Computer Networks and Communications
http://ojs.wiserpub.com/index.php/CNC/

UNIVERSAL WISER PUBLISHER

Research Article

SecureFormer—A Lightweight Transformer-Based Architecture for Network Intrusion Detection in Smart Home Environments

Oluwasegun Apejoye*, Nemitari Ajienka, Xiaoqi Ma, Jun He

Department of Computer Science, Nottingham Trent University, Nottingham, United Kingdom
*Correspondence: n0966184@my.ntu.ac.uk

Received: 13 May 2026; **Revised:** 3 June 2026; **Accepted:** 5 June 2026; **Published:** 22 June 2026

Abstract: This study presents SecureFormer, a lightweight transformer-based model designed for network-based intrusion detection in resource-constrained environments. The increasing complexity of IoT networks necessitates intrusion detection systems that can capture complex traffic patterns while remaining efficient enough for edge deployment. To address this challenge, the architecture combines optimised embedding dimensions, compact self-attention design, and a domain-specific tokenisation strategy that converts network flow features into sequence tokens, enabling effective modelling of network behaviour without reliance on large-scale Transformer architectures. The model was evaluated on the flow-based variant of two complex IoT-focused network intrusion detection benchmark datasets, CICIoT2023 and the Edge-IIoTset, with a reduced set of 15 network- and environment-agnostic features selected via feature importance analysis. Experimental results show that SecureFormer achieves a weighted F1 score of 0.99 on CICIoT2023 and 0.96 on Edge-IIoTset while maintaining a footprint of fewer than one million parameters and low inference overhead. The study demonstrates that compact transformer architectures can effectively capture high-level dependencies in structured network traffic, even under aggressive feature reduction. This capability validates their practical viability for real-time intrusion detection in IoT environments.

***Keywords*:** intrusion detection system, smart home security, transformers, network flows, Internet of Things (IoT)

1. Introduction

The domain of Intrusion Detection Systems (IDS) has witnessed a definitive paradigm shift over the last decade, moving from signature-based detection mechanisms, which rely on static patterns of known threats, to anomaly-based detection powered by Machine Learning (ML) and Deep Learning (DL) [1, 2]. This transition is driven by the necessity to detect novel, zero-day attacks and polymorphic malware that easily evade static signatures [3, 4]. An essential element needed to help power the transition to these ML-based solutions is the availability of high-quality network-traffic datasets annotated with ground-truth labels. Consequently, the research community has produced several benchmark datasets, including Edge-IIoT, CICIoT2023, CIC-IDS2017, CSE-CIC-IDS2018, NF-BoT-IoT-v2, NF-ToN-IoT-v2, ToN-IoT, BoT-IoT, and IoT-23, to provide the high-quality data needed for these AI-based systems [5].

With the growing processing power of computing systems and the increasing availability of massive datasets, machine learning algorithms have led to major breakthroughs in many different areas of computer security such as for malware

Copyright ©2026 Oluwasegun Apejoye, et al.
DOI: https://doi.org/10.37256/cnc.41202610097
This is an open-access article distributed under a CC BY license
(Creative Commons Attribution 4.0 International License)
https://creativecommons.org/licenses/by/4.0/

detection, vulnerability discovery, and binary code analysis [6–8]. Though classical ML models such as Support Vector Machine (SVM), Random Forest, and k-NN have shown improved performance in these areas of cybersecurity research, their reliant on manual feature extraction, which often fails to capture the non-linear, high-dimensional patterns of modern polymorphic attacks found in IoT security [4]. This has necessitated a transition toward more sophisticated paradigms, specifically DL models, which employ hierarchical architectures that enable automated feature representation learning to address the evolving complexity of Network Intrusion Detection Systems (NIDS) [9].

Recent advances in deep learning have demonstrated the effectiveness of transformer-based architectures for sequence modelling tasks [6, 10, 11]. Unlike recurrent models, transformer models use self-attention for long-range dependencies, making them well-suited to capture temporal and contextual patterns in network flows [12–14]. However, their high computational complexity renders them impractical for deployment on edge devices in smart home networks, necessitating the increased focus by researchers on lightweight solutions, including the use of DistilBERT-like encoders, shallow attention stacks, reduced heads, and model compression strategies such as quantisation, knowledge distillation, and pruning [15–17]. This study contributes to this solution by architecting a minimalist, yet effective lightweight Transformer encoder specifically optimised for network security applications in resource-constrained environments.

