Approach using Transforming Structural Data into Image for Detection of Malicious MS-DOC Files based on Deep Learning Models

Shaojie Yang, Wenbo Chen*, Shanxi Li and Qingxiang Xu
Lanzhou University, Lanzhou, China
E-mail: {yangshj18,chenwb,lisx,xuqx}@lzu.edu.cn

***Abstract*—Malicious MS-DOC file has a long history in cybersecurity and has rapid growth with tremendous appearance of advanced persistent threat (APT) attacks. Due to its obfuscation and complexities, regular detection methods are not ideal, and the specific detection methods are limited, either. This paper presents a new approach for malware detection of MS-DOC files. Inspired by analysis of MS-DOC files and tremendous success made by convolutional neural network (CNN) in the field of feature identification, especially image identification, a new approach including data extraction and conversion is designed to identify MS-DOC malicious files and benign files. Based on three CNN models, experiment results show that the accuracy rate of detection for test dataset reaches 94.09%, and in simulated zero-day malware detection experiment, the average accuracy rate reaches 94.70%. The approach proves the feasibility of MS-DOC malicious file detection based on convolutional neural network and proposes a new idea to detect zero-day MS-DOC malware.**

I. INTRODUCTION

The Word Binary File Format is a persistence format that supports word processing tasks for content in documents and document templates. These tasks include authoring and manipulating text, images, tables, and the layout of pages, and managing custom XML schemas that are associated with document content. In this paper, we describe these files as MS-DOC files or doc files.

In recent years, sending victims malicious doc files to invade the operation system has become a common method, especially in advanced persistent threat (APT) attacks. There are three reasons why the doc files have become so popular in attack activities. First, the malicious code embedded in the doc files is widely used. Second, the doc file has many users and complex structure. In addition, it is easy to embed an object in the doc file without being discovered [1].

In the APT attack, hackers always use social engineering techniques or malware, or combine both, so the doc file is an ideal carrier. Besides, the fly always changes by recompiling malware code and using encryption for obfuscation to avoid detection, and complex structures of the doc file provides a good condition for mutation [2]. Because of the wide-scale operations of APT attacks, the MS-DOC malicious files are becoming an agency for cybersecurity.

However, there's no specific way to detect malicious doc-format files. Most protection solutions have utilized the traditional detection as their primary choice, which consists of static analysis based on signature matching [3] and dynamic behavior analysis based on the detection of sensitive API calls [4]. These methods have an ideal rate on detecting known malware. However, they also have limitation in detecting the malware encrypted, disguised [5], or based on undiscovered attack methods [6], which have been common in new attack activities.

Recently, machine learning and deep learning has become a research point; some researchers combine them with security detection and get some positive results.

Yao Wang and his team have some research on the detection of malicious JavaScript code. They adopted Stacked denoising auto-encoder (SDAE) to extract high-level features from JavaScript code and take logistic regression as a classifier to distinguish between malicious and Benign JavaScript. The experimental results indicated that this process could detect malicious code with an accuracy of 95%, and the false positive rate can be reduced to 4.2% [7].

Sitalakshmi et al. are committed to executable-detection research. Firstly, they converted executable files into grayscale maps with data visualization technique, then the maps was classified with convolutional neural networks. After classification, samples in the same class were reversed and their API calls were extracted to make a series of similarity mining. Finally, the data was utilized to train a support vector machine model (SVM). The test result showed that the accuaracy of identification would vary in different method, and the highest accuracy could reach over 95% [8].

Wookhyun et al. proposed a new method to detect zero-day flash malware. They extracted features from malicious samples manually and constructed two dimensions matrix with the features for each sample, and then the matrix was used to train the CNN model. The result shows that with the increase of training samples, the accuracy will develop, too and finally remains above 97% [9].

Aviad Cohen et al. proposed a new structural feature extraction methodology –SFEM, to detect unknown malicious XML-based documents with machine learning algorithms. They collected 830 malicious files and 16180 files to evaluate SFEM. The result showed that SFEM had better performance than the most top, leading anti-virus engine in detecting malicious docx files, such as The-AVAST anti-virus engine achieved a TPR of only 0.777, while SFEM can achieve 0.97. What's more, SFEM can also detect unknown XML-based Office documents, which proved that detection approaches based on machine learning have its own advantages (versus signature-based approaches) [10].

