ISSN: 2993-5377

Research Article

Open Access Journal of Applied Science and Technology

Enhancing Intrusion Detection Systems with Transformer Models

Mohamed Amine Ouni[1], Farah Jemili[2*] and Ouajdi Korbaa[2]

[1]University of Sousse, EPI-International Multidisciplinary School, Sousse, Tunisia

[2]University of Sousse, ISITCom, MARS Laboratory, LR17ES05, H. Sousse, Tunisia

[*]Corresponding Author
Farah Jemili, University of Sousse, ISITCom, MARS Laboratory, LR17ES05, H. Sousse, Tunisia.

Submitted: 2025, Nov 12; **Accepted:** 2025, Dec 30; **Published:** 2026, Jan 16

Citation: Ouni, M. A., Jemili, F., Korbaa, O. (2026). Enhancing Intrusion Detection Systems with Transformer Models. *OA J Applied Sci Technol, 4*(1), 01-14.

Abstract

In the dynamic field of network security, the continuous emergence of cyber threats demands advanced technologies that provide adaptive and scalable defense mechanisms. This paper introduces a novel approach to Intrusion Detection Systems (IDS) using Transformer models. Traditional IDS methods struggle with scalability and adapting to new threats, whereas our Transformer-based architecture leverages the model's scalability and parallel processing capabilities for real-time analysis of large volumes of network traffic. Additionally, the adaptability and generalization abilities of our model enable it to identify new and sophisticated threats, thus enhancing the robustness of network infrastructures against advanced cyber attacks. This study addresses a critical gap in IDS research by demonstrating the efficacy of Transformer models in this domain. Through rigorous empirical evaluations on diverse datasets, we showcase the superior performance and effectiveness of our proposed IDS, underscoring the transformative potential of this technology in network security. Our findings not only contribute to the growing body of literature on Transformer-based IDS but also highlight the need for ongoing research and innovation in this crucial field.

Keywords: Intrusion Detection System (IDS), Transformer Models, Machine Learning, Real-time Analysis

1. Introduction

In today's digital era, securing computer systems is a paramount concern due to the rise in sophisticated cyberattacks. Intrusion Detection Systems (IDS) are critical components of cybersecurity, essential for safeguarding networks and systems against both internal and external threats. Traditional intrusion detection methods, which rely on predefined signatures or specific rules, often fall short in identifying new or disguised threats [1-5].

Our research aims to address these limitations by developing a robust and adaptable IDS using advanced machine learning techniques, specifically Transformer models. Transformer models offer significant advantages over traditional methods, including scalability, parallel processing capabilities, and the ability to handle imbalanced data effectively. This study proposes leveraging the features extracted by these models to enhance intrusion detection in specific environments. This paper presents the theoretical context of the project, provides an overview of state-of-the-art techniques in intrusion detection, and details our methodology and implementation. We also discuss the results and implications, highlighting future research directions in this critical area of cybersecurity.

As cloud computing rapidly expands, offering numerous benefits in terms of accessibility and flexibility, the security and privacy of data in cloud environments are increasingly concerning, with various potential attacks such as IP spoofing, port scanning, and DDoS attacks. While firewalls serve as a first line of defense, they are often insufficient against sophisticated threats, necessitating additional measures to protect cloud infrastructures effectively [6]. The integration of artificial intelligence (AI) into vulnerability detection has become indispensable. Several approaches exist, including rule-based detection (conditional programming), lightweight machine learning models like Logistic Regression, and neural networks. Our project focuses on the innovative use of neural networks, specifically Transformer models, which

offer significant advantages over traditional recurrent neural networks (RNNs). Transformers excel in capturing fine details and performing parallel computations, making them particularly suited for intrusion detection [7].

Transformer models bring several benefits to intrusion detection, especially when dealing with imbalanced data. Unlike traditional RNNs, Transformers are adept at handling class imbalances in training data. Their self-attention mechanism allows them to focus on relevant parts of the input sequence, facilitating learning from minority class samples even in imbalanced datasets. Additionally, by capturing long-term dependencies across entire sequences, Transformers can better detect rare or sporadic events often found in minority classes. Their flexibility also allows for the easy incorporation of data augmentation techniques, such as synthetic oversampling, to enhance learning from these classes. However, employing Transformer models does not automatically resolve the issue of imbalanced data. Careful adaptation of data preprocessing, model architecture, and training strategies is crucial to fully exploit their potential in this context [8].

Our primary objective is to design and implement a robust AI model based on Transformer learning to enhance the security of computer systems against malicious attacks. This model aims to detect and respond to attacks in real-time, providing proactive protection against emerging threats. By leveraging the self-attention and parallel processing capabilities of Transformers, we seek to develop an adaptable solution capable of continuously monitoring network traffic and swiftly identifying suspicious behaviors. We will train our model on diverse and representative datasets, including the IDS-CIC2018 dataset, to ensure its ability to generalize and effectively detect a wide range of attacks, including new and sophisticated ones. We will rigorously evaluate our model's performance using standard metrics such as accuracy, recall, and F-measure, and compare it to other intrusion detection approaches to demonstrate its effectiveness and superiority. AI and machine learning are increasingly integrated into cybersecurity to detect and counter cyber threats. AI systems, by analyzing data patterns in network traffic, can identify warning signs and alert cybersecurity professionals. AI's capability to analyze vast datasets and spot potential threats, even those that may elude human observation, is particularly valuable for detecting subtle or concealed threats. Additionally, AI can automate repetitive and tedious cybersecurity tasks, such as applying patches and updates, thereby allowing cybersecurity professionals to focus on more complex challenges. By generating reports and alerts, AI provides valuable insights that guide cybersecurity decisions [9]. The potential benefits of AI in cybersecurity are significant. By enhancing the speed and accuracy of threat detection and response, AI helps mitigate the impact of cyberattacks. Furthermore, by automating certain tasks, AI optimizes the operational efficiency of cybersecurity teams [10].

