Early Network Intrusion Detection Enabled by Attention Mechanisms and RNNs

Taki Eddine Toufik Djaidja*, Bouziane Brik†, Sidi Mohammed Senouci*,
Abdelwahab Boualouache‡ and Yacine Ghamri-Doudane§
* *Université de Bourgogne* DRIVE Lab, Nevers, France;
{taki-eddine.djaidja,sidi-mohammed.senouci}@u-bourgogne.fr
†Computer Science Department, College of Computing and Informatics, Sharjah University, UAE;
bbrik@sharjah.ac.ae
‡*FSTM*, University of Luxembourg, Luxembourg; abdelwahab.boualouache@uni.lu
§*L3I Laboratory*, Univ. La Rochelle, France; yacine.ghamri@univ-lr.fr

***Abstract*—Current flow-based Network Intrusion Detection Systems (NIDSs) have the drawback of detecting attacks only once the flow has ended, resulting in potential delays in attack detection and increasing the risk of damage due to the infiltration of a greater number of malicious packets. Moreover, the delay provides attackers with an extended period of presence within the network, enabling them to execute subsequent attacks. To overcome this drawback, this work addresses the issue of early flow classification in NIDSs that incorporates a Deep Learning (DL) model. This model leverages Recurrent Neural Networks (RNNs), including Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), coupled with attention mechanisms. This strategic combination allows the system to harness the inherent sequential nature of packets within network flows, enhancing the efficiency of early flow classification. We conducted experiments on two up-to-date network intrusion datasets, namely CIC-IDS2017 and 5G-NIDD. Our findings demonstrate the effectiveness and accuracy of the proposed NIDS in classifying network flows. Additionally,our approach showcases its efficacy by promptly identifying and detecting attacks in their early stages without the need for flow termination. This results in a reduction in both the number of initial packets required for classification and the time needed for detection.**

***Index Terms*—Network Intrusion Detection Systems, Recurrent neural networks, Attention Mechanisms, Early intrusion detection.**

I. Introduction

Networking and telecommunication technologies, such as the Internet and mobile networks, have become essential components in numerous sectors, including transportation systems, smart cities, the Internet of Things (IoT), cyber-physical systems, and social media. With the increasing reliance on these technologies, especially after the introduction of 5G and its potential to connect almost everything everywhere, it has become crucial to research intelligent and effective mechanisms that guarantee network availability, confidentiality, and integrity. This is essential to ensure the security and privacy of individuals and organizations using these networking and communication technologies.

Network Intrusion Detection Systems (NIDSs) are an essential component of modern security systems; their role consists on detecting network-based attacks [1], [2]. NIDSs are typically placed at the level of routers, switches, and gateways, depending on the network topology. They analyze incoming and outgoing traffic in real-time, looking for patterns and behaviors that may indicate suspicious activities. NIDSs trigger alerts so that the mitigation component takes the appropriate countermeasures to prevent the intrusion. NIDSs rely on network packets to detect intrusions. However, those NIDSs that depend on deep packet inspection, where the packet payload is analyzed, present two main drawbacks: firstly, payload encryption makes it challenging to perform effective analysis, and secondly, analyzing the payload takes considerable time, resulting in significant computational overhead, especially in modern networks with high traffic volumes [3]. To address the aforementioned drawbacks, current NIDSs utilize packet headers [4]. This approach involves creating a sequence of exchanged packets between a source and a destination, which is known as a network flow. The flow (sequence) starts when the source sends its first packet and has two termination conditions: either a termination signal is sent in the packet, or a timeout is reached [5]. The sequence is then aggregated into a single data point that contains statistical information about the packets, such as their arrival time, length, direction, and flags. This method is more efficient, and its effectiveness in detecting threats at lower TCP/IP stack levels is demonstrated in the literature [6]. Several challenges have appeared in the field of NIDSs, including detection rate, false alerts rate, and zero days attacks [7]. Researchers are continuously exploring new levers and architectures, especially Deep Learning (DL), as it holds promising potential in addressing these challenges. DL-based approaches guarantee high detection rates, low false alarm rates, and the ability to detect zero-day attacks [8].

Flow-based NIDSs, referred to as conventional flow-based NIDSs in this paper, focus on the classification task solely upon meeting the flow termination condition. However, we have identified drawbacks in this approach, particularly concerning detection time, which we illustrate in the following two scenarios: (i) Firstly, a flow may potentially reveal abnormal traffic traces before its termination. In such cases, waiting for the flow to terminate becomes unnecessary and results in delayed detection. Given that we can infer the flow type earlier,

upon the appearance of an attack trace, waiting for termination unnecessarily prolongs the detection process. (ii) Secondly, if the last packet of a network flow is transmitted after a certain duration, before reaching the timeout and without including a termination signal, the NIDS overlooks the flow termination and must wait until the timeout condition is met to initiate the classification. This delay introduces a prolonged period before taking appropriate action against the malicious user.

The scenarios described above highlight a critical challenge that has not yet been thoroughly addressed in the literature. This challenge concerns the efficacy of NIDSs in detecting malicious traffic at the earliest possible stage. A NIDS must detect suspicious traffic in its nascent form to prevent the persistence of attackers and malicious traffic flows in the network.

To contribute to the advancement of NIDSs and confront the challenge of early attack detection, this paper introduces a preemptive approach designed to minimize the time required for identifying malicious traffic within the framework of flow-based NIDSs. Our innovative methodology focuses on the often-overlooked sequential nature of packets in network flows. To accomplish this, we leverage modern DL techniques, specifically Recurrent Neural Networks (RNN) and attention mechanisms.

This study addresses the issue of early attack detection in NIDSs, and as far as our knowledge extends, it is the first to delve into this aspect. The key contributions of our work can be summarized as follows:

- We propose a novel approach for flow-based NIDS. Our method embraces the sequential nature of packets within network flows and utilizes RNNs with attention mechanisms— state-of-art DL models in sequence processing.
- We demonstrate the efficacy of our proposed model in classification by employing up-to-date datasets.
- We conduct a comprehensive comparison to assess the capability of our model in addressing the early detection challenge of NIDSs. Our findings reveal two crucial aspects. First, our model significantly reduces the number of initial packets required for attack classification. Second, we showcase the early detection capabilities in terms of the time needed for the detection process. Both metrics are compared against the performance achievable by conventional NIDSs.

The remainder of this paper is organized as follows. Section II provides a comprehensive review of existing approaches in the field of NIDSs. Section III presents the proposed approach, including the preparation of data, a detailed description of the proposed Attention RNN model, and an explanation of the training phase. In Section IV, the datasets utilized in the research are presented. The classification performances of the proposed model are also demonstrated. Additionally, this section examines the early detection capabilities of the model. Finally, Section V concludes the paper.

II. Related Work

The problem of network intrusion detection has been a topic of great interest to researchers, and its development is still ongoing. Researchers constantly strive to develop new techniques and architectures to address the challenges and issues related to NIDSs.

Some NIDSs treat each network packet as a separate data point that must be analyzed and classified. This analysis can be based on the packet's header or payload. Many existing NIDSs methods use signatures, which typically involve testing whether the packet matches a header, port, or payload attack condition using regular expressions; please refer to [39] for details about signatures-based NIDSs. Recent works use advanced techniques, especially DL-based approaches. [9] experimented Machine Learning (ML) for a packet classification task where the authors proposed a tool that parses packet payloads into a fixed-size byte vector. The resulting payload vector is then labeled and tested using several ML algorithms, including Random Forest, K-Nearest Neighbors (KNN), Adaboost, Multi-Layer Perceptron (MLP), and convolutional neural network (CNN). In the same context, [12] proposed a novel approach for packet classification inspired by techniques used in natural language processing. The approach utilizes an embedding technique that learns a vector representation of the payload. A neural network is trained to learn byte embedding from the surrounding bytes in the same payload. The packet payload bytes are then aggregated to obtain a payload embedding for the packet, which is passed to a KNN model for final packet classification. However, packet-based approaches have several drawbacks. Firstly, payload encryption makes it difficult to perform effective analysis. Secondly, analyzing the payload takes considerable time. Moreover, these methods do not take any contextual information into account, which means they may fail to detect abnormal traffic that consists of a set of packets where each packet separately is benign, but the whole traffic is malicious.