struct of benign doc files	struct of macros virus
'\x01CompObj'	'\x01CompObj'
'\x05DocumentSummaryInformation'	'\x05DocumentSummaryInformation'
'\x05SummaryInformation'	'\x05SummaryInformation'
'1Table'	'1Table'
'Data'	'Data'
	Macros/G9b4Gjelvo/\x01CompObj'
	'Macros/G9b4Gjelvo/\x03VBFrame'
	'Macros/G9b4Gjelvo/f'
	'Macros/G9b4Gjelvo/o'
	'Macros/PROJECT'
	'Macros/PROJECTwm'
	'Macros/VBA/FVXAnubdU4'
	'Macros/VBA/G9b4Gjelvo'
	Macros/VBA/ThisDocument'
	'Macros/VBA/_VBA_PROJECT'
	'Macros/VBA/dir'
'WordDocument'	'WordDocument'

Table. 1. Structure of MS-doc files

In this paper, we proposed a new detection approach for MS-DOC malicious files based on data visualization and deep learning. We presented the following main contributions:

1) We gave an analysis of malicious doc files, mainly focused on structures and suspicious features.
2) Then we provided a new approach to process doc files, including data extraction and conversion.
3) After that, we applied three CNN deep learning models to classify the samples and evaluate our method.
4) Finally, we prepared some simulated zero-day samples and tested them with the models to evaluate the models' capability of unknown malware detection.

II. File Analysis

Embedding of malicious codes may cause the difference between the malicious MS-DOC file and the benign one in a number of respects, such as structure and components. So an analysis of doc files is needed for identifying malicious files.

A. Analysis for macros virus

The MS-DOC file, like other OLE binary files, has clear structure and composition. Table 1. presents structures of a standard benign doc file and a macros virus file. Compared with the normal structure, the malware structure has several Macros streams, which might consist of VBA script. Since many benign files also have VBA script to finish some operations automatically, a more detailed analysis is needed.

We extracted the VBA macros code from the malicious malware. As Fig. 1 reveals, the code includes some suspicious sentences marked as red color, such as strange executable file name and calls of sensitive APIs, all these features appear usually in a malware. As a result, the sample could be judged as a macros virus.

B. Analysis of other malware

Contrast to macros virus, other malware are more difficult to analyze and detect as it has the same or similar structure with the benigns. Yet, there are still some distinct features which may help identify these files manually.

```
...

Private Sub Docement_Open()
If ActiveDocument.Variables("wykYqM").Value <> "juryt" And Not gKelhjy("VmRemoteGuest.exe")
And Not gKelhjy("tee.exe") Then egsdDwwbNSzoBn
...
End Sub

Private Function gKelhjy(prfs As String) As Boolean
...
Set uSgjr = ugkFg.ExecQuery("Select * from Win32Process Where Name = '" & prfs & "'")
...
End Function
```

Fig. 1. VBA macros from a malicious sample

```
00000640: 00 78 00 78 00 78 00 78  00 1A 00 78 00 78 00 78  .x.x.x.x...x.x.x
00000650: 00 78 00 78 00 78 00 78  00 78 00 78 00 78 00 78  .x.x.x.x.x.x.x.x
00000660: 00 78 00 78 00 78 00 78  00 78 00 78 00 78 00 78  .x.x.x.x.x.x.x.x
00000670: 00 78 00 78 00 78 00 78  00 78 00 78 00 78 00 05  .x.x.x.x.x.x.x..
00000680: 00 78 00 78 00 78 00 78  00 78 00 1A 00 78 00 78  .x.x.x.x.x...x.x
00000690: 00 78 00 78 00 78 00 78  00 78 00 78 00 78 00 78  .x.x.x.x.x.x.x.x
000006A0: 00 78 00 78 00 78 00 78  00 78 00 78 00 78 00 78  .x.x.x.x.x.x.x.x
000006B0: 00 78 00 78 00 78 00 78  00 78 00 78 00 78 00 78  .x.x.x.x.x.x.x.x
000006C0: 00 05 00 78 00 78 00 78  00 78 00 78 00 11 00 78  ...x.x.x.x.x...x
000006D0: 00 78 00 78 00 78 00 78  00 78 00 78 00 78 00 78  .x.x.x.x.x.x.x.x
000006E0: 00 78 00 78 00 78 00 78  00 78 00 78 00 78 00 78  .x.x.x.x.x.x.x.x
000006F0: 00 05 00 78 00 78 00 78  00 78 00 78 00 0C 00 78  ...x.x.x.x.x...x
00000700: 00 78 00 78 00 78 00 78  00 78 00 78 00 78 00 2E  .x.x.x.x.x.x.x..
00000710: 00 78 00 78 00 78 00 0B  00 09 0E C6 02 D8 66 A6  .x.x.x........f.
```

