A Neuro-Symbolic Artificial Intelligence Network Intrusion Detection System

Alice Bizzarri*†, Brian Jalaian†‡, Fabrizio Riguzzi*, Nathaniel D. Bastian§
*University of Ferrara, Ferrara, Italy
†Florida Institute for Human-Machine Cognition, Pensacola, USA
‡University of West Florida, Pensacola, USA
§Army Cyber Institute, United States Military Academy, West Point, USA
*{alice.bizzarri, fabrizio.riguzzi}@unife.it, ‡bjalaian@uwf.edu, §nathaniel.bastian@westpoint.edu

***Abstract*—Ever-changing cyber threats require strong and flexible network security solutions. This paper suggests a new method to improve the performance of detecting both known and unknown attacks using a neuro-symbolic artificial intelligence (NSAI) network intrusion detection system (NIDS). Deep neural networks (DNN) learn complex network data patterns, which create a detailed overview of cyber-attack characteristics. Symbolic logic integration into the DNN allows for model training guidance by applying penalties when the DNN fails to differentiate between malicious and benign network traffic. This improves our model's adaptability to new attacks and overcomes traditional signature-based NIDS limitations. By testing our NSAI NIDS on a large cyber dataset that includes novel attack scenarios, we show that it delivers an improvement in how accurately it detects attacks compared to traditional DNN methods. While our system maintains its high accuracy in recognizing known attacks, it outperforms conventional NIDS in discovering unknown attacks. This work improves cybersecurity by introducing a new way to detect both known and unknown network intrusions by combining DNNs with symbolic logic.**

***Index Terms*—Neuro-Symbolic AI, Network Security, Intrusion Detection System**

INTRODUCTION

Nowadays, cybersecurity solutions are of utmost importance for the protection of critical network assets. The expansion of Internet of Things (IoT) devices, for example, has led to the rapid increase in cyber threats, where traditional network security approaches are often insufficient and Network Intrusion Detection Systems (NIDS) are essential for defending against these threats. These systems, whether operating on the basis of predefined "signatures" or detecting anomalies, play a crucial role in safeguarding the digital infrastructure [1].

The integration of artificial intelligence (AI) into an NIDS has led to the development of more sophisticated cyber threat detection techniques [2]. An AI-driven NIDS has the ability to adapt to evolving threats, process vast datasets and discern intricate patterns. However, this approach poses some challenges. AI models frequently have an opaque nature, require extensive training data, and suffer from overfitting [3].

As cyber threats are evolving in sophistication, NIDS encounter unparalleled obstacles, as they are commonly trained/tested exclusively against recognized attack classes, making them ineffective against unprecedented attacks. These approaches mainly use symbolic AI, such as rule-based systems, or non-symbolic (neuro) AI, such as deep neural networks. With the rise of IoT devices and the growth of increasingly sophisticated threats, detecting unknown attacks (also known as zero-day attacks) is one of the most important challenges for NIDS.

Neuro-symbolic artificial intelligence (NSAI), a paradigm that integrates traditional rule-based AI methods with contemporary deep learning techniques, is sparking a new wave of AI research. NSAI has the potential to surpass the performance of conventional deep learning models across diverse fields, such as image and video reasoning [4], [5]. The use of NSAI in cybersecurity represents progress against cyber threats. Combining neural pattern recognition and symbolic reasoning creates a dynamic, robust, and transparent system for combating cyber threats. However, integrating complex systems remains a challenge. Scalability and the trade-off between accuracy and interpretability are among the many hurdles. But the potential of NSAI is undeniable. The application of these approaches in cybersecurity will result in the advancement of integration techniques, optimization of NSAI systems for high-stress scenarios, and self-updating systems that constantly refine their knowledge base in real-time. Creating not only a more interpretable system, but also a more robust system capable of combat ever-evolving threats [6].

In the field of cybersecurity, the robustness is a key issue. As technology continues to evolve, NIDS must combat ever new and sophisticated threats. Our research contributes to this field by demonstrating how implementing a NSAI NIDS can improve the detection of unseen threats. As such, the objective of this research is to improve the detection of unknown attacks. Our primary contribution is the creation of a NSAI NIDS that leverages a Logic Tensor Network (LTN) [7] with a *Hybrid Loss*, integrating the real logic of the LTN with standard cross entropy loss.

The paper is structured as follows. We first provide an overview of the literature in the fields of NIDS and NSAI by placing our work in that context. The preliminaries section provides necessary background details for better understanding this work, while the methodology section reviews the dataset, outlines the mathematical formulations of our approach, and illustrates our experiments. We then discuss results and finish with conclusions and future work.

Related Works

Challenges in the field of cybersecurity are constantly evolving. Technological advancements are rendering traditional network security measures, specifically those relying on signature detection, increasingly ineffective. While these systems are successful in defending against known threats, they have difficulty with unknown attacks. AI is viewed as a potential solution to these systems' major challenges, in that it can efficiently process vast amounts of data in real-time, enabling speedy detection and response. Nevertheless, this advanced technology poses certain difficulties such as the requirement for large sets of annotated data for training purposes and the inherent opaqueness of certain deep learning models. Additionally, these systems are susceptible to adversarial attacks [1].

NSAI is an interdisciplinary approach that may be the key to addressing these challenges and becoming the next generation of cybersecurity solutions. NSAI has found extensive application in various fields, including healthcare, autonomous vehicles, and Natural Language Processing (NLP). In NLP, some systems combine symbolic and neural components [8], [9] to solve visual reasoning tasks and unsupervised learning of visual sentences and concepts. NSAI is currently being utilized to create predictive models for medical diagnosis and prognosis [10], [11]. Furthermore, NSAI is currently being investigated for the creation of autonomous driving systems, providing a robust and comprehensible method.

In the field of cybersecurity, NSAI represents a promising approach. A number of NSAI NIDS models have emerged, incorporating a fusion of symbolic reasoning with neural techniques [12]–[14]. These models incorporate automatic tuning relying on feedback provided by system operators to achieve improvements in their accuracy. In [14], a hybrid system utilizing k-means and random forest in combination with GNN and LSTM achieved a higher true positive rate when detecting anomalies within network traffic. The authors employed a GNN to learn normal network behavior and identify anomalies accordingly. They then integrated this system with a GNNExplainer [15] and an ontology to reduce false positives and enhance explainability.

