A Reminiscent Intrusion Detection Model Based on Deep Autoencoders and Transfer Learning

Roger R. dos Santos, Eduardo K. Viegas, Altair O. Santin
Graduate Program in Computer Science (PPGIa)
Pontifical Catholic University of Parana (Pontifícia Universidade Católica do Paraná), Brazil
{robson.roger, eduardo.viegas, santin}@ppgia.pucpr.br

***Abstract*—Machine learning techniques for network-based intrusion detection often assume that network traffic does not change over time or that model updates can be easily performed. This paper proposes a novel, reminiscent intrusion detection model based on deep autoencoders and transfer learning to ease the model update burden in a twofold implementation. First, a deep autoencoder is used as an additional feature extraction stage to obtain a historical feature representation of network traffic. Second, at model updates, the deep autoencoder parameters are updated through a transfer learning procedure, thus, significantly decreasing the amount of needed labeled training data and the computational costs. Experiments performed on a 8TB dataset containing real and valid network traffic ranging for one year have shown that approaches in the literature cannot handle with the network traffic behavior changes over time, requiring impractical amounts of labeled data to be provided during model training tasks. In addition, if no model updates are performed, the proposed scheme can improve the true-negative rate by up to 23.9%. If done so, it can provide similar accuracy rates of traditional techniques while demanding only 22% of labeled training data and 28% of computational costs.**

***Index Terms*—Intrusion Detection, Deep Autoencoder, Transfer Learning.**

I. Introduction

Network-based Intrusion Detection Systems (NIDS) have been widely used to detect network threats [1], wherein authors often make use of *behavior-based* techniques that analyze the event behavior to achieve such a goal [2]. Several approaches have been proposed in such a context, in which machine learning (ML) through pattern recognition (*shallow*) techniques have yielded promising results [1]. In practice, a behavioral model is extracted from a training dataset, typically composed of millions of labeled samples from both *normal* and *attack* events [3]. The dataset must be made of a representative number of samples from the production environment, considering that the model will be built accordingly. The built model can then be used in production for the classification of other events.

The network environment's behavior is highly variable and changes over time, either due to the provision of new services or due to the discovery of new attacks [4]. Over time, the ML model will become outdated, making a NIDS unreliable, taking into account that the error rates will be increased when compared to the measurements in the test phase [3]. It is known that the underlying model of ML-based NIDS must be updated regularly. However, the model update task is not easily achieved [5]. In such a case, the network data from the production environment must be monitored for an extended period, resulting in a dataset with millions of network flows, wherein each event must be previously and correctly labeled before the training procedure can even be executed [6], [7]. However, the labeling of network events is a challenging task, as it often demands expert assistance for the assessment of collected events, which is not always available or available with a high cost [8].

Model updates in ML-based NIDS literature remain mostly overlooked [2], [4]. In general, proposed approaches either neglects the model update procedure or assumes that it can be easily performed periodically, despite the challenges it presents [3]. In contrast, traditional ML-based techniques discard the current outdated model in each update task. Thus, further increasing the computational costs of updating the model, as well as the amount of labeled training data required for proper model training. [1]. As a result, despite the highly accurate proposals in ML-based NIDS literature over the last decades, it remains mainly as a research topic, hardly deployed in production environments [4].

This paper proposes a twofold implementation of a reminiscent intrusion detection model based on deep autoencoders and transfer learning. First, intrusion detection is performed by a pattern recognition classifier (*shallow*), built on top of the outputs of a deep encoder (from autoencoder). The goal is to easiness model update procedure, as the classifier will be built taking into account the output set of features by the encoder layers, which are used as a representation of the network historical behavior. Second, to easiness model update procedures, at each model update, a transfer learning technique is applied on the outdated autoencoder, made of both encoder and decoder layers. Consequently, prior knowledge from data can be used during model update, thus, decreasing the demanded number of required labeled events as well as the computational training costs. The insight of our proposal is that deep autoencoder can be used as a historical feature representation of network data, thus, further decreasing the number of labeled events that must be provided during model updates. In summary, the main contributions of this paper are:

- We evaluate traditional ML-based NIDS concerning their classification performance over time according to the training dataset size. The experiments have shown that traditional approaches demand unfeasible amounts of

labeled training data for reliable intrusion detection.
- We propose and evaluate a new reminiscent intrusion detection model through transfer learning on deep autoencoders. The proposed scheme can significantly ease the model update procedure, significantly decreasing the demanded dataset training size and computational costs at model updates.