(a)

```
00050240: 63 30 6C 61 6E 74 6A 79  31 7A 2B 30 7A 35 35 32  c0lantjylz+0z552
00050250: 31 71 36 50 6A 68 36 32  7A 32 62 56 48 4D 67 71  1q6Pjh62z2bVHMgq
00050260: 74 2B 63 34 32 2B 2F 64  4B 43 70 6A 32 57 47 70  t+c42+/dKCpj2WGp
00050270: 39 71 52 4C 6B 37 55 41  65 45 5A 4A 72 31 47 75  9qRLk7UAeEZJr1Gu
00050280: 71 6D 38 5A 75 59 49 45  75 33 4F 32 31 63 52 72  qm8ZuYIEu3O21cRr
00050290: 55 5A 5A 6F 6A 38 6E 32  4B 4D 34 4E 66 76 41 48  UZZoj8n2KM4NfvAH
000502A0: 32 34 37 32 79 38 39 2F  46 65 39 65 78 69 50 49  2472y89/Fe9exiPI
000502B0: 6B 30 4D 6B 37 35 66 7A  4C 4B 34 41 69 2F 77 42  k0Mk75fzLK4Ai/wB
000502C0: 55 7A 35 53 71 72 39 71  66 49 66 73 4F 4D 62 6E  Uz5Sqr9qfIfsOMbn
000502D0: 37 63 33 6E 47 48 57 32  45 2B 4E 50 49 43 35 44  7c3nGHW2E+NPIC5D
000502E0: 68 51 50 6F 49 64 78 6F  39 6E 65 61 44 76 6D 6D  hQPoIdxo9neaDvmm
000502F0: 2F 43 30 70 41 2F 77 41  31 5A 46 45 2B 75 74 66  /C0pA/wA1ZFE+utf
00050300: 48 41 76 4C 2F 41 4D 36  77 66 39 6A 58 64 54 36  HAvL/AM6wf9jXdT6
00050310: 2F 6C 35 65 76 72 6D 7A  30 70 2F 47 6F 39 63 62  /l5evrmz0p/Go9cb
```

(b)

Fig. 2. Unmeaningful characters (a) and encrypted strings (b) from a malicious sample

One of the features is massive unmeaningful characters, which usually are filled in malware to trigger one or more vulnerability or avoid detection from anti-virus solutions such as junk instruction and encrypted code, like Fig. 2 shows. However, some tricksters will generate lots of unmeaningful files and fill with trash codes for fun, so it cannot be a definitive evidence.

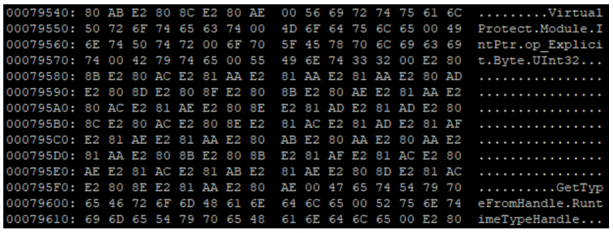
Another feature is still suspicious code. Like Fig. 3 shows, which include sensitive API strings and suspicious executable filenames. It may be significant for identifying malware.

However, the analysis also has some drawbacks. For example, the suspicious codes may be hidden in massive components and locating the features is a time-consuming process. Furthermore, maybe known features detection is easy for professional solutions, if the feature is encrypted, hidden or changed, it might not work so well.

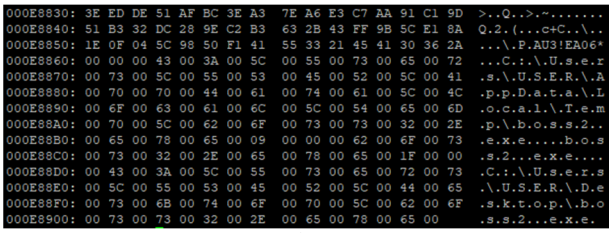
Deep learning is a new machine learning technique, which shows a high accuracy in feature identification and image classification. Based on the conclusion of file analysis, we believe that it is efficient for MS-DOC file detection with deep learning models.

