A Deep Learning based Intrusion Detection System for 5G Networks

FILIPPO PEDRAZZINI

FILIPPOP@KTH.SE

DAVID TOMIC

DTOMIC@KTH.SE

KTH, Royal Institute of Technology

Contents

Abstract

With industry evolving, demand for a convergent high-speed mobile network (5G) is rising. In order to enable use-cases imagined for the future, a great amount of research has to be put into technological enablers of 5G. One such enabler is a virtualized intrusion detection system, ensuring both sustainability in the network development process, as well as network security. The paper at hand details the concept and software development behind a deep learning based intrusion detection system, along with respective evaluations.

1 Introduction

Mobile networks have transformed and advanced considerably over the past decades, due to an ever-evolving society and demand for high-speed services – which increase efficiency in both the business and everyday context. An unprecedented level of requirements is being posed onto such networks – all coming down to high expectations in terms of reliability, availability, efficiency and throughput. Given industrial evolvement and limitations to the existing technology, there is an increasing demand for advanced mobile networks, representing a firm foundation for all fields of cyber activity to come. This fact drives research among telecom operators and standardization organizations, which is to yield a convergent 5th generation mobile network. This shall be enabled by supporting technologies such as massive MIMO, NFV and SDN [1].

All enablers for 5G consider a highly virtualized environment, where network functions would be containerized and executed in the cloud. Supporting applications running on top of those containers would offer a plethora of network services. One such service is intrusion detection, which should ensure that all incoming traffic is properly inspected such as not to cause harm to the overall system [2, 3].

The paper at hand outlines research motivated by previous approaches to intrusion detection - traditional (rule based, anomaly detection) and artificial intelligence. Given that artificial intelligence methods have shown to detect intrusions more efficiently and at a higher accuracy than traditional approaches, it is evident that **further improvements should go into this direction** [4, 5].

As deep learning has shown to outperform machine learning based systems, there is every reason to employ it in intrusion detection as well.

With previous research already conducted in this field, it is at task to **find out** whether the use case of intrusion detection proves to better be handled by deep learning, **using stacked denoising auto-encoders** as a feature extraction mechanism.

Assuming success in implementing a well trained Deep Neural Network with stacked auto-encoders, **higher accuracy** in detecting intrusion is expected.

The paper contains 7 sections. Following a summary and background description, Section 2 gives a brief explanation on previous work in the field. Section 3 clearly states the research question and hypothesis. Section 4 discusses the research methodology (the dataset and implementation components). The results obtained are discussed and analyzed in Section 5. The paper concludes in Section 6 with suggestions on future work.

2 Theoretical framework/literature study

The task of building a network intrusion detection system has been approached over various algorithms and techniques. The ones considered here use the same open NSL-KDD dataset, to test the performance of the models and for result comparison with the scientific community. In [4] for example, an autoencoder was built in the first layer of the network to extract the most meaningful features and reduce the dimension of the data. After that, the processed input was fed into a standard artificial neural network (ANN) for classification. *Our research builds on the future work of this paper, utilizing both the training and testing NSL-KDD data. The model may be improved by focusing on the autoencoder, which can be brought to another level using a stacked autoencoder*.
Generally, most work in this field can be divided into two cases - using only training, or using both training and testing data.

2.1 Training data for both training and testing

The work found in [6] used a J48 decision tree classifier, and only the training set. The feature set was reduced to 22. Contrary to this, the approach in [7] used 2-level classification with the full feature set of 41. Using PCA for feature set reduction, and SVM for classification, a high level of accuracy was achieved. Reducing the feature set demonstrated higher accuracy, but lower performance.

2.2 Training data for training, testing data for testing

Various implementations have gone into using fuzzy classification [8], unsupervised clustering [9] or a k-point algorithm [10], demonstrating that accuracy is lower when utilizing both train and test data, than forming a training and testing dataset from the train data only.

3 Research questions, hypotheses

Having assumed that *Stacked Denoising Autoencoders* can be a feature extraction method which yields better intrusion detection classification, it is at task to implement a model either supporting or invalidating the premise.

