Computer Networks 178 (2020) 107275

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Accurate and fast URL phishing detector: A convolutional neural network approach

Wei Wei [a,b,*], Qiao Ke [c], Jakub Nowak [d], Marcin Korytkowski [d], Rafał Scherer [d], Marcin Woźniak [e]

[a] School of Computer and Engineering, Xi'an University of Technology, Xi'an, 710048, China
[b] Shaanxi Key Laboratory for Network Computing and Security Technology, China
[c] School of Software, Northwestern Polytechnical University, Xi'an, 710129, China
[d] Częstochowa University of Technology, Al. Armii Krajowej 36, Częstochowa 42-200, Poland
[e] Faculty of Applied Mathematics, Silesian University of Technology, Kaszubska 23, Gliwice 44-100, Poland

ARTICLE INFO

MSC:
68M11
68T99

Keywords:
Phishing
Urls
Machine learning
Convolutional neural network

ABSTRACT

Along with the development of the Internet, methods of fraud and ways to obtain important data such as logins and passwords or personal sensitive data have evolved. One way of obtaining such information is to impersonate a page the user knows. Such a site usually does not provide any services other than collecting sensitive information from the user. In this paper, we present a way to detect such malicious URL addresses with almost 100% accuracy using convolutional neural networks. Contrary to the previous works, where URL or traffic statistics or web content are analysed, we analyse only the URL text. Thus, the method is faster and detects zero-day attacks. The network we present is appropriately optimised so that it can be used even on mobile devices without significantly affecting its performance.

1. Introduction

The enormous amount of data in computer systems, large organized datasets and powerful parallel computing machines caused rapid adoption of machine learning in real-world systems. Machine learning allows automatic model building from data or using a set of rules governing the problem (such as the game of chess or Go). Modern machine learning methods are able to solve nearly any real-world tasks [1–3], especially when we have a lot of data to our disposal. They can iteratively adapt to incoming new data or import knowledge from other models by so-called transfer learning. Nowadays we face the problem of analyzing big data that is impossible to read and understand by humans. The same applies to modern methods of crimes – the aforementioned rapid development of computing techniques allowed executing cyber attacks at an unprecedented scale. Malware, denial-of-service, man-in-the-middle or SQL injection attacks can be launched by vast networks of devices (botnets) with magnitudes unseen before.

In this paper, we present a method dealing with zero-day phishing with almost perfect accuracy by machine learning. Phishing is a fraud by obtaining sensitive information by mimicking a reputable sender in a communication channel. Phishing is extensively used to obtain information or money under false pretences. Usually, a message contains malicious software or links. It is relatively easier to prepare a counterfeit message than to break system security. Moreover, phishing campaigns can be launched relatively cheaply from anywhere in the world due to openness and anonymity of the Internet. The phishing message contains logotypes and texts that are intended to mimic legitimate ones. Many sources claim that more than 90% of data breaches are performed with some sort of phishing. Moreover, most ransomware is delivered by phishing. Attackers change domain and subdomains names to make URLs look like legitimate ones. Very often, even experienced users do not check carefully URLs they click. These attacks relate mainly to pages that present a specific value to the offender. First of all, the parties that process the data needed for bank transfers are exposed. Thus, financial institutions are especially concerned with phishing. Another example is the theft of e-mail credentials what is easier because there is no multistage authorisation as in the case of bank transactions. Moreover, attacks on social media accounts and theft of Internet identity are more and more common. Many users believe that the HTTPS protocol and "green padlock" in the browser guarantees security. One of the last reports by Phishlabs [4] shows that in the third quarter of 2018 as much as 49% of phishing websites used SSL certificates.

In this work, we are mainly interested in the following attack methods. The most commonly used method is the registration of domains with a slightly changed structure (typosquatting) [5]. This involves omitting or adding one character in the address, domain registration

* Corresponding author.
E-mail addresses: weiwei@xaut.edu.cn (W. Wei), qiaoke@nwpu.edu.cn (Q. Ke), jakub.nowak@pcz.pl (J. Nowak), marcin.korytkowski@pcz.pl (M. Korytkowski), rafal.scherer@pcz.pl (R. Scherer), marcin.wozniak@polsl.pl (M. Woźniak).

https://doi.org/10.1016/j.comnet.2020.107275
Received 26 January 2020; Received in revised form 1 April 2020; Accepted 17 April 2020
Available online 28 April 2020

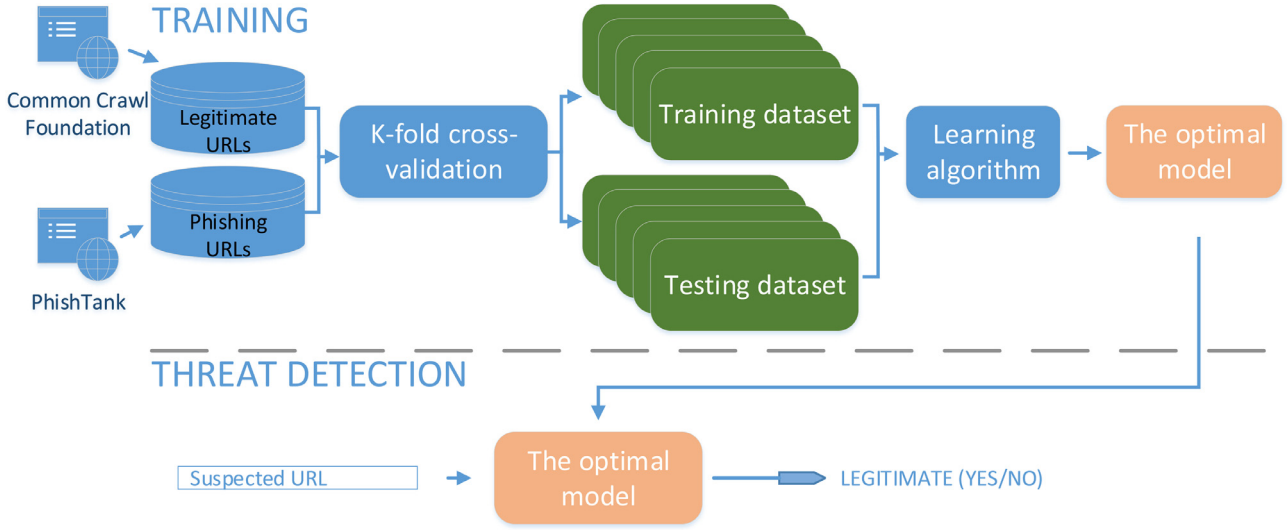

Fig. 1. Simplified machine learning workflow used in the paper. We do not use any hand-crafted features and the model can be updated any time if the new phishing URL resources are available.

