2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovations

Accelerating a Secure Programmable Edge Network System for Smart Classroom

Watipatsa W. Nsunza, A-Q. Ransford Tetteh and Xiaojun Hei
School of Electronic Information and Communications
Huazhong University of Science and Technology, Wuhan, China, 430074
Email: {nsunza, ranstaq, heixj}@hust.edu.cn

***Abstract*—Internet-of-Things (IoT) applications have been challenged by vast emerging security threats. The increasing number of connected devices on an IoT network may encounter potential attacks while relaying messages over the public Internet. Security methods in traditional wireless networks are often specific to single architectures and are therefore ineffective in solving security issues that may surface on hybrid IoT networks. In this paper, we propose a software defined end-edge-cloud network architecture for smart IoT applications. We apply a deep-learning (DL) based intrusion detection system (IDS) to secure this architecture. We implement this system based on the Caffe framework and train several popular convolutional neural networks (CNNs) including LeNet, AlexNet, and ResNet-50. We evaluate the performance of this system in terms of accuracy, precision, recall, and F_1 score. We then investigate the FPGA-based acceleration technique to reduce the training and runtime of CNNs. We implement the FPGA acceleration based on a Xilinx KU115 board which achieves efficient performance per watt while training and deploying the IDS. For accelerating CNNs on the FPGA, we investigate a Winograd convolution engine in the Xilinx SDAccel development environment which offers automation schemes for accelerating computations. Our preliminary study may provide some insights into the experimental support for advancing IoT research and development for securing various smart applications.**

***Index Terms*—Intrusion Detection, End-edge-cloud Network Architecture, Software Defined Networking, Smart Classroom**

I. Introduction

Security is one of the major aspects on Internet devices since security measures are only realized after launching a new technology. The physical objects in IoT are more sensitive to security, and contain embedded systems to communicate between machine-to-machine and machine-to-people. IoT poses potential risks to the traditional wireless network devices such as advanced encryption standard (AES) public/private key exchange methods, unprotected transmission control protocol (TCP)/Internet protocol (IP) networks from intrusion through devices, and unprotected pre-shared keys from reverse engineering through a micro-controller unit (MCU) debugger [1]. Software defined networking (SDN) offers various benefits to IoT; however, several studies have revealed security holes in SDNs which may result in security threats such as:

- Selfish attacks which generate more flows and consume larger bandwidth.
- Flow-table overloading which mount DDoS attacks.
- An SDN switch can be compromised and behave like robot.
- An SDN network being unable to mitigate and detect advanced persistent threats.
- Match-action algorithm failures to reveal matching entries for invoking deep packet inspection.

Security risks in software defined networks (SDN) are largely due to lacking of integration of existing security mechanisms and SDN's inability to provide deep packet inspection. This leads to demands for external mechanisms to introduce packet risk analysis before routing to the next levels of wireless networking. SDN security requires support for authentication and authorization classes of the network administrators at every plane, but the proceeding results may prevent access to flow management policies. This also demands constructing novel security mechanisms specific to the SDN protocol. Although SDN security is still at an early stage for integration into IoT, we envision that comprehensive studies will be necessary, which brings forth the motivation of our study on deep-learning based security methods.

A network intrusion detection system analyzes and gathers information on multiple levels of a computer or network and identifies security breaches including "intrusions", attacks outside the network area and "misuse", attacks from within the network. A vulnerability assessment is normally conducted to examine the security. Data have been considered the most important aspect to protect in organizations [2] and operations are only carried out once the data are secured. Data are however under constant threats from malicious attacks as hackers and crackers develop new ways to breach organizational networks. SDN offers control knobs for fast reaction to security threats, granular traffic filtering, dynamic security policy deployment, and flexible traffic management. In this paper, we propose a software defined end-edge-cloud network architecture for smart IoT applications. In particular, we apply a deep-learning (DL) based intrusion detection system (IDS) to secure this architecture and conduct a performance evaluation of this proposed IDS system.

