DEEP RESNIDS: A MULTISTAGE AI FRAMEWORK FOR NOVELTY DETECTION IN NETWORK TRAFFIC

Anonymous authors
Paper under double-blind review

ABSTRACT

Ensuring computer and network system security is crucial in today's digital landscape. Network intrusion detection systems (NIDS) monitor network traffic to identify potential threats. However, traditional NIDS struggle to adapt to evolving cyberattack tactics. To address this, we propose an AI-enabled novelty detection framework to handle zero-day, out-of-distribution, and adversarial evasion attacks. Our framework comprises three sequential deep neural network architectures: one for the classifier and two for specific autoencoders, designed to effectively detect both known attack patterns and novel, previously unseen samples. We use innovative transfer learning, unfreezing specific neurons, and layer combinations to enhance resilience. Leveraging the one-shot learning approach in the transfer learning component of the framework, we demonstrate continuous improvement in detection accuracy for both known and novel network traffic patterns. Our experiments on benchmark intrusion detection data sets achieved, on average, 98.5% accuracy in detecting various attacks.

1 INTRODUCTION

Network intrusion detection systems (NIDS) are designed to serve as vigilant sentinels, continuously monitoring network traffic. Their primary function involves meticulously detecting unauthorized or potentially malicious activities that threaten the confidentiality, availability, and integrity of the protected computer and network systems. Today, cutting-edge anomaly-based NIDS detection methods leverage the capabilities of deep learning (DL) algorithms (refer to Table 1). However, cyber attackers continuously devise new tactics to orchestrate a variety of sophisticated maneuvers, including the creation of adversarial attacks to evade the defender's detection mechanisms. Moreover, they can ingeniously devise novel zero-day attacks, exploiting vulnerabilities that have yet to be identified or patched by security systems. This poses a formidable challenge to detect intrusions in network traffic, compelling the need for continuous improvement and innovation in the detection capabilities of the NIDS. The ongoing battle between defenders and attackers necessitates enhancing NIDS capabilities, ensuring their ability to discern and counteract the ever-changing adversarial strategies.

In the intricate realm of network security, the efficacy of classifiers and anomaly detectors emerges as a critical focal point. Each detection mechanism wields unique strengths and grapples with distinct limitations, creating a duality of capabilities and challenges. A DL-based classifier, harnessed to its training data set, demonstrates commendable proficiency in identifying familiar attack patterns. Its well-learned parameters enable it to efficiently recognize known threats, contributing to a robust defense posture. However, the classifier's rigid adherence to its training data renders it less effective in identifying novel attacks, emerging from the ever-evolving landscape of cyber threats, resulting in a surge of false negatives. On the other end of the spectrum, the anomaly detector introduces a distinct paradigm to differentiate between benign and malicious traffic. The DL-based detector thrives on its ability to detect deviations from historical network communication patterns, irrespective of whether these anomalies correspond to known or novel attacks. Its ability to transcend the boundaries of familiarity makes it an invaluable asset in the dynamic cybersecurity landscape. However, this broad sensitivity to deviations often exacts a cost in terms of false positives. The misidentification of benign or atypical network behavior results in a potentially higher volume of false alarms, consequently amplifying the backlog of alerts awaiting investigation by human security analysts within a resource-constrained cybersecurity environment. This, in turn, delays the inspection of genuine alerts within the backlog, compromising the defending organization's security posture.

Recent literature on state-of-the-art NIDS approaches (refer to Table 1) has leveraged both supervised and unsupervised learning paradigms to develop classifiers and anomaly detectors. These studies have created various types of DL-

Table 1: Recent state-of-the-art approaches in DL-based NIDS and their key characteristics

Data source	Paper	Year	Learning paradigm	Detection paradigm		Data set	Model	Robustness	
				Classification	Anomaly			Zero-day	Adversarial
Flow data	Yang et al. (Yang et al., 2020)	2020	Unsupervised	✗	✓	NSL-KDD, UNSW-NB15	AE	✓	✗
	Andresini et al. (Andresini et al., 2020)	2020	Supervised	✓	✗	KDD Cup'99, UNSW-NB15, CICIDS2017	AE, CNN	✗	✗
	Bovenzi et al. (Bovenzi et al., 2020)	2020	Unsupervised, Supervised	✓	✓	Bot-IoT	AE	✓	✗
	Verkerken et al. (Verkerken et al., 2023)	2023	Unsupervised, Supervised	✓	✓	CICIDS2017, CICIDS2018	AE, DNN	✓	✗
	Lam et al. (Lam, 2021)	2021	Supervised	✓	✗	CICIDS2018	CNN	✗	✗
	Merna et al. (Gamal et al., 2021)	2021	Supervised	✓	✗	Bot-IoT, UNSW-NB15	CNN	✓	✗
	Aljumah et al. (Aljumah, 2021)	2021	Supervised	✓	✗	Bot-IoT	CNN	✗	✗
	Akhtar et al. (Akhtar & Feng, 2021)	2021	Supervised	✓	✗	NSL-KDD	CNN	✗	✗
	Yu et al. (Yu & Bian, 2020)	2020	Supervised	✓	✗	NSL-KDD, UNSW-NB15	CNN, DNN	✗	✗
	Jiang et al. (Jiang et al., 2020)	2020	Supervised	✓	✗	NSL-KDD, UNSW-NB15	CNN, LSTM	✗	✗
	Yao et al. (Yao et al., 2021)	2021	Supervised	✓	✗	NSL-KDD, KDD Cup'99	CNN, LSTM	✗	✗
	Liu et al. (Liu et al., 2021)	2021	Supervised	✓	✗	NSL-KDD, CICIDS2017	CNN, LSTM,	✗	✗
	Wang et al. (Wang & Li, 2021)	2021	Supervised	✓	✗	CICDDoS2019	CNN, Transformer	✗	✗
	Yin et al. (Yin et al., 2021)	2021	Supervised	✓	✗	NSL-KDD, CICIDS2017	CNN, MHA	✗	✗
	Li et al. (Li et al., 2022)	2022	Supervised	✓	✗	CICIDS2017, CICDoS2017	GNN	✗	✗
	Caville et al. (Caville et al., 2022)	2022	Self-supervised	✓	✓	CICIDS2018, UNSW-NB15	GNN	✗	✗
	Lan et al. (Lan et al., 2022)	2022	Supervised	✓	✗	UNSW-NB15	GNN	✗	✗
Packet data	Sun et al. (Sun et al., 2020)	2020	Supervised	✓	✗	CICIDS2017	CNN, LSTM	✗	✗
	Farrukh et al. (Farrukh et al., 2022)	2022	Supervised	✓	✗	CICIDS2017, UNSW-NB15	CNN, LSTM, DNN	✗	✗
	Liu et al. (Liu et al., 2022b)	2022	Supervised	✓	✗	CICIDS2017, ISCX2012, CSIC2010	LSTM, CNN, MHA	✗	✗
	De Lucia et al. (De Lucia et al., 2021)	2021	Supervised	✓	✗	UNSW-NB15	1d-CNN, DNN	✗	✗
	Bierbrauer et al. (Bierbrauer et al., 2023)	2022	Supervised	✓	✗	CICIDS2017,UNSW-NB15	1d-CNN, RF	✗	✗
	Yu et al. (Yu et al., 2021)	2021	Supervised	✓	✗	CICIDS2017, CICIDS2018	CNN	✗	✗
	Hore et al. (Hore et al., 2023b)	2023	Unsupervised	✗	✓	CICIDS2017	AE	✓	✗
	Premkumar et al. (Premkumar et al., 2023)	2023	Supervised	✓	✗	CICIDS2017	GNN, DNN	✗	✗
	Deep ResNIDS (this study)		**Unsupervised, Supervised**	✓	✓	**CICIDS2017, CICIDS2018**	**AE, DNN**	✓	✓

based models, including autoencoders (AE), convolutional neural networks (CNNs), deep neural networks (DNNs), long short-term memory (LSTM), multi-head attention (MHA), and graph neural networks (GNNs), among others. These models were trained using either flow-based or packet-based network data. While only a handful of studies have targeted identifying zero-day attacks, the majority have exclusively focused on detecting known attack patterns. To the best of our knowledge, no prior work has comprehensively investigated novelty detection in network traffic (zero-day, OOD, and adversarially perturbed evasion attacks) through the analysis of granular packet-level data.