2. Related Work

Attentive Transformer deep learning algorithms for intrusion detection in IoT systems using automatic explainable feature selection propose a novel model, TabNet-IDS, for Intrusion Detection Systems (IDS) in IoT environments [3]. This model uses attentive mechanisms to automatically select important features from tabular datasets common in machine learning tasks. By leveraging the TabNet algorithm within the PyTorch deep-learning framework, the model aims to achieve high accuracy in detecting security threats such as DoS and DDoS attacks. While TabNet-IDS provides interpretable results and addresses challenges related to model explainability and feature selection, it does have limitations. Despite achieving high accuracies of 97% on CIC-IDS2017, 95% on CSE-CICIDS2018, and 98% on CIC-DDoS2019 datasets, the approach may still face challenges with scalability and real-time processing in highly dynamic IoT environments.

In the realm of cloud computing security, protecting network-accessible resources against evolving threats is paramount. The efficacy of a Network Intrusion Detection System (NIDS) hinges on its adaptability and precision in detecting intrusions while minimizing false positives. The Transformer based network intrusion detection approach for cloud security introduces a novel NIDS algorithm tailored for the complexities of cloud environments, harnessing the power of the Transformer model. This algorithm integrates the principles of intrusion detection with the advanced attention mechanism of Transformers, enabling a nuanced examination of the relationships between input features and various intrusion types, thus bolstering detection accuracy. The model achieves an accuracy exceeding 93%, demonstrating robustness and viability for cloud security. However, its dependence on largescale computational resources and potential latency issues may limit its practical deployment in real time scenarios.

The Intrusion Detection Model Based on Improved Transformer presents an innovative model built on the Transformer architecture, addressing significant challenges in contemporary intrusion detection systems [4]. This model tackles issues such as prolonged training durations, inaccurate identification of overlapping classes, and substandard performance in multi-class classification scenarios. It employs a robust data processing strategy, incorporating a stacked auto-encoder for initial dimensionality reduction and a novel hybrid sampling methodology combining K-nearest neighbors (KNN) under-sampling with Borderline-SMOTE over-sampling. This effectively balances datasets and improves accuracy in detecting overlapping data classes. Additionally, an enhanced position encoding mechanism captures feature dependencies more effectively, boosting classification accuracy. The two-stage learning strategy initially performs binary prediction to identify illegal intrusions, followed by refined multi-class prediction. The model achieves an accuracy of 88.7% and an F1-score of 88.2% in binary classification, and an accuracy of 84.1% and an F1-score of 83.8% in multi-class classification. Despite these advancements, several limitations persist, such as response time challenges and low performance in detecting benign instances, with an accuracy of only 80%. The computational complexity also poses a barrier to deployment on resource-constrained devices, highlighting the need for further refinement and optimization.

While the aforementioned models have advanced the field of IDS (Table 1), significant limitations remain. The need for scalable, real-time processing and adaptability to new and evolving threats is critical. Our research addresses these gaps by developing a robust AI model based on Transformer learning, specifically designed for real-time attack detection. By leveraging the self-attention and parallel processing capabilities of Transformers, our approach provides a more adaptable and efficient solution for continuously monitoring network traffic and quickly identifying suspicious behaviors.

Model	Dataset(s)	Techniques Used	Strengths	Limitations	Performance Metrics
TabNet-IDS [3]	CIC-IDS2017, CSE-CICIDS2018, CICDDoS2019	Attentive mechanisms for feature selection, TabNet algorithm in PyTorch framework	High accuracy and explainability in IoT environments	Scalability and real-time processing challenges in dynamic IoT environments	Accuracy: 97% (CIC-IDS2017), 95% (CSE-CICIDS2018), 98% (CICDDoS2019)
Transformerbased NIDS for Cloud Security[1]	Cloud environments	Transformer model with advanced attention mechanisms	High adaptability, precise intrusion detection with minimal false positives	Large-scale resource dependency and potential latency issues in real-time deployment	Accuracy: >93%
Improved Transformer IDS [4]	N/A	Stacked autoencoder, hybrid sampling (KNN under-sampling, Borderline-SMOTE oversampling), enhanced position encoding	Effective feature dependency capture, good for overlapping class detection	Response time issues, suboptimal benign instance detection, high computational complexity	Accuracy: 88.7% (binary), 84.1% (multi-class); F1-score: 88.2% (binary), 83.8% (multi-class); Benign accuracy: 80%

Table 1: Related Work

Our proposed model significantly addresses the critical limitations observed in existing systems, such as the lack of scalability, inefficiencies in real-time processing, and challenges in handling class imbalances. Unlike prior approaches, including TabNet-IDS and Transformer-based models, which face issues with dynamic environments and resource dependency, our model leverages the Transformer architecture with self-attention and parallel processing to deliver robust real-time intrusion detection. By training on diverse and representative datasets, such as IDS-CIC2018, our model ensures greater generalization across a broader range of attacks, surpassing the adaptability and precision demonstrated by previous methods. Rigorous evaluations using standard performance metrics, including accuracy, recall, and F-measure, will further validate our model's superior capability to provide scalable, efficient, and accurate intrusion detection, making it a more effective solution for modern cybersecurity challenges. In summary, our contribution lies in addressing the critical limitations of current IDS models by providing a scalable, adaptable, and efficient solution capable of real-time attack detection. This advancement not only enhances the security of computer systems but also opens new avenues for research and development in the field of cybersecurity.

3. Methodology

In the methodology section, we delve into the systematic approach used to develop our machine learning model (Figure 1).

3.1. Data Collection

We utilize Azure cloud services for their extensive AI capabilities and smooth integration within a multi-cloud environment [11-13]. Specifically, we leverage Azure cloud services such as containers with storage accounts for efficient data storage, particularly accommodating our 4.4 GB dataset. Concurrently, we employ Azure AI for model training, capitalizing on its robust AI capabilities and effortless integration within our multi-cloud environment. In the data collection and analysis phase, we utilized the CIC-DS2018 dataset, comprising 10 distinct collections, to gather comprehensive insights into network traffic behavior [12]. As a crucial step in data integration, we concatenated all these collections into a unified dataset, ensuring a holistic representation of network activities.

3.2. Data Preprocessing and Analysis

In the data preprocessing phase, we focused on optimizing the dataset for subsequent analysis by removing noninformative columns. Notably, columns such as 'Flow ID' and 'Timestamp' were identified as not contributing significantly to our analysis objectives. we standardized the data types of columns to ensure uniformity and consistency across the dataset [14,15].

