Development of Artificial Intelligence Based Module to Industrial Network Protection System

Filip Holik[1], Petr Dolezel[1], Jan Merta[1], and Dominik Stursa[1]

University of Pardubice, Pardubice, Czech Republic,
petr.dolezel@upce.cz,
WWW home page: http://www.upce.cz/fei

Abstract. The paper deals with the software-defined networking concept applied to industrial networks. This innovative concept supports network programmability and dynamic implementation of customized features, including security related ones. In a previous work of the authors, the industrial network protection system (INPS) was designed and implemented. The INPS provides complex security features of various traditional and modern security solutions within a single system. In this paper, the AI module, which is one of the crucial parts of the INPS, is dealt with. In particular, a detailed report focused on the development of the AI module decision function is provided. As a result, an artificial neural network, used for the network traffic evaluation in the AI module, is developed and comprehensively tested.

Keywords: artificial neural network, industrial networks, software defined networks

1 Introduction

The recent transformation of industrial networks from private and closed networks into standardized IP networks brought many advantages, but also introduced new security risks. These networks become connected to cloud data centers and centralized management systems and integrated Internet of Things (IoT) devices. These changes lead to increase of traffic volume, heterogeneity and complexity, resulting in wider scope for potential attacks. This danger is magnified by the fact, that these networks are nowadays connected to the Internet all the time and they can be therefore theoretically accessed by anyone. To perform an attack is nowadays easier than anytime in the past. Attacking tools are now publicly accessible and they require no deep knowledge in order to use them.

Defense against these threats requires a more comprehensive approach than just a manual filtering by a human element. An automation with involvement of artificial intelligence (AI) is nowadays a necessity. There are many commercially available protection tools which use cloud-based artificial intelligence, but on the other hand, not so many, which could be deployed locally. This could be an important fact, if data privacy is an issue.

One of the most promising approaches in this area is utilization of software-defined networking (SDN), which provides a solid prefiguration for AI implementation. Unfortunately, the utilization of SDN in security areas is still being researched. Traditional firewalls were implemented in SDN in many works [3, 6, 18, 19], but AI-based firewalls, on the other hand, were rarely researched. Only the paper [5] presented a firewall with machine learning for securing cloud data centers. However, the operation of this firewall was severely limited, as it supported only two functions: allow and block packets. Such a system does not meet criteria for modern security systems.

Therefore, a protection system with AI was designed and developed by the authors of this article in [11]. The solution was called Industrial Network Protection System (INPS) and its aim was to provide complex security features within a single system. One of the most crucial parts of the system was the AI module, used for network traffic evaluation. The work [11] utilized only the basic design of artificial neural networks and did not explore multiple approaches. The goal of this contribution is to provide a detailed report focused on the development of the AI module decision function. Hence, two approaches to the AI module decision function are defined, implemented and tested in order to get the suitable functionality of the AI module.

The rest of the article is organized as follows. In the next chapter, the idea of an industrial network protection system is summarized. Then, the AI module is proposed. The main contribution of this article, the AI module decision function architecture development, is described within this section. Afterwards, as the last part of the article, the AI module performance is tested and evaluated.

2 Industrial Network Protection System

This section describes only the overall architecture of INPS. The more detailed definition can be found in [11].

The INPS is developed to comply with the main operational requirements of industrial networks, i.e. component lifetime, critical infrastructure, fault tolerance, high availability, limited component access, non-upgradability, performance, proprietary communication protocols and system certification [12, 23]. It provides centralized network management with monitoring of data flows in real-time. Each data flow can be blocked or allowed and the system also supports more advanced filtering features including redirecting the flow to specified ports, setting QoS values, and storing payload for the application layer inspection. All these features can be performed manually or automatically by the AI module.

The architecture of the system can be segmented into four components, as shown in Fig. 1.

