Research on threat detection in cyber security based on machine learning

PRIME
PACIFIC RIM MEETING ON ELECTROCHEMICAL AND SOLID STATE SCIENCE
HONOLULU, HI
Oct 6–11, 2024
Abstract submission deadline:
April 12, 2024
Learn more and submit!
Joint Meeting of
The Electrochemical Society
•
The Electrochemical Society of Japan
•
Korea Electrochemical Society

Research on threat detection in cyber security based on machine learning

Qiwei Ke[1]

[1] Beijing University of Technology, Beijing, 100124, China

*Corresponding author's e-mail: keqiwei@emails.bjut.edu.cn

Abstract. The volume of the data has been rocketed since the new information era arrives. How to protect information privacy and detect the threat whenever the intrusion happens has become a hot topic. In this essay, we are going to look into the latest machine learning techniques (including deep learning) which are applicable in intrusion detection, malware detection, and vulnerability detection. And the comparison between the traditional methods and novel methods will be demonstrated in detail. Specially, we would examine the whole experiment process of representative examples from recent research projects to give a better insight into how the models function and cooperate. In addition, some potential problems and improvements would be illustrated at the end of each section.

1. Introduction

In recent years, 5G technology has established a solid stage for the internet of things (IoT). The widespread of IoT devices strengthens our connection with technical equipment around us. However, it also brings problematic challenges about cyber security at the same time as the information exchange becomes more frequent than before. Upgrading our threat detection approaches seems extremely necessary. In the previous literature, there were a few articles about the overall review of threat detection which covered diverse related fields. In this paper, it is aimed to make a survey research, which gives an outline of various aspects of machine learning-based threat detection systems by explaining the model used in each experiment and evaluate the corresponding results. At the same time, a comparative study is presented for depicting the differences between the older technique and the latest technique.

The rest of the paper is organized as follows.

Related works about threat detection are demonstrated in section 2. In the next section examples of intrusion detection are given. In section 4, recent researches on the measures of malware detection are depicted. In the following section, we will represent the achievement within the area of vulnerability detection. Finally, the paper is concluded with a discussion of potential improvements and problems.

2. Related work

This section discusses the surveys and review articles on threat detection. Al-Mhiqani et al. [1] described the insider threat detection and made a classification that depended on the insider. Arslan et al. [2] investigated the common attacks on mobile phones and analyzed the results from the captured data packets. Shaukat et al. [3] surveyed three machine-learning detection methods, namely decision tree, deep belief network, and support vector machine. Yuan et al. [4] covered the reason why the deep learning method was needed in insider threat detection and listed some existing examples to better emphasize the conclusion. Salloum et al. [5] represented the applications to intrusion detection based on machine learning, deep learning, and data mining techniques. Sahoo et al. [6] reviewed the contributions of feature presentation and learning algorithms for URL detection.

The main contributions of this review can be summarized as follows.

Firstly, most of the previous reviews mainly focused on one specific domain in threat detection while intrusion detection, malware detection, and vulnerability detection are discussed in this essay. Secondly, comprehensive detection procedures and detailed results are given in each topic rather than merely providing theoretical support. Finally, both traditional machine learning techniques and deep learning algorithms are covered in the essay.

3. Network intrusion detection

3.1 Background

With the booming usage of the Internet, it has a deeper influence on daily life, including work and study. However, a gradual rise of threats in the network has been detected in recent years. So lots of research efforts have been made to develop an Intrusion Detection System (IDS) to monitor security events and make the judgment whether they are intrusive or not. IDS can be categorized into signature-based and anomaly detection-based [7]. The former system is effective when the intrusion is known because it relies on the pre-installed signatures in it. The latter one is proven to be more practical in reality since most of the attacks are novel. To build the above system, machine learning methods, like Support Vector Machines (SVM) and Random Forests, and deep learning methods which depend on the calculation results from stacked deep networks can be utilized [10]. The crucial process behind intrusion detection is feature selection which can greatly reduce the error rate [8, 9]. In order to improve the accuracy of the detection model, deep learning techniques are now dominant in this area instead of pure machine learning techniques. The main reason is that deep learners can independently extract more concrete features from the samples while the majority of machine learning techniques require expert interaction to distinguish the features. The differences can be found in the following experiments.

3.2 Previous solution

In 2016, Javaid et al. [7] adopted a Self-taught Learning (STL) model based on a deep learning technique to solve the problem of intrusion detection. Two steps were implemented to classify the samples. Firstly, a clear feature representation could be extracted from a large collection of unlabeled data. The sparse auto encoder was chosen because it could be put into practice easily and had a relatively outstanding performance. Secondly, this representation was used for testing with labeled data.

