MLAB-BiLSTM: Online Web Attack Detection Via Attention-Based Deep Neural Networks

Jun Yang[1], Mengyu Zhou[1(✉)], and Baojiang Cui[2]

[1] School of Computer Science, Beijing University of Posts and Telecommunications, Beijing, China
{junyang,manassehzhou}@bupt.edu.cn
[2] School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing, China
cuibj@bupt.edu.cn

Abstract. With the continuous development of Web threats such as SQL injection and Cross-Site Scripting, Numerous web applications have been plagued by various forms of security threats and cyber attacks. And web attack detection has always been the focus of web security. Because the attacking payloads are often multiple small segments hidden in the long original request traffics, traditional machine learning based methods may have difficulties in learning useful patterns from the original request. In this study, we proposed an MLAB-BiLSTM method that can precisely detect Web attacks in real time by using multi-layer attention based bidirectional LSTM deep neural network. Firstly, due to the malicious payloads contains similar keywords, we used a keyword enhanced embedding method to transfer the original request to feature vectors. Then the features are divided into different segments. The words in the segments are firstly inputted into the bidirectional LSTM model with an attention mechanism to generate a encoded representation of different segments. Then the segments of the requests are input into another BiLSTM model with an attention mechanism to generate the encoded representation of the original request. Finally, the generated features are input into the Convolutional Neural Network to find out which kind of attack payload it is. The MLAB-BiLSTM model was tested on CSIC dataset and CTF competition traffic, the experiment results show that the accuracy of the model is above 99.81%, the recall of 99.56%, the precision of 99.60%, and the F1 Score is 0.9961, which outperformed both traditional rule-based methods like Libinjection or deep learning based methods like OwlEye.

Keywords: Network security · Malicious request detect · Long-short term memory · Attention

1 Introduction

With the rapid expansion of the Internet, more and more devices are connected to the Internet. In recent years, the majority of enterprises chose to save their data

S. Yu et al. (Eds.): SPDE 2020, CCIS 1268, pp. 482–492, 2020.
https://doi.org/10.1007/978-981-15-9129-7_33

or provide service through web applications. Due to the flexibility of Web applications, there are many known or unknown ways to attack it, leading to leakage of user sensitive data. Symantec Internet Security Report [11] reveals that more than 1.3 million unique web attacks were blocked on endpoint machines every day in 2018. While analyzing all the URLs, 1 in 10 URLs analyzed was identified as being malicious. All these facts show that web security deserves to be taken seriously.

Given the severity to protect web attacks, there are many methods to detect web attack through malicious URLs proposed by scholars from all over the world. In summary, these methods can be divided into three categories, which are traditional rule based methods, machine learning based, and deep learning based. In rule based methods, such as ModSecurity [6], the web application firewall (WAF) usually used regular expression to match the requests, and the advantage of rule based detection is the low false alarm rate and the speed. However, this method is highly based on the experiences of the creator of the rule database, resulting in the weakness in detecting novel attack payloads. The machine learning based methods are based on the features extracted with artificial experiences and statistical models, giving them the ability to detect new types of intrusion as deviations from normal behavior [1]. But due to the selection of large amount of artificial features needed to be extracted and the performance of different detection methods, it's hard to use such methods in real life detection tasks. Lately, deep learning methods have achieved promising performance in malicious URLs classification tasks [14]. By learning and extracting features from the original web requests, it is significantly more efficient and more flexible to adapt to more complex attack behaviors due to the absence of the most time-consuming feature engineering phase. But during extracting, it usually treats all words with same weight, but in the real world, the researchers can classify the request with only few keywords in the URL.

In this paper, we proposed MLAB-BiLSTM model that can detect web attacks with both attention from word level and segment level. In our approach, we firstly used the word level attention to focus on which of the words are important in different segments of the web requests. Then the segment level attention focus on where the attacking payload may exist. The extracted features are fed into a CNN based classification network to decide which kind of attack this request belongs to.

To evaluate the model, we chose web requests both from CSIC dataset and CTF competition web traffic as training set. These requests including web attack methods such as XSS, SQLi. We compared proposed model with other web attack detect methods. The experimental results show that compared to other models the detection model we proposed is very effective and can obtain 99.7% accuracy.