4 Research Methodology

Due to the scope of the project, only secondary data was used throughout research. The implementation is based on existing algorithms, with results given in both raw

and plotted form. The quantitative approach taken is mainly due to lack of time and resources.

4.1 Dataset

The NSL-KDD dataset [11] served in both training and testing the network. To do so, it had to first undergo preparation, in order to ensure appropriate training data - and eventually accuracy. The reason for using this dataset is that it is a trace of intrusion traffic obtained in the Knowledge Discovery and Data Mining Tools Competition (with improvements) - thereby reflecting well what the developed ANN may be faced with in real-world environments. The dataset includes normal traffic, as well as one originating from different types of attacks. Contrary to the "old" KDD dataset, this one has been cleaned from redundant records, which might bias classification towards a specific type of attack.
Each record has 41 attributes concerning features of the flow, with an additional attribute designating the type of the record. Types can be subsumed under 5 classes [12]:

- Normal
- DoS - depletion of a victim's resources. Relevant features: "source bytes" and "percentage of packets with errors"
- Probing - gaining information about the victim. Relevant features: "duration of connection" and "source bytes"
- R2L - gaining root access at the victim. Relevant features: "number of file creations" and "number of shell prompts invoked"
- U2R - unauthorized access to a remote machine. Relevant features: "duration of connection", "service requested". "number of failed login attempts"

Out of 41 attributes, 3 are nominal, 4 are binary, and remaining 34 are continuous. Some attacks found in the test set are "novel", i.e. not known during training using the train set, increasing the classification difficulty by a considerable amount [13].

4.2 Stacked Denoising Auto-encoders

Autoencoders in general take raw data as input, pass it through a hidden layer, and then try to reconstruct it at the output. Minimizing the difference between "output" (predicted input) and input yields to output accuracy. This approach can point out significant data features, which can then be used to train a model. A

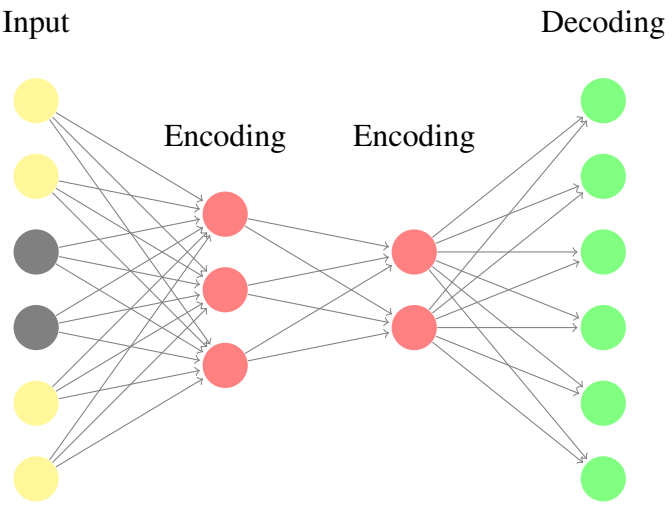

Figure 1: Stacked auto-encoder

stacked autoencoder contains several hidden layers. Noise can be added to the input before passing it to the hidden layers (see Figure 1) - minimizing the difference between the predicted and original input yields features with even higher significance [10]. The implementation made in this research minimizes the RMSE (Root Mean Square Error) loss function.
Examples of noise include:

- Salt and pepper noise - a fraction of the input elements (chosen at random for each sample) is set to either the minimum or maximum possible value
- Masking noise - a fraction of the input elements is set to 0 (i.e. artificially introducing blanks)

Denoising is only possible due to the dependencies between dimensions in a distribution.
The task of feature extraction was consequently handled by the stacked denoising auto-encoder, ensuring that only relevant features are considered for classification. The encoded input is passed into the soft-max regression network for classification. Although only two classes are considered at the output, scalability is ensured - given that the network supports N output classes. After training and tuning of the best parameters using cross validation, the final model is fed with testing data, and evaluated in terms of accuracy and f1.

5 Results and Analysis

5.1 Implementation

The development went through the following phases:

- Visualization and data preprocessing.
- Implementing a stacked denoising auto-encoder.
- Using a simple soft-max regression for the classification task.
- Finding the best parameters using *k-fold* cross-validation on the train set.
- Testing the classifier using the given test set.

As discussed above the dataset is composed of a train and test set, but in order to simulate a real environment the two samples are completely different from each other. This leads to a high accuracy using *k-fold* cross-validation (or hold out) on the train set, but poor results on the test set. For facing this problem, different approaches have been used - here focus was given to a particular deep method (stacked denoising auto-encoders) suggested in [4], where good results were obtained on the test set applying sparse auto-encoders for extracting the most meaningful features and using the encoded representation of the input as feed for the soft-max regression classifier.
A first visualization of the dataset characteristics was done finding out the already explained properties in Section 4. The dataset is balanced and for this reason there was no need to use any kind of over or under sampling method.
The second step was to transform the categorical features using *one-hot* encoding [5] and then scaling all of them using a standard scalar (removing mean and scaling to unit variance), which resulted in 122 overall features.
After the first preprocessing phase the soft-max regression algorithm was implemented. Using *k-fold* cross-validation an idea about the accuracy on the train set was obtained. As mentioned, even with simple methods it is easy to obtain high accuracy on the train set (hold out, *k-fold* cross-validation). For this reason no trust in the first results was given and the implementation of the feature extractor was continued.
The built network uses 3 fully connected layers with respectively 60, 30, 60 hidden units. The encoded input results in just 30 features, which are fed into the soft-max regression network for final classification. The learning is done separately; first the auto-encoder learns a representation of the input minimizing the difference between the decoded input and the original input - then using the encoded representation the soft-max regression network iterates the learning using

the input representation. The classifier in this case is simple, since a major part of the work is contained in the auto-encoder to learn a good and effective representation of the input. For this reason, no tuning of this network's parameters was done - the same parameters as in [4] were used (upon consultation with the authors). The stacked denoising auto-encoder parameters however were tuned using 5 fold cross-validation on 80% of the trainset. In this case the amount of parameters was not too large and a normal machine could handle computation. After finding the best parameters the model was tested on the test set, reaching a higher accuracy than [4] - due to a more robust feature extraction method.

5.2 Metrics

To certify the performance and robustness of the technique all the main classification metrics were used:

- Accuracy: correctly classified samples over the total number of samples.
- Precision (P): number of true positives (TP) samples divided by the number of true positives (TP) and false positives (FP) classified samples.

$$P = \frac{TP}{(TP + FP)} \tag{1}$$

- Recall (R): number of true positives samples divided by the number of true positives and false negatives (FN) classified samples.

$$R = \frac{TP}{(TP + FN)} \tag{2}$$

- F-Measure (F): harmonic mean of precision and recall and represents a balance between them.

$$F = \frac{2 \cdot P \cdot R}{(P + R)} \tag{3}$$

5.3 Performance Evaluation

As mentioned in the previous sections the algorithm was tested on the binary classification task of detecting normal and infected packets. The performance was measured using 5-fold cross-validation on the training data and then using the entire test set for the final accuracy. With 5-fold cross-validation on the train set the model gives high accuracy for all the metrics (95% - 98%). While training on the entire train and testing using the test set the performance was lower compared to

the previous values. All the metrics, in the best case, had around 88% - 92% of accuracy, which can anyway be considered as a good final result shown in Figure 2. The problem which must be spotted regards the variance in the results, with both sparse and stacked denoising auto-encoders we found out that the results on the test set can not be considered stable. Due to the fact that there are many random variables (e.g., noise fraction, weight initialization) and the learning is split in two phases, this brings to an high instability in the model. Regardless from this point of view, the results outreached the previous work [4] and gave a final good model.