However, the performance of machine learning–based solutions depends not only on the choice of learning architecture but also on the quality and structure of the feature space used to train the model. For instance, high-dimensional network traffic data often exhibits multicollinearity and high-cardinality attributes, which can inflate computational cost, hinder generalisation, and limit the feasibility of lightweight, real-time deployment in resource-constrained smart home environments [12, 14, 18, 19]. To address these challenges, this study adopts a feature selection strategy based on Random Forest feature importance to identify a compact yet informative subset of traffic features prior to training the proposed SecureFormer model. This approach is motivated by the need to balance representational richness with computational efficiency, enabling systematic evaluation of how feature reduction impacts both classical and transformer-based NIDS architectures. By explicitly assessing model performance under full and reduced feature configurations, we show that feature reduction did not degrade performance and could improve detection accuracy.

This study makes the following scientific contributions to the field of network intrusion detection:

1. It demonstrates that lightweight Transformer architectures can effectively replace recurrent and hybrid deep learning models for structured network flow analysis in resource-constrained environments.

2. It provides empirical evidence that high-level contextual representations of network traffic can be preserved using a compact self-attention mechanism with significantly reduced model capacity.

3. It establishes that aggressive feature reduction does not necessarily degrade detection performance, offering insight into the relationship between feature dimensionality and model generalisation in IoT intrusion detection.

4. It introduces and evaluates a structured tokenisation strategy that enables transformer-based models to operate effectively on non-textual network flow data.

Together, these contributions advance the understanding of how Transformer architectures can be adapted and optimised for deployment in practical, resource-constrained cybersecurity settings.

1.1 *Study aims*

Our research aims to develop a ML model that generalises more effectively across a wide range of scenarios and can be deployed effectively to protect resource-constrained IoT devices. Therefore, we design SecureFormer for smart homes taking into account the IoT device resource constraints and we validated it utilising two comprehensive and diverse public benchmark datasets from diverse environments, CIC-IoT23 [20] and Edge-IIoT [21]. Key metrics, such as storage requirements, Random Access Memory (RAM) consumption, and prediction time, were combined with the diverse network traffic data to evaluate the model's generalisability and effectiveness. The goal of this paper is to present SecureFormer as a novel, lightweight Transformer-based intrusion detection model and to provide initial validation on public, complex benchmark datasets. The paper also presents a comparison between SecureFormer and other studies across key metrics such as latency, resource usage, and performance. The primary objective of this study is not to maximise predictive performance in isolation, but to establish whether a compact transformer architecture can achieve strong intrusion detection performance while remaining feasible for edge deployment.

2. Related work

The landscape of NIDS has undergone a significant paradigm shift, transitioning from static, rule-based systems to highly adaptive, AI-driven architectures. This section reviews the evolution of these methodologies, categorized into classical machine learning, deep learning frameworks, and the emerging field of Transformer-based architectures.

2.1 *Classical machine learning approaches*

Early ML-based NIDS widely adopted supervised learning models (e.g., SVMs, k-Nearest Neighbors (K-NN), decision trees, and ensemble methods such as Random Forests) trained on flow-level statistical features. Comprehensive reviews note that these methods can achieve strong performance when informative features are available, and they remain attractive due to interpretability and relatively modest training costs [18, 22, 23].

Random Forests and related tree ensembles have been repeatedly highlighted for their robustness to noisy features, capability to capture non-linear interactions, and provision of feature importance estimates that support feature selection and dimensionality reduction – an aspect particularly relevant for high-cardinality telemetry extracted from IoT traffic [24, 25].

According to a recent survey study on machine learning by Kumar et al. [24], the Random Forest (RF) model has consistently been recognised as the most accurate, being identified as the best performer in approximately twelve instances. Using RF, [25] achieved a notable accuracy rate of 94% in detecting Man-in-the-Middle (MiTM) attacks.

However, noted that classical models often fail to capture the complex, non-linear correlations present in modern encrypted traffic, and they often require well-crafted features [4]. Despite these strengths, classical ML approaches tend to depend on manual feature engineering and stable traffic distributions; both assumptions are frequently violated in smart-home environments where device behaviours drift (e.g., firmware updates) and traffic mixes change over time. Recent Surveys emphasise that such concept drift and heterogeneity degrade shallow model generalisability unless continuous re-training and careful feature curation are applied [7, 26].

