JTE HCMUTE
ISSN: 2615-9740

JOURNAL OF TECHNICAL EDUCATION SCIENCE
Ho Chi Minh City University of Technology and Education
Website: https://jte.edu.vn
Email: jte@hcmute.edu.vn

A Malware Family Classification Approach Based on Deep Sequence Modeling

Xuan Sam Nguyen[1*], Han Nguyen[2]
[1]*Ho Chi Minh City University of Technology and Education, Vietnam*
[2]*University of South Florida, United States of America*

[*]Corresponding author. *Email: samnx@hcmute.edu.vn*

ARTICLE INFO

Received: 23/01/2024
Revised: 25/02/2024
Accepted: 27/02/2024
Published: 28/08/2025

KEYWORDS

One Dimension Convolutional Neuron Network (1D-CNN);
Bidirectional Long Short-Term Memory (BiLSTM);
Malware Family Classification (MFC);
Sequential Data (SD);
Opcode Pattern Extraction Mechanism (OPEM).

ABSTRACT

In cyber networks, malware can come in various forms and different families. Classifying malware into families helps respond to specific threats more effectively. Because the executable files contain instructions and opcodes to identify types of malwares, presenting in sequential data. Sequence learning models are necessary for improving performance of malware family classification. In this work, We proposed hybrid models based on a one-dimensional convolutional neural network and bidirectional long short-term memory where one dimensional convolutional neural network works as a preprocessing mechanism in the extracting malware features from raw data and bidirectional long short-term memory networks process the sequential data in both forward and backward directions. Simulating results shown that our proposal was able to classify 21 malware families with training and testing accuracy 95%, significantly better than one directional convolutional neuron network, training accuracy with 98% and testing accuracy 91%. Similarity, loss of our model in the training and the testing is decreased smoothly, compared to one dimension convolutional neuron network.

Doi: https://doi.org/10.54644/jte.2025.1527

1. Introduction

Recently, malware attacks can occur to any devices, operating systems at any time with different types, such as viruses, spyware, adware, backdoor, etc. These types of malwares can damage, change, and steal personal information. Thus, preventing malware infection is one of critical tasks to maintaining the confidentiality, integrity, and availability of your devices and organizations information systems. Because malware comes from variant and different forms, each of them has specific characteristics. Therefore, advanced detecting algorithms and classifying methods play a critical role in identifying sophisticated malware and classifying different malware threats with high accuracy.

There are different advanced malware classification (MC) and malware family classification (MFC) methods. Convolutional neural networks (CNNs) are well applied for malware detection and classification [1], the paper provided image-based (two dimensions) CNNs for MC and it achieved full advantages of CNN when it works with image and video analysis. Recurrent Neural Networks (RNNs) are a specific type of CNNs, RNNs are used to handle sequential data of malicious malware. Unlike traditional feedforward neural networks, RNNs have connections that form cycles, allowing them to maintain a memory of previous inputs. To solve long-term dependencies of sequence data (SD) problems, Long Short-Term Memory (LSTM), a type of RNNs, is provided to address tasks involving sequence-based (one dimension) malware classification [2].

The work is to investigate the performance of 1D-CNNs and BiLSTM on executable files for malware classification. The executable files contain instructions and opcodes to identify malware. Our main contributions in the paper not only improve performance of malware family classification but also reduce loss with opcode pattern extraction mechanism.

The rest of this paper is organized as follows. In section 2, we discuss related works on dataset, advanced malware classification methods, and performance metrics. An opcode pattern extraction mechanism (OPEM) is proposed as inputs to CNN and BiLSTM, which make use for efficient learning

are described in section 3. Experiments and evaluation are presented and analyzed in Section 4. Finally, conclusions and discusses the future work are shown in section 5.

2. Materials and Methods

2.1. Dataset

A growing number of risks and threats come from various types of malwares [3]. Thus, malware classification is critical to help incident responders react rapidly. Currently, many families of malware are studied in the research [2] and each family includes various samples and types of malwares. Institutively, families of malware are grouping malware that have similar characteristics into the same family and the characteristics can be recognized by patterns. In order to correctly categorize types of malwares into malware family, which have similar characteristics. In the below list, we cover different types of malwares that are grouped into families in tables 1.