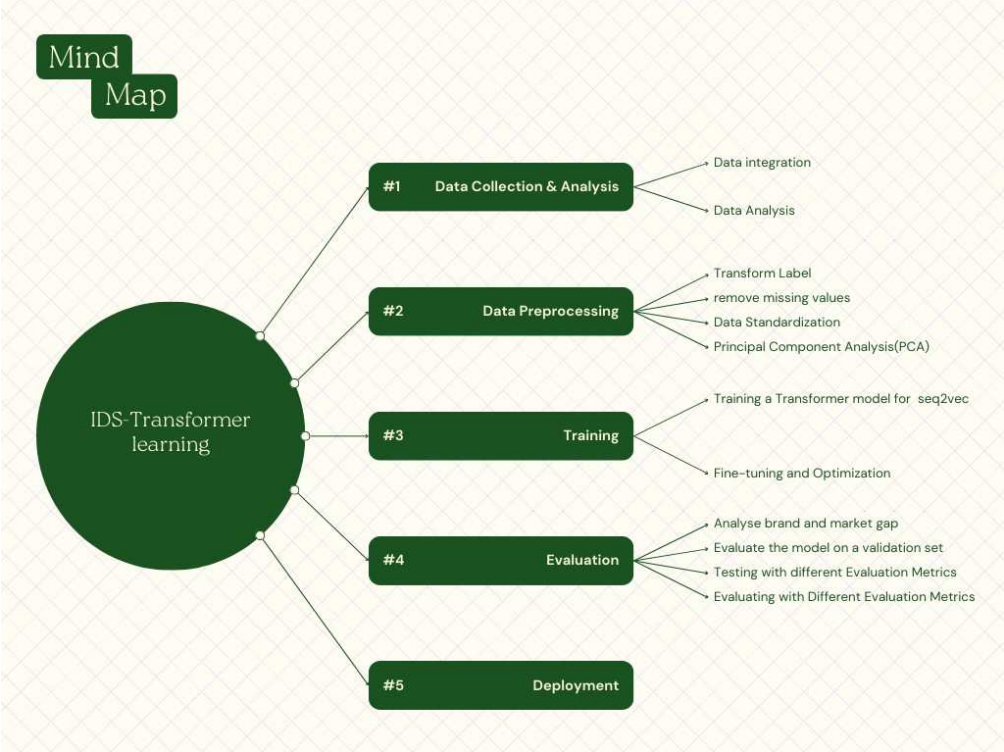

Figure 1: Workflow: From Data Preprocessing to Deployment

Label	**Count Percentage (%)**	
Benign	13484708	83.07
DDOS attack-HOIC	686012	4.23
DDoS attacks-LOICHTTP	576191	3.55
DoS attacks-Hulk	461912	2.85
Bot	286191	1.76
FTP-BruteForce	193360	1.19
SSH-Bruteforce	187589	1.16
Infilteration	161934	1.00
DoS attacks- SlowHTTPTest	139890	0.86
DoS attacks-GoldenEye	41508	0.26
DoS attacks-Slowloris	10990	0.07
DDOS attack-LOIC-UDP	1730	0.01
Brute Force -Web	611	0.00
Brute Force -XSS	230	0.00
SQL Injection	87	0.00

Table 2: Label Distribution

The distribution of labels in the dataset (Table 2) reveals a classic case of unbalanced data, where certain categories vastly outnumber others. We propose applying transformations to the label distribution to achieve a more balanced distribution (Table 3): Deletion of Certain Labels: Labels containing ”SQL Injection”, ”FTP-BruteForce”, ”Brute Force -XSS”, and ”Brute Force -Web” are removed from the dataset. Transformation of DDoS Attack Labels: Labels related to different types of DDoS attacks, including ”DDOS attack-HOIC”, ”DDOS attack-LOIC-UDP”, and ”DDoS attacks-LOIC-HTTP”, are transformed into a single category labeled as ”DDoS-attacks”.

Transformation of DoS Attack Labels: Labels associated with various forms of DoS attacks, such as ”DoS attacks-Hulk”, ”DoS attacks-SlowHTTPTest”, ”DoS attacks-GoldenEye”, and ”DoS attacks- Slowloris”, are transformed into a unified category labeled as ”Dos-attacks”.

Label	Count Percentage (%)	
Benign	13484708	84.08
DDoS attacks	1263933	7.88
Dos attacks	654300	4.08
Bot	286191	1.78
SSH-Bruteforce	187589	1.17
Infilteration	161934	1.01

Table 3: Updated Label Distribution

The data still isn’t perfectly balanced, meaning some types of attacks are more common than others. But compared to before, it’s much better now. We’ve grouped similar attacks together, like putting different kinds of DDoS attacks in one category. This makes the data more fair overall (See Figure 2).

Next we have preprocessed our dataset by removing samples containing null and infinite values, followed by the elimination of duplicate samples. These preprocessing steps were feasible due to the size of our dataset, comprising 16 million samples. This extensive dataset affords us the flexibility to retain only the most relevant and representative samples, ensuring the quality and integrity of our data analysis.