Our work addresses the challenges within both supervised and unsupervised learning paradigms, culminating in harmoniously integrating their respective capabilities. The overarching goal of our study is to develop an adaptive and responsive network intrusion detection framework that is robust and resilient in identifying known attacks, detecting novel attacks, and discerning benign anomalies. We propose an innovative AI-enabled framework, harnessing the power of supervised, unsupervised, and transfer learning methodologies, designed to detect novel patterns within network traffic. At the core of this AI framework are three distinct sequential deep neural network (DNN) architectures, each representing a separate stage. These stages have been constructed to effectively distinguish between recognized attack patterns and the emergence of anomalies within network traffic. The models in the framework are trained using packet-level network data, which enhances its suitability for real-time detection compared to a flow-level approach. We discuss the limitations of flow-based NIDS in Section 2.

This study makes several notable contributions. The primary contribution lies in the development of a multistage *deep* learning NIDS framework, *Deep ResNIDS*, for enhanced security and resilience to novel attacks. The study also introduces an innovative transfer learning technique as another distinctive feature of the approach. This technique involves selectively unfreezing specific neurons and layer combinations within the DNN architectures, combined with the introduction of new layers during one-shot learning. The goal is to enhance the framework's adaptability to novel attacks while retaining its proficiency in recognizing known attack patterns. Experiments conducted using publicly available network intrusion data sets demonstrate the effectiveness of our approach. The first-stage malicious packet classifier excels in identifying known threats but faces challenges with new attacks, OOD, and perturbed known-attack samples. This demonstrates the shortcoming of the DL-based classifiers found in literature. In contrast, the second-stage autoencoder-based anomaly detector identifies false negatives by analyzing benign traffic from the first-stage classifier and extends its detection ability to include zero-day, OOD, and adversarially perturbed network packets. The third-stage autoencoder-based novelty detector further automates the identification of perturbed known attack samples and effectively segregates novel network traffic patterns, thereby reducing the cognitive burden of the human security analysts in identifying camouflaged known attacks and routing them for effective mitigation. The *Deep ResNIDS* framework's resilience is substantiated through experiments, showcasing that retrained malicious packet and novelty detectors maintain accuracy against adversarial samples without compromising in-distribution attack pattern detection. This underscores the scalability of the proposed multistage AI framework. Collectively, these insights propel the advancement of adaptive network intrusion detection, substantially benefiting the cybersecurity community.

The rest of this paper is organized as follows. Section 2 reviews literature on DL-based NIDS. Section 3 details our AI framework, featuring a classifier and two autoencoders, along with an innovative transfer learning mechanism. Section 4 discusses numerical experiments, and Section 5 presents the results and analysis. Lastly, Section 6 provides conclusions from our study.

2 LITERATURE REVIEW

2.1 NETWORK INTRUSION DETECTION SYSTEMS (NIDS) AND CHALLENGES IN FLOW-BASED APPROACHES

Advancements in computing algorithms and the increased availability of computing resources have propelled the utilization of DNN architectures in detecting malicious activities within NIDS. In a supervised learning approach, DL-based NIDS are trained using both malicious and benign historical data. Conversely, in an unsupervised learning approach, only benign data is employed to train the intrusion detection model (Ahmad et al., 2021; Imran et al., 2022). DL-based NIDS are developed by utilizing features extracted either from network flows or directly from packet data. The primary method for developing NIDS involves extracting features from flow data, where network communications (flows) are analyzed by aggregating information from the respective packets. Flow-based features are typically obtained from packet header data, often using tools like CIC FlowMeter (Sharafaldin et al., 2018). DL models are then trained with these features for anomaly detection or malicious network traffic classification tasks. Flow-based NIDS has specific limitations. They primarily analyze completed sender-receiver flows, which makes them suitable for offline analysis (He et al., 2023). Since their focus is primarily on lower TCP/IP levels, they encounter challenges in detecting higher-level attacks (Lee et al., 2001). For example, *DDoS* attacks often target headers, while *SQL injection* attacks affect payloads (Liu et al., 2022a). In a flow-based approach, attacks are identified based on flow features, often neglecting packet-level behavior. (A detailed literature review on DL-enabled NIDS using flow data is presented in Appendix B.2.)

2.1.1 DL-ENABLED NIDS BASED ON PACKET DATA

An alternative approach to developing DL-enabled NIDS involves directly extracting features from individual packets and training the model with these packet-based features. (Sun et al., 2020) introduced DL-IDS, which employs a hybrid network combining CNN and LSTM to extract spatial and temporal features from raw network traffic data, enhancing intrusion detection. To address the challenge of an unbalanced number of samples across different attack types during model training, DL-IDS used a category weight optimization method for increased robustness. In another study (Farrukh et al., 2022), a tool named "payload-byte" was introduced to extract packet payload bytes from publicly available data sets containing raw packet capture (pcap) files. Using these extracted packets, several DL-based models were trained to detect various attack types. Experimental results demonstrated that packet-based NIDS can be equally effective as flow-based NIDS.

(Liu et al., 2022b) utilized raw packet data to construct a block sequence by analyzing the payload data within the packets. This approach captures both short-term and long-term dependency relationships among malicious bytes within the payload data. Similarly, (Yu et al., 2021) introduced a CNN-based IDS called PBCNN, which processes raw packet data to create a single image encompassing a certain number of packets belonging to a flow or session. This method exhibits high efficiency when analyzing contemporary data sets such as CICIDS2017 and CICIDS2018. (Hore et al., 2023b) conducted a comparative assessment of various autoencoder models for detecting anomalies within packet data. They introduced a framework for deploying an autoencoder-based NIDS tailored to packet data and proposed a novel metric to gauge reconstruction errors in packet-based autoencoders. (Premkumar et al., 2023) employed GNNs and graph embedding techniques to construct a context-aware NIDS. They explored various approaches to represent network data as graphs, both at the network flow and packet levels, using the CICIDS2017 data set. The outcomes demonstrated that incorporating context from the GNN model enhances the attack detection performance.

2.2 EVASION ATTACKS ON NIDS

Evasion attacks were first observed in computer vision by Szegedy et al., who demonstrated that minor image alterations could mislead deep neural networks (Szegedy et al., 2013). Evasion attacks on NIDS can be categorized based on the adversary's knowledge of the classifier (Oprea & Vassilev, 2023). Black-box attacks involve zero knowledge of the classifier, hyper-parameters, and training features. Gray-box attacks lack knowledge of the classifier but assume some understanding of preprocessing functions leading to the training feature set. White-box attacks assume complete

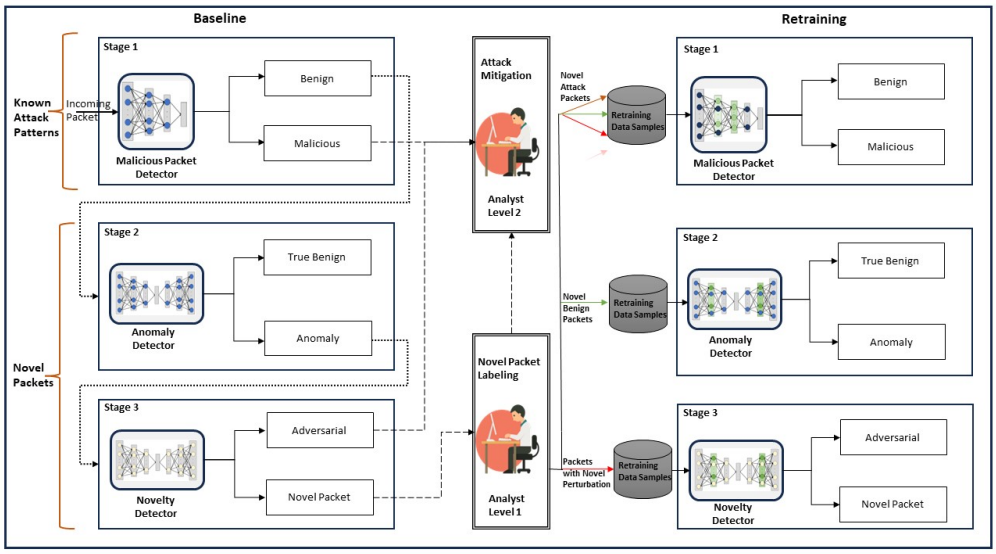

Figure 1: *Deep ResNIDS* - A multistage AI framework for novelty detection in network traffic

knowledge of the classifier and its training feature set. Adversarial attacks on NIDS involve manipulating data features to deceive the underlying machine learning models. Studies in literature have investigated modeling various types of adversaries and generating adversarial samples to deceive different types of NIDS. Flow-based adversarial attacks, which attempt to reverse engineer perturbations from network flow into actual packets, are impractical (Rosenberg et al., 2021). Hence, recent literature has focused on directly generating adversarial packets. (Homoliak et al., 2018) crafted gray-box attack samples using tools like NetEM and Metasploit. (Hashemi et al., 2019) used trial and error method to construct white-box attacks. (Kuppa et al., 2019) used manifold approximation (gray-box attack), (Han et al., 2021) used generative adversarial network (gray-box attack), and (Sharon et al., 2022) used long short-term memory-based method (black-box attack). (Hore et al., 2023a) developed a deep reinforcement learning (DRL) agent to craft the adversarial packets by constraining the perturbations such that the new packet maintains functionality and maliciousness.