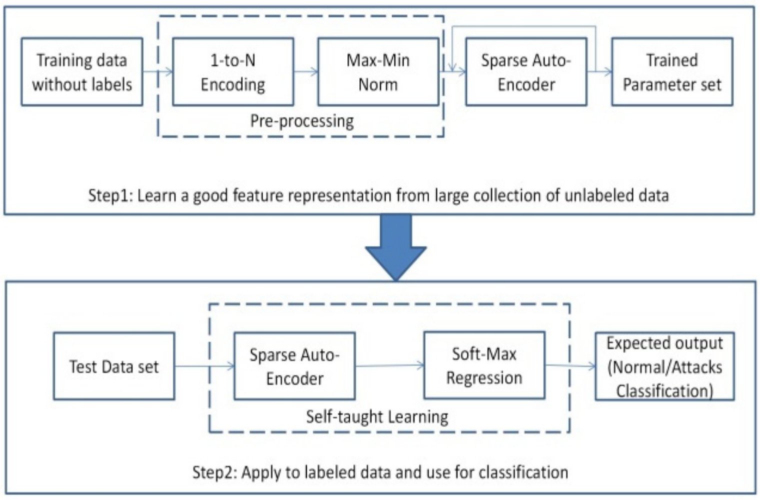
Training data without labels
1-to-N Encoding
Max-Min Norm
Pre-processing
Sparse Auto-Encoder
Trained Parameter set
Step1: Learn a good feature representation from large collection of unlabeled data
Test Data set
Sparse Auto-Encoder
Soft-Max Regression
Self-taught Learning
Expected output (Normal/Attacks Classification)
Step2: Apply to labeled data and use for classification

Figure 1. The detailed procedure of approaches involved in Javaid's team.

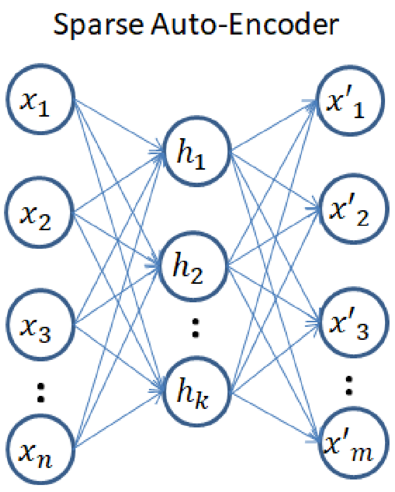

Figure 2. Structure of sparse auto-encoder. Note that bias (b) and weight (w) are not shown in the graph for clarity.

3.2.1 NSL-KDD Data Set

NSL-KDD Data Set is an updated version of the previous KDD Cup 99 Data set. Each sample contains 41 characteristics and is categorized into a specific group. It removes the redundancy of the old data set which may result in poor classification and less realism. A brief explanation of the attack type is given below.

- Denial of Service Attack (DOS): the attackers make the machine crush by sending loads of requests to occupy all the usable resources provided.
- User to Root Attack (U2R): attackers managed to obtain the root access by taking advantage of vulnerabilities based on a normal user account.
- Remote to Local Attack (R2L): attackers are able to send packets to the user's server from a remote server.
- Probing attack: attackers collect private information about computers without notice from security controls.

Table 1. Data set used in Javaid's team.

NSL-KDD	Training	Test
Normal	67343	9711
DoS Attack	45927	7458
U2R Attack	52	67
R2L Attack	995	2887
Probe	11656	2421

3.2.2 Evaluation

True Positive: the number of positive examples correctly classified
True Negative: the number of negative examples correctly classified as negative
False Positive: the number of negative examples wrongly classified as positive
False Negative: the number of positive examples wrongly classified as negative
Accuracy: the percentage of correctly classified records over the total number of records.

Precision: the ratio of the number of true positives samples divided by the sum of true positives and false positives samples.

Recall: the ratio of the number of true positives samples divided by the sum of true positives and false negatives samples.

F-Measure: the harmonic mean of precision and recall and represents a balance between them.

Table 2. Evaluation when using self-taught learning and soft-max regression (apply directly without feature learning).

2-class training data	Soft-max regression	Self-taught learning
Accuracy	97.0%	98.2%
Precision	97.2%	98.8%
Recall	96.2%	97.6%
F-measure	96.8%	98.2%

From the table, the conclusion can be drawn that with the assistance of feature learning, all four criteria for STL are at least 1% higher than the pure soft-max regression. This kind of difference can be considerable if the samples are large enough.