III. Detection Methodology

A. Data conversion



(a)



(b)

Fig. 3. API calls (a) and suspicious code (b) from a malicious sample

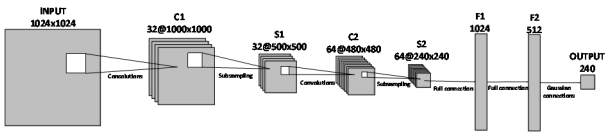


Fig. 4. Modified Lenet-5 structure.

Considering general application of CNN in image identification as well as the results of our predecessors [11,12,13], converting the files into images is necessary. The advantage of image conversion is that the deep learning model has a more developed and productive application in image classification.

Fig. 3 and Fig. 5 indicates that the files can be interpreted as hex streams, and every two hex characters can be interpreted as one 8-bit values. Thus, we can generate a pixel with the value and finally generate an 8-bit scale gray image.

B. Convolutional neural networks

Convolutional neural network is a type of neural network models which has a grid pattern like images and others. The model gets inspiration from the organization of animal visual cortex and is designed to automatically and adaptively learn spatial hierarchies of features, from low-to-high level patterns [14]. CNN models have developed rapidly from Le-Net5 [15] to AlexNet [16] and VGG [17] with the tremendous promotion of accuracy. Until now Convolutional neural networks have had successful achievement in image classification.

A CNN can be considered as a series of repeated structures that consist of a convolutional layer and pooling layer. In a convolutional layer, the coordinates $u_{i,j}$ in the feature map can be written as

$$u_{i,j}^{[k]} = f\left(\sum\sum_{s=0}^{m-1}\sum_{t=00}^{n-1} w_{s,t}^{[k,c]} x_{(i+s),(j+t)}^{[c]} + b^{[k]}\right)$$

where $u^{[k]}$ is the feature map, k is the index of the filter, $x^{[c]}$ is the previous feature map, c is a channel, $w^{[k,c]}$ is the weight, and $b^{[k]}$ is the bias. The size of the filter is m × n and f is an activation function [18].

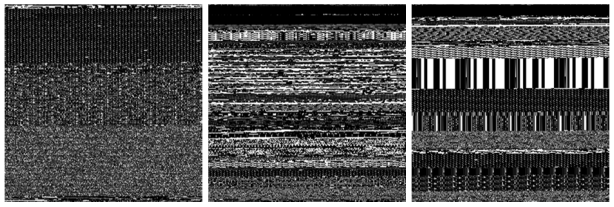


Fig. 5. Images of malicious samples

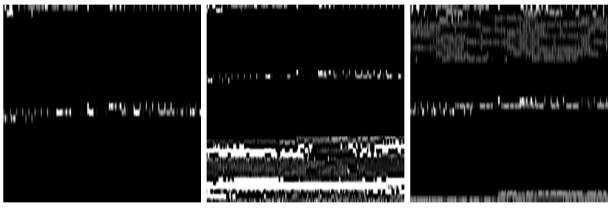


Fig. 6. Images of benign samples

The function of the convolutional layer is doing convolution computation to extract features. Each layer has some convolution kernels. The input image will be converted into a matrix and then be convolved with the convolution kernel at the layer. The pooling layer is supposed as a feature filter to reduce the dimension of data generated in the previous convolution layer and ensure the main features.

In this work, Le-Net5 model is adopted, which structure is shown in Fig. 4. The input layer is the gray image converted from .doc samples with a size of 1024×1024. C1 and C2 are convolution layers which have a kernel size of 4×4 and 1×1, Moreover, a kernel quantity of 32 and 64. The output of the convolution layers are matrices which have a dimension equals to kernel quantity. S1 and S2 are pooling layers with the same kernel size of 4×4. The number of neurons in F1 layer is 1024 and 2 in F2 layer. The output layer contains two neurons determined by two target groups of benign and malicious samples.

In addition, we will also adopt AlexNet and VGG, which are the improved CNN models, to test other CNN models' capability in the detection of malicious doc-format files.

IV. EXPERIMENT AND RESULTS

A. Data collection

To validate the detection of our approach, we collected various samples of malicious and benign OLE Microsoft Word documents (*.doc) totaling 1796 unique documents as Table. 2 shows, which include 978 malicious files and 818 benign files. The malicious files were gathered from VirusTotal, and benign files were from Internet.

B. Data Processing

Firstly, the samples were simplified. Considering that "Table" section and "Data" section are usually table and text streams, they were cut and the rest generated a new sample. The extraction will influence the accuracy of detection, which will be solved in future works.