To the best of our knowledge, no prior research has focused on developing a robust and resilient network intrusion detection mechanism capable of accurately identifying known attacks as well as detecting zero-day, OOD, and evasion attacks at granular packet-level network data. Additionally, the literature lacks a methodological approach for efficiently retraining NIDS as new attack patterns emerge within the network, without compromising the accuracy of detecting previously known attacks.

3 METHODOLOGY

Figure 1 shows an illustration of the multistage *Deep ResNIDS* framework. The initial stage consists of a DNN classifier responsible for identifying malicious packets. The second stage employs a DNN anomaly detector, which compares incoming benign packets (classified by the first stage DNN) against historical benign packets found in the network. The third stage employs a DNN novelty detector to identify packets exhibiting novel characteristics. The framework also incorporates a human-in-the-loop feedback mechanism, as found in a typical cyber operations environment in real-world (Ganesan et al., 2016; Shah et al., 2019; 2023). This process involves labeling novel packets following analyst investigation, leading to updates in the learning process of each DNN within the framework. Next, we describe the framework's components.

3.1 MALICIOUS PACKET DETECTOR

The forefront of the multistage framework houses the malicious packet detector. A DNN-based classification model is developed using a data set containing both benign and attack packets observed in the network. The objective of this

model is to identify known malicious packets while minimizing false positives. The formal description of this task is given below:

$$f_C(x_i, \theta_C) = P(y_i = Malicious \mid x_i, \theta_C), \tag{1}$$

where θ_C denotes the set of learned weight parameters of the classifier, x_i represents the i-th sample packet, and $f(.)$ stands for the estimator function. This function seeks to ascertain the probability of a packet, x_i, being categorized as malicious. The parameter values are estimated using a training data set of known labeled packets, S_{known}, comprising both benign and malicious instances. The successfully identified malicious packets from the model are forwarded to human analysts for appropriate action (see Figure 1).

3.2 ANOMALY DETECTOR

The anomaly detector in the second stage of the framework is developed to differentiate between true benign and novel packets, including zero-day, OOD, and adversarial attacks that evaded the first stage detector. A packet that passes the malicious packet detector with a classification label of benign is now passed through the anomaly detector. The anomaly detection is achieved using the encoder-decoder architecture that reconstructs the input packet data. A reconstruction error is then calculated, and a threshold is set to differentiate between previously seen benign packets and novel packets. The anomaly detection task can be formally defined as follows:

$$x'_i = f_{BAE}(x_i; \theta_{BAE}) \tag{2}$$

$$y_i = \begin{cases} Benign & \text{if } R_{BAE}(x_i, x'_i) \leq \tau_{BAE} \\ Anomalous & \text{otherwise} \end{cases} \tag{3}$$

In Equation 2, x'_i denotes the reconstructed version of the packet x_i by the encoder-decoder network, and θ_{BAE} are the set of parameter weights for the autoencoder model. The optimal weights are obtained by training the model on the set of known benign data, ${S_{known}}^{benign}$, where ${S_{known}}^{benign} \subset S_{known}$. In Equation 3, the reconstruction loss is computed using an error function $R_{BAE}(.)$ that measures the similarity between the original and the reconstructed sample. If the reconstruction loss is greater than the selected threshold, τ_{BAE}, the packet is identified as an anomaly to the true benign distribution in the network. The aim is to detect anomalies in the form of OOD attack packets misclassified as benign by the malicious packet detector, adversarial examples crafted to evade the malicious packet detector, and benign packets from unknown distributions.

3.3 NOVELTY DETECTOR

This stage follows the identification of anomalies in the preceding phase. The novelty detector employs an encoder-decoder network architecture and is trained using artificially generated adversarial data samples. These samples are crafted to successfully evade the detection capabilities of the malicious packet detection model. The primary objective of the novelty detector is to determine whether a given packet is a perturbation of a known attack crafted using a known toolchain or a genuinely novel packet. Any packets detected as novel are then directed to human analysts for careful examination and labeling. The novelty detector holds significant importance within the proposed framework for the following reasons. i) It effectively lightens the workload of human analysts by sifting out recognized adversarial attack packets. ii) It can identify unfamiliar adversarial perturbations that share similarities with the synthetically generated examples. iii) It serves as a barrier against the possibility of compromising the malicious packet detector. Introducing artificially crafted adversarial packets into the training data set of the malicious packet detector can significantly disrupt the decision boundary of the classification model. The novelty detection task can be formally defined as:

$$x'_i = f_{AAE}(x_i; \theta_{AAE}) \tag{4}$$

$$y_i = \begin{cases} Adversarial\ attack & \text{if } R_{AAE}(x_i, x'_i) \leq \tau_{AAE} \\ Novelty & \text{otherwise} \end{cases} \tag{5}$$

The workings of the novelty detection stage share great resemblance with anomaly detection since both employ autoencoders and rely on the reconstruction error metric to suggest if a particular anomalous packet data is truly novel. In Equation 4, x_i and x'_i carry the usual meaning of original and reconstructed packet data, respectively. θ_{AAE} denotes the learned parameter weights of the novelty detector model. In Equation 5, the threshold under which an anomalous

packet can be categorized as adversarial is set at τ_{AAE}. Any packet classified as novel is passed on to human analysts for further investigation and labeling. (Algorithm 1 in the Appendix outlines the steps involved in executing the baseline framework, as depicted in Figure 1.)

3.4 RETRAINING WITH ONE-SHOT LEARNING

For an organization, the continual enhancement of its intrusion detection mechanism is critical in protecting against evolving threats. Our framework introduces an innovative transfer learning technique to update the models within the three stages, utilizing newly labeled (novel) data samples, as depicted in Figure 1. Given that a NIDS maintains ongoing surveillance of network traffic, it is imperative to minimize the retraining time and its complexity. Equally important is the need to avoid compromising the models' ability to detect previously encountered attack patterns when incorporating new data attributes. With these considerations in mind, we present a one-shot learning approach to integrate the newly labeled packet data into the three DNNs.

Unlike other approaches in literature where either only select neurons from a layer are unfrozen for retraining, or additional layer(s) are added during retraining, our approach combines the strengths of both these transfer learning methods. In our hybrid approach, an extra layer is introduced, and certain neurons within the adjacent layer(s) are unfrozen. This combination enables the network to assimilate new information while also permitting the revision of previously acquired knowledge. Training a neural network in a normal setting can be mathematically expressed as:

$$f : R^n \rightarrow R^p$$

$$f(x) = O \circ g_K \circ ... \circ g_2 \circ g_1 \tag{6}$$

$$g_k(x) = a(W_k * x + b_k) \quad \forall k \in K \tag{7}$$

$$min_{w_k \forall k \in K} L(.) \tag{8}$$

Here, n and p are the input and output dimensions, respectively. $f(.)$ is the estimator function which is a composition of multivariate functions $g_1, g_2, ..., g_K$ and K is the set of hidden layers in the network as shown in Equation 6. $O(.)$ is the output function. Each function $g_k()$ is itself a multivariate function as well and can be defined by Equation 7, where W_k and b_k are weights and bias associated with the respective layer k. Equation 8 shows the objective of the neural network training. The objective is to minimize the loss function, $L(.)$ by adjusting the weight parameters of all the K layers. The weight parameters are updated during backpropagation, generally, using a gradient descent algorithm, by an amount $\lambda * \frac{\partial(L)}{\partial(w_{kj})}$, where λ is the learning rate and $\frac{\partial(L)}{\partial(w_{kj})}$ is the impact of the j-th neuron in the set of neurons J_k of the k-th layer on the loss function L. The new weight can be written as:

$$w_{kj} \leftarrow w_{kj} - \lambda \frac{\partial(L)}{\partial(w_{kj})} \quad \forall k \in K, j \in J_k \tag{9}$$