II. Preliminaries

A. Machine Learning for Intrusion Detection

A typical NIDS is composed of four sequential modules [9]. First, the *Data Acquisition* module continuously collects network events from the monitored environment, e.g., a network interface card (NIC). The collected data is then analyzed by a *Feature Extraction* module, which extracts a set of behavioral features to compound a feature vector. In general, in NIDS, the behavior of network events is represented through a network flow, which typically summarizes the communication between hosts and services in a specified time window, e.g., number of sent network packets in a 15 second interval [3]. The extracted feature vector is used as input by a *Classification* module, which applies a ML model for the event classification as either *normal* or *attack*, which is then signaled by an *Alert* module.

Several works have proposed highly accurate ML-based techniques for intrusion detection, mainly through pattern recognition means [1]. In general, the authors wrongly assume that the behavior of networked environments does not change over time [3]. As a result, little or no effort is given to facilitate the model update process [2], which should require as few as possible labeled network events, and low computational costs. However, traditional pattern recognition techniques are unable to provide such characteristics, considering that when a model update occurs, the outdated model is discarded, and a new training dataset with millions of labeled events must be built [10].

B. Deep Autoencoders

In recent years, significant research effort was given on deep neural network techniques [11], which currently provides the most superior results in fields such as image classification and object detection. One of such deep neural architectures is the deep autoencoders, which is typically used for dimensionality reduction [12], and image denoising, through a two-step process, namely *encoder* and *decoder*. The *encoder* goal is to perform the compression of the deep neural network input, e.g., by mapping a N sized input feature vector to a $N/3$ sized output. Therefore, to achieve such a goal, the *encoder* must learn a set of helpful feature relations of its input. In contrast, the *decoder* goal is to perform the decompression of its input, e.g., by mapping a $N/3$ sized input feature vector back to its N sized output. Thus, the deep autoencoder output is a learned representation of the most valuable relations between its input features. Research on deep autoencoders for intrusion detection, although with several proposals, is still in its beginnings [13]. In general, proposed schemes rely on such context makes use of it for dimensionality reduction [12], or even outlier detection [14], aiming as a final goal accuracy improvements on their detection scheme.

III. Related Works

Authors generally deal with intrusion detection as a supervised classification task through pattern recognition (shallow) approaches, giving little or no attention to model update challenges [3]. For instance, A. Ahmim *et al.* [15] proposes an ensemble of tree-based (shallow) classifiers for intrusion detection. Their proposed scheme was able to improve accuracy when compared to single classifiers. However, the model update challenge is overlooked. Another ensemble of tree-based classifiers is proposed by J. Kevric *et al.* [16], where the authors evaluated a series of classifier combination schemes. Similarly, model updates are not addressed.

Deep-based classification techniques are also widely explored in intrusion detection tasks. For instance, R. Vinayakumar *et al.* [17] have shown that convolutional neural networks yield more accurate detection than shallow-based classification techniques. Similarly, H. Yang. *et al.* [18] relies on convolutional neural networks for feature extraction of wireless attacks, which also improves detection accuracy when compared to its shallow counterpart. Although several works have used deep-based techniques in intrusion detection, the model update challenge is generally overlooked. J. Zhang *et al.* [5] proposes a transfer learning approach on neural networks for intrusion detection. Their technique filters data from new applications and uses them with transfer learning for model updates. However, although model updates are addressed, the authors assume that new services can be identified and labeled as needed, a wrong assumption in NIDS.

