Indonesian Journal of Electrical Engineering and Computer Science
Vol. 23, No. 2, August 2021, pp. 1059~1067
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v23.i2.pp1059-1067

Accelerating the update of a DL-based IDS for IoT using deep transfer learning

Idriss Idrissi, Mostafa Azizi, Omar Moussaoui
MATSI Research Lab., ESTO, Mohammed First University, Oujda, Morocco

Article Info

Article history:

Received Jun 8, 2021
Revised Jul 4, 2021
Accepted Jul 7, 2021

Keywords:

Convolutional neural networks
Deep learning
Internet of things
Intrusion detection system
System updates
Transfer learning

ABSTRACT

Deep learning (DL) models are nowadays broadly applied and have shown outstanding performance in a variety of fields, including our focus topic of "IoT cybersecurity". Deep learning-based intrusion detection system (DL-IDS) models are more fixated and depended on the trained dataset. This poses a problem for these DL-IDS, especially with the known mutation and behavior changes of attacks, which can render them undetected. As a result, the DL-IDS has become outdated. In this work, we present a solution for updating DL-IDS employing a transfer learning technique that allows us to retrain and fine-tune pre-trained models on small datasets with new attack behaviors. In our experiments, we built CNN-based IDS on the Bot-IoT dataset, and updated it on small data from a new dataset named TON-IoT. We obtained promising results in multiple metrics regarding the detection rate and the training between the initial training for the original model and the updated one, in the matter of detecting new attacks behaviors and improving the detection rate for some classes by overcoming the lack of their labeled data.

Corresponding Author:

Idriss Idrissi
MATSI Laboratory, ESTO
Mohammed First University
BP 473 Campus universitaire Al Qods, Oujda 60000, Morocco
Email: idrissi@ump.ac.ma

1. INTRODUCTION

Deep learning has succeeded in a variety of fields in recent years, including medical image processing, natural language processing, and cybersecurity. In the IoT cybersecurity [1], the machine trains on numerous gathered and labeled cyberattacks as well as normal traffic to learn them; this data comes from many datasets containing several attacks, from which this machine may eventually detect new similar attacks. However, the area of cybersecurity continues to confront several problems in identifying evolving threats. Models trained on known attack datasets produce excellent results [2]. These findings, however, are directly dependent on the data on the training dataset. The issue is that these attacks change in behavior (or some of them are zero hour) making them undetected by the model, which must be retrained on these new behaviors [3].

This problematic of data dependency [4] along with the lack of big datasets that contain new and updated attacks behaviors, makes the creation of an updated IDS a big challenge. And here, transfer learning comes to help solving these problems where training and testing data is not mandatory. This technique gains the existing knowledge from a trained model on a certain dataset and merges it with the new one trained on a small dataset that contains new attacks behaviors [5]. Thus, the new updated model will not be built from scratch or from a poor dataset [4].

Transfer learning is a technique that benefits from an already trained weight on big datasets for a long period of time and transfers this knowledge [6]. As an example, the Google's language representation

model “BERT” has 110 million parameters and was trained for 4 days on 16 Cloud TPUs [7], the OpenAI’s language generator GPT3 which is a model that has 175 billion parameters [8], or the real-time object detection “YOLO” with 89 million parameters in the fifth version [9]. The idea to retrain these models from scratch with new data is very expensive, both in terms of time and resources. Therefore, implementing transfer learning is more practical and beneficial for this situation.

The rest of this paper is structured as follows. In the section 2, we describe the background, the section 3 presents the related works, the section 4 presents our methodology, the section 5 we present and discuss the obtained results, and finally a conclusion as a section 6.

2. BACKGROUND

2.1. Convolutional neural networks

CNN or ConvNets are a class of deep neural networks that are used in many fields but mostly in pattern recognition [10]. It is composed of an input layer, many hidden layers in between, and an output layer as shown in Figure 1 like the multi-layer perceptron (MLP) networks. Best known and used layers are: convolution [11], activation or ReLU, and pooling [12], [13]. In contrast to standard feature selection algorithms, it is capable of dynamically learning better features and categorizing traffic. Furthermore, because it shares the same convolution matrix (kernel), it can accomplish better classification and learn new features with more traffic data, reducing the number of parameters and training calculation total substantially. In contrary to other deep-learning or machine learning algorithms that can be over-fitted with huge vast data, CNN can recognize the type of an assault quickly. Furthermore, literature findings suggest that using CNNs in the intrusion detection field produces superior results than other methods [14].