The remaining parts of this article are organized as follows. In Sect. 2 we firstly summarized the related works. Section 3 considers the architecture of the proposed MLAB-BiLSTM neural network. In Sect. 4, we used both data from CSIC dataset and real-world traffic captured during CTF competitions. Finally, we draw our conclusion and discussions in Sect. 5.

2 Related Works

In this section, we analyzed different kinds of methods to detect web attacks. We divided these methods into three categories and discusses the advantage and disadvantages of these methods.

2.1 Rule Based Methods

Traditional rule based methods including blacklisting and heuristic approaches. Blacklisting means generating their rules by collecting a large amount of web traffics that contains malicious traffic. When a new visit was made, it tries to look up the database to find whether the request is in the database or not. Blacklist based rules need to keep obtaining different web attack traffic in real life, and with the growth of the malicious traffic database, the performance becomes worth. Future more, it is easy for attackers to find different payloads with same function to bypass the detection of the WAF [12], which is critical for protecting sensitive user data in big companies. Despite the weakage in generalization, nowadays there are still a large number of WAFs that use these approaches thanks to their simplicity and efficiency. To improve the ability of generalization, run-time application self-protection (RASP) approaches are proposed [4]. Instead of collecting the malicious traffic, extract the function call signature among those traffics with artificial experience and check if the traffic match any malicious signature, then raise an alarm to indicate the request is malignant. While the types of attacks and the flexibility of the web applications, it's pretty easy for attackers to find different ways to bypass the detection. As well as there is many kind of languages to develop a web application, so it's hard to develop a RASP firewall for each language.

2.2 Machine Learning Based Methods

Various machine learning methods have been used to detect malicious web attacks over the years. These approaches try to extract feature representations of web requests and training a prediction model on training data of both malicious and benign URLs. Some researchers have used statistical methods to locate key elements of malicious requests, extracted features for machine learning models [13]. Chandrasekhar et al. suggested a technique that combines Support Vector Machine classifier, K-means along with fuzzy neural network [2]. Chen et al. [3] proposed hybrid flexible neural-tree-based IDS with the help of flexible neural tree, evolutionary algorithm, and particle swarm optimization. Test results proved that their method is high-efficiency. With the help of machine learning methods, the models can detect new kinds of attacks with only the request traffic. But the requirement of expert experience and the difference between the systems make it hard for today's real world.

2.3 Deep Learning Based Methods

Nowadays deep learning [9] models have achieved great success in data-driven learning. Compared with knowledge-driven methods, data-driven approaches usually have a better generalization ability in detecting unseen attacks. Compared to machine learning methods, they don't need artificial feature extraction. Besides, they are more friendly in quick deployment and capable of rapid updating. Anomaly detection methods such as the HMM-Web [5] have shown their ability to capture unseen web attacks. And with the improvements of feature extraction such as convolutional gated-recurrent-unit [15], the speed of deep learning methods is improving.

2.4 Related Work: Summary

After a review of available technologies, we found out most deep learning based methods treat the word in the web requests with same weight. But in real world web attack traffic, there will only 5%–20% of the web requests contains the payloads used to attack the website which result in the speed was compromised in real world usage. We proposed an MLAB-BiLSTM method, it firstly generates a encoded representation of the original request to rule out the noise in the request and focusing on the more suspicious part that contains attacking payloads. Then the Text-CNN classify which kind of attack this attacking payload is. After experimental verification, the proposed method performs well in speed and accuracy of detecting both known and unknown web attacks.

3 Method

In this section, we described our proposed MLAB-BiLSTM model for detecting web attacks. The model contains two parts, firstly, the MLAB-BiLSTM generate a encoded representation contains the suspicious web attack from web request traffic, then a Text Convolution Neural Network is used to judge whether the requests contain valid attacking payload or not, and classify it to different types. In Fig. 1 is the brief structure of proposed MLAB-BiLSTM network.

To generate a encoded representation from the original web request, which is like the idea of document summarizing in neural language process works. It can find the key element from the document, which in our case is the malicious payload. Supposing we denote the original web request as R, and we can divide the original web traffic into several segments (S_i) by URL delimiter, \n for headers and delimiter for post data. The MLAB-BiLSTM is used to firstly choose suspicious parts from each segment S_i, then select the most suspicious segments from the request R. It firstly processes the document with a keyword enhanced traffic embedding module, then uses generated result as input of a multi-layer neural network, then it outputs the encoded representation containing suspicious attacking payload of the original request. Finally, the generated structure is feed into a Text-CNN model to evaluate which category of attack type this request may belong to.