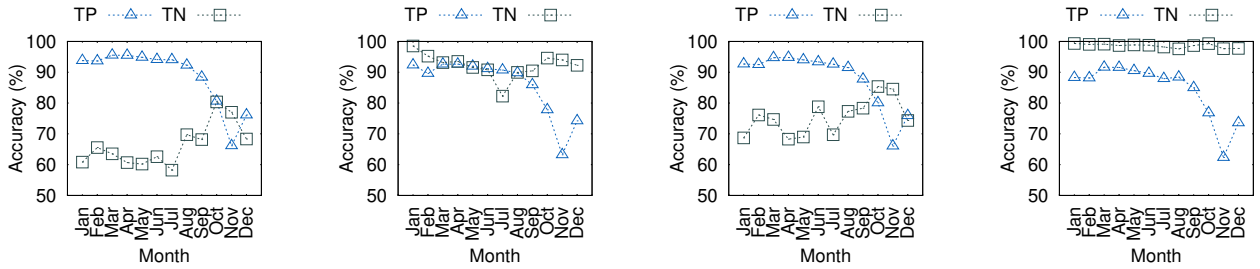
Several works are also exploring the application of deep autoencoders in recent years in NIDS. X. Li *et al.* [12] copes a deep autoencoder with a random forest feature selection approach to decrease model update computational costs. However, the authors assume that events can be labeled as needed and only aim to decrease computational costs, regardless of the required number of labeled instances at model updates. On the other hand, Y. Yan *et al.* [13] applies a series of autoencoders for the feature extraction task. Their proposed scheme can increase accuracy while the model update task is overlooked.

IV. Problem Statement

In general, proposed ML-based NIDS in the literature does not take into account the non-stationary behavior of networked environments [4]. In this section, we further investigate how network behavior changes over time impact ML-based classification approaches.

A. MAWIFlow Dataset

Current datasets available in the literature assume that the behavior of network traffic does not change over time. Proposals built through such data cannot evaluate how their approach performs while facing network traffic behavior changes over time.



(a) DT (monthly model updates with 7 days) (b) DT (monthly model updates with 30 days) (c) GBT (monthly model updates with 7 days (d) GBT (monthly model updates with 30 days)

Fig. 1: Accuracy behavior over time on *MAWIFlow* dataset on a monthly basis. Classifiers are updated in a monthly periodicity using either the last 7 or 30 days as labeled training data.

Our work relies on *MAWIFlow* [3] dataset. The dataset was built through the Samplepoint-F from the MAWI [19] archive, hence, being made of real network traffic collected daily for an interval of 15 minutes from a transit link between Japan and the USA

The whole collected network traffic of 2016 was used to evaluate widely used ML-based NIDS. The built dataset comprises more than 8TB of data, compounding around 300 billion network packets. For the prior labeling of events, we apply an unsupervised ML technique from MAWILab [8], which automatically labels the input records as either *normal* or *attack*. MAWILab employs several unsupervised machine learning algorithms to find anomalies in MAWI data without individual or human assistance for the event labeling task. The found anomalies are labeled as *attack*, while the remaining data are assumed to be *normal* events. For the feature extraction task, the BigFlow [3] was used, which grouped events in intervals of 15 seconds while extracting 21 flow-based features from Nigel [19] work.

