618
GRU	0.912585	0.104065
LSTM	0.910792	0.102177
CNN	0.918294	0.110328
DNN	0.913217	0.101871
DBN	0.72792	0.208663

Table 5: Mean macro recall

Algorithms	Mean macro recall	Standard deviation
KNN	0.902046	0.13998
SVM	0.839384	0.164799
DT	0.91879	0.128607
SNN	0.892313	0.153098
LDA	0.840608	0.140665
NB	0.769796	0.160368
LR	0.849119	0.160558
BIRNN	0.917347	0.125257
GRU	0.897161	0.135103
LSTM	0.905939	0.12934
CNN	0.906886	0.140209
DNN	0.901669	0.134277
DBN	0.712761	0.211599

With respect to F1-score, as shown in Table 6, the Decision tree algorithm (DT) also has the highest mean macro F1 score (92%), followed by Bi-current neural network (91%), K-nearest neighbors and convolutional neural network (90%).

Table 6: Mean macro F1 score

Algorithms	Mean macro F1	Standard deviation
KNN	0.902045	0.135833
SVM	0.835152	0.16631
DT	0.921015	0.124675
SNN	0.885644	0.151851
LDA	0.828643	0.143769
NB	0.711445	0.186051
LR	0.847228	0.162234
BIRNN	0.912403	0.123942
GRU	0.891755	0.135984
LSTM	0.89846	0.127872
CNN	0.901051	0.141848
DNN	0.897381	0.132782
DBN	0.693995	0.222658

With respect to the AUC score, except Deep Belief Network, the neural-network-based models (including both shallow models and deep models) provide the best results: 98% for macro average and 99% for micro average. K-Nearest Neighbors, Logistic Regression and Support Vector Machine, Linear Discriminant Analysis and Decision Tree follow in the descending order.

Table 7: AUC score

	Average AUC macro	AUC macro standard deviation	Average AUC micro	AUC micro standard deviation
KNN	0.975734	0.041656	0.987502	0.01905
SVM	0.971825	0.034548	0.982436	0.024198
DT	0.964422	0.058582	0.977233	0.034675
SNN	0.983121	0.028601	0.990764	0.018357
LDA	0.967906	0.038474	0.977425	0.028059
NB	0.939083	0.052421	0.941202	0.061316
LR	0.973842	0.035182	0.984041	0.02464
BIRNN	0.987053	0.023581	0.992149	0.017022
GRU	0.985026	0.026885	0.992042	0.017875
LSTM	0.983709	0.030532	0.991774	0.018853
CNN	0.984182	0.030828	0.991099	0.019323
DNN	0.984799	0.028502	0.99142	0.01819
DBN	0.565444	0.130193	0.578729	0.166229

From the results shown from Table 3 and Fig. 4 to Table 7, we conclude that among of all classification algorithms, Deep Belief Network and Naïve Bayes always keep the worst performance. This can be explained due to the nature of the algorithms. In Naïve Bayes, it assumes that every attribute is independent of each other, hence the probability that a point belongs to a class can be calculated through the probability of that point's attributes. But in fact, the mean absolute correlation coefficient as seen in Table 1 varies from 0.143 to 0.469, which negates the assumption. The equation for Naïve Bayes is as follows:

$$p(c|x) = \frac{p(x|c) * p(c)}{p(x)} = \frac{p(c)}{p(x)} * \prod p(x_i|c)$$

where: c is is the targeted class

x is the data point

i is the data point's attribute index

Deep Belief Network consists of the layer of Restricted Boltzmann machines, which can self-learn the pattern in the unsupervised manner. In [20] , the authors stated that the algorithm ignores the top-down inference, and it only learns a layer of feature at a moment, never calibrates the low-level

parameter. This leads to incorrect representation of the input, eventually wrong prediction.