2.2 *Deep learning frameworks*

Deep learning was introduced to reduce over-reliance on handcrafted features by learning hierarchical representations from traffic statistics, packet sequences, or hybrid views. Surveys and comparative studies report that DNNs, CNNs, RNNs (including Long Short-Term Memory (LSTM)/Gated Recurrent Unit (GRU)), and autoencoder-based models can improve detection of complex attacks by capturing non-linear and temporal dependencies that are difficult to express with shallow learners [9, 27].

CNN-based NIDS are typically motivated by their ability to extract local motifs from structured traffic representations (e.g., feature sequences or matrix reshaping). Although effective, CNNs may require deeper stacks or multi-scale design to capture long-range dependencies, which can increase compute overhead [28, 29].

RNN family models (LSTM/GRU) address temporal structure by processing ordered sequences of packets or flows, and they are frequently used to capture behaviours such as scanning, beaconing, and staged attacks. However, RNN-based architectures face sequential bottlenecks and the vanishing gradient problem, which hampers their ability to handle long-range dependencies necessary for detecting multi-stage “low-and-slow” persistent threats [12]. Additionally, their recurrent computation limits parallelism, leading to latency issues in real-time edge environments, especially when compared to fully parallel attention-based models [14].

Autoencoder-based anomaly detection gained prominence for scenarios where labelled attack data are limited or rapidly evolving. For example, Kitsune [30] demonstrates an online, lightweight ensemble of autoencoders designed for deployment even on constrained devices, illustrating that unsupervised learning can be viable for local network monitoring when engineering is aligned with resource constraints.

2.3 *Transformer-based and large language models like architectures*

The introduction of the Transformer model by [11] marked a significant shift in sequence modelling by replacing recurrence with a self-attention mechanism. This allows for massive parallelisation and the ability to capture global dependencies regardless of their distance in the sequence. Recently, researchers have begun adapting transformer-based models for the cybersecurity domain. Multiple surveys and empirical studies now identify transformer-based models as an emerging trend in intrusion detection and threat intelligence analysis [31–33].

BERT-style encoders extend transformer ideas via bidirectional pretraining objectives and have demonstrated that rich contextual representations can be transferred to downstream classification with modest task-specific heads. This paradigm motivated the treatment of network traffic as a language of tokens derived from flow features or packet elements [10, 34].

Empirical studies demonstrate that cybersecurity-adapted BERT architectures consistently outperform traditional deep learning approaches on real-world security datasets. A notable study is SecurityBERT [14], which pre-trains a BERT-like architecture on a massive corpus of Edge_IIoT data to understand the language of cyberthreats. They use a privacy-preserving BERT-based model optimised for IoT environments, demonstrating that large language model architectures can be designed for resource-constrained environments. Similarly, work by Yang and Peng [35] applied BERT-based encoders directly to network traffic data, treating individual flow fields as tokens. Unlike standard DL models, these BERT-inspired architectures utilise bidirectional context, allowing the downstream classifier to understand a packet's significance based on both preceding and succeeding traffic [35]. FlowTransformer has been proposed as a modular framework for evaluating transformer components across flow-based datasets. The author reports that architectural choices, particularly the classification head and input encoding, can materially affect performance, model size, and speed [10].

Moreover, the growing need for on-device inference—driven by concerns about privacy, latency, and cost—has spurred research into developing compact, lightweight models for edge deployment [36, 37]. Additionally, performance is enhanced as smaller models demonstrate significantly lower latency than their larger counterparts, thereby improving the overall throughput of the deployment. Studies on TinyML and edge-AI such as [15, 38], have demonstrated that transformer-class models can now run on microcontrollers under strict memory constraints using concepts such as knowledge distillation, quantisation, pruning, and new architectures that allow small models to approach the performance of much larger systems at a fraction of the cost.

Taken together, the literature indicates a shift in emphasis from demonstrating transformer superiority in controlled settings to engineering transformer-based NIDS that preserve representational benefits under stringent latency and memory constraints typical of smart-home gateways. Our work builds upon this trajectory by optimising a BERT-like architecture specifically for high-speed network telemetry, addressing the latency concerns often associated with large-scale Transformer models.