In our approach, we utilize an LTN-based NSAI NIDS trained with a customized loss to improve the detection of unseen examples. Our methodology differs from the previously mentioned methods as we employ a *Hybrid loss* that merges the logic aspect of LTN with the standard cross-entropy loss.

Preliminaries

NSAI aims at integrating the pattern recognition capabilities of neural networks with the explicit reasoning of symbolic systems. This integration is achieved in different ways, each with a unique algorithm and learning methodology. Below we present LTNs a state-of-the-art model on which our method is based.

Logic Tensor Networks

LTNs are a NSAI formalism that effectively merges symbolic AI with neural computation, providing a coherent framework for various AI tasks such as data clustering, multi-label classification, and relational learning [7].

LTNs introduce a differentiable version of first-order logic, called *Real Logic*, into deep learning models. This integration enables the use of logical operators in neural network computations, allowing the model to reason about the relationships between various inputs. One of the pivotal advantages of LTNs is their capability to merge symbolic and sub-symbolic representations within a single model. This allows for the assimilation of both structured and unstructured data within the same framework, proving especially beneficial in domains like natural language processing and computer vision [16].

Expanding upon the foundational Neural Tensor Network (NTN) [17], LTNs employ Real Logic, an infinite-valued Fuzzy Logic language. To account for inherent real-world uncertainty, the fuzzy semantics can be employed to render the formulas of this logic partially true. Tensors in the real field interpret domains in this logic.

Grounding in logic denotes the substitution of variables in a term or formula with constant values or terms lacking variables. However, in LTNs the term grounding is used synonymously with interpretation, which is the assignment of truth values (within $[0, 1]$ in *Real Logic*) to a formula.

Terms are converted into tensors of real numbers and formulas are transformed into real numbers on the interval of [0,1] to represent their truth value. Real logic enables objects to be encoded as points in a feature space, and both features and predicates can be learned. LTNs are trained to maximize the satisfiability of a logical formula, with inference via feedforward propagation [16].

In LTNs, a theory is defined as $\mathcal{T} = (\mathcal{K}, \mathcal{G}(\cdot \mid \theta), \Theta)$, in which $\mathcal{K}$ comprises a set of closed first-order logic formulas defined on the symbol set $S = D \cup X \cup C \cup F \cup P$ representing domains, variables, constants, functions, and predicate symbols. The truth values of the formulas in $\mathcal{K}$ are then aggregated using a *formula aggregating* operator $\mathrm{SatAgg} : [0, 1]^* \rightarrow [0, 1]$.

$\mathcal{G}(\cdot \mid \theta)$ denotes a parametric grounding for all symbols $s \in S$. Furthermore, $\Theta = \{\Theta_s\}_{s \in S}$ is the hypothesis space for every set of parameters θ_s linked with symbol s. The process of learning involves searching for the parameter values θ^* that maximize satisfaction in relation to a given aggregator [7]:

$$\theta^* = \mathrm{argmax}_{\theta \in \Theta} \, \mathrm{SatAgg}_{\phi \in \mathcal{K}} \mathcal{G}_\theta(\phi) \tag{1}$$

where $\mathcal{G}_\theta(\phi)$ is *grounding* of formula ϕ. For training with gradient descent, the loss is defined as:

$$\mathcal{L} = 1 - \mathrm{SatAgg}_{\phi \in \mathcal{K}} \mathcal{G}_\theta(\phi) \tag{2}$$

In LTNs, aggregators play a crucial role in combining the truth values of logical formulas. One such aggregator, A_{pME}, is designed for *forall* quantification, $\forall$. A_{pME}, the generalized mean over error, operates on a set of truth values $a_1, \ldots, a_n$

in the interval [0,1]. This aggregator essentially quantifies the overall truth value by taking into account the error or complement of each individual truth value. Consider a scenario in which each truth value a_i represents the degree of satisfaction or truth of a particular condition. The aggregator A_{pME} computes the generalized mean of the errors, where each error is given by $(1 - a_i)^p$. Here, p is a parameter greater than or equal to 1 that affects the sensitivity to errors. The formula for A_{pME} is expressed as follows:

$$\forall : A_{pME}(a_1, \ldots, a_n) = 1 - \left(\frac{1}{n} \sum_{i=1}^{n} (1 - a_i)^p \right)^{\frac{1}{p}} \quad (3)$$

This expression captures the essence of truth value aggregation under error. As p increases, the aggregator becomes more sensitive to errors, emphasizing the importance of minimizing discrepancies between expected and true values.

By quantifying *exists*, $\exists$, the generalized mean A_{pM} is used instead. It operates on a set of truth values $a_1, \ldots, a_n$ within the interval [0,1]. This aggregator quantifies the overall truth value by taking into account the individual truth values of the conditions.

Considering the same scenario as before, the aggregator A_{pM} computes the generalized average of truth values. The formula for A_{pM} is expressed as follows:

$$\exists : A_{pM}(a_1, \ldots, a_n) = \left(\frac{1}{n} \sum_{i=1}^{n} a_i^p \right)^{\frac{1}{p}} \quad (4)$$

In summary, A_{pME} provides a flexible and interpretable mechanism for aggregating truth values with respect to errors, making it particularly suitable for scenarios where minimizing inaccuracies is critical. Instead, A_{pM} provides a versatile mechanism for aggregating truth values, offering a balance between individual truth values and sensitivity to extreme values. It is suitable for scenarios in which it is essential to identify the presence of at least one true condition.

Example: In a multi-class classification problem using a DNN in predicate P, theory $\mathcal{T}$ posits that each example is assigned to a class. So, the axioms for each class can be defined as follows:

$$\forall c \forall x_c P(x_c, l_c) \quad (5)$$

where c represents the class, x_c is a sample of class c, l_c is a label of class c, and $P(x, l)$ denotes the fact that item x is classified as l.