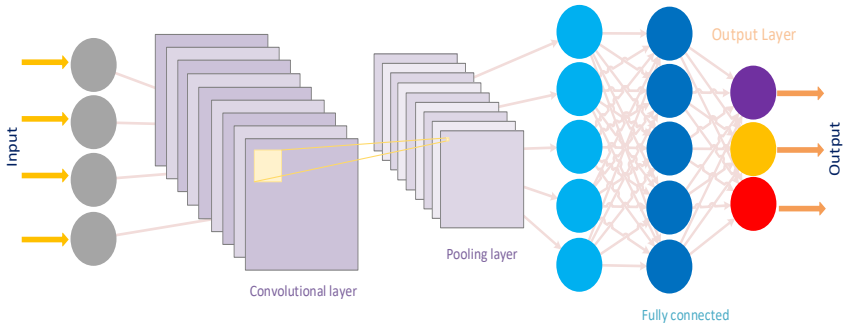

Figure 1. CNN architecture

2.2. Transfer learning

TL is a machine and deep learning research area that aims to transfer knowledge from one or more source tasks to one or more target tasks [15]. Supposing a source domain D_S, a learning task T_S, a target domain D_T, and a learning task T_T; TL serves in improving the learning of the target predictive function $f_T(.)$ in D_T using the knowledge in D_S and T_S, where $D_S \neq D_T$, or $T_S \neq T_T$ [16]. TL is all about using the features learned on one problem and leveraging them on a new similar problem. For example, taking the characteristics of a model who has learned to identify cats, it can be useful in creating a model for identifying tigers [17]. TL is typically performed for tasks where there is a little dataset to train a full-scale model from scratch. In our experiments, this involves updating an already existing IDS model with new attack behaviors from a small dataset containing these new behaviors and without the need of building a new big dataset and retraining it.

There are different methods to pull out the transfer learning in the deep learning context, it depends on how much data we have got. For example, it could freeze all layers and train only on the last one, or freeze most layers and train the last ones, or training all layers by initializing the weights on the pre-trained ones. In our experiments, we freeze most of the layers and training the last ones using CNN on datasets Bot-IoT (source domain D_S) and TON-IoT (target domain D_T).

Transfer learning can be achieved by removing the original model classifier, then adding a new one that fits our purposes as shown in Figure 2, and finally fine-tuning this new model rendering on one of three approaches [18]:

a) Training the whole model. In this approach, we use the original architecture of the pre-trained model and train it accordingly to the new dataset. This means that the model will be retrained from scratch. In this approach we need a big dataset;

b) Training only some layers. In this approach, we freeze some layers where we retrain the remaining ones, in the case of a small dataset and a large number of parameters, we froze more layers to avoid overfitting. But in the case of a large dataset and a small number of parameters, where overfitting is not a problem, we can improve our model by retraining more layers to our new task;
c) Freezing the convolutional base. This approach is more like the last one but its main idea is to keep the convolutional base (which is a stack of convolutional and pooling layers) in its original form, then use its outputs to feed the classifier (which is generally the fully connected layers) as shown in Figure 2. In this case, we can use just a small dataset with a minimum of computational power. Basing on these advantages, we will use this last approach in our proposed method.

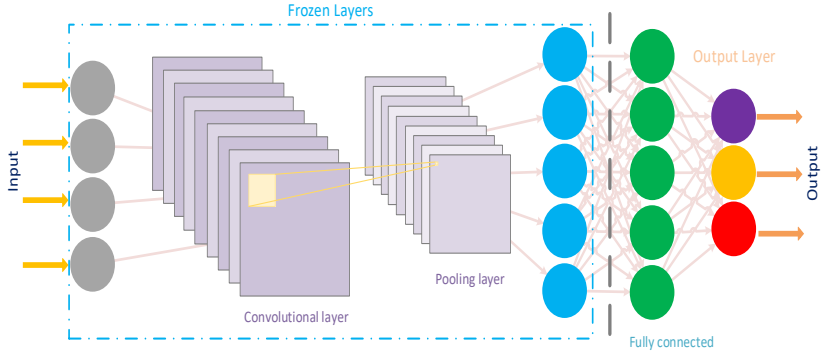

Figure 2. Transfer learning (freezing the convolutional base)

2.3. Datasets

Bot-IoT: It is a cybersecurity dataset that was created specifically for IoT systems by an actual network milieu. The environment incorporates a combination of usual normal and bad traffic, with 4 types of attacks and 10 subcategories. Namely, reconnaissance (service scanning and OS fingerprinting), DDoS (TCP, UDP, and HTTP), DoS (TCP, UDP, and HTTP), theft (keylogging and data exfiltration) [19].