Adware, standing for advertising-supported software, is a type of potentially unwanted program (PUP) that displays unwanted advertisements to the user. The aim of adware is to generate revenue for its developers by delivering advertisements, often in an intrusive or disruptive manner.

VirTool is a detection prefix commonly used by antivirus and anti-malware programs to identify certain types of tools, utilities, or components that may be used by malware. The term "VirTool" indicates that the detected item is not inherently malicious by itself, but it could be utilized or exploited by malicious software.

Spyware, a type of malicious software, is designed to secretly monitor and collect information about a user's activities on a computer or device. The purpose of spyware is to gather sensitive data without the user's knowledge or consent. This data may include personal information, login credentials, browsing habits, and other sensitive details.

Worm, a type of malicious software, is aimed to self-replicate and spread independently across computer networks. Unlike viruses, worms do not require a host program to attach themselves to. Instead, they exploit vulnerabilities in network services or operating systems to spread from one computer to another. Worms can cause network congestion, degrade system performance, and sometimes deliver payloads with malicious intent.

Trojan, standing for Trojan horse, is a type of malicious software. Unlike viruses and worms, Trojans do not replicate themselves but rely on social engineering to trick users into installing or executing them. Once activated, Trojans can perform various malicious actions without the user's knowledge.

Backdoor, a type of malicious software, creates a secret and unauthorized entry point into a computer system. The primary purpose of a backdoor is to allow unauthorized access to the system, providing attackers with a hidden and often persistent means of control. Backdoor malware can be used for various malicious activities, and it poses significant security risks.

Password stealer, a type of malicious software, aimed to covertly steal sensitive login credentials, including usernames and passwords, from the compromised systems they infect. These malware variants aim to gain unauthorized access to users' accounts, leading to potential identity theft, financial loss, or unauthorized access to personal and sensitive information.

Rogue is malicious software that exhibits rogue or deceptive behavior, often with the intention of misleading or defrauding users.

Table 1. *Type of each malware family*

Index	Family	Type	Index	Family	Type
1	Adload	Trojan	12	Renos	Trojan
2	Agent	Trojan	13	Rimecud	Worm
3	Allaple	Worm	14	Small	Trojan
4	BHO	Trojan	15	Toga	Trojan

5	Bifrose	Backdoor	16	VB	Backdoor
6	CeeInject	VirTool	17	VBinject	VirTool
7	Cycbot	Backdoor	18	Vobfus	Worm
8	FakeRean	Rogue	19	Vundo	Trojan
9	Hotbar	Adware	20	Winwebsec	Rogue
10	Injector		21	Zbot	Password Stealer
11	OnLineGame				

2.2. Malware classification methods

2.2.1. Malware classification based on CNNs

Recently, convolutional neural networks (CNNs) have shown powerful feature learning for image classification. Many CNNs-based approaches have been proposed for MC and MFC based on the conversion from binary executables to images [4]. Unlike higher dimensions CNNs-based, one dimension CNN (1D-CNN) [5] can avoid loss from image inversion processes, and therefore, it can improve the accuracy of classification. The proposed 1D-CNN for detecting malware was introduced by [6] and the other proposed for classifying malware was proposed by [7].