The *grounding* function $\mathcal{G}(P|\theta)$ maps x and l to their corresponding grounding by taking the dot product of $l^\top$ (l represented as one-hot vector) and the output of a $\mathrm{MLP}_\theta(x)$ passed through a $\mathrm{softmax}$ function, where the MLP has n output neurons corresponding to n classes. Multiplying the MLP's output by $l^\top$ gives the truth degree corresponding to space.

For additional information on Learning, Reasoning, and Querying in *Real Logic*, please refer to the original paper [7].

METHODOLOGY

Our NSAI NIDS approach seeks to improve zero-day attack detection, where zero-day attacks are attacks that have not been previously observed on the network (commonly referred to as unknown attacks or out-of-distribution samples).

Mathematical Formulation

Our NSAI NIDS is a LTN-based system. Below, we outline our approach, define the domains, variables, constants, and predicates, and present the axioms and loss function. We also introduce the functions $\mathbf{D}$ and $\mathbf{D}_{in}$. These functions return the domains of variables or constants and the domains of the arguments of a function or predicate, respectively.

Domains:

$items$ denoting the examples from the dataset.
$labels$ denoting the class labels.

Variables:

x_b, x_a for the positive examples of benign and attack class.
x, used to denote all the examples.
$\mathbf{D}(x_b) = \mathbf{D}(x_a) = \mathbf{D}(x)$

Constants:

l_b, l_a where $l_b = 0$, $l_a \neq 0$ and 0 is the benign class.
$\mathbf{D}(l_b) = \mathbf{D}(l_a) = labels$.

Predicates:

$P(x, l)$, denotes the fact that item x is labelled as l
$\mathbf{D}_{in}(P) = items, labels$.

Axioms: We use less stringent axioms which are applicable only to the categories of benign and non-benign (attack), rather than being universally relevant across all categories:

$$\forall x_b P(x_b, l_b) \wedge \forall x_a \neg P(x_a, l_b) \quad (6)$$

Grounding:

$\mathcal{G}(items) = \mathbb{R}^{1500}$; the examples from the data set are describe using 1500 features, one for each payload byte.
$\mathcal{G}(labels) = \{0, 1\}^{15}$; we use one-hot encoding to represent classes.
$\mathcal{G}(x_n) \in \mathbb{R}^{m_n \times 1500}$, is a sequence of m_n examples of class n.
$\mathcal{G}(P|\theta) : x, l \mapsto l^\top \cdot \mathrm{softmax}(\mathrm{MLP}_\theta(x))$

Learning: The logical operators and connectives are approximated using the product fuzzy logic as in [7] with $p = 2$.

$$\textbf{\textit{SAT loss}} = 1 - \mathrm{SatAgg}_{\phi \in \mathcal{K}} \mathcal{G}_\theta(\phi) \quad (7)$$

Our contribution is to integrate Cross-Entropy (CE) loss [18] with ***SAT loss*** the to enhance the network's capability to differentiate between benign and attack scenarios, along with distinguishing individual categories. By emphasizing benign and non-benign events during training, our network can even

recognize unknown attacks. To optimize our DNN also w.r.t. this further aspect, we need to add to the CE loss to the ***SAT loss***, as described by the following equation:

$$\boldsymbol{L}_{CE} = -\frac{1}{N}\sum_{i=1}^{i=N} y_i \cdot \log\left(\widehat{y}_i\right) \tag{8}$$

where N is the number of samples, $\widehat{y}_i$ and y_i are the network's prediction and the true label as one-hot vector for sample i, log is applied element-wise and $\cdot$ is the dot product. This is important to consider the classification task's contribution to the overall loss. Thus, we derive the *Hybrid-Loss*, which is:

$$\textit{Hybrid-Loss} = \boldsymbol{L}_{CE} + \boldsymbol{SAT\ Loss} \tag{9}$$

$$\textit{Hybrid-Loss} = -\frac{1}{N}\sum_{i=1}^{i=N} y_i \log\left(\widehat{y}_i\right) + (1 - \mathrm{SatAgg}_{\phi\in\mathcal{K}}\mathcal{G}_\theta(\phi)) \tag{10}$$

Figure 1 shows the system trained with the above loss function.

In the testing and validation phase, we only utilize the encapsulated DNN, $\mathrm{MLP}_\theta(x)$, as a standard model.

The ***SAT Loss*** is a regularization term that penalizes the network for misclassifying a benign/non-benign example. Essentially, our approach uses the LTN structure to "encapsulate" an MLP (or other DNN) and add a regularization term. This regularization term expresses the network's ability to distinguish between benign and non-benign examples. This is particularly interesting for our problem because we have a multiclassification problem, as well as a problem of classifying unknown attacks. In this classification, the network's task is to determine whether an attack has occurred, rather than identifying the specific type of attack.

EXPERIMENTS

For each experiment, we trained a benchmark DNN using the identical architecture and train-val-test dataset used in the *Hybrid-LTN* model. We built a 1D Convolutional Neural Network (CNN) depicted in Figure 2.

Each convolutional layer has a ReLU activation function. The first dense layer has a ReLU activation function, and the output layer has a softmax activation function.

To better understand the results, it is important to differentiate between the training and test phases.

In this sub-section, we describe the dataset and the train and test phase of our experiments.

Dataset

The CIC-IDS2017 dataset, created by the Canadian Institute for Cybersecurity in 2017 [19], was used herein. This dataset contains two parts: packet-based data in packet capture (PCAP) format and flow-based data in CSV format, obtained by extracting 80 features from PCAP using CICFlowMeter.

CICFlowMeter is a tool for generating and analyzing network traffic flows. It can calculate over 80 statistical traffic characteristics, including duration, number of packets, number of bytes, and packet length. The tool supports bidirectional flows, where the first packet determines the forward and return direction. Characteristics can be calculated separately for each direction. CICFlowMeter allows users to select features from a pre-existing list, add new features, and control the flow timeout duration. The output is a CSV file with six labeled columns for each flow, including flow ID, source and destination IP addresses, source and destination ports, and protocol [20].