3.3 Update solution

In 2018, Shone et al. [11] proposed a model that combined the Non-Symmetric Deep Auto Encoder (NDAE) (deep-learning model) with the Random Forest (RF) (machine learning model). The combination showed better or at least match results from similar research and it dramatically shortened the training time. The auto encoder was an unsupervised feature extraction algorithm based on a neural network that using backpropagation to minimize the difference between the inputs and expected output. Significantly, the subsequent higher layers extracted higher-order features. As for the reason why the NDAE was applied, it could decrease the overheads of computations and time without a large impact on accuracy and efficiency. RF could be considered as an algorithm to boost the cooperation between the weak learners. In the experiment, the processed feature pattern was sent to the RF after two stacked NDAE structures. The details of the NDAE and classification model are shown below, including the parameters for the input layer and hidden layers.

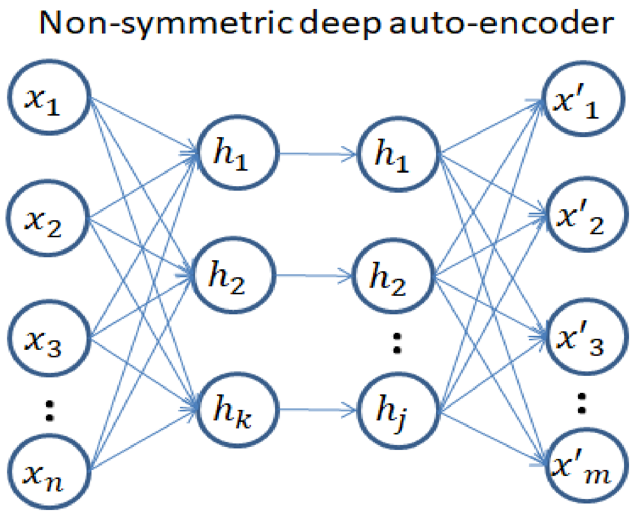

Figure 3. Structure of non-symmetric deep auto encoder. Note that bias (b) and weight (w) are not shown in the graph for clarity.

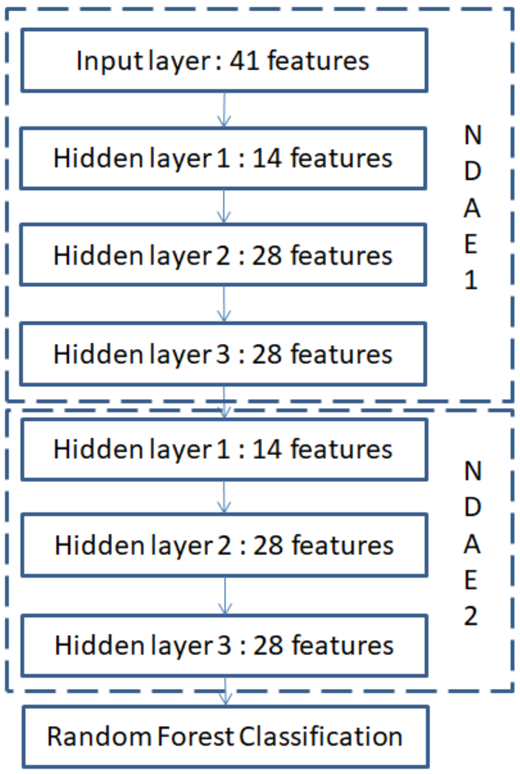
Input layer : 41 features
Hidden layer 1 : 14 features
Hidden layer 2 : 28 features
Hidden layer 3 : 28 features
NDAE 1
Hidden layer 1 : 14 features
Hidden layer 2 : 28 features
Hidden layer 3 : 28 features
NDAE 2
Random Forest Classification

Figure 4. Stacked NDAE Classification Model proposed by Shone's team

Table 3. Data set used by Shone's team.

NSL-KDD	Training	Test
Normal	67343	9711
DoS attack	45,927	5,741
U2R attack	52	37
R2L attack	995	2199
Probe	11,656	1,106

3.3.1 Evaluation

When it came to the comparison between deep belief networks and Stacked Non-Symmetric Deep Auto-Encoder, it was evident that S-NDAE took a lead in all five criteria. By maximizing the benefit of both the deep learning method and traditional machine learning method, more practical features could be obtained from the samples which significantly increased the effects of S-NDAE.

Table 4. Evaluation for 5-class performance based on Deep Belief Networks and Stacked Non-Symmetric Deep Auto-Encoder.

5-class	DBN	S-NDAE
Accuracy	80.58%	85.42%
Precision	88.10%	100.00%
Recall	80.58%	85.42%
F-measure	84.08%	87.37%
False Alarm	19.42%	14.58%

3.4 Discussion
About the better choice between the STL model and S-NDAE, the further experiment should be carried out. Although it is noticeable that the accuracy of STL (98%) is much higher than the S-NDAE (85%), some details should be emphasized.