In Equation 9, w_{kj} is the weight associated with k-th layer's j-th neuron, which is updated for each layer, $k \in K$, and for every neuron, j, in those layers. In our proposed hybrid one-shot transfer learning approach, we only allow certain weights (old and new) to be updated. Hence, we revise the previous Equation 9 as follows:

$$w_{k'j'} \leftarrow w_{k'j'} - \lambda \frac{\partial(L)}{\partial(w_{k'j'})} \quad \forall k' \in K', j' \in J'_{k'} \tag{10}$$

where all $k' \in K'$ are the trainable layers and all $j' \in J'_{k'}$ are the trainable neurons in the k'-th trainable layer. It is to be noted that the total number of neurons that are trained using our hybrid approach of freezing weights is significantly smaller than the total number of neurons that need to be trained in a full network. The hybrid approach strikes a balance between accommodating novel data and preserving the existing knowledge, resulting in an updated intrusion detection mechanism that efficiently integrates new insights without sacrificing the detection performance for previously recognized attack patterns. (Figure 2 in the Appendix illustrates distinct transfer learning strategies that can be employed during the one-shot retraining process, and Algorithm 2 in the Appendix shows the steps in the one-shot transfer learning method.)

4 EXPERIMENTAL SETUP

In this section, we outline the numerical experiments performed to evaluate the proposed framework. We begin by describing the experimental data, followed by the baseline training and retraining of the models in the framework.

4.1 DATA DESCRIPTION

We conducted the numerical experiments using two publicly available popular data sets: CICIDS2017 (Sharafaldin et al., 2019) and CICIDS2018 (Sharafaldin et al., 2018). Notably, compared to other publicly available data sets like NSL-KDD (for Cybersecurity, 2009), KDD-CUP (Tavallaee et al., 2009), the CICIDS data sets are more recent, and consequently more pragmatic representations of modern network traffic (Hindy et al., 2020). With the availability of raw pcap files in the CICIDS data sets, the dependency on flow-level extracted features is reduced. The data sets contain different attack and benign communication in pcap files distributed across certain days of the week. To showcase the effectiveness of our framework, we create several training and testing data sets for the development of the models at the various stages of the framework. We processed these pcap files as described in (Hore et al., 2023a) to extract packet data from the raw pcap files.

We utilized the CICIDS2017 data set for initially training the different models in the framework. The data set contains the following attacks: Port Scan, DoS, DDoS, Infiltration, Brute Force, Web Attack, and Botnet. To emulate the limited knowledge of the baseline models in the framework, we use all attack types in our training, except Botnet. For each attack type, we bifurcate the available data into train and test with 20% of data retained for testing. Further, to emulate the OOD and zero-day attacks, we use attack data from the CICIDS2018 data set and Botnet attack data from the CICIDS2017 data set, respectively. For the retraining experiments, we split the CICIDS2018 and Botnet data into training and testing in a 1:9 ratio. We use the large hold-out set for testing and the smaller set for training. This setup is to emulate the one-shot learning aspect of the framework, in which a smaller number of data samples are used for retraining. We created a synthetically generated perturbed attack data set from the CICIDS2017 attack data samples using the methods in (Ghadermazi et al., 2023). This data set is used to train the baseline novelty detector model in the third stage of the framework. To show the framework's effectiveness against adversarial attacks, we crafted new perturbed samples that specifically evade the DNN-based malicious packet detector, using the DRL method in (Hore et al., 2023a). Note that this new adversarial data is created using a different toolchain (DRL method) when compared to the synthetically generated adversarial data used for training. We split the newly generated adversarial samples into a 1:9 train-test ratio. The train set of adversarial samples is used for retraining the third-stage autoencoder and the testing data set is used to evaluate its performance after retraining. We provide a GitHub link for the code files and data sets used in this study. [1]

4.2 BASELINE TRAINING OF THE FRAMEWORK

We first describe the setup for the malicious packet detector, followed by that of the anomaly detector and novelty detector. We obtained the best hyperparameter values after trying different values for training the malicious packet detector. The input size was selected to be the entire packet length of 1525. We used only three hidden layers, which gave us the opportunity to expand the network if needed during the retraining of the malicious packet detector with OOD and zero-day attacks. We also used a single neuron at the output layer with a sigmoid activation function due to the choice of binary cross-entropy loss. In total, we use 12 hidden layers in the entire encoder-decoder network of the anomaly detector. A combination of ReLU and Leaky ReLU activations are used in the network with mean absolute error (MAE) as loss and Adam as the optimizer.

The majority of packets in the CICIDS data sets are way shorter in length than the maximum segment size. From our analysis, we found out that the mean length of packets in the CICIDS2017 data set is under 200 bytes. Hence, we use the first 200 features (bytes) of the benign packet data to train the anomaly detector. The novelty detector training setup is very similar to the anomaly detector with the exception of the hidden layer architecture and the input size.

The input size is chosen to be 500 instead of 200 to facilitate the detection of adversarial perturbations encountered at the tail end or at the middle of the packet data. We observed that a deeper autoencoder yields lower reconstruction error for both OOD and adversarial attack samples, thereby missing out on the detection of novel attacks. This is because the diversity amongst the adversarial samples and the sample size is limited. Hence, we selected a shallower

[1] https://github.com/Anonymousgit2024/RESNIDS

autoencoder for the novelty detection task compared to the anomaly detector. (Table 4 and Table 5 in the Appendix show the hyperparameter values selected for the models.)

4.3 Retraining experiments

We performed the retraining with a one-shot transfer learning methodology. We retrained the malicious packet detector with zero-day, OOD, and adversarial attack packets. We used a maximum of 20% of data samples from different attack types along with a maximum of 20% in-distribution attack data to retrain the malicious packet detector. Similarly, to retrain the anomaly detector we perform one-shot learning with 25% of the available OOD benign samples. Lastly, the novelty detector is trained with the train set of adversarial samples, as described earlier in this section. We conducted comprehensive experiments to fix the location of the added trainable layer and the trainable neurons. (Table 6 in the Appendix shows the final architecture of the retrained models.)

5 Analysis of Results

5.1 Baseline performance of the framework

Table 2 shows the detection accuracy of the malicious packet detector. The tables breaks down the % accuracy obtained for each attack type and benign traffic. The classifier's performance is exceptional in detecting the in-distribution samples, as evidenced by a few recent studies (De Lucia et al., 2021; Bierbrauer et al., 2023). However, the attack detection accuracy significantly drops when the classifier is subjected to OOD and zero-day attack samples. We also noted that the classifier could not detect any test adversarial samples created using the DRL approach from the literature targeting the DNN classifier (Hore et al., 2023a).

We performed the anomaly detection task to detect true benign packets by setting a threshold based on the 95-percentile reconstruction error. We obtained the reconstruction threshold value of 0.09 by testing the trained autoencoder on the benign data samples from the CICIDS2017 test set, similar to the work in (Hore et al., 2023b). All packets passing through the anomaly detector exceeding this threshold value were considered anomalies. Table 2 shows the detection accuracy (in %) of the anomaly detector. It can be observed that the autoencoder detects anomalies (OOD samples of the various attack types) with a 99% accuracy. The anomaly detector is also able to detect the newest traffic pattern from the zero-day attack with more than 92% accuracy. Finally, the model is also adept at recognizing adversarial samples generated using a different toolchain (DRL) with an average detection rate of more than 97%. However, its detection accuracy decreases when subjected to benign packets from the CICIDS2018 data set. It is only able to detect 25% of them, resulting in many false positives. To counter this, we perform retraining with one-shot learning to redefine the new trend for benign traffic and present the retraining results in the following subsection.