without a dot after www (wwwdomainname), adding the dash (www-domainname, domainname-anyname). There is a similar bitsquatting technique [6], which involves changing one bit, i.e. one key press different, in the URL to cause a network error or error in the computer hardware. If the error causes one bit to be flipped the user is moved to the undesired page. In this case, the only way to avoid the attack is an additional system to check if the given string definitely belongs to the page we expect. Another method is a homograph attack [7]. It aims to convert characters into a similar graphics look, for example, capital 'i' and small 'l' (I l), zero and O, etc. The user does not notice the difference in a link consisting of this kind of conversion. In this type of attacks, the visually impaired are particularly vulnerable. Similarly, we can also exchange one character into two ('m' changed to 'rn', 'ci' to 'G', 'cl' to 'd' and vice versa). Another method is the punycode representation [8]. It consists in representing Unicode characters with a set of allowed characters in the name, i.e. ASCII letters, numbers and dashes.

Anti-phishing systems are certainly not a new research area. They can be generally divided into list-based and machine learning-based ones. Blacklists seem at first sight an easy way to solve the problem. Yet, phishing sites are active for a very short period, making blacklisting usually useless. Blacklist-based systems are not able to detect new and so-called zero-day attacks and require frequent updates. As we have already mentioned, the number of phishing attacks does not stop to grow every month; neither do the blacklist sizes. That requires substantial system resources to compare URL strings. Thus, we believe that machine learning approaches are better suited to the task than plain blacklisting. Most of the proposed so far machine learning systems are based on engineered (hand-crafted) features. In URL-based approaches, the features describe the URL addresses or their statistics. In content-based approaches, the content of the linked website is analysed.

All the aforementioned groups of phishing methods are aimed at deceiving the user by impersonating a different page using an address very similar to the original one. That is why in our research we focus on analysing only URLs using our experience of analysing this type of data using convolutional networks. The process is outlined in Fig. 1. Unlike the blacklist-based approaches, our method detects zero-day attacks by analysing suspected URLs. We use convolutional neural networks with certain modifications. Usually, word-level encoding or some word-embedding are used to present text data to neural networks. That usually requires the use of a word dictionary and makes the system language-dependent. In the paper, we use one-hot character-level encoding to input URLs to the neural network. The rationale behind this choice is that URLs are composed of words in various languages or strings not constituting words. Then, the encoded string is treated as an image by a convolutional neural network. Our method is much faster than other approaches listed in Section 5 as it does not have to analyse the suspected page content. At the same time, it has better accuracy. We also compare the presented method with [9] by implementing their recurrent LSTM network and training on the same dataset. Our method gives slightly better accuracy and is much faster to train. Through this research, we highlight the following features and contributions of the proposed model.

- We present the first convolution neural network-based anti-phishing system.
- Our work provides new insights, showing that the URL dictionaryless analysis provides nearly 100% accurate security.
- Accurate, solely URL-based detection provides zero-day defense, contrary to blacklist or white-list approaches (unless a website with the legitimate domain is compromised).
- The trained system is small and fast and can be used effortlessly on mobile devices. Moreover, newer mobile chipsets have special instructions for machine learning computations.

The remainder of the paper is organised as follows. In Section 2, we discuss selected previous works on phishing prevention. Convolutional neural networks are described in Section 3. The method proposed in the paper is presented in Section 4. Experiments on URLs taken from PhishTank and Common Crawl websites are shown in Section 5. Finally, conclusions and discussions of the paper are presented in Section 6.

2. Related works

The identification of phishing attacks has repeatedly been studied, but none of the systems was utterly perfect. Some of the solutions, despite their accuracy, required complicated calculations, which hindered their subsequent use. As we aforementioned, the easiest method of protection is a blacklist of phishing URL addresses, which is later analysed by a browser, antivirus system or firewall [10–12]. The blacklist usually has to be created by merging many sources of phishing addresses. Similarly, it is possible to curate a white list of safe websites. A system white-listing legitimate websites is created in [13]. The authors claim that white lists are safer than black lists as they are smaller and more accurate than black lists. The system achieved 86.02% accuracy. In [14] the white list is created from websites visited by the user. Counteracting an attack, in this case, is very simple and effective as long as

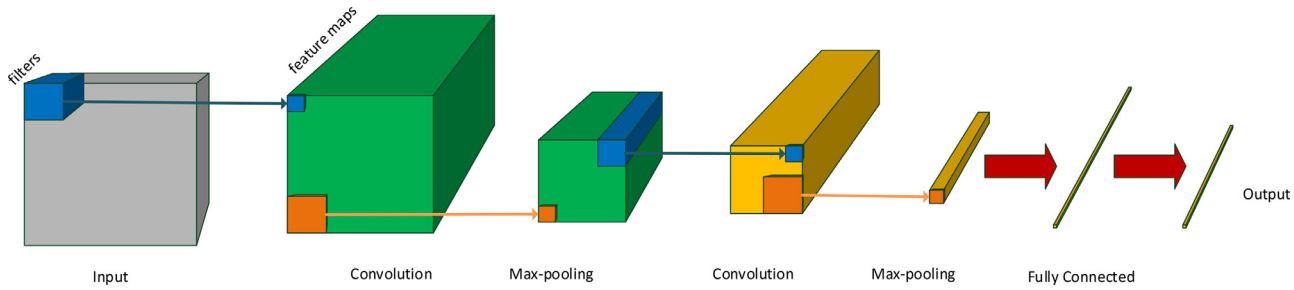

Fig. 2. General diagram of the convolutional neural network.

the address is on the blacklist. The address has to be already identified and added to this list. Another disadvantage is the need to search the list of suspicious addresses whenever we refer to a new URL.