To overcome these limitations, the new NIDSs are based on network flows. A network flow is a continuous packet stream representing a communication session between a source and a destination. A network flow begins when the source sends its first packet and terminates when one of two conditions is met: either a termination signal is sent in the packet, such as the FIN signal in TCP, or a timeout is reached. Usually, the timeout period is set to 60 seconds [40]. Most flow-based NIDSs utilize packet headers to extract relevant information. The data extracted from packet headers is then consolidated into a single data point containing statistical information about the packets. This statistical information include the packet's arrival time, length, direction, and flags. The aggregated vector may contain over 100 features. In the introduction section, we referred to this type of NIDS as conventional NIDS.

In the literature, a variety of models and classifiers have been developed for flow-based NIDSs, and recent research has focused on utilizing advanced ML/DL techniques. ML techniques are employed in various ways, including supervised learning. In binary-class supervised NIDS, the classifier is trained to learn and predict whether the flow is benign or malicious [13], [15], [17]. In contrast, in multi-class supervised IDS, the classifier is trained to predict the type of attack [18]–[21]. Another approach is semi-supervised learning, where the ML/DL model is trained only on benign traffic and is then

TABLE I
SUMMARY OF RELATED WORKS

Work	Approach			Time-series	Classification		ML/DL model	Dataset	Detection Time	Cons
	Packet	Flow			Binary	Multiclass				
		Headers	Payload							
[9]	✓		✓	✗		✓	KNN, Adaboost, MLP/CNN	[10], [11]	packet arrival	- NIDS Computing overhead/Payload encryption - Does not take context into consideration
[12]	✓		✓	✗		✓	E+KNN			
[13]		✓		✗	✓		LR/SVM/NB/RF	[14]	flow termination	-Waiting for flow termination condition.
[15]		✓		✗	✓		DT	[16]		
[17]		✓		✗	✓		MLP	[14]		
[18]		✓		✗		✓	S-EA	[14]		
[19]		✓		✗		✓	MLP	[14]		
[20]		✓		✗		✓	CNN	[10], [14]		
[21]		✓		✗	✓	✓	LR/DT/RF/ANN	[11]		
[22]		✓		✗	✓		AE/VAE	[10]		
[23]		✓		✗	✓		AE	[10], [11], [14]		
[24]		✓		✗		✓	Transformers	[10], [25]		Misusing of sequence based models (RNNs/Transformers)
[26]		✓		✗		✓	CNN+BILSTM	[10], [14]		
[27]		✓		✗	✓		GRU	[10], [14]		
[28]		✓		✗	✓		Transformers + CNN	[25]		
[29]		✓		✗	✓		CNN+RNN	[10]		
[30]		✓		✗	✓	✓	LSTM/biLSTM/GRU	[14], [31], [32]		
[33]		✓		✗		✓	LSTM	[34]		
[35]		✓	✓	✓		✓	CNN+LSTM	[10]		- Payload analysis (encryption)
[36]		✓	✓	✓	✓		Transformers	[10], [37]		- Detection time not discussed
[38]	/	/	/	✓	✓		Transformers	[10]	periodic (0.5s)	- Can not identify the attacker/ and the number of attacks in the network

* **E**: Embeedings, **LR**: Logistic Regression, **SVM**: Support Vector Machines, **AE**: AutoEncoder, **VAE**: Variational AE, **RF**: Random Forest, **DT**: Decesion Trees

used to determine whether a given flow belongs to the benign class. Any flow that does not fit this class is considered as an attack [22], [23].

Conventional flow-based NIDSs rely on flow termination conditions. As stated in the introduction, it is important to note that these approaches can result in increased delays. Moreover, when aggregating the network flow (sequence of packets) into a single data point, there is a loss of the temporal dimension in the sequence. This loss of temporal information can impact the performances of the model [35], [36]. It is worth noting that certain sequence-based DL models like RNNs and transformers have been misused in some flow-based NIDS works. Approaches discussed in [24], [26]–[30], [33] treat the flow aggregated vector, which is a data point with no temporal property, as a temporal sequence and feed it into a sequence-based DL model. We strongly believe that these approaches are conceptually flawed and should be reconsidered.

Recent proposals for flow-based NIDSs have suggested leveraging the sequence nature of network flows instead of flow aggregation to enhance the detection performance. The authors of [35] proposed a model that processes flow packets, including both the header and payload, in a 2D format consisting of a sequence of packets and their corresponding features. The model then passes this input to a CNN layer, with the resulting output being fed into a LSTM layer for flow classification. Additionally, [36] proposed a transformer-based model for flow classification, which considers both the header and payload of a packet. The payload is a vector containing the frequency counts of its bytes. The sequence of packets in the flow is fed to a transformer model for classification. [35], [36] demonstrated the effectiveness of using sequence-based DL models, with [36] in particular highlighting the advantages of using transformers, which rely on attention mechanisms to improve performance. However, both approaches rely on packet payload, making them prone to computational overhead and payload encryption issues. [38] attempted to address the detection time issue and highlighted that relying on termination timeouts for flows can cause delays in the detection process. The authors have proposed a framework that continuously monitors the state of the network to determine if it is experiencing any anomalies. The proposed approach periodically checks various network parameters to identify potential deviations from expected network behavior. The authors have considered 55 features, including statistics related to the number and length of packets, as well as the active flows in the network. The NIDS performs inference every 0.5 seconds to detect anomalies in the behavior. The framework leverages a transformer model, which performs binary classification to determine whether the current network behavior is normal or not. The model considers the network's previous states during the last 5 seconds. Through comparison with other models, the authors have demonstrated the effectiveness of their transformer-based approach. The proposed framework is effective at detecting the presence of an attack in the network, but it does not provide information about the attacker's identity or attack type. Moreover, the approach cannot ascertain whether a singular or multiple attacks are being executed, and it is unable to discern if a lone attacker or multiple attackers are involved. These limitations can make it challenging for the mitigation module to take the appropriate actions.

Table I presents a comprehensive overview of the works discussed earlier. It includes essential details such as the ML/DL models used, classification type (binary or multi-class), and the datasets utilized. Furthermore, the table provides a concise summary of the limitations associated with each of the works.

To overcome the aforementioned limitations, we introduce in the following a flow-based NIDS that focuses on packet headers and considers the sequential nature of network flows. Our approach capitalizes on the benefits offered by attention mechanisms, showcasing their effectiveness in early flow classification.

III. METHODOLOGY OVERVIEW

Our approach involves the processing of network packets to categorize network flows, which inherently exhibit a sequential pattern. To effectively manage and comprehend this sequential nature, we leverage RNNs and attention mechanisms. RNNs, introduced in [41], stand out as powerful DL models, holding significant potential for sequence classification tasks. To enhance their capabilities, we adopt subsequent variants, namely

Long Short-Term Memory (LSTM) [42] and Gated Recurrent Unit (GRU) [43], which have been developed to improve their overall effectiveness. These techniques prove particularly advantageous for processing sequential data, excelling in capturing patterns within a sequence.