3.4.3. Analysis on mean training time

Table 8: Average training time and accuracy of each algorithm

Algorithm	Average training time	Average accuracy	Accuracy standard deviation
KNN	0.013997	0.937794	0.083374
SVM	25.06169	0.897184	0.106732
DT	2.524976	0.943016	0.092289
SNN	433.7381	0.930663	0.094895
LDA	0.807953	0.875488	0.106782
NB	0.122958	0.767922	0.162704
LR	18.47428	0.900716	0.104607
BIRNN	4475.73	0.93752	0.092374
GRU	3909.609	0.938038	0.092709
LSTM	4610.378	0.938064	0.092509
CNN	946.8942	0.937628	0.095822
DNN	1273.057	0.933238	0.091461
DBN	1760.674	0.822522	0.134453

In this section, we compare the average training time of those 13 ML algorithms. For that, we run each algorithm over the 25 datasets and take the training time average of those 25 experiments. The results are shown in Table 8. Fig. 5 show the mean training time of all 13 algorithms compare with the referenced algorithm - Long Short-Term Memory, which has the highest average training time.

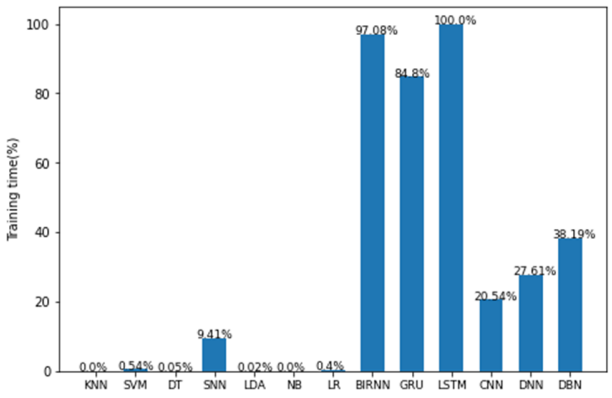


Fig. 5: Average training time of 13 algorithms by percentage (compared to the longest training time of LSTM)

As can be seen in Fig. 5 and Table 8, the mean training time of neural networks is much longer than those of shallow models. Even SNN, which took the longest time to train among the shallow models – is still more than twice as fast as CNN, which has the shortest time to train among the deep models. Some shallow models such as NB or KNN even take less than 0.01% of the time that LSTM takes. And yet none of the more complicated models outperforms the Decision Tree at 94.3% accuracy. This proves that the datasets are not complex enough to warrant the use of more complex algorithms.

4. Discussion

The purpose of this work is to give an analysis for researchers to preliminarily evaluate the classification performance of the different machine-learning algorithms for a certain network context and purpose, and with an implicit network data set. Thereby, we can have a rough prediction in advance which type of machine learning algorithm should be suitable to deal with the classification/attack detection task of a certain network scenario.

We, therefore, analyze the data characteristics and try to connect find a relationship of it to the detection performance of various machine-learning classification algorithms. The findings and insights on the relationships elaborated in our paper could be used to give some rough guidance which algorithm should be suitable for a certain training data set.

From our experiments and analyses, no conclusive relationship between non-discriminant measures (basic, statistic and information theoretic) and the detection performance has been able to be found.

We have found that Decision Tree has the best performance on all metrics except AUC on experimental datasets, and also has a much shorter training time than neural network approaches.

Deep learning models have much more flexibility, and may perform better with hyperparameter optimization, regularization or training data, and may have faster training time with more powerful hardware such as GPU. However, we doubt all that effort is worth it for these network datasets that already perfom very well on much simpler algorithms. The deep learning models perform better than the shallow models at a specific range of within- and between- class variance in terms of accuracy. Whilst the shallow models outperform in another range. Out of those 2 ranges, these two groups of algorithms perform roughly the same. The reasons could be:

– If the variances are too low, the datasets maybe too simple and all algorithms can perfom well

- If these variances are high, the deep learning models are weaker at picking up the slight extra complexity possibly due to inefficient optimization or having too many parameters which leads to overfitting, and thus underperforming the shallow ones.
- If the variances then go higher, the shallow models will have difficulty formulating more complex underlying relationships that the deep models can now detect due to their higher number of parameters, thus deep models now outperform shallow ones.
- Finally, if the variances are too high, both types of models will face too much difficulty in the classification task, and their performances are about similar again. Though, we think that with more tuning, deep models will perform better.