Both data were captured during simulated network traffic in packet-based and bidirectional flow-based formats, including the latest attacks and benign traffic. For our study, we used only packet-based data. The dataset collects simulated traffic information for an acquisition period of five days.

The packet-based information in CIC-IDS2017 is unlabeled, making it necessary to use the Payload-Byte tool [21] for the extraction and labeling of network traffic packet capture files using the metadata provided in the dataset. The tool uses the features described in PCAPs to match packets with flow-based labeled data instances. Due to the variability in packet size, Payload-Byte uses a maximum payload length of 1500 bytes, with each byte converted to an integer feature between 0 and 255. Upon data labeling, any duplicate instances and those devoid of payload data are eliminated.

We under-sampled the dataset to reduce the number of benign samples and achieve a more balanced dataset. We removed five attack classes from the training set, which constitute less than 1% of the total samples, and used them only in the test phase as *zero-day* attacks (i.e., out-of-distribution inputs). Then, we ensured an equal number of examples for each attack by selecting for each one the same number of examples equal to the one with the least number available, i.e., FTP-Patator, resulting in 31,843 examples per class. Additionally, we arbitrarily decreased the total number of benign examples from 362,108 to 200,000 to reduce complexity. The new dataset contains 454,744 examples, with 200,000 samples being benign and 254,744 being attack samples. Table I shows statistics about the under-sampled dataset.

Classes	**Resampled DS**
BENIGN	**200,000**
ATTACKS	**254,744**
DoS Hulk	31,843
DDoS	31,843
DoS GoldenEye	31,843
DoS slowloris	31,843
Infiltration	31,843
DoS Slowhttptest	31,843
SSH-Patator	31,843
FTP-Patator	31,843
Zero-day attacks	**31,966**
Heartbleed	13,486
Brute Force (Web Attack)	11,754
XSS (Web Attack)	3,341
Bot	2,543
PortScan	830
Sql Injection (Web Attack)	12

TABLE I: Number of samples for each class after under-sampling.

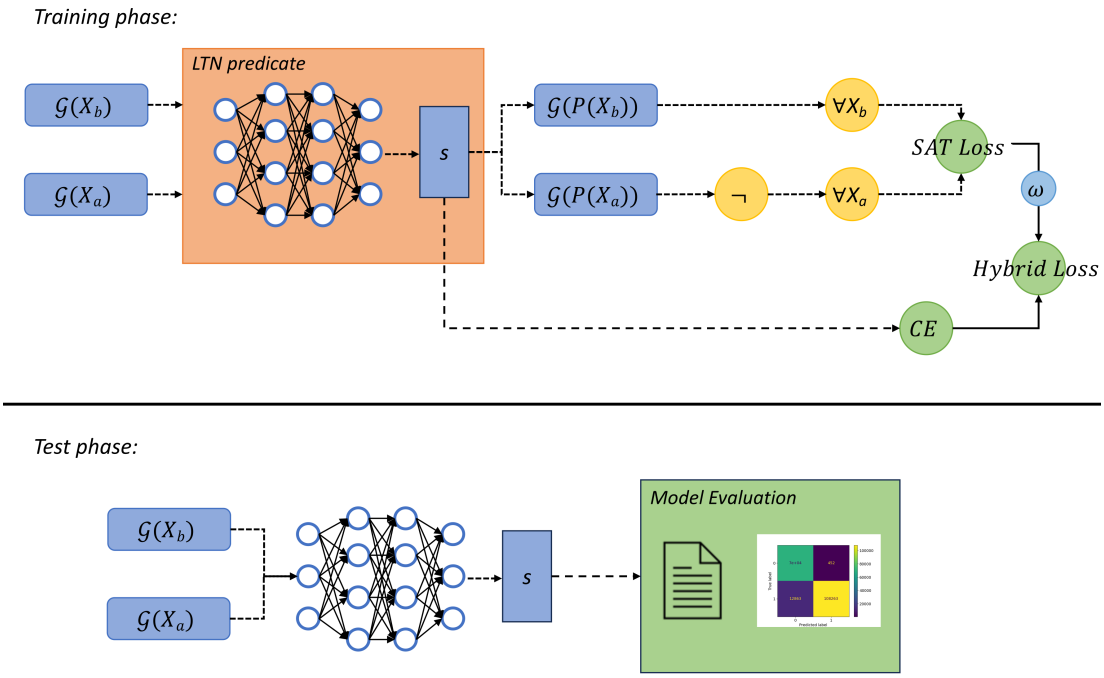


Fig. 1: *Training phase (top):* the LTN predicate $\mathcal{G}(P)$ takes $\mathcal{G}(X_b)$ and $\mathcal{G}(X_a)$ as inputs and produces $\mathcal{G}(P(X_b))$ and $\mathcal{G}(P(X_a))$ as outputs. *Test phase (bottom):* the network takes $\mathcal{G}(X_b)$ and $\mathcal{G}(X_a)$ as inputs and produces score s as output. s is used to evaluate model's prediction.

We divided the new dataset into 80%, 10%, 10% parts for training, validation and testing in a stratified manner. *Zero-day* attacks are used only in the test phase in order to measure the robustness of our approach.

Training phase

In this phase, we train the network using an LTN framework. As shows in Fig 1 (top), we have $\mathcal{G}(X_b)$ and $\mathcal{G}(X_a)$ as input to our network. In our experiments, we used a 1D CNN with batch size 128 and x_n is represented as grayscale image with dimension 1x1500, so our input shape is (128, 1, 1500, 1). The output is a multi-class classification defined as $s = \text{softmax}\left(\text{1DCNN}_\theta(x)\right)$, we have 9 known classes (8+1), so our output have shape (128, 9). To calculate the ***SAT Loss***, we use $\mathcal{G}(P(X_b))$ (or $\mathcal{G}(P(X_a))$), which represents the network prediction score for the Benign (or Attacks) class. As defined above, $\mathcal{G}(P(X_n))$ is define as $l^\top \cdot \text{softmax}\left(\text{MLP}_\theta(x)\right)$, and using 1D CNN we have $l^\top \cdot \text{softmax}\left(\text{1DCNN}_\theta(x)\right)$. In other words $\mathcal{G}(P(X_n)) = l^\top \cdot s$. We calculate Categorical Cross-Entropy Loss using s. We sum CE and ***SAT Loss***. We can also use a parameter ω to improve the weight of ***SAT Loss***. In our experiments, we use $\omega = 1$. Then, we back-propagate the *Hybrid Loss* through the network.

