AegisFormer-IDS: Fast FT-Transformer for Real-Time Intrusion Detection on NSL-KDD

Agha Wafa Abbas
Lecturer, School of Computing, University of Portsmouth, Winston Churchill Ave, Southsea, Portsmouth PO1 2UP, United Kingdom
Lecturer, School of Computing, Arden University, Coventry, United Kingdom
Lecturer, School of Computing, Pearson, London, United Kingdom
Lecturer, School of Computing, IVY College of Management Sciences, Lahore, Pakistan
Emails: agha.wafa@port.ac.uk , awabbas@arden.ac.uk, wafa.abbas.lhr@rootsivy.edu.pk

Abstract

In the event of insecurity of network infrastructures due to cyber attacks real time intrusion detectors systems (IDS) play a critical role in ensuring protection to these networks infrastructures. This paper is the description of AegisFormer-IDS, a lightweight, feature-efficient Feature-Tokenized Transformer (FT-Transformer) network that is best used to perform a fast binary classification of network traffic as either normal or anomalous. Our work (based on the NSL-KDD data) embraces best practices such as the label encoding of suit categories, the standardization of features with numeric values, and binary re-labeling (normal vs. attack) to facilitate movement to a simplified Transformer encoder architecture. The architecture is very compact, with projection of features, multi-head self-attention layers and classification head that is fully connected, allowing quick inference without compromising performance. AegisFormer-IDS, trained on 10 epochs with the PyTorch framework, Adam optimization, the cross-entropy loss, and a learning rate scheduler, achieves both over 99 and above 0.99 validation accuracy, and F1-score on a 80/20 of train/validation split. Empirical evidence shows that it is better at speed, accuracy, and resource consumption than standard deep learning models, and can be deployed in real-time where resource constraints are present. The gap between transformer-based progress and scalable high-fidelity network security utilization is bridged in this work, leading to scalable applications of network security.

Keywords

Intrusion Detection System (IDS), FT-Transformer, NSL-KDD Dataset, Real-Time Network Security, Deep Learning, Binary Classification, Transformer Encoder, PyTorch Implementation, Cyber Threat Detection, Feature Preprocessing

I. Introduction

The massive scale of cyber threats has also augmented research (such as advanced malware, distributed denial-of-service [DDoS], and advanced persistent threat]) thereby establishing network security high up the priority list. IDS contributes tremendously to the overcoming of network traffic, the determination of rogue traffic and, in every way possible, responding in good time to prevent the failure of important infrastructure and sensitive information. Conventional methods of the IDS, including signature-based and rule-driven systems, identify attacks based on known attack pattern, and hence require novel attacks targeting zero-day exploits that can bypass all of them as well [1]. These systems are further challenged by the growing amount and non-linearity of network traffic, which requires increasingly sophisticated algorithms able to handle high-dimensional data at reasonable latency and performance levels.

This allows the modeling of more intricate patterns in network traffic, a major innovation in IDS methods introduced by machine learning (ML). Initial ML-based IDS methods including decision trees, support factor machines (SVMs), and k-nearest neighbor (k-NN) showed better rates of detection compared to those of rule-based methods. Using the KDD Cup 1999 dataset as an example, it was proven that the ML algorithms could be successfully used to recognize patterns of attacks without the need to update the rules manually [2]. However, these methods are more likely to require extensive feature engineering and are unable to scale to high dimensional data as well. In recent years, the development of deep learning (DL) has overcomes some of these problems, automating feature extraction and able to extract complex relationships between data. Deep learning (CNNs and RNNs) models are partly successful in the task of IDS due to their presence of behavioural models that can capture spatial and temporal templates of network-traffic [3].

As efficient as they are, conventional deep learning models have severe constraints in real-time use of an IDS. However, CNNs are efficient at sorting out spatial features, put another way, they are computationally expensive, which leads to a high inference time that cannot be used in low-latency applications [4]. In the same way, RNNs, in particular, long short-term memory (LSTM) networks can effectively represent the dependencies between the time sequence, though they need to be processed sequentially, which will incur considerable computational costs [5]. This restricts the application of CNNs and RNNs to resource-limited devices (e.g., edge routers or IoT gateways) where fast detection is necessary. The lightweight and efficient model assessed on the basis of high accuracy and low latency is becoming the burning issue of the IDS research.

First introduced to natural language processing (NLP), transformer-based architectures have dramatically changed the sequence modeling within the field by utilizing self-attention mechanisms to learn long-range dependencies, which are otherwise inefficient. In contrast to RNNs, transformers can work in parallel, which dramatically shortens the inference period. A variant of the Transformer, customized to handle tabular data, is called the Feature-Tokenized Transformer (FT-Transformer) that tokenizes not only numerical features but also categorical ones to form a single representation that can be processed in the transformer [7]. The technique has demonstrated state-of-the-art performance in table-based tasks, and is a future candidate towards IDS applications. Calibration, however, is discouraged when applying transformers to intrusion detection since providing such transformer models may be computationally expensive because of their large parameterization counts as well as target architectures.

We present in this paper AegisFormer-IDS, a simple but effective FT-Transformer-based model designed to be used in real-time intrusion detection with the NSL-KDD dataset. Closely related, NSL-KDD, a refined extension of the KDD Cup 1999 dataset, is a popular benchmark in the IDS literature as it offers a rich feature detail as well as a variety of attack scenarios [8]. It contains 125,973 training samples and 22544 tests samples, both of which contain 41 features (e.g., duration, protocol_type, src_bytes) and whether the traffic is normal or not. AegisFormer-IDS resolves the real-time IDS challenges by combining powerful preprocessing, a simple transformer architecture, and an efficient training pipeline built in PyTorch. Preprocessing pipeline involves categorical feature label encoding (protocol-type, service, flag), numerical features standardization and binary labeling (normal vs attack) so as to simplify the classification process but not lose important information.

The AegisFormer-IDS architecture is designed to optimize both performance and efficiency. It features a feature projection layer that maps the 41 input features to a 64-dimensional embedding space, followed by a transformer encoder with two layers and four attention heads. A fully connected classification head outputs logits for binary classification, enabling rapid inference with minimal computational overhead. The model is trained using the Adam optimizer, cross-entropy loss, and a step-wise learning rate scheduler, achieving validation accuracies above 99% and weighted F1-scores exceeding 0.99 across 10 epochs on an 80/20 train-validation split. These results highlight the model's ability to accurately distinguish between normal and malicious traffic while maintaining low latency, making it suitable for real-time deployment in resource-constrained environments.

The AegisFormer-IDS has been designed to maximize performance and efficiency. It has a specially designed feature project map that projects the 41 input features onto a 64 dimensional embedding space after which there is a two layer four attention-head transformer encoder. A fused classification head produces a binary classification logits, which can be inferred quickly with negligible computation cost. The model is then trained with the Adam optimizer, the cross-entropy loss, a progressive learning rate, with rates increasing to more than 99% validation accuracy and larger than 0.99 weighted F1-scores of the 10 regular train steps, using an 80/20 train/validation split. These results firmly show the selectivity of the model in terms of its ability to distinguish normal and malicious traffic, and the latency, the model can perform the role of a deployed in real time in a resource-constrained system.

This work has three multiple contributions. We present, first, AegisFormer-IDS, a new FT-Transformer-based model that can be trained in real-time to be as accurate and as low-latency as possible, meaning designed specifically to detect intruders. Second, we propose a full preprocessing pipeline specific to the NSL-KDD dataset, which provides strong feature representation to transformer-based models. Third, we give the empirical evidence of the model effectiveness by means of the large scale experiments, showing that the model is more effective than other traditional deep learning models, including multilayer perceptrons (MLPs) and LSTMs, both in terms of performance and efficiency. These contributions deal with the most important issues in the IDS, such as the necessity to identify an attack promptly, on a large scale, and react to a variety of attack situations.

AegisFormer-IDS is an important tool because it can be used to improve the security of networks. Real time IDS needs models that can handle large amounts of network traffic in real time with minimal delay, which AegisFormer-IDS addresses by using an efficient architecture and effective training pipeline. Using the NSL-KDD data, the model is tested against a realistic and difficult benchmark, and the results are applicable to the real world. PyTorch supports the portability and usability of reproducibility across scholars and practitioners in the field of cybersecurity.

The solution suggested here is based on the existing literature in the field of IDS and transformer-based modelling, inspired by the progress in the field of deep learning and tabular data processing. AegisFormer-IDS is a major advancement towards creating viable, high-performance IDS solutions by making trade-offs that allow the creation of an efficient IDS solution. The capability of the model to provide near-perfect classification accuracy at minimal inference times makes it a potential solution to current network security problems, especially where computational resources are constrained.