The organization of this paper is as follows: We first propose a software defined end-edge-cloud architecture for the emerging smart applications in Section II. In Section III, we study security mechanisms for the proposed architecture. In Section IV, we conduct a performance evaluation study for the proposed system. Section V discusses some recent research progress in security challenges for software defined IoT networking. Section VI concludes this paper.

II. A Software Defined End-Edge-Cloud Architecture

A. Smart Classroom

The prototype system of the software defined end-edge-cloud network architecture [3] can be implemented by integrating several open-source projects including OpenFlow, OpenNetVM, Floodlight, Odin, and Caffe. The smart classroom application is a typical scenario for this architecture. An artificial-intelligence-enabled smart classroom may contain a variety of end devices with sensors, which places new demands on computation at the edge due to the tight delay constraint. The devices may stream a large amount of input data and are ideal targets for machine learning applications. Smart classrooms enhance learning among students and communication between students and teachers. This requires real-time sensing and machine intelligence using advanced algorithms and optimized software frameworks. In our edge system design for smart classroom, we have integrated OpenVSwitch for managing the inner network using the OpenFlow protocol and we have integrated the Caffe deep learning framework into the design for machine learning support on the smart edge system. The computation units are virtualized to be accessible by the devices on the network. The guiding system may provide various learning services.

B. DL-based IDS Design

A model representation of our deep learning system is illustrated in Fig. 1. We consider a semi-supervised deep learning system with multiple hidden layers where each layer computes a non-linear evolution of the previous layer. When a new packet arrives at the switches, the packet headers are extracted from OpenFlow (OF) packets by an SDN controller which forwards them to the deep learning IDS for analysis. The proposed deep learning FPGA framework can also analyze the network performance when packets are classified at the controller to specific ports on a switch and then generate a weight matrix to assist the controller in determining optimal destination routes.

C. Framework Design

SDN security risks are largely due to its inability to provide deep packet inspection, attacks can occur at either the data plane or the control plane. An optimal solution can also be realized through training the algorithm to monitor multiple vectors on packet_in messages arriving at the controller as a result of flow-misses from unknown sources which will always forward all subsequent packets to the controller for computations. If packets arrive from a classified malicious user, the system may automatically block all subsequent requests and quarantine the host. To avoid false-alarms in the algorithm, our IDS depends on every packet to classify attacks and for evolving the system. Once an attack is classified, IDS may alert the controller.

Our deep learning algorithm intelligently collects network statistics on all activities between two node pairs; source and destination. Whenever a new device joins the network or sends a flow to the system for the first time, the controller starts building a trust table between nodes on the internal and external network deeply analyzing the activity of nodes on different layers of the network based on a number of parameters such as the frequency of message exchanges and interactions, this will allow us to train a deep learning system with sufficient data to determine the purpose of each packet and assist the controller with efficient, and secure operations as shown in Fig. 1.

III. An FPGA-based IDS

A. Overview

We have used the KU115 Xilinx® Kintex® Ultrascale FPGA platform to implement the Caffeinated FPGAs framework. We have integrated this framework into Caffe in a similar fashion to the GPP and GPU brews based on the official Caffe release. The Cuda back-end and OpenCL have also been implemented to support the FPGA execution. The OpenCL backend synchronizes the memory between the host processor and FPGA through the host codes. The device codes also run on the FPGA platform to enable the deep learning implementation. Individual layers for accelerating the AlexNet CNNs are created using the OpenCL device kernel. The Caffeinated FPGAs framework enables the acceleration by creating "forward_fpga" propagation functions in Caffe similar to the existing "forward_gpu" and "forward_cpu" functions. This results into three sets of the AlexNet layer implementation in Caffe that enables three brew options at runtime:

- C++ based brew for GPP execution;
- CUDA brew for GPU execution;
- OpenCL brew for FPGA execution.

The brew options can easily be selected by users at runtime by specifying the desired platform with the following arguments for "-gpu=1" for using GPUs, "-ocl=1" for FPGAs, and "-cpu=1" for CPU runs. The numeric value index specifies the device number when multiple options for one architecture are available.