B. The Non-stationary Behavior of Network Traffic

The evaluation aims at answering two main research questions: (***RQ1***) *How much-labeled training data should be provided for proper training of ML-based NIDS?* (***RQ2***) *What is accuracy behavior of ML-based NIDS with periodic model updates and widely available labeled training data?*

Two widely used shallow classifiers were evaluated: Decision Tree (DT) and Gradient Boosting (GBT). The DT classifier was implemented through a C4.5 algorithm, with a confidence factor of 0.25 and $gini$ as the node split quality metric. The GBT classifier relies upon a 0.1 learning rate value, with $deviance$ as the loss function, using 100 decision trees as its base-learner. A random undersampling without replacement is used in the training procedure to balance the occurrence between the classes. The classifiers were implemented through $scikit-learn$ API v0.24. The classifiers were evaluated according to their True-Positive (TP) and True-Negative (TN) rates. The TP denotes the ratio of *attack* instances correctly classified as *attack*, while the TN denotes the ratio of *normal* instances correctly classified as *normal*.

The first experiment aims at answering $RQ1$ and evaluates the accuracy performance of the selected classifiers according to the training dataset size while performing monthly model updates. Each classifier was evaluated periodically - updated monthly with the data that occurred on the last 7 or 30 days. The goal is to evaluate if the number of provided labeled instances can be decreased without a significant impact on NIDS accuracy due to the high cost involved in labeling events in production environments. Figure 1 shows the accuracy performance of the evaluated classifiers with monthly updates using 7 days (Figs. 1a and 1c) or 30 days (Figs. 1b and 1d) of labeled data during model update. It is possible to note that both DT and GBT could not provide a high detection rate with only 7 days of model update training data. In contrast, when 30 days worth of data is used at model updates, as shown in Figures 1b and 1d, the evaluated approaches were able to provide higher classification accuracies throughout *MAWIFlow* dataset for up until October 2016. In such a case, in $Oct.$, $Nov.$ and $Dec.$, even further labeled training data should be provided at model updates to increase their accuracies, making such techniques unfeasible for ML-based NIDS.

The second experiment aims at answering question $RQ2$, and further investigates how the selected approaches perform when a month worth of labeled training data is provided during monthly model updates. The selected interval, 1 month of model lifespan $vs.$ 1 month of labeled training dataset size, was chosen assuming that the network operator would have access to the label of all *MAWIFlow* events, an unrealistic assumption on production environments. However, such an evaluation serves as a baseline to determine how accurate a correctly updated ML-based NIDS can be, despite being unfeasible in production. Figures 1b and 1d shows the accuracy performance over time of the evaluated classifiers with monthly model updates. It is possible to note that when a proper amount of training data is provided, the classifiers are able to provide reasonable accuracy rates throughout the majority of *MAWIFlow* dataset. However, in some months, such as $Oct.$, $Nov.$ and $Dec.$, the accuracy rate is degraded. Noteworthy, in such a case, the model update task is not easily achieved, considering that more than 30 days of labeled data must be provided for a proper model building procedure to be

executed. Therefore, to address the network traffic behavior changes over time, ML-based approaches must be able to provide easiness on model updates, taking into account the number of needed training data and the computational costs of model retraining.

V. A Reminiscent Intrusion Detection Model Based on Deep Autoencoder and Transfer Learning

To address the challenge mentioned above of networked environments' non-stationary behavior, we propose a novel reminiscent intrusion detection model based on transfer learning on deep autoencoders. The proposed scheme aims at facilitating the model update task considering both computational costs and the amount of required labeled training data. To achieve such a goal, the proposed model (Figure 2), is implemented in two modules: *Classification* and *Easiness on Model Updates*.

The classification procedure starts with a to-be-classified network event collected from the monitored environment. The behavior of the collected data is then extracted by a feature extraction module, which outputs a set of behavior features compounding a feature vector. The feature vector comprises a set of values that represents the behavior of the analyzed event, which is used as input to a deep encoder layer (Figure 2, *Encoder*). The *encoder* outputs an additional set of behavioral features, which is used as input by a shallow classifier (e.g. GBT), that classifies it as either *normal* or *attack*. The insight of our proposal is that the encoder layers of a deep autoencoder can be used as an additional feature extraction module, which considers the historical behavior of network events. The rationale of our proposal is that the encoder, built through transfer learning, is able to leverage prior knowledge of network data. At model updates, the proposal leverages both *encoder* and *decoder* layers of the outdated deep autoencoder. Consequently, model updates are executed through a transfer learning scheme, decreasing computational costs and the amount of needed labeled training data.