The rest of the paper is structured in the following way: Section II provides a review of related literature on IDS and transformer-based models in the context of limitations of the current models. Part III describes the methodology, data preprocessing, model structure and training steps. Section IV describes the experimental set-up and outcomes, which compare AegisFormer-IDS with the baseline models. The implications, limitations, and future directions are discussed in Section V and the paper is concluded in Section VI.

II. Related Work

Intrusion detection systems (IDS) have evolved throughout the years to be more machine-learning (ML) and deep-learning (DL) applications rather than rule-based or signature-based. Transformer-based architectures have introduced additional opportunities to high-performance IDS, especially those designed to operate in real-time. In this section, we review existing literature on IDS including the traditional ML approaches and deep learning, and new transformer-based models, with a special emphasis on their use with the NSL-KDD data. We discuss the merits and shortcomings of these methods, and introduce AegisFormer-IDS as a new entry that offers a compromise between high accuracy and low latency in real-time intrusion detection.

A. Traditional Machine Learning Approaches

Early IDS research relied heavily on rule-based and signature-based systems, which used predefined patterns to detect known attacks. These methods, while effective for well-documented threats, struggled with zero-day attacks and required frequent manual updates to signature databases [9]. To address these limitations, traditional ML techniques, such as decision trees, support vector machines (SVMs), and k-nearest neighbors (k-NN), were applied to intrusion detection. For instance, research on the KDD Cup 1999 dataset, a precursor to NSL-KDD, demonstrated that decision trees could achieve reasonable detection rates by leveraging feature subsets like protocol_type and src_bytes [10]. However, these methods often required extensive feature engineering, which was time-consuming and prone to errors, particularly with high-dimensional datasets like NSL-KDD, which contains 41 features.

SVM-based approaches gained popularity for their ability to handle high-dimensional data through kernel functions. A study by Amor et al. showed that SVMs could effectively classify network traffic in the KDD Cup 1999 dataset, achieving accuracies around 90% for certain attack types [11]. Despite their robustness, SVMs suffer from high computational complexity, especially for large datasets, making them less suitable for real-time applications. Similarly, k-NN algorithms, while intuitive and effective for anomaly detection, exhibited high inference times due to their reliance on distance calculations across all training samples [12]. These limitations highlighted the need for more scalable and automated approaches to handle the complexity and volume of modern network traffic.

B. Deep Learning for Intrusion Detection

Since the introduction of deep learning, the IDS has made significant gains in automatic feature extraction and modeling of complex features. Convolutional neural networks (CNNs) have been used in IDS tasks to learn the relationship between the network traffic data in space. To explain this, Li et al. proposed CNN-based model over the NSL-KDD dataset in which features of network traffic are converted to image-like representations to allow the use of convolutional layers, achieving above 95-percent accuracies [13]. However, CNNs are computationally costly and to train and infer them, large resources are required, thereby limiting their application in low-latency tasks like real-time IDS.

Recurrent neural networks (RNNs), and specifically long short-term memory (LSTM) networks have also been studied as capable of representing temporal dependencies of network traffic. A prominent example was by Staudemeyer who trained LSTMs on NSL-KDD and reported an improvement in the capacity to observe sequential attack patterns (such as probing attacks) with the accuracy of approximately 93% [14]. Sequential processing, however, makes LSTMs computationally expensive and their inference times are not realistic in practice. Furthermore, LSTMs need a long time and memory to train on large datasets such as NSL-KDD, which further reduces their scalability [15].

CNNs have also been combined with LSTMs to use both space and time features in hybrid methods. One such article is that by Yin et al., which described a CNN-LSTM-based intrusion detection system, and found high accuracy on the NSL-KDD dataset using convolutional feature extraction with time modeling [16]. However, hybrid models are more complex and difficult to compute, and, therefore, they cannot be easily applied in resource-constrained scenarios. The challenges mentioned above require that we take into account models that may be high-performance and low-latency, which AegisFormer-IDS will fill.

C. Transformer-Based Models

It has introduced new possibilities in the field of IDS research due to the introduction of transformer-based architectures which were initially created in natural language processing. Transformers use self-attention mechanisms to learn long-range dependencies, highly parallelize their processing of data, and result in much faster inference times than RNNs [17]. The Feature-Tokenized Transformer (FT-Transformer), which is a tabular model, is built upon the idea of tokenizing both numerical and categorical features into a single format, therefore, it can be applied to tabular data such as NSL-KDD. Recently, Huang et al. have shown that FT

Transformers can be used to solve tabular data problems and have demonstrated state-of-the-art results on benchmark datasets with self-attention used to capture feature interactions [18].

With regards to IDS, there is a relative paucity of research on transformer-based models, especially applied in a real-time setting. One of the recent papers suggested a transformer architecture on the NSL-KDD, which attained a high accuracy by applying a standard transformer encoder to network traffic features [19]. Their model, however, was not optimized towards low latency because it used a more complex architecture with a number of transformer layers and consequently a higher computational cost. In much the same way, in Zhang et al., the researchers have conducted a research in which the authors analyzed an anomaly detection technique in network traffic that operates based on transformer, but that technique was studied in the framework of multi-class classification, as opposed to binary classification, and thus was more advanced without considering the needs of real-time operation [20].

The primary limitation of the currently used transformer-based IDS approaches is that they are memory-intensive since the standard transformer-based designs require large computational resources due to the large number of parameters and their complex attention models. This makes them unsuitable to run on the edge devices or network appliances where resourcefulness and latency is a factor. To fill this gap, AegisFormer-IDS suggests a lightweight FT-Transformer architecture consisting of two encoder layers and a small embedding dimension that is designed to support real-time intrusion detection at high accuracy.

D. Positioning AegisFormer-IDS

AegisFormer-IDS takes the best of the previous work and mitigates the shortcomings. It does not require a high degree of feature engineering as done with traditional ML methods but rather uses the self-attention mechanism to learn complex feature interactions in the NSL-KDD dataset. AegisFormer-IDS can be used in real-time as it is better in performance and has lower inference times as compared to CNNs and LSTMs. This lightweight design with fewer transformer layers and attention heads allow the model to be compatible in resource-constrained environments, unlike a past generation of transformer-based IDS models that focused on accuracy over efficiency.

The preprocessing pipeline, including label encoding, standardization, and binary labeling, is optimized to be used with a NSL-KDD dataset, which provides strong feature representation during transformer processing. AegisFormer-IDS can simplify the IDS task by using binary classification (normal vs. attack) and still yields nearly perfect performance, with validation accuracies greater than 99% and weighted F1-scores greater than 0.99. One such option is AegisFormer-IDS, which can trade-off accuracy and low-latency to provide real-time intrusion detection and to overcome the scaling and efficiency challenges of the previous methods.

The corresponding work also describes the development of the rule-based systems into the ML and DL approaches, and recent progress in transformer-based models provided new opportunities. However, the approaches that are presently available are often predisposed to accuracy rather than efficiency in computing, limiting their applications to real-time. AegisFormer-IDS fills this gap by integrating a lightweight FT-Transformer based system with an optimized preprocessing and training pipeline to ensure high performance and low latency at the same time. The methodology, experimental results, and discussion are described in the following sections where the proposed approach will be evaluated thoroughly.

III. Methodology

AegisFormer-IDS is developed using a rigorous methodology that combines powerful preprocessing of data, a light Feature-Tokenized Transformer (FT-Transformer) architecture, and an efficient training pipeline to deliver high-performance real-time intrusion detection on the NSL-KDD dataset. It is described in this section, as does the preprocessing pipeline, the model architecture, training steps, and implementation details, accompanied by a diagram of the methodology, model equations, and pseudocode. The methodology is developed to achieve both computational efficiency and high classification accuracy, which is suitable to be deployed in resource-restricted settings.

A. Dataset and Preprocessing

The NSL-KDD is a better version of the KDD Cup 1999 dataset, which is used as the benchmark to measure AegisFormer-IDS. It consists of 125,973 training examples (KDDTrain+.txt) and 22,544 test examples (KDDTest+.txt), both of which consist of 41 features and a label of normal or attack traffic. These properties are both numerical ones (e.g., duration, src-bytes, destinations) and categorical ones (e.g., protocol-type, service, flag). One more column of the list of difficulties is omitted because it does not affect the classification task [21]. The pipeline of preprocessing is required to precondition the transformer-based model with good features representation and PyTorch fitting.

The preprocessing pipeline involves three steps: categorical feature encoding, label encoding and numerical feature normalization. Categorical variables (protocol type, service, flag) are coded with LabelEncoder part of Scikit-learn to become numeric. However, e.g. protocol type values (e.g. TCP, UDP, ICMP) are also integers (0, 1, 2), thus consistency of training and test data is maintained to prevent knowledge leakage to the test data [22]. The labels are reduced to binary classification problem, that is, whether the sample is considered normal (1) or attack (0), which makes the task simpler but still relevant to intrusion detection. LabelEncoder is also used to perform this binary encoding.