In conclusion, within- and between-class variance may be good indicators of the complexity of a dataset. The Deep models will be good for more complex datasets and have much more room for expansion, however they will require a lot of processing power, as well as optimization efforts. The Shallow models should be used if the dataset is simple, if short training time is required, if thorough model optimization cannot be done, or simply if they perform well enough.

At the bottom line, more conclusions could be derived from our results, and in the future, we would like to test on more datasets and metrics to find more trends and make better recommendations.

Acknowledgments

This research is funded by the Hanoi University of Science and Technology (HUST) under project number T2020-SAHEP-010. We also thank for the technical contribution of Miss. Nguyen Thuy Linh – our student.

References

[1] C.E. Brodley, Addressing the selective superiority problem: Automatic algorithm/model class selection, in: Proceedings of the tenth international conference on machine learning, 1993, pp. 17–24.

[2] P. Brazdil, J. Gama, B. Henery, Characterizing the applicability of classification algorithms using meta-level learning, in: Proc. European Conference on Machine Learning, 1994,

[3] Ricardo Vilalta, Christophe Giraud-Carrier, Pavel Brazdil, Carlos Soares: Using meta-learning to support data Mining. IJCSA. 1(1), pp.31-45, 2004

[4] C. Giraud-Carrier, R.Vilalta and P. Brazdil, ―Introduction to the special issue on meta-learning‖, Machine Learning 54, 187–193, 2004.

[5] G. Wang, Q. Song, X. Zhu, An improved data characterization method and its application in classification algorithm recommendation, Appl. Intell. 43 (4) (2015) 892–912.

[6] R. Ali, S. Lee, T.C. Chung, Accurate multi-criteria decision making methodology for recommending machine learning algorithm, Expert Syst. Appl. 71 (4) (2017) 257–278

[7] S. Gore, N. Pise, Dynamic algorithm selection for data mining classification, Int. J. Sci. Eng. Res. 4 (12) (2013) 2029–2033

[8] D.H. Wolpert, W.G. Macready: No free lunch theorem for search, Technical Report SFI-TR-05-010, Santa Fe Institute, Santa Fe, NM, 1995

[9] I. Ullah and Q. H. Mahmoud, A Technique for Generating a Botnet Dataset for Anomalous Activity Detection in IoT Networks, vol. 2020-October, no. April 2021. Springer International Publishing, 2020

[10] Y. Meidan et al., "N-BaIoT-Network-based detection of IoT botnet attacks using deep autoencoders," IEEE Pervasive Comput., vol. 17, no. 3, pp. 12–22, 2018, doi: 10.1109/MPRV.2018.03367731.

[11] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection," no. February, pp. 18–21, 2018, doi: 10.14722/ndss.2018.23204.

[12] I. Vaccari, G. Chiola, M. Aiello, M. Mongelli, and E. Cambiaso, "Mqttset, a new dataset for machine learning techniques on mqtt," Sensors (Switzerland), vol. 20, no. 22, pp. 1–17, 2020, doi: 10.3390/s20226578.

[13] H. Hindy, E. Bayne, M. Bures, R. Atkinson, C. Tachtatzis, and X. Bellekens, "Machine Learning Based IoT Intrusion Detection System: An MQTT Case Study (MQTT-IoT-IDS2020 Dataset)," Lect. Notes Networks Syst., vol. 180, pp. 73–84, 2021, doi: 10.1007/978-3-030-64758-2_6.

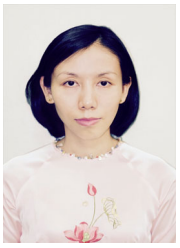
[14] A. Alsaedi, N. Moustafa, Z. Tari, A. Mahmood, and Adna N Anwar, "TON-IoT telemetry dataset: A new generation dataset of IoT and IIoT for data-driven intrusion detection systems," IEEE Access, vol. 8, pp. 165130–165150, 2020, doi: 10.1109/ACCESS.2020.3022862.