Most of the proposed so far machine learning systems are based on engineering (so-called hand-crafted) features. One of the most cited antiphishing system is CANTINA [15] which uses the most frequent words on the page to check the page in a search engine. The system uses the term frequency-inverse document frequency weighting scheme to compute word frequency on the page. The five terms with the highest scores constitute the page descriptor used to query the search engine. If the domain name is within the top N search results than the web page is legitimate. The newer version, CANTINA+ [16] used additionally fifteen attributes extracted from the page HTML. This system was a breakthrough at the time. The disadvantage of the two solutions is the use of a search engine to find out whether the address certainly matches the desired page causing additional network load. Moreover, the attacker could promote the phishing website in the search engine to make it seem legitimate.

The authors of [17] analyse pages using the support vector machine algorithm (SVM). They use six hand-made features of the URL and achieved 95.80% accuracy. In [18] association rule mining algorithm is used with the term frequency-inverse document frequency weighting to generate detection rules. They achieved 93% accuracy and very good detection speed. Other solutions analyse the whole page with trees DOM [19]. This solution is efficient if the offender makes a mistake in the design of the webpage. If he knows exactly how the system works, he can modify the code of the page so that the system identifies the website as legitimate.

Neural networks were used so far to detect phishing only on the base of some attributes of the URL. In [20] multilayer perceptron (MLP) is used to detect phishing from engineered features with 96% accuracy. There are also other important methods useful in data analysis feature extraction from data sets [21], which find applications in ad-hoc networks [22].

In [23] a neural network classifies addresses using 17 URL features such as the length of the URL address, the occurrence of the IP address etc. The authors called the system "based on Self-Structuring Neural Network", whereas it is a plain multilayer perceptron which size is incremented to reach the minimal error. They achieved 94% accuracy on relatively small URL dataset.

Authors of [24] achieved 97.71% accuracy using a modified MLP with a novel learning scheme. They used 30 features from publicly available URL dataset. All the aforementioned URL-based works used hand-crafted features. The idea is similar to the proposed in the current paper only that the URL is analysed without relying on other web page features, such as HTML, and external databases. The above papers used statistics regarding the URL (extracted features), whereas in [9] the URL text is directly analysed by recurrent neural networks (RNN) what coincides with our approach. RNNs are suitable to model, inter alia, temporal phenomena, in [9] they are used to analyse URL characters sequentially. The authors used a modern version of RNNs called Long Term Short Term memory (LSTM). They obtained 98.76% accuracy, and in the approach presented in this paper, we use similar data. We also analyse the URL text, but with convolutional neural networks with embeddings and one-hot encoding. We are interested to obtain as much details from the text analysis as possible. Although this solution uses only part of the available data from the website it offers very good results what will be shown in Section 5.

3. Convolutional neural network with embedding layer

Artificial neural networks are currently a very dynamically developing field of science. Every day we witness new applications of neural networks. The basic model of the neuron consists of inputs, weights assigned to each input, adder and an activation function. The activation function form affects the learning process. Recently, the Rectified Linear Unit (ReLU) activation function [25], [26] gained popularity, as it reduces the vanishing gradient problem in deep networks. Because a single neuron is not able to solve complicated tasks, we must combine them into neural networks usually called multilayer perceptrons (MLP).

Convolutional neural networks (CNN) evolved from (fully-connected) artificial neural networks. The first CNN resembling today's structure was the Neocognitron network [27]. The Neocognitron was designed to recognize the basic graphics characters in the image. Inspirations for the network model were taken from the discoveries of Hubel and Wiesel in 1959 on the operation of human eyesight receptive field. We owe the Neocognitron one of the layers called max-pooling used today in almost unchanged form. The first real CNN network, which was widely used, was the LeNet 5 network [28]. This network practically solved the problem of recognizing handwritten numbers. A MNIST database containing 30 thousand examples of handwritten numbers from the range (0–9) with a size of 32×32 pixels was created for the research. CNN networks require a high demand for equipment during training; therefore for the next success in image recognition, we had to wait until 2012 when the AlexNet network [29] solved the ImageNet competition better than algorithms based on hand-crafted image features.

Convolutional networks consist of several layers: the convolutionary layer, max-pooling or fully-connected layers (Fig. 2). The basic layer is the convolutional layer. In the case of image processing, its task is to detect characteristic features in the image, at higher and higher level with every layer. Each layer consists of a number of feature maps (FM) depending on the complexity of the problem, and in a well-designed network, each of these layers should detect a different shape. For each such layer there is a set of filters. In the first layer, in the case of RGB images, there are only three FMs, a separate layer for each colour. The size of the filter must be determined by the designer. In the case of image classification, usually square-shaped filters (7×7, 5×5, 3×3) are used, then such a filter traverses each FM vertically and horizontally. In the case of text processing, we try to use a rectangular filter where one size fills the entire FM. This filter analyzes the encoded text only along one axis.

Another layer is the max-pooling layer (Fig. 3). The layer has a two-fold purpose. The first one is the acceleration of the network operation.

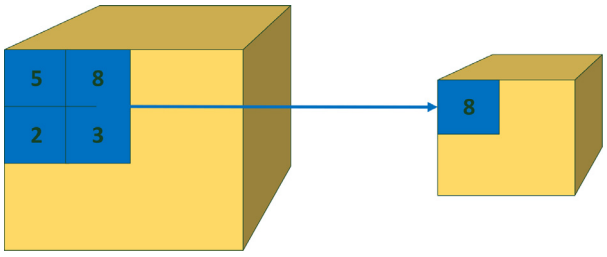

Fig. 3. Max-pooling operation example used after convolutional layers.

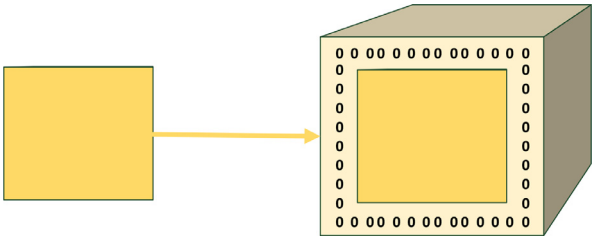

Fig. 4. Example of padding operation where the image is extended to have the resulting image after convolution the same dimensions when the stride is one.

If each FM is reduced by pooling, there are fewer calculations at the next convolution layer. The second one is the improvement of the entire convolutional network performance. By choosing only the most important features, the network achieves better results after the training process. In the first versions of CNNs a subsampling technique was used to calculate the average value from the filter, but this solution increased computation time without improving the network performance.