- To what extent may the number of kinds of classification influences the accuracy. (2-class in STL model and 5-class in S-NDAE model)
- To what extent may the pre-processing process (1-to-N Encoding and Max-Min Norm) in the STL model influences the accuracy.

Also, we can not ignore the challenges that may encounter in the future.

- How to perfect our feature selections from samples.
- More labeled samples should be created for further network training.
- How to prepare for the drastic growth in the volume of network data.
- How to design a proper norm to cooperate with diverse network protocols.

4. Malware detection

4.1 Background
With the development of mobile technology, Android devices have become a popular choice for users. However, devices with Android applications installed are exposed to diverse security threats due to the openness [12]. Third-party applications from the Android market which provide various choices for users may hide suspicious programs at the same time. Attackers take advantage of this trend and attempt to obtain user's private information including personal photos and records like address books via tempting victims to install malware. It is the malware used by attackers to infiltrate the system like spyware and worms [13]. By simply warning the users of required permissions, the success is limited since it needs domain-level knowledge to distinguish [12]. With the assistance of newly-designed malware detection techniques, this threat started to be mitigated.

4.2 Previous solution
In 2014, Yuan et al. [14] put forward a machine learning-based method that specific features could be extracted from the static and dynamic analysis of applications. Significantly, these methods could be time-saving because there was no need to design and upgrade detection rules manually. By fully making use of these, Yuan's team managed to reach 96% accuracy under real-world simulation. The detailed detection procedure is illustrated below.

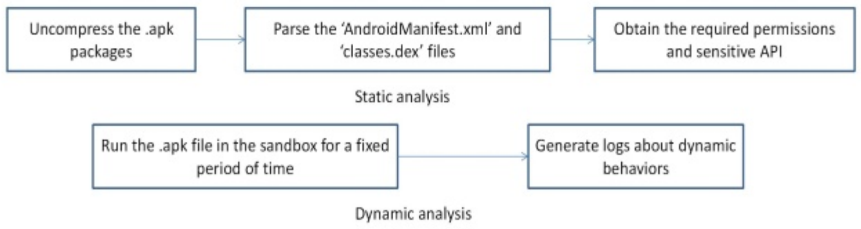
Uncompress the .apk packages
Parse the 'AndroidManifest.xml' and 'classes.dex' files
Obtain the required permissions and sensitive API
Static analysis
Run the .apk file in the sandbox for a fixed period of time
Generate logs about dynamic behaviors
Dynamic analysis

Figure 5. Detailed procedure of static and dynamic analysis

After uncompressing and parsing the corresponding apk files, required permissions from this application and sensitive application program interface (API) can be revealed. In addition, through simulating the application in the sandbox, researchers can understand the dynamic behaviors with the help of logs. These kinds of samples (more than 200 features) facilitate the later training process in the deep learning model provided below.

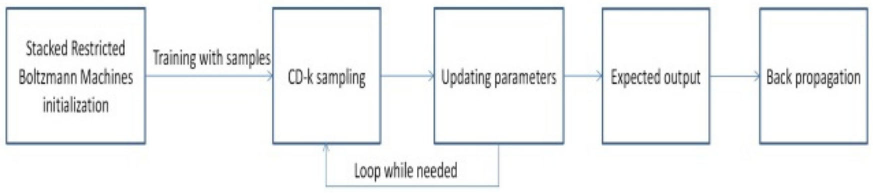

Figure 6. Deep learning model implemented in Yuan's team.

In the pre-training phase, a deep belief network that contains stacked restricted Boltzmann machines is utilized to characterize Android apps. Parameters can be updated continuously while the features input into the model. After a certain period of training section, the model can be used to distinguish the unknown samples from test sets. CD-k sampling not only increases the accuracy but also optimizes the training efficiency. For the back-propagation, the network is fine-tuned by labeled samples in a supervised manner. Yuan's team tried multiple traditional methods, namely logistic regression (LR), multi-layer perception (MLP), and support vector machine (SVM), and compared the actual effect with the deep learning method. The test result is given below for further analysis.

4.2.1 Evaluation

Under the same training and testing environment, those who took traditional methods as their model received unsatisfying results (78.0%,79.5%,80.0%). Conversely, the deep learning method achieved 96.5% accuracy among 200 testing samples which seemed to be applicable in real circumstances.

Table 5. Accuracy comparison between traditional method (LR&MLP&SVM) and deep learning method.