1. The main module - it provides the basic INPS functionality including its control via two separate web pages - one for traffic monitoring and filtering, and the second for the system settings. The module is integrated into

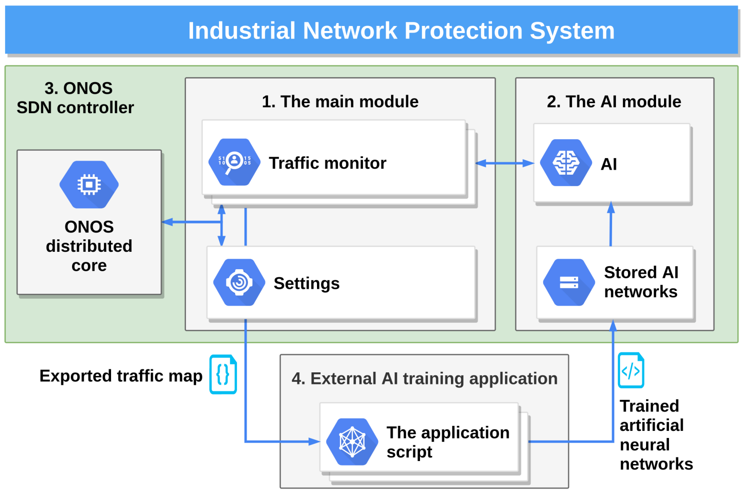

Fig. 1. INPS architecture

the ONOS controller and it interacts with its distributed core directly via customized internal interfaces.

2. The AI module - it performs optional automated filtering. It is located in a separate package, which uses files with trained artificial neural networks for determining traffic decisions. These files can be imported directly or via the application web interface.
3. The ONOS SDN controller - it performs standard networking functions. Based on the network topology, this can include forwarding, routing, loop prevention, load-balancing etc. The controller also has the web user interface for configuration and management of provided features.
4. External AI training application - it is implemented as a stand-alone application. It takes an exported traffic map - a file with data flows and manually set firewall rules. Based on this map, it trains artificial neural networks by one-time offline learning. The output of the application is a file with trained neural networks. This file can then be imported to the AI module.

In the next section, a suitable architecture for the AI module decision function is discussed and developed.

3 Artificial intelligence in INPS

The AI in the INPS has functionality of a decision method. In simple words, this element determines one of the decision states according to incoming flow characteristics. The state space includes the following items: allow, block, forward to selected ports, application layer inspection, and four levels of QoS settings (low, normal, high and critical).

Communication protocols used in industrial networks can be classified into two basic types - network layer protocols (L3) and transport layer protocols (L4) [1]. In order to keep the clarity and transparency of the article at the acceptable level, the more general L4 communication is considered in this work. However, the more granular functionality, which includes other possible protocols, can be achieved by the INPS by performing similar steps.

Hence, the AI-based decision method is supposed to decide, based on the incoming flow characteristics defined by both source and destination IP addresses and also by the amount of traffic expressed by packets per second. Each IP address is composed from four octets, which are used as unique inputs. In addition, source and destination port numbers are considered as relevant inputs. Therefore, eleven inputs specify the decision. It is shown to advantageously solve such input-output mapping problems using feedforward multilayer neural networks [13, 7].

An artificial neural network is a group of algorithms that, generally, aims to recognize relationships in a set of data through a process that emulates the way the biological neural network operates. A feedforward multilayer neural network is one of the mostly applied architectures [10]. It consists of two or more subsequently connected layers of artificial neurons, with signal propagated only in forward direction.

During last decades, two types of feedforward multilayer neural networks have proven themselves to be particularly competent for input-output mapping problems. The first of them is a feedforward neural network with dense (fully connected) layers (FFNN), the second one is a feedforward convolutional neural network (CNN). Both mentioned types are considered in this approach as possible architectures for the AI module decision function.

The procedure of a neural network design involves training set, and validation set acquisition, training, pruning and validating. The essential information related to this procedure, as it is adapted for the INPS, is described in the following sections. More information regarding the design, as well as the discussion to each part of the process, can be found in [10].

3.1 Traning dataset

The used dataset is simulating a highly utilized industrial network and the traffic was generated by a custom developed application [11]. The traffic map, generated for a neural network training, contains 80 000 unique data patterns. The dataset is then divided into training set (70 %), validation set (15 %) and testing set. Training set is used for a neural network parameters adaptation during training process, validation set is used to identify the best network configuration during training, and testing set is used for final AI module evaluation.