The samples considered anomalous by the autoencoder model in the second stage are passed to the novelty detector in the final stage to identify if they are adversarial perturbations of known attacks. We found the reconstruction error value of 0.105 to effectively differentiate between known adversarial perturbations and novel packets. Table 2 shows the performance of the novelty detector against the various anomalous packets from the previous stage. The model is able to detect, on average, more than 89% of attack packets from the CICIDS2018 data set (OOD). Since the adversarial samples used for testing the framework were created from a different toolchain (DRL) than the ones used to train the autoencoder, the model identifies 75% of them as novel packets for the baseline performance of the framework. All the packets identified as novel are to be forwarded to human analysts for investigation and labeling. We emulate the presence of analysts and apply the true labels to these packets to demonstrate the retraining of the DNNs using these small number of novel samples identified at the end of the third stage in the framework.

5.2 Performance evaluation after one-shot learning

We evaluated various retraining techniques using the one-shot learning paradigm, as outlined in Appendix (refer to Figure 3). Experiments show that our proposed hybrid approach, with a significantly smaller number of trainable parameters (refer to Table 7 in the Appendix), can achieve superior accuracy, making the computation viable for real-time NIDS, compared to the other approaches from literature.

We evaluate the framework's performance against zero-day, OOD, and adversarial attack samples following retraining. Table 3 displays the first-stage malicious packet detector's performance. This classifier demonstrates exceptional accuracy, exceeding 98%, in detecting zero-day attacks (Botnet) from the CICIDS2018 data set, and it also identifies

Table 2: Baseline performance of the framework

Attack Type	Malicious Packet Detector				Anomaly Detector				Novelty Detector		
	In-distribution	Out-of-distribution	Zero-day attack	Adversarial Samples	In-distribution	Out-of-distribution	Zero-day attack	Adversarial Samples	Out-of-distribution	Zero-day attack	Adversarial Samples
PortScan	99.97	-	-	0	-	-	-	97.67	-	-	75
DoS	99.21	68.34	-		-	99	-		88	-	
DDoS	99.89	6.75	-		-	99	-		98	-	
Infiltration	99.79	19.19	-		-	99	-		95	-	
Brute Force	99.81	61.42	-		-	99	-		86	-	
Web Attack	99.92	35.35	-		-	99	-		81	-	
Botnet	-	-	38.78		-	-	92.61		-	99.02	
Benign	99.47	91.29	-		95	25	-		85	-	

Table 3: Retraining performance of the malicious packet detector

Attack type	In-distribution	Out-of-distribution	Zero-day	Adversarial Samples
Port Scan	99.94	-	-	97.03
DoS	97.85	99.49	-	
DDoS	99.34	99.94	-	
Infiltration	99.73	93.06	-	
Brute Force	99.18	98.45	-	
Web Attack	99.39	99.03	-	
Botnet	-	-	98.25	
Benign	99.06	96.25	-	

over 97% of the adversarial samples in the test set. Notably, the model's detection accuracy remains consistent for both in-distribution and OOD samples. Results from the second-stage anomaly detector, retrained with benign packets from the CICIDS2018 data set, reveal its ability to detect nearly 90% of OOD benign packets and 99% of in-distribution samples. After retraining, the novelty detector successfully identifies most of the adversarial attack packets. Only 0.01% of the adversarial samples, in contrast to the baseline's 75%, are flagged as novel packets for analyst inspection, indicating improved performance. We also performed an ablation study validating the effectiveness of the various components and their arrangement in the framework. The results are shown in the Appendix (F.1).

5.3 Comparing against recent state-of-the-art method from literature

Finally, we compare the *Deep ResNIDS* framework with a recent multi-stage NIDS (Verkerken et al., 2023). While the original method used flow-based data, we applied it to packet-level data for a direct comparison. The ResNIDS framework achieved an overall accuracy of over 99%, while the literature's framework only reached 38.15% accuracy when tested with CICIDS2017 data. The literature's method excelled in detecting benign traffic with 100% accuracy but struggled with attack samples due to the poor performance of the one-class support vector machine (OC-SVM). In contrast, the proposed ResNIDS framework had a slightly higher false positive rate with a benign detection accuracy of 94.57% but achieved 100% accuracy in detecting attack packets.

6 Conclusions and Future Directions

In this paper, we developed the *Deep ResNIDS* framework for novelty detection in network traffic, employing supervised, unsupervised, and transfer learning methods. Our experiments, conducted on publicly available network intrusion data sets, demonstrate the effectiveness of our framework in detecting in-distribution, OOD, zero-day, and adversarial attacks. Our study showcases a promising approach for defending organizations against evolving adversaries, thereby enhancing system robustness and resilience to novel attacks. Our main focus was on developing the novelty detection framework's ability to adapt quickly with limited samples. Future research could explore optimal retraining timing and sample sizes for DNNs to maintain accuracy for both seen and unseen samples. Additionally, future work can investigate the best strategy for allocating human resources to analyze and label novel samples.

Reproducibility Statement

To facilitate the reproducibility of both our proposed framework and the associated results, we have incorporated the following components within our submission: 1) To enable the replication of the underlying mechanisms of the proposed models within our framework, we have provided Algorithm 1 and Algorithm 2 in the Appendix A. 2) We have provided the hyperparameter values used for the baseline training and retraining experiments of the models in Table 4, Table 5, and Table 6 in the Appendix. 3) We have made the source code and data sets available for implementation in an anonymous GitHub link (https://github.com/Anonymousgit2024/RESNIDS). 4) Keeping the reproducibility of the results in mind, we have published the trained models and the data sets used for different experiments evaluating the proposed ResNIDS framework (as described in Section 4.1). We have also included an appropriate README file in the GitHub folder to navigate the code files and the data sets.

References

Zeeshan Ahmad, Adnan Shahid Khan, Cheah Wai Shiang, Johari Abdullah, and Farhan Ahmad. Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Transactions on Emerging Telecommunications Technologies*, 32(1):e4150, 2021.

Muhammad Shoaib Akhtar and Tao Feng. Deep learning-based framework for the detection of cyberattack using feature engineering. *Security and Communication Networks*, 2021:1–12, 2021.

Abdullah Aljumah. Iot-based intrusion detection system using convolution neural networks. *PeerJ Computer Science*, 7:e721, 2021.

Giuseppina Andresini, Annalisa Appice, Nicola Di Mauro, Corrado Loglisci, and Donato Malerba. Multi-channel deep feature learning for intrusion detection. *IEEE Access*, 8:53346–53359, 2020.

David A Bierbrauer, Michael J De Lucia, Krishna Reddy, Paul Maxwell, and Nathaniel D Bastian. Transfer learning for raw network traffic detection. *Expert Systems with Applications*, 211:118641, 2023.

Giampaolo Bovenzi, Giuseppe Aceto, Domenico Ciuonzo, Valerio Persico, and Antonio Pescapé. A hierarchical hybrid intrusion detection approach in iot scenarios. In *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, pp. 1–7, 2020. doi: 10.1109/GLOBECOM42002.2020.9348167.

Evan Caville, Wai Weng Lo, Siamak Layeghy, and Marius Portmann. Anomal-e: A self-supervised network intrusion detection system based on graph neural networks. *Knowledge-Based Systems*, 258:110030, 2022.

Michael J De Lucia, Paul E Maxwell, Nathaniel D Bastian, Ananthram Swami, Brian Jalaian, and Nandi Leslie. Machine learning raw network traffic detection. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications III*, volume 11746, pp. 185–194. SPIE, 2021.

Yasir Ali Farrukh, Irfan Khan, Syed Wali, David Bierbrauer, John A. Pavlik, and Nathaniel 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)*, pp. 58–67, 2022. doi: 10.1109/BDCAT56447.2022.00015.

Canadian Institute for Cybersecurity. NSL KDD Dataset. `https://www.unb.ca/cic/datasets/nsl.html`, 2009. [Online; accessed 19-August-2023].

Merna Gamal, Hala M Abbas, Nour Moustafa, Elena Sitnikova, and Rowayda A Sadek. Few-shot learning for discovering anomalous behaviors in edge networks. *Computers, Materials & Continua*, 69(2), 2021.

Rajesh Ganesan, Sushil Jajodia, Ankit Shah, and Hasan Cam. Dynamic scheduling of cybersecurity analysts for minimizing risk using reinforcement learning. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8 (1):1–21, 2016.

Jalal Ghadermazi, Ankit Shah, and Nathaniel Bastian. Towards real-time network intrusion detection with image-based sequential packets representation. *TechRxiv*, 2023. doi: https://doi.org/10.36227/techrxiv.23291588.v1.