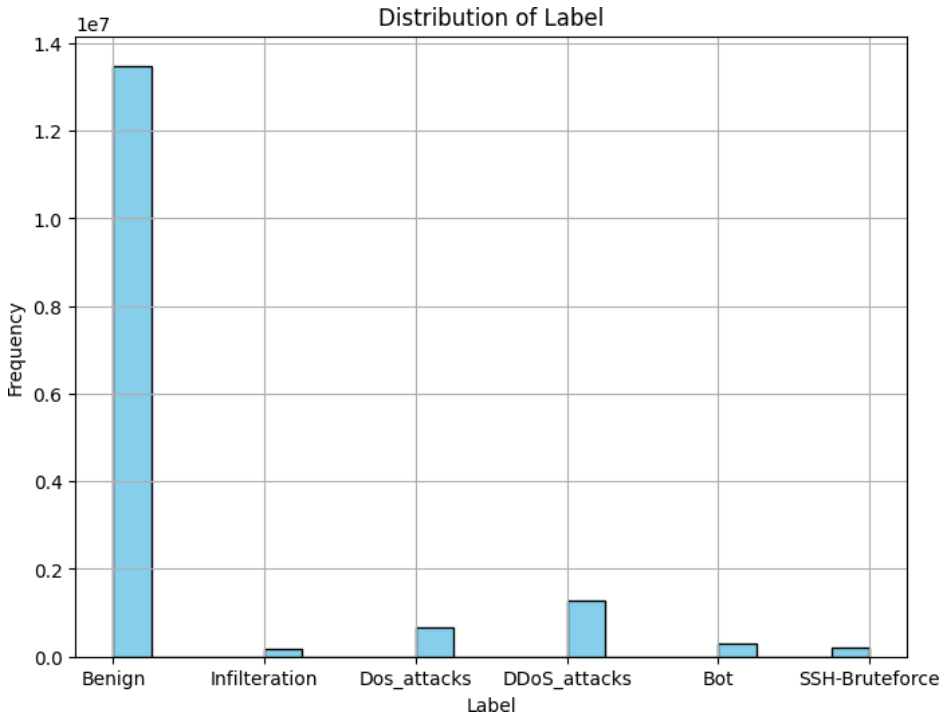

Figure 2: Label Distribution

The Mann-Whitney U test is a non-parametric method used to evaluate whether two independent samples originate from the same distribution. By comparing the ranks of observations between the two groups, it determines if there is a tendency for one group to have higher values than the other. This test is particularly useful when the assumptions required for parametric tests, such as the t-test, are not satisfied, including situations where the data do not follow a normal distribution. In this section, we leverage statistical functions from the SciPy library. The following code demon- strates our approach (See Figure 3):

```
normal_traffic = data[data['Label'] == 0]
intrusion_traffic = data[data['Label'] == 1]

# Features to compare
features = ['Dst Port', 'Protocol', 'Flow Duration', 'Tot Fwd Pkts', 'Tot Bwd Pkts']

for feature in features:
    normal_values = normal_traffic[feature]
    intrusion_values = intrusion_traffic[feature]

    # Perform Mann-Whitney U test
    statistic, p_value = stats.mannwhitneyu(normal_values, intrusion_values)

    # Set significance level
    alpha = 0.05

    # Interpret results
    if p_value < alpha:
        print(f"Reject the null hypothesis: There is a significant difference in {feature} between norma
    else:
        print(f"Fail to reject the null hypothesis: There is no significant difference in {feature} betw
```

Figure 3: Statistical Analysis of Network Traffic Features using Mann-Whitney U Test

- Comparison of Network Traffic Patterns Using Mann-Whitney U Test:

Here, we present the findings of our analysis concerning the distinctions in network traffic patterns between normal traffic and traffic containing intrusions. We subjected five key features—Destination Port (Dst Port), Protocol, Flow Duration, Total Forward Packets (Tot Fwd Pkts), and Total Backward Packets (Tot Bwd Pkts)—to statistical tests. We employed the Mann-Whitney U test to compare the distributions of the selected features between normal traffic and traffic containing intrusions. The statistical analysis revealed significant differences in the distributions of all examined features ($p < 0.05$ for all features), indicating noticeable differentiation in network traffic patterns between the two categories.

In this step, our data is structured as a table with dimensions (11915218, 79). We partition the data into 90% for training and 10% for testing. The following Figure 4 displays the first 4 samples.

	Dst Port	Protocol	Flow Duration	Tot Fwd Pkts	Tot Bwd Pkts	TotLen Fwd Pkts	TotLen Bwd Pkts	Fwd Pkt Len Max	Fwd Pkt Len Min	Fwd Pkt Len Mean	...	Fwd Seg Size Min	Active Mean	Active Std	Active Max	Active Min	Idle Mean	Idle Std	Idle Max	Idle Min	Label
0	443	6	94658	6	7	708	3718	387	0	118.0	...	20	0.0	0.0	0	0	0.0	0.0	0	0	0
1	443	6	206	2	0	0	0	0	0	0.0	...	20	0.0	0.0	0	0	0.0	0.0	0	0	0
2	445	6	165505	3	1	0	0	0	0	0.0	...	20	0.0	0.0	0	0	0.0	0.0	0	0	0
3	443	6	102429	6	7	708	3718	387	0	118.0	...	20	0.0	0.0	0	0	0.0	0.0	0	0	0
4	443	6	167	2	0	0	0	0	0	0.0	...	20	0.0	0.0	0	0	0.0	0.0	0	0	0

Figure 4: The First 4 Samples of the Data

- **Following that, We Perform Normalization using Min Max Scaler:**

MinMaxScaler is a technique used for feature scaling in machine learning, where each feature is scaled to a specified range, typically [0, 1].

$$X_{\text{scaled}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}}$$

- **Then we use PCA to Reduce the Dimensionality:**

Principal Component Analysis (PCA) is a dimensionality reduction technique widely used in machine learning and statistics. Its purpose is to transform high-dimensional data into a lower-dimensional space while retaining the most significant information. PCA accomplishes this by identifying new orthogonal axes, known as principal components, along which the data shows the greatest variance. These principal components represent the directions of maximum variability in the original data and are ranked according to the variance they explain. By selecting only a subset of principal components that account for the majority of the variance, such as 90%, PCA effectively reduces the dataset's dimensionality with minimal information loss (See Figure 5).