Model	Training / testing	Accuracy
Logistic Regression	300/200	78.0%
Multi-layer Perception	300/200	79.5%
Support Vector Machine	300/200	80.0%
Deep learning	300/200	**96.5%**

4.3 Update solution

In 2018, Kim et al. [15] designed an advanced approach to detect malware and got a more promising result. Comparing to Yuan's method, they made some essential changes to the previous model. The whole procedure is shown below for reference.

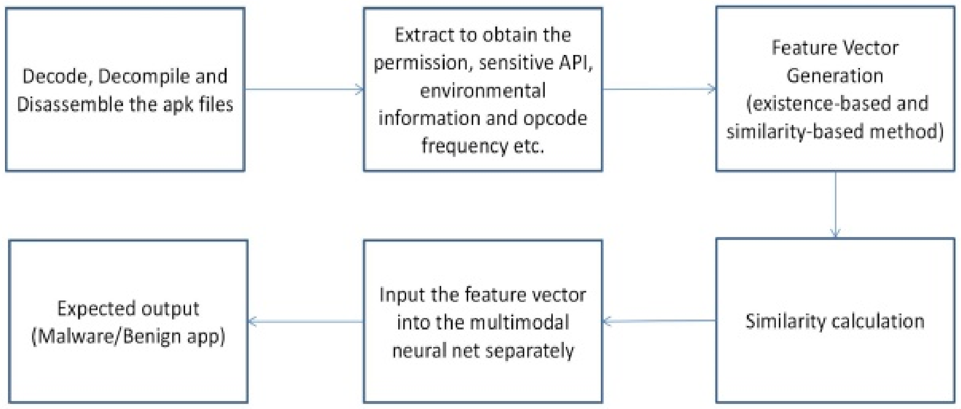
Decode, Decompile and Disassemble the apk files
Extract to obtain the permission, sensitive API, environmental information and opcode frequency etc.
Feature Vector Generation (existence-based and similarity-based method)
Similarity calculation
Input the feature vector into the multimodal neural net separately
Expected output (Malware/Benign app)

Figure 7. The overall architecture of framework proposed by Kim's team

Existence-based and similarity-based feature extraction methods were cooperated to represent the characteristics. Due to this, the advantages of encompassing multiple features could be strengthened to handle complex tasks when many properties were extremely similar between normal and benign apps. Different initial neural networks processed different types of features. The similarity-based feature (method opcode, method API, and shared library function feature vectors) focused on the elements which were similar to the malware representatives in the malicious feature database. The existence-based feature (string, permission, component, and environmental feature vectors) focused on the elements which only represented the existence of features in the malicious feature database.

Table 6. The Parameters of multimodal neural network.

Network	Layer	Number of neurons	Activation function
Initial Deep Neural Network	Input	5000	ReLU
	Hidden	2500	ReLU
	Hidden	1000	ReLU
Final Deep Neural Network	Merging	500	ReLU
	Hidden	100	ReLU
	Hidden	10	ReLU
	Output	1	Sigmoid

The initial network was built by five deep neural networks with separate input of five feature vectors (the permission/component/ environmental feature, String feature, method opcode feature, method API feature, shared library function opcode feature). The final network was one deep neural network and could output final results. The ReLU was able to make the model free of vanishing gradient issues.

4.3.1 Evaluation
The naïve binary and frequencies of the words were not comparable with the multi-feature-based algorithm, especially the formal one, which even had less than half the accuracy of the discussed method above. The substantial number of the samples contributed to the success of Kim's team.

Table 7. The evaluation among different chosen feature vectors.

Model	Malware samples/benign samples	Accuracy
uses a naïve binary as a feature vector	21260/20000	43%
uses the frequencies of the words	21260/20000	62%
Kim's team	21260/20000	**98%**

4.4 Discussion
One thing worth to be demonstrated is that the second method is purely based on static observation. Better results may probably be achieved by adding dynamic feature patterns into the design. The first method has already witnessed the beauty of dynamic analysis and it gives us an insight into how to take the first step. However, recent researches also ring the alarm that we should consolidate our processing power in order to meet the challenge from polymorphic and metamorphic as they can alter their behaviors in a short period of time [20].

5. Vulnerability detection

5.1 Background
High quality and security codes lay a solid foundation for a secure system. Tracing back to 2011, the code audit still depended on manual review which required high-level knowledge [16]. The common strategy at that time was shown below. They extracted the API symbols to help the auditor accelerate the discovery process but most of the work still demanded human interactions. Recent years have seen huge progress on vulnerability detection that the multiple software representations and multimodal deep learning ease the pressure of the auditors. These kinds of implementation can be divided into two approaches, which are similarity-based static analysis and pattern-based static analysis (it can be further divided into rule-based and machine learning-based analysis) [17]. The former is appropriate for code-cloning-based vulnerabilities while the latter requires human-defined rules or vulnerability patterns.