TON-IoT: It is also a cybersecurity dataset that was created specifically for Industry 4.0/IoT and Industrial IoT (IIoT) in 2019. It contains heterogeneous data, collected from IoT and IIoT sensors, operating systems (Windows and Linux), and network traffic.

These datasets were collected at the cyber range and IoT Labs of the UNSW Canberra Cyber from a realistic and large-scale network designed. These datasets contain nine attack categories. Namely, scanning attack, DoS, DDoS, ransomware attack, backdoor attack, injection attack, cross-site scripting (XSS) attack, password attack, and man-in-the-middle (MITM) attack, against vulnerable elements of IoT/IIoT applications, operating systems, and network systems [20].

3. RELATED WORKS

In 2019, Singla *et al.* [21] showed experimentally on the UNSW-NB15 dataset (9 sub-datasets) that Transfer Learning is effective in reducing the quantity of labeled data required for training neural network models for network IDS, by observing the classification accuracy (up to 26.4% better accuracy) and precision metrics for comparing and evaluating the resulting neural networks models.

In 2019, Zhang *et al.* [22] proposed an autonomous classifier update scheme, challenging the matter of updating an existing network traffic classifier. They evaluated their work by using MLP and CNN models on the ISCX VPN-nonVPN dataset and the metrics recall, precision, and F1 score. They also demonstrated that their proposed classifier update scheme can help building a dataset of the new application from an active network traffic.

In 2019. Sameera *et al.* [23] applied transfer learning to detect unlabeled R2L from the NSL-KDD dataset by making use of labeled DoS attacks of NSL-KDD dataset, and got an accuracy of 89.79%) and an FPR of 0.15%. Then, in [24], the authors proposed a deep transductive transfer learning framework for detecting zero-day attacks exhibited as the target domain with no labeled examples. They employ the DAMA manifold alignment approach, creating target soft labels to compensate for the absence of labeled target instances through the application of cluster correspondence processes. They investigated their proposed models using multiple machine learning algorithms such kNN, SVM, RF, DT, and DNN with two test scenarios. In the first one, they investigated and tested using the NSL-KDD dataset, and in the second one, they detected zero-day attacks of the CIDD cloud dataset by considering the DoS part of the NSL-KDD as a source dataset.

In 2020, Qureshi *et al.* [25] proposed a deep neural network and adaptive self-taught based transfer learning (named DST-TL) technique that uses self-taught learning to develop their IDS. They conducted their

experiment by comparing their technique to the performance of conventional classifiers (like MLP), nonlinear principal component analysis (NLPCA), and deep belief network (DBN)). First, by extracting features and passing the original feature set of NSL-KDD dataset through the pre-trained network. Then, a mixture of original and extracted features is used in the training of the sparse auto-encoder. The proposed approach (DST-TL) in [25] showed a clear improved prediction accuracy.

In 2019, Zhang *et al.* proposed in [21] a transfer learning framework based on the domain-adversarial training technique for intrusion detection in a Smart Grid environment. They use a realistic smart grid security dataset, which is collected through hardware-in-the-loop simulators. Their framework was capable of mixing different baseline machine learning classifiers in order to improve performance, where they got improvements from 7% up to 36.8% using AdB, kNN, SVM, and RF classifiers, and an average improvement over 2.5% for both CART and ANN.

In these related works, some researchers used only one dataset to perform both the pretraining and the Transfer Learning, while others exploited different datasets for the model's pretraining regarding the transfer learning model. Furthermore, some of the used datasets are more general and not really collected from attacks on IoT systems, or have worked with machine learning. In our experiments, we present a proof of concept of the DL-IDS update using two recent IoT datasets. The first one (Bot-IoT) is generated in 2018 and the second one (TON-IoT Network) is generated in 2019 using the best-known Deep Learning algorithm CNN.

4. PROPOSED METHOD

The purpose of our method is to create a CNN-based IDS model using the Bot-IoT dataset first, and then update it with the TON-IoT dataset as shown in Figure 3 by following the steps 1st phase-6th phase below:

a) 1st Phase: Bot-IoT Dataset preprocessing; In this step, we preprocessed our dataset (we worked with the entire dataset; more than 72 million records of data, with all features) by altering the raw data and normalizing its values, and then convert it into image shape;

b) 2nd Phase: TON-IoT Dataset preprocessing; Firstly, we need to adapt the new data with the old dataset, and that by generating the missing features like (Stddev, State, Mean, Min, Max, Seq, Srate, Drate, …) from the pcap files using the Argus tool [26]. Then secondly, we preprocessed it using the same preprocess of the Bot-IoT dataset preprocessing (we took only 50000 records of data from each class; 30000 for the training dataset, 10000 for the validation dataset, and the remaining other 10000 for the test dataset), to ensure the same adaptivity between the two datasets;