Moreover, the incorporation of attention mechanisms enriches the model by providing additional contextual information. These mechanisms selectively focus on specific parts of the input sequence that are most relevant to the classification task, as expounded in [44]. The synergy between RNNs and attention mechanisms renders the proposed DL-based NIDS model novel. This approach holds promise for early flow detection, and we will delve deeper into this concept in the subsequent paragraph.

In contrast to traditional approaches that aggregate network flow data into a vector containing predefined statistical attributes defined by experts, our method diverges by adopting an unsupervised approach to embed the intrinsic characteristics of network flows. Instead of explicitly defining attributes, our deep learning model autonomously learns to represent the evolving sequence, progressively refining its understanding with each received packet. The RNN component is responsible for learning this representation. Furthermore, the attention mechanism creates a context vector that assigns a weight to each packet in the input flow, reflecting its importance to the current flow class.

Both outputs—the RNN output and the context vector, which together form the encoder module of our DL model—are then fed into the final part: a decoder that predicts the flow class.

The proposed DL model is designed to learn and classify network flows. The encoder module aids in identifying the packets or sequence within the flow that are most likely to indicate an attack. In deployment, following the occurrence of each packet, the encoder will perform a real-time encoding of the flow, then passing it to the decoder. This approach enables the early classification of network flows, allowing for the detection of attacks as soon as traces of intrusion appear.

In this section, we provide an overview of our proposed flow-based NIDS. We delve into the various components and steps involved in its implementation. We cover the data processing phase, the utilization of RNNs and attention mechanisms, as well as the DL-model training process.

A. Transformation of the network traffic data to network flows

Network security and monitoring data is collected and stored in the form of *pcap* files. PCAP stands for Packet Capture data, and it represents a file format used to store network traffic data that has been captured by a network sniffer tool. These *pcap* files serve as repositories for detailed information about individual network packets, including their headers and payloads. The *pcap* files are processed by various tools to extract aggregated flows and convert them into CSV format. Examples of such tools include Argus, NetFlow, CiCFlowmeter. These tools are commonly employed in different open-source datasets for network analysis. However, they primarily focus on providing aggregated flow-level information from *pcap* files. Although these tools are widely used in the literature for flow-based NIDSs, they are not suitable for NIDSs that rely on packets sequentiality within flows.

Given the importance of processing packet-level sequentiality for flow-based NIDSs, we have created a Python-based tool called *py_flows*. This tool segments the fully captured raw network traffic files (provided in many open-source datasets), into individual network traffic files. Each segmented file contains the packet sequences associated with their respective flow. A flow session is uniquely identified by the source IP address and port, destination IP address and port, and the protocol used. Traditionally, when flows surpass the timeout period, they are subdivided into multiple sub-flows. However, in our work, we take a different approach by preserving the entire session flow without splitting it. The flows labelling is conducted in a semi-manual manner. Open-source network intrusion datasets provide metadata on how the datasets were generated. These metadata include attacks information such as the attacker's IP address, the victim's IP address, the type of attack, and the time when the attack was initiated. The dataset's metadata are utilized for labeling the flow files generated by the *py_flows* tool. These labels serve the purpose of identifying, whether a flow is tagged as benign or corresponds to a specific type of attack. The dataset containing the labeled flows can be used to train NIDS classifiers in a supervised or semi-supervised manner.

B. Data preparation and pre-processing

The dataset generated from the previous step consists of a collection of labeled flows in pcap format. In our approach, we used ScaPy[1] library to read the packets within each flow and extract the header data. The extracted features are:

- *Packet relative time*: represents the arrival time of a packet (p_i) relative to the first packet (p_1) in the flow. This feature is expressed in seconds and calculated as $\text{time}(p_i) - \text{time}(p_1)$,
- *Inter arrival time*: refers to the time interval between the arrival of a packet (p_i) and its preceding packet (p_{i-1}) in the flow sequence,
- *Packet direction*: can be categorized as either forward or backward. Forward (fwd) packets are the ones sent by the flow initiator, determined by the source of the first packet in the sequence. Conversely, backward (bwd) packets refer to the ones sent by the destination,
- *Destination port*: the flow destination port,
- *Packet length*,
- *Packet payload length*,
- *Time-To-Live (TTL)*: field in the IP header specifying the number of routers that a packet can pass through before it is discarded,
- *Protocol*: IP protocol, TCP or UDP,
- *TCP flags*: control bits found in the TCP packet header. There are 8 flags (PSH,SYN,RST,FIN,ACK,URG,ECE,CWR) and please refer to [45] for details on TCP/IP networking.

[1] scapy.net

After extracting features from *pcap* files, the features are stored in a *csv* file, where each file represents a labeled sequence of the extracted features.

To prepare the data for the model training, we performed encoding and normalization techniques on the features. We used the z-score function to normalize the packet length and packet payload length. Additionally, since the TTL field is an integer ranging from 0 to 255, we normalized it using the min-max function to scale it within the range of 0 to 1. Regarding the destination port, which is a categorical type field, we decided to retain only the commonly used destination ports found in the dataset, such as 22, 80, 8080, and 5353. It is important to note that these ports may vary depending on the testbed used to generate the data and the servers/services deployed. After identifying the common ports, we proceeded to encode them. As for the remaining ports that were not part of the common set, we assigned them a default code. Due to the variable lengths of packets in flow sequences, we adopted a simplification approach by truncating the sequences to a maximum length, denoted as MAX LENGTH ($= 128$).

At this stage, the data is prepared as input to the NIDS model. The dataset consists of a collection of network flows, where each flow is represented as a tuple $< P, label >$. In this representation, $P = \{p_i\}/i \in [0, l]$ is the sequence of packet headers belonging to the flow, l is the length of the flow (number of packets) $l = ||P||/l \in [1, \text{MAX LENGTH}]$. A packet header, denoted as p_i, has the following structure:

$$p_i :<\text{rt, iat, fwd, bwd, d\_port, len\_p, len\_payload, ttl,}$$
$$\text{proto}, f_p, f_s, f_r, f_f, f_a, f_u, f_e, f_c >$$

(rt), (iat), (len_p), (len_payload), (ttl) represents the packet's relative time, inter arrival time, packet length, payload length and the time-to-live, respectively. fwd and bwd are boolean variables that represent the packet's direction, proto represents the type of IP protocol (TCP or UDP), The flags $f_p, f_s, f_r, f_f, f_a, f_u, f_e$ and f_c are boolean variables that indicate control bits of the TCP packet header. If the packet is a UDP packet, then these flags are set to false.

C. Deep Learning Model

This subsection provides a description of our DL model, which consists of three key modules: a projection layer, an encoder module using an RNN with attention mechanism, and a decoder layer. The model's architecture overview is illustrated in Figure 1.

1) Projection Layer: The model operates by accepting a sequence P as its input. To start, each individual input $p_i \in P$ is projected through a feed forward (FF) layer. The FF layer is followed by a dropout() function, serving as a regularization technique that aim to prevent over-fitting. The dropout layer randomly sets a fraction of the input tensor elements to zero, during training with a specified dropout probability, denoted as p_{dropout}.