B. Winograd Convolution

Our framework uses a Winograd Convolution Engine to perform column-wise and row-wise partial transforms. We have integrated this framework on a KU115 FPGA platform. We followed the Xilinx SDAccel OpenCL FPGA programming model to increase the number of computational elements for correlative distribution of computations while exploiting a combination of parallel computing (i.e. data, model, pipeline, and computation) to improve the system performance. Following the distributed deep neural network (DDNN) architecture, our trained DCNN is distributed locally at the edge across the FPGA and GPP, and onto a cloud server with GPU and GPP resources.

$$X = A^T dB \tag{1}$$

A Wingograd Transpose equation used in the Caffeinated FPGAs framework is shown in Equation (1) showing the input tile transformation of the filter transformation. X is an $(m +$

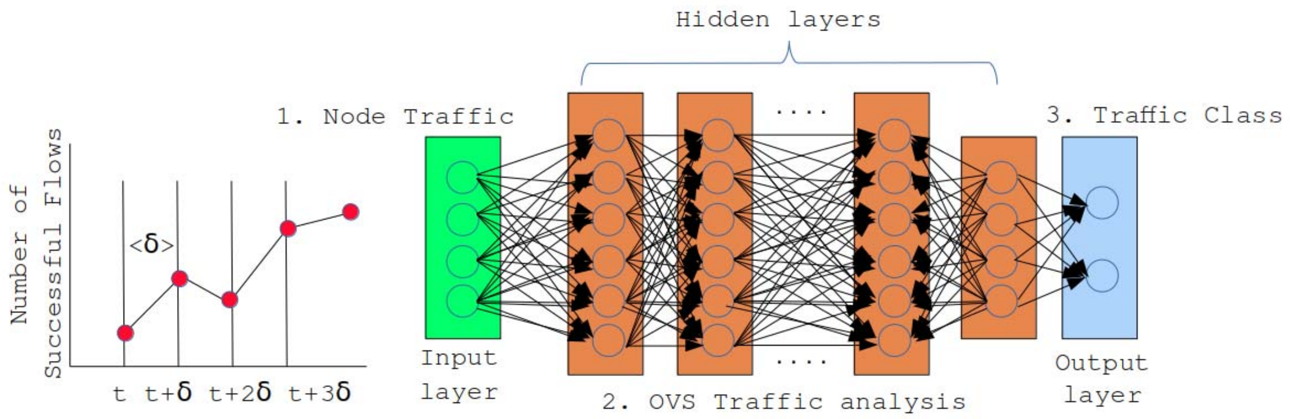

Fig. 1. The IDS architecture for software defined wireless networks

$r-1) \times (m+r-1)$ transformed data matrix; d is an $(m+r-1) \times (m+r-1)$ input matrix; B is an $(m+r-1) \times (m+r-1)$ transform matrix.

For output $Y = (3\times3, 5\times5)$ input tile, d is 7×7 generated by a sliding window. After the transformation, the tile is stored back into the memory. Four times more memory is required to handle the overlaps in the tiles.

C. Data Preparation

In this work, we propose a data preparation method for CSV-type data and generate a script to normalize the data and perform the conversion from CSV to image for the CNN computation. The CSV type data can be logged from multiple input devices sources from networks such as AI-enabled smart classroom components and other input sensors. For each type of IoT network, compatible executables can be developed for specific applications to log the data into specified formats as a step before the image conversion. As a case study, we implement a real-time network intrusion detection system. We developed a feature extractor to log several parameters on IP packet headers. We then performed several training sessions on popular CNNs including LeNet, AlexNet, GoogleNet, and ResNet for intrusion detection. Fig. 2 and Fig. 3 provide a detailed illustration of the data preparation process and the resulting image samples.

D. Normalization Function

We proposed a normalization function to scale CSV type data between 0 and 255 to reflect gray scale image pixels. The NSL-KDD dataset [4] contains symbolic and continuous features. For symbolic features, we have simply represented the data by numeric values. For example, the protocol field has three values (tcp, udp, icmp), we first represent these features by numeric values $(1, 2, 3)$ respectively. Once all the symbolic data has been numerically represented in the script, we then use the normalization function to normalize a column vector (x) in a CSV file between 0 and 255. The new column vectors (x') are then used to generate grey-scale image pixels using the OpenCV library. Algorithm 1 illustrates the functions of our data conversion script.