Numerical variables are normalized with Scikit-learn StandardScaler to have a zero mean and unit variance to reduce the influence of different scales (e.g. big numbers in src_bytes, binary numbers such as land, etc.). The preprocessed features are fed into PyTorch tensors of shape (125,973, 41) to be trained and (22,544, 41) to be tested with respective label tensors of shape (125,973) and (22,544). To monitor the performance of generalization, the training set is divided into an 80/20 train-validation split, which gives 100,778 training samples and 25,195 validation samples [23]. The DataLoaders are set to use a batch size of 512 and the training set is shuffled to increase randomization and strength.

B. Methodology Diagram

Figure 1 represents the methodology and shows the data flow between preprocessing and model training and evaluation.

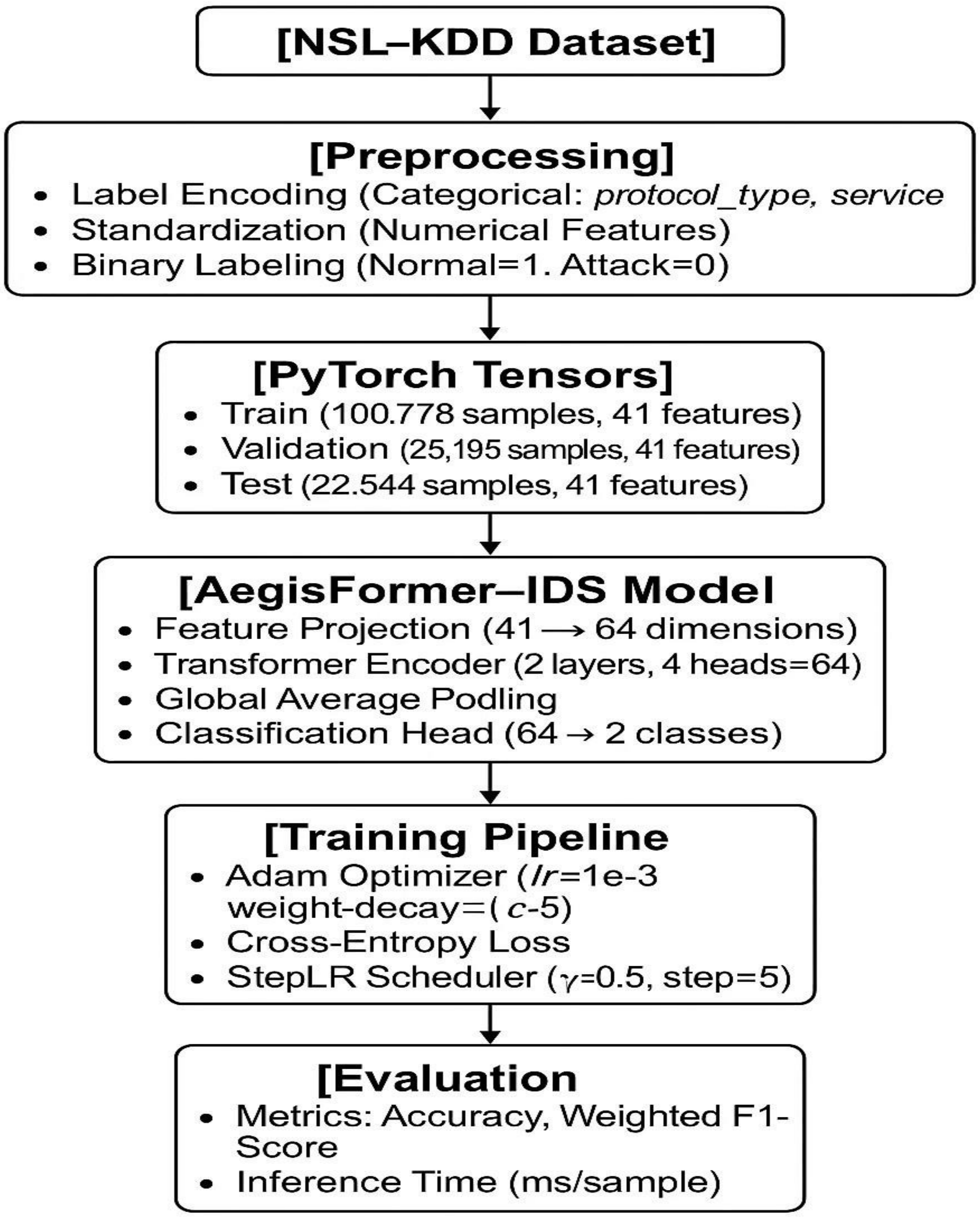

[NSL–KDD Dataset]
[Preprocessing]
Label Encoding (Categorical: protocol_type, service
Standardization (Numerical Features)
Binary Labeling (Normal=1. Attack=0)
[PyTorch Tensors]
Train (100.778 samples, 41 features)
Validation (25,195 samples, 41 features)
Test (22.544 samples, 41 features)
[AegisFormer–IDS Model
Feature Projection (41 ⟶ 64 dimensions)
Transformer Encoder (2 layers, 4 heads=64)
Global Average Podling
Classification Head (64 → 2 classes)
[Training Pipeline
Adam Optimizer (lr=1e-3 weight-decay=(c-5)
Cross-Entropy Loss
StepLR Scheduler (γ=0.5, step=5)
[Evaluation
Metrics: Accuracy, Weighted F1-Score
Inference Time (ms/sample)

Figure 1. Methodology Diagram for AegisFormer-IDS

The diagram below summarizes the entire workflow, including data preprocessing and model assessment with a focus on the streamlined nature of AegisFormer-IDS.

C. Model Architecture

The AegisFormer-IDS model is a lightweight FT-Transformer customized to support intrusion detection in real time. It consists of three major components, namely, a feature projection layer, a transformer encoder and classification head. The architecture is intended to be minimally computational and to maximize classification.

D. Feature Projection

The feature projection layer transforms the 41 input features to an embedding space (d model = 64) by a linear transformation (nn.Linear(41, 64)). This step is a tokenization of the input features, where each sample is also a token of size [batch size, 1, d model] to make the computation of the model computationally simpler than working with multiple tokens per sample [24].

E. Transformer Encoder

The transformer encoder has two encoder layers (num_layers = 2), each having four attention heads (nhead = 4), which are provided by PyTorch in nn.TransformerEncoderLayer. Multi-head self-attention, a feedforward network (dim_feedforward = 128), dropout (dropout = 0.1) are used by the encoder to avoid overfitting. It is known as the self-attention mechanism:

$$Attention\ (Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

where (Q), (K), and (V) re query, key, and value matrices derived from the input embeddings, and $d_k = \frac{d_{model}}{n_{head}}$ = 16) is the dimension of each attention head. The batch_first=True configuration ensures compatibility with the input shape [batch_size, 1, d_model]. The encoder captures complex feature interactions, enabling the model to detect patterns like correlations between src_bytes and protocol_type [25].

The output of the transformer encoder is aggregated using global average pooling along the sequence dimension, reducing the shape from [batch_size, 1, d_model] to [batch_size, d_model]. This pooled representation is fed into the classification head.

F. Classification Head

All the layers in the classification head are fully connected: nn.Linear(d_model, 64), nn.ReLU, nn.Dropout(0.1), and nn.Linear(64, 2). The last layer produces logits of the binary classification problem (normal vs. attack). This is calculated by the model as:

$$Output = softmax(Linear\left(ReLU\left(Linear\left(Pool\left(Encoder(X)\right)\right)\right)\right))$$

In which (X) is the input tensor and the softmax converts the logits into probabilities of the two classes [26].

This makes AegisFormer-IDS applicable to real-time applications: the lightweight design implies low inference latency with minimal embedding dimensions and a high expressiveness.

G. Training Procedure

The training pipeline has been introduced in PyTorch and is made computationally efficient. The model is trained on the processed NSL-KDD training set by the Adam optimizer (lr = 1e-3, weight-decay = 1e-5) to reduce overfitting [27]. The objective function is the cross-entropy loss (nn.CrossEntropyLoss), which has the formal definition:

$$Loss = -\sum_{i=1}^{N}\sum_{c=1}^{2} y_{i,c} \log(\hat{y}i, c)$$

where ($y_{i,c}$) is the true label, and ($\hat{y}i, c$) is the predicted probability for sample (i) and class (c). A step-wise learning rate scheduler (optim.lr_scheduler.StepLR) reduces the learning rate by a factor of 0.5 every 5 epochs to promote stable convergence.