However, unlike prior Transformer-based IDS approaches that primarily emphasise either representational capacity or accuracy, SecureFormer is designed explicitly around the trade-off between contextual modelling strength and deployment efficiency, with the aim of proposing a transformer-based intrusion detector practical for resource-constrained IoT edge environments.

2.4 *Benchmark datasets for IoT/Edge NIDS*

The credibility of NIDS evaluation is strongly coupled to dataset realism, protocol diversity, and attack coverage. Recent datasets aim to address limitations of legacy benchmarks by incorporating modern IoT devices, diverse protocols, and broader attack taxonomies [5].

CICIoT2023 [20] is a large-scale IoT intrusion benchmark collected on an extensive topology with real IoT devices, executing 33 attacks grouped into seven categories (e.g., DDoS, DoS, reconnaissance, web-based, brute force, spoofing, and Mirai). Its scale and heterogeneity make it suitable for evaluating both multi-class detection performance and robustness under class imbalance.

Edge-IIoTset [21] provides a comprehensive IoT/IIoT dataset designed for both centralised and federated learning evaluation. The dataset documentation describes a layered testbed spanning cloud-to-perception components, includes multiple IoT/IIoT devices and protocols (e.g., Message Queuing Telemetry Transport (MQTT) and Modbus Transmission

Control Protocol/Internet Protocol (TCP/IP) among others), and enumerates fourteen attacks grouped into several threat categories, with a reduced set of 61 selected features to balance fidelity and tractability.

These datasets are particularly relevant for this study because they include diverse attack families and modern IoT traffic characteristics that stress-test generalisation beyond narrow lab conditions, while also enabling explicit study of efficiency constraints when targeting edge deployment. They also provide room for comparison with other studies.

3. SecureFormer: a lightweight transformer architecture for smart home NIDS

SecureFormer is a domain-specific, lightweight Transformer encoder designed explicitly for IoT network intrusion detection. Unlike BERT [34], which is a deep bidirectional transformer pre-trained on large-scale text corpora, SecureFormer is a lightweight transformer model tailored to meet the unique requirements of IoT security tasks. These requirements include low computational overhead, minimal memory usage, and efficient operation in resource-constrained environments such as IoT devices. SecureFormer preserves the contextual modelling strengths of transformers while significantly reducing memory footprint and computational overhead. Unlike conventional transformer-based security models, SecureFormer is trained from scratch, employs a single encoder block, and operates on structured traffic feature sequences rather than natural language tokens. This design enables transformer-level expressiveness while maintaining a footprint suitable for real-time edge deployment.

The design philosophy for SecureFormer follows three core principles: minimal depth and width to reduce model complexity by using a shallow transformer depth with reduced attention dimensionality; the absence of masked language modelling or linguistic pre-training; feature-aware tokenisation of network traffic; and carefully placed regularisation mechanisms (e.g., dropout, layer normalisation) for robustness without over-parameterisation

To meet the stringent resource constraints of smart home gateways, SecureFormer uses a minimal transformer encoder trained from scratch without linguistic pre-training, which significantly reduces the depth and width of the original BERT architecture, resulting in a compact model with fewer parameters, lower memory footprint, and reduced inference latency. While other studies, such as [14, 35], employ a multi-layer BERT-derived model, SecureFormer adopts a fundamentally different design philosophy. Rather than compressing a language model, SecureFormer removes language-modelling assumptions entirely, retaining only the self-attention mechanism required to model global dependencies among structured network features. Overall, SecureFormer retains the representational power of transformer-based models while addressing the practical deployment challenges associated with edge-based NIDS in smart home environments.

3.1 *Architecture overview*

The architecture of SecureFormer incorporates core principles of transformer models, such as self-attention mechanisms and positional embeddings. However, it is optimised to reduce computational demands while maintaining high performance. Here, we provide a detailed overview of its layers and design, compare it with other lightweight language models for IoT security, and justify its lightweight nature.

Figure 1 illustrates the architecture of SecureFormer, featuring an embedding layer comprising both token and positional embeddings. This is followed by a single transformer block, which consists of multi-head self-attention and a position-wise feed-forward network that captures inter-feature dependencies, leading into a pooler layer followed by a task-specific classification head. Table 1 presents the default parameters used in the architecture. The PyTorch implementation of the model is available on GitHub [39].