Dongqi Han, Zhiliang Wang, Ying Zhong, Wenqi Chen, Jiahai Yang, Shuqiang Lu, Xingang Shi, and Xia Yin. Evaluating and improving adversarial robustness of machine learning-based network intrusion detectors. *IEEE Journal on Selected Areas in Communications*, 39(8):2632–2647, 2021.

Mohammad J Hashemi, Greg Cusack, and Eric Keller. Towards evaluation of nidss in adversarial setting. In *Proceedings of the 3rd ACM CoNEXT Workshop on Big DAta, Machine Learning and Artificial Intelligence for Data Communication Networks*, pp. 14–21, 2019.

Ke He, Dan Dongseong Kim, and Muhammad Rizwan Asghar. Adversarial machine learning for network intrusion detection systems: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 2023.

Hanan Hindy, David Brosset, Ethan Bayne, Amar Kumar Seeam, Christos Tachtatzis, Robert Atkinson, and Xavier Bellekens. A taxonomy of network threats and the effect of current datasets on intrusion detection systems. *IEEE Access*, 8:104650–104675, 2020.

Ivan Homoliak, Martin Teknos, Martín Ochoa, Dominik Breitenbacher, Saeid Hosseini, and Petr Hanacek. Improving network intrusion detection classifiers by non-payload-based exploit-independent obfuscations: An adversarial approach. *arXiv preprint arXiv:1805.02684*, 2018.

Soumyadeep Hore, Jalal Ghadermazi, Diwas Paudel, Ankit Shah, Tapas K Das, and Nathaniel D Bastian. Deep packgen: A deep reinforcement learning framework for adversarial network packet generation. *arXiv preprint arXiv:2305.11039*, 2023a.

Soumyadeep Hore, Quoc Nguyen, Yulun Xu, Ankit Shah, Nathaniel Bastian, and Trung Le. Empirical evaluation of autoencoder models for anomaly detection in packet-based nids. *TechRxiv*, 2023b. doi: https://doi.org/10.36227/techrxiv.24043608.v1.

Muhammad Imran, Noman Haider, Muhammad Shoaib, Imran Razzak, et al. An intelligent and efficient network intrusion detection system using deep learning. *Computers and Electrical Engineering*, 99:107764, 2022.

Kaiyuan Jiang, Wenya Wang, Aili Wang, and Haibin Wu. Network intrusion detection combined hybrid sampling with deep hierarchical network. *IEEE access*, 8:32464–32476, 2020.

Aditya Kuppa, Slawomir Grzonkowski, Muhammad Rizwan Asghar, and Nhien-An Le-Khac. Black box attacks on deep anomaly detectors. In *Proceedings of the 14th international conference on availability, reliability and security*, pp. 1–10, 2019.

Nguyen Tung Lam. Detecting unauthorized network intrusion based on network traffic using behavior analysis techniques. *International Journal of Advanced Computer Science and Applications*, 12(4), 2021.

Jin Lan, Jia Z Lu, Guo G Wan, Yuan Y Wang, Chen Y Huang, Shi B Zhang, Yu Y Huang, and Jin N Ma. E-minbatch graphsage: An industrial internet attack detection model. *Security and Communication Networks*, 2022, 2022.

Wenke Lee, Salvatore Stolfo, Philip Chan, Eleazar Eskin, Wei Fan, Matt Miller, S. Hershkop, and Junxin Zhang. Real time data mining-based intrusion detection. volume 1, pp. 89–100 vol.1, 02 2001. ISBN 0-7695-1212-7. doi: 10.1109/DISCEX.2001.932195.

Yuzhen Li, Renjie Li, Zhou Zhou, Jiang Guo, Wei Yang, Meijie Du, and Qingyun Liu. Graphddos: Effective ddos attack detection using graph neural networks. In *2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pp. 1275–1280. IEEE, 2022.

Chao Liu, Zhaojun Gu, and Jialiang Wang. A hybrid intrusion detection system based on scalable k-means+ random forest and deep learning. *Ieee Access*, 9:75729–75740, 2021.

Jiaxin Liu, Xucheng Song, Yingjie Zhou, Xi Peng, Yanru Zhang, Pei Liu, Dapeng Wu, and Ce Zhu. Deep anomaly detection in packet payload. *Neurocomputing*, 485:205–218, 2022a. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2021.01.146. URL `https://www.sciencedirect.com/science/article/pii/S0925231221016349`.

Jiaxin Liu, Xucheng Song, Yingjie Zhou, Xi Peng, Yanru Zhang, Pei Liu, Dapeng Wu, and Ce Zhu. Deep anomaly detection in packet payload. *Neurocomputing*, 485:205–218, 2022b.

Jielun Liu, Ghim Ping Ong, and Xiqun Chen. Graphsage-based traffic speed forecasting for segment network with sparse data. *IEEE Transactions on Intelligent Transportation Systems*, 23(3):1755–1766, 2022c. doi: 10.1109/TITS.2020.3026025.

Alina Oprea and Apostol Vassilev. Adversarial machine learning: A taxonomy and terminology of attacks and mitigations (draft). Technical report, National Institute of Standards and Technology, 2023.

Augustine Premkumar, Madeleine Schneider, Carlton Spivey, John Pavlik, and Nathaniel D Bastian. Graph representation learning for context-aware network intrusion detection. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications V*, volume 12538, pp. 82–92. SPIE, 2023.

Ishai Rosenberg, Asaf Shabtai, Yuval Elovici, and Lior Rokach. Adversarial machine learning attacks and defense methods in the cyber security domain. *ACM Computing Surveys (CSUR)*, 54(5):1–36, 2021.

Ankit Shah, Rajesh Ganesan, Sushil Jajodia, and Hasan Cam. A two-step approach to optimal selection of alerts for investigation in a csoc. *IEEE Transactions on Information Forensics and Security*, 14(7):1857–1870, 2019. doi: 10.1109/TIFS.2018.2886465.

Ankit Shah, Rajesh Ganesan, Sushil Jajodia, Hasan Cam, and Steve Hutchinson. A novel team formation framework based on performance in a cybersecurity operations center. *IEEE Transactions on Services Computing*, pp. 1–13, 2023. doi: 10.1109/TSC.2023.3253307.

Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp*, 1:108–116, 2018.

Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. A detailed analysis of the cicids2017 data set. In *Information Systems Security and Privacy: 4th International Conference, ICISSP 2018, Funchal-Madeira, Portugal, January 22-24, 2018, Revised Selected Papers 4*, pp. 172–188. Springer, 2019.

Yam Sharon, David Berend, Yang Liu, Asaf Shabtai, and Yuval Elovici. Tantra: Timing-based adversarial network traffic reshaping attack. *IEEE Transactions on Information Forensics and Security*, 17:3225–3237, 2022.

Pengfei Sun, Pengju Liu, Qi Li, Chenxi Liu, Xiangling Lu, Ruochen Hao, and Jinpeng Chen. Dl-ids: Extracting features using cnn-lstm hybrid network for intrusion detection system. *Security and communication networks*, 2020:1–11, 2020.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A Ghorbani. A detailed analysis of the kdd cup 99 data set. In *2009 IEEE symposium on computational intelligence for security and defense applications*, pp. 1–6. Ieee, 2009.

Miel Verkerken, Laurens D'hooge, Didik Sudyana, Ying-Dar Lin, Tim Wauters, Bruno Volckaert, and Filip De Turck. A novel multi-stage approach for hierarchical intrusion detection. *IEEE Transactions on Network and Service Management*, pp. 1–1, 2023. doi: 10.1109/TNSM.2023.3259474.

Haomin Wang and Wei Li. Ddostc: A transformer-based network attack detection hybrid mechanism in sdn. *Sensors*, 21(15):5047, 2021.

Yanqing Yang, Kangfeng Zheng, Bin Wu, Yixian Yang, and Xiujuan Wang. Network intrusion detection based on supervised adversarial variational auto-encoder with regularization. *IEEE access*, 8:42169–42184, 2020.

Ruizhe Yao, Ning Wang, Zhihui Liu, Peng Chen, and Xianjun Sheng. Intrusion detection system in the advanced metering infrastructure: a cross-layer feature-fusion cnn-lstm-based approach. *Sensors*, 21(2):626, 2021.