The primary objective of this projection is to map the inputs to a higher-dimensional space, enabling richer representation and capturing more complex patterns. The projected input ($\widehat{p}_i$) has the shape (d,), and is calculated as follows:

$$\widehat{p}_i = \text{dropout}(W^{\top}_{\text{projector}} * p_i + b) \quad (1)$$

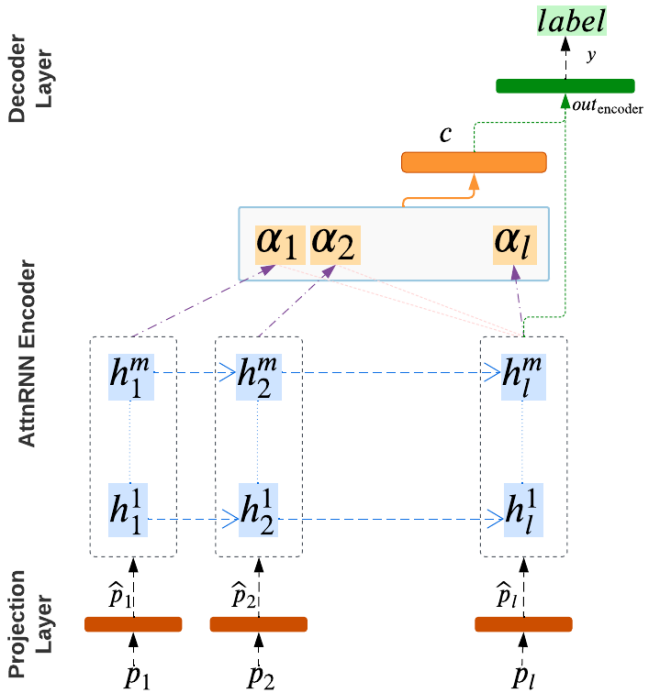

Fig. 1. Attention-RNN Model Architecture.

$W_{\text{projector}}$ is a learnable weight matrix of shape ($||p_i||, d$) and $b_{\text{projector}}$ is the bias vector of shape (d,).

2) AttnRNN Encoder: The purpose of this block is to process the projected packets within the flow sequence and encode it in the form of a latent representation, extracting meaningful and important information from the flow sequence. This representation will then be used by the decoder layer to classify the input sequence effectively. Our encoder building block consists of stacked RNN layers and an attention module, which we detail in the subsequent paragraphs.

*a) **RNN block**:* RNNs are a type of artificial neural network is designed to process sequential data [41]. The RNN component, depicted in blue in Figure 1, maintains an internal hidden state that enables it to retain and use information from previous data points within a sequence.

The RNN cell accepts as input a projected input $\widehat{p}_i$ with shape (d,). An RNN has a memory vector, known as the hidden state at step (i), with shape (d,) and an initial value h_0 of zero. The formula for calculating the hidden state h_i is as follows:

$$h_i = tanh(W^{\top} * \widehat{p}_i + b_x + V^{\top} * h_{i-1} + b_h) \quad (2)$$

Where W and V are learnable weight matrices of shapes (d, d). b_x and b_h are the bias vectors of shape (d,). The function $tanh()$ is the hyperbolic tangent activation function.

An RNN model may include multiple layers of RNN cells. In such a model, each layer ($j, j \leq m$), except for the first layer ($j = 1$), takes the output h_i^{j-1} of the previous layer as input.

In our contribution, we will utilize LSTM and GRU, which are two advanced variants of RNNs designed to enhance their performance, both approaches have been shown to be

effective in modeling long-term dependencies in sequential data [42], [43]. While both LSTM and GRU rely on the concept of hidden states, they also incorporate additional gates and internal states to facilitate their functioning. For the sake of paper clarity, we will not delve into the specifics of the LSTM and GRU networks. We recommend referring to [42], [43] for further details on these networks.

*b) **Attention module**:* Attention mechanisms are designed to improve the performance of neural networks when processing sequential data. When applied to a DL-based NIDS model, attention mechanisms enable the DL model to assign weights to the packets in the flow sequence, which helps to selectively focus on the most important and relevant packets during the classification process. The key idea behind attention is to create a context vector (c) and provides it as an additional input to the decoder. This context vector is calculated by combining the hidden states of the encoder h^m, which represent the input flow sequence, with attention weights that determine the importance of each hidden state h_i^m. The context vector is calculated as follows:

$$c = (\alpha * h^m)^\top \tag{3}$$

α is the vector of the attention weights, it has the shape of (1,l) where l is the flow sequence length, and its calculation is shown in equation 4; h^m is the hidden states of the RNN's last layer (m), $h^m = \{h_i^m / i \in [1, l]\}$. The context vector c has the shape of (d,).

$$\alpha = \{\text{softmax}(\alpha_i) / i \in [1, l]\} \tag{4}$$

α_i is the attention weight of the input p_i, it quantifies its importance and the relevance to predicted flow. There are several methods for calculating attention weights in the context of attention mechanisms. the commonly used methods include content base attention, additive attention, location attention, general attention, dot product attention, and scaled dot product attention, refer to [46] for further details.

Our DL model utilizes additive attention, a technique that was first introduced in [47]. This approach is robust in handling sequences of varying lengths and provides interpretable weight scores. The attention weights are parameterized by a feed-forward (FF) layer, which is jointly trained with the other components of the model. The attention weight α_i of the packet p_i is calculated as follows:

$$\alpha_i = v_{\text{attn}}^\top * tanh(W_{\text{attn}}[h_l^m; h_i^m]^\top) \tag{5}$$

Where v_{attn} and W_{attn} are learnable weight matrices of shapes $(d, 2d)$ and $(d,)$, respectively.

3) Decoder Layer: The last stage of the classification process involves decoding the encoded representation of the flow and inferring the flow's class. The output of the encoder building block, denoted as out_{encoder}, is passed through a feed-forward (FF) layer, the final output vector y is computed as follows:

$$y = U^\top * out_{\text{encoder}} + b_y \tag{6}$$

Where U is a learnable weight matrix and b_y is the bias vector.

In the case where the encoder does not include the attention module, out_{encoder} corresponds to the last hidden state h_l^m of the RNN. On the other case, if the encoder does have an attention module, then out_{encoder} is obtained by concatenating the RNN's output and the context vector (c), resulting in $out_{\text{encoder}} = [h_l^m, k]$. Consequently, the shape of U is either (d, k) or $(2d, k)$, depending on the specific configuration of the model, where k represents the number of classes.

The output vector y has a shape of (k,). Each element y_z in the output vector corresponds to the probability of the input flow belonging to the corresponding class z. The predicted class (predicted_label), is determined by the NIDS based on the index (z) that corresponds to the highest probability value in the vector y.

D. DL Model Training

To train our neural network model, we used the back-propagation algorithm to compute gradients and update the model's learnable parameters ($W_{\text{projector}}$, $b_{\text{projector}}$, W^j, V^j, v_{attn}, W_{attn}, U, b_y). We employed Stochastic Gradient Descent (SGD) as our optimization algorithm. SGD adjusts the model's parameters by taking steps proportional to the learning rate (lr) to control the updates. Cross-Entropy is used as a loss function, which computes the cross-entropy loss between the predicted output vector, y, and the target value, k. Additionally, to address the issue of gradient explosion that can occur in RNNs, we used the gradient clipping technique. Gradient clipping promotes more stable updates and aids in preventing the occurrence of infinity values in the gradients (pytorch NaN).

Furthermore, we opted for mini-batch gradient descent, where the training data is split into small batches of size (B), and the model parameters are adjusted after processing each batch. Given that the sequences in the dataset have different lengths, we apply right-padding with zeros ($p_0 = \overrightarrow{0}$) within each batch to align them with the maximum sequence length present in that batch.

When computing the attention context vector, we apply a masking technique to ignore the padded values (p_0) in the flow sequence. As a result, these padded points do not contribute to the calculation of attention weights for the context vector. Similarly, during the calculation of the loss, the padded values are also disregarded.

The explained model training process is repeated for (E) iterations.

IV. Performances Analysis

The following section delves into the performance analysis. It begins by describing the datasets used and then proceeds to illustrate the accuracy of the proposed DL models in classifying network intrusions, along with their early network intrusion detection capabilities.