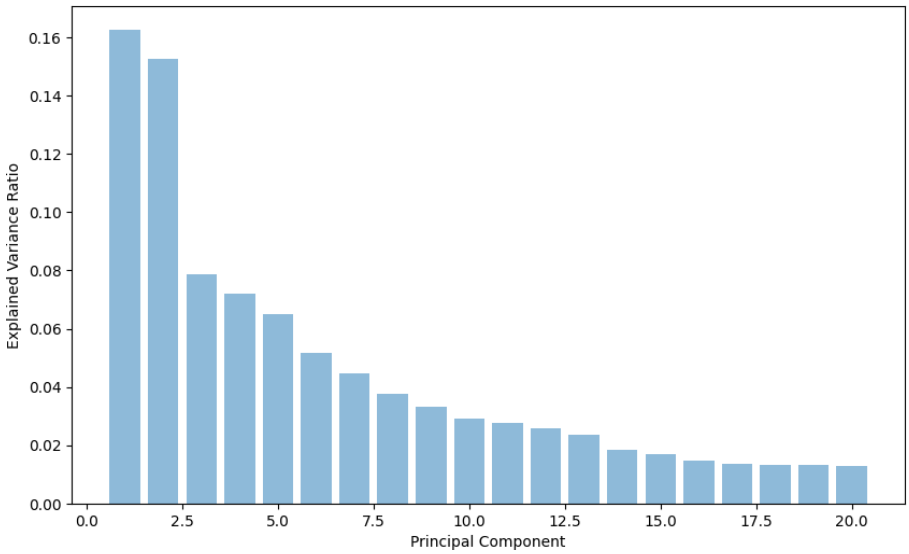

Figure 5: Explained Variance for the First 20 Principal Component

We plotted all the axes information, and our decision is to retain 90% of the data variance. Therefore, the first 20 axes preserve this 90%. Our new data now has a shape of (11915218, 20), indicating that we've retained 90% of the variance in our dataset while reducing its size by 75%. This reduction significantly speeds up the training process and provides an alternative representation of the data, excluding less important features. This can lead to improved model performance. The following Figure 6 displays the first 4 samples of our new data.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	-2.516888	0.591043	0.849619	1.475652	-3.654469	0.408912	0.271580	0.490459	-0.156305	-0.015096	-0.016338	-0.164961	-0.137077	0.145394	0.042627	0.075262	-0.055648	0.105723	-0.045109	-0.025036
1	-2.856154	0.579327	-0.275561	-0.368706	1.568547	-0.064568	0.102646	-0.826018	0.253524	0.135502	0.385627	0.888160	0.374913	-1.161812	-0.306627	-0.050148	-0.422375	0.456920	0.266090	-0.060393
2	3.358644	-0.587420	-0.337132	-0.566618	-0.042068	1.210370	-0.175062	0.483922	-0.503001	-0.014278	0.212912	0.145469	0.396910	-0.967706	-1.196252	-1.244673	0.838602	-0.569828	-0.128403	-0.216304
3	3.902184	-0.696059	-0.486854	-0.797841	0.120791	1.268800	-0.326343	-0.246766	-0.176452	0.076809	-1.501341	-0.071960	0.301129	0.565580	-0.095878	-0.364249	0.466451	-0.265305	-0.488369	-0.627658
4	3.218063	-0.581511	-0.402965	-0.585097	0.195374	1.014914	-0.281511	-0.897772	0.328285	0.106478	-0.933151	-0.925962	0.336061	1.575415	0.594097	1.138106	-0.820469	0.615004	0.538333	0.780189

Figure 6: Data has been Subjected to both Normalization and PCA

3.3. Model Architecture

The Transformer architecture has redefined the field of natural language processing (NLP) by introducing an innovative approach to sequence modeling [16]. Diverging from traditional recurrent neural networks (RNNs) and convolutional neural networks (CNNs), the Transformer relies entirely on selfattention mechanisms to simultaneously capture dependencies between input and output tokens. This design enables more efficient training and better handling of long-range dependencies. Originally introduced in the landmark paper 'Attention is All You Need', the Transformer has served as the foundation for cutting-edge NLP models such as BERT, GPT, and T5, excelling in tasks ranging from machine translation and text generation to language understanding [17-20].

Beyond NLP, the Transformer architecture (Figure 7) has demonstrated exceptional versatility, proving its efficacy in tasks like multi-class classification [7]. The goal in multi-class classification is to categorize input data into one of several possible classes. Thanks to its ability to uncover intricate patterns and relationships within sequences, the Transformer is particularly well-suited for this challenge. By leveraging the self-attention mechanism, the Transformer efficiently learns representations of input sequences, whether these sequences are sentences, images, or time-series data [6]. This capability enables it to excel in multi-class classification tasks by extracting rich features from input data and making precise predictions. Recently, especially since 2023, Transformer-based models have garnered significant attention in the field of intrusion detection, as they excel at identifying complex

patterns in sequential data, such as network traffi [20]. This unique strength has the potential to significantly enhance the effectiveness of intrusion detection systems.

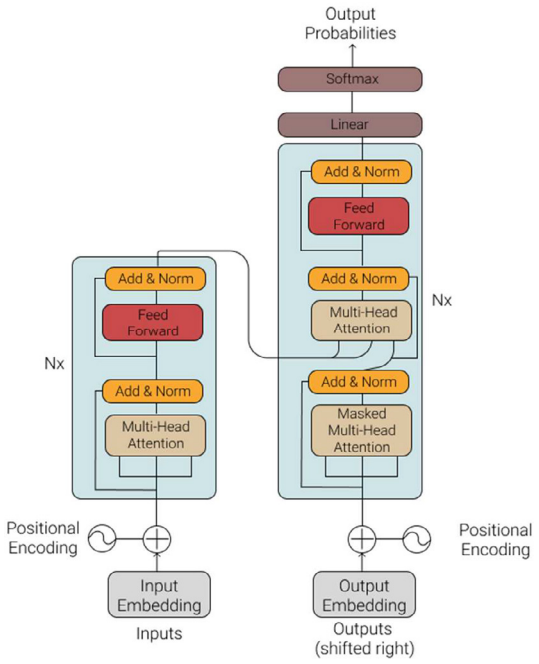

Figure 7: The Transformer Model Architecture [2]

3.3.1. Multi-Head Attention

Multi-head Attention (Figure 8) is a mechanism that processes multiple attention mechanisms in parallel. Each independent attention output is concatenated and linearly transformed to the desired dimension, enabling the model to simultaneously focus on different parts of the sequence and capture both long-term and short-term dependencies.

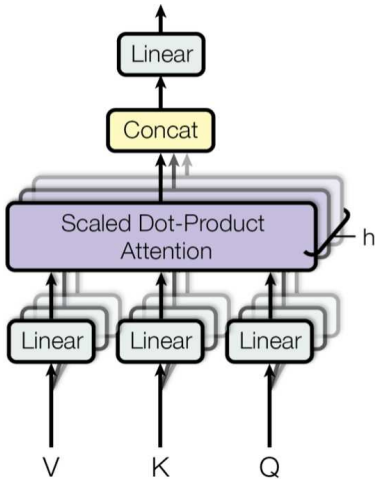

Figure 8: Multi-Head Attention

The multi-head attention formula is expressed as: MultiHead(Q, K, V) = [head1, ..., headh]W0, where headi = Attention(QWQ, KWK, VWV). Here, W represents learnable parameter matrices.

3.3.2. Position-Wise Feed-Forward Networks

Position-wise Feed-Forward Networks process each input position independently using two fully connected layers with a ReLU activation function between them. This additional processing step refines the features extracted by the attention mechanism, enhancing the model's representational capabilities.