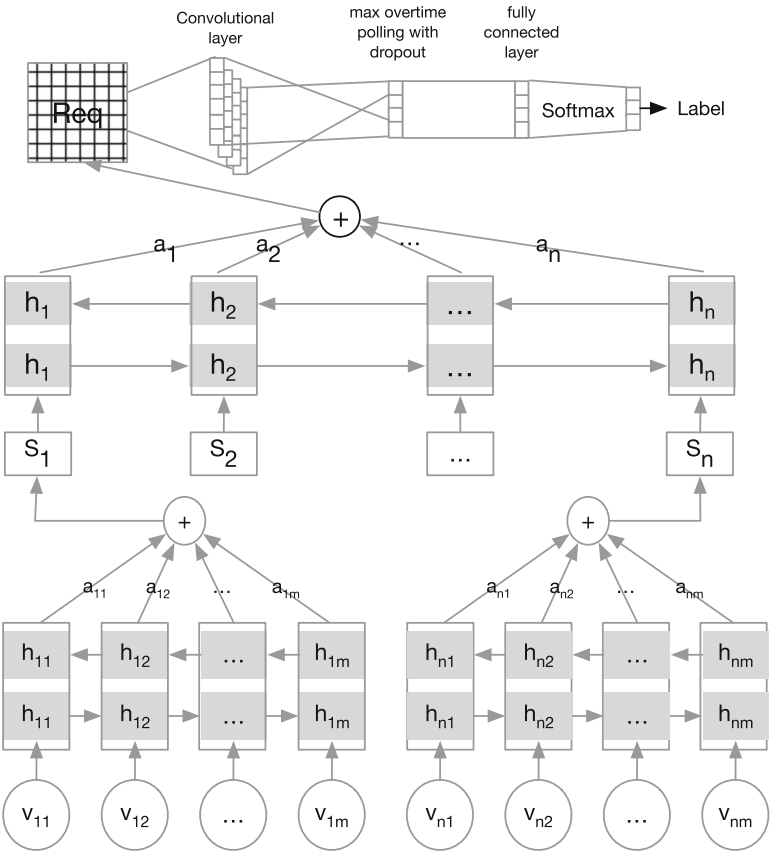

Fig. 1. MLAB-BiLSTM: the first layer is a word-level encoder for every segment in the original request, followed by a segment-level encoder. There is an attention layer after each encoder. Finally, the generated encoded representation will be fed into a Text-CNN network to output the original request into different attack types.

3.1 Keyword Enhanced Traffic Embedding

In most cases, the web requests contain such as URL addresses, methods, header, post data and etc., and most of the detect methods the web request are embedded in character level or 1-gram format. Using the letters of printable English characters, the original request is mapped to its corresponding vector representation, which is then embedded into the multidimensional feature space. It treats characters independently, but in web attack payloads the words are not completely independent, such as “script” for XSS payloads, “union”, “select” for SQLi payloads. To keep more key information as possible, in our implementation, we firstly collected a keyword list with artificial experience as briefly shown in Table 1, and then used it for request embedding.

Table 1. Web request keywords list

Type	List
SQLi	and, or, union, select, substr, xp_, hex, ascii, concat, order, exec, benchmark, sleep, information_schema
XSS	script, iframe, eval, alert, onerror, onclick, cookie, document, href, src
Normal	GET, POST, GET, OPTIONS, HTTP/1.1, HTTP/2

3.2 Word Feature and Attention

Suppose we have a request (R) with n segments, let $R = (s_1, s_2, \cdots, s_n)$. Each segment contains m words. After word embedding, the segment was transformed into a vector of $d * m$ dimension, let denote $S = (w1, w2, \cdots, w_m)$. The w_i is a d dimensional word embedding for the i^{th} word. In this paper, we use a bidirectional LSTM is used to encode the words in the segments with the focus on information from both forward and backward direction. So the hidden state h_t that contains the general information of the whole segment including both the forward and the backward part.