A. Datasets

In order to showcase the strength and effectiveness of our proposed approach, we selected two modern and up-to date

TABLE II
CIC-IDS2017 DATASET OVERVIEW

Label	N. of packets	N. of flows	
Benign	5628248	248587	56.71%
Portscan	217530	107692	24.57%
DDoSLOIT	1265657	45168	10.30%
DoSHulk	2137508	14108	3.22%
DoSGoldenEye	106177	7574	1.73%
DoSSlowhttptest	37924	4212	0.96%
FTP-Patator	110256	3958	0.90%
DoSslowloris	45510	3835	0.87%
SSH-Patator	136073	2464	0.56%
Botnet	9862	735	0.17%

TABLE III
5G-NIDD DATASET OVERVIEW

Label	N. of packets	N. of flows	
BENIGN	624154	75625	40.51%
Goldeneye	900385	27467	14.71%
TCPConnect	20341	20032	10.73%
UDPScan	15921	15890	8.51%
Torshammer	555319	15837	8.48%
SYNscan	10064	10014	5.36%
SYNScan	10054	10009	5.36%
SYNflood	26458	7566	4.05%
Slowloris	61916	4247	2.27%

intrusion detection datasets that provide full network packet capture records and the corresponding labels. These datasets are CIC-IDS2017 [10] and 5G-NIDD [48]. For both datasets, we followed the process outlined in sub-section III-A to extract the flow sequences from the packet capture records. The labeling phase varies depending on the nature of the available metadata. To prepare the labeled flow sequences for model input, we proceeded to pre-process them as outlined in sub-section III-B. Tables II and III present the labels, as well as the number of packets in the original dataset and the extracted flows for datasets CIC-IDS2017 and 5G-NIDD, respectively. Each datset is divided into two sets, with 80% of the data allocated for training and 20% allocated for testing

1) CIC-IDS2017: The CIC-IDS2017 dataset was proposed by the Canadian Institute of Cybersecurity and has gained significant popularity in the literature. It has been referenced over 1000 times and is widely regarded as a reference dataset in the field of network intrusion detection. The CIC-IDS2017 dataset encompasses both benign network traffic and various types of attacks, such as Bot, DDoS, DoS, Patator, PortScan, Web Attack, Heartbleed, and Infiltration. Due to the limited representation of certain classes in the dataset, some of these classes were excluded from this contribution. The website of the dataset provides the necessary metadata for labeling, which includes information such as the attack lunch time and IP addresses of the attackers.

2) 5G-NIDD: The 5G Network Intrusion Dataset (5G-NIDD) is a recently developed dataset designed for intrusion detection in 5G networks. This dataset was generated using the 5G Test Network at the University of Oulu in Finland. The data in this dataset is collected from the gateways deployed in two base stations. It consists of various types of network traffic, including benign traffic and different types of attacks: ICMP Flood, UDP Flood, SYN Flood, HTTP Flood, Slowrate DoS, SYN Scan, TCP Connect Scan, and UDP Scan. The dataset is organized in such a way that each attack is represented by a separate (*.pcap*) file. Additionally, the IP address of the attacker is (10.41.150.68). This metadata was utilized to accurately label the extracted flow sequences within the dataset.

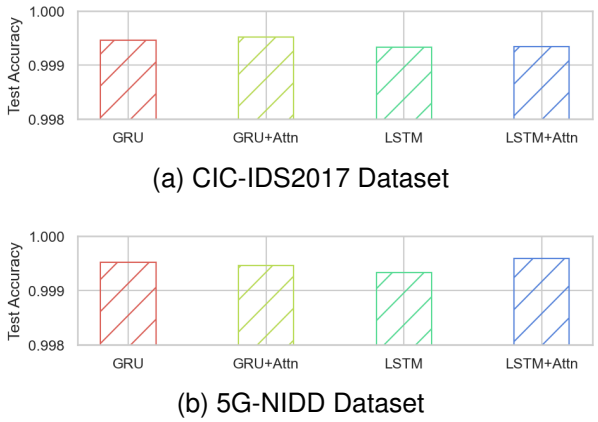

Fig. 2. NIDS classification accuracy on test data

B. Model Training Environment

We conducted the model training using the PyTorch framework on a system with the following configuration: Intel Core i7-10700, 32GB RAM, Nvidia RTX 3070. The training hyper-parameters are summarized in Table IV.

TABLE IV
TRAINING HYPER-PARAMETERS

parameter		value
DL model	d	128
	m	1
	k	**10** CIC-IDS2017/ **9** 5G-NIDD
	p_{dropout}	0.2
lr		$1 * 10^{-3}$
Train	Optimiser	SGD
	Loss	Cross-Entropy
	B	32
	E	10

C. Classification Accuracy

Figure 2a presents the test accuracy of various experimented models, including LSTM, LSTM with Attention, GRU, and GRU with Attention, on the CIC-IDS2017 dataset. The corresponding results for the 5G-NIDD dataset can be found in Figure 2b. It is evident from both datasets that the models achieved remarkably high accuracy rates (99%), which aligns with previous findings in the literature. Furthermore, the inclusion of attention mechanisms in the models resulted in slight performance enhancements. The utilization of attention context vector aided the learning process, contributing to these improvements. The GRU with Attention model demonstrated superior performance on the CIC-IDS2017 dataset, whereas the LSTM with Attention model outperformed others on the 5G-NIDD dataset. Therefore, we will select these respective models to conduct further evaluations on each dataset.

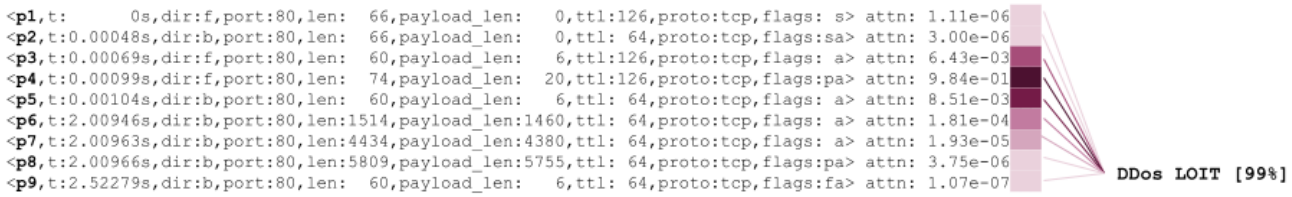

Fig. 3. Visualizing a network flow, its predicted class, and the attention weights.

D. Visualizing Attention

Figure 3 illustrates a flow sequence extracted from the CIC-IDS2017 dataset. This particular flow is a TCP flow, with a destination port of 80. It consists of nine (9) packets, and each packet within the flow is represented by its header. The total duration of the flow is 2.52 seconds. Notably, the LSTM attention model accurately classified this flow, assigning it the label DDosLOIT. The figure also depicts the attention weights (α). These weights are used to calculate the context vector (c) as shown in equation 3. The context vector (c) is then concatenated with the LSTM output of the last packet (h_9). This concatenated representation is subsequently fed into the decoder for the classification task.

We observe that the fifth packet (p_5) possesses the highest attention weight (α_5) among all the packets in the flow. This particular observation suggests that p_5 significantly influenced the decoder's classification decision, leading to the assignment of the DDosLOIT label. This finding suggests that the presence of (p_5) in the flow may contain crucial information related to the underlying attack. Additionally, the high attention weights observed in the flow, particularly in the case of packet p_5, can be interpreted as an indication of the presence of hidden traces of an attack within these specific packets.

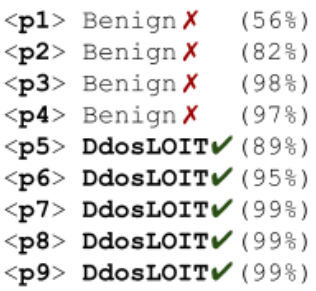

Fig. 4. Step-by-step predicted class of a network flow

To further explore the notion that some packets in the flow contains relevant traces of the produced labels, we employ a step-by-step classification. For each packet in the flow, denoted as p_t where $t \leq l$, we feed the hidden state (h_t) and the context vector (c) calculated using the attention weights (α_j) up to the t-th packet to the decoder y. This step-by-step classification allows us to observe how the encoder progressively represents the flow, and see how the decoder decodes this representation.