3.3.3. Positional Encoding

The Positional Encoding mechanism supplements the Transformer model by embedding positional information into token representations. Using sinusoidal functions, it enables the model to incorporate the sequence order, which is essential for tasks requiring an understanding of token positions.

3.3.4. Encoder Blocks

Each Encoder Block (Figure 9) consists of a multi-head self-attention mechanism, a position-wise feed-forward network, residual connections, and layer normalization. These components collectively allow the encoder to process and represent intricate patterns within the input data.

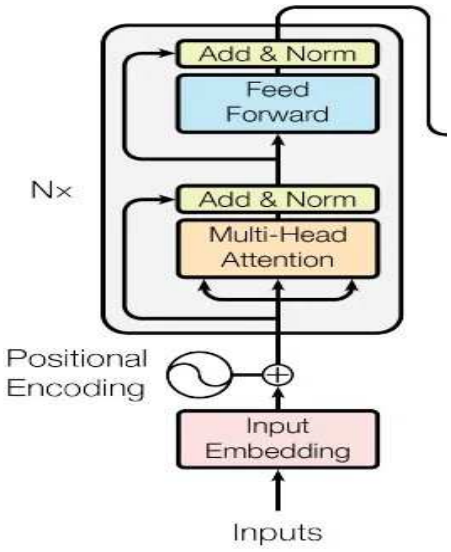

Figure 9: The Encoder Part of the Transformer Networks

3.3.5. Decoder Blocks

The Decoder Blocks (Figure 10) integrate self-attention, cross-attention with encoder outputs, and position-wise feed-forward networks. These blocks enable the model to generate outputs while attending to both the input and previously generated tokens.

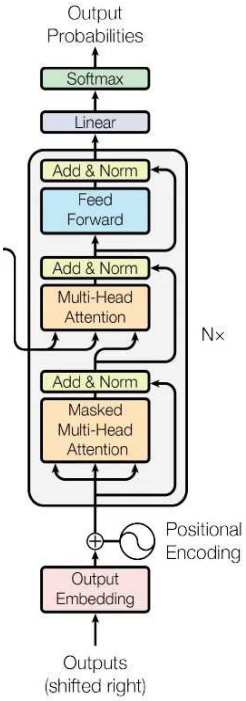

Figure 10: The Decoder Part of the Transformer Networks

3.3.6. Combining the Encoder and Decoder Layers

The complete Transformer network combines the encoder and decoder layers alongside components such as embeddings, positional encodings, and masking mechanisms. This cohesive design makes the Transformer suitable for a variety of sequence-to-sequence tasks, including machine translation and text summarization.

3.4. Evaluation Metrics

Evaluation metrics are quantitative measures used to assess the performance of a machine learning model or algorithm. These metrics provide insights into how well the model is performing on a given task, such as classification, regression, or clustering. Evaluation metrics help researchers, practitioners, and stakeholders understand the strengths and weaknesses of the model and make informed decisions about its deployment or improvement.

Accuracy is a fundamental evaluation metric used in classification tasks to measure the overall cor- rectness of predictions made by a model. It calculates the proportion of correctly classified instances (both true positives and true negatives) out of the total instances in the dataset.

Mathematically, accuracy is defined as: $\frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$

Where:

- Number of Correct Predictions: The sum of correct predictions across all classes.
- Total Number of Predictions: The total number of instances in the dataset.

While accuracy is a widely used metric, it may not always be suitable for imbalanced datasets, where one class dominates the others. In such cases, accuracy can be misleading, as the model may achieve high accuracy by simply predicting the majority class most of the time.

In these situations, it's important to consider other evaluation metrics, such as precision, recall, F1 score, or class-specific metrics, to get a more comprehensive understanding of the model's performance.

For these reasons, we have chosen to utilize three evaluation metrics in our analysis. Firstly, we consider accuracy as a primary metric to assess the overall correctness of our model's predictions across all classes.

Secondly, we incorporate the F1-score, which provides a balanced measure of precision and recall and is particularly useful for assessing model performance when there is class imbalance.

Lastly, we employ class-wise accuracy, also known as per-class accuracy, to evaluate the performance of our model on each individual class, providing insights into its effectiveness in correctly predicting instances belonging to specific classes.

In multiclass classification, the F1-score for each class i can be calculated as follows:

$$\text{Precision}_i = \frac{\text{True Positives}_i}{\text{True Positives}_i + \text{False Positives}_i}$$

$$\text{Recall}_i = \frac{\text{True Positives}_i}{\text{True Positives}_i + \text{False Negatives}_i}$$

$$\text{F1-score}_i = 2 \times \frac{\text{Precision}_i \times \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i}$$

Where:

- True Positives (TP_i) are the number of instances correctly predicted as class i.
- False Positives (FP_i) are the number of instances incorrectly predicted as class i when they actually belong to another class.
- False Negatives (FNi) are the number of instances incorrectly predicted as not class i when they actually belong to class i.

To obtain the overall performance of the model, the macro F1-score is calculated as:

$$\text{Macro F1-score} = \frac{1}{N} \sum_{i=1} \text{F1-score}_i$$

Where N is the total number of classes.

4. Results & Discussion

In our effort to make the model work better, we carefully picked certain settings after trying different options. We had three main goals: make the model perform well, keep it from getting too big, and make sure it doesn't take too long to train.

We looked at how well the model did its job, how big it was, and how long it took to train. We wanted it to be both good at its job and not take forever to train, all while not being too big.

To make sure our decisions were based on solid evidence, we tested the model with a smaller version of the data, making sure the tests were fair.

Parameter	Value
src vocab size	5000
num classes	6

d model	512
num heads	8
num layers	6
d ff	2048
max seq length	20
epochs	5
dropout	0.1
Learning rate	0.0001

Table 4: Parameter Configuration of the Model

Table 4 summarizes the key hyperparameters used for training the Transformer model. It includes the source vocabulary size (5000), number of target classes (6), model dimensionality (512), attention heads (8), encoder/decoder layers (6), feed-forward network dimension (2048), maximum sequence length (20), training epochs (5), dropout rate (0.1), and learning rate (0.0001). These parameters define the model's architecture and optimization settings, ensuring effective learning and performance.

We executed the model, and here are the results of the model on the testing set:

- Accuracy: 0.9873
- F1-score: 0.9873

Now, we delve into the depths, and here are the results for all categories.