c) 3rd Phase: Building the original model; afterward, we build and train our model on a training dataset (60% of the Bot-IoT dataset), and we use the validation dataset (20% of the dataset) to validate the accuracy of our model;

d) 4th Phase: Evaluating the model on new data; after building the original model, firstly evaluated with the test dataset of the Bot-IoT dataset (the remaining 20% of the dataset), then we evaluate on the new attack's behaviors on the test dataset of the TON-IoT dataset (20% of the dataset) by predicting attacks;

e) 5th Phase: Updating the model; from the original and already trained model, we freeze the Convolutional base and then use its outputs to feed the Classifier. We retrain the classifier layers on top of the frozen ones on a training dataset (50% of TON-IoT dataset and 10% of BoT-IoT dataset), and using the validation dataset (15% of the dataset TON-IoT dataset and 5% of BoT-IoT dataset) we validate the accuracy of our model. The idea behind retraining on new and old data is that we do not want our model to be influenced only on the new attack's behaviors;

f) 6th Phase: Evaluating the updated model; after updating the model, we evaluate it with the test dataset of the TON-IoT dataset (15% of the dataset TON-IoT dataset and 5% of BoT-IoT dataset) by predicting attacks.

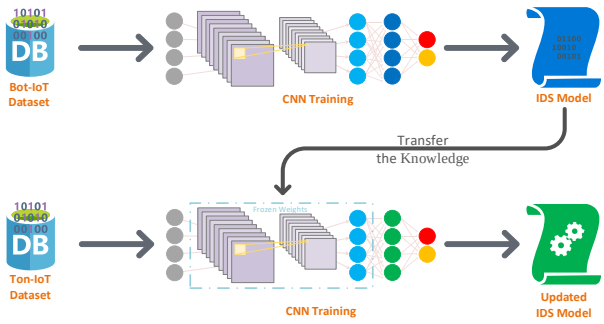

Figure 3. Our scheme of updating the DL-based IDS model

4.1. Building the original model

We build our IDS model resorting on CNN as in our previous work [1], this model was defined by an input layer with 16 input neurons, five hidden layers Convolution1D layer, MaxPooling1D layer, Flatten layer, ReLU layer, Dense layer, and an output layer. We initially trained our model in 10 epochs (the whole dataset is passed through the neural network 10 times) with a batch size of 32 (the amount of training samples in a single batch is 32). The neural network includes 16 input neurons (the same number as the features), with 4 intermediate (hidden) layers, 16 (Convolution1D), 8 (MaxPooling1D), 256 (Flatten), 256 (ReLU) neurons, 44 (Dense) neurons, and 4 output neurons for the multiclass classification as shown in Figure 4. We trained and tested our model on the Bot-IoT dataset that contains around 72 million records of data traffic simulated IoT environment. The training and test dataset contains 11 classes which reflect 10 types of attacks within 4 attacks categories and the normal traffic; meaning 5 classes if we work only with categories, which is in our case in the goal to have the same class with the target dataset.

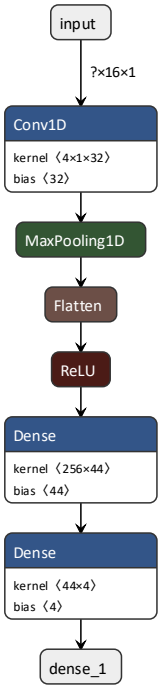

Figure 4. Specification of our IDS model layers

4.2. Building the updated model.

As shown in Figure 3, from the previous and already trained model on the original dataset (Bot-IoT), we freeze the convolutional base (keep it in its original form to avoid destroying any of the information they contain during future training rounds; meaning transferring the parameters); which is the stack of Convolution1D layer, MaxPooling1D layer, Flatten layer, and ReLU layer. The important goal of the convolutional base is to generate features from the base dataset (Bot-IoT), and then use its outputs to feed the classifier; which is the stack of the dense layers, and the output layer (these are fully the connected layers). We retrain the classifier layers on top of the frozen ones. These retrained layers will learn to turn the old features into predictions on the new dataset (TON-IoT).

5. RESULTS AND DISCUSSION

5.1. Hardware characteristics

The results we obtained were performed on a high-performance computing (HPC) infrastructure with the following hardware characteristics:

a) CPU: two Intel Xeon Gold 6148 (2.4 GHz/20 cores)
b) RAM: 192 Gb
c) GPU: two NVIDIA Tesla P100 (12 Gb) with cuda v10.1

In our experiments, we worked with Keras (2.4.0) [27]; an open-source python Deep Learning library which is running on top of Google's open-source data flow software, and uses TensorFlow-GPU (2.3.0) [28] as a backend engine.