The dimension of a convolution layer is closely related to the size of the input layer, the size of the filter we apply and the stride (step) for the convolution filter we apply. An image after a convolution will always be smaller than the original one. If we wish to design the network so that the image does not decrease we have to add padding which adds extra outer pixels to the image, see Fig. 4. With this technique, we obtain a larger input image by adding 0 to each map. Most modern machine learning libraries implement auto-padding function that ensure that the size of the layer is optimal for the size of the filter we use.

Usually, the last layers are fully-connected layers composed of the standard neurons described at the beginning of this section forming MLP network. The final feature map matrix is the input to the MLP network allowing final pattern classification to be performed.

4. Phishing detection by convolutional neural networks

The first step in machine learning research is to collect a sufficiently large set of training and testing data. We describe the collected data, text coding for the CNN purposes and the neural network used in experiments.

4.1. Phishing dataset

Our experiments are based on publicly available PhishTank phishing sites database [30]. The database downloaded during the article writing contained 10,604 records. To obtain legitimate websites, the second part of the training dataset was downloaded from the Common Crawl Foundation (http://commoncrawl.org/). The initial experiments were performed with The Moz Top 500 [31] and the Alexa datasets. As they contain top-level domains only, it caused a strong bias towards short, top-domain URLs. Then, we downloaded 10,604 random unique URLs from the Common Crawl database. We wanted the legitimate set not to dominate addresses from the second class. In this case, we have an ideal situation for the classification and a balanced set of 50% of phishing

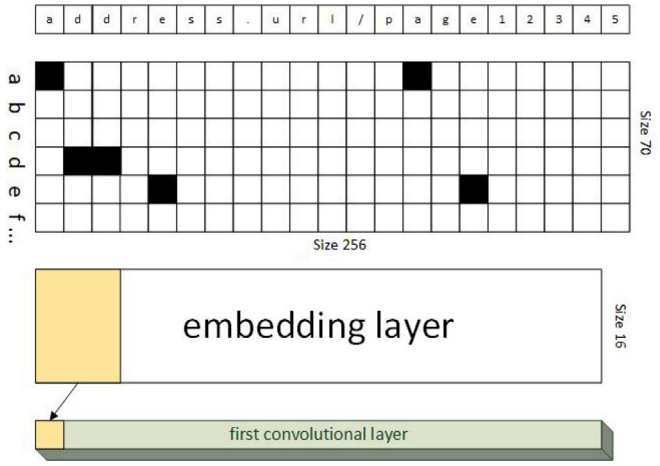

Fig. 5. Example URL address.url/page12345 in the one-hot character-level encoding for the convolutional neural network used in the paper. The first layer is embedding transforming the sparse input into a concise vector of real numbers.

addresses and 50% of the top URL addresses used every day by Internet users. In order to leave as much space as possible to encode characters, each URL from the database was deprived of the http:// or https:// protocol designation. Thus, we did not have to encode 7 or 8 characters that do not contribute to the classification decision. Some examples of the URLs used in the experiments are presented in Table 1. We can see that the legitimacy of a URL is not straightforward, and it is tough for humans to distinguish the two classes. We also believe that using only statistics is also insufficient to detect phishing accurately. The same applies to use solely hand-crafted features for text analysis (Table 3).

4.2. Text coding

A URL is an address that allows locating a website on the Internet. The user encounters it mainly when using a web browser. However, URLs can be requested by applications running in the background such as antiviruses, system updates, etc. Each user's computer uses different applications and at a different time. For URLs, it is rather useless to use dictionaries containing predefined words because each minimal change in the letter in the address can refer to an impersonation attempt. The construction of the URL has been repeatedly addressed in various articles [17,32]. In our experiment, we encode the entire address without dividing its subsequent parts. The condition is that the characters that make up the URL should be in the previously defined set containing 70 unique characters We were inspired here by Zhang et al [33] to present text data in the form of a one-hot vector at the character level. The dictionary consisted of 70 characters:

```
abcdefghijklmnopqrstuvwxyz_0123456789
```

If a character was not from the above alphabet it was removed. An example of a bitmap with encoded URL is shown in Fig. 6. Statistics concerning the data are in Table 2.

In the case of convolutional neural networks with one-hot encoding, we must determine the maximum length of the URL. Analysing all addresses, the average address length was 186 characters. The longest address was 1149 characters. To optimize the network at this stage, we used 256 characters to encode the URLs. Because while designing the network we have to take care of the appropriate dimensions of the convolutional layers, and only 9% of addresses were longer than 256 characters, we decided to shorten the longest addresses to 256 characters. The one-hot character-level encoding scheme with the first CNN layers is shown in Fig. 5.

Table 1
Examples of URLs from the dataset used in the experiments.

Phishing
https://pages-officialsupport.ga/check.php
https://pages-officialsupport.ga/site-verification.html
https://pages-officialsupport.ga/
http://www.dalmer.hu/images/jquery/autoatendimentabb/pontossmiles/regularizar.php
http://www.atualizacao-cadastral-bb.com
http://sysbbclientonlines.tk/
http://smilesgol-premiado2019.cloudaccess.host/areadocliente/b1/index1.php
http://portalbb99mobi.cloudaccess.host/1/pagina-inicial/ http://ocasiao.com.br/pbb/pagina-inicial/
https://apsceses.000webhostapp.com/FB/R/mobile-facebook-verification.htm
Legitimate
http://14-85.com/stories-captured/love-takes-flight/
http://1499934.com/a/20190315/421291.html
http://15.onemp3.co/08-caliber-fuse-box.html
http://akaiwapara.blog64.fc2.com/
https://zone-fitness.fr/categorie/food/recette-sucree/
https://zwemmeninwaalwijk.nl/aquasporten/aquavaria
https://www.tampabayhousehunting.com/homes/5750-oak-hollow-lane/oviedo/f1/32765/89637685/

Table 2
Statistics of the URL dataset used in the experiments.

Number of all URLs	21,208
Number of phishing URLs	10,604 (50%)
Number of legitimate URLs	10,604 (50%)
Number of characters in the CNN input vector	256
Dictionary size for one-hot encoding	70
The longest URL	1149 characters
The shortest URL	5 characters
Average URL size	186

Table 3
Frequency of special characters.