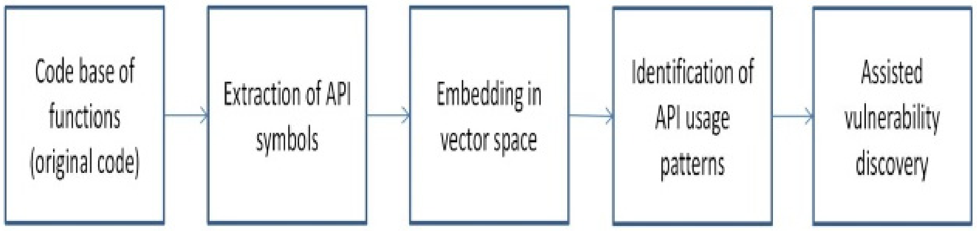
Code base of functions (original code)
Extraction of API symbols
Embedding in vector space
Identification of API usage patterns
Assisted vulnerability discovery

Figure 8. Initial schematic of vulnerability detection.

5.2 One possible solution
In 2020, Heidbrink et al. [18] devised to use three common multimodal deep learning models (namely correlation neural network, joint auto encoder, bidirectional deep neural network) to detect vulnerabilities and test their effects. The correlation neural network contained two (or more) inputs/outputs and a loss function term that maximized correlation across the different modalities. The joint auto encoder had extra encoder and decoder layers for each modality. It was called private branches which strikes a balance between the contributions of the modality and cross-modal mixing layers. The bidirectional deep neural network equipped the ability of multimodal representational learning via two separate networks. In the experiment, after the data normalizations, they used 5-fold cross-validation to analyze samples. The same settings were used for each architecture to test the effect. Regarding the database, the Juliet Test Suite is a well-known data set for vulnerability detection and covers common flaws within the field.

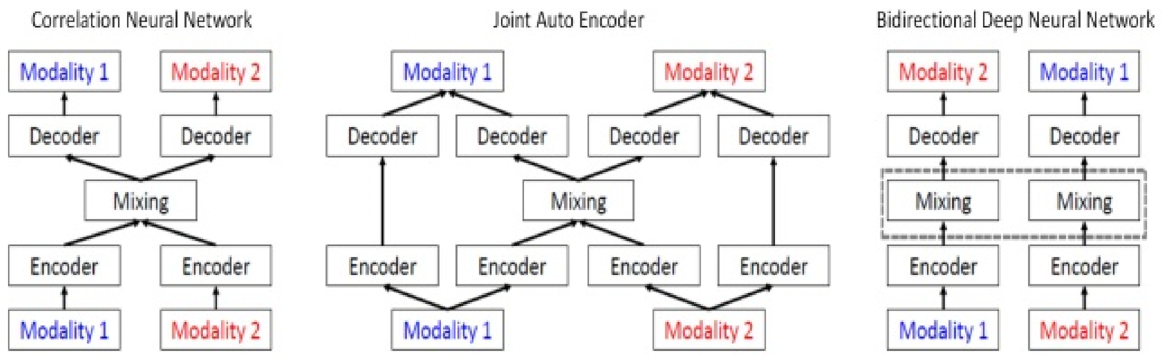

Figure 9. The common multimodal deep learning models used by Heidbrink's team

Table 8. Juliet Test Suite Data Summary.

CWE	Flaw Description	Flaw/Not Flaw
78	OS Command Injection	6102/15602
121	Stack Based Buffer Overflow	6346/16868
190	Integer Over flow	3296/12422
369	Divide by Zero	1100/4142
377	Insecure Temporary File	146/554
416	Use After Free	152/779
476	NULL Pointer Dereference	398/1517
590	Free Memory Not on Heap	956/2450
680	Integer to Buffer Over flow	368/938
789	Uncontrolled Memory Allocation	612/2302
78	OS Command Injection	6102/15602

5.2.1 Evaluation

Table 9. Flaw Detection Results on Juliet Test Suite based on Correlation Neural Network, Joint Auto Encoder, and Bidirectional Deep Neural Network (the best method among them is shown in boldfaced).