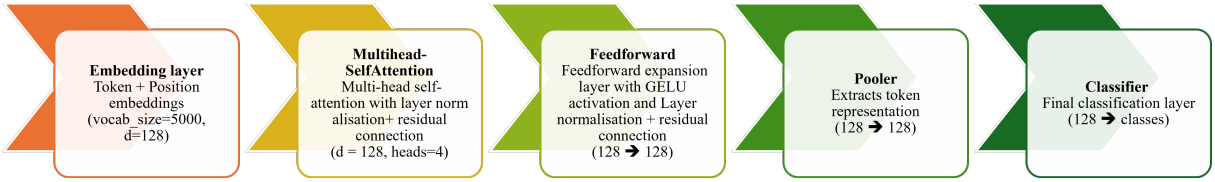


Figure 1. SecureFormer encoder block diagram

3.2 *Layer overview and design*

Table 1 presents the parameter values adopted in SecureFormer. In this subsection, we expand on each of the layers used in SecureFormer and how they each contribute to its efficiency and effectiveness:

Table 1. Architecture parameters in SecureFormer

Parameter	Value	Description
Vocabulary Size	5000	Size of token vocabulary for the BPE tokeniser and the model's token embedding.
Embedding Dimension	128	Dimension of token and position embeddings.
Max Position Embeddings	512	The maximum sequence length the model can process.
Number of Encoder Blocks	1	Number of transformer encoder layers (Attention + FFN).
Number of Attention Heads	4	Number of parallel attention heads in multi-head attention.
FF activation	GELU	Activation function.
Feed-Forward Hidden Dimension	512	Intermediate dimension in FFN (4 × embed_dim).
Attention Head Size	32	Calculated as embed_dim / num_attention_heads (128 / 4).
Dropout Probability	0.1	Dropout rate across layers.
Layer Normalisation Epsilon	1e-12	Numerical stability parameter for layer normalisation.

3.2.1 *Embedding layer*

The embedding layer includes token embeddings and positional embeddings, with a vocabulary size of 5000 and an embedding dimension of 128. This layer incorporates layer normalisation and dropout to improve generalisation and stability. Compared to traditional BERT models, which often use embedding dimensions of 768 or higher [40, 41], SecureFormer's smaller embedding dimension significantly reduces memory usage and computational cost. This choice is particularly important for IoT devices, where hardware limitations necessitate compact and efficient models.

3.2.2 *Multi-head-attention layer*

SecureFormer employs multi-head self-attention with four heads, striking a balance between expressive power and efficiency. The embedding dimension is divided among the heads, reducing the computational complexity of the attention mechanism. Each head processes a subset of the embedding dimension, enabling parallel computation while maintaining the ability to capture diverse feature representations. Dropout is applied to the attention probabilities to prevent overfitting, ensuring robust performance across diverse IoT security datasets. Only a single attention block (no stacked attention layers) was used. By limiting the number of heads and eliminating deep stacking, SecureFormer avoids the quadratic cost explosion typically associated with transformers—an essential requirement for smart home edge deployments [14]. The attention output is passed through a linear projection followed by application of layer normalisation and dropout. This design stabilises training while maintaining a low parameter count.

3.2.3 *Feed-Forward Network (FFN)—intermediate and output layers*

The intermediate layer expands the embedding dimension to 512 using a dense layer followed by a GELU activation. This expansion allows the model to capture nonlinear feature interactions before projecting the representation back to the embedding dimension (128). Layer normalisation and residual connections are used in the output layer to stabilise training and improve gradient flow, addressing potential issues with vanishing or exploding gradients in deep networks [42].

3.2.4 *Pooling layer*

SecureFormer utilises a BERT-style pooling layer that leverages the structural [CLS] anchor as a global flow descriptor. Through the self-attention mechanism, the model aggregates the full sequence context—including extracted feature values and [NULL] markers—into this single classification token. Rather than averaging all tokens, the pooler extracts the final hidden state of the [CLS] token and passes it through a Tanh-activated dense layer. This yields a compact, fixed-size vector optimised for the final classification head. Distilling the entire network flow into this single dense representation ensures the computational efficiency and low latency required for constrained IoT environments.

3.2.5 *Classification head*

The transformer model generates a sequence that must be converted into a classification through a classification head [41]. Consequently, the pooled representation is processed by a single fully connected layer designed for multi-class output, corresponding to the number of attack types present in the IoT security datasets. This classification head facilitates multi-class intrusion detection, enabling SecureFormer to distinguish benign traffic from various attack types. To enhance generalisation and minimise the risk of overfitting on the training data, dropout is applied before the classification layer.