In this specific case, the observed flow analysis reveals interesting behavior. At the beginning of the flow, the NIDS initially classified the flow as Benign. The decoder indicated that the traffic appeared to be normal with a high level of confidence. This initial classification is expected since the first three packets contain the TCP handshake exchange, making it difficult to infer any attack solely from these packets. Therefore, the benign classification is reasonable given the limited information available in the initial packets. However, a shift occurred when the 5th packet (p_5) was observed. At this point, the NIDS reevaluated the flow and reclassified it as DDoS LOIT. Initially, the classification confidence was 89%, indicating a fair level of certainty. As the analysis progressed with each subsequent packet, the NIDS became increasingly confident in its classification, reaching a high confidence level of 99%.

These observations are consistent with our previous analysis of attention weights. When examining the attention weights across all the packets, we found that the 5th packet has the highest attention weight, indicating its significance in determining the label DDoSLOIT. The DL model successfully generated this label by decoding the information utilizing h_5 and the context vector c obtained through the attention mechanism, which incorporates the attention weights α_j for $j \leq 5$.

Our model successfully detected the attack immediately following the occurrence of packet (p_5). Early detection of the attack can be achieved if an alert is triggered after (p_5). As a result, the remaining four packets (from packet p_6 to p_9) of the flow would not take place. Additionally, the analyzed flow represents a TCP connection that is closed with a FIN flag. Conventional flow-based NIDS can only analyze this flow once it has been terminated, as indicated by the event (p_9) approximately 2.52 seconds after the initiation of the flow. In contrast, our approach enables detection to be carried out much earlier, precisely right after (p_5), at approximately 0.001 seconds.

E. Early detection capabilities

The previous sub-section highlighted the early flow detection capabilities in comparison to conventional NIDS from two perspectives: the requirement of fewer packets for attack detection and the ability to classify flows at an earlier stage. In this sub-section, we will generalize these observations based on the experimental datasets.

1) Packets required for attack classification: Figure 5 provides an overview of the performance concerning sufficient initial successive packets required for correct attack flow classification. The x-axis represents the flow length, which corresponds to the number of packets in each flow. Each

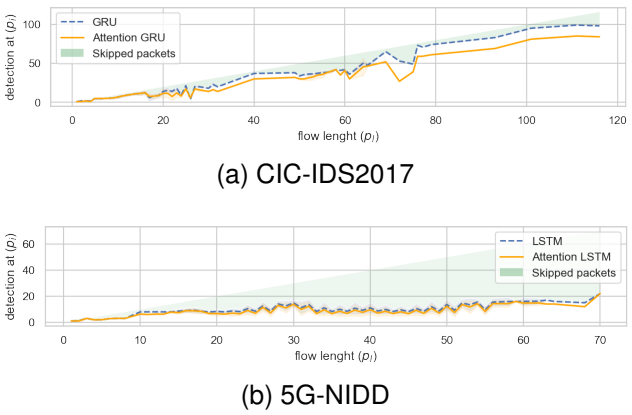

Fig. 5. **Number of initial packets required for correct attack classification.** AttnRNN (orange line), RNN (blue line), Conventional flow-based NIDS (green line)

element on the x-axis groups flows of the same length together. On the other hand, the y-axis represents the number of initial packets required for the NIDS to detect an attack.

Conventional flow based IDS analyses the flow using all its packets: hence, its graph is represented by the diagonal line. The dashed lines correspond to the RNN model, while the continuous line represents the RNN model with an attention mechanism. The RNN model is GRU for CIC-IDS2017 (displayed in 5a), and LSTM for 5G-NIDD (shown in 5b), chosen based on the accuracy metric illustrated in figure 2b.

In both scenarios, both the RNN and AttnRNN models demonstrated the capability to achieve early classification. The plots representing their respective results clearly illustrate data points positioned below the diagonal line, indicating their advancements compared to conventional flow-based IDS. Moreover, the Attn RNN model demonstrated a slight improvement over the RNN model, highlighting the effectiveness of attention mechanisms in enhancing performance.

In figure 5, the green-colored area represents the remaining skipped packets, signifying the number of packets ignored due to early attack detection. The green-colored packets represent malicious traffic that can be prevented from occurring within the network. The green area appears narrower in 5a compared to 5b. This observation indicates that in the 5G-NIDD dataset, attacks were more readily classified at an early stage, where the attack patterns were relatively easier to identify.

Figure 6 highlights attack labels, and the sub-figures provide further evidence that the proposed models (RNN and AttnRNN) can facilitate early detection, with certain attacks are detected sooner than others. This observation suggests that the characteristics of the attack and its corresponding traces vary depending on the specific type of attack. An interesting attack label worth noting is the udpscan. Our models demonstrate similar performance to the conventional model in detecting this type of attack. The udpscan attack, characterized by only two packets, is passive in nature. Interestingly, the proposed models cannot infer the attack based on the first packet, possibly due to its resemblance to normal traffic behavior. Consequently, both our RNN and attnRNN models require the two packets of the flow to identify the udpscan attack.

2) Detection time: Figure 7 illustrates the detection time, indicating the moment at which the model identifies a flow as an attack. It is essential to note that the model inference time has not been taken into account in this representation. The x-axis represents flows as data points, while the bars represent the duration of each flow. The green bars represent the detection time of the conventional flow-based IDS, while the orange bars represent the attnRNN model. Sub-Figure 7a showcases the performance results on the CIC-IDS2017 dataset, and Sub-Figure 7b displays the performance on the 5G-NIDD dataset.

The conventional flow-based NIDS performs detection once the flow is terminated, which occurs under two conditions: either when the FIN flag is sent or when the timeout period (60s) is reached. In both datasets (represented by sub-figures 7a and 7b), AttnRNN demonstrates a reduced detection time. This figure highlights the effectiveness of AttnRNN in reducing the required time (in seconds) to classify an attack flow. Minimizing the attack detection time can effectively restrict the duration of an attacker's presence within the network.

Figure 8 displays the detection results for specific attack labels from both datasets. While AttnRNN effectively reduces the number of packets in the doshulk attack (shown in sub-figure 7a) compared to conventional NIDS, both approaches show similar detection times. This is mainly attributed to the extremely short flow duration (around 1 second) and the small inter-arrival time between packets within the flow.

In the case of a UDP scan attack (shown in sub-figure 7b), conventional IDS systems wait for the timeout period as the flow is UDP. As mentioned previously, all approaches require two packets of the flow for accurate detection. However, AttnRNN demonstrates the ability to detect the UDP scan immediately after the occurrence of the second packet, whereas conventional NIDS systems have to wait until the end of the flow, resulting in a significantly longer detection time (around 60 seconds compared to just a few milliseconds).

To summarize this subsection, our proposed model demonstrates the capability to promptly classify malicious flows upon the emergence of an attack trace. As discussed in paragraph IV-E1, our model can reduce the number of packets required for analysis. Furthermore, as illustrated in paragraph IV-E2, our model significantly decreases the time needed for flow detection. These findings related to our model's capabilities offer multiple advantages, including minimizing the impact of attacks, reducing computational overhead associated with analyzing skipped packets, improving attack response time, and preventing further malicious activities within the network.