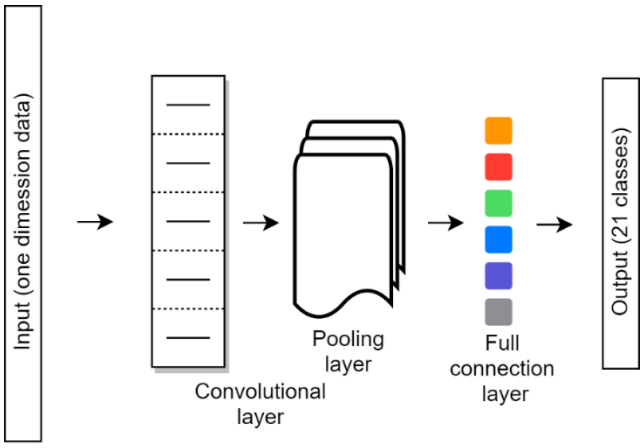

Figure 1. *1D-CNN*

As shown in figure 1, 1D-CNN consists of three layers as follows: convolutional layer, pooling layer, and fully connected layer. In this paper, the input contains the list of tokens created from previous step, OPEM, then convolutional layer extracts the list of tokens by 64 filters with 3x3 kernel size, RELU activation function is used in the layer. In general, each filter would allow the 1D-CNN model to learn one feature, thus, we can extract 64 different features. The pooling layer reduces the complexity of the model, therefore reducing the possibility of overfitting, we apply max pooling, it reduces the spatial size of the output by selecting maximum value of the region of the feature map. We provide pooling size 2 with stride 1 for the layer. The fully connected layer computes the class scores for output, and thus layers are fully connected to perform classification. The layer includes learnable parameters, which can be used to optimize performance results. Also, we also added dropout layer [8] to prevent overfitting problem to 1D-CNN.

2.2.2. Malware classification based on BiLSTM

In [9], LSTM was proposed by Hochreiter and Schmidhuber. It was used to solve the learning long-term dependencies problems in RNNs. LSTM can hold information for an extended period, and thus, it is well-suited for sequential data. LSTM networks include the chain of memory blocks, which is called LSTM cells. Each LSTM cell consists of three gates, which are the input gate, the forget gate, and the output gate. The “input/update” gate controls the flow of information and decides what information is

added to LSTM cell. Forget gate allows what information will be removed from LSTM cell. The output gate controls which information will go out of LSTM cell. The Architecture of the LSTM cell is described in figure 2.

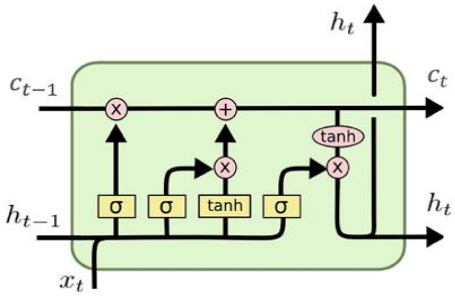

Figure 2. *Architecture of LSTM cell*

Bidirectional long-short term memory (BiLSTM) is an extended version of LSTM. Unlike LSTM, input sequences in BiLSTM are transmitted in both directions, allowing it to use information from both sides. The architecture of BiLSTM [10] consists of two LSTM that process input sequences in both forward and backward directions. The first one receives the input sequences in one direction, while the other one gets input sequences in the opposite direction. BiLSTM returns a probability vector as output, and the final output is a combination of both of these probabilities. Mathematically, the calculation of LSTM cell can be summarized as follows:

First, the forget gate f_t is computed in equation (1) which take previous long-term memory h_{t-1} as an input. Because of the sigmoid activation function σ, the outcome of f_t is bounded between 0 and 1, thus, if value of f_t is close to 0, it removes the previous state h_{t-1} or if value of f_t is close to 1, it keeps the previous state h_{t-1}

$$f_t = \sigma(W_t * [h_{t-1}, x_t] + b_f \tag{1}$$

Then, the "input/update" gate i_t is computed in equation (2). The value of i_t tells the cell which new information to store in the internal cell state.

$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i) \tag{2}$$

We compute k_t, representing the new input as an update state:

$$k_t = tanh(W_k[h_{t-1}, x_t] + b_k) \tag{3}$$

Next, the cell state c_t is calculated from the forget gate f_t and the previous cell state c_{t-1}. The result is summed with update state k_t:

$$c_t = c_{t-1} \odot f_t + i_t \odot k_t \tag{4}$$

Next, output gate o_t is computed in equation (5). The value of o_t controls the output, the value that gets passed on to the next cell.

$$o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o) \tag{5}$$

Finally, output of LSTM cell h_t is computed in equation (6).

$$h_t = \begin{cases} o_t \odot tanh(c_t) & t > 0 \\ 0 & t > 0 \end{cases} \tag{6}$$

Where, $\odot$ is Hadamard multiplication, x_t is the input sequence at the time t, b_o, b_i, b_f, b_k are bias.

The architecture of BiLSTM [10] has been achieved full improvement for sequential data. In our context, we added the process of OPEM to reduce noise and help BiLSTM focusing on important features to improve its performance. The process of OPEM will be described in a later section.