Sheng-lin Yin, Xing-lan Zhang, and Shuo Liu. Intrusion detection for capsule networks based on dual routing mechanism. *Computer Networks*, 197:108328, 2021.

Lian Yu, Jingtao Dong, Lihao Chen, Mengyuan Li, Bingfeng Xu, Zhao Li, Lin Qiao, Lijun Liu, Bei Zhao, and Chen Zhang. Pbcnn: packet bytes-based convolutional neural network for network intrusion detection. *Computer Networks*, 194:108117, 2021.

Yingwei Yu and Naizheng Bian. An intrusion detection method using few-shot learning. *IEEE Access*, 8:49730–49740, 2020.

A Appendix

B Additional Literature Review

B.1 Computer security incident response team (CSIRT) process

Organizations of various sizes have CSIRTs responsible for cybersecurity operations, including incident management, response, and mitigating the impact of cyber-attacks. The CSIRT is a unique combination of personnel and technology, working in tandem to protect the organization from cyber-attacks. The standard procedure followed by the CSIRT involves assigning NIDS-generated alerts to their cybersecurity analysts for the purpose of identifying and mitigating attacks. Analysts conduct two levels of alert inspections: primary (Level 1) and secondary (Level 2). Level 1 inspection focuses on identifying novel traffic samples, while Level 2 analysts conduct in-depth investigations and remediation of the attack incidents. The findings from these investigations are then documented to build threat intelligence.

B.2 DL-enabled NIDS based on flow data

Convolutional neural networks (CNNs), known for their spatial information-capturing capabilities, are widely used in image classification and object detection. They have also shown promise in intrusion detection systems (Lam, 2021; Aljumah, 2021; Akhtar & Feng, 2021; Gamal et al., 2021). Since flow-based data lack spatial relationships, CNNs are often combined with other DL models like recurrent neural networks (RNNs) and multi-head attention (MHA) networks in hybrid systems. In such setups, CNNs serve as feature extractors, while RNNs are used for classification (Jiang et al., 2020). Some notable models in the NIDS domain include Yu et al.'s (Yu & Bian, 2020) few-shot learning-based model, which utilizes DNN and CNN for feature extraction and dimension reduction. Yao et al. (Yao et al., 2021) designed an intrusion detection model for advanced metering infrastructure (AMI) by combining CNN and long short-term memory (LSTM) for feature fusion. Liu et al. (Liu et al., 2021) developed a two-stage NIDS using k-means and random forest for binary classification, followed by deep learning algorithms for further categorization. Wang et al. (Wang & Li, 2021) introduced DDosTC, a hybrid neural network combining self-attention mechanisms with CNN to detect DDoS attacks in software-defined networks. Additionally, Yin et al. (Yin et al., 2021) proposed a deep capsule NIDS with an attention mechanism to enhance feature extraction and prioritize impactful features on the model output.

AE is a DL method that employs unsupervised learning paradigm, operating on data samples without labels. In AE, the objective is to reconstruct inputs while minimizing the discrepancy between input and output. Yang et al. (Yang et al., 2020) introduced a NIDS framework called supervised variational autoencoder with regularization and deep neural network (SAVAER-DNN) for identifying less common and zero-day attacks. Their evaluation on NSL-KDD and UNSW-NB15 data sets revealed reduced accuracy in identifying minority attack categories. Andresini et al. (Andresini et al., 2020) introduced a multistage NIDS model that incorporates the concept of AE. This model includes a convolutional layer and two fully connected layers. It employs two separate AEs during an initial unsupervised phase, trained on normal and attack data to reconstruct samples. In the supervised stage, these reconstructed samples create an augmented data set, serving as input for a 1D-CNN. The model achieved superior performance compared to other DL models on data sets such as KDD Cup'99, UNSW-NB15, and CICIDS2017. Bovenzi et al. (Bovenzi et al., 2020) proposed a hierarchical multistage approach that integrates a multi-modal denoising autoencoder (DAE) in the first stage and soft output classifiers in the second stage. The soft output classifier identifies unknown attacks by applying a confidence threshold to predictions. This approach employs the open-set approach in multi-class classification, optimizing the confidence threshold for each known attack type. Similarly, Verkerken et al. (Verkerken et al., 2023) introduced an innovative multi-stage strategy for hierarchical intrusion detection, accommodating both binary and multiclass detection while minimizing latency and bandwidth demands and addressing known and zero-day attacks.

Graph neural networks (GNNs) and graph representation learning (GRL) have also been proposed for intrusion detection in NIDS (Li et al., 2022; Caville et al., 2022; Lan et al., 2022). Li et al.(Li et al., 2022) introduce GraphDDoS, which uses GNNs to identify low-rate and high-rate DDoS attacks by analyzing packet connections and flow relationships. GraphSAGE (Liu et al., 2022c) was developed to tackle scalability challenges in GNNs by selecting a set number of nodes from a given node's neighborhood, instead of considering the entire neighborhood. Caville et al. (Caville et al., 2022) made a significant enhancement to the GraphSAGE model by implementing it in a self-supervised manner. This novel approach, known as Anomal-E, utilizes the same graph structure (with flows and endpoints representing edges and nodes, respectively) without necessitating any labels. Lan et al. (Lan et al., 2022)

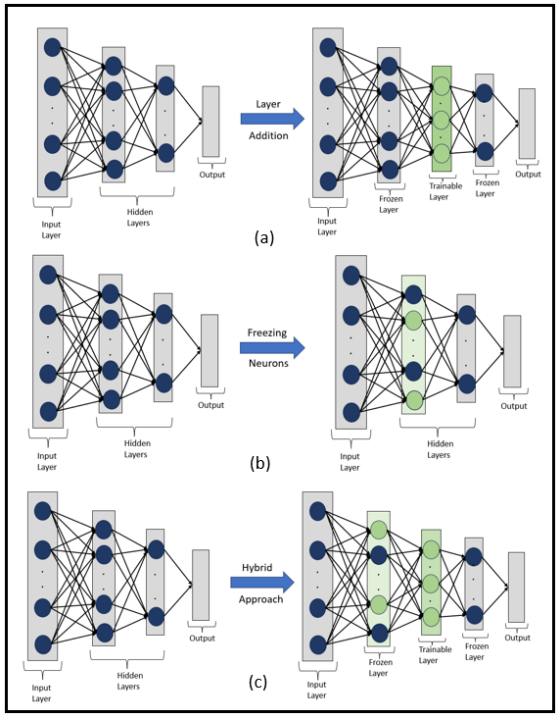

Figure 2: Retraining strategies for the deep neural networks

introduced another graph-based NIDS known as E-minBatch GraphSAGE. This approach incorporates a pre-sampling step prior to model training, leading to the creation of smaller graphs and improved scalability.

C Retraining strategies for the deep neural networks

Figure 2 illustrates distinct transfer learning strategies that can be employed during the one-shot retraining process. In Figure 2 (a), a fully trained network undergoes a transfer learning scheme involving the freezing of weights associated with trained layers. An additional layer is introduced, with only the newly added layer being allowed for training. Figure 2 (b) demonstrates a scenario where select neurons from a layer are unfrozen for retraining, without necessitating the addition of new weights to the pre-trained network. Our proposed hybrid approach, illustrated in Figure 2 (c), combines the strengths of both of the aforementioned transfer learning methods.

D Algorithms for Deep ResNIDS

E Numerical Experiments: Hyperparameters of the models

F Additional Results

F.1 Ablation study

In the subsequent analysis, we emphasize the value of distinct models within the *Deep ResNIDS* framework by employing an ablation study, a methodical approach aimed at gaining deeper insights into intricate systems. Our framework consists of three principal components: a malicious packet detector, an anomaly detector, and a novelty detector. The novelty detector, situated in the third stage, plays a pivotal role in sifting out adversarially manipulated instances from known attack categories within novel network packets. When this autoencoder is utilized in isolation within the framework, the result will entail the aggregation of all other packets—those unaffected by adversarial perturbations,

Algorithm 1: *Deep ResNIDS* algorithm

Input: Malicious packet detector parameters (θ_C), Anomaly detector parameters (θ_{BAE}), Novelty detector parameters (θ_{AAE})
Output: Decision regarding the incoming network traffic (packet classification and mitigation decision).