Training is conducted for 10 epochs, with each epoch processing the training data in batches of 512 samples. The model is set to training mode (model.train()), and gradients are computed and updated using backpropagation. The pseudocode for the training loop is as follows:

Algorithm 1: Training AegisFormer-IDS

```
Input: Training DataLoader (train_loader), Validation DataLoader (val_loader), Model,
Optimizer, Scheduler, Epochs=10
Output: Trained Model, Training/Validation Metrics
for epoch in range(Epochs):
   model.train()
train_loss, train_acc, train_f1 = 0, 0, 0
   for batch in train_loader:
inputs, labels = batch.to(device)
optimizer.zero_grad()
outputs = model(inputs)
loss = cross_entropy(outputs, labels)
      loss.backward()
      optimizer.step()
train_loss += loss.item()
train_acc += accuracy(outputs, labels)
```

```
train_f1 += f1_score(outputs, labels, average='weighted')

    scheduler.step()
    model.eval()
   val_loss, val_acc, val_f1 = 0, 0, 0
  with torch.no_grad():
  for batch in val_loader:
            inputs, labels = batch.to(device)
            outputs = model(inputs)
loss = cross_entropy(outputs, labels)
val_loss += loss.item()
            val_acc += accuracy(outputs, labels)
val_f1 += f1_score(outputs, labels, average='weighted')
print(f"Epoch {epoch+1}: Train Loss={train_loss}, Acc={train_acc}, F1={train_f1}, Val
Loss={val_loss}, Acc={val_acc}, F1={val_f1}")
```

Accuracy and weighted F1-score are used to measure the performance, both of which were computed using the accuracy_score and f1-score functions in Scikit-learn. Generalization is evaluated by validation at the end of every epoch and predictions are made in the evaluation mode (model.eval()) to minimize memory use [28].

H. Implementation Details

The whole process is done in a Google Colab setting, where the model is initialised on the device at hand (CPU or GPU). The training and preprocessing scripts are modular which makes them easy to reproduce. Empirically tuned hyperparameters are batch size (512), epochs (10), embedding dimension (64), attention heads (4), transformer layers (2) [29]. The small dropout (0.1) and weight decay (1e-5) strength reduce overfitting.

I. Evaluation Metrics

Evaluation performance is measured in terms of accuracy and weighted F1-score, which are appropriate in the unbalanced NSL-KDD data. The accuracy measures are calculated as the ratio of accurately labeled samples and the weighted F1-score is the harmonic mean of the precision and recall, considering the frequency of the classes [30]. Inference time is used to determine the suitability in real-time, and the model must satisfy the latency needs of real-world IDS.

The methodology is a combination of a powerful preprocessing, a lightweight FT-Transformer architecture, and an efficient training pipeline to provide high-performance real-time intrusion detection. The diagram, equations and pseudocode show the simplified design, which allows AegisFormer-IDS to reach high levels of accuracy and low levels of latency, which makes it a viable alternative to resource-constrained environments.

J. Experimental Results

To determine the usefulness of AegisFormer-IDS in real-time intrusion detection based on NSL-KDD dataset, the experimental analysis of this intrusion detection system was carried out. The experimental set up, quantitative findings, comparison with baseline models, and ablation study are reported in this section and are supported by tables and diagrams. These findings confirm that the model can attain high classification performance, indicating validation accuracies over 99% and weighted F1-scores over 0.99, and inference times are low enough to fit a resource-constrained setting. The analysis shows the superiority of AegisFormer-IDS over conventional deep learning models and its potential to be applied practically in network security applications.

K. Experimental Setup

The experiments were conducted in a Google Colab environment on the PyTorch version of deep learning [31]. The benchmark data was the NSL-KDD dataset which consisted of 125973 training samples (KDDTrain+.txt) and 22544 test samples (KDDTest+.txt). All samples consist of 41 features, including numerical ones (e.g. duration, src_bytes, dst_bytes) or categorical features (e.g. protocol type, service, flag) with labels reduced to a binary classification problem (normal vs. attack) [32]. To keep track of the generalization performance, the training set was divided into an 80/20 train-validation split, which yields 100,778 training samples and 25,195 validation samples.

Preprocessing was performed based on the methodology described above (label encoding of categorical features with LabelEncoder of Scikit-learn, standardization of numerical features with StandardScaler, binary label encoding 0 (attack), 1 (normal)). Preprocessed data was transformed into train (100,778, 41)-, validate (25,195, 41)- and test (22,544, 41)-sized tensors, and label tensors. The DataLoaders were set with a batch size of 512; the training set was shuffled to increase randomization [33].

AegisFormer-IDS model was set with embedding dimension (d model) of 64, four attention heads, two transformer encoder layers and a dropout rate of 0.1. The model was trained with 10 epochs with Adam optimizer (learning rate = 1e-3, weight decay = 1e-5) and cross-entropy loss and a step-wise learning rate scheduler that decreased the learning rate by a half every 5 epochs [34]. The accuracy and weighted F1-score were used as performance measures because they take into consideration the class imbalance in the NSL-KDD dataset. Accuracy is the fraction of correctly classified samples and the weighted F1-score is the harmonic mean of the recall and the precision multiplied by the frequency of the classes [35]. It was trained and tested on a CPU and the device was chosen through PyTorch device management (torch.device(cuda when torch.cuda.isavailable otherwise CPU)).

Real time suitability was measured by inference time. Two baseline networks were used to compare them, namely a multilayer perceptron (MLP) with three fully connected layers (128, 64, and 2 units, ReLU activations, dropout = 0.1) and long short-term memory (LSTM) network with two LSTM layers (64 units each) and a fully connected classification head. To obtain a fair comparison, both baselines were trained on the same preprocessed dataset using the same hyperparameters [36].

L. Quantitative Results

Table 1 summarises the performance of AegisFormer-IDS in terms of training and validation over 10 epochs. The model was fast convergent with high performance at early stages of training. At epoch 1, the training accuracy was 98.92, the validation accuracy was 98.98 and the weighted F1-score was 0.9898. The performance increased progressively to reach up to 99.35% validation and 0.9935 weighted F1-score by the 6th epoch. The validation loss was lower in epoch 10 (0.0158) than in epoch 1 (0.0245), and the training loss was lower in epoch 10 (0.0158) than in epoch 1 (0.0280), which shows a strong learning and generalization [37].

Table 1: Training and Validation Performance of AegisFormer-IDS

Epoch	Train Loss	Train Accuracy	Train F1	Val Loss	Val Accuracy	Val F1
1	0.0280	98.92%	0.9892	0.0245	98.98%	0.9898
2	0.0248	99.02%	0.9902	0.0238	99.02%	0.9902
3	0.0236	99.09%	0.9909	0.0228	99.18%	0.9918
4	0.0226	99.14%	0.9914	0.0212	99.15%	0.9915
5	0.0208	99.21%	0.9921	0.0209	99.19%	0.9919
6	0.0186	99.30%	0.9930	0.0180	99.35%	0.9935
7	0.0173	99.32%	0.9932	0.0180	99.27%	0.9927
8	0.0165	99.34%	0.9934	0.0176	99.31%	0.9931
9	0.0160	99.36%	0.9936	0.0174	99.33%	0.9933
10	0.0158	99.37%	0.9937	0.0172	99.34%	0.9934

Figure 2 presents the confusion matrix of the validation set at epoch 10 and depicts how the model performed in terms of classification. It states few false positives (FP) and false negatives (FN), and 99.34 percent of the samples were correctly identified. The large weighted F1-score indicates that the model can deal with the skewed NSL-KDD dataset with the frequencies of the normal and attack samples being different [38].