We also trained a 1D CNN, a binary LTN, and a categorical LTN to compare with our system. For all of these, we used the same 1D CNN architecture used in our NSAI approach (Fig. 2).

Test phase

In the test phase, we used the 1D CNN that was trained in the previous phase. We extracted the network from the LTN predicate, so we will only have the score vector s as output.

As shows in Fig. 1 (bottom), we then evaluated the model in a conventional manner, calculating accuracy for multi-class, binary, and zero-day scenarios, as well as F1-Score.

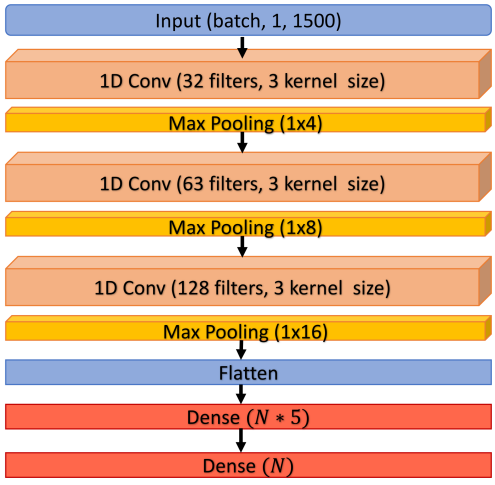


Fig. 2: 1D CNN Model Architecture, where N represents the number of classes.

Results and Discussion

In this section, we compare the benchmarks and *Hybrid-LTN* models results after 30 or 50 epochs

It is important to note that our comparative analysis involved a basic approach, using a simple 1D CNN as the underlying network architecture. This deliberate simplification aligns with the central goal of our study, which is not to pursue exceptional accuracy, but rather to demonstrate the potential improvement in robustness and detection of novel attacks through the incorporation of logic.

By focusing on a methodologically simpler approach, we can attribute any observed improvements to the integration of logic, providing a more transparent demonstration of innovative contributions to improving system resilience against evolving cyber threats.

Table II show the results. For each experiment, we evaluated the accuracy of both models, *Hybrid-LTN* and Benchmark, in the following ways:

- Multi-class accuracy on the test set consisting only of the 9 known classes (benign and 8 attacks).
- Binary accuracy on the test set consisting only of the 9 known classes (benign vs. attacks).
- Multi-class accuracy on a test set consisting of all 15 known and unknown classes (benign and 14 attacks).
- Binary accuracy on the test set consisting of all 15 classes, including both known and unknown (benign vs. attacks).
- Binary accuracy on the test set consisting only of 6 unknown attacks (benign vs. attacks).

The F1-score was calculated for binary classification only.

Figure 3 shows the confusion matrices for both the benchmarks and our NSAI NIDS. Both the matrices for the 9 (8+1) known classes and the matrix for unknown attacks for each tested system, vanilla 1D CNN, basic multiclass LTN (M-LTN), basic binary LTN (B-LTN) and our *Hybrid-LTN* are shown.

In Figure 4 we can see the training and validation loss as a function of the epoch for our system and for the baselines. Considering the unbalanced nature of the dataset, despite resampling efforts, we decide to display both accuracy and F1-score, as the latter better shows the performance of the system in the case of unbalanced classes.

Interestingly, although there is a small difference between the multi-class accuracy of the two networks on the known classes, the binary accuracy of our method is several percentage points better for unknown attacks when compared to the reference network.

When comparing *Hybrid-LTN* with the baselines, we can observe that the *Hybrid-LTN* model outperformed the benchmarks in terms of detection of unknown attacks. This is due to the *Hybrid-LTN*'s use of logic during the training phase, which provides a key advantage. The ability to detect unknown threats is critical, and the model's ability to understand logical patterns and relationships between different classes can contribute to better generalization.

The binary accuracy for unknown attacks reflects the model's effectiveness in distinguishing between benign traffic and previously unobserved threats. This outcome can be attributed to the *Hybrid-LTN* model's ability to learn complex logic patterns that could indicate the presence of anomalous activity. The *Hybrid-LTN* model's higher binary accuracy for unknown attacks suggests that the logic built in during training contributes to greater robustness. This is particularly relevant in real-world scenarios where the ability to detect previously unknown threats is crucial to system security.

Our *Hybrid-LTN* approach stands out when compared to a Binary LTN-based system, underscoring its relevance and innovation in addressing the complexity of cyber threats. Unlike the straightforward application of Binary LTN, the integration of logic as a regularization factor in *Hybrid-LTN* demonstrates superior robustness. While logic alone improves resilience compared to traditional 1D CNN, the fusion of logic with neural networks achieves even more significant enhancements. This nuanced integration proves crucial in fortifying intrusion detection systems against evolving threats.

These results demonstrate how the use of NSAI NIDS brings real benefits in the field of cybersecurity. Detecting previously unseen threats is one of the key challenges in cybersecurity. Our system shows how the use of logic during the training phase can bring benefits in terms of robustness. In fact, both systems, *Hybrid-LTN* and 1D CNN, have the same ability to detect known attacks, but our NSAI NIDS has better performance than the pure neural system in detecting unknown attacks. By evaluating only unseen attacks, *Hybrid-LTN* achieves an accuracy of $8 \sim 12\%$ and F1-score $7 \sim 10\%$ higher than traditional deep learning techniques.