After that, each sample was converted into a hex stream and then generated an 8-bit gray map which had a resolution of

Type of datasets	Number of samples		Total
	malicious	benign	
Training datasets	873	722	1595
Test datasets	105	96	201

Table. 2. Statics of dataset.

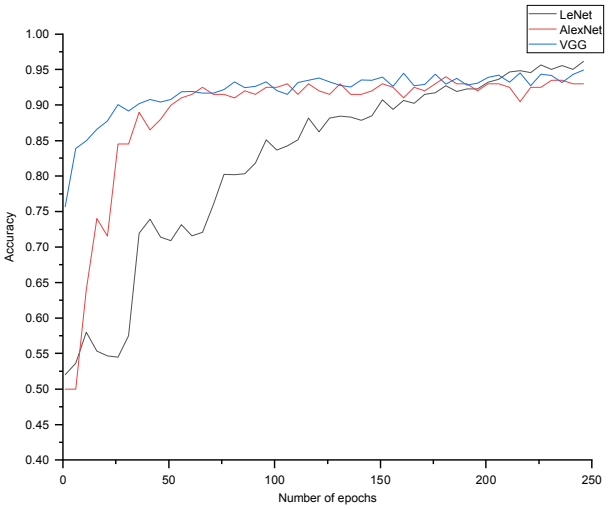


Fig. 7. Accuracy of the validation set with different models.

1024×1024.The grayscale images of the extracted data are shown in Fig. 5 and Fig. 6.

Finally, two new datasets were created. One part was used for training and the other for the validation, the concrete allocation is shown in Table. 2.

C. *Model training and testing*

The models were trained with the training dataset in different epochs and the training accuracy are shown in Fig. 7.

To test models' accuracy and Robustness, each model was trained ten times and was tested with the validation dataset after training to evaluate accuracy of the models. As Table. 4 presents, the accuracy of detecting samples all reached above 90%, and VGG got the highest single accuracy of 94.94% as well as average accuracy of 94.09%.

D. *Zero-day simulating detection*

In this work, we collected eight new-type malicious samples. After testing general samples, a new dataset consist of eight new-type malicious samples and twelve benign samples was tested with trained models, therefore we could simulate zero-day malware detection and evaluate whether the models have ability to identify unknown malware.

The test results of zero-day malware simulating detection are shown in Table. 5. All detection accuracy of the three models could reach above 85%, which indicated that all models have a certain detection capability of unknown malicious doc-format files. Besides, Alexnet reached an average accuracy of 94.70%, which was much higher than other two models.

V. CONCLUSION AND DISCUSSION

In this paper, we proposed a new detection method of malicious doc-format files with CNN models. A method of data visualization to convert samples into the image was adopted,

Num	Accuracy		
	Le-Net5	AlexNet	VGG
1	92.50%	92.50%	93.38%
2	94.89%	93.50%	94.31%
3	92.67%	92.50%	94.64%
4	93.67%	93.00%	93.82%
5	91.44%	92.50%	93.40%
6	94.56%	94.50%	93.61%
7	92.89%	91.50%	94.24%
8	94.22%	92.00%	94.94%
9	92.78%	92.00%	93.75%
10	94.00%	93.00%	94.78%
Average	93.36%	92.70%	94.09%

Table. 3. Accuracy of test dataset with different models.

Num	Accuracy		
	Le-Net5	Alexnet	VGG
1	84.00%	92.00%	84.49%
2	89.67%	95.00%	83.96%
3	90.33%	95.00%	83.96%
4	89.67%	94.00%	95.19%
5	89.33%	95.00%	83.96%
6	84.25%	96.00%	89.30%
7	89.33%	95.00%	83.96%
8	89.33%	93.00%	89.30%
9	89.58%	96.00%	83.96%
10	89.25%	96.00%	83.96%
Average	88.47%	94.70%	86.20%

Table. 4. Accuracy of zero-day simulating dataset with different models.

which could transform a file-detection problem into an image-recognition problem. Three different models were employed, and the test accuracy could reach 94.09%. In addition, some unknown samples (which type were never trained and tested before) were used as simulated zero-day malware to test the models' detection capability of unknown or zero-day malicious files; the results showed that the models could reach a highest average accuracy of 94.70%.

The experiment confirmed the feasibility of applying deep learning in the office malware detection and provided a new proposal to detect zero-day and unknown MS-DOC malware.