CWE	Correlation Neural Network	Joint Auto Encoder	Bidirectional Deep Neural Network
190		97.00	
369		94.00	
377	**96.00**	94.00	**96.00**
416	**72.00**	66.00	70.00
476		97.00	
680		94.00	
78	93.00	**94.00**	93.00
789	90.00	**92.00**	**92.00**
121		1.00	
590			

Among the three models, the bidirectional deep neural network gave the best results but the differences were relatively small in all the test cases. The worst performance on CWE416 may attribute to the limited samples of the training set as the deep learning mechanism expected a large number of corresponding features and the difficulties of processing the specific flaw type (Use After Free).

5.3 Alternative solution

In 2020, another group of researchers demonstrated a new structure of an updated cell-based joint auto encoder that combined the neural architecture search (NAS) and multimodal learning models [19]. Specifically, the cell represented a portion of the architecture and was defined using a densely connected directed acyclic graph. NAS performed an automated search across diverse neural network architectures to find the best solutions. In the updated joint auto encoder, they replaced the decoders with a linear layer to merge the previous outputs. Among all the NAS techniques, GDAS was adopted to improve the cell search which was a gradient-based search algorithm through a differentiable architecture sampler. 10 cross-validation offered a pessimistic estimate of the generalization error.

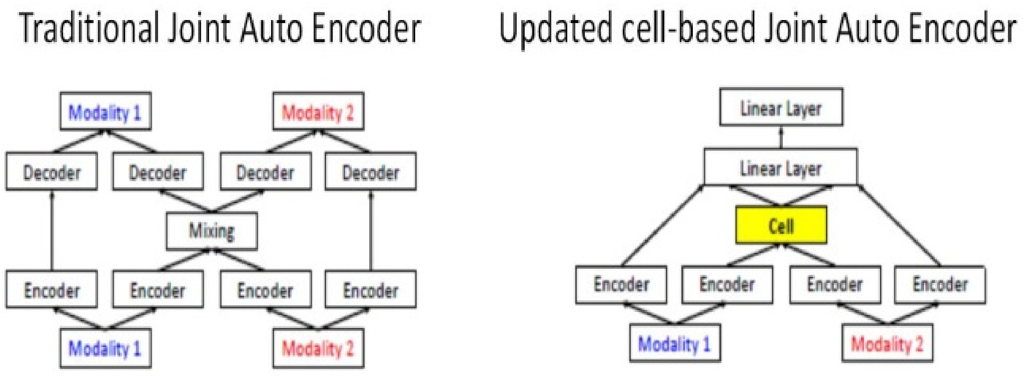

Figure 10. The comparison between the traditional joint auto encoder architecture and the updated cell-based joint auto encoder architecture

5.3.1 Evaluation
As is described in the table, there is not an obvious difference between the traditional JAE and cell-based JAE. Further work needs to be done to figure out the reason for the slight improvement. Future research can concentrate on the proper structure and functionality of the cell. Whether the effect of the removal of the decode module and the additional linear layers brings negative outputs should be investigated.

Table 10. Averaged accuracy using 10 cross-validation (better results between traditional JAE and cell-based JAE are shown in boldfaced).

CWE	Traditional JAE	Cell-based JAE
121	**0.9975±0.0008**	0.9970±0.0012
190	**0.9907±0.0059**	0.9884±0.0067
369	0.9500±0.0203	**0.9703±0.0220**
377	0.9285±0.0420	**0.9514±0.0410**
416	0.9359±0.0471	**0.9468±0.0400**
680	0.9356±0.0115	**0.9417±0.0197**
78	0.9360±0.0143	**0.9427±0.0155**
789	0.9630±0.0183	**0.9683±0.0215**
476	1.0000±0.0000	
590		

5.4 Discussion
Besides the exception of CWE416, similar accuracy has been found in all four architectures under the same test suite. Some details are worth to be explored in the next stage.

- To what extent may the different cross-validation methods have influences on the accuracy. (5-fold cross-validation and 10 cross-validation)
- The potential improvement of other NAS techniques comparing to the GDAS.

6. Conclusion
In this paper, several recent approaches towards intrusion detection, malware detection, and vulnerability have been discussed. Among all the fields, deep learning techniques considerably optimize the accuracy comparing to the conventional method by introducing feature extraction and representation. At the cost of acceptable extra complexity and training periods, the large number of expert interactions can be reduced. The combination of machine learning and deep learning also brings new possibilities to threat detection in web security. More and more classification methods and neural

network architecture have been designed and put into practice. Some of them have achieved productive results and are ready for additional examination in the future. The others may need to be re-designed based on their drawbacks but the overall trend is optimistic. Similar measures can be taken into account to help overcome the obstacles. First, about the feature generation method, more mechanisms should be developed to boost up precision and efficiency. Improper feature representations may lead to a huge rise in the cost of the training section. Second, about the training and testing database, the extremely limited dataset is available to the researchers in the open environment. This burning issue results from a large number of demands in the feature pattern generation. I hope most of the researchers can share their private datasets for investigation use to push the development of machine learning in all kinds of application fields, besides threat detection.