Confusion matrices (CMs) are a valuable tool for exploring the performance of intrusion detection models. Figure 3 shows that logic-based approaches (LTNs) have fewer False Negatives than traditional neural networks (1D CNN). LTN approaches, including *Hybrid-LTN*, maintain significantly lower False Negatives than 1D CNN. In the context of cybersecurity, it is crucial to identify attacks accurately to avoid serious consequences. The *Hybrid-LTN* exhibits a lower number of False Positives compared to the other LTNs. This indicates its ability to limit misclassifications of benign traffic as attacks, resulting in higher overall accuracy. The *Hybrid-LTN* model stands out for having the lowest number of False Negatives when dealing with unknown attacks. This confirms its superior ability to detect even previously unobserved threats. The analysis of the confusion matrix offers a detailed view of the performance of each model. Among the logic-based approaches, particularly *Hybrid-LTN*, emerge as the leaders in reducing False Negatives and accurately handling False Positives. These results suggest greater reliability and accuracy in identifying attacks, including unknown attacks.

Observing the difference between training and validation accuracy during the learning process can reveal crucial aspects about the robustness of the *Hybrid-LTN* model. Reducing the discrepancy between these two measures suggests a lower incidence of overfitting, which translates into significant benefits

Accuracy				
30 epochs				
Test Set	***Hybrid-LTN***	**1D CNN**	**LTN multi-class**	**LTN binary**
Multi-class 9 known classes	81.04%	80.88%	80.13%	-
Binary 9 known classes	99.53%	99.44%	99.06%	99.45%
Multi-class 15 classes	67.48%	67.35%	66.73%	-
Binary 15 classes	92.20%	90.68%	91.69%	90.44%
Binary 6 unknown classes	**55.70%**	**47.13%**	**54.98%**	**45.56%**
50 epochs				
Test Set	***Hybrid-LTN***	**1D CNN**	**LTN multi-class**	**LTN binary**
Multi-class 9 known classes	81.08%	80.99%	80.41%	-
Binary 9 known classes	99.57%	99.42%	99.46%	99.42%
Multi-class 15 classes	67.52%	67.45%	66.96%	-
Binary 15 classes	93.03%	90.88%	92.19%	89.40%
Binary 6 unknown classes	**60.47%**	**48.34%**	**56.06%**	**39.40%**
F1-Score				
30 epochs				
Test Set	***Hybrid-LTN***	**1D CNN**	**LTN multi-class**	**LTN binary**
Binary 9 known classes	99.58%	99.50%	99.16%	99.51%
Binary 15 classes	93.47%	92.12%	93.00%	91.88%
Binary 6 unknown classes	**71.55%**	**64.07%**	**70.95%**	**62.60%**
50 epochs				
Test Set	***Hybrid-LTN***	**1D CNN**	**LTN multi-class**	**LTN binary**
Binary 9 known classes	99.62%	99.49%	99.52%	99.49%
Binary 15 classes	94.20%	90.88%	93.47%	90.93%
Binary 6 unknown classes	**75.37%**	**65.18%**	**71.80%**	**56.61%**

TABLE II: Accuracy and F1-Score from 30 and 50 epochs of training with the *Adamax* optimizer for both the *Hybrid-LTN* and benchmark models.

for cybersecurity.

Figure 4 highlights that, in the case of LTN models, the difference between training and validation accuracy is smaller than in the purely neural approach. This is a critical indicator, as a significant difference could indicate that the training is too focused on specific data, lacking generalization.

A smaller difference between training and validation performance suggests a lower level of overfitting. Overfitting can lead to overspecialization to training data, making the model less adaptable to new data or situations. Reducing overfitting is crucial to ensure that the model can generalize accurately to new scenarios, enhancing its robustness.

The model's hybrid nature, which combines logic and deep learning techniques, helps reduce overfitting. The logic brings a semantic understanding component that can help more accurately model the underlying patterns in the training data without falling into overspecialization.

These findings open the door for the development of NSAI-based systems that can enhance interpretability and robustness in cybersecurity contexts. By reducing overfitting, these models become more reliable and can adapt to evolving threats without sacrificing accuracy.

Conclusions and Future Work

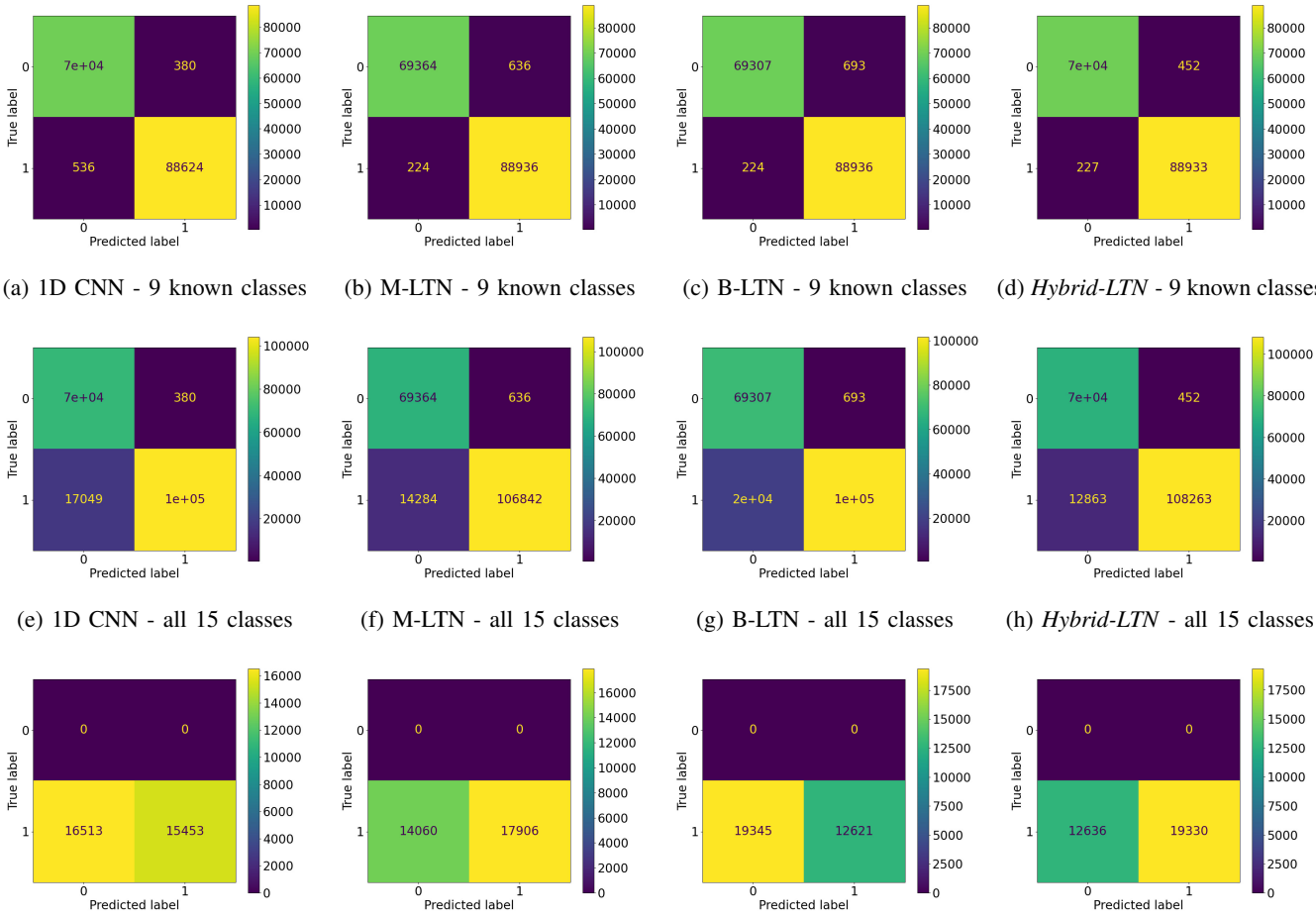
The objective of this research was to improve the robustness of NIDS using a NSAI approach. The primary focus was to improve the detection of malicious activity, particularly those that have not been previously encountered (i.e., zero-days). Identifying previously unknown attacks is of vital importance in network security. As technology continues to advance, we encounter new and more complex forms of attacks. This work aims to address these new challenges.