2.3. Performance metrics

The metrics used to evaluate the above models are listed as follows: 1) we used the cross-entropy loss function for multi-class classification model where the output label is assigned integer value (1, 2, …., 21), 2) Accuracy is the percentage of all categories that are correct, 3) Precision is the ratio of the number of correctly recognized malware to the total number, 4) Recall is the ratio of the number of correctly recognized malware to the number of samples that should be recognized, and 5) F1 score is calculated based on previous metrics, it is an indicator used in statistics to measure the accuracy of the model. Mathematically, the metrics are listed in following equations below:

$$Cross\ entropy = \sum_{i=1}^{21} o_t \log(\widehat{o_t}) \tag{7}$$

$$\text{Accurancy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \tag{8}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{9}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{TN}} \tag{10}$$

$$\text{F1 score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{11}$$

where, TP presents the number of given samples to be positive in the positive category, TN presents number of given samples to be negative in the negative category, FP presents number of given samples to be positive in the negative category, FN presents the number of given samples to be negative in the positive category.

3. Opcode pattern extraction mechanism

In order to obtain the significant features from the dataset [3], we proposed a method that focus to extract PE opcode files into input data files. Typically, the opcodes are a group of bits that define operations such as addition, subtraction, shift, complement, etc. and they are presented in binary files, which can convert to assembly format file (asm file). It usually contains four predefined segments, .text segment, .idata segment, .rdata segment, and .data segment. Because the .text segment stores program instructions, we need to extract the opcode patterns to SD and present them in vectors so that our models can learn the feature vectors to improve accuracy.

The process of OPEM consists of three phases, including 1) disassembling, 2) Filtering, and 3) Normalization. In the disassembling phase, the contents of sequential data are disassembled into csv files, then the first characteristic of instructions in assembly code, opcodes, will be filtered and convert the opcode into vector in Filtering phase. In order to improve the performance of investigated models, scaling the input features to a common scale can reduce the impact of outliers and reduce loss, thus we added the phase at last. It is worth noting that a new vector is generated in phase 2, focusing on opcodes. It is an important key to solve our problem with sequential learning model. The process of the OPEM is shown in figure 3 and general MFC is described in figure 4, including to three tiers, where the first tier is processing data, the second tier consists of training models, and the third tier is testing. In this work, we investigated our proposed model compared to 1D-CNN as shown at tier-2.

Figure 3. *Opcode pattern extraction mechanism (OPEM)*

JTE HCMUTE
ISSN: 2615-9740
JOURNAL OF TECHNICAL EDUCATION SCIENCE
Ho Chi Minh City University of Technology and Education
Website: https://jte.edu.vn
Email: jte@hcmute.edu.vn

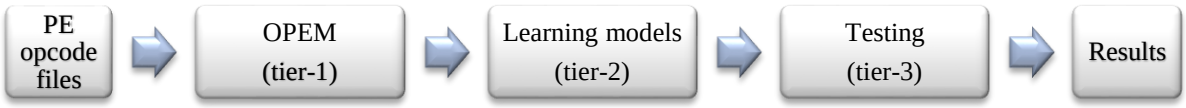

Figure 4. *OPEM and MFC*

4. Experiments and evaluations

4.1. Simulation setup

Experiments are conducted on PYTHON 3.11 and KERAS is the framework. Libraries are used in experiments such as PANDAS, NUMPY, MATPLOTLIB, SKLEARN, etc. They are integrated in Anaconda framework or manual installation. We investigate and evaluate 1D-CNN and BiLSTM models with input as opcode feature vectors files (csv). Configuration and architecture of the models are shown in table 2, table 3, and table 4, as summary reports of the 1D-CNN and BiLSTM.