The following subsections further describe our proposed model, including the components that implement it.

A. Classification

Traditional ML-based NIDS performs the event classification through the evaluation of flow-based features. Consequently, the training procedure of such proposals demands unfeasible amounts of labeled events be provided. This is because the used set of features does not consider their variations throughout time, according to the network traffic behavior changes.

In light of this, our proposal leverages the *encoder* layers of a deep autoencoder as an additional feature extraction stage. The rationale of such an approach is that the *encoder* layers, which are continuously updated through a transfer learning scheme, will be able to portray the meaningful relations in the feature values over time, hence, extracting meaningful features that can withstand more extended periods, despite the changes in the underlying network traffic. To make use of such an approach, our classification procedure (Figure 2), extracts the standard flow-based features from the collected network events and use them as input to the *encoder* layers that output the meaningful features established through a continuous training process through transfer learning. The *encoder* output is used as input to a shallow classifier (e.g. GBT), which assign a related event label, e.g. *normal* or *attack*.

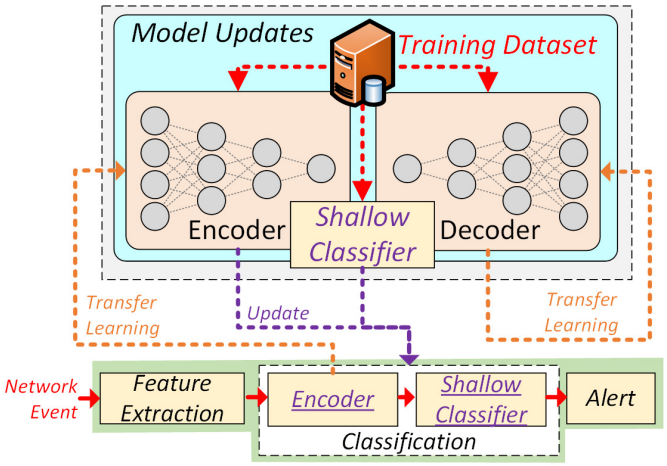
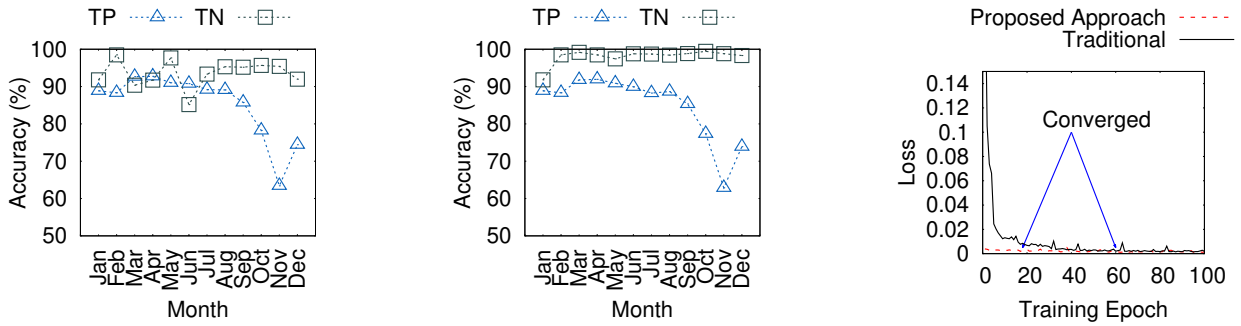


Fig. 2: Proposed reminiscent intrusion detection model through transfer learning on deep autoencoders.

B. Easiness on Model updates

Traditional model update approaches on ML-based NIDS discard the outdated model and builds a new one on the updated labeled training dataset. As a consequence, such proposals demands significant amounts of labeled data during model updates (e.g., Figs. 1c *vs* 1d) making them unfeasible to be properly used in production.