This confusion matrix (Figure 11) illustrates the performance of a classification model across six classes: Benign, Bot, DDoS_ attacks, Dos_attacks, Infiltration, and SSH-Bruteforce. The diagonal elements represent correctly predicted instances, while off-diagonal elements indicate misclassifications. For example, the model classified 2,111,415 instances of the "Benign" class correctly, with minimal errors for other classes. The performance for other classes, such as "DDoS_attacks" and "SSH-Bruteforce," also demonstrates high accuracy. However, there are noticeable misclassifications, particularly in the "Infiltration" class, where 27,664 instances were incorrectly classified as "Benign".

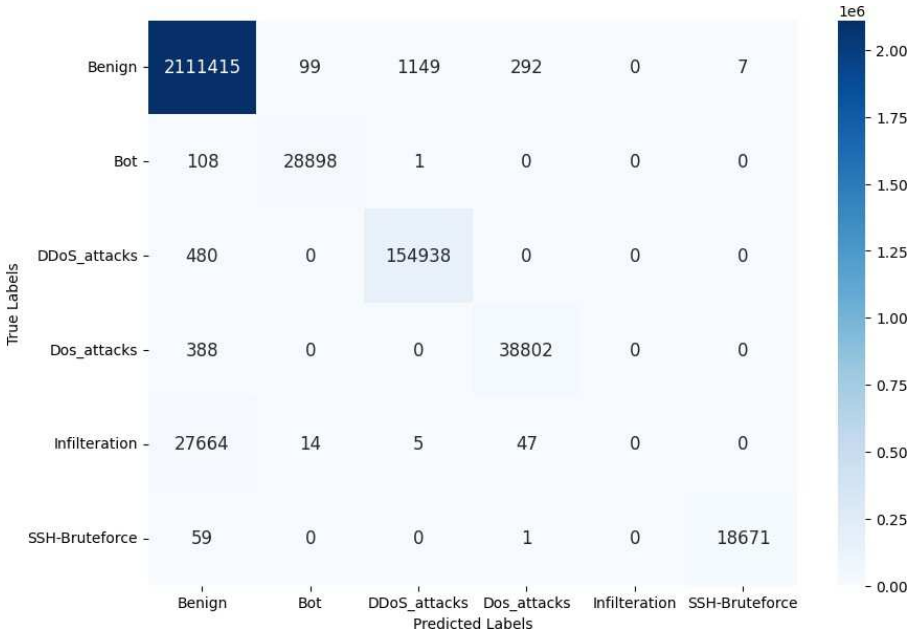

Figure 11: Confusion Matrix

Class	Accuracy	Precision	Recall	F1-score
Benign	0.9993	0.9866	0.9993	0.9929
Bot	0.9992	0.9961	0.9962	0.9962

DDoS attacks	0.9999	0.9926	0.9969	0.9948
Dos attacks	0.9991	0.9913	0.9901	0.9907
Infiltration	0.8952	0.7752	0.8014	0.8121
SSH-Bruteforce	0.9998	0.9996	0.9968	0.9982

Table 5: Evaluation Metrics

Upon reviewing the general metrics of accuracy and F1score, both recorded at 0.9873 (Table 5), we confirm the model's reliability. However, upon closer examination, we find exceptional performance in detecting in- trusions such as 'Bot,' 'DDoS attacks,' 'Dos attacks,' and 'SSHBruteforce,' which significantly contribute to the high accuracy. Nevertheless, a notable drawback emerges regarding the 'Infilteration' category, where our model exhibits deficiencies in detection. This limitation represents a significant area for im- provement, warranting further refinement to enhance the model's overall efficacy and coverage across all intrusion types.

4.1. Overfitting: When a model learns the training data too closely, capturing noise and irrelevant pat- terns, leading to poor performance on new data.

4.2. Underfitting: When a model is too simple to capture the underlying structure of the data, resulting in poor performance on both training and test datasets. Our model demonstrates neither underfitting nor overfitting, as evidenced by its consistent high accuracy of over 98% on both the training and test sets.

5. Discussion

We made comparisons with the related works previously cited (Table 6). All comparisons are made with CIC-DS2018.

In this section, we use two models to compare with our model:

- **NIDS** [4]
- **TabNet-IDS** [3]

Score %					
Class	**Model**	**Accuracy**	**Precision**	**Recall**	**F1-Score**
	Our Model	99.93	98.66	99.93	99.29
Benign	NIDS	93.572	94.6515	98.3251	96.4533
	TabNet-IDS	79	-	-	-
	Our Model	99.92	99.61	99.62	99.62
Bot	NIDS	99.9891	99.7199	99.3589	99.5391
	TabNet-IDS	100	-	-	-
	Our Model	99.99	99.26	99.69	99.48
DDoS attacks	NIDS	94.7668	64.4014	39.4349	48.9167
	TabNet-IDS	98	-	-	-
	Our Model	99.91	99.13	99.01	99.07
Dos attacks	NIDS	99.8151	99.2901	89.1539	93.9494
	TabNet-IDS	96	-	-	-
	Our Model	89,52	77,254	80,142	81,215
Infiltration	NIDS	98.7943	44.2619	11.776	13.228
	TabNet-IDS	92	-	-	-
	Our Model	99.98	99.96	99.68	99.82
SSH-Bruteforce	NIDS	99.9612	17.2378	36.9811	12.1062
	TabNet-IDS	95	-	-	-

Table 6: Intrusion Detection Comparison the '-' Symbol Indicates the Missing Score

In the realm of intrusion detection, our Transformer-based model exhibits exceptional performance across a wide array of attack types. This section discusses the comparative strengths and limitations of our model in relation to existing approaches, such as NIDS and TabNet-IDS, and explores the broader implications of our findings for cybersecurity. Our model demonstrates significantly higher performance metrics in detecting various types of intrusions, particularly benign traffic, DDoS, brute-force, and DoS attacks. The key highlights include:

5.1. Benign Traffic Detection: Our model achieves an accuracy of 99.93%, surpassing NIDS and TabNet-IDS. This high accuracy ensures that normal network behavior is correctly identified, minimizing false positives and reducing unnecessary alerts.

5.2. Bot Intrusions: While all models show strong performance in detecting bot intrusions, our model excels with an accuracy of 99.92%. This consistency underscores our model's reliability in maintaining high detection rates for common attack vectors.