3.2 FFNN design

In order to have a capability to solve the problem, the FFNN needs to follow some statements [10]. Specifically, at least one hidden layer with enough neurons needs to be implemented. Besides, monotonic, continuous and bounded transfer (activation) functions must be applied in neurons of the hidden layer. In this contribution, a number of hidden layers is set based on experimental results. A hyperbolic tangent transfer function is considered for the neurons in the hidden layers. Since the FFNN is intended to be used as a decision element, the softmax transfer function is considered for the neurons in the output layer. Apparently, the number of output neurons is defined by the number of elements in the output state space.

FFNN design especially consists of training and pruning. The result of training provides suitable weights and biases of FFNN. The pruning is superior to the training and it should convert the redundantly-designed network into a simpler one with no decrease of the performance. In our work, the pruning is based on the repeated training of various FFNNs with different topologies. A mean square error function (E), applied to the validation set, works for us as a cost function. It detects the performance quality of the network. Since the training is a stochastic process, 100 training performances for every considered topology are executed in order to get statistically significant results.

The training parameters are set according to the pilot study and based on the previous authors' experience. Specifically, the Nguyen-Widrow technique [21] is used for the initial setting of the weights and biases in the beginning of the training. Then, the Levenberg-Marquardt search technique [9] is applied for the weights and biases adaptation. The input values in the training set are normalized in order to avoid the unequal influence of individual values during the training process. The parameters of the training and pruning process, including the formal parameters of all the applied techniques, are summarized in Table 1.

Table 1. Parameters of the experiments with FFNN

Training algorithm	Levenberg-Marquardt search technique
Initialization	Nguyen-Widrow technique
Maximum epochs	500
Stopping criterion	Maximum epochs reached
Adaptive coefficient μ	0.001
Increment μ	10
Decrement μ	0.1

Box plots with the resulting cost function value obtained for various FFNN topologies during training process are shown in Fig. 2. Each box plot consists of median value (central mark), 25^{th} and 75^{th} percentiles (edges of the boxes) and extreme data points (the whiskers). It is clearly visible, that the most suitable performance is provided using a topology with ten neurons in one hidden layer. The topologies with two hidden layers fail to provide better results. However, best representatives of all topologies will be also tested using testing set.

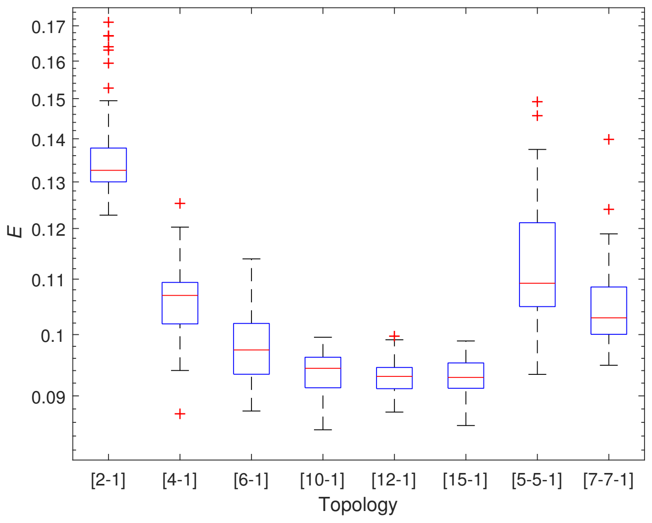

Fig. 2. Resulting values of error function for various FFNN topologies.

3.3 CNN design

With current possibilities in parallel computing, CNNs are considered a leading topology among neural networks. A list of well-recognized CNN topologies can be found in [2].

Convolutional neural network layers mainly include three representatives, namely convolutional layer, pooling layer and dense (fully-connected) layer. A good summary of convolutional neural networks and how to implement them can be found in [8].