5.2. Evaluation metrics

To evaluate our models, we used the specified metrics: Accuracy (Acc), Loss (Ls), Precision (P), Recall (R), F1 Score ($F1$), and Confusion Matrix (CM). These metrics are calculated using the following four different measures [29]:

1) True Positive ($\boldsymbol{TP}$): is the number of positive class records correctly classified.
2) True Negative ($\boldsymbol{TN}$): is the number of negative class records correctly classified.
3) False Positive ($\boldsymbol{FP}$): is the number of negative class records wrongly classified.
4) False Negative ($\boldsymbol{FN}$): is the number of positive class records wrongly classified.

– Accuracy (Acc): is the percentage of true detection over the total traffic records and calculated using the following formula (1):

$$Acc = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

– Loss (Ls): is the difference between the predicted value and the true value. The most used loss function in deep neural networks is cross-entropy [30], and calculated using the following formula (2):

$$Ls = -\sum_{c=1}^{M} y_{o,c} log\ (p_{o,c}) \quad (2)$$

While:
M : is the number of classes (5 in our case)
y : is the binary indicator (true 1 or false 0) if class label c is the correct classification for observation o
p: is the predicted probability observation o of class c

– Precision (P): is the percentage of predicted attacks traffic that are truly attacks, and calculated using the following formula (3):

$$P = \frac{TP}{TP+FP} \quad (3)$$

– Recall (R): is the percentage of attacks traffic versus all the attacks traffic obtainable, and calculated using the following formula (4):

$$R = \frac{TP}{TP+FN} \quad (4)$$

– F1 Score ($F1$): is a measure of the test's accuracy. It is calculated from the precision and recall of the test, and calculated using the following formula (5):

$$F1 = \frac{2}{P^{-1}+R^{-1}} \quad (5)$$

– Confusion Matrix (CM): is a table that gives a visualization of the performance of the model by representing the instances in the predicted class in a row while represents the instances in an actual class in the columns.

5.3. Evaluating the model

In the following Figures 5-7, we present the accuracy training, loss training, accuracy validation, loss validation, and for each class, we present the Recall and Precision, and with the F1 score for the original model which was trained over 10 epochs. As shown in Figure 5 for the original model in 10 epochs (8.78 hours of training) the accuracy reached 99,99% and the loss attained 0,15% in training and for the validation as shown in Figure 6 it reached 99,99% in accuracy and 0,12% in loss, and a 100% for the testing set. But the accuracy fails to control for size imbalances in the classes that's why it doesn't allow us to have a clearer view on how the model is doing for each class; it does not give a per-class metric for multi-class problems. Regarding the Recall and the Precision, we get a clearer result for each class. We see here that in some classes such as 5: "Theft" we obtained very low measures in all the metrics (0%), and for the for the class 3: "Normal traffic", we obtained very low measures in both recall and F1 score as shown in Figure 7, meanwhile the precision is falsely high (100%) due to weakness of this metric, and it rarely predicts this class. For the Class 2: "DoS", the recall metric also has a weakness where it can achieve very high measurement in this class by always predicting it and that could mean lots of incorrect guesses (in the DoS class with the new data in Figure 8). Each metric can be biased for different classes and that due to the unbalanced data in these classes, so using the harmonic mean of Precision and Recall, the F1-score that gives us a better result of the incorrectly classified cases than the accuracy metric.

We benchmarked the original model on new data from the TON-IoT network dataset and got a decrease down to 56% on the F1 score metric; and especially for the normal and theft classes, we obtained respectively only 3% and 27% for the same metric as shown in Figure 8. This decrease was actually due to two causes; the first one is that we tested our model on balanced data over different classes which allowed to make balanced metric results; while the second cause is related to new behaviors or mutations in the attacks switching from the old to the new datasets.

In Figures 9-11, we present the accuracy training, loss training, accuracy validation, loss validation, and for each class, we present the Recall and Precision, and along with F1 score of the updated model that was retrained over 10 epochs in just 170 seconds of training (17s in every epoch). After updating the model using transfer learning, we reached an accuracy of 99.43% and a loss of 0.36% in training and for the validation, it also reaches 99.47% in accuracy and 0.23% in loss, for the F1 score it reaches 99% as shown in Figure 11 We can remarque an improvement for the DoS, DDoS, and Scan attacks, and a bigger one for the remaining classes; Theft and Normal traffic, meaning that our IDS has not only been updated but also overcome the lack of labeled data in these classes, especially the normal one because its importance is when differencing between good and bad traffic.