The samples still have too many idle components for malware detection, future work could focus on the promotion of models and denoising of the inputs to improve the accuracy as well as reduce the costs. Moreover, since VGG had the highest accuracy in the detection of known samples while Alexnet had the highest accuracy in the detection of unknown samples, it is considerable to combine both in practice, which could be another research point.

ACKNOWLEDGMENT

We would like to thank the High-Performance Computing Center at Lanzhou University of Lanzhou, China, for supporting this research. Many thanks also to VirusTotal for granting us complimentary access to their samples.

REFERENCES

[1] Gao, Y., & Qi, D. (2011, 10-13 July 2011). *Analyze and detect malicious code for compound document binary storage format.* Paper presented at the 2011 International Conference on Machine Learning and Cybernetics.

[2] Tankard, C. (2011). Advanced Persistent threats and how to monitor and deter them. *Network Security, 2011*(8), 16-19. Retrieved from http://www.sciencedirect.com/science/article/pii/S1353485811700861. doi:10.1016/s1353-4858(11)70086-1

[3] Bayer, U., Moser, A., Kruegel, C., & Kirda, E. (2006). Dynamic analysis of malicious code. *Journal in Computer Virology, 2*(1), 67-77.

[4] Bergeron, J., Debbabi, M., Desharnais, J., Erhioui, M. M., Lavoie, Y., & Tawbi, N. (2001). Static detection of malicious code in executable programs. *Int. J. of Req. Eng, 2001*(184-189), 79.

[5] Bilge, L., & Dumitraş, T. (2012). *Before we knew it: an empirical study of zero-day attacks in the real world.* Paper presented at the Proceedings of the 2012 ACM conference on Computer and communications security.

[6] Rad, B. B., Masrom, M., & Ibrahim, S. (2012). Camouflage in malware: from encryption to metamorphism. *International Journal of Computer Science and Network Security, 12*(8), 74-83.

[7] Wang, Y., Cai, W. D., & Wei, P. C. (2016). A deep learning approach for detecting malicious JavaScript code. *Security and Communication Networks, 9*(11), 1520-1534. Retrieved from <Go to ISI>://WOS:000379053800021. doi:10.1002/sec.1441

[8] Venkatraman, S., & Alazab, M. (2018). Use of Data Visualisation for Zero-Day Malware Detection. *Security and Communication Networks, 2018*, 13. Retrieved from <Go to ISI>://WOS:000453813700001. doi:Unsp 1728303 10.1155/2018/1728303

[9] Jung, W., Kim, S., & Choi, S. (2015). *Poster: deep learning for zero-day flash malware detection.* Paper presented at the 36th IEEE symposium on security and privacy.

[10] Cohen, A., Nissim, N., Rokach, L., & Elovici, Y. (2016). SFEM: Structural feature extraction methodology for the detection of malicious office documents using machine learning methods. *Expert Systems with Applications, 63*, 324-343. Retrieved from <Go to ISI>://WOS:000382273700026. doi:10.1016/j.eswa.2016.07.010

[11] Kancherla, K., & Mukkamala, S. (2013). *Image visualization based malware detection.* Paper presented at the 2013 IEEE Symposium on Computational Intelligence in Cyber Security (CICS).

[12] Kim, J. Y., Bu, S. J., & Cho, S. B. (2018). Zero-day malware detection using transferred generative adversarial networks based on deep autoencoders. *Information Sciences, 460*, 83-102. Retrieved from <Go to ISI>://WOS:000441494000006. doi:10.1016/j.ins.2018.04.092

[13] Makandar, A., & Patrot, A. (2015). *Malware analysis and classification using Artificial Neural Network.* Paper presented at the 2015 International conference on trends in automation, communications and computing technology (I-TACT-15).

[14] Yamashita, R., Nishio, M., Do, R. K. G., & Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. *Insights Imaging, 9*(4), 611-629. Retrieved from https://www.ncbi.nlm.nih.gov/pubmed/29934920. doi:10.1007/s13244-018-0639-9

[15] Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*(11), 2278-2324. Retrieved from <Go to ISI>://WOS:000076557300010. doi:Doi 10.1109/5.726791

[16] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). *Imagenet classification with deep convolutional neural networks.* Paper presented at the Advances in neural information processing systems.

[17] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[18] Kasugai, T., Tsuzuki, Y., Sawada, K., Hashimoto, K., Oura, K., Nankaku, Y., & Tokuda, K. (2018). *Image Recognition Based on Convolutional Neural Networks Using Features Generated from Separable Lattice Hidden Markov Models.* Paper presented at the 2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC).