applied sciences

Article

A Neuro-Symbolic Classifier with Optimized Satisfiability for Monitoring Security Alerts in Network Traffic

Darian Onchis [1,†], Codruta Istin [2,*,†] and Eduard Hogea [1,†]

[1] Department of Computer Science, West University of Timisoara, 300223 Timisoara, Romania
[2] Department of Computer and Information Technology, Politehnica University Timisoara, 300223 Timisoara, Romania
* Correspondence: codruta.istin@upt.ro
† These authors contributed equally to this work.

Abstract: We introduce in this paper a neuro-symbolic predictive model based on Logic Tensor Networks, capable of discriminating and at the same time of explaining the bad connections, called alerts or attacks, and the normal connections. The proposed classifier incorporates both the ability of deep neural networks to improve on their own through learning from experience and the interpretability of the results provided by the symbolic artificial intelligence approach. Compared to other existing solutions, we advance in the discovery of potential security breaches from a cognitive perspective. By introducing the reasoning in the model, our aim is to further reduce the human staff needed to deal with the cyber-threat hunting problem. To justify the need for shifting towards hybrid systems for this task, the design, the implementation, and the comparison of the dense neural network and the neuro-symbolic model is performed in detail. While in terms of standard accuracy, both models demonstrated similar precision, we further introduced for our model the concept of interactive accuracy as a way of querying the model results at any time coupled with deductive reasoning over data. By applying our model on the CIC-IDS2017 dataset, we reached an accuracy of 0.95, with levels of satisfiability around 0.85. Other advantages such as overfitting mitigation and scalability issues are also presented.

Keywords: neuro-symbolic model; IT alerts; cognitive threat hunting; classifier; Logic Tensor Networks

check for updates

Citation: Onchis, D.; Istin, C.; Hogea, E. A Neuro-Symbolic Classifier with Optimized Satisfiability for Monitoring Security Alerts in Network Traffic. *Appl. Sci.* **2022**, *12*, 11502. https://doi.org/10.3390/app122211502

Academic Editors: Sangkyun Lee and Yunheung Paek

Received: 24 October 2022
Accepted: 8 November 2022
Published: 12 November 2022

1. Introduction

An Intrusion Detection System (IDS) is able to monitor suspicious and malicious activity through network traffic and to generate alerts when irregularities are detected. While there exists systems based on signature detection [1,2] and hybrid detection [3,4], the focus of this work is on constructing an anomaly detection model based on a self-explainable deep learning model. In this approach, all future behavior is compared to this model, and any anomalies are labeled as potential threats and generate alerts.

Two main ways of constructing such a model are either by using a modern deep learning [5,6] network with a post hoc explainable artificial model on top, such as, e.g., Local interpretable model-agnostic explanations (LIME) [6–8] or by using symbolic reasoning as in symbolic AI [9]. While deep learning can recognize patterns and is very well capable of learning on its own through experience with a suitable set of data, it lacks model interpretability. On the other hand, symbolic reasoning does not have any of those problems, but the rules by which the algorithm establishes connections between the inputs and outputs needed to be hard-coded by humans. A large set of rules may be an ideal method that covers all possible use cases, but its number is directly proportional to the computing power needed.

As a solution to incorporate data and logic and to overcome these deficiencies, Logic Tensor Networks (LTN) [10] use a neuro-symbolic way to combine neural networks and first-order logic language with fuzzy logic. This framework provides reasoning over data, and in our model it is used to construct a multi-class classifier. LTNs are used to reduce the learning problem of a given formula by optimizing its satisfiability. The network will try to optimize the groundings to bring the truth of the formula close to 1. Our framework will be used to contribute to the emergent development of cognitive threat hunting [11]. This new class of algorithms will mimic the role of threat hunters leading to high fidelity detection, economies of scale and the overall replacement of human expert hunters. According to a recent study conducted by ATOS [11], a global leader in digital transformation and cyber-security company, a rapid evolution is expected in the field, by extending with reasoning the previous deep learning solutions, with early operational implementations to start around 2023 and with the number of use cases growing rapidly by 2026.

Related Work

Using only deep learning techniques, there have been many recent architectures proposed in the scientific literature for the constructions of intrusion detection systems. In the work [12], a convolutional deep learning model with regularized multi-layer perceptron was proposed, instead of a fully connected feed-forward neural network for a network intrusion detection system. Even though the results presented were promising, the authors themselves recognized that there is room for improvements and they proposed feature reduction methods and transfer learning. The need for secure operational conditions was observed also in fault detection [13] or recognition systems [14].

In the systematic review on deep learning-based IDS from [15] and in the surveys from [16,17], various deep learning techniques are presented such as deep autoencoders, recurrent neural networks, Boltzmann machines, deep belief networks or other types of convolutional neural networks. The overall conclusion was that providing an efficient and high-performance IDS approach to deal with a wide variety of security attacks by using only deep learning is very challenging.

The paper from [18], proposes a hybrid deep learning model for cyber attack detection in Internet of Vehicles. The proposed model is based on long short-term memory (LSTM) and the gated recurrent unit (GRU) but without incorporating logic and reasoning, as in our model.

Other authors have already recognized the potential of hybrid systems for predictive neural models such as [19,20]. While in the latter one, the term "hybrid system" is not directly used in there, the authors explain how it is crucial for deep learning models to become able to make connections and associations in raw data, and process them in a way suitable for further use, and how doing that is possible by adding the benefits of Symbolic AI to Deep Learning models. Both papers also demonstrate the advancements of neuro-symbolic AI.

Another related work providing a comparison study between multiple algorithms is [21]. The authors also include an explanation of how the algorithms work on a specific dataset (in their case MNIST), then as a part of experiments; in the end, they compare various results. Other prospective methods could detect the alerts pattern based on a homological spanning forest framework [22] or frame methods [23].

Given the related works context, our work is positioned as a pioneering technology approach that is able to deal with the emergent cognitive threat hunting challenge.

2. Materials and Methods

The basis of Logic Tensor Networks (LTN) is called real logic, which is described by a first-order language that contains constants, variables, functions, predicates, and logical connectives and quantifiers [24]. The way LTN uses real logic is by transforming the provided formulas into TensorFlow computational graphs. Those formulas can then be used for querying, learning, and reasoning over the data.

Grounding in LTN [24]. Constants are depicted as tensors that can be of any rank. Tensors can be represented as n-dimensional arrays, as shown below. Let the following expression (1):

$$\bigcup_{i_1,i_2,\dots,i_k\in\mathbb{N}^*} \mathbb{R}^{i_1\times i_2\times\dots\times i_k} \tag{1}$$

denote the set of tensors of any rank. The tensor of rank 0 represents a scalar value, a rank 1 tensor represents a one-dimensional vector, a rank 2 one represents a matrix, and so forth.

For the constant α, let us refer to the associated tensor by $\mathbb{G}(\alpha)$. As an example shown in (2), consider the constants x, y, z and the associated tensors $\mathbb{G}(x), \mathbb{G}(y), \mathbb{G}(z)$ of different ranks.

$$\mathbb{G}(x) = \begin{pmatrix} 1 & 2 \end{pmatrix}, \mathbb{G}(y) = \begin{pmatrix} 1 & 2 \\ 6 & 7 \end{pmatrix}, \mathbb{G}(z) = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \tag{2}$$

Predicates in LTN are mapped to functions or tensor operations and return a value in [0, 1] that denotes a satisfaction level. It can be interpreted as a truth degree. A predicate can be a simple λ function or a neural network of any shape and activation function (the only exception would be on the last layer, where a sigmoid function should be used to return a value of partial truth). Connectives implemented in LTN are modeled to the semantics of first-order fuzzy logic [25]. Some of these are negation, implication, conjunction, and disjunction. Quantifiers supported in LTN are $\forall$ (universal) and $\exists$ (existential) and are defined as aggregation functions. In the following experiments, only the $\forall$ p-mean error aggregator is used. A parameter p used in (3)

$$X(a_1, a_2, a_3, \dots, a_{k-1}, a_k) = 1 - \left(\frac{1}{k}\sum_{i=1}^{k}(1-a_i)^p\right)^{\frac{1}{p}} \tag{3}$$

adjusts the focus on the deviation, and it is especially useful in querying where the focus is on the formulas, contrary to training, where a lower value is used to prevent overfitting and make it possible to generalize. For example, a $p = 2$ would result in an aggregation function that outputs the standard deviation of the inputs and a $p = \infty$ would output the minimum value.

3. Results Overview

The main contributions of this work are the proposed neuro-symbolic model with optimized satisfiability for monitoring security alerts in network traffic, the detailed explanations of the implementation and the experimental results that highlight some aspects that should be taken into consideration by anyone who wants to build such a system. The introduction of the novel interactive accuracy alongside with the standard accuracy unleash the potential of hybrid systems and put forward their usage in real-world scenarios such as the KDD99 and CIC-IDS2017 datasets, to observe and to interpret the results.

The global solution could be synthesized in the generic Algorithm 1 below:

Algorithm 1 The generic hybrid solution

```
1: Load the dataset. Split in training set (X) and testing set (Y).
2: Batch the data into $X_{tensor}, Y_{tensor}$.
3: Define predicates $\mathcal{P}(x1, x2)$, constants $\mathcal{G}(c)$, variables $x1, x2$ and mapping functions $f(X, Y)$.
4: Define the Knowledge Base (KB). Add constraints if needed.
5: Train the model and maximize the satisfiability over rules $Max(sat_{rule1}, sat_{rule2} \dots)$.
6: Query the satisfiability level $\mathcal{G}(\phi_{formula})$ of any defined formula.
7: Interpret the results.
```

4. Discussions Alongside Numerical Experiments

Based on the generic Algorithm 1, the following hybrid systems classifiers have been designed and implemented:

4.1. Classifier Based on Protocol Type and Status of the Connection

The dataset we used is KDD-Cup'99, also known as KDD99. It consists of 41 features and 494,020 records and contains a wide variety of intrusions in network traffic. We used multi-label classification and LTN (Logic Tensor Networks) to classify our records according to the protocol used and the status of the connection. In this first example, real logic is used to ask complex queries to interpret the data. To avoid possible causes of errors based on the huge number of replicated records, we eliminated them and the number dropped from 494,020 to 145,585 records.

Our protocol types and connection statuses used to write the grounding in real logic are displayed in the Table 1 below:

Table 1. Protocol types and connection statuses used to create our grounding. The Knowledge Base will contain a set of rules based on these.

0. TCP	**1. ICMP**	**2. UDP**	**3. SF**	**4. S1**
5. REJ	6. S2	7. S0	8. S3	9. RSTO
10. RSTR	11. RSTOS0	12. OTH	13. SH	

Our input layer of the network has the same size as the number of classes mentioned above. The set of axioms in this example will include all the protocol types and connection statuses, but with no constraints. We split the samples into training and testing, 120,000 for training and the rest for testing.

The set of axioms in this case:

$$\begin{array}{cc}
\forall x_{tcp} : P(x_{tcp}, tcp) & \forall x_{s0} : P(x_{s0}, s0) \\
\forall x_{icmp} : P(x_{icmp}, icmp) & \forall x_{s3} : P(x_{s3}, s3) \\
\forall x_{udp} : P(x_{udp}, udp) & \forall x_{rsto} : P(x_{rsto}, rsto) \\
\forall x_{sf} : P(x_{sf}, sf) & \forall x_{rstr} : P(x_{rstr}, rstr) \\
\forall x_{s1} : P(x_{s1}, s1) & \forall x_{rstos0} : P(x_{rstos0}, rstos0) \\
\forall x_{rej} : P(x_{rej}, rej) & \forall x_{oth} : P(x_{oth}, oth) \\
\forall x_{s2} : P(x_{s2}, s2) & \forall x_{oth} : P(x_{oth}, oth)
\end{array}$$

Training the model, we define some metrics. We measure:

1. The level of satisfiability of the Knowledge Base of the training data.
2. The level of satisfiability of the Knowledge Base of the test data.
3. The training accuracy (we used Hamming loss function).
4. The test accuracy (same thing, but for the test samples).
5. The level of satisfiability of a formula $\phi 1$. $\forall x : p(x, udp) \rightarrow \neg p(x, tcp)$ (every udp connection is not a tcp connection and vice-versa)
6. The level of satisfiability of a formula $\phi 2$. $\forall x : p(x, udp) \rightarrow \neg p(x, icmp)$ (every udp connection is not a $icmp$ one and vice-versa)
7. The level of satisfiability of a formula $\phi 3$. $\forall x : p(x, tcp) \rightarrow \neg p(x, icmp)$ (every tcp connection is not a $icmp$ one and vice-versa)

Looking at our $results.csv$ file that has all the metrics enumerated, we can see $\phi 1$ and $\phi 3$ have high satisfaction levels, almost perfect ones. This means that, every udp connection cannot be a tcp connection and every tcp connection cannot be a $icmp$ one. For $\phi 2$, we reached a satisfaction level of 0.10, denoting that between udp and $icmp$ connections have features that are highly correlated.

4.2. LTN-Based Intrusions Detection in Network Traffic

Our second example aims to distinguish between normal connections and intrusion attacks on the same dataset. The ideal case for an intrusion detection model is to detect the exact type of attack, but as it can be observed in Figure 1, where a logarithmic scale was needed, the data for some attacks is scarce.

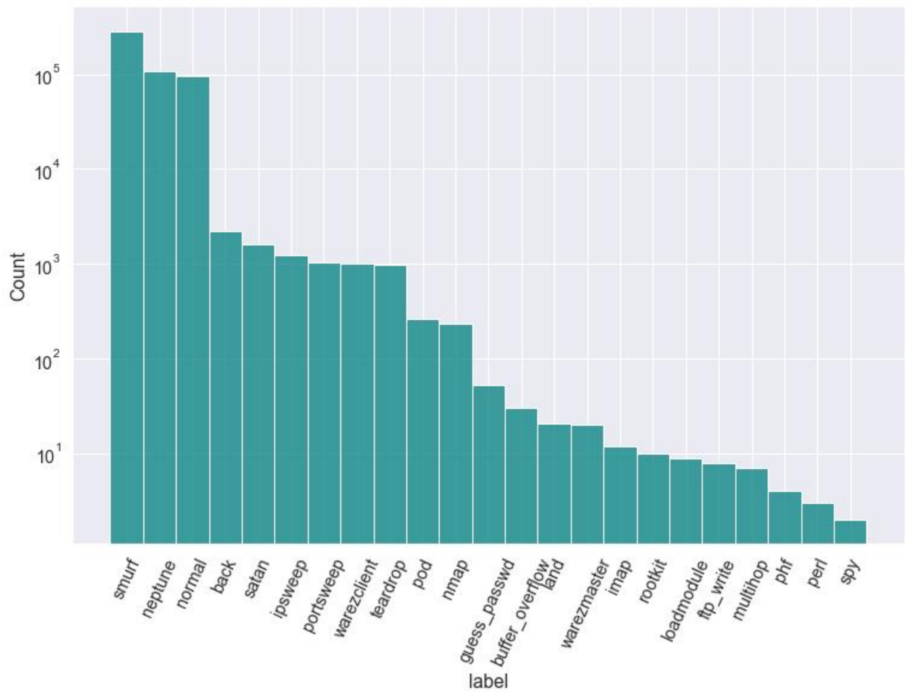

Figure 1. Distribution of connection types before removal of duplicates

It is helpful to note that in our dataset, each connection has a feature called *label* that depicts the type. With the help of a Python Data Analysis Library called Pandas [26] and some background knowledge provided in [27], we know that all attack types fall into the category of DOS, R2L, U2R, or probe. As such, to help with the scarcity of some attacks and make a good prediction possible, we grouped all the intrusions by the category that they belong to. With **DOS (denial of service)** attack types, a huge amount of repeated server requests is sent with the malicious intent of using the victim resources, blocking any possible connections, or even forcing a shutdown/reboot. "back", "land", "neptune", "pod", "smurf" and "teardrop" are attacks from KDD99 dataset that fall into this category. **R2L** are attacks aimed to gather local data from another computer or server. The way this is conducted may be different, but the goal is the same. Attacks can try to query the computer/server, "guess" the password via social engineering, or create local Trojan files to log any activity. In our dataset, "ftp_write", "imap", "multihop", "phf", "spy", "warezclient" and "warezmaster" are R2L attacks. **U2R** attacks happen when the attacker has access to another computer/user from the targeted network. It is meant to gain control of the root. This is possible by exploiting weaknesses in software and while it can be avoided, a complex software is prone to fall victim to this attack. In KDD99, "buffer_overflow", "loadmodule", "perl" and "rootkit" are attacks that target the root. **Probe** attacks are different from the ones mentioned before. Some of these attacks, such as "ipsweep", "nmap", "portsweep" and "satan" search for any possible weakness in the targeted computer. Usually in the form of connected IPs or open ports.

A very helpful analysis of the KDD99 dataset and the impact of the features in intrusion detection can be found in [28]. Taking a look at that analysis and our exploration of the dataset, connections normal, smurf and neptune represent the vast majority and after grouping them, DOS and normal types are predominant.

It is to be noted that most DOS attacks work by sending a large volume of traffic to the network system. This attack category is without a doubt the most popular one, with smurf attack type being by far the most predominant one in this category in the original dataset, see Figure 2. After removal of duplicates, with the intent of reducing the size of the dataset and improving its quality, that is no longer true. The results after the removal of duplicates can be observed in Figure 3.

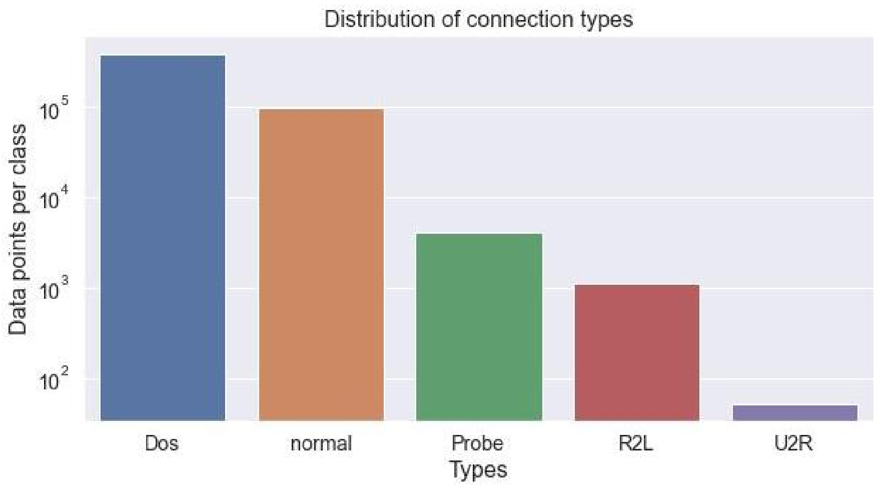

Figure 2. Distribution of data before removal of duplicates.

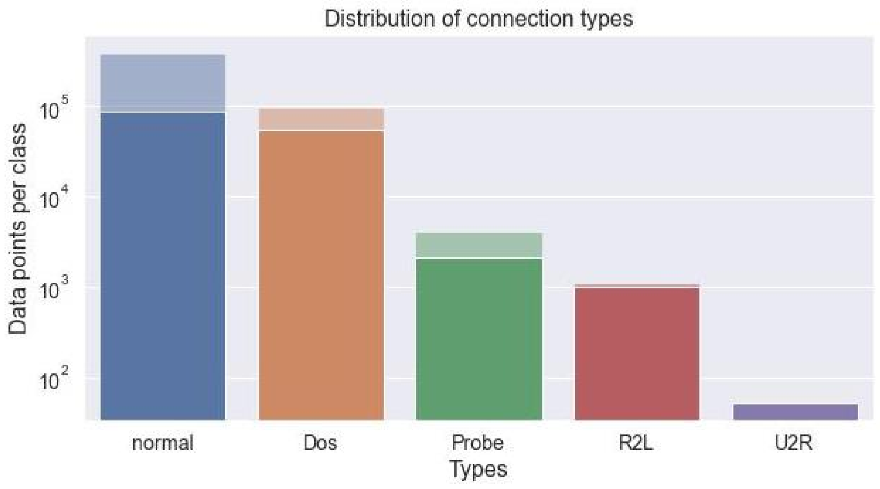

Figure 3. Data distribution after removal of duplicates. The transparent part in each type of connection represents the removed data after cleaning.

To classify the intrusions correctly, we normalized the data, one-hot encoded the categorical values, and kept only the relevant features. This feature relevance shapes the logic part of our neuro-symbolic framework and makes it possible to use real logic to detect possible attacks. The relevant features are used to split the dataset into a training set and a testing set, which are later used for training the model. As described, our goal with this exampled is to be able to use LTN to distinguish between a normal connection and an intrusion while also categorizing the latter.

For that, logic needs to be defined and training of the model is necessary. Each of the possible groups of connections is represented as a label and is used to define the axioms. To cover all the possible scenarios and add some constraints to make overfitting less likely, we used the following set of real logic axioms.

$$\forall x_{normal} : P(x_{normal}, normal)$$
$$\forall x_{DOS} : P(x_{DOS}, DOS)$$
$$\forall x_{probe} : P(x_{probe}, probe)$$
$$\forall x_{R2L} : P(x_{R2L}, R2L)$$
$$\forall x_{U2R} : P(x_{U2R}, U2R)$$
$$\forall x : \neg(P(x, normal) \wedge P(x, DOS))$$
$$\forall x : \neg(P(x, normal) \wedge P(x, probe))$$
$$\forall x : \neg(P(x, normal) \wedge P(x, R2L))$$
$$\forall x : \neg(P(x, normal) \wedge P(x, U2R))$$
$$\forall x : \neg(P(x, DOS) \wedge P(x, probe))$$
$$\forall x : \neg(P(x, DOS) \wedge P(x, R2L))$$
$$\forall x : \neg(P(x, DOS) \wedge P(x, U2R))$$
$$\forall x : \neg(P(x, R2L) \wedge P(x, probe))$$
$$\forall x : \neg(P(x, R2L) \wedge P(x, U2R))$$
$$\forall x : \neg(P(x, probe) \wedge P(x, U2R))$$

An explanation for the proposed set of axioms is needed. In top-down order:

- All the non-attacks should have label normal;
- All the *DOS* attacks should have label *DOS*;
- All the probe attacks should have label probe;
- All the *R2L* should have label *R2L*;
- All the *U2R* attacks should have label *U2R*;
- If an example x is labelled as normal, it cannot be labelled as *DOS* too;
- If an example x is labelled as normal, it cannot be labelled as probe too;
- If an example x is labelled as normal, it cannot be labelled as *R2L* too;
- and so forth...

Based only on the defined logic, we have calculated the initial satisfiability level to be 0.59471. We concluded that training the model to detect intrusions is crucial. To make a comparison possible, most of the metrics used in training are similar, except for the 3ϕ formulas.

1. The level of satisfiability of the Knowledge Base of the training data.
2. The level of satisfiability of the Knowledge Base of the test data.
3. The training accuracy (calculated as the fraction of the labels that are correctly predicted).
4. The test accuracy (same thing, but for the test samples).
5. The level of satisfiability of a formula we expect to be true. $\forall x : p(x, normal) \rightarrow \neg p(x, DOS)$ (every *normal* connection cannot be a *DOS* connection and vice-versa)
6. The level of satisfiability of a formula we expect to be false. $\forall x : p(x, normal) \rightarrow p(x, probe)$ (every *normal* connection is also a *probe* one)
7. The level of satisfiability of a formula we expect to be false. $\forall x : p(x, normal) \rightarrow p(x, normal)$ (every $smurf$ connection has a *normal* connection status)

The training metrics are stored in a .csv file that is also represented in Figure 4 and they can be interpreted as follows. In the first epoch, the level of satisfiability of the Knowledge Base increased from the initial value of 0.54725 to 0.6759 for the training data and 0.7216 for the test data. The results in the beginning of the training may seem atypical, with a higher accuracy for the test data, but it can be explained by the non-uniform intrusion distribution. Satisfiability level in a neuro-symbolic models is similar to a loss function from deep learning and its evolution can be conceptualized as optimizing the model under relaxed first-order logical constraints. This level is introduced in this paper under the name interactive accuracy to keep the terminology as similar and comparable as possible. As the level of satisfiability for our Knowledge Base increased, the standard accuracy also improved, with values ranging from 0.7757 for our train data and 0.9162 for our test data in the first epoch and up to 0.9954 and 0.9947, respectively, in the last epoch of training. Comparably, the accuracy for the dense deep neural network was 0.98863 for train data and 0.9869 for test data.

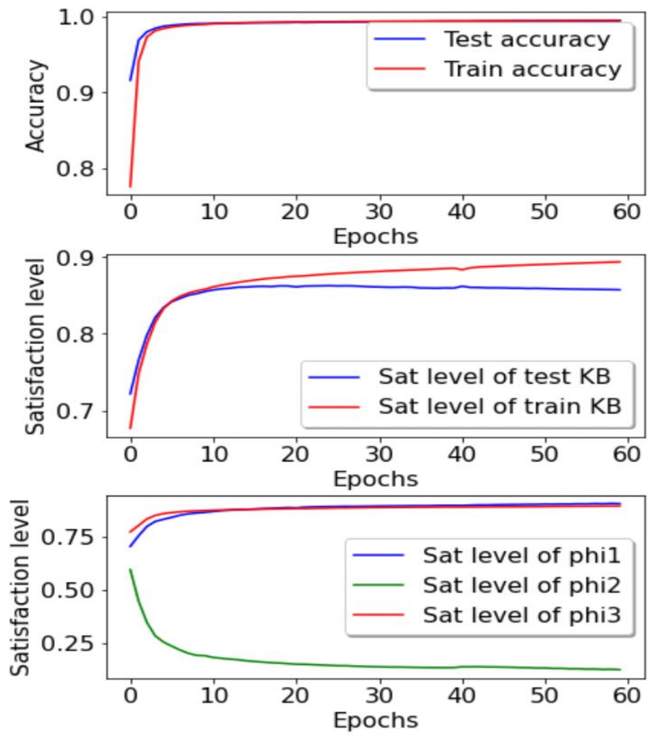

Figure 4. Comparison of accuracy, satisfiability and querying of constraints.

One of the most important aspects of the proposed model is that Symbolic reasoning makes it possible for humans to understand how the model gave a specific output. One can query the satisfaction level of any defined formula and the results or query the constraints while also measuring the accuracy. The satisfaction level is depicted with a fuzzy logic with values in [0, 1], as opposed to the usual boolean value. The network uses the maximization of the satisfaction rules as an objective in learning by optimizing the groundings. Figure 4 shows it is possible to have an accuracy metric (number of correct predictions divided by total number of predictions), specific to deep learning but also have an interactive accuracy consisting of both the provided formulas and the queried constraints. Spikes in first two graphs also make sense, as an incorrect prediction leads to a drop in the truth values of the axioms. While these are related, the accuracy level is almost perfect while the satisfaction level can see some improvements. A higher satisfiability level means a better performance of the model. The hybrid network can also generalize better and use learned knowledge when querying constraints with impressive results. This way overfitting can be avoided by providing accurate logic constraints in axioms. On the other side, some tasks may need extensive logic defined and the large number of axioms affects the computational power, therefore the scalability of such a hybrid system could be an issue.

Deep Neural Network approach for KDD99. The same experiment presented before is conducted again, without the symbolic part. As such, any kind of reasoning over data is conducted only with the accuracy measurement, making deductive reasoning, interactive accuracy, and the ability to add constraints via real logic impossible. For the sake of consistency, in this experiment, the following aspects were kept the same as the ones in the LTN implementation. Data normalization, one-hot encoding of the categorical data, hidden layers number, and the number of neurons on each hidden layer have all not been altered. Notable differences between these two approaches are in the batching of the dataset, for LTN data was first stored in tensors and then batched; in the output layer, for the LTN we used a $sigmoid$ activation function and for DNN we used a $softmax$ one and lastly, the hidden layers in the neuro-symbolic case have elu as an activation function as opposed to $relu$ in the other case.

As Figure 5 shows, the results of the intrusion detection classifier implemented with a Deep Neural Network are very good. With accuracy values reaching almost perfect values, 0.98863 for both train data and 0.9869 for validation data.

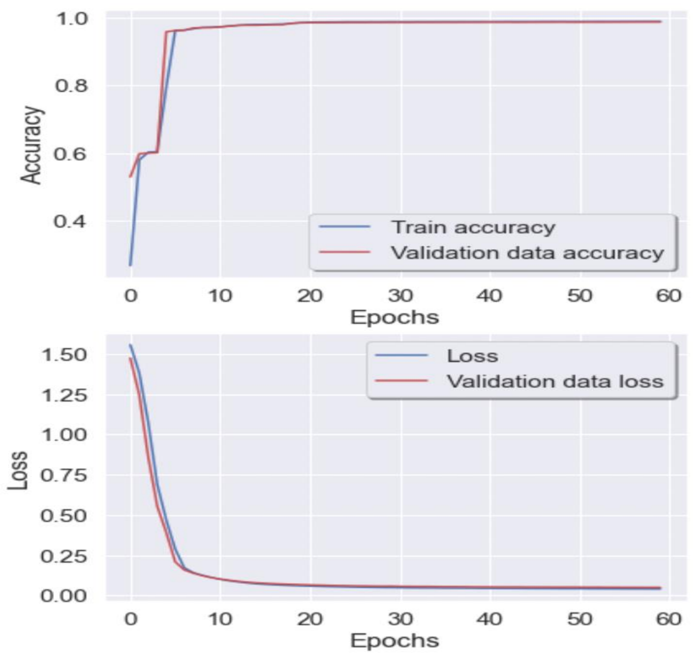

Figure 5. Evolution of loss and accuracy in training.

4.3. LTN and DNN on CIC-IDS2017

While KDD99 is a well-known and extensively used dataset for intrusion detection, it is nowadays fairly old and has received multiple critics regarding its inconsistency and the false representation of a modern real-world scenario [29–31]. To demonstrate the capabilities of LTN, we continued our experiments with a multiclass classification but, in this last case, made it single-label instead of multi-label. CIC-IDS2017 dataset aims to solve all the previous faults of intrusion detection datasets and provide some up-to-date attacks and a reshaped network traffic analysis. A comprehensive explanation of how the dataset was created, all the used attacks, and the feature importance for each one of them is offered in the related paper [32].

This IDS Evaluation Dataset offers the labeled network flow for actual weekdays days when attacks happened and in this experiment, we decided to see how well the attacks that happened on Friday, 7 July afternoon, can be predicted using LTN. The connection types, in this case, were either DDoS, PortScan, or BENIGN (normal connection). A quick exploration after eliminating the redundancy in the dataset shows that there are 212,718 BENIGN, 128,005 DDoS, and 57,305 PortScan connections.

For the neuro-symbolic model, data were batched into tensors and a set of axioms was implemented. The knowledge base in this scenario is simpler; we defined only that all the DDoS attacks should have a "DDoS" label and all BENIGN connections should have a "BENIGN" one and PortScan should have a "PortScan" one. There was no need for constraints since the number of classes was small. The function grounded in LTN semantics was kept the same (a neural network with no change in the shape of hidden layers) and the metrics measured were also kept, with the removal of ϕ queries.

As shown in Figure 6, there is a clear evolution in the accuracy of our LTN model, but there is also room for improvement. We reached an accuracy of 0.950, with levels of satisfiability being very close. Both graphs are very similar, as is expected with a simpler knowledge base as ours.

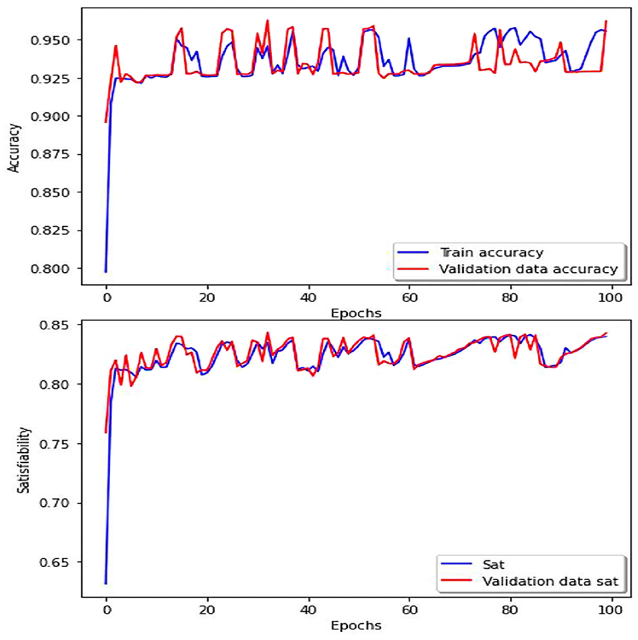

Figure 6. Evolution of satisfiability and accuracy in training LTN model.

For the DNN part, the network was implemented almost identically to the one used in the LTN function before. The results can be observed in Figure 7, and it should be remarked that while the training accuracy is steadily improving, the accuracy for the validation data still has some drops in performance. As a direct comparison, the LTN model provides more constant results and is better at generalizing to new data with the help of the defined Knowledge Base.

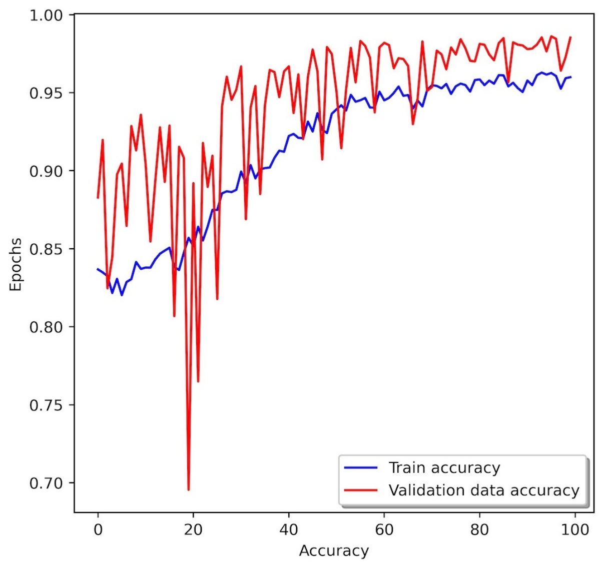

Figure 7. Evolution of accuracy in training DNN model.

5. Conclusions

The neuro-symbolic model proposed in this paper was constructed using LTN networks with custom-designed axioms for developing specific IDS models. The experiments carried out for the KDD99 and CICIDS2017 datasets gave us superior results in terms of reasoning, which makes it possible for humans to understand how the model gave a specific output. One can query the satisfaction level of any defined formula and the results, and query the constraints while also measuring the accuracy. The obtained results have been compared in detail to the ones provided by well-known classifiers such as Deep Neural Networks in the same training and testing conditions. The addition of symbolic reasoning can also help to avoid overfitting, can speed up the training time and therefore can only improve the quality of a possible IDS based on hybrid systems. Overall, we can conclude that advancements in the neuro-symbolic field provide suitable models to address the emergent cognitive threat hunting challenge.

Author Contributions: Conceptualization, D.O. and E.H.; methodology, D.O.; software, E.H.; validation, D.O., C.I. and E.H.; formal analysis, C.I.; investigation, D.O.; resources, E.H.; data curation, E.H.; writing—original draft preparation, E.H. and D.O.; writing—review and editing, E.H. and D.O.; supervision, D.O. All authors have read and agreed to the published version of the manuscript.

Funding: The second author was supported by the project AISIMIA, contract No. 10163/11.06.2021, at the Politehnica University of Timisoara.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not aplicable.

Data Availability Statement: Available online: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html (accessed on 1 May 2021); Available online: https://www.unb.ca/cic/datasets/ids-2017.html (accessed on 15 February 2022).