Table 2. *Configuration of the models*

Parameter	Types
Activation function	ReLu, Tanh, Softmax
Learning rate	0.001
Optimizer	Adam
Loss	sparse_categorical_crossentropy
Epochs	50
Training/Testing	80/20

Table 3. *Architecture of LSTM*

Layer (type)	Output Shape	Param #
lstm_1 (BiLSTM)	(None, 128)	66560
dense_2 (Dense)	(None, 21)	2709
Total params:		69,269
Trainable params:		69,269
Non-trainable params:		0

Table 4. *Architecture of 1D-CNN*

Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 499, 64)	256
max_pooling1d (MaxPooling1D)	(None, 249, 64)	0
conv1d_1 (Conv1D)	(None, 124, 128)	24704
max_pooling1d_1 (MaxPooling 1D)	(None, 62, 128)	0
dropout (Dropout)	(None, 62, 128)	0
max_pooling1d_2 (MaxPooling 1D)	(None, 31, 128)	0
flatten (Flatten)	(None, 3968)	0
dense (Dense)	(None, 21)	83349
activation (Activation)	(None, 21)	0
Total params:		108,309
Trainable params:		108,309
Non-trainable params:		0

4.2. Experimental results and evaluations

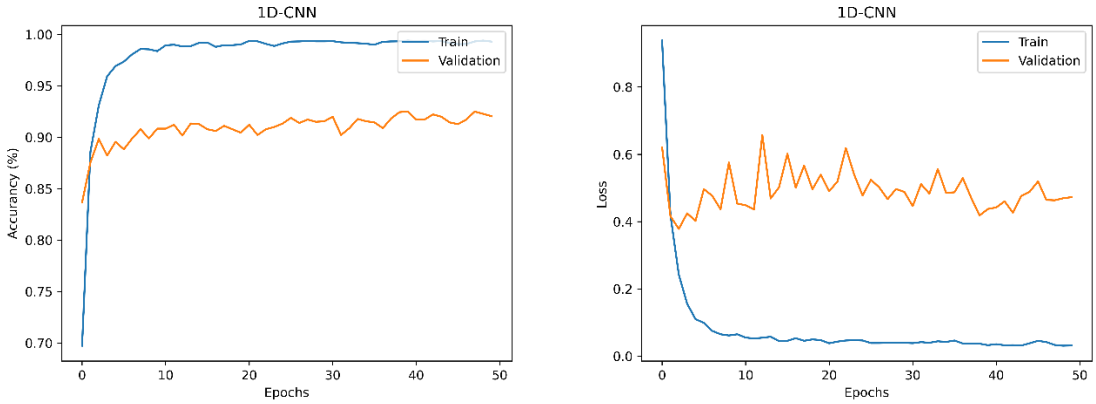

Figure 5. *Accuracy and Loss of 1D-CNN*

As shown in figure 5, the loss and accuracy of 1D-CNN + OPEM are presented. Though the training accuracy of the model reaches 99%, the testing accuracy of the model is very low, 91%. In terms of loss, it is decreasing in the training but not in the testing. Overall, the model performs very well for training data, but it has poor performance with test data. It is a worse case of overfitting.