We also compared the training time for the initial training for the original model and the updated one as shown in Figure 12 where we see a big difference between the two, from 31590 seconds in the initial training of the original IDS (trained over 10 epochs; around 31590 seconds in every epoch) to only 170 seconds were updating it (the update of the DDoS, Dos, and Reconnaissance was done only in 3 epochs, and we prolonged the training 7 more epochs to train the other classes; around 17 seconds in every epoch). This proves that transfer learning is a solution for transfer learning with minimal computing power and much fewer data compared to the original training.

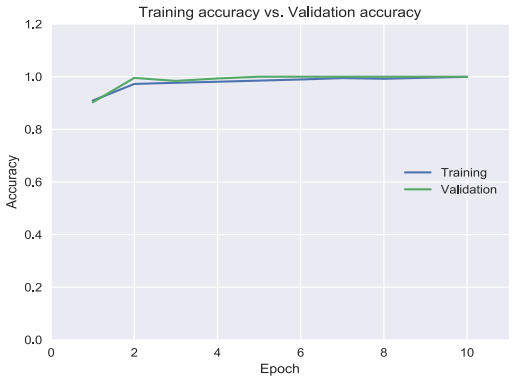

Figure 5. The accuracy in the training and the validation (original model)

Figure 6. The loss in the training and the validation (original model)

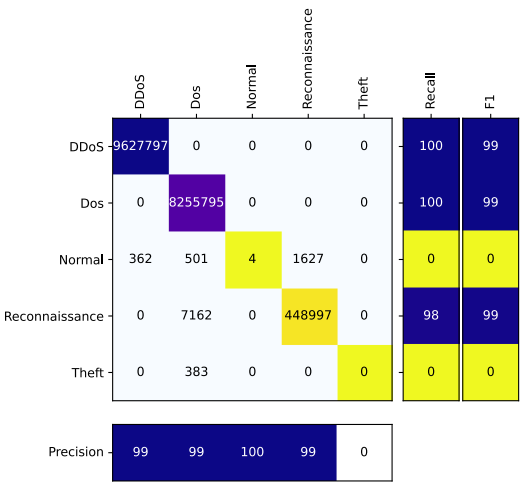

Figure 7. Confusion matrix (for the original model on the original dataset

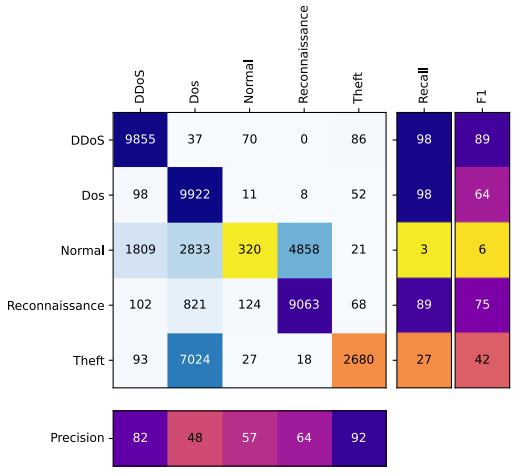

Figure 8. Confusion matrix (for the original model on the new dataset)

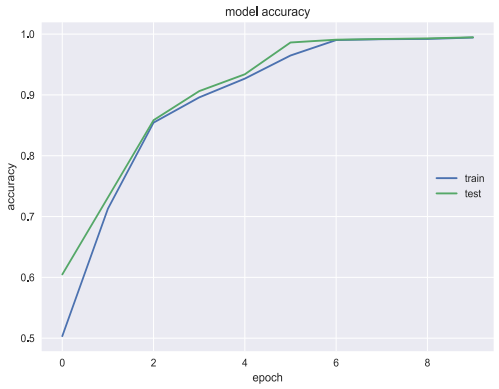

Figure 9. The accuracy in the training and the validation (updated model)

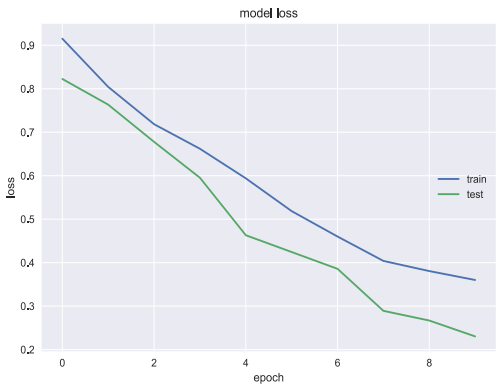

Figure 10. The loss in the training and the validation (updated model)