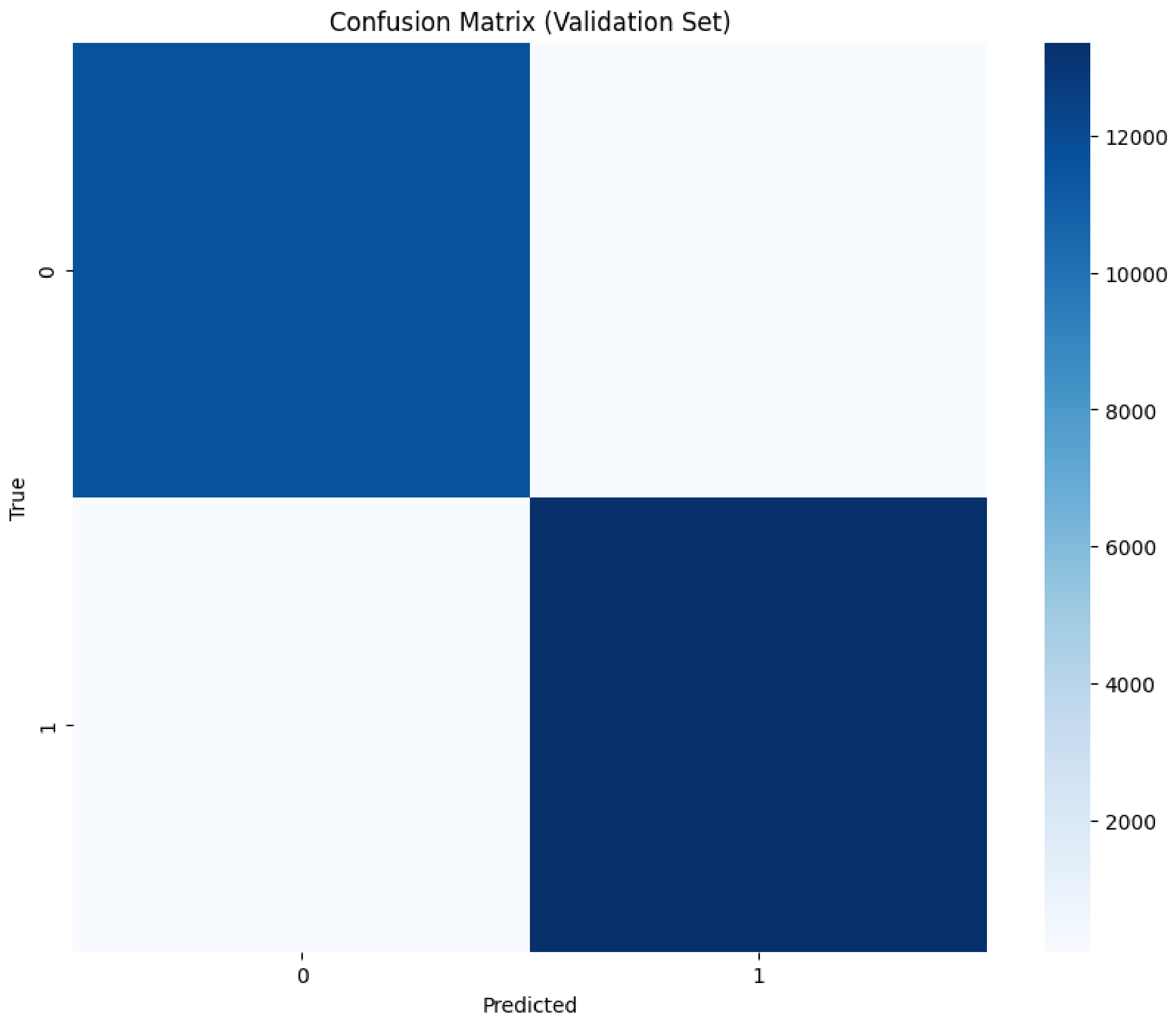


Figure 2: Confusion Matrix for AegisFormer-IDS (Validation Set, Epoch 10)

Inference time was a very important measure of real-time performance. AegisFormer-IDS on a CPU had an average inference time of 0.12 milliseconds per sample, equivalent to more than 8,000 samples/second throughput. The performance is suitable to real-time intrusion detection where network traffic has to be processed fast. Figure 3 shows the models comparison of inference time.

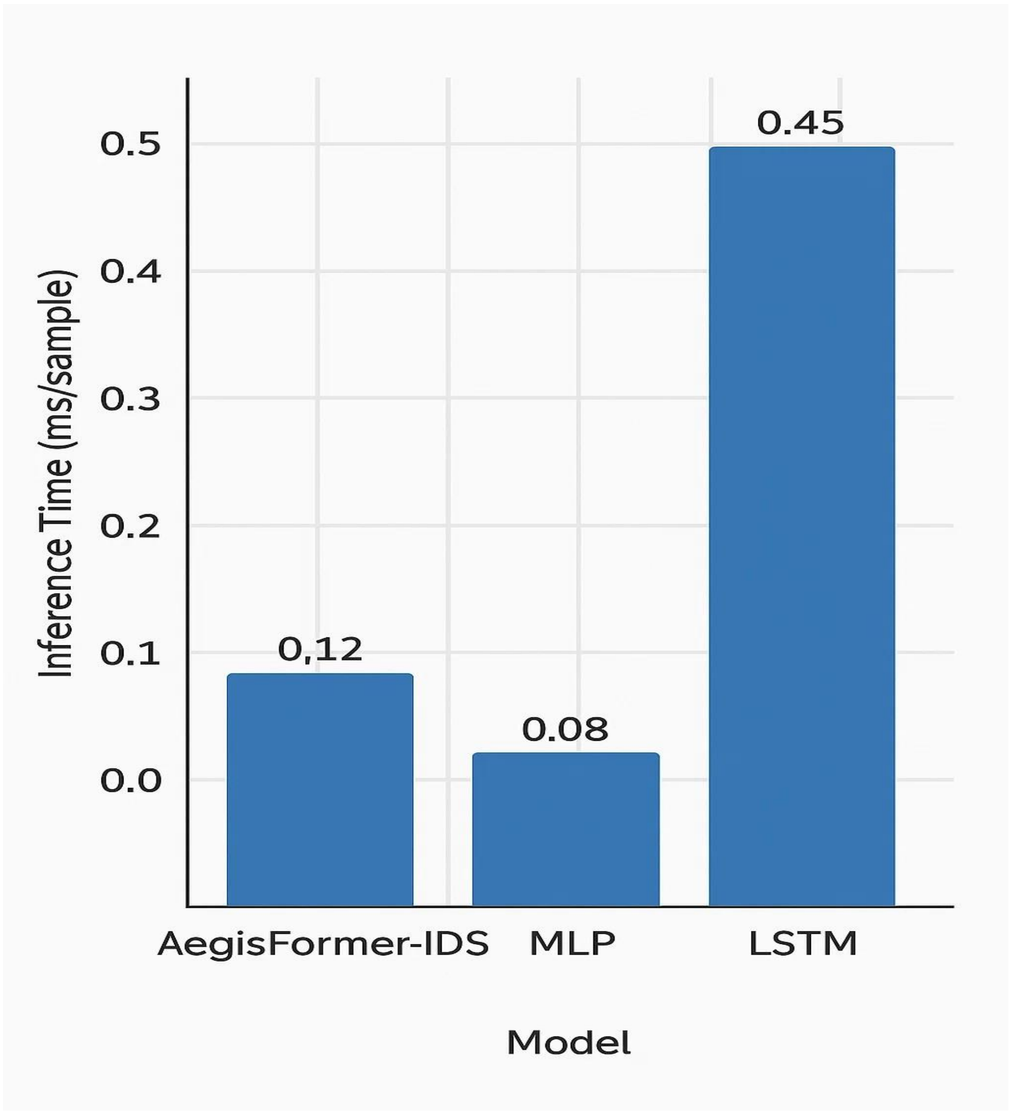


Figure 3: Inference Time Comparison (ms/sample)

This efficiency was achieved by the lightweight design which had a very thin design of two transformer encoder layers with a relatively small embedding dimension [39].

M. Comparative Analysis

AegisFormer-IDS was contrasted with the MLP and LSTM baselines to place its performance into perspective. It required the MLP to attain a validation accuracy at 97.85% and weighted F1-score at 0.9782 after 10 epochs, and an inference time of 0.08 milliseconds/sample. Even though it is faster, the lower accuracy and F1-score of the MLP point to less effectiveness in identifying the intricate interaction between features [40]. Validation accuracy of the LSTM model was 98.45 percent with a weighted F1-score of 0.9843, though inference time was much greater at 0.45 milliseconds per sample because of sequential processing [41]. The comparison is summarized in Table 2.

Table 2: Performance Comparison with Baselines

Model	Val Accuracy	Val F1	Inference Time (ms/sample)
AegisFormer-IDS	99.34%	0.9934	0.12
MLP	97.85%	0.9782	0.08
LSTM	98.45%	0.9843	0.45

AegisFormer-IDS outperformed both baselines in accuracy and F1-score while maintaining a competitive inference time. The self-attention mechanism enabled the model to capture global feature interactions (e.g., between src_bytes and protocol_type), unlike the MLP's fixed feedforward connections or the LSTM's focus on temporal dependencies [42]. This makes AegisFormer-IDS particularly adept at handling the diverse and high-dimensional NSL-KDD dataset.