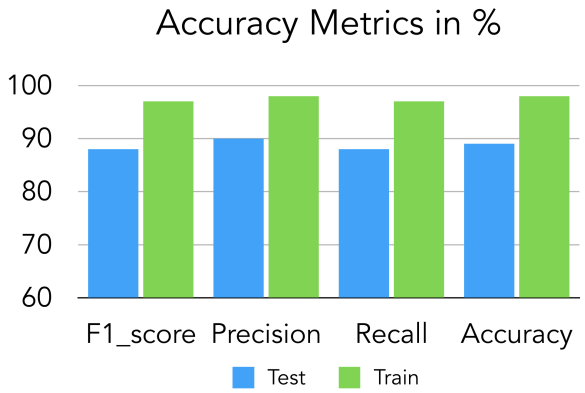

Figure 2: F-Measure, Precision, Recall, and Accuracy scores on both train and test data

6 Discussion

A more efficient and powerful mobile network shall not only bring about improvements in the business context, but also ensure greater environmental protection through novel technology, thereby ensuring sustainability. As explained above, crucial enablers of 5G are applications running on top of containerized network functions, among which intrusion detection represents one with top priority.
Considerable research is therefore going into this direction, including the one described in this paper.
After a thorough examination of previous work in this field, the most functional approach has been chosen to build upon. Given that deep neural networks have shown high accuracy in detecting intrusion, a classifier from this domain was developed and fed with well prepared data for training. The classifier chosen is an artificial neural network, with a stacked denoising auto-encoder in the first layers

for feature extraction. The encoded input is passed through the soft-max regression simple classifier to reach the wanted results.
Using this solution the previous results and performances reached using sparse auto-encoders [4] as feature extractor were outreached. The results are comparable and the difference is not significant.
As mentioned above the main problem in this model regards the variance in the results. To avoid this issue a further tuning of the parameters as well as a different optimizer can bring to a more stable solution regardless from the feature extraction technique (sparse [4] or stacked auto-encoders). Another solution could also be to change the classifier, opting for a more general and stable one (Random Forest).
Although traffic analysis might be associated with ethical considerations (as in "observing" data without user consent), it is a critical precondition for ensuring security.

References

[1] Ericsson, "5g systems - enabling digital transformation," *White Paper*, 2017.

[2] 5GPPP, "View on 5g architecture," *White Paper*, 2016.

[3] NGMN-Alliance, "5g white paper," *Next generation mobile networks, white paper*, 2015.

[4] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," in *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*, pp. 21–26, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2016.

[5] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," in *Wireless Networks and Mobile Communications (WINCOM), 2016 International Conference on*, pp. 258–263, IEEE, 2016.

[6] H.-s. Chae, B.-o. Jo, S.-H. Choi, and T.-k. Park, "Feature selection for intrusion detection using nsl-kdd," *Recent Advances in Computer Science*, pp. 184–187, 2013.

[7] F. E. Heba, A. Darwish, A. E. Hassanien, and A. Abraham, "Principle components analysis and support vector machine based intrusion detection system," in *Intelligent Systems Design and Applications (ISDA), 2010 10th International Conference on*, pp. 363–367, IEEE, 2010.

[8] I. Syarif, A. Prugel-Bennett, and G. Wills, "Unsupervised clustering approach for network anomaly detection," *Networked Digital Technologies*, pp. 135–145, 2012.

[9] P. Gogoi, M. H. Bhuyan, D. Bhattacharyya, and J. K. Kalita, "Packet and flow based network intrusion dataset," *Contemporary Computing*, pp. 322–334, 2012.

[10] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[11] "NSL-KDD | Datasets | Canadian Institute for Cybersecurity." `http://www.unb.ca/cic/datasets/nsl.html`. [Online; accessed 2017-12-03].

[12] L. Dhanabal and S. Shantharajah, "A study on nsl-kdd dataset for intrusion detection system based on classification algorithms," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, no. 6, pp. 446–452, 2015.

[13] S. Revathi and A. Malathi, "A detailed analysis on nsl-kdd dataset using various machine learning techniques for intrusion detection," 2013.