[15] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," 2015 Mil. Commun. Inf. Syst. Conf. MilCIS 2015 - Proc., 2015, doi: 10.1109/MilCIS.2015.7348942.

[16] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," ICISSP 2018 - Proc. 4th Int. Conf. Inf. Syst. Secur. Priv., vol. 2018-January, no. Cic, pp. 108–116, 2018, doi: 10.5220/0006639801080116.

[17] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy," Proc. - Int. Carnahan

Conf. Secur. Technol., vol. 2019-October, no. Cic, 2019, doi: 10.1109/CCST.2019.8888419.
[18] A. H. Lashkari, G. D. Gil, M. S. I. Mamun, and A. A. Ghorbani, “Characterization of tor traffic using time based features,” ICISSP 2017 - Proc. 3rd Int. Conf. Inf. Syst. Secur. Priv., vol. 2017-January, no. January, pp. 253–262, 2017, doi: 10.5220/0006105602530262.
[19] Bill Fulkerson (1995) Machine Learning, Neural and Statistical Classification, Technometrics, 37:4, 459, DOI: 10.1080/.1995.10484383
[20] Neelam Agarwalla et al, "Deep Learning using Restricted Boltzmann Machines" in International Journal of Computer Science and Information Technologies, Vol.7(3), 2016, 1552-1556



TRUONG THU HUONG received the B.Sc.degree in electronics and telecommunications from Hanoi University of Science and Technology (HUST), Vietnam, in 2001, the M.Sc. degree in information and communication systems from the Hamburg University of Technology, Germany,in 2004, and the Ph.D. degree in telecommunications from the University of Trento, Italy, in 2007. She came back to work for Hanoi University of Science and Technology as a Lecturer, in 2009, and became an Associate Professor, in 2018. Her research interests are oriented toward network security, artificial intelligence, traffic engineering in next generation networks, QoE/QoS guarantee for network services, green networking, and development of the Internet of Things ecosystems and applications.

TA PHUONG BAC received the B.Sc.degree in Electronics and Telecommunications from the Hanoi University of Science and Technology (HUST), Vietnam, in 2020. He is currently a Master’s student at Soongsil University (SSU), Korea. He has been also a Research Assistant in the Distributed Cloud and Network Laboratory, School of Electronic Engineering, SSU, Korea. His research interests include Computer Networking, Cloud-Edge Computing, and Artificial Intelligence.

BUI DOAN THANG is a senior student of the talented program in Electronics and Telecommunications Engineering, School of Electronics and Telecommunications, Hanoi University of Science and Technology. Thang has been working at the Future Internet Laboratory for 2 years. His research interest includes IoT, network security, machine learning/AI and its application

DAO MINH LONG is a senior student of the talented program in Electronics and Telecommunications Engineering, School of Electronics and Telecommunications, Hanoi University of Science and Technology. Long has been a research assistant at the Future Internet Laboratory for 2 years. His research interest includes IoT, network security, machine learning/AI and its application.

NGUYEN MINH DAN is a junior student of the talented program in Electronics and Telecommunications Engineering, School of Electronics and Telecommunications, Hanoi University of Science and Technology. Working as a research assistant at the Future Internet Laboratory since 2020, he shows his interest in research areas of network security and Internet of Things.

LE ANH QUANG is a junior student of the talented program in Electronics and Telecommunications Engineering, School of Electronics and Telecommunications, Hanoi University of Science and Technology. Quang has been a research assistant at the Future Internet Laboratory since 2020. His field of interest includes Deep Learning and Edge Computing.

NGUYEN VIET HOANG is a student in Electronics and Telecommunications Engineering, School of Electronics and Telecommunications, Hanoi University of Science and Technology. Hoang has been a research assistant at the Future Internet Laboratory since 2020. His field of interest includes Machine Learning and Network security.