$$h_t = [\overrightarrow{LSTM}(w_t, \overrightarrow{h_{t-1}}), \overleftarrow{LSTM}(w_t, \overleftarrow{h_{t-1}})]$$

Supposing each unidirectional LSTM contains u hidden units, $H_s \in \mathbb{R}^{m \times 2u}$ can denote the whole LSTM hidden states.

$$H_s = (h_1, h_2, \cdots, h_m)$$

What we want is to extract the suspicious payload sequence from the original segment. Instead of treating all the words with same weight, an attention mechanism is introduced, paying more or less attention to words that affect the function of the segment. To achieve this goal, we used the hidden states as input, and output the weight of each word.

$$a_s = softmax(w_{s_2} tanh(w_{s_1} H_s^T))$$

Where $w_{s_2} \in \mathbb{R}^{k \times 2u}$ and $w_{s_1} \in \mathbb{R}^k$ is learned from the data; k is a manually set hyperparameter. Finally we uses softmax to normalize the attention weights.

Then we can obtain a weighted representation of original segment s_i which is the weighted sum of the attention vector a_s and LSTM hidden states H_s (Fig. 2).

$$s_i = a_s H_s$$

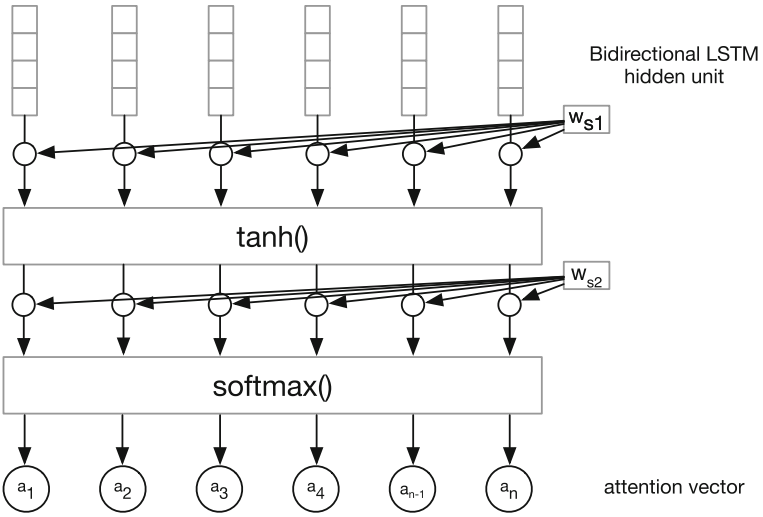

Fig. 2. The attention units take bidirectional LSTM hidden state as input, and output the attention vector a_s. The circles in the graph represent for unit in MLP layer. w_{s1} is a $k \times 2u$ dimensional matrix, and w_{s2} is a k dimensional vector. k is a manually set hyperparameter.

3.3 Segment Feature and Attention

To prevent hackers from hiding the payloads separately in two segments of the web request, we also focus on the relationship between the segments in the requests. Now we have the segment vector s_i, similarly, we can get the request LSTM hidden state representing as h_{st}.

$$h_{st} = [\overrightarrow{LSTM}(s_i, \overrightarrow{h_{t-1}}), \overleftarrow{LSTM}(s_i, \overleftarrow{h_{t-1}})]$$

With the forward and backward hidden states, it has the ability to focus on the adjacent segment information around i^{th} segment. Supposing each unidirectional LSTM contains u hidden units, the overall hidden state for the request can be represented as following.

$$H_r = (h_{s1}, h_{s2}, \cdots, h_{sn})$$

To extract important segment from the request, we also use an attention mechanism to take H_r as input to generate an output of attention vector representing the importance of the segments.

$$a_r = softmax(w_{s_2} tanh(w_{s_1} H_r^T))$$

Finally, we can obtain a encoded representation of the original request r which is the weighted sum of the attention vector a_r and the LSTM hidden states.

$$r = a_r H_r$$

3.4 Classification Network

After the multiple attention based bidirectional LSTM, the suspicious web request will be presented in a encoded representation. Then a convolutional neural network is applied to analyze the request is malicious or not. It performs a text classification task with input of the extracted request feature vector, and output the attack type of the given web requests.