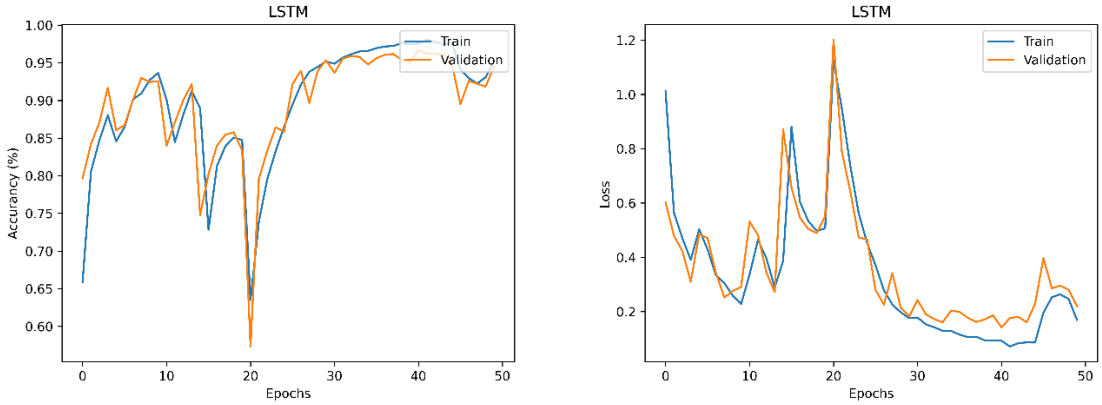

Figure 6. *Accuracy and Loss of BiLSTM*

As shown in figure 6, loss and accuracy of BiLSTM + OPEM are presented. The training and the testing accuracy of the model strongly fluctuated within 20 epochs but both training and testing changed together and achieved 95%. Moreover, the model does not involve overfitting. Overall, overfitting fitting could be solved by BiLSTM. This shows that our proposed model has a higher capacity to remove the overfitting problem and find the correct malware families respectively. It is worth because the goals of malware detection and classification models are to make accurate predictions on new malicious and unseen data. We summary the classification performances of our model against 1D-CNN at 40 epochs in table 4. The results show that our model consistently outperforms compared to 1D-CNN. For the BiLSTM, more epochs for training are required to improve accuracy, however, consuming more time for training makes BiLSTM become unreal.

Table 5. *The performance of 1D-CNN and BiLSTM at 10 epochs*

Parameter	Our Proposal	1D-CNN
Accuracy	0.940	0.937
Precision	0.937	0.933
Recall	0.937	0.935
F1-score	0.938	0.931

5. Conclusions and future works

In this paper, We study opcode-based malware classification with hybrid deep learning models for improving performance of malware classification. The approach provide a robust way to classify malware families caused by the conversion into two direction dimension image which is a common input for 1D-CNNs and sequential learning models. Though the experimental results show that our proposal works well and removes overfitting compared to 1D-CNN, the work has limitations when We have not fully investigated many scenarios.

In the future work, We will consider either pretrained models or self-attention mechanisms which are suitable for SD such as large malware families with more features and different instruction set architectures.

Conflict of Interest

The authors declare no conflict of interest.

REFERENCES

[1] M. Jain, W. Andreopoulos, and M. Stamp, "Convolutional neural networks and extreme learning machines for malware classification," *Journal of Computer Virology and Hacking Techniques*, vol. 16, pp. 229-244, 2020.
[2] H. Sandeep, "Static analysis of android malware detection using deep learning," in *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*, 2019: IEEE, pp. 841-845.
[3] M. Stamp, M. Alazab, and A. Shalaginov, *Malware analysis using artificial intelligence and deep learning*. Springer, 2021.
[4] D. Vasan, M. Alazab, S. Wassan, H. Naeem, B. Safaei, and Q. Zheng, "IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture," *Computer Networks*, vol. 171, p. 107138, 2020.
[5] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, and D. J. Inman, "1D convolutional neural networks and applications: A survey," *Mechanical systems and signal processing*, vol. 151, p. 107398, 2021.
[6] M. Azizjon, A. Jumabek, and W. Kim, "1D CNN based network intrusion detection with normalization on imbalanced data," in *2020 international conference on artificial intelligence in information and communication (ICAIIC)*, 2020: IEEE, pp. 218-224.
[7] X. Chong, Y. Gao, R. Zhang, J. Liu, X. Huang, and J. Zhao, "Classification of malware families based on efficient-net and 1D-CNN fusion," *Electronics*, vol. 11, no. 19, p. 3064, 2022.
[8] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929-1958, 2014.
[9] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735-1780, 1997.
[10] Z. Cui, R. Ke, Z. Pu, and Y. Wang, "Deep bidirectional and unidirectional LSTM recurrent neural network for network-wide traffic speed prediction," *arXiv preprint arXiv:1801.02143*, 2018.

Xuan Sam Nguyen received the Bachelor of engineering in Electronic and Communication Engineering from PTIT, Hanoi, Vietnam in 2002, the Master of science in Information and Communications Engineering from the Andong National University, and the Doctor of Philosophy in Computer Engineering from Korea University (Seoul campus), Republic of Korea in 2009 and 2016, respectively. He is currently a faculty member of FIT, HCMUTE. His research interests include distributed computing, real-time embedded systems, artificial intelligence for Internet of things, and cyber security.
Email: samnx@hcmute.edu.vn. ORCID: https://orcid.org/0009-0005-7225-9104

Han Nguyen is currently an undergraduate student majoring in Computer Science at the University of South Florida, United States of America. Her research interests include artificial intelligence, machine learning and cyber security.
Email: han316@usf.edu. ORCID: https://orcid.org/0009-0007-3684-6226