Non-English or Math	Frequency	Comments
Ø	1 in 1000	For Swedish names
π	1 in 5	Common in math
$	4 in 5	Used in business
Ψ_1^2	1 in 40,000	Unexplained usage

4.3. Convolutional neural network

We analyze URLs using convolutional neural networks (CNN) [34]. The architecture we use is very good at analyzing the natural language (NLP), however, the analysis of url addresses in terms of the occurrence of phishing attacks will be something new. Convolutionary networks are well suited for the classification of images [29]. Due to their construction, they are very resistant to distortions in the image. In our case, we want to sensitize the network so that it can detect the address distortions used in the previously described technique. We want to optimize the architecture presented by us so that the network can be implemented in mobile devices with limited memory and computing power.

The one-hot character-level representation of URLs is rather sparse. To improve the performance of the convolutional layers we applied an embedding layer at the network input. Its aim is to translate the sparse input representation into a concise vector of real numbers. The translation (embedding) matrix is developed during trained from data. Moreover, after the training, similarities between terms can be visualized by the t-SNE method [35] This technique has excellent effects in natural language processing [36] where by changing the representation of words we obtain better results for a given classifier. In our case, embedding is mainly designed to speed up the network performance through another mapping of 70 characters from the dictionary using a smaller, hidden vector of real numbers. The rationale for introducing the embedding layer is on the one hand sparsity on the second hand the enormous amount of phishing data. It is challenging to hand-craft features for such dataset, and the embedding layer in the course of data-driven training can fit to the one-hot encoding to concise vector mapping. Our experience shows that the embedding layer of size 16 can easily represent 70 characters. The convolutional network is presented in Fig. 8. The first layer is embedding with output 128×16. Then, there are standard CNN layers, and the last layer is a fully connected layer with two outputs (legitimate/phishing URL).

The Sequence diagram of the proposed method is shown in Fig. 7. Upon the HTTP request from the client, the system checks the URL and decides if it can be requested. Every day the system monitors if there are new phishing addresses in the Phishtank repository and trains the model with a new, larger dataset. The training lasts about three minutes and the URL checking about two milliseconds.

5. Experiments

The CNN method proposed in the paper was compared with other approaches, especially with the LSTM network proposed in [9] by recreating their experiment. Both, CNN and LSTM networks were implemented using the CNTK library (https://github.com/Microsoft/CNTK). Learning and testing has been carried out on the GeForce GTX 1080 Ti GPU with 11GB of RAM installed in an Intel i7-7800X 3.50 GHz 32GB RAM machine. We trained the networks with Stochastic Gradient Descent [37] with minibatch size 45, momentum 0.5. Learning coefficient was gradually lowered, from 0.1 in the first epoch, 0.01 in the second one, 0.001 in the next four epochs and in the rest of training it was set to 0.0005. Dropout rate was set to 0.5. Both networks were trained through 15 epochs, and it took 3.50 min. for CNN and 5.18 min. for LSTM. The

Fig. 6. Example of a part of the input image (inverted) created from a suspected URL. The image is created by character-level one-hot encoding.

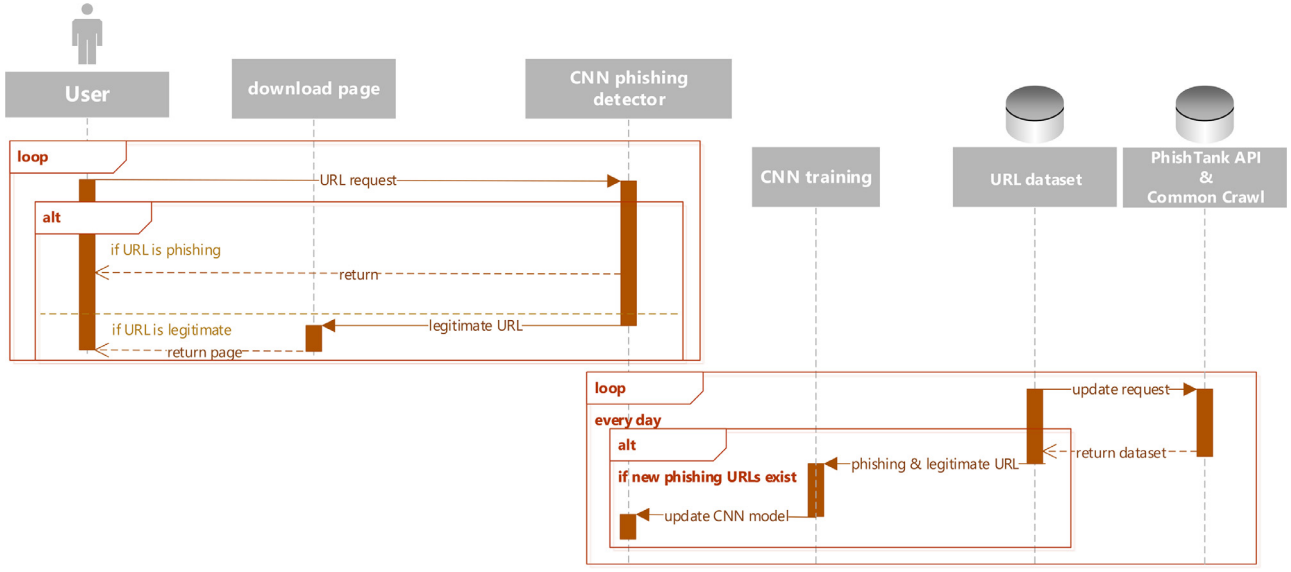

Fig. 7. Sequence diagram of the proposed method. Upon the HTTP request from the client, the system checks the URL and decides if it can be requested. Every day the system monitors if there are new phishing addresses in the Phishtank repository and trains the model with a new, larger dataset. The training lasts about three minutes and the URL checking about two milliseconds.

Table 4
Comparison of the LSTM network from [9] and the CNN network proposed in the paper.