The model architecture of the is similar to the Text-CNN proposed by Kim [8]. A 5-layer convolutional neural network with different filters and a max-pooling layer with dropouts. The different filters give CNN the ability to identify payloads accurately with multiple features. These features are reshaped and passed into three fully connected layers with Rectified Linear Unit. The output is probabilities of different attack types after being processed by a softmax layer. The loss function is composed of a multi-class cross entropy loss and a L_2 regular term.

$$L = -\frac{1}{N}\sum_{i}^{N}\sum_{j}^{K} y_{ij} log(p_{ij}) + \lambda \|W\|$$

4 Implementation and Experiments

To evaluate our proposed model, we used CSCI dataset and web traffic collected from CTF competitions. We prepared our proposed model with traditional rule base methods such as Libinjection, and HMM based methods like OwlEye [10].

4.1 Implementation Details

The train of MLAB-BiLSTM is optimized with Adam optimizer. The learning rate and weight decay was set to 1e−6 and 0.99. For the input of the neural network, the word embedding layer has a dimension of 8, and the hidden state of the LSTM is set to 100. So the segment encoder and request encoder in our model will output a 400 dimension combination from both forward and backward LSTM. The word and sentence attention context vector also have a dimension of 400. We split the segments firstly by the '\n' in the requests, and split the URL with '&' symbol. During training, the batch size was set to 64. The requests we selected has a maximum segment count of 150, and each segment contains no more than 200 words.

4.2 Performance on CSCI Dataset

The CSCI database is a publicly available labeled dataset contains web traffic to an e-Commerce Web Application contains 36,000 normal requests and more than 25,000 anomalous requests [7]. After reviewing the dynamic attacks among the anomalous traffic, we found there are 2000 SQLi and 1500 XSS samples.

As our model only detects SQLi and XSS attacks, other attacks are classified as benign traffics.

The results in Table 2 shows that our proposed model performs better than OwlEye and better recall rate than Libinjection. In terms of the F1-Score, our proposed model performs better than the other two methods.

Table 2. Test result on CSCI dataset

Method	Accuracy	Precision	Recall	F1-score
Libinjection	95.84%	99.81%	55.59%	0.7060
OwlEye	99.30%	96.21%	95.91%	0.9606
Proposed method	99.69%	98.23%	98.34%	0.9828

4.3 Performance on CTF Competition Traffic

To evaluate the MLAB-BiLSTM model, we also captured web traffic during different CTF competitions. In the dataset we collect 1 million web traffic logs with 245,000 attack samples collected from XSS and SQLi problems. The attack samples are labeled as SQLi or XSS by artificial experience.

Results. The performance of the selected methods are listed in Table 3. Our method obtains the second highest precision. OwlEye obtains the highest recall, slightly better than LTD, while it suffers from high False Positive Rate (FPR), while high FPR in production environment can interfere with normal user operation, so it is usually required to be less than 0.01%. For cyber attack detection tasks, there is a balance between FPR and recall, but low FPR is an important prerequisite in a production environment. Therefore it is unacceptable considering a system with high FPR may block normal user requests. In this experiment, OwlEye's FPR was considerably higher than our proposed approach. This is because, as an anomaly-based approach, OwlEye will classify anything not previously seen as malicious. In addition, if the web application is updated, OwlEye must be retrained with a new sample to conform to the new user behavior patterns, or the FPR will increase significantly. For these reasons, the OwlEye method is not suitable for real-time detection but is a good choice for offline analysis.

Rule based method Libinjection is widely used by many Internet corporations, such as Google. It is being known as first successful WAF. In our experiment, it achieves 100% precision (almost zero FPR), which is slightly better than our proposed method. However, due to the incomplete rule set, it has a recall rate of just over 85%, while our proposed method can detect over 99% of the attacks. So the over all accuracy of our method is much higher than Libinjection. In Table 4, we list some key element from misclassified samples by

Table 3. Test result on CTF competition dataset

Method	Accuracy	Precision	Recall	F1-score
Libinjection	96.37%	100%	85.20%	0.9201
OwlEye	99.65%	98.89%	99.69%	0.9929
Proposed method	99.81%	99.60%	99.56%	0.9961

Libinjection, the first two samples contain obvious SQLi features but the Libinjection missed. And the fourth and the fifth sample contains SQLi keywords such as "GRANT" and "Union", which appears in a lot of rules in the database lead to the misclassification of these examples. The last example was a normal session in PHP language, while Libinjection classified it as an XSS attack vector. As we can see from the above examples, even though the Libinjection method analyzed the lexical and syntax content of the request, it still cannot avoid certain misclassifications. Compared to Libinjection, our method based on deep learning methods exhibit better flexibility, better generalization and higher adaptability to avoid these misclassifications. Finally, in term of the F1-score, the LTD method outperforms all other three methods.