3.3 *Dataset used*

Apejoye et al. [5] highlight several high-quality benchmark datasets suitable for pre-training and evaluating NIDS models in IoT environments, out of which the CICIoT2023 [20] and the Edge-IIoT dataset [21] stood out as one of the most diverse and realistic sets of datasets in recent times for IoT-focused NIDS training. The network flow from this dataset was used for validating SecureFormer.

CICIoT23 is a large-scale IoT intrusion-detection benchmark released by the Canadian Institute for Cybersecurity (University of New Brunswick). It was generated in a controlled IoT testbed comprising 105 real IoT devices, with 33 distinct attack scenarios executed and labelled to support supervised learning for NIDS/IDS research. The dataset organises attacks into seven high-level categories—Distributed Denial of Service (DDoS), Denial of Service (DoS), Reconnaissance, Web-based attacks, Brute Force, Spoofing, and Mirai—and provides traffic records with 46 extracted features suitable for machine-learning and deep-learning evaluation.

Edge-IIoTset is a cybersecurity dataset designed for IoT/IIoT intrusion detection, intended to support evaluation in both centralised and federated learning settings. It is built from a layered testbed architecture spanning multiple computing/networking layers (e.g., cloud/fog/edge and associated orchestration components), and includes both benign and malicious traffic representative of IIoT/IoT deployments. The dataset includes 14 attack types, grouped into five broad threat families: DoS/DDoS, information gathering, man-in-the-middle, injection, and malware. It is distributed with 61 extracted features alongside attack labels, enabling benchmarking of ML/DL-based IDS pipelines.

3.3.1 *Dataset-level feature optimisation*

The effectiveness of any machine learning–based system depends not only on the chosen algorithm but also critically on the quality of the training data. Public IoT intrusion detection benchmarks such as CICIoT23 and Edge-IIoT are characterised by (i) high-dimensional feature spaces with substantial redundancy and multicollinearity, and (ii) severe class imbalance, where a small number of traffic classes dominate the dataset while many security-critical attacks are sparsely represented. If left unmitigated, these properties can bias the learning process toward majority classes, inflate overall accuracy, and significantly degrade recall for rare but operationally important attacks, thereby limiting the reliability of

supervised NIDS models in real-world deployments [44]. They could also significantly increase computational cost, which is particularly problematic for lightweight, real-time deployments.

3.3.1.1 *Preliminary study—feature optimisation for environment-agnostic learning*

To tackle the challenges posed by high dimensionality and enhance generalisation across varied network environments, we performed a preliminary study of a dataset-level feature optimisation step to identify a compact and transferable feature subset before SecureFormer training. This stage is a standalone study, separate from SecureFormer training pipeline, and form the baseline study against which SecureFormer performance is further compared. The feature selection employed a domain-informed and statistically grounded approach using RF importance ranking, with robustness validated through cross-validation and performance stability analysis on both datasets.

To achieve this, environment-specific features such as IP addresses, port numbers, and timestamp-related attributes were eliminated, along with multicollinearity by removing features that exhibited a correlation coefficient above 0.9 with other features. Subsequently, the top 15 features were selected based on RF importance rankings, resulting in significant dimensionality reduction – 75% for Edge-IIoT and 67% for CICIoT2023 – while maintaining 97% and 85% of the cumulative feature importance, respectively. This reduced feature set also yielded improved macro-F1 scores of 0.93 (same as in full set) for Edge-IIoT and 0.77 (0.74 in full set) for CICIoT2023.

The 15-feature subset corresponds to the elbow point in both datasets, beyond which each additional feature contributes less than 0.5% incremental importance, indicating diminishing returns. Thus, the best-15 features provide an optimal balance between detection effectiveness, computational efficiency, and cross-dataset generalisability by capturing core, protocol-level attack characteristics invariant across IoT environments, effectively reducing redundancy without compromising performance. Table 2 presents the best 15 features identified in both datasets, and Section 4.1 reports the findings from this preliminary study.