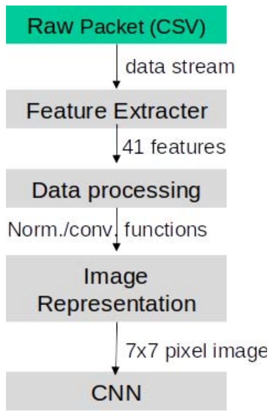

Fig. 2. Data Preparation

Algorithm 2 depicts the steps involved in the functionality of our NIDS. OpenFlow messages trigger different events in Floodlight. The Packet_In handlers perform most of the actions compared to other service modules. When a packet-in message is dispatched from a switch, the controller is responsible for handling the packet to install the necessary flow table rules using the FlowMod message. The core module in this project implements a Packet_In listener to manage all Packet_In messages to be accessed by other service modules through the REST API in Floodlight.

IV. Performance Evaluation

We conducted the experiments on a Dell Optiplex 7040 workstation. Caffe was implemented as a deep learning framework. We performed several training sessions on popular CNNs including LeNet, AlexNet, and ResNet for intrusion detection on the NSL-KDD dataset. We used 100 epochs on a 256 batch size to train our CNN models. We used "Adam" and other Caffe solver with a gradient decent optimizer.

Algorithm 1: Convert CSV to PNG

```
Input: new packet in CSV
Output: normalize the packet data and convert to an image
begin
packet in list = 0
    while $packet < 1000$ do
        Receive incoming packets from the switch
        //Convert symbolic type data
        Switch convert symbol begin
            Case $x : 1$
            Case $y : 2$
            Case $z : 3$
            break;
        end
        for $Column = x$ do
            $x_{min} = x$
            $x_{max} = x$
        end
        if $x < x_{min}$ then
            $x_{min} = x;$
            else $x > x_{max}$
                $x_{max} = x;$
            end
        end
    end
```

Algorithm 2: Packet In Handler Module

```
Input: new packets at the network controller
Output: Extracted packet headers (TCP, UDP, ICMP)
begin
$packetin\_list \leftarrow 0$
$trust\_tbl \leftarrow 0$
while PIR Module buffer is not full do
        Receive incoming packets from the switch
        Store packet headers in packeti_n list
        if the packet is a flow-table miss then
            Compute the packet source pkt src
            Compute the trust table trust_tbl
            if pkt src trust_tbl then
                Install a flow rule in the switch
                else flow trust_tbl
                    Add a flow in trust_tbl
                    Output the packet to the outport
                end
            else
                Output the packet to the outport
            end
        else
            Quarantine the node for inspection
            Trigger a security breach
        end
    end
```

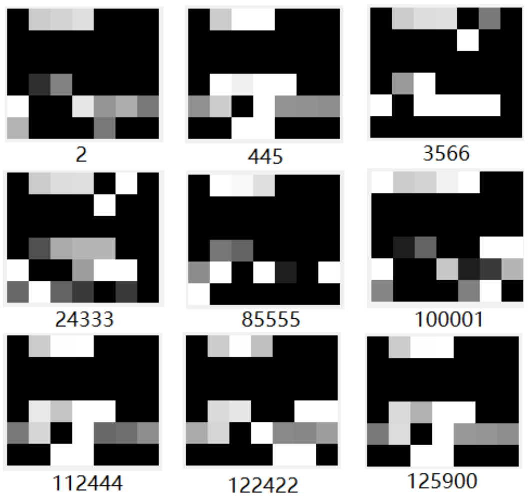

Fig. 3. The NSL KDD conversion procedure

A. Security Performance Metrics

For our NIDS system, accuracy was the major metric for comparing the security performance with other methods. Other considerable indicators include precision, recall, and F_1 score. Preceding our experiments on the IDS, we developed a log function to achieve the column normalization after obtaining a minimum and maximum of every column. We tested the efficiency of this function in retaining the data integrity during the conversion with the minimum complexity. This script converts the $148,517$ samples of training and test data in less “20s” in our system.