Our proposal relies on deep autoencoders and transfer learning at model updates (Figure 2, *Model Updates*), as an approach to decrease both computational costs, and the amount of needed labeled training data. On the one hand, the deep autoencoders are used as an additional feature extraction stage to depict the network traffic behavior changes over time. This is because as it is continuously updated through transfer learning, it can extract meaningful relations from the input feature set as time passes, according to changes in network traffic behavior over time. On the other hand, transfer learning is used to decrease computational costs during the model update and keep the historical behavior already present on the deep autoencoder layers. Our model can decrease the amount of needed labeled training data and computational costs at each model update. In summary, the model update task is performed as follows. The outdated *encoder* layers being used by the classification module is retrieved and coped with the stored *decoder* layers. The retrieved deep autoencoder is then updated through transfer learning, using a small time window of events, e.g., a week worth of data, considering that the historical network behavior is already available on the outdated deep autoencoder. After the deep autoencoder update task is performed, the *encoder* layers are used as input by a



(a) Model training at January with only 7 days, and no updates throughout time

(b) Model updates in a monthly periodicity, using only 7 days of data

(c) Comparison of training convergence at February

Fig. 3: Proposed approach performance on *MAWIFlow* dataset (Section IV-A).

shallow classifier (e.g., GBT), which is also updated, taking into account the newly deep autoencoder.

VI. Evaluation

This section further evaluates the proposed reminiscent intrusion detection model, by answering three main research questions: (***RQ3***) *How does the proposed approach perform without model updates and few labeled training data?* (***RQ4***) *Is the proposed approach able to perform model updates with fewer training instances?* (***RQ5***) *Does the proposed transfer learning scheme decrease computational costs?*

The proposed model (Figure 2) was implemented and evaluated on top of our used dataset (see Section IV-A). The deep autoencoder was implemented with the following architecture: ***Input***. The extracted 21 features are fed as input to the deep autoencoder. ***Encoder***. Three $dense$ layers implemented with a $relu$ activation function, with 512, 256, and 128 units respectively. ***Encoder Output***. A *encoder* output layer implemented with a $linear$ activation function, and 10 units. ***Decoder***. Three layers with a $relu$ activation function, with 128, 256, and 512 units respectively. ***Decoder Output***. A output layer with 21 units, and a $sigmoid$ activation function.

The implemented deep autoencoder architecture used the $rmse$ formula as $loss$ using $adam$ optimizer. For each model building procedure, either from scratch or at model updates, 1000 epochs are executed with a batch size of 512. The architecture was implemented through $keras$ API $v.2.4.0$, and $tensorflow$ API $v.2.4.1$. It is important to note that the used parameters were set similarly to related works, and no significant differences were found while varying them. The shallow classifier (Figure 2) was implemented through the Gradient Boosting (GBT) algorithm. Similarly as made in Section IV, it uses 0.1 learning rate value, with $deviance$ as the loss function, using 100 decision trees as its base-learner, this, implemented on top of $scikit-learn$ API v0.24. A random undersampling without replacement is used in the training procedure to balance the occurrence between classes.

A. Doing More With Less Data

To answer the question $RQ3$, we further investigate if our proposed model can provide high classification accuracies with less labeled training data, even when no model updates are performed. To achieve such a goal, in contrast to Section IV that uses 30 days of labeled training data, we perform the training procedure of our proposal with only 7 days of labeled events. Hence, our proposed scheme is trained with the data that occurred in *MAWIFlow* dataset from January 1st to 7th of 2016, while not being updated throughout time, evaluating the most extreme situation our proposal could face.