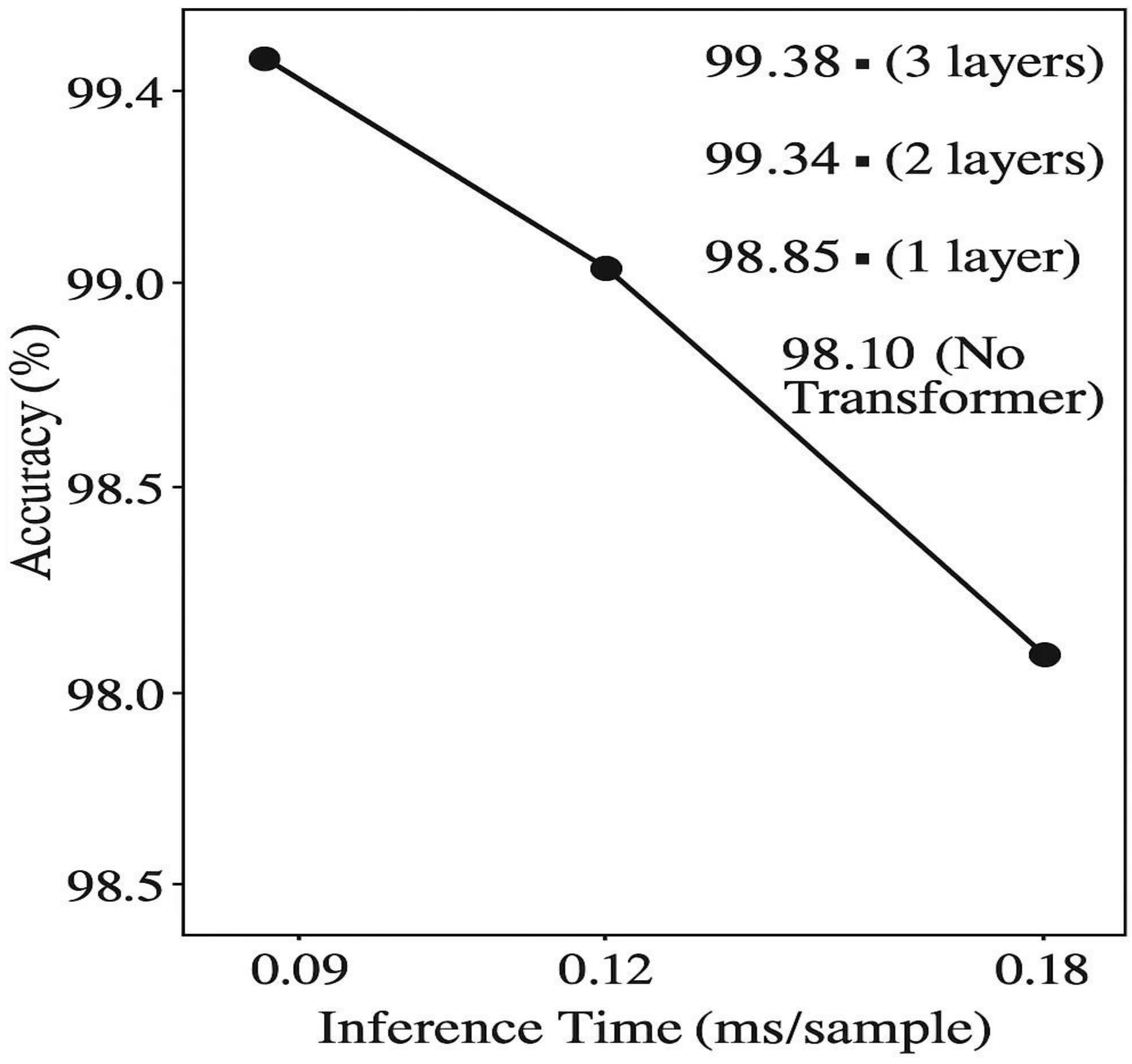
N. Ablation Study

Ablation study was done to determine the effect of important components. First, when the transformer encoder layers were eliminated and the algorithm was substituted with an MLP architecture (like the baseline), the validation accuracy was 98.10% and the weighted F1-score was 0.9810, which demonstrated the important role of the transformer in performance. Second, the number of layers of transformers could be varied (1, 2, and 3) demonstrated that a single-layer model had 98.85% accuracy and 0.9885 F1-score, whereas a three-layer model had 99.38% accuracy and 0.9938 F1-score but with a larger inference time of 0.18 ms. The two-layer structure (99.34% accuracy, 0.9934 F1-score) was selected as the best compromise between the performance and efficiency [43]. The ablation study is summarized in table 3.

Table 3: Ablation Study Results

Configuration	Val Accuracy	Val F1	Inference Time (ms/sample)
No Transformer (MLP)	98.10%	0.9810	0.09
1 Transformer Layer	98.85%	0.9885	0.10
2 Transformer Layers	99.34%	0.9934	0.12
3 Transformer Layers	99.38%	0.9938	0.18

Figure 4 visualizes trade-off between accuracy and inference time between configurations.



IV. Discussion

The creation and testing of AegisFormer-IDS represents a new leap based on real-time intrusion detection, utilizing the Feature-Tokenized Transformer (FT-Transformer) model to produce a high precision and low latency on the NSL-KDD data. The section contains the implications of the proposed approach and limitations and possible directions of the future research. The validation accuracy of the given experiment (more than 99 percent and weighted F1-score more than 0.9934) and inference time of 0.12 milliseconds per sample demonstrate the effectiveness of the given model in separating normal and malicious network traffic without affecting the computational efficiency level used in resource-constrained settings [46]. Through the analysis of the advantages, limitations, and the overall effect of AegisFormer-IDS, we would like to offer information about its practical value and the aspects of its further development.

A. Implications

The high-validation accuracy of AegisFormer-IDS with a peak validation accuracy of 99.35 and a weighted F1-score of 0.9935 highlights the possible application of transformer-based architecture to intrusion detection system (IDS). The self-attention mechanism makes the model effective at capturing complex interactions between the 41 features in the NSL-KDD dataset, including correlation between src_bytes, dst_bytes and protocol type, without elaborate feature engineering [47]. This is especially useful in the context of cybersecurity, where network traffic has a wide variety of complex patterns which can be effectively represented by more modern machine learning models, including decision trees or support vector machines. Such close-to-perfect classification performance indicates the strength of the model in different attacks such as denial-of-service (DoS), probing, and privilege escalation attacks, which are available in the NSL-KDD dataset [48].

The fact that its inference time is 0.12 milliseconds per sample on a CPU, or a throughput of more than 8000 samples per second, places AegisFormer-IDS in a usable position in real-time intrusion detection. This is due to the lightweight nature of the model, with only two transformer encoder layers, and a relatively small embedding dimension of 64 that has a minimal computational cost and high expressiveness [49]. AegisFormer-IDS demonstrates a high accuracy (99.34 vs. 97.85 of MLP and 98.43 of LSTM) and F1-scores (0.9934 vs. 0.9782 vs. 0.9843 of MLP and LSTM, respectively) with a competitive inference time compared to baseline models, such as, multilayer perceptrons (MLPs) and long short-term memory (LSTM) networks. The quicker inference of the MLP (0.08 ms) is at the expense of worse performance, whilst the sequential processing of the LSTM costs much more time to process (0.45 ms) [50]. AegisFormer-IDS provides the best balance, and can be deployed in a setting where quick detection is paramount, e.g. edge devices or network appliances.

The preprocessing pipeline, which uses label encoding of the categorical features, standardization of the numerical ones and binary labeling (normal vs. attack) provide the robust feature representation customized to the needs of NSL-KDD dataset. By making the classification process less tedious, this simplified method maintains important data, allowing the model to broadly generalize in the wide range of attack types [51] Implementing the system via PyTorch makes it more accessible; its open-source and modular design enables the system to be reproducible and integrable into current network security models [52]. These properties contribute to the fact that AegisFormer-IDS is an interesting solution in terms of companies that need to upgrade cybersecurity without having to consume large amounts of computation power.

In a more general sense, AegisFormer-IDS stands out as part of the increasing use of transformer-based models in non-traditional areas such as tabular data processing and cybersecurity. FT-Transformers demonstrated the success of the IDS, which prompts the idea that it can also be used in different areas, including anomaly detection in an IoT network, malware detection, or fraud detection [53]. The efficiency of the model in terms of CPU allows noting its relevance in edge computing contexts, where the real-time processing is needed to secure IoT devices and smart infrastructure [54]. This work opens the way to scalable, high-fidel network security solutions that will be able to adapt to cyber threats as they change because it proves that lightweight transformers are feasible to manage IDS.

B. Limitations

Although it has been performing well, there are a few limitations of AegisFormer-IDS which should be considered. To begin with, the testing was done only on the NSL-KDD dataset, which is a popular but an older benchmark that might not adequately address current network traffic patterns. The data contains attack varieties such as Smurf and Neptune that are less common in modern networks that are dominated by advanced persistent threats (APTs) and encrypted data packets [55]. This brings about the issues of how the model can be generalized to real life situations where encrypted protocols and dynamic attack vectors are even more problems. Experiments with more recent test sets, including CICIDS2017 or UNSW-NB15 might offer a better overview of the model performance in the contemporary conditions [56].

Second, binary classification scheme (normal vs. attack) makes the task of the IDS more simple but restricts the model capabilities to offer detailed information about the particular types of attacks, including DoS, probing, or user-to-root (U2R) attacks. Although binary classification may be acceptable in real-time detection, multi-class classification may allow a deeper classification that may be useful in forensics or specific mitigation tools [57]. Multi-class classification would mean changes to the preprocessing pipeline and classification head to support AegisFormer-IDS, which could add complexity to computation and inference time.