B. CNN Model Training

We completed the initial training and testing on a LeNet Model in Caffe using the NSL-KDD image dataset which we have pre-processed. The full model Test set “KDDTest+” with all the old and new attacks achieves an accuracy of 75.45% which is average when compared to other Deep Learning models. Next, we trained the AlexNet model for deep learning in CPU_only mode. This is a middle-scale DCNN model. The model training lasted for 4 weeks on our CPU architecture running a full $360,000$ iterations using “step” learning rate policy. The accuracy was 76.67% on the “KDDTest+” Test set. We also completed the initial ResNet50 training 4. The performance fluctuated for every instance using the default learning algorithm. We paused the training at around 12000 iterations and experimented with a multi-step learning algorithm. The performance improved and the accuracy stabilized around 82.83% which is very good when compared with the related works.

C. Comparative Analysis

We used two test datasets to evaluate the CNN model performance. Table I shows the details of the training and the test datasets. The Test+ dataset contains all the data misclassified by the 21 learners. The accuracy of our CNNs

TABLE I
Performance evaluation of the binary labeled classes

CNN Model Test	Accuracy	Precision	Recall	F_1 Score
LeNet NSL-KDD Test +	79.24%	82.99%	68.55%	90.36%
AlexNet NSL-KDD Test +	79.54%	82.81%	62.42%	79.27%
ResNet 50 NSL-KDD Test +	82.83%	92.99%	100%	92.52%
ResNet 50 NSL-KDD Test + [5]	81.84%	81.84%	100%	90.01%

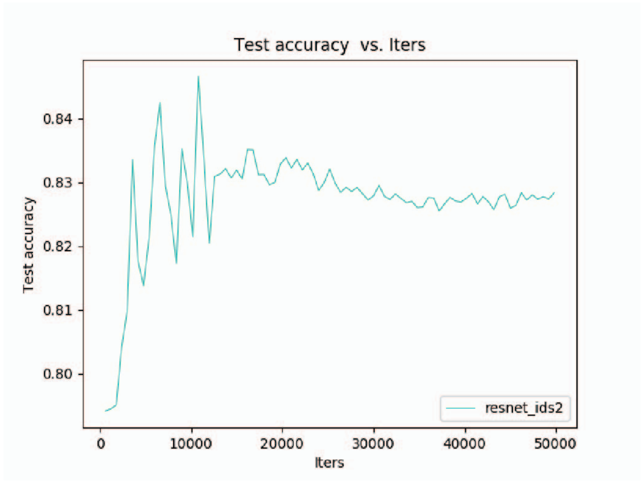

Fig. 4. ResNet optimization test

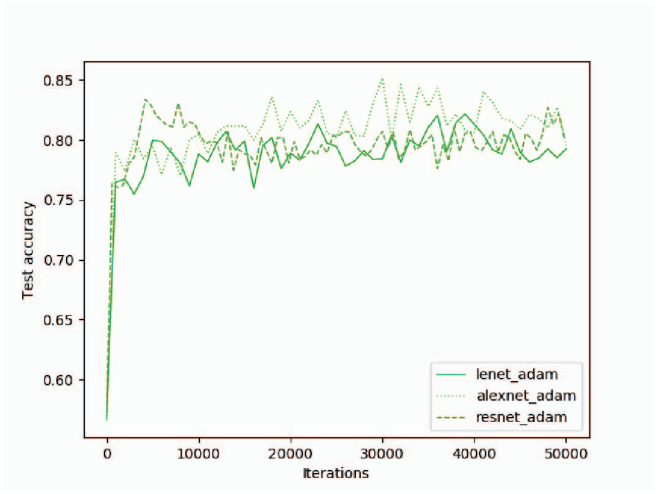

Fig. 5. The NIDS Accuracy on the Adam Solver

(LeNet and AlexNet) has shown that the average performance between $77 - 80\%$. Our ResNet50 DCNN training results, however, have demonstrated between $80 - 83\%$ accuracy on a binary classification. Table II shows the accuracy of our models.