Reference

[1] Al-Mhiqani, M. N., Ahmad, R., Zainal Abidin, Z., Yassin, W., Hassan, A., Abdulkareem, K. H., ... & Yunos, Z. (2020). A review of insider threat detection: Classification, machine learning techniques, datasets, open challenges, and recommendations. Applied Sciences, 10(15), 5208.
[2] Arslan, B., Gunduz, S., & Sagiroglu, S. (2016, April). A review on mobile threats and machine learning based detection approaches. In 2016 4th International Symposium on Digital Forensic and Security (ISDFS) (pp. 7-13). IEEE.
[3] Shaukat, K., Luo, S., Chen, S., & Liu, D. (2020, October). Cyber Threat Detection Using Machine Learning Techniques: A Performance Evaluation Perspective. In 2020 International Conference on Cyber Warfare and Security (ICCWS) (pp. 1-6). IEEE.
[4] Yuan, S., & Wu, X. (2021). Deep learning for insider threat detection: Review, challenges and opportunities. Computers & Security, 102221.
[5] Salloum, S. A., Alshurideh, M., Elnagar, A., & Shaalan, K. (2020, March). Machine Learning and Deep Learning Techniques for Cybersecurity: A Review. In AICV (pp. 50-57).
[6] Sahoo, D., Liu, C., & Hoi, S. C. (2017). Malicious URL detection using machine learning: A survey. arXiv preprint arXiv:1701.07179.
[7] Javaid, A., Niyaz, Q., Sun, W., & Alam, M. (2016). A deep learning approach for network intrusion detection system. Eai Endorsed Transactions on Security and Safety, 3(9), e2.
[8] Yin, C., Zhu, Y., Fei, J., & He, X. (2017). A deep learning approach for intrusion detection using recurrent neural networks. Ieee Access, 5, 21954-21961.
[9] Karatas, G., Demir, O., & Sahingoz, O. K. (2018, December). Deep learning in intrusion detection systems. In 2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT) (pp. 113-116). IEEE.
[10] Dong, B., & Wang, X. (2016, June). Comparison deep learning method to traditional methods using for network intrusion detection. In 2016 8th IEEE International Conference on Communication Software and Networks (ICCSN) (pp. 581-585). IEEE.
[11] Shone, N., Ngoc, T. N., Phai, V. D., & Shi, Q. (2018). A deep learning approach to network intrusion detection. IEEE transactions on emerging topics in computational intelligence, 2(1), 41-50.
[12] Yuan, Z., Lu, Y., & Xue, Y. (2016). Droiddetector: android malware characterization and detection using deep learning. Tsinghua Science and Technology, 21(1), 114-123.
[13] Hardy, W., Chen, L., Hou, S., Ye, Y., & Li, X. (2016). DL4MD: A deep learning framework for intelligent malware detection. In Proceedings of the International Conference on Data Science (ICDATA) (p. 61). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).
[14] Yuan, Z., Lu, Y., Wang, Z., & Xue, Y. (2014, August). Droid-sec: deep learning in android malware detection. In Proceedings of the 2014 ACM conference on SIGCOMM (pp. 371-372).
[15] Kim, T., Kang, B., Rho, M., Sezer, S., & Im, E. G. (2018). A multimodal deep learning method

for android malware detection using various features. IEEE Transactions on Information Forensics and Security, 14(3), 773-788.

[16] Yamaguchi, F., Lindner, F., & Rieck, K. (2011, August). Vulnerability extrapolation: Assisted discovery of vulnerabilities using machine learning. In Proceedings of the 5th USENIX conference on Offensive technologies (pp. 13-13).

[17] Li, Z., Zou, D., Xu, S., Chen, Z., Zhu, Y., & Jin, H. (2021). Vuldeelocator: a deep learning-based fine-grained vulnerability detector. IEEE Transactions on Dependable and Secure Computing.

[18] Heidbrink, S., Rodhouse, K. N., & Dunlavy, D. M. (2020). Multimodal deep learning for flaw detection in software programs. arXiv preprint arXiv:2009.04549.

[19] Cooper, A., Zhou, X., Heidbrink, S., & Dunlavy, D. M. (2020). Using Neural Architecture Search for Improving Software Flaw Detection in Multimodal Deep Learning Models. arXiv preprint arXiv:2009.10644.

[20] Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., & Venkatraman, S. (2019). Robust intelligent malware detection using deep learning. IEEE Access, 7, 46717-46738.