V. Conclusion

In this research work, we addressed the challenge of early detection of network intrusions. We introduced a novel NIDS that capitalizes on packet headers. Our innovative NIDS integrates attention mechanisms and RNNs, harnessing the sequential nature inherent in network flows. The efficacy of our approach was thoroughly assessed through experiments conducted on two contemporary network intrusion datasets.

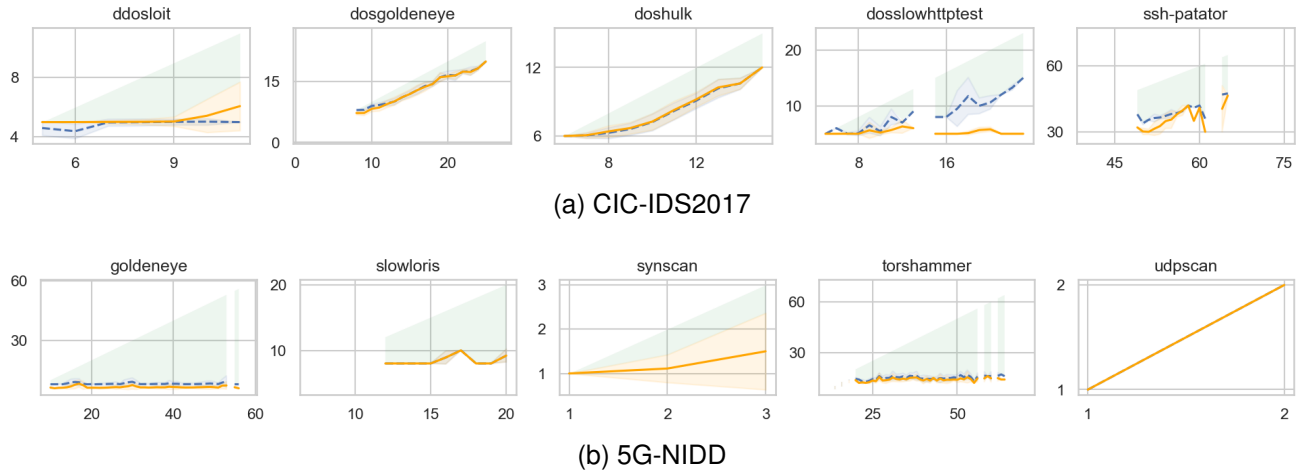

Fig. 6. **Number of initial packets required for correct attack classification of a selected attack classes.** AttnRNN (orange line), RNN (blue line), Conventional flow-based NIDS (green line)

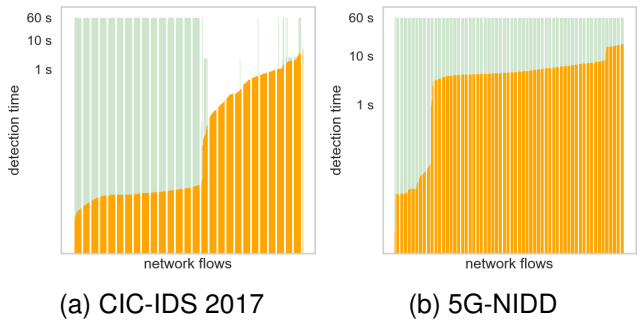

Fig. 7. **Overall Attack Detection Time.** AttnRNN (orange bars), Conventional flow-based NIDS (green bars)

Furthermore, we delved into the early detection capabilities of our model, highlighting its advantages from two perspectives: (*i*) reducing the initial required packets for flow classification, and (*ii*) minimizing the time needed for detection compared to existing flow-based NIDSs approaches. The proposed NIDS exhibits significant potential in fortifying the security and resilience of networks across diverse environments, including the evolving landscapes of 5G and beyond networks. In our ongoing research, we aim to extend our proposed NIDS by delving into several promising avenues, including the validation of our NIDS in real-world network environments, using different datasets to assess its performance, scalability, and effectiveness.

Acknowledgments

This work was supported by the 5G-INSIGHT bilateral project (ID: 14891397) / (ANR-20-CE25-0015-16), funded by the Luxembourg National Research Fund (FNR), and by the French National Research Agency (ANR).

References

[1] A. Fuchsberger, "Intrusion detection systems and intrusion prevention systems," *Information Security Technical Report*, vol. 10, no. 3, pp. 134–139, Jan. 2005. [Online]. Available: https://doi.org/10.1016/j.istr.2005.08.001

[2] M. Garuba, C. Liu, and D. Fraites, "Intrusion techniques: Comparative study of network intrusion detection systems," in *Fifth International Conference on Information Technology: New Generations (itng 2008)*, 2008, pp. 592–598.

[3] H. Alaidaros, M. Mahmuddin, A. Al Mazari *et al.*, "An overview of flow-based and packet-based intrusion detection performance in high speed networks," in *Proceedings of the International Arab Conference on Information Technology*, 2011, pp. 1–9.

[4] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Computers & Security*, vol. 86, pp. 147–167, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016740481930118X

[5] M. F. Umer, M. Sher, and Y. Bi, "Flow-based intrusion detection: Techniques and challenges," *Computers & Security*, vol. 70, pp. 238–254, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404817301165

[6] R. Di Pietro and L. V. Mancini, *Intrusion detection systems*. Springer Science & Business Media, 2008, vol. 38.

[7] D. Chou and M. Jiang, "A survey on data-driven network intrusion detection," *ACM Computing Surveys*, vol. 54, no. 9, pp. 1–36, Oct. 2021. [Online]. Available: https://doi.org/10.1145/3472753

[8] J. Lansky, S. Ali, M. Mohammadi, M. K. Majeed, S. H. T. Karim, S. Rashidi, M. Hosseinzadeh, and A. M. Rahmani, "Deep learning-based intrusion detection systems: A systematic review," *IEEE Access*, vol. 9, pp. 101 574–101 599, 2021.

[9] 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)*, 2022, pp. 58–67.

[10] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proceedings of the 4th International Conference on Information Systems Security and Privacy*. SCITEPRESS - Science and Technology Publications, 2018. [Online]. Available: https://doi.org/10.5220/0006639801080116

[11] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 Military Communications and Information Systems Conference (MilCIS)*, 2015, pp. 1–6.

[12] M. Hassan, M. E. Haque, M. E. Tozal, V. Raghavan, and R. Agrawal, "Intrusion detection using payload embeddings," *IEEE Access*, vol. 10, pp. 4015–4030, 2022.

[13] M. C. Belavagi and B. Muniyal, "Performance evaluation of supervised machine learning algorithms for intrusion detection," *Procedia Computer Science*, vol. 89, pp. 117–123, 2016, twelfth International Conference on Communication Networks, ICCN 2016, August 19– 21, 2016, Bangalore, India Twelfth International Conference on Data Mining and Warehousing, ICDMW 2016, August 19-21, 2016, Bangalore, India Twelfth International Conference on Image and Signal Processing, ICISP 2016, August 19-21, 2016, Bangalore, India. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S187705091631081X