Figure 3a shows our proposed model classification accuracy over time with only 7 days of labeled training data and no updates being performed. Our proposed model provides similar classification accuracies, even when compared to traditional approaches that rely on 30 days of labeled training data and perform periodic model updates (Figure 1). For instance, if compared to its counterpart without deep autoencoder (Figure 1c), it improved the TN rate by 3.8% and the TP rate by 0.3%, with only $\approx$ 22% of labeled training data, while not being updated throughout time. Hence, with a deep autoencoder, the proposed scheme can provide highly accurate intrusion detection while demanding less training data.

To answer the question $RQ4$, we evaluate how periodic model updates affect our proposed scheme by performing monthly model updates with only 7 days worth of data. For instance, at February 1st, our proposed scheme is updated with the data that occurred between 25th to 31th of January. Recalling that our model update procedure leverages the outdated model through a transfer learning approach (Section V-B).

Figure 3b shows the classification accuracy of our proposal with monthly model updates. When model updates are performed, classification accuracy is improved, even with only a week-worth of labeled training data. The monthly model updates in our proposed model were able to improve the TN rate by 5.4% when compared to its no-update counterpart (Figure 3a). We also investigate how our proposed model performs when compared to traditional techniques with periodic model updates. Figure 4 shows the average error rate (avg. of $1-TP$ and $1-TN$) when monthly model updates are performed. Recalling that our proposed model uses only 7 days of training data, while both GBT and DT rely on 30 days of training data. Our proposed scheme can also decrease the average

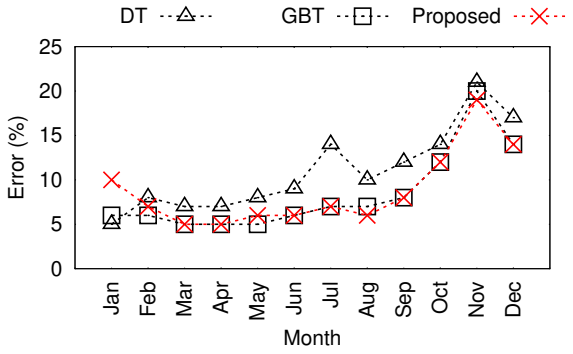


Fig. 4: The average error rate of evaluated approaches in *MAWIFlow* dataset with monthly model updates. Traditional techniques are updated with 30 days, while our proposal is updated with only 7 days of labeled data. We are able to provide similar results while demanding only $\approx$ 22% of labeled training data.

error rate by 3.4% when compared to the traditional DT. In addition, if compared to the GBT, but without our proposed deep autoencoder approach (Fig. 1d) our proposed approach increases the error rate by only 0.2%, while demanding only $\approx$ 22% of training instances.

Finally, to answer the question $RQ5$, we investigate how our proposed model can also decrease the computational costs during model updates by evaluating the number of epochs needed by our deep autoencoder during model training task, i.e., training convergence. Figure 3c shows the model convergence on model update at February 2016 on *MAWIFlow* dataset. It is possible to note that our proposed scheme that relies on transfer learning on deep autoencoders can converge at model updates with significantly fewer training epochs. In summary, our proposed model was able to decrease computational time at model updates by by an average of 28%, reduce the needed labeled data by up $\approx$ 22%, while improving TN rate by up 23.9% (Fig. 3b $vs.$ 1a).

VII. Conclusion

This work has challenged the literature assumption regarding how network traffic behavior changes affect ML-based NIDS, showing that it can significantly affect classification accuracy while demanding unfeasible amounts of labeled training data during model updates to be provided. In addition, we proposed a novel reminiscent intrusion detection model based on deep autoencoders and transfer learning to be used as a historical feature extraction stage. Our proposed scheme was able to provide significantly higher accuracy rates while demanding lower computational costs on model updates and fewer training instances to be provided during model updates.

Acknowledgment

This work was partially sponsored by Brazilian National Council for Scientific and Technological Development (CNPq), grant nº 315322/2018-7.

References

[1] B. Molina-Coronado, U. Mori, A. Mendiburu, and J. Miguel-Alonso, "Survey of network intrusion detection methods from the perspective of the knowledge discovery in databases process," *IEEE Trans. on Network and Service Management*, vol. 17, no. 4, pp. 2451–2479, Dec. 2020.