5.3. DDoS Attacks: Our model's accuracy of 99.99% in detecting DDoS attacks is markedly superior to that of NIDS, which struggles with a significantly lower precision and recall. This suggests that our Transformer-based approach is particularly adept at identifying large-scale, distributed attacks that can overwhelm traditional IDS.

5.4. DoS Attacks: With a 99.91% accuracy, our model again outperforms the alternatives. The precision and recall scores indicate that our model not only identifies these attacks accurately but also does so consistently across varied datasets.

5.5. SSH-Brute Force Attacks: Achieving a 99.98% accuracy, our model significantly outperforms NIDS, which exhibits much lower detection rates. This indicates our model's effectiveness in identifying and mitigating brute-force attempts which are critical for maintaining secure SSH services.

Despite its overall superior performance, our model shows lower efficacy in detecting infiltration attacks, achieving an accuracy of 89,52%. In contrast, other models like NIDS, while still not perfect, manage better detection rates for this specific class. This gap suggests several potential areas for improvement:

5.6. Infiltration Detection: The poor performance in this category highlights a critical area where our model needs refinement. Future work could focus on enhancing the model's sensitivity to subtle infiltration attempts, possibly by integrating more nuanced feature extraction techniques or additional training data specific to infiltration scenarios. Feature Selection and Model Complexity: While our model leverages the advanced capabilities of Transformer architectures, the complexity and computational demands may hinder its performance in real-time and resource-constrained environments. Optimizing the model to balance accuracy and computational efficiency could enhance its practical deployment across diverse network settings.

6. Conclusion

The high performance of our Transformer-based model in most categories underscores its potential to significantly enhance network security by providing reliable, real-time intrusion detection. However, the identified limitations in detecting infiltration attacks point to the need for ongoing research and refinement:

Scalability and Real-Time Processing: Our model's ability to process large volumes of data in parallel makes it well-suited for real-time applications. Future work could explore scalable deployment strategies to ensure consistent performance across different network environments.

Adaptability to New Threats: The adaptability of Transformer models can be further leveraged by incorporating continuous learning mechanisms, allowing the IDS to evolve in response to emerging threats.

Broader Application: Extending the application of our model to other domains such as industrial control systems, IoT environments, and cloud services can help generalize its utility and robustness.

In conclusion, our research presents a significant advancement in IDS technology, demonstrating superior detection capabilities for most attack types. Addressing the identified gaps, particularly in infiltration detection, will further solidify its position as a leading solution in the ever-evolving landscape of cybersecurity.

Ethical Statements

The authors declare that this manuscript is original, has not been published before and is not currently being considered for publication elsewhere. II. The authors declare that they have no funding. III. The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper. IV. The authors have read and agree to the Terms and Conditions of the journal.

Data Availability

The datasets generated during and/or analyzed during the current study are available from the authors on request.

References

1. Liu, Y., & Wu, L. (2023). Intrusion detection model based on improved transformer. *Applied Sciences, 13*(10), 6251.
2. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.
3. Zegarra Rodriguez, D., Daniel Okey, O., Maidin, S. S., Umoren Udo, E., & Kleinschmidt, J. H. (2023). Attentive transformer deep learning algorithm for intrusion detection on IoT systems using automatic Xplainable feature selection. *Plos one, 18*(10), e0286652.
4. Long, Z., Yan, H., Shen, G., Zhang, X., He, H., & Cheng, L. (2024). A Transformer-based network intrusion detection approach for cloud security. *Journal of Cloud Computing, 13*(1), 5.
5. Abdallah, E. E., & Otoom, A. F. (2022). Intrusion detection systems using supervised machine learning techniques: a survey. *Procedia Computer Science, 201*, 205-212.
6. Tsai, C. F., Hsu, Y. F., Lin, C. Y., & Lin, W. Y. (2009). Intrusion detection by machine learning: A review. *expert systems with applications, 36*(10), 11994-12000.
7. Li, Y., & Li, Y. (2023). A novel intrusion detection system based on machine learning algorithms. *Journal of Network and Computer Applications*.
8. Sharma, A., & Singh, S. (2023). Deep learning-based intrusion

detection system for IoT networks. *IEEE Internet of Things Journal.*
9. Kim, J., & Park, S. (2023). An ensemble approach for intrusion detection using machine learning. *Expert Systems with Applications.*
10. Chen, Z., & Zhang, T. 2023. Intrusion detection in cloud computing using machine learning techniques. *Future Generation Computer Systems.*
11. Wang, L., & Liu, Y. A survey of machine learning techniques for intrusion detection in wireless sensor networks. Ad Hoc Networks.
12. Gupta, R., & Verma, (2023). A Intrusion detection in smart grids using machine learning algorithms. *Sustainable Cities and Society.*
13. Patel, N., & Shah, R. (2023). Machine learning-based intrusion detection system for industrial control systems. *Computers & Security.*
14. Jemili, F., & Korbaa, O. (2024). Active intrusion detection and prediction based on temporal big data analytics. *International Journal of Knowledge-based and Intelligent Engineering Systems, 28*(2), 389-418.
15. Aljabri, A., Jemili, F., & Korbaa, O. (2024). Intrusion detection in cyber-physical system using rsa blockchain technology. *Multimedia Tools and Applications, 83*(16), 48119-48140.
16. Jemili, F., Meddeb, R., & Korbaa, O. (2024). Intrusion detection based on ensemble learning for big data classification. *Cluster Computing, 27*(3), 3771-3798.
17. Abid, A., Jemili, F., & Korbaa, O. (2023). Distributed deep learning approach for intrusion detection system in industrial control systems based on big data technique and transfer learning. *Journal of Information and Telecommunication, 7*(4), 513-541.
18. Albertshiu (2024). "Build a Transformer model based on Pytorch with bare hands". KKCOMPANY innovation day 2024.
19. Aljabri, A., Jemili, F., & Korbaa, O. (2024). Convolutional neural network for intrusion detection using blockchain technology. *International Journal of Computers and Applications, 46*(2), 67-77.
20. Belhor, M., & Jemili, F. (2016, November). Intrusion detection based on genetic fuzzy classification system. In *2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA)* (pp. 1-8). IEEE.

Copyright: *©2026 Farah Jemili, et al. This is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.*