Table 4. Misclassified examples by Libinjection

Payload	Libinjection	MLAB-BiLSTM
1337) INTO OUTFILE 'xxx'–	N	Y
or 1<@. union select 1,version()#	N	Y
2104 GRANT AVE #A	Y	N
Manufacturing Workers' Union (MMWS)	Y	N
ONLINETOOLS_PHPSESSID = 3	Y	N

5 Conclusion

In this paper, we proposed an MLAB-BiLSTM method that can precisely detect Web attacks in realtime by using multi-layer attention based bidirectional LSTM deep neural network. Due to the malicious payloads contains similar keywords, we used a keyword enhanced embedding method to transfer the original request to feature vectors. Then we used a bidirectional LSTM model with attention to generate the encoded representation of the segments in the original request. Then uses the similar method to generate a encoded representation of the original requests. Then the structures are input into the CNN network to output the category of the attack. Moreover, the experimental results show that the MLAB-BiLSTM model can achieve better results on web attack detection than traditional methods. Later on, we would modify the model to add more malicious classes, adjust parameters, and minimize the use of parameters between networks to achieve better performance.

References

1. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: a survey. ACM Comput. Surv. (CSUR) **41**(3), 1–58 (2009)
2. Chandrasekhar, A., Raghuveer, K.: Intrusion detection technique by using k-means, fuzzy neural network and SVM classifiers. In: 2013 International Conference on Computer Communication and Informatics, pp. 1–7. IEEE (2013)
3. Chen, Y., Abraham, A., Yang, B.: Hybrid flexible neural-tree-based intrusion detection systems. Int. J. Intell. Syst. **22**(4), 337–352 (2007)
4. Čisar, P., Čisar, S.M.: The framework of runtime application self-protection technology. In: 2016 IEEE 17th International Symposium on Computational Intelligence and Informatics (CINTI), pp. 000081–000086. IEEE (2016)
5. Corona, I., Ariu, D., Giacinto, G.: HMM-web: a framework for the detection of attacks against web applications. In: 2009 IEEE International Conference on Communications, pp. 1–6. IEEE (2009)
6. Firewall, A.: Modsecurity (2009)
7. Giménez, C.T., Villegas, A.P., Marañón, G.Á.: HTTP data set CSIC 2010. Information Security Institute of CSIC (Spanish Research National Council) (2010)
8. Kim, Y.: Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882 (2014)
9. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
10. Liu, X., Yu, Q., Zhou, X., Zhou, Q.: OwlEye: an advanced detection system of web attacks based on HMM. In: 2018 IEEE 16th International Conference on Dependable, Autonomic and Secure Computing, 16th International Conference on Pervasive Intelligence and Computing, 4th International Conference on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech), pp. 200–207. IEEE (2018)
11. O'Gorman, B., et al.: Internet security threat report. Technical report. Symantec Corporation (2019)
12. Sheng, S., Wardman, B., Warner, G., Cranor, L., Hong, J., Zhang, C.: An empirical analysis of phishing blacklists (2009)
13. Torrano-Gimenez, C., Nguyen, H.T., Alvarez, G., Franke, K.: Combining expert knowledge with automatic feature extraction for reliable web attack detection. Secur. Commun. Netw. **8**(16), 2750–2767 (2015)
14. Vinayakumar, R., Soman, K., Poornachandran, P.: Evaluating deep learning approaches to characterize and classify malicious URL's. J. Intell. Fuzzy Syst. **34**(3), 1333–1343 (2018)
15. Yang, W., Zuo, W., Cui, B.: Detecting malicious URLs via a keyword-based convolutional gated-recurrent-unit neural network. IEEE Access **7**, 29891–29900 (2019)