NN type	GPU memory requirements	Parameter set size	Number of URLs per sec.
LSTM	352 MB	880 KB	460/s
CNN	473 MB	502 KB	480/s

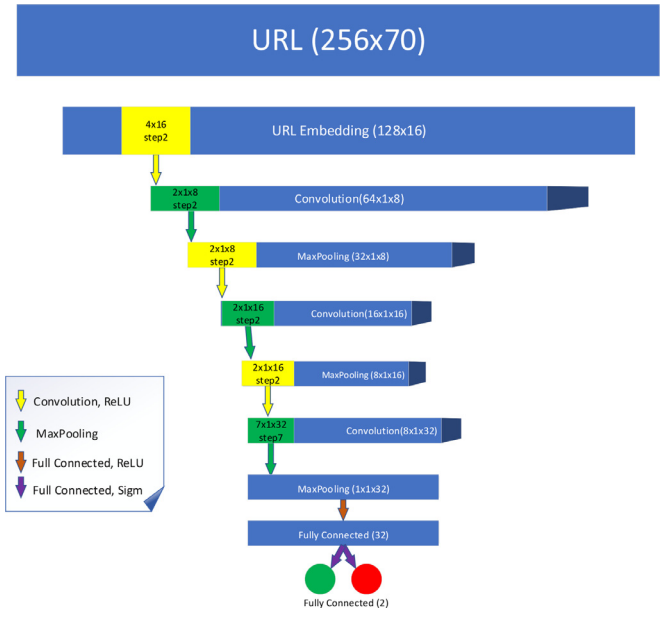

Fig. 8. Convolutional neural network used in the paper.

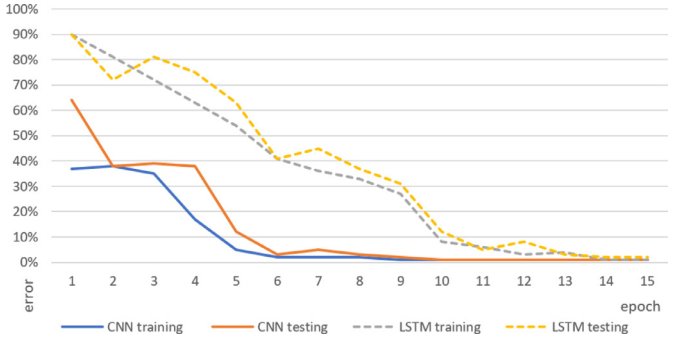

Fig. 9. Training error (accuracy) of the LSTM network from [9] and the CNN network proposed in the paper. In fact, apart from the faster convergence, CNN's epoch is almost 50% faster (shorter computing time on the same hardware).

dataset described in Section 4.1 was divided into 80% training set and 20% testing set and 5-fold cross-validation was applied. The training error for both networks is shown in Fig. 9. Memory requirements and computation times for both neural networks are shown in Table 4. In the case of calculating the memory demand during the training, the given values also contain the loaded training data in similar configurations. The number of examples tested over time includes the time to load both data and the network weights. Testing has been done using the same GPU as for the training. The ROC curves for both networks are shown in Figs. 10 and 11. We show the results for the best network size from Table 6.

As we did not have the same set of data as used in [9], we compared their approach on the dataset collected at the time of the experiments (Section 4.1). Table 5 shows the numbers taken from [9] in the first two rows and the results obtained on the current dataset (row 3 and 4). Table 7 shows the comparison of the accuracy obtained by other approaches using similar datasets and by our method.

Table 5
Comparison of the LSTM network and random forest from [9] on their dataset (two first rows) and LSTM (3rd row) the CNN network proposed in the paper (4th row) on the dataset described in Section 4.1.

Method	AUC	Accuracy	Recall	Precision	F1-score
Random Forest and data from [9]	0.984482	0.9347	0.9328	0.9323	0.9346
LSTM and data from [9]	0.999097	0.987622	0.98927	0.986023	0.987642
LSTM (CNTK and current data)	0.98467	0.985	0.99278	0.978672	0.98568
Proposed CNN	0.99799	0.997916	0.99998	0.995689	0.997840

Table 6
Training error comparison for various sizes of the CNN network proposed in the paper. We changed the number of feature maps in relation to the base value x from Fig. 8.

	Number of feature maps					
Number of conv. layers	0.25x	0.5x	x	2x	4x	8x
2 layers	30.6%	29.8%	12.00%	12.40%	12.30%	12.30%
3 layers	5.0%	4.0%	0.02%	0.02%	1.04%	0.03%
4 layers	4.0%	0.4%	0.02%	0.02%	0.03%	0.03%
5 layers	1.0%	0.5%	0.04%	0.05%	1.12%	1.12%
6 layers	1.5%	1.2%	0.89%	3.30%	3.00%	1.40%

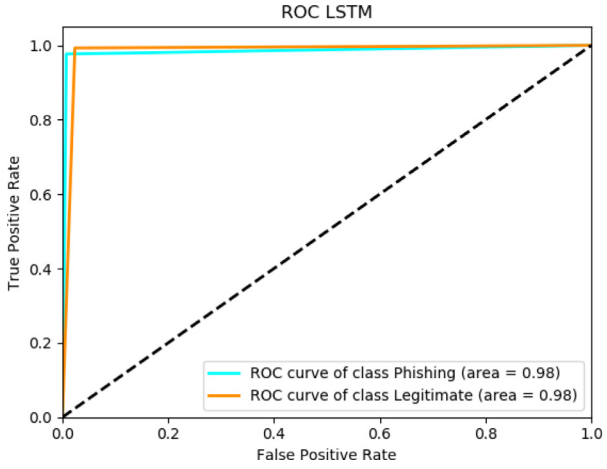

Fig. 10. ROC curve for the LSTM network from [9] implemented on CNTK and trained in the current dataset described in Section 4.1.

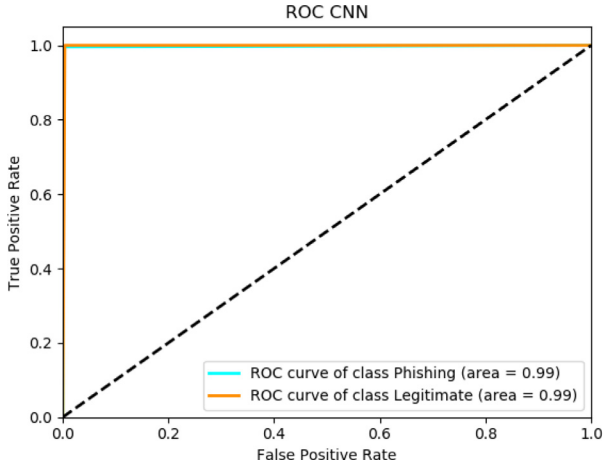

Fig. 11. ROC curve for the proposed CNN network implemented in CNTK and trained on the current dataset described in Section 4.1.