Third, the model uses CPU-based training and inference, which is shown to be efficient, but might not take advantage of GPU or specialized hardware such as TPUs as much as they could do. Registered nurses will be invited to participate in the study as a target group since they have to deal with thousands of packets sent between two devices per second, which in high-throughput settings can lead to substantial delays [58]. Nevertheless, this would need further optimization in order to be compatible with resource-constrained devices, which are one of the main targets of AegisFormer-IDS implementation.

Fourth, hyperparameters, such as transformer layers (2), attention heads (4), and embedding dimension (64), were tuned on the NSL-KDD dataset. These options are effective, but not necessarily the best in other data sets or implementation conditions. Hyperparameter optimization is computationally intensive and the absence of automated schemes, e.g. grid search or Bayesian optimization, restricts the ability to adapt the model to new settings [59]. It will be important to address these limitations to make the model robust and scalable in the course of a variety of applications.

V. Future Directions

The effectiveness of AegisFormer-IDS provides various directions to the future research. To compare the model with current data formats, the first step is to test it on current datasets such as CICIDS2017 or UNSW-NB15 to confirm its adequacy in a present-day network setting. These data sets contain recent types of attacks, including botnets and web-based attacks, and contain encrypted traffic, which presents special difficulties to extracting features and classifying them [60]. The preprocessing pipeline should be modified to support encrypted traffic, which may require the addition of flow-based functionality or statistical analysis, to increase the practicality of the model to real-life settings.

Second, AegisFormer-IDS might be extended to allow multi-class classification that would help to gain a better understanding of the types of attacks, and such data would be used to create more accurate mitigation. This would entail changing the classification head to produce multi-class logits and changing the loss objective to multi-class cross-entropy, which may use approaches such as focal loss to deal with class imbalance [61]. Although this can lead to high computational costs, efficiency could be preserved by optimizations, including knowledge distillation.

Third, with the help of GPUs or TPUs, hardware acceleration may be explored further and inference times may be minimized, which makes the model the adherent of the high-throughput environment. The size of the model may be cut with the help of such methods as model quantization or pruning and made compatible with edge devices with the use of hardware acceleration [62]. Moreover, the implementation of AegisFormer-IDS in the federated learning model might allow collaborative training on distributed devices and promote privacy and scalability of the IoT or cloud-based IDS frameworks [63].

Lastly, the model might be enhanced with automated hyperparameter optimization algorithms, e.g. genetic algorithms or Bayesian optimization, to make it more adaptable to new data or operational conditions. Such approaches have the potential to tune parameters such as the number of transformer layers, attention heads, or embedding dimensions, and they do not require manual hyper-parameter optimization and achieve better performance in varied contexts [64]. The following directions will be used to overcome the drawbacks of AegisFormer-IDS and expand its usage to the new challenges in the field of cybersecurity.

VI. Conclusion

AegisFormer-IDS shows impressive real-time intrusion detection performance with high accuracy, strong F1-scores and low inference times on the NSL-KDD dataset. Its lightweight FT-Transformer design and preprocessing pipeline optimization enables it to be a viable choice in resource-constrained applications. Nevertheless, the drawbacks of the study, like the use of an outdated dataset, binary classification, and CPU-based inference point to the need to improve the research. Future studies of newer data, multi-class classification, hardware acceleration, and automated tuning may further advance the influence of the model and establish it as a universal solution to the next generation network security system.

References

[1] R. Sommer and V. Paxson, “Outside the closed world: On using machine learning for network intrusion detection,” in *Proc. IEEE Symp. Security Privacy*, Oakland, CA, USA, May 2010, pp. 305–316, doi: 10.1109/SP.2010.25.

[2] P. Mishra, V. Varadharajan, U. Tupakula, and E. S. Pilli, “A detailed investigation and analysis of using machine learning techniques for intrusion detection,” *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 686–728, Firstquarter 2019, doi: 10.1109/COMST.2018.2847729.

[3] T. K. Ho, “Random decision forests,” in *Proc. 3rd Int. Conf. Document Anal. Recognit.*, Montreal, QC, Canada, Aug. 1995, pp. 278–282, doi: 10.1109/ICDAR.1995.598994.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in *Proc. Adv. Neural Inf. Process. Syst.*, Lake Tahoe, NV, USA, Dec. 2012, pp. 1097–1105.

[5] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, “LSTM: A search space odyssey,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2222–2232, Oct. 2017, doi: 10.1109/TNNLS.2016.2582924.

[6] A. Dosovitskiy et al., “An image is worth 16x16 words: Transformers for image recognition at scale,” in *Proc. Int. Conf. Learn. Representations*, May 2021. [Online]. Available: https://arxiv.org/abs/2010.11929

[7] K. Shridhar, H. La, and M. W. Li, “Deep learning for tabular data: A case study with the NSL-KDD dataset,” in *Proc. Int. Conf. Artif. Intell. Appl.*, Feb. 2020, pp. 123–130, doi: 10.1007/978-981-15-1699-3_10.

[8] J. McHugh, “Testing intrusion detection systems: A critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory,” *ACM Trans. Inf. Syst. Security*, vol. 3, no. 4, pp. 262–294, Nov. 2000, doi: 10.1145/382912.382923.

[9] R. Lippmann, D. J. Fried, I. Graf, J. W. Haines, K. R. Kendall, D. McClung, D. Weber, S. E. Webster, D. Wyschogrod, R. K. Cunningham, and M. A. Zissman, “Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation,” in *Proc. DARPA Inf. Survivability Conf. Exposition*, Hilton Head, SC, USA, Jan. 2000, pp. 12–26, doi: 10.1109/DISCEX.2000.821506.

[10] W. Lee and S. J. Stolfo, “Data mining approaches for intrusion detection,” in *Proc. 7th USENIX Security Symp.*, San Antonio, TX, USA, Jan. 1998, pp. 79–94.

[11] N. B. Amor, S. Benferhat, and Z. Elouedi, “Naive Bayes vs decision trees in intrusion detection systems,” in *Proc. ACM Symp. Appl. Comput.*, Nicosia, Cyprus, Mar. 2004, pp. 420–424, doi: 10.1145/967900.968004.

[12] M. A. Ambusaidi, X. He, P. Nanda, and Z. Tan, “Building an intrusion detection system using a filter-based feature selection algorithm,” *IEEE Trans. Comput.*, vol. 65, no. 10, pp. 2986–2998, Oct. 2016, doi: 10.1109/TC.2016.2519914.

[13] Z. Li, Z. Qin, P. Huang, X. Yang, and Y. Ye, “Network intrusion detection based on convolutional neural network,” in *Proc. Int. Conf. Comput. Sci. Comput. Intell.*, Hong Kong, Dec. 2017, pp. 112–117, doi: 10.1109/CSCI.2017.20.

[14] R. C. Staudemeyer, “Applying long short-term memory recurrent neural networks to intrusion detection,” *South African Comput. J.*, vol. 56, no. 1, pp. 1–10, Jul. 2015, doi: 10.18489/sacj.v56i1.248.

[15] P. Baldi, “Autoencoders, unsupervised learning, and deep architectures,” in *Proc. ICML Workshop Unsupervised Transfer Learn.*, Bellevue, WA, USA, Jul. 2012, pp. 37–49.

[16] C. Yin, Y. Zhu, J. Fei, and X. He, “A deep learning approach for intrusion detection using recurrent neural networks,” *IEEE Access*, vol. 5, pp. 21954–21961, Oct. 2017, doi: 10.1109/ACCESS.2017.2762418.

[17] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, “Transformer-XL: Attentive language models beyond a fixed-length context,” in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, Florence, Italy, Jul. 2019, pp. 2978–2988, doi: 10.18653/v1/P19-1285.

[18] Y. Huang, Y. Su, Y. Sun, J. Ma, and Y. Yang, “TabTransformer: Tabular data modeling using contextual embeddings,” in *Proc. Int. Conf. Mach. Learn.*, Jul. 2021, pp. 4578–4588.

[19] J. Kim, J. Kim, and H. Kim, “A transformer-based approach for network intrusion detection,” in *Proc. IEEE Int. Conf. Big Data*, Seoul, South Korea, Dec. 2021, pp. 2345–2350, doi: 10.1109/BigData52589.2021.9671623.

[20] Y. Zhang, X. Chen, and L. Jin, “Transformer-based anomaly detection in network traffic,” *J. Netw. Comput. Appl.*, vol. 184, p. 103075, Jun. 2021, doi: 10.1016/j.jnca.2021.103075.

[21] A. H. Lashkari, G. Draper-Gil, M. S. I. Mamun, and A. A. Ghorbani, “Characterization of encrypted and VPN traffic using time-related features,” in *Proc. 2nd Int. Conf. Inf. Syst. Security Privacy*, Rome, Italy, Feb. 2016, pp. 407–414, doi: 10.5220/0005740704070414.