[2] C. Gates and C. Taylor, "Challenging the anomaly detection paradigm: A provocative discussion," in *Proc. of the Workshop on New Security Paradigms (NSPW)*, 2006, pp. 21–29.

[3] E. Viegas, A. Santin, A. Bessani, and N. Neves, "BigFlow: Real-time and reliable anomaly-based intrusion detection for high-speed networks," *Future Generation Computer Systems*, vol. 93, pp. 473–485, Apr. 2019.

[4] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *2010 IEEE Symposium on Security and Privacy*. IEEE, 2010.

[5] J. Zhang, F. Li, H. Wu, and F. Ye, "Autonomous model update scheme for deep learning based network traffic classifiers," in *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, Dec. 2019.

[6] E. Viegas, A. O. Santin, and V. A. Jr, "Machine learning intrusion detection in big data era: A multi-objective approach for longer model lifespans," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 1, pp. 366–376, Jan. 2021.

[7] R. L. Tomio, E. K. Viegas, A. O. Santin, and R. R. dos Santos, "A multi-view intrusion detection model for reliable and autonomous model updates," in *ICC 2021 - IEEE International Conference on Communications*. IEEE, Jun. 2021.

[8] R. Fontugne, P. Borgnat, P. Abry, and K. Fukuda, "MAWILab: Combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking," in *Proc. of the 6th Int. Conf. on emerging Networking EXperiments and Technologies (CoNEXT)*, 2010.

[9] E. K. Viegas, A. O. Santin, V. V. Cogo, and V. Abreu, "A reliable semi-supervised intrusion detection model: One year of network traffic anomalies," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*. IEEE, Jun. 2020.

[10] F. Ramos, E. Viegas, A. Santin, P. Horchulhack, R. R. dos Santos, and A. Espindola, "A machine learning model for detection of docker-based APP overbooking on kubernetes," in *ICC 2021 - IEEE International Conference on Communications*. IEEE, Jun. 2021.

[11] J. Mallmann, A. O. Santin, E. K. Viegas, R. R. dos Santos, and J. Geremias, "PPCensor: Architecture for real-time pornography detection in video streaming," *Future Generation Computer Systems*, vol. 112, pp. 945–955, Nov. 2020.

[12] X. Li, W. Chen, Q. Zhang, and L. Wu, "Building auto-encoder intrusion detection system based on random forest feature selection," *Computers & Security*, vol. 95, p. 101851, Aug. 2020.

[13] Y. Yan, L. Qi, J. Wang, Y. Lin, and L. Chen, "A network intrusion detection method based on stacked autoencoder and LSTM," in *ICC 2020 IEEE Int. Conf. on Communications (ICC)*. IEEE, Jun. 2020.

[14] Z. Cheng, E. Zhu, S. Wang, P. Zhang, and W. Li, "Unsupervised outlier detection via transformation invariant autoencoder," *IEEE Access*, vol. 9, pp. 43 991–44 002, 2021.

[15] A. Ahmim, L. Maglaras, M. A. Ferrag, M. Derdour, and H. Janicke, "A novel hierarchical intrusion detection system based on decision tree and rules-based models," in *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. IEEE, May 2019.

[16] J. Kevric, S. Jukic, and A. Subasi, "An effective combining classifier approach using tree algorithms for network intrusion detection," *Neural Computing and Applications*, vol. 28, no. S1, pp. 1051–1058, Jun. 2016.

[17] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 525–550, 2019.

[18] H. Yang and F. Wang, "Wireless network intrusion detection based on improved convolutional neural network," *IEEE Access*, vol. 7, pp. 64 366–64 374, 2019.

[19] MAWI, "MAWI Working Group Traffic Archive - Samplepoint F," 2021. [Online]. Available: https://mawi.wide.ad.jp/mawi/