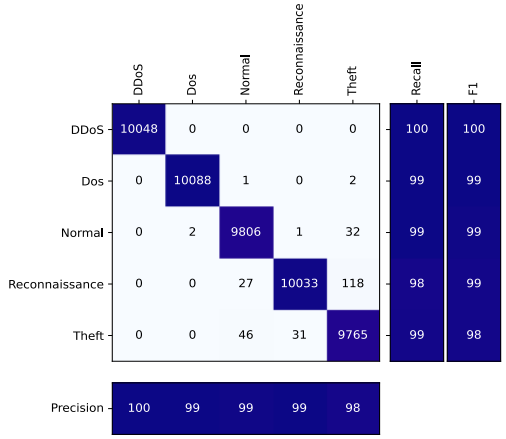

Figure 11. Confusion matrix (the updated model)

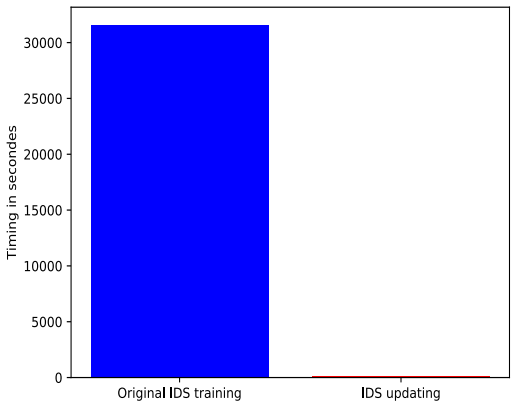

Figure 12. Training time

6. CONCLUSION

The enormous network traffic data between IoT objects dispatched around the world have taken a big challenge to traditional intrusion detection system (IDS). Researchers tend to build IDS based on Deep learning due to its outstanding performance in various fields, which itself got some problems like data-dependent or lack of labeled data. Our proposed "Updated deep transfer learning-based intrusion detection system for IoT" was built initially with convolutional neural networks (CNN) on Bot-IoT dataset and been updated on a small amount of data from the TON-IoT dataset, after several experiments, we obtained a reasonable detection rate on this new updated IDS. By analyzing the obtained results, we concluded that Transfer Learning can be an ideal solution not only to compensate the lack of data in some attack classes but also to update the IDS systems with just a minimal computing power and effort. As future works, we will deploy our IDS in a real IoT environment, and on a lightweight IoT device while optimizing it with no accuracy loss while studying its performance on real IoT network traffic data.

ACKNOWLEDGEMENTS

This research was supported through computational resources of HPC-MARWAN provided by the National Center for Scientific and Technical Research (CNRST) Rabat, Morocco.

REFERENCES

[1] I. Idrissi, M. Boukabous, M. Azizi, O. Moussaoui, and H. El Fadili, "Toward a deep learning-based intrusion detection system for iot against botnet attacks," *IAES International Journal of Artificial Intelligence(IJ-AI)*, vol. 10, no. 1, pp. 110–120, Mar. 2021, doi: 10.11591/ijai.v10.i1.pp110-120.

[2] I. Idrissi, M. Mostafa Azizi, and O. Moussaoui, "A Lightweight Optimized Deep Learning-based Host-Intrusion Detection System Deployed on the Edge for IoT," *International Journal of Computing and Digital Systems*, 2021.