/* Set Novel benign traffic buffer B_{benign}, Novel malicious traffic buffer B_{attack}, Adversarial attack buffer B_{adv} to []. */
for *each incoming packet* **do**
    Extract normalized packet-level features
    Pass extracted packet, x_i, to the malicious packet detector, $f_C(.)$.
    if $f_C(x_i, \theta_C)$ == *'Malicious'* **then**
        Forward the communication for attack mitigation to analyst level 2
    end
    else
        Forward x_i to the anomaly detector, $f_{BAE}(.)$.
        if $f_{BAE}(x_i, \theta_{BAE})$ == *'Benign'* **then**
            Declare the communication as benign
        end
        else
            Forward x_i to the novelty detector, $f_{AAE}(.)$
            if $f_{AAE}(x_i, \theta_{AAE})$ == *'Novelty'* **then**
                Forward the communication for further investigation to analyst level 1
                if analyst level 1 feedback == *'Novel Benign'* **then**
                    Store the communication in B_{benign}
                end
                if analyst level 1 feedback == *'Novel Attack'* **then**
                    Store the communication in B_{attack}
                end
                if analyst level 1 feedback == *'Novel Adversarial Attack'* **then**
                    Store the communication in B_{adv}
                end
            end
            else
                Declare the communication as an adversarial attack
                Forward the communication for attack mitigation to analyst level 2
            end
        end
    end
end
if enough samples in B_{benign} and B_{attack} and B_{adv} **then**
    Retrain malicious packet detector, $f_C(.)$ using Algorithm 2
    Obtain $\theta_{C,retrained}$
    Set $\theta_C = \theta_{C,retrained}$
end
if enough samples in B_{benign} **then**
    Retrain anomaly detector, $f_{BAE}(.)$ using Algorithm 2
    Obtain $\theta_{BAE,retrained}$
    Set $\theta_{BAE} = \theta_{BAE,retrained}$
end
if enough samples in B_{adv} **then**
    Retrain anomaly detector, $f_{AAE}(.)$ using Algorithm 2
    Obtain $\theta_{AAE,retrained}$
    Set $\theta_{AAE} = \theta_{AAE,retrained}$
end
return *Network traffic classification and mitigation action*

Algorithm 2: Retraining with one-shot transfer learning algorithm

Input: Baseline malicious packet detector parameters (θ_C), Baseline anomaly detector parameters (θ_{BAE}), Baseline novelty detector parameters (θ_{AAE})
Output: Retrained malicious packet detector parameters ($\theta_{C,retrained}$) / Retrained anomaly detector parameters ($\theta_{BAE,retrained}$) / Retrained novelty detector parameters ($\theta_{AAE,retrained}$)

for *each model in Deep ResNIDS Framework* **do**
    Select h number of data points from the respective packet buffer (B_{benign}, B_{attack}, B_{adv})
    Add an additional hidden layer in the DNN classifier/autoencoder
    Fine-tune the position of the added layer for best performance
    Select certain perceptrons from the adjacent layers of the newly added layer to make them trainable
    Fine-tune the quantity and positions of the trainable perceptrons
    Retrain the models with the selected data points
end
return *Retrained malicious packet detector parameters $\theta_{C,retrained}$ / Retrained anomaly detector parameters $\theta_{BAE,retrained}$ / Retrained novelty detector parameters, $\theta_{AAE,retrained}$*

Table 4: Training hyperparameters for the malicious packet detector

Training Hyperparamters	Values
Input Size	1525
No. of Classes	2
No. of Hidden Layers	3
Hidden Layer Sizes	128,64,32
Activations	ReLU
Output Activation	Sigmoid
Loss	Binary Crossentropy
Optimizer	Adam
Metrics	Accuracy
Batch Size	1024
Epoch Size	50

Table 5: Training hyperparameters for the anomaly detector and the novelty detector

Training Hyperparamters	Anomaly Detector	Novelty Detector
	Values	Values
Input Size	200	500
Hidden Layer Architecture (Encoder)	150,150,75, 50,25,12	400,300,250, 125
Hidden Layer Architecture (Decoder)	12,25,50, 75,150,150	125,250,300, 400
Activations	Leaky ReLU, ReLU	Leaky ReLU, ReLU
Latent Dimension	6	62
Loss	MAE	MAE
Optimizer	ADAM	ADAM
Batch Size	256	256
Epoch Size	500	500

Table 6: Classifier and autoencoder architectures prior to and during one-shot transfer learning

Model	Hidden Layer Architecture		Location and Number of Trainable Neurons	Retraining Data
	Before	After		
Malicious Packet Detector	128,64,32	128,64,32,32	3th Layer (21) 4th Layer (32)	CICIDS2018, Botnet
Malicious Packet Detector	128,64,32,32	128,64,32,32,32	4th Layer (10) 5th Layer (32)	Adversarial Examples
Anomaly Detector	Encoder- 150,150,75,50, 25, 12 Decoder- 12,25,50,75,150,150	Encoder- 150,150,75,50,50, 25, 12 Decoder- 12,25,50,50,75,150,150	Encoder- 4th Layer (16) 5th Layer (50) 6th Layer (8) 7th Layer (4) Decoder- 1st Layer (4) 2nd Layer (8) 3rd Layer(16) 4th Layer (50)	OOD Benign
Novelty Detector	Encoder- 400,300,250,125 Decoder- 400,300,250,125	Encoder- 400,300,250,125,125 Decoder- 125,125,250,300,400	Encoder- 3rd Layer (83) 4th Layer (41) 5th Layer (125) Decoder- 1st Layer (125) 2nd Layer (41) 3rd Layer (83)	Adversarial Examples

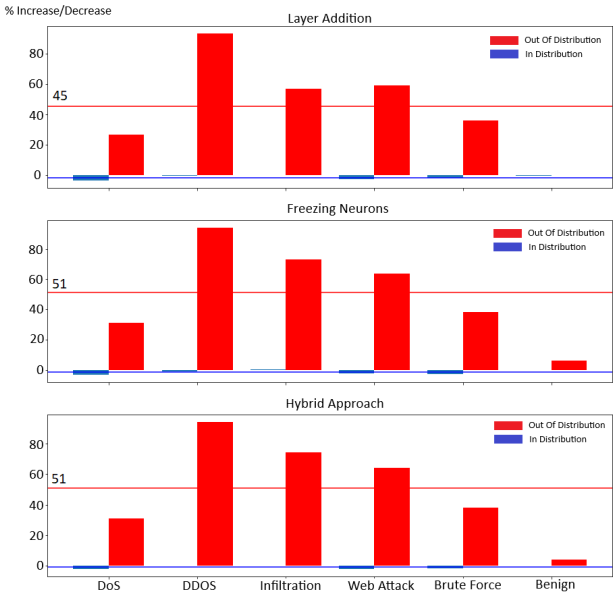

Figure 3: In-distribution and out-of-distribution sample detection evaluation after retraining

Table 7: Comparison of the number of trainable parameters for different retraining strategies

Retraining Method	Number of Trainable Parameters for Stage 1 DNN	Number of Trainable Parameters for Stage 2 DNN
Layer Addition	1056	3850
Freezing Neurons	10336	8346
Hybrid Approach	11392	13446
Naive Approach	2532847	189624

Table 8: Ablation study on the ResNIDS framework

Packet Type	Model Detection Accuracy	
	Malicious Packet Detector (Classifier)	Anomaly Detector (Autoencoder)
Port Scan	99.97	99.75
DoS	99.21	99.7
DDoS	99.89	89.88
Infiltration	99.79	99.96
Brute Force	99.81	97.94
Web Attack	99.92	99.93
Benign	99.47	95

including benign and malicious packets—under the umbrella of anomalies. However, such an approach is not practical as it would generate a high volume of false alarms due to the presence of benign traffic.

Next, we demonstrate the significance of each of the other two models by studying them individually. We presented the baseline performance of the malicious packet detector in Table 2, revealing that when operating on its own, the classifier overlooks a substantial proportion of OOD and zero-day attack instances. As previously reported, the anomaly detector in the second stage, as shown in Table 2, effectively detects a significant portion of samples that the classifier misses, thereby justifying this sequential arrangement. In Table 8, we provide a comparative analysis of the performance of the malicious packet detector and the anomaly detector using the same data set comprising benign and malicious samples. The results highlight that the classifier's accuracy outperforms in a significant majority of attack and benign network traffic scenarios. Furthermore, the classifier assists human analysts by providing appropriate labels for effective attack mitigation strategies, validating its position as the primary line of defense in network attack detection.