In these days, a huge number of various topologies of CNNs are available for implementation. In this paper, five different architectures are selected for possible application. Architectures Net1 and Net2 are relatively simple. They consist of the sequence of convolutional layers and max-pooling layers. Both architectures end with a last hidden dense layer with 512 neurons. Then, a softmax layer is applied as the output layer to classify the output. Both architectures are adapted from [20]. In addition to these networks, more complex and widely accepted topologies are selected; LeNet-5 [4, 17], AlexNet [16] and VGG-16 net [22].

As well as in the previous case, the mentioned architectures are trained in order to map correctly the dataset described in section 3.1. However, it is generally accepted feature of CNNs, that the performance is especially high when applied to multidimensional data processing. Image processing can be stressed as one of the most obvious examples [15]. Hence, it could be useful to find an operation of transformation, which transforms eleven inputs, considered as inputs to the AI module decision function, into two or three-dimensional structure, preferably a graphical figure. As the first engineering approach to this transformation, a polar line chart is suggested in this article, as demonstrated in Fig. 3. The operation is referred to in the further text as depiction.

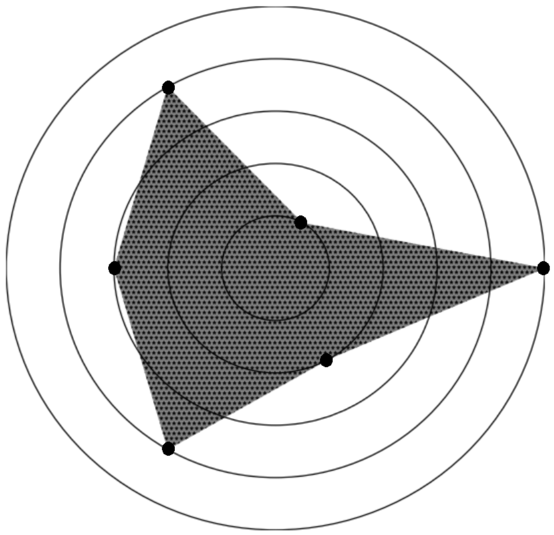

Fig. 3. Demonstration of visualization of multidimensional data in 2D. In this demonstration, a 6-dimensional vector [1, 0.2, 0.8, 0.6, 0.8 , 0.4] is visualized.

Therefore, the whole dataset (see Section 3.1) is normalized and transformed to a set of filled polar line chart figures. The charts are stored as [122 x 122] px grayscale images. A selected group of resulting images is demonstrated in Fig. 4.

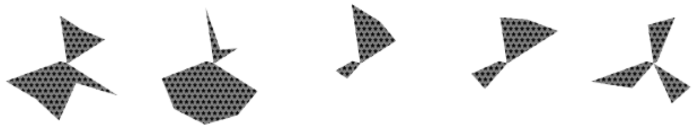

Fig. 4. Examples of transformed dataset.

Consequently, the training of the selected architectures is performed. The ADAM search technique is used as an optimizer based on its generally acceptable performance [14]. Initial weights are set randomly in this case, with Gaussian distribution (location = 0, scale = 0.05).Similarly to the previous training, the training processes are executed a hundred times and the same cost function is computed over the validation set - see Table 2 for all the parameters of the training processes. The resulting values are shown in Fig. 5. After the training process, LeNet-5 is indicated to be the correct CNN to be implemented for the AI module decision function. However, the best representatives of each architecture are tested in the next section.

Table 2. Parameters of the experiments with CNN

Input shape	122 x 122 x 1
Training algorithm	ADAM algorithm
Initialization	Normal distribution (mean = 0, std = 0.05)
Maximum epochs	50
Stopping criterion	Maximum epochs reached
Learning rate α	0.001
Exponential decay rate 1 β_1	0.9
Exponential decay rate 2 β_2	0.999