TABLE II
Accuracy comparison of different methods

Classifier	Test+	Test-21
J48	81.05%	63.97%
Naive Bayes	76.56%	55.77%
NB Tree	82.02%	66.16%
Random forest	80.67%	62.26%
Random tree	81.59%	58.51%
Multi-layer Perception	77.41%	57.34%
SVM	69.52%	42.29%
ResNet50 [5]	79.14%	81.57%
GoogLeNet [5]	77.04%	81.84%
Proposed LeNet	79.24%	80.88%
Proposed AlexNet	79.54%	81.42%
Proposed ResNet	80.15%	82.17%

In this experiment, we have analyzed as mentioned in other works that the uneven distribution of the Test-21 data, CNNs tend to data into attack data classes. The percentage of anomaly traffic is small in practice. The IDS seems to recognize attacks in different scenarios. The performance as shown in Table I and Fig. 5 illustrates that our proposed method achieves very good accuracy when compared with other methods. Tavallaee et al [6] have measured the NSL-KDD accuracy using other methods.

V. Related Work

A. Security Advances for Software Defined Wireless Networks

Wireless local area networks have becoming more dense and require more infrastructure coordination [7]. Software defined networking has been proposed to introduce programmability to wireless networks [8], [9], [10], [11]. Kloti et al. in [12] describe many DoS attacks on the OpenFlow protocol. In ProtoGENI [13], a network testbed has been examined to analyze a number of attacks. Many techniques have proposed for detecting various threats for SDN. From the number of investigations taken, the control and data layers are major points of attacks. Flauzac et al. [14] have proposed a multiple SDN controller architecture for ad-hoc networks based on the assumption that equal interaction is default on an SDN controller, which grants it full access to the switch and ensures all controllers have the same rules.

B. Intrusion Detection

Using machine learning techniques, intrusion detection systems can make quite accurate predictions. The most famous intrusion dataset is the KDD Cup-99 [4] which was produced to distinguish between attacks and normal connections. The intrusions were simulated on a military network environment in 1999. The dataset has been improved over the years and the latest data is under NSL-KDD. In [2] Roy et al. have developed a multi-layer feed forward network with 400 hidden layer output and input neurons that provide representations in deep learning for an IDS using rectifier and softmax activation functions. In [15] Gao et al. implemented a NIDS on the KDD-Cup 99 dataset using a RBM based DBN with a neural network as a classifier. Kang et al. [16] proposed a NIDS for in-vehicular network security which used DBN and results in an improved detection accuracy compared to former methods. In [17], Javaid et al. implemented a deep learning based NIDS using NSL-KDD dataset by employing a self-taught learning algorithm using a sparse auto-encoder instead of RBM for feature reduction and also separated the evaluation for the training and testing datasets.

C. *FPGA-based Deep Learning*

Many implementations of deep learning on FPGAs have focused on image recognition. Design size has been one of the most limiting factors for deep learning techniques being implemented on FPGAs due to the trade-off between design reconfiguration and density which makes the FPGA circuits less dense than hardware alternatives and the implementation of large neural networks impossible [18]. The best notable performance for forward propagation of CNNs on FPGAs was achieved by a Microsoft team. Krizhevsky et al. in [19] reported a throughput of 134 images/second at a $25W$ energy consumption rate on the ImageNet $1K$ dataset, a throughput $3\times$ that of the results reported by Zhang et al. [20] using a "Xilinx Virtex 7 485T". The performance has been reported to increase by using other top-line FPGAs such as the "Arria 10 GX1150" while consuming the same power up to 233 images/second when compared to other high-performing GPU implementations of Caffe + cuDNN which perform at $500-824$ images/second while consuming $235W$ of power. This is achieved on Microsoft designed FPGAs for data-center applications [21].

VI. Conclusion

In this paper, we propose a software defined end-edge-cloud network architecture for smart IoT applications. We apply a deep-learning (DL) based intrusion detection system (IDS) to secure this architecture by implementing this system based on the Caffe framework and training several popular Convolutional Neural Networks (CNNs) including LeNet, AlexNet, and ResNet-50 to examine the performance issues. We also investigate the FPGA-based acceleration technique to reduce the training and runtime of CNNs based on a Xilinx KU115 board and the Xilinx SDAccel development environment. Our preliminary results demonstrate the technical feasibility of the software tool chain and the hardware platform for accelerating secure programmable edge network system for emerging smart applications.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (No. 61370231), and in part by the Fundamental Research Funds for the Central Universities (No. HUST:2016YXMS303).