Table 7
Comparison of the accuracy obtained by the approaches using similar datasets and the CNN network proposed in the paper.

Method	Accuracy
SVM [17]	95.80%
CANTINA+ [16]	92.00%
SVM [19]	99.00%
td-idf [38]	99.62%
CNN proposed in the paper	99.98%

6. Conclusion

In the paper, we proposed the method of identifying phishing websites based on only the URL address text by a deep neural network with convolutional layers. To this end, we encoded URLs as one-hot character-level vectors and presented them as inputs to a convolutional neural network. We showed that the URL dictionaryless analysis provides nearly 100% accurate security. Accurate, URL-based detection provides zero-day defence, contrary to blacklist or white-list approaches (unless a website with the legitimate domain is compromised). The results were compared mainly with the LSTM network presented in [9] because we analysed the entire URL text in a similar way. We implemented the LSTM network from [9] and trained it with a newer version of the phishing dataset (PhishTank and Common Crawl) to compare LSTM with the proposed method. We also checked many variants of CNNs against testing error to achieve the best data generalisation. Moreover, we found out that the use of the embedding layer improved the results. The embedding layer transformed the sparse input image into a dense vector of real numbers. The results presented show that the CNN network dealt with the URL classification better than LSTM. In our experiments, we can observe the minimal advantage of CNN over LSTM in terms of accuracy. A significant difference can be seen in the training of both solutions. The CNN network is much easier to train than LSTM. The faster convergence is shown in Fig. 9. Furthermore, the time needed for one training epoch is nearly 50% shorter, which translates into four-fold training acceleration. It is essential in the case the phishing detector is fine-tuned with new data periodically. The trained phishing detector is small and fast and can be used effortlessly on mobile devices. The training lasts about three minutes and the URL checking about two milliseconds. Moreover, newer mobile chipsets have special instructions for machine learning computations.

Declaration of Competing Interest

No conflict of interest.

CRediT authorship contribution statement

Wei Wei: Funding acquisition, Resources. **Qiao Ke:** Funding acquisition, Resources. **Jakub Nowak:** Conceptualization, Data curation, Validation. **Marcin Korytkowski:** Conceptualization, Writing - original draft. **Rafał Scherer:** Conceptualization, Supervision, Writing - original draft. **Marcin Woźniak:** Conceptualization, Supervision, Writing - original draft.

Acknowledgements

This job is supported by the National key R&D Program of China under Grant (NO. 2018YFB0203901). Please add this funding number in the beginning of Acknowledgements parts. This work is supported by the Key Research and Development Program of Shaanxi Province (no. 2018ZDXM-GY-036) and Shaanxi Key Laboratory of Intelligent Processing for Big Energy Data (no.IPBED7) and by the National Natural Science Foundation of China (grant no. 61761042, no. 61941112), Key Research and Development Program of Yanan (grant no. 2017KG-01, 2017WZZ-04-01).