This paper introduced a NSAI NIDS that used LTNs with Hybrid Loss to extend previous approaches in order to focus more on distinguishing benign network traffic from malicious. A logic term was included in the loss to evaluate the network's misclassification of benign and non-benign examples. The results demonstrated how this approach leads to better classification of unknown class examples.

However, NSAI approaches present a number of challenges, such as scalability. Although our approach incorporates simple constraints to mitigate resource limitations, scalability remains an issue. As the complexity of network environments and cyber threats continues to evolve, ensuring the scalability of NSAI NIDS solutions will be critical to their practical deployment and effectiveness.

Furthermore, while integrating neural and symbolic elements, our approach does not inherently improve explainability, highlighting the need for systems that can justify their decisions. In future work, we aim to address these challenges and further enhance our NSAI NIDS. Specifically, we plan to integrate additional information sources, such as NetFlow data, into the training process, alongside payload data. This holistic approach seeks to improve the system's robustness and explainability, leveraging the strengths of neural-symbolic integration.

In conclusion, while there are challenges ahead, our research lays the foundation for building NSAI NIDS capable of effectively detecting and mitigating emerging cyber threats. By embracing an interdisciplinary approach and continuing to innovate, we can better safeguard digital ecosystems against evolving security risks.



(a) 1D CNN - 9 known classes (b) M-LTN - 9 known classes (c) B-LTN - 9 known classes (d) *Hybrid-LTN* - 9 known classes

(e) 1D CNN - all 15 classes (f) M-LTN - all 15 classes (g) B-LTN - all 15 classes (h) *Hybrid-LTN* - all 15 classes

(i) 1D CNN - 6 unknown classes (j) M-LTN - 6 unknown classes (k) B-LTN - 6 unknown classes (l) *Hybrid-LTN* - 6 unknown

Fig. 3: Confusion matrix of the experiment with 50 epochs.

ACKNOWLEDGEMENTS

This work was supported in part by the U.S. Military Academy (USMA) under Cooperative Agreement No. W911NF-23-2-0108, the U.S. Army Combat Capabilities Development Command Army Research Laboratory under Support Agreement No. USMA 21050, and the Defense Advanced Research Projects Agency under Support Agreement No. USMA 23004. The views and conclusions expressed in this paper are those of the authors and do not reflect the official policy or position of the U.S. Military Academy, U.S. Army, U.S. Department of Defense, or U.S. Government.

REFERENCES

[1] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016.

[2] A. Drewek-Ossowicka, M. Pietrołaj, and J. Rumiński, "A survey of neural networks usage for intrusion detection systems," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, pp. 497–514, 2021.

[3] 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*, pp. 305–316, 2010.

[4] Z. Susskind, B. Arden, L. K. John, P. Stockton, and E. B. John, "Neuro-symbolic ai: An emerging class of ai workloads and their characterization," *arXiv preprint arXiv:2109.06133*, 2021.

[5] G. Hadash, E. Kermany, B. Carmeli, O. Lavi, G. Kour, and A. Jacovi, "Estimate and replace: A novel approach to integrating deep neural networks with existing applications," *arXiv preprint arXiv:1804.09028*, 2018.