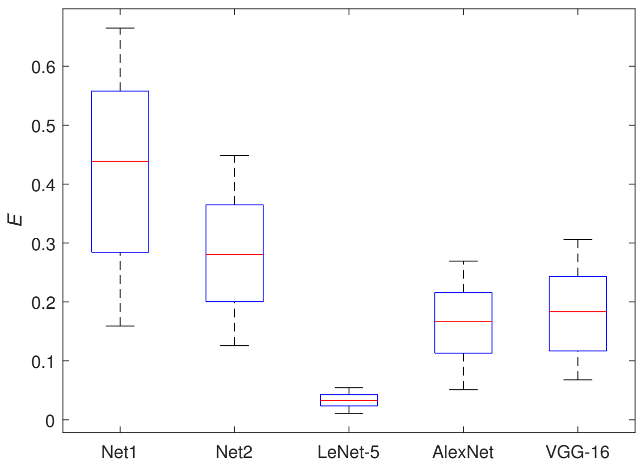

Fig. 5. Resulting values of error function for various CNN topologies.

4 FFNN and CNN testing and evaluation

In the previous sections, two architectures of feedforward multilayer artificial neural networks are designed to be implemented as a decision method in the AI module. Both are drawn up in Fig. 6.

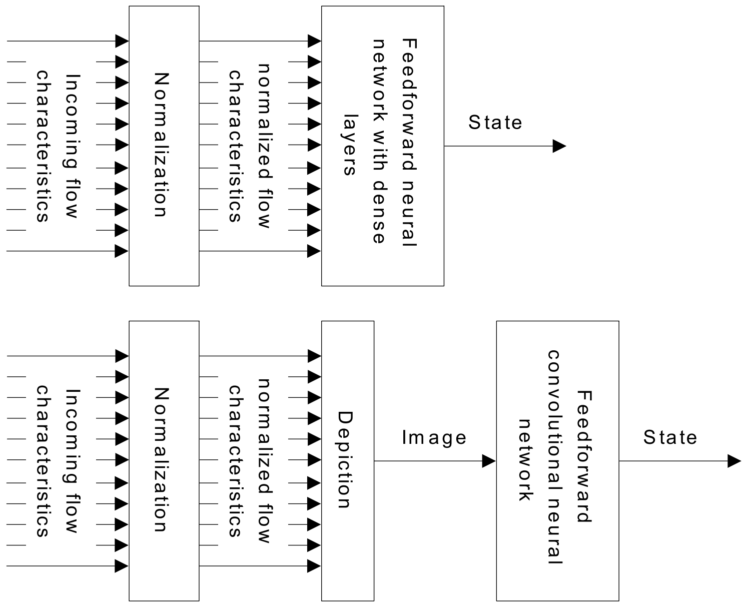
Incoming flow characteristics
Normalization
normalized flow characteristics
Feedforward neural network with dense layers
State
Incoming flow characteristics
Normalization
normalized flow characteristics
Depiction
Image
Feedforward convolutional neural network
State

Fig. 6. Considered architectures of AI module.

Now, the best representatives of both architectures are tested using the testing set (see Section 3.1). Note that the data in the testing set are not used during training. Accuracy, defined as the ratio of correctly made decisions to all performed decisions, is used as the metric. The resulting values of the metric are shown in Table 3.

The results show a number of interesting outcomes. Above all, the highest accuracy is provided by the VGG-16 architecture in combination with depiction of inputs. Thus, a best value of error function during training does not guarantee a best accuracy over the testing set. Then, feedforward neural networks with dense layers provide generally less variant results. Convolutional architectures, on the other hand, go from totally unacceptable to a very reasonable behavior.

5 Conclusion

As a continuation of previous work of the authors, the development of the AI module, which is a part of the industrial network protection system, is dealt with in this article. Two architectures, based on feedforward multilayer neural networks, are considered for implementation as decision function for the AI module. The extensive development of both architectures is performed then in order to achieve a high accuracy of decision making of the AI module. The tests presented at the end of the paper indicate, that the highest accuracy is provided by the VGG-16 architecture in combination with depiction of inputs.

Table 3. Testing results

Topology	**Accuracy**
2-1	0.7920
4-1	0.9088
6-1	0.9126
10-1	0.9188
12-1	0.8918
15-1	0.9097
5-5-1	0.9084
7-7-1	0.8664
Net1	0.8301
Net2	0.8752
LeNet	0.8346
AlexNet	0.7262
VGG-16	0.9501

However, this outcome should definitely be understood as the preliminary result, since many aspects of the development procedure still need to be examined. First of all, there exist many possibilities of depiction process. In this paper, only one is considered. It is indispensable possibility of totally different results with different depiction process. The other thing is a computational complexity. The protection system is supposed to provide efficient real-time traffic monitoring and depiction process could be one of the weak spots from this point of view. Hence, these aspects are aimed to be deal with in the future works.