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Ioulianou, P.; Vasilakis, V.; Moscholios, I.; Logothetis, M. A signature-based intrusion detection system for the internet of things. *Inf. Commun. Technol. Form* 2018, *in press*.
2. Kumar, V.; Sangwan, O.P. Signature based intrusion detection system using SNORT. *Int. J. Comput. Appl. Inf. Technol.* **2012**, *1*, 35–41.
3. Aydın, M.A.; Zaim, A.H.; Ceylan, K.G. A hybrid intrusion detection system design for computer network security. *Comput. Electr. Eng.* **2009**, *35*, 517–526. [CrossRef]
4. Smys, S.; Basar, A.; Wang, H. Hybrid intrusion detection system for internet of things (IoT). *J. ISMAC* **2020**, *2*, 190–199. [CrossRef]
5. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef]
6. Schmidhuber, J. Deep learning in neural networks: An overview. *Neural Netw.* **2015**, *61*, 85–117. [CrossRef]
7. Molnar, C. *Interpretable Machine Learning*; Independently Published: Chicago, IL, USA, 2022; ISBN 979-8411463330.
8. Onchis, D.M.; Gillich, G.R. Stable and explainable deep learning damage prediction for prismatic cantilever steel beam. *Comput. Ind.* **2021**, *125*, 103359. [CrossRef]
9. Haugeland, J. *Artificial Intelligence: The Very Idea*; MIT Press: Cambridge, MA, USA, 1989.
10. Badreddine, S.; d'Avila Garcez, A.; Serafini, L.; Spranger, M. Logic Tensor Networks. *Artif. Intell.* **2022**, *303*, 103649. [CrossRef]
11. ATOS Scientific Community Report. 2022. Available online: https://atos.net/content/journey/unlocking-virtual-dimensions-atos-scientific-community-report.pdf (accessed on 23 October 2022).
12. Ashiku, L.; Dagli, C. Network Intrusion Detection System using Deep Learning. *Procedia Comput. Sci.* **2021**, *185*, 239–247. [CrossRef]
13. Onchis, D. Observing damaged beams through their time–frequency extended signatures. *Signal Process.* **2014**, *96*, 16–20. [CrossRef]
14. Mihail, G.; Onchis, D. Face and marker detection using Gabor frames on GPUs. *Signal Process.* **2014**, *96*, 90–93. [CrossRef]
15. Lansky, J.; Ali, S.; Mohammadi, M.; Majeed, M.K.; Karim, S.H.T.; Rashidi, S.; Hosseinzadeh, M.; Rahmani, A.M. Deep Learning-Based Intrusion Detection Systems: A Systematic Review. *IEEE Access* **2021**, *9*, 101574–101599. [CrossRef]
16. Karatas, G.; Demir, O.; Koray Sahingoz, O. Deep Learning in Intrusion Detection Systems. In Proceedings of the 2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT), Ankara, Turkey, 3–4 December 2018; pp. 113–116. [CrossRef]
17. Berman, D.S.; Buczak, A.L.; Chavis, J.S.; Corbett, C.L. A Survey of Deep Learning Methods for Cyber Security. *Information* **2019**, *10*, 122. [CrossRef]
18. Ullah, S.; Khan, M.A.; Ahmad, J.; Jamal, S.S.; e Huma, Z.; Hassan, M.T.; Pitropakis, N.; Arshad; Buchanan, W.J. HDL-IDS: A Hybrid Deep Learning Architecture for Intrusion Detection in the Internet of Vehicles. *Sensors* **2022**, *22*, 1340. [CrossRef]
19. Garnelo, M.; Shanahan, M. Reconciling deep learning with symbolic artificial intelligence: Representing objects and relations. *Curr. Opin. Behav. Sci.* **2019**, *29*, 17–23. [CrossRef]
20. d'Garcez, A.; Lamb, L.C. Neurosymbolic AI: The 3rd wave. *arXiv* **2020**, arXiv:2012.05876.

21. Palvanov, A.; Im Cho, Y. Comparisons of deep learning algorithms for MNIST in real-time environment. *Int. J. Fuzzy Log. Intell. Syst.* **2018**, *18*, 126–134. [CrossRef]
22. Diaz-del Rio, F.; Real, P.; Onchis, D. A parallel Homological Spanning Forest framework for 2D topological image analysis. *Pattern Recognit. Lett.* **2016**, *83*, 49–58. [CrossRef]
23. Feichtinger, H.; Grybos, A.; Onchis, D. Approximate dual Gabor atoms via the adjoint lattice method. *Adv. Comput. Math.* **2014**, *40*, 651–665. [CrossRef]
24. Badreddine, S.; d'Avila Garcez, A.; Serafini, L.; Spranger, M. Logic Tensor Networks. 2022. Available online: https://github.com/logictensornetworks/logictensornetworks (accessed on 1 February 2022).
25. Hájek, P. *Metamathematics of Fuzzy Logic*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2013; Volume 4.
26. Wes McKinney, W. Data Structures for Statistical Computing in Python. In Proceedings of the 9th Python in Science Conference, Austin, TX, USA, 28 June–3 July 2010; van der Walt, S., Millman, J., Eds.; 2010; pp. 56–61. [CrossRef]
27. KDD99 Dataset. 1999. Available online: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html (accessed on 1 May 2021).
28. Kayacik, H.G.; Zincir-Heywood, A.N.; Heywood, M.I. Selecting features for intrusion detection: A feature relevance analysis on KDD 99 intrusion detection datasets. In Proceedings of the Third Annual Conference on Privacy, Security and Trust. St. Andrews, NB, Canada, 12–14 October 2005; Volume 94, pp. 1722–1723.
29. Al Tobi, A.M.; Duncan, I. KDD 1999 generation faults: A review and analysis. *J. Cyber Secur. Technol.* **2018**, *2*, 164–200. [CrossRef]
30. Brugger, T. KDD Cup '99 Dataset (Network Intrusion) Considered Harmful. 2007. Available online: https://www.kdnuggets.com/news/2007/n18/4i.html (accessed on 1 May 2021).
31. Thapa, N.; Liu, Z.; Shaver, A.; Esterline, A.; Gokaraju, B.; Roy, K. Secure Cyber Defense: An Analysis of Network Intrusion-Based Dataset CCD-IDSv1 with Machine Learning and Deep Learning Models. *Electronics* **2021**, *10*, 1747. [CrossRef]
32. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp* **2018**, *1*, 108–116.