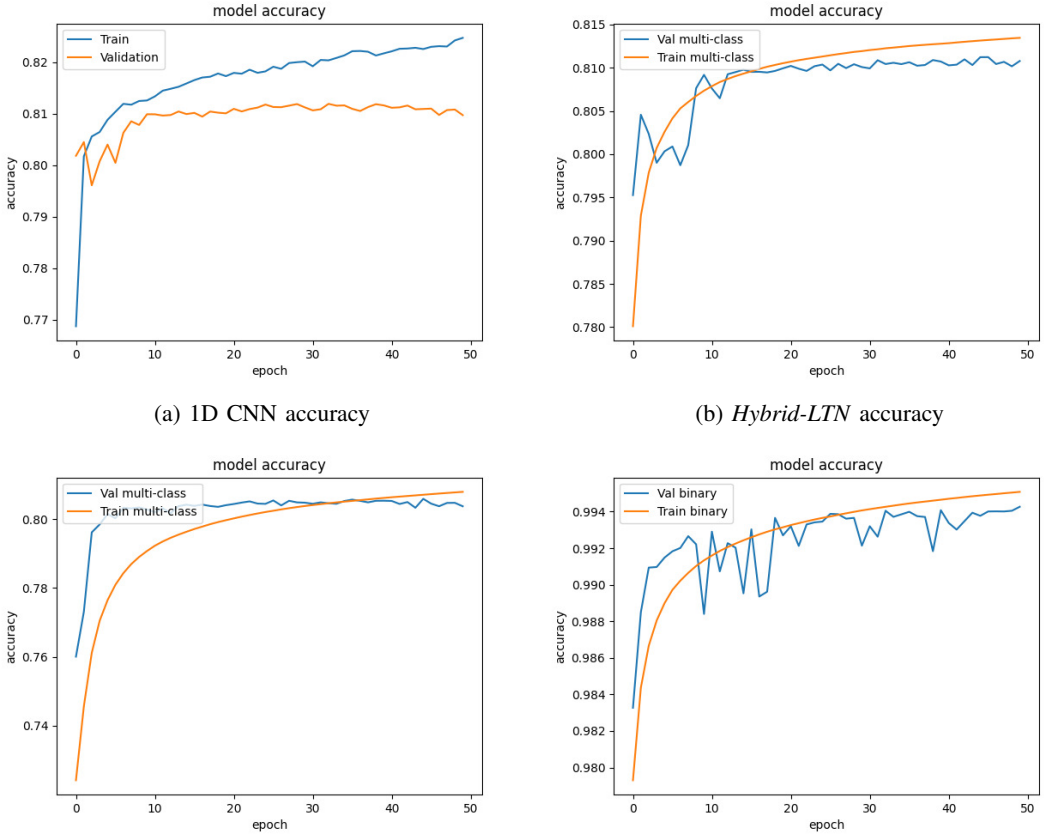
[6] B. Jalaian and N. Bastian, "Neurosymbolic ai in cybersecurity: Bridging pattern recognition and symbolic reasoning." in *Proceedings of the 2023 IEEE Military Communications Conference.*, 2023.

[7] L. Serafini and A. d. Garcez, "Logic tensor networks: Deep learning and logical reasoning from data and knowledge," in *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2016, pp. 2235–2242.

[8] J. Mao, C. Gan, P. Kohli, J. B. Tenenbaum, and J. Wu, "The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision," *CoRR*, vol. abs/1904.12584, 2019. [Online]. Available: http://arxiv.org/abs/1904.12584

[9] X. Chen, C. Liang, A. W. Yu, D. Zhou, D. Song, and Q. V. Le, "Neural symbolic reader: Scalable integration of distributed and symbolic representations for reading comprehension," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=ryxjnREFwH

[10] V. Patil, M. Madgi, and A. Kiran, "Early prediction of alzheimer's disease using convolutional neural network: a review," *The Egyptian Journal of Neurology, Psychiatry and Neurosurgery*, vol. 58, no. 1, pp. 1–14, 2022. [Online]. Available: https://ejnpn.springeropen.com/counter/pdf/10.1186/s41983-



(a) 1D CNN accuracy

(b) *Hybrid-LTN* accuracy

(c) M-LTN accuracy

(d) B-LTN accuracy

Fig. 4: Validation and training accuracy of the experiment with 50 epochs.

022-00571-w

[11] A. N. Fadja, M. Fraccaroli, A. Bizzarri, G. Mazzuchelli, and E. Lamma, "Neural-symbolic ensemble learning for early-stage prediction of critical state of covid-19 patients," *Medical & Biological Engineering & Computing*, vol. 60, no. 12, pp. 3461–3474, 2022.

[12] S. S. Sivatha Sindhu, S. Geetha, M. Marikannan, and A. Kannan, "A neuro-genetic based short-term forecasting framework for network intrusion prediction system," *International Journal of Automation and Computing*, vol. 6, no. 4, pp. 406–414, oct 21 2009. [Online]. Available: http://dx.doi.org/10.1007/S11633-009-0406-Y

[13] D. Onchis, C. Istin, and E. Hogea, "A neuro-symbolic classifier with optimized satisfiability for monitoring security alerts in network traffic." *Applied Sciences*, vol. 12, no. 22, p. 11502, nov 12 2022. [Online]. Available: http://dx.doi.org/10.3390/app122211502

[14] C. Liu, Z. Gu, and J. Wang, "A hybrid intrusion detection system based on scalable k-means+ random forest and deep learning," *Ieee Access*, vol. 9, pp. 75 729–75 740, 2021.

[15] Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, "Gnnexplainer: Generating explanations for graph neural networks," *Advances in neural information processing systems*, vol. 32, 2019.

[16] S. Badreddine, A. d. Garcez, L. Serafini, and M. Spranger, "Logic tensor networks," *Artificial Intelligence*, vol. 303, p. 103649, 2022.

[17] R. Socher, D. Chen, C. D. Manning, and A. Ng, "Reasoning with neural tensor networks for knowledge base completion," *Advances in neural information processing systems*, vol. 26, 2013.

[18] S. Mannor, D. Peleg, and R. Rubinstein, "The cross entropy method for classification," in *Proceedings of the 22nd international conference on Machine learning*, 2005, pp. 561–568.

[19] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." *ICISSp*, vol. 1, pp. 108–116, 2018.

[20] G. Draper-Gil., A. H. Lashkari., M. S. I. Mamun., and A. A. Ghorbani., "Characterization of encrypted and vpn traffic using time-related features," in *Proceedings of the 2nd International Conference on Information Systems Security and Privacy - ICISSP*, INSTICC. SciTePress, 2016, pp. 407–414.

[21] Y. A. Farrukh, I. Khan, S. Wali, D. Bierbrauer, J. A. Pavlik, and N. D. Bastian, "Payload-byte: A tool for extracting and labeling packet capture files of modern network intrusion detection datasets," in *2022 IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT)*. IEEE, 2022, pp. 58–67.