References

1. IEC 61850-5: Communication networks and systems in substation (2003)
2. Aloysius, N., Geetha, M.: A review on deep convolutional neural networks. In: 2017 International Conference on Communication and Signal Processing (ICCSP). pp. 0588–0592 (April 2017)
3. Bakhareva, N., Polezhaev, P., Ushakov, Y., Shukhman, A.: SDN-based firewall implementation for large corporate networks (2019)
4. Bottou, L., Cortes, C., Denker, J.S., Drucker, H., Guyon, I., Jackel, L.D., LeCun, Y., Muller, U.A., Sackinger, E., Simard, P., Vapnik, V.: Comparison of classifier methods: a case study in handwritten digit recognition. In: Proceedings of the 12th IAPR International Conference on Pattern Recognition, Vol. 3 - Conference C: Signal Processing (Cat. No.94CH3440-5). vol. 2, pp. 77–82 vol.2 (Oct 1994)
5. Cheng, Q., Wu, C., Zhou, H., Zhang, Y., Wang, R., Ruan, W.: Guarding the perimeter of cloud-based enterprise networks: An intelligent SDN firewall. In: 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS). pp. 897–902 (June 2018)
6. Fiessler, A., Lorenz, C., Hager, S., Scheuermann, B.: Fireflow - high performance hybrid SDN-firewalls with OpenFlow. vol. 2018-October, pp. 267–270 (2019)

7. Gencay, R., Liu, T.: Nonlinear modelling and prediction with feedforward and recurrent networks. PHYSICA D 108(1-2), 119–134 (SEP 15 1997)
8. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016), http://www.deeplearningbook.org
9. Hagan, M., Menhaj, M.: Training feedforward networks with the Marquardt algorithm. IEEE Transactions on Neural Networks 5(6), 989–993 (Nov 1994)
10. Haykin, S.: Neural Networks: A Comprehensive Foundation. Prentice Hall (1999)
11. Holik, F., Dolezel, P.: Industrial network protection by SDN-based IPS with AI. Communications in Computer and Information Science (2020, In press)
12. Holik, F.: Meeting smart city latency demands with SDN. Studies in Computational Intelligence pp. 43–54 (2020)
13. Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural Networks 2(5), 359–366 (1989)
14. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR abs/1412.6980 (2014), http://arxiv.org/abs/1412.6980
15. Kizuna, H., Sato, H.: The entering and exiting management system by person specification using deep-cnn. In: 2017 Fifth International Symposium on Computing and Networking (CANDAR). pp. 542–545 (Nov 2017)
16. Krizhevsky, A., Sutskever, I., Hinton, G.: Imagenet classification with deep convolutional neural networks. vol. 2, pp. 1097–1105 (2012)
17. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11), 2278–2324 (Nov 1998)
18. Li, H., Wei, F., Hu, H.: Enabling dynamic network access control with anomaly-based IDS and SDN. pp. 13–16 (2019)
19. Mahamat Charfadine, S., Flauzac, O., Nolot, F., Rabat, C., Gonzalez, C.: Secure exchanges activity in function of event detection with the SDN. Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST 275, 315–324 (2019)
20. Millstein, F.: Deep Learning with Keras. CreateSpace Independent Publishing Platform (2018)
21. Nguyen, D., Widrow, B.: Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights. pp. 21–26 (1990)
22. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv 1409.1556 (09 2014)
23. Stouffer, K.A., Falco, J.A., Scarfone, K.A.: Sp 800-82. guide to industrial control systems (ICS) security: Supervisory control and data acquisition (SCADA) systems, distributed control systems (DCS), and other control system configurations such as programmable logic controllers (PLC). Tech. rep., Gaithersburg, MD, United States (2011)