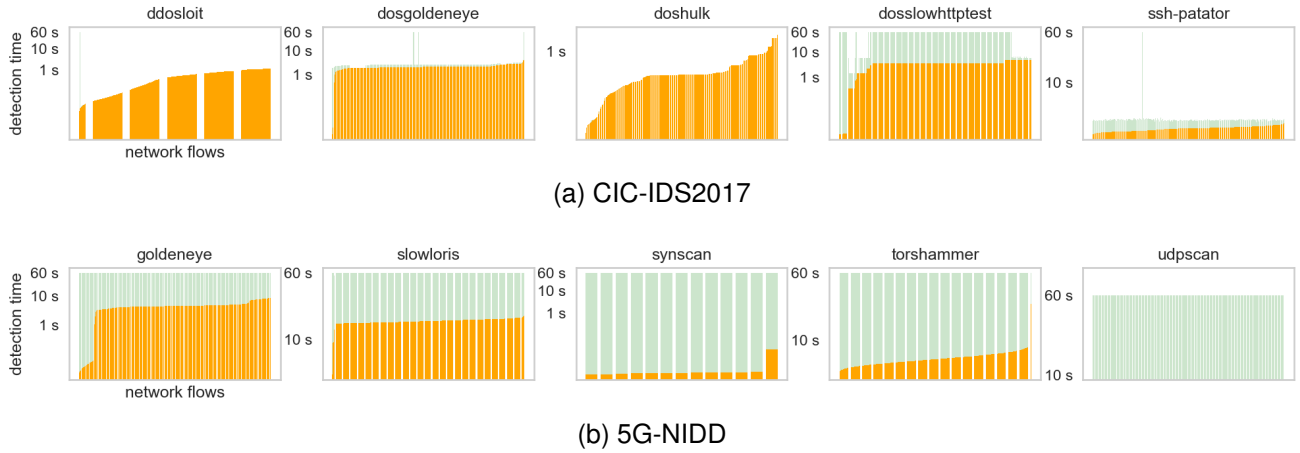

Fig. 8. **Attack Detection Time for a set of Labels.** AttnRNN (orange bars), Conventional flow-based NIDS (green bars) of a selected attack classes

[14] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009, pp. 1–6.

[15] I. H. Sarker, Y. B. Abushark, F. Alsolami, and A. I. Khan, "Intrudtree: A machine learning based cyber security intrusion detection model," *Symmetry*, vol. 12, no. 5, 2020. [Online]. Available: https://www.mdpi.com/2073-8994/12/5/754

[16] "Network intrusion detection dataset," https://www.kaggle.com/datasets/sampadab17/network-intrusion-detection.

[17] A. Iqbal, , and S. Aftab, "A feed-forward and pattern recognition ANN model for network intrusion detection," *International Journal of Computer Network and Information Security*, vol. 11, no. 4, pp. 19–25, Apr. 2019. [Online]. Available: https://doi.org/10.5815/ijcnis.2019.04.03

[18] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41–50, 2018.

[19] Akashdeep, I. Manzoor, and N. Kumar, "A feature reduced intrusion detection system using ann classifier," *Expert Systems with Applications*, vol. 88, pp. 249–257, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417417304748

[20] F. Yan, G. Zhang, D. Zhang, X. Sun, B. Hou, and N. Yu, "TL-CNN-IDS: transfer learning-based intrusion detection system using convolutional neural network," *The Journal of Supercomputing*, May 2023. [Online]. Available: https://doi.org/10.1007/s11227-023-05347-4

[21] A. R. Bahlali and A. Bachir, "Machine learning anomaly-based network intrusion detection: Experimental evaluation," in *Advanced Information Networking and Applications*. Springer International Publishing, 2023, pp. 392–403. [Online]. Available: https://doi.org/10.1007/978-3-031-28451-9_34

[22] S. Zavrak and M. İskefiyeli, "Anomaly-based intrusion detection from network flow features using variational autoencoder," *IEEE Access*, vol. 8, pp. 108 346–108 358, 2020.

[23] B. Min, J. Yoo, S. Kim, D. Shin, and D. Shin, "Network anomaly detection using memory-augmented deep autoencoder," *IEEE Access*, vol. 9, pp. 104 695–104 706, 2021.

[24] Z. Wu, H. Zhang, P. Wang, and Z. Sun, "Rtids: A robust transformer-based approach for intrusion detection system," *IEEE Access*, vol. 10, pp. 64 375–64 387, 2022.

[25] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy," in *2019 International Carnahan Conference on Security Technology (ICCST)*. IEEE, Oct. 2019. [Online]. Available: https://doi.org/10.1109/ccst.2019.8888419

[26] K. Jiang, W. Wang, A. Wang, and H. Wu, "Network intrusion detection combined hybrid sampling with deep hierarchical network," *IEEE Access*, vol. 8, pp. 32 464–32 476, 2020.

[27] T. A. Tang, D. McLernon, L. Mhamdi, S. A. R. Zaidi, and M. Ghogho, "Intrusion detection in SDN-based networks: Deep recurrent neural network approach," in *Deep Learning Applications for Cyber Security*. Springer International Publishing, 2019, pp. 175–195. [Online]. Available: https://doi.org/10.1007/978-3-030-13057-2_8

[28] H. Wang and W. Li, "Ddostc: A transformer-based network attack detection hybrid mechanism in sdn," *Sensors*, vol. 21, no. 15, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/15/5047

[29] M. A. Khan, "Hcrnnids: Hybrid convolutional recurrent neural network-based network intrusion detection system," *Processes*, vol. 9, no. 5, 2021. [Online]. Available: https://www.mdpi.com/2227-9717/9/5/834

[30] I. Ullah and Q. H. Mahmoud, "Design and development of rnn anomaly detection model for iot networks," *IEEE Access*, vol. 10, pp. 62 722–62 750, 2022.

[31] H. Hindy, C. Tachtatzis, R. Atkinson, E. Bayne, and X. Bellekens, "Mqtt internet of things intrusion detection dataset," 2020. [Online]. Available: https://ieee-dataport.org/open-access/mqtt-internet-things-intrusion-detection-dataset

[32] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset," 2018.

[33] S. Nayyar, S. Arora, and M. Singh, "Recurrent neural network based intrusion detection system," in *2020 International Conference on Communication and Signal Processing (ICCSP)*, 2020, pp. 0136–0140.

[34] "The caida ucsd "ddos attack 2007" dataset," https://www.caida.org/catalog/datasets/ddos-20070804_dataset.

[35] P. Sun, P. Liu, Q. Li, C. Liu, X. Lu, R. Hao, and J. Chen, "Dl-ids: Extracting features using cnn-lstm hybrid network for intrusion detection system," *Security and Communication Networks*, vol. 2020, pp. 1–11, 08 2020.

[36] X. Han, S. Cui, S. Liu, C. Zhang, B. Jiang, and Z. Lu, "Network intrusion detection based on n-gram frequency and time-aware transformer," *Computers & Security*, vol. 128, p. 103171, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404823000810

[37] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Computers & Security*, vol. 31, no. 3, pp. 357–374, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404811001672

[38] M. Tan, A. Iacovazzi, N.-M. M. Cheung, and Y. Elovici, "A neural attention model for real-time network intrusion detection," in *2019 IEEE 44th Conference on Local Computer Networks (LCN)*, 2019, pp. 291–299.

[39] S. Kumar, "Survey of current network intrusion detection techniques," http://www.cse.wustl.edu/~jain/cse571-07/ftp/ids/.

[40] "Cisco netflow configuration," https://www.cisco.com/c/dam/en/us/td/docs/security/stealthwatch/netflow/Cisco_NetFlow_Configuration.pdf.

[41] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, and S. Valaee, "Recent advances in recurrent neural networks," 2018.

[42] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[43] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014.

[44] Z. Niu, G. Zhong, and H. Yu, "A review on the attention mechanism of deep learning," *Neurocomputing*, vol. 452, pp. 48–62, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S092523122100477X

[45] "Transmission control protocol (TCP)," Tech. Rep., Aug. 2022. [Online]. Available: https://doi.org/10.17487/rfc9293

[46] L. Weng, “Attention? attention!” *lilianweng.github.io*, 2018. [Online]. Available: https://lilianweng.github.io/posts/2018-06-24-attention/
[47] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” 2016.
[48] S. Samarakoon, Y. Siriwardhana, P. Porambage, M. Liyanage, S.-Y. Chang, J. Kim, J. Kim, and M. Ylianttila, “5g-nidd: A comprehensive network intrusion detection dataset generated over 5g wireless network,” 2022.