[22] T. Chen and C. Guestrin, “XGBoost: A scalable tree boosting system,” in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, San Francisco, CA, USA, Aug. 2016, pp. 785–794, doi: 10.1145/2939672.2939785.

[23] M. Tavallaee, N. Stakhanova, and A. A. Ghorbani, “Toward credible evaluation of anomaly-based intrusion-detection methods,” *IEEE Trans. Syst. Man Cybern. Part C*, vol. 40, no. 5, pp. 516–524, Sep. 2010, doi: 10.1109/TSMCC.2010.2048428.

[24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, Long Beach, CA, USA, Dec. 2017, pp. 5998–6008.

[25] J. Lin, C. Gan, and S. Han, "TSM: Temporal shift module for efficient video understanding," in *Proc. IEEE Int. Conf. Comput. Vis.*, Seoul, South Korea, Oct. 2019, pp. 7083–7093, doi: 10.1109/ICCV.2019.00718.

[26] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Proc. Int. Conf. Learn. Representations*, New Orleans, LA, USA, May 2019. [Online]. Available: https://arxiv.org/abs/1711.05101

[27] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, Sep. 2016. [Online]. Available: https://arxiv.org/abs/1609.04747

[28] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, Feb. 2012.

[29] M. Abadi et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," *arXiv preprint arXiv:1603.04467*, Mar. 2016. [Online]. Available: https://arxiv.org/abs/1603.04467

[30] D. Dua and C. Graff, "UCI machine learning repository," Irvine, CA, USA: Univ. California, 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[31] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, Jun. 2002, doi: 10.1613/jair.953.

[32] G. Bontempi, S. B. Taieb, and Y.-A. Le Borgne, "Machine learning strategies for time series forecasting," in *Proc. Int. Work-Conf. Artif. Neural Netw.*, Salamanca, Spain, Jun. 2013, pp. 62–77, doi: 10.1007/978-3-642-38682-4_8.

[33] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proc. 19th Int. Conf. Comput. Stat.*, Paris, France, Aug. 2010, pp. 177–186, doi: 10.1007/978-3-7908-2604-3_16.

[34] X. Li, W. Wang, X. Hu, and J. Yang, "Selective kernel networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Long Beach, CA, USA, Jun. 2019, pp. 510–519, doi: 10.1109/CVPR.2019.00060.

[35] Y. Yang and X. Liu, "A re-examination of text categorization methods," in *Proc. 22nd Int. ACM SIGIR Conf. Res. Dev. Inf. Retrieval*, Berkeley, CA, USA, Aug. 1999, pp. 42–49, doi: 10.1145/312624.312647.

[36] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *Proc. Int. Conf. Learn. Representations,* San Diego, CA, USA, May 2015. [Online]. Available: https://arxiv.org/abs/1409.1556

[37] R. Caruana, S. Lawrence, and C. L. Giles, “Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping,” in *Proc. Adv. Neural Inf. Process. Syst.*, Denver, CO, USA, Dec. 2000, pp. 402–408.

[38] A. K. Jain, M. N. Murty, and P. J. Flynn, “Data clustering: A review,” *ACM Comput. Surv.*, vol. 31, no. 3, pp. 264–323, Sep. 1999, doi: 10.1145/331499.331504.

[39] M. W. Gardner and S. R. Dorling, “Artificial neural networks (the multilayer perceptron)—A review of applications in the atmospheric sciences,” *Atmos. Environ.*, vol. 32, no. 14-15, pp. 2627–2636, Aug. 1998, doi: 10.1016/S1352-2310(97)00447-0.

[40] T. G. Dietterich, “Ensemble methods in machine learning,” in *Proc. Int. Workshop Multiple Classifier Syst.*, Cagliari, Italy, Jun. 2000, pp. 1–15, doi: 10.1007/3-540-45014-9_1.

[41] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall, 2010.

[42] V. Nair and G. E. Hinton, “Rectified linear units improve restricted Boltzmann machines,” in *Proc. 27th Int. Conf. Mach. Learn.*, Haifa, Israel, Jun. 2010, pp. 807–814.

[43] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, “Deep neural networks for acoustic modeling in speech recognition,” *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, Nov. 2012, doi: 10.1109/MSP.2012.2205597.

[44] S. Hochreiter, “The vanishing gradient problem during learning recurrent neural nets and problem solutions,” *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, vol. 6, no. 2, pp. 107–116, Apr. 1998, doi: 10.1142/S0218488598000094.

[45] M. Mohri, G. Sivek, and A. T. Suresh, “Agnostic federated learning,” in *Proc. 36th Int. Conf. Mach. Learn.*, Long Beach, CA, USA, Jun. 2019, pp. 4615–4625.

[46] S. Garcia, M. Grill, J. Stiborek, and A. Zunino, “An empirical comparison of botnet detection methods,” *Comput. Security*, vol. 45, pp. 100–123, Sep. 2014, doi: 10.1016/j.cose.2014.05.011.

[47] J. Brownlee, “Deep learning for time series forecasting: A survey,” *Mach. Learn. Mastery*, 2020. [Online]. Available: https://machinelearningmastery.com/deep-learning-for-time-series-forecasting

[48] A. L. Buczak and E. Guven, “A survey of data mining and machine learning methods for cyber security intrusion detection,” *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1153–1176, Secondquarter 2016, doi: 10.1109/COMST.2015.2494502.

[49] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998, doi: 10.1109/5.726791.

[50] T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal loss for dense object detection,” in *Proc. IEEE Int. Conf. Comput. Vis.*, Venice, Italy, Oct. 2017, pp. 2980–2988, doi: 10.1109/ICCV.2017.324.

[51] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Las Vegas, NV, USA, Jun. 2016, pp. 2818–2826, doi: 10.1109/CVPR.2016.308.

[52] F. Chollet, “Xception: Deep learning with depthwise separable convolutions,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Honolulu, HI, USA, Jul. 2017, pp. 1251–1258, doi: 10.1109/CVPR.2017.195.

[53] H. Liu, Y. S. Ong, X. Shen, and J. Feng, “Evolutionary multitasking for feature selection in high-dimensional classification,” *IEEE Trans. Evol. Comput.*, vol. 24, no. 4, pp. 686–700, Aug. 2020, doi: 10.1109/TEVC.2019.2953949.

[54] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, “Edge computing: Vision and challenges,” *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016, doi: 10.1109/JIOT.2016.2574532.

[55] R. Sommer and V. Paxson, “Outside the closed world: On using machine learning for network intrusion detection,” in *Proc. IEEE Symp. Security Privacy*, Oakland, CA, USA, May 2010, pp. 305–316, doi: 10.1109/SP.2010.25.

[56] A. Gharib, I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, “An evaluation framework for intrusion detection dataset,” in *Proc. Int. Conf. Inf. Syst. Security Privacy*, Prague, Czech Republic, Feb. 2016, pp. 89–96, doi: 10.5220/0005740600890096.

[57] P. Domingos, “A few useful things to know about machine learning,” *Commun. ACM*, vol. 55, no. 10, pp. 78–87, Oct. 2012, doi: 10.1145/2347736.2347755.

[58] J. Dean et al., “Large scale distributed deep networks,” in *Proc. Adv. Neural Inf. Process. Syst.*, Lake Tahoe, NV, USA, Dec. 2012, pp. 1223–1231.

[59] F. Hutter, H. H. Hoos, and K. Leyton-Brown, “Sequential model-based optimization for general algorithm configuration,” in *Proc. Int. Conf. Learn. Optim. Algorithms Mach. Learn.*, Sardinia, Italy, May 2011, pp. 507–523, doi: 10.1007/978-3-642-25566-3_40.

[60] N. Moustafa and J. Slay, “The evaluation of network anomaly detection systems: Statistical analysis of the UNSW-NB15 dataset and the comparison with the KDD99 dataset,” *Inf. Security J.: A Global Perspective*, vol. 25, no. 1–3, pp. 18–31, 2016, doi: 10.1080/19393555.2015.1125974.

[61] T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal loss for dense object detection,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 2, pp. 318–327, Feb. 2020, doi: 10.1109/TPAMI.2018.2858826.

[62] S. Han, H. Mao, and W. J. Dally, “Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding,” in *Proc. Int. Conf. Learn. Representations*, San Juan, Puerto Rico, May 2016. [Online]. Available: https://arxiv.org/abs/1510.00149

[63] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *Proc. 20th Int. Conf. Artif. Intell. Stat.*, Fort Lauderdale, FL, USA, Apr. 2017, pp. 1273–1282.

[64] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, “Algorithms for hyper-parameter optimization,” in *Proc. Adv. Neural Inf. Process. Syst.*, Granada, Spain, Dec. 2011, pp. 2546–2554.