[3] K. Bartos, M. Sofka, and V. Franc, *Optimized Invariant Representation of Network Traffic for Detecting Unseen Malware Variants*. 2016.
[4] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11141 LNCS, 2018, pp. 270–279.
[5] S. T. Krishna and H. K. Kalluri, "Deep learning and transfer learning approaches for image classification," *International Journal of Recent Technology and Engineering*, vol. 7, no. 5S4, pp. 427–432, 2019.
[6] L. Mou and Z. Jin, *Tree-Based Convolutional Neural Networks*, Springer Singapore, 2018, doi: 10.1007/978-981-13-1870-2.
[7] Q. Yang, Y. Zhang, W. Dai, and S. J. Pan, *Transfer learning*, Cambrige, United Kingdom: Cambridge University Press, 2020, doi: 10.1017/9781139061773.
[8] T. B. Brown, *et al.*, "Language Models are Few-Shot Learners," *arXiv:2005.14165*, 2020.
[9] "GitHub - ultralytics/yolov5: YOLOv5 in PyTorch > ONNX > CoreML > iOS." [Online]. Available: https://github.com/ultralytics/yolov5 (accessed Aug. 12, 2020).
[10] M. Berrahal and M. Azizi, "Review of DL-Based Generation Techniques of Augmented Images using Portraits Specification," in *4th International Conference on Intelligent Computing in Data Sciences, ICDS 2020*, Nov. 2020, pp. 1–8, doi: 10.1109/ICDS50568.2020.9268710.
[11] K. Uchida, M. Tanaka, and M. Okutomi, "Coupled convolution layer for convolutional neural network," *Neural Networks*, vol. 105, pp. 197–205, Sep. 2018, doi: 10.1016/j.neunet.2018.05.002.
[12] M. Sun, Z. Song, X. Jiang, J. Pan, and Y. Pang, "Learning Pooling for Convolutional Neural Network," *Neurocomputing*, vol. 224, pp. 96–104, Feb. 2017, doi: 10.1016/J.NEUCOM.2016.10.049.
[13] M. Boukabous and M. Azizi, "Review of Learning-Based Techniques of Sentiment Analysis for Security Purposes," in *Innovations in Smart Cities Applications Volume 4*, 2021, pp. 1–14, doi: doi.org/10.1007/978-3-030-66840-2_8.
[14] I. Idrissi, M. Azizi, and O. Moussaoui, "IoT security with Deep Learning-based Intrusion Detection Systems: A systematic literature review," in *4th International Conference on Intelligent Computing in Data Sciences, ICDS 2020*, 2020, pp. 1–10, doi: 10.1109/ICDS50568.2020.9268713.
[15] M. Boukabous and M. Azizi, "A Comparative Study of DL-Based Language Representation Learning Models," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 22, no. 2, 2021, doi: 10.11591/ijeecs.v22.i2.pp1032-1040.
[16] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345-1359, Oct. 2010, doi: 10.1109/TKDE.2009.191.
[17] Y. Guo, H. Shi, A. Kumar, K. Grauman, T. Rosing, and R. Feris, "SpotTune: Transfer Learning Through Adaptive Fine-Tuning." *arXiv:1811.08737*, pp. 4805–4814, 2019.
[18] "Transfer learning from pre-trained models | by Pedro Marcelino | Towards Data Science." 2018. [Online]. Available: https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751 (accessed Sep. 07, 2020).
[19] "The BoT-IoT Dataset." [Online]. Available: https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/bot_iot.php (accessed Feb. 22, 2020).
[20] N. Moustafa, "ToN_IoT datasets," *IEEE Dataport*, 2019, doi: 10.21227/fesz-dm97.
[21] A. Singla, E. Bertino, and D. Verma, "Overcoming the Lack of Labeled Data: Training Intrusion Detection Models Using Transfer Learning," *2019 IEEE International Conference on Smart Computing (SMARTCOMP)*, 2019, pp. 69-74, doi: 10.1109/SMARTCOMP.2019.00031.
[22] J. Zhang, F. Li, H. Wu, and F. Ye, "Autonomous Model Update Scheme for Deep Learning Based Network Traffic Classifiers," *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1-6, doi: 10.1109/GLOBECOM38437.2019.9014036.
[23] N. Sameera and M. Shashi, "Transfer learning based prototype for zero-day attack detection," *International Journal of Engineering and Advanced Technology (IJEAT)*, vol. 8, no. 4, pp. 1326–1329, Apr. 2019.
[24] N. Sameera and M. Shashi, "Deep transductive transfer learning framework for zero-day attack detection," *ICT Express*, Mar. 2020, doi: 10.1016/j.icte.2020.03.003.
[25] A. S. Qureshi, A. Khan, N. Shamim, and M. H. Durad, "Intrusion detection using deep sparse auto-encoder and self-taught learning," *Neural Comput. Appl.*, vol. 32, no. 8, pp. 3135-3147, Apr. 2020, doi: 10.1007/s00521-019-04152-6.
[26] "openargus - Using Argus." [Online]. Avaialable: https://openargus.org/index.php/using-argus (accessed Sep. 11, 2020).
[27] N. Ketkar, "Introduction to Keras," in *Deep Learning with Python*, Berkeley, CA, United States: Apress, 2017, pp. 97-111, doi: 10.1007/978-1-4842-2766-4_7.
[28] M. Abadi, *et al.*, "TensorFlow: A System for Large-Scale Machine Learning," *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI '16)*, 2016, pp. 265–283.
[29] N. Japkowicz, "Why Question Machine Learning Evaluation Methods? (An illustrative review of the shortcomings of current methods)," *AAAI workshop on evaluation methods for machine learning*, pp. 6-11, 2006.
[30] Q. Wang, Y. Ma, K. Zhao, and Y. Tian, "A Comprehensive Survey of Loss Functions in Machine Learning," *Annals of Data Science*, pp. 1-26, 2020, doi: 10.1007/S40745-020-00253-5.