References

[1] K. Sood, S. Yu, and Y. Xiang, "Software-defined wireless networking opportunities and challenges for Internet-of-things: A review," *IEEE Internet of Things Journal*, vol. 3, no. 4, pp. 453–463, Aug 2016.

[2] S. S. Roy, A. Mallik, R. Gulati, M. S. Obaidat, and P. V. Krishna, "A deep learning based artificial neural network approach for intrusion detection," in *International Conference on Mathematics and Computing (ICMC)*, 2017, pp. 44–53.

[3] C. Zhang, H. Cheng, X. Hei, and B. Bensaou, "Learning multi-paths for edge networks in a stochastic approximation approach," in *IEEE International Conference on Consumer Electronics - Taiwan (ICCE-TW)*, May 2018.

[4] "KDD Cup 99," accessed on 2017-11-20. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[5] Z. Li, Z. Qin, K. Huang, X. Yang, and S. Ye, "Intrusion detection using convolutional neural networks for representation learning," in *Neural Information Processing*, 2017, pp. 858–866.

[6] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009.

[7] Y. Gao, L. Dai, and X. Hei, "Throughput optimization of multi-BSS IEEE 802.11 networks with universal frequency reuse," *IEEE Transactions on Communications*, vol. 65, no. 8, pp. 3399–3414, Aug 2017.

[8] T. Zahid, X. Hei, W. Cheng, A. Ahmad, and M. Pasha, "On the tradeoff between performance and programmability for software defined WiFi networks," *Wireless Communications and Mobile Computing*, 2018.

[9] W. W. Nsunza and X. Hei, "Design and implementation of a smart home router based on Intel Galileo gen 2," in *International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities (TRIDENTCOM)*, Sep 2017.

[10] W. W. Nsunza, S. Rutunda, and X. Hei, "Design and implementation of a low-cost software defined wireless network testbed for smart home," in *International Workshop on Network Optimization and Performance Evaluation (NOPE)*, Dec 2017.

[11] J. Kang, X. Hei, and J. Song, "A comparative study of Zynq-based OpenFlow switches in a software/hardware co-design," in *International Workshop on Network Optimization and Performance Evaluation (NOPE)*, Dec 2017.

[12] R. Klöti, V. Kotronis, and P. Smith, "OpenFlow: A security analysis," in *IEEE International Conference on Network Protocols (ICNP)*, Oct 2013, pp. 1–6.

[13] D. Li, X. Hong, and J. Bowman, "Evaluation of security vulnerabilities by using ProtoGENI as a launchpad," in *IEEE Global Telecommunications Conference*, Dec 2011, pp. 1–6.

[14] O. Flauzac, C. González, A. Hachani, and F. Nolot, "SDN based architecture for IoT and improvement of the security," in *IEEE International Conference on Advanced Information Networking and Applications Workshops*, March 2015, pp. 688–693.

[15] N. Gao, L. Gao, Q. Gao, and H. Wang, "An intrusion detection model based on deep belief networks," in *International Conference on Advanced Cloud and Big Data*, Nov 2014, pp. 247–252.

[16] M.-J. Kang and J.-W. Kang, "Intrusion detection system using deep neural network for in-vehicle network security," *PLOS ONE*, vol. 11, no. 6, pp. 1–17, June 2016.

[17] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," in *International Conference on Bio-inspired Information and Communications Technologies*, 2016.

[18] G. Lacey, G. W. Taylor, and S. Areibi, "Deep learning on FPGAs: Past, present, and future," *ArXiv e-prints*, Feb. 2016.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[20] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing FPGA-based accelerator design for deep convolutional neural networks," in *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2015, pp. 161–170.

[21] "Microsoft research: catapult, 2015," accessed on 2018-01-19. [Online]. Available: https://www.microsoft.com/en-us/research/project/project-catapult/