References

[1] S. Opałka, B. Stasiak, D. Szajerman, A. Wojciechowski, Multi-channel convolutional neural networks architecture feeding for effective eeg mental tasks classification, Sensors 18 (10) (2018) 3451.
[2] J. Walczak, T. Poreda, A. Wojciechowski, Effective planar cluster detection in point clouds using histogram-driven kd-like partition and shifted mahalanobis distance based regression, Remote Sens. (Basel) 11 (21) (2019) 2465.
[3] R. Kumarratneshk, E. Weilleweill, F. Aghdasi, P. Sriram, A strong and efficient baseline for vehicle re-identification using deep triplet embedding, J. Artif. Intell. Soft Comput. Res. 10 (1) (2020) 27–45.
[4] E. Volkman, 49 percent of phishing sites now use HTTPS, 2018, https://info.phishlabs.com/blog/49-percent-of-phishing-sites-now-use-https.
[5] T. Moore, B. Edelman, Measuring the perpetrators and funders of typosquatting, in: International Conference on Financial Cryptography and Data Security, Springer, 2010, pp. 175–191.
[6] N. Nikiforakis, S. Van Acker, W. Meert, L. Desmet, F. Piessens, W. Joosen, Bitsquatting: exploiting bit-flips for fun, or profit? in: Proceedings of the 22nd International Conference on World Wide Web, ACM, 2013, pp. 989–998.
[7] E. Gabrilovich, A. Gontmakher, The homograph attack, Commun. ACM 45 (2) (2002) 128.
[8] A. Costello, Punycode: a bootstring encoding of unicode for internationalized domain names in applications (idna), rfc 3492, 2003, http://www.rfc-editor.org/rfc/pdfrfc/rfc3492.txt.pdf.
[9] A.C. Bahnsen, E.C. Bohorquez, S. Villegas, J. Vargas, F.A. González, Classifying phishing urls using recurrent neural networks, in: 2017 APWG Symposium on Electronic Crime Research (eCrime), 2017, pp. 1–8, doi:10.1109/ECRIME.2017.7945048.
[10] ESET North America, How does anti-phishing work in my eset product?, 2019, https://support.eset.com/kb3100/?locale=en_US&viewlocale=en_US.
[11] Microsoft Corporation, Microsoft phishing filter: a new approach to building trust in e-commerce content, 2008, (White Paper).
[12] Mozilla Project, Phishing protection: design documentation, 2019, https://wiki.mozilla.org/Phishing_Protection:_Design_Documentation.
[13] A.K. Jain, B.B. Gupta, A novel approach to protect against phishing attacks at client side using auto-updated white-list, EURASIP J. Inf. Security 2016 (1) (2016) 9.
[14] W. Han, Y. Cao, E. Bertino, J. Yong, Using automated individual white-list to protect web digital identities, Expert Syst. Appl. 39 (15) (2012) 11861–11869, doi:10.1016/j.eswa.2012.02.020.
[15] Y. Zhang, J.I. Hong, L.F. Cranor, Cantina: a content-based approach to detecting phishing web sites, in: Proceedings of the 16th International Conference on World Wide Web, ACM, 2007, pp. 639–648.
[16] G. Xiang, J. Hong, C.P. Rose, L. Cranor, Cantina+: a feature-rich machine learning framework for detecting phishing web sites, ACM Trans. Inf. Syst. Security (TISSEC) 14 (2) (2011) 21.
[17] M. Zouina, B. Outtaj, A novel lightweight url phishing detection system using svm and similarity index, Hum.-centric Comput. Inf. Sci. 7 (1) (2017) 17.
[18] S.C. Jeeva, E.B. Rajsingh, Intelligent phishing url detection using association rule mining, Hum.-centric Comput. Inf. Sci. 6 (1) (2016) 10.
[19] Y. Li, R. Xiao, J. Feng, L. Zhao, A semi-supervised learning approach for detection of phishing webpages, Optik (Stuttg) 124 (23) (2013) 6027–6033, doi:10.1016/j.ijleo.2013.04.078.
[20] O.K. Sahingoz, S.I. Baykal, D. Bulut, Phishing detection from urls by using neural networks, Comput. Sci. Inf. Technol. 8 (17) (2018) 41–54.
[21] Q. Ke, J. Zhang, H. Song, Y. Wan, Big data analytics enabled by feature extraction based on partial independence, Neurocomputing 288 (2018) 3–10.
[22] W. Wei, Y. Qi, Information potential fields navigation in wireless ad-hoc sensor networks, Sensors 11 (5) (2011) 4794–4807.
[23] R.M. Mohammad, F. Thabtah, L. McCluskey, Predicting phishing websites based on self-structuring neural network, Neural Comput. Appl. 25 (2) (2014) 443–458.
[24] F. Feng, Q. Zhou, Z. Shen, X. Yang, L. Han, J. Wang, The application of a novel neural network in the detection of phishing websites, J. Ambient Intell. Humaniz. Comput. (2018) 1–15.
[25] X. Glorot, A. Bordes, Y. Bengio, Deep sparse rectifier neural networks, in: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, 2011, pp. 315–323.
[26] K. Jarrett, K. Kavukcuoglu, M. Ranzato, Y. LeCun, What is the best multi-stage architecture for object recognition? in (iccv'09), IEEE 4 (6) (2009) 7.
[27] K. Fukushima, Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position, Biol. Cybern. 36 (4) (1980) 193–202.
[28] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, et al., Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[29] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.
[30] PhishTank, Phishing website downloadable database, 2019, data.phishtank.com/data/online-valid.csv.
[31] Moz, Inc., Moz's list of the top 500 domains and pages on the web, 2019, https://moz.com/top500.
[32] A. Blum, B. Wardman, T. Solorio, G. Warner, Lexical feature based phishing url detection using online learning, in: Proceedings of the 3rd ACM Workshop on Artificial Intelligence and Security, ACM, 2010, pp. 54–60.
[33] X. Zhang, J. Zhao, Y. LeCun, Character-level convolutional networks for text classification, in: Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1, NIPS'15, MIT Press, Cambridge, MA, USA, 2015, pp. 649–657.
[34] Y. Kim, Convolutional neural networks for sentence classification, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Doha, Qatar, 2014, pp. 1746–1751, doi:10.3115/v1/D14-1181.
[35] L.v. d. Maaten, G. Hinton, Visualizing data using t-sne, J. Mach. Learn. Res. 9 (Nov) (2008) 2579–2605.
[36] A. Komninos, S. Manandhar, Dependency based embeddings for sentence classification tasks, in: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2016, pp. 1490–1500.
[37] L. Bottou, Stochastic Gradient Descent Tricks, vol. volume 7700 of Lecture Notes in Computer Science (LNCS), Springer, neural networks, tricks of the trade, reloaded edition, pp. 430–445.
[38] G. Ramesh, I. Krishnamurthi, K.S.S. Kumar, An efficacious method for detecting phishing webpages through target domain identification, Decis. Support Syst. 61 (2014) 12–22.

Wei Wei is a senior member of IEEE, an associated professor of School of Computer Science and Engineering Xi'an University of Technology, Xi'an 710048, China. He received his Ph.D. and M.S. degrees from Xian Jiaotong University in 2011 and 2005, respectively. He ran many funded research projects as principal investigator and technical members. His research interest is in the area of wireless networks, wireless sensor networks Application, Image Processing, Mobile Computing, Distributed Computing, and Pervasive Computing, Internet of Things, Sensor Data Clouds, etc. He has published around one hundred research papers in international conferences and journals. He is an editorial board member of FGCS, AHSWN, IEICE, KSII, etc. He is a TPC member of many conferences and regular reviewer of IEEE TPDS, TIP, TMC, TWC, and many other Elsevier journals.

Qiao Ke received her Ph.D. degree in Applied Mathematics from Xi'an Jiaotong University, Xi'an, China, in 2019. She is currently a PostDoc in Northwestern Polytechnical University, Xi'an, China.Her research interests are in the areas of deep learning, statistics learning, Bayesian learning, image processing and Internet of Things.

Jakub Nowak received his MSc degree in computer science from the Czestochowa University of Technology, Poland, in 2016. He is currently pursuing a Ph.D. degree in computer science at Czestochowa University of Technology. Her present research interests include machine learning for computer system security.

Marcin Korytkowski received the Ph.D and degree in computer science from Czestochowa University of Technology, Poland, in 2007. Currently he is an associate professor at Czestochowa University of Technology. He has published over 70 technical papers. His present research interests include deep learning architectures and their applications in computer security, databases and image retrieval. He is involved in organization of various scientific events and conferences.

Rafał Scherer received his MSc degree in computer science from the Czestochowa University of Technology, Poland, in 1997 and his PhD in 2002 from the same university. Currently, he is an associate professor at Czestochowa University of Technology. His present research interests include machine learning and neural networks for image processing, computer system security, prediction and classification.

Marcin Woźniak received diplomas in applied mathematics and computational intelligence. He is an Assoc. Professor at Institute of Mathematics of the Silesian University of Technology in Gliwice, Poland. In his scientific career, he was visiting University of Wrzburg, Germany, University of Lund, Sweden and University of Catania, Italy. His main scientific interests are neural networks with their applications together with various aspects of applied computational intelligence. He is a scientific supervisor in editions of “the Diamond Grant” and “The Best of the Best” programs for highly gifted students from the Polish Ministry of Science and Higher Education. Marcin Woźniak served as an editor for various special issues of IEEE ACCESS, Sensors, Frontiers in Human Neursocience, etc., and as an organizer or a session chair at various international conferences and symposiums, including IEEE SSCI, ICAISC, WorldCIST.