Table 2. Best 15 features selected from both data using random forest feature importance

CICIoT23 RF15 Feature Set	Edge-IIoT RF15 Feature Set
IAT	tcp.options
Min	tcp.ack
fin_flag_number	tcp.seq
syn_count	icmp.checksum
Max	tcp.flags
Tot sum	tcp.ack_raw
Header_Length	tcp.len
Protocol Type	icmp.seq_le
syn_flag_number	tcp.payload
psh_flag_number	tcp.checksum
flow_duration	tcp.connection.rst
UDP	http.content_length
Rate	tcp.connection.syn
rst_count	arp.opcode
urg_count	mqtt.hdrflags

3.4 *SecureFormer training workflow*

The lightweight design of our model stems from a carefully considered architectural framework, as outlined in Table 3, along with optimised model training and a streamlined prediction pipeline. These decisions collectively reduce model size, training time, and inference latency, as shown in the results section. This section delves into the specifics of the training process, emphasising the technical aspects of our approach.

Table 3. Lightweight choices made in SecureFormer design

Component	Lightweight Strategy
Embedding	Low dimensionality (128)
Attention	Single layer, 4 heads
FFN	Single expansion block
Regularisation	Dropout at all major stages
Tokeniser	Small, domain-trained BPE
Parameters	No pre-trained weights

Figure 2 illustrates the high-level workflow for SecureFormer training and evaluation, describing the progression from network flow data ingestion to model training and final prediction. The workflow outlines a systematic methodology for processing network flow data, beginning with the partitioning of the dataset into train/validation and testing subsets to mitigate the risk of data snooping [43]. The test set is used strictly for final model evaluation, whereas the training/validation subset is used for tokeniser training and model development. CICIoT2023 was evaluated using its predefined dataset split provided my dataset author, while Edge-IIoTset was partitioned using a class-stratified split for training/validation and testing. This is followed by a preprocessing phase that encompasses class balancing through a hybrid resampling strategy, column-aware feature encoding, and tokenisation of the encoded feature text. The workflow culminates in training SecureFormer on the resulting network flow tokens and evaluating its effectiveness for attack detection using the held-out test set.

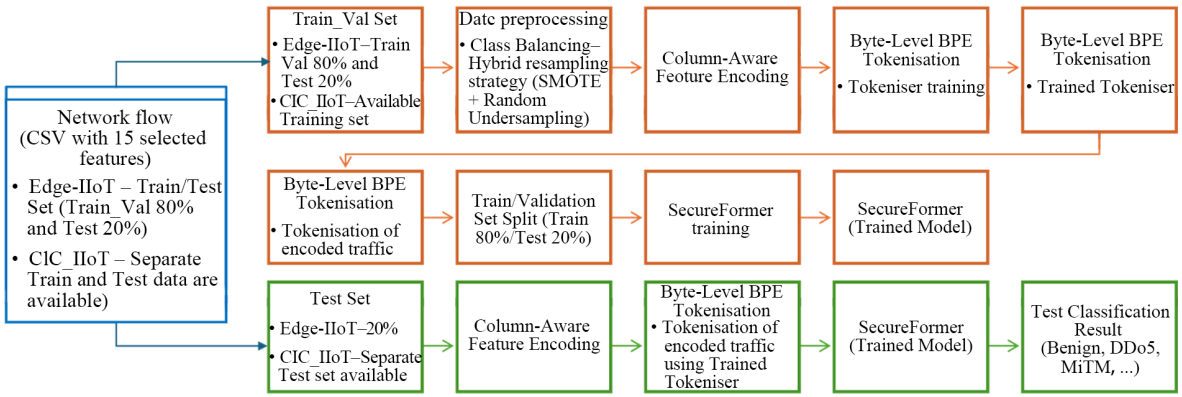


Figure 2. SecureFormer training workflow

The training process was carried out on a Dell laptop with the following features: an Intel 2 Core i5-9300H CPU running at 2.40 GHz, 32 GB of RAM, a 458 GB Solid-state drive, and a 4GB GPU (Nvidia GeForce GTX 1650).

3.4.1 *Data pre-processing*

3.4.1.1 *Class balancing approaches—SMOTE-based hybrid resampling*

Class imbalance was explicitly addressed prior to model training. This is because, network intrusion detection datasets such as CICIoT23 and Edge-IIoT exhibit substantial class imbalance, where a small number of traffic classes dominate the dataset while many security-critical attacks are severely under-represented. In CICIoT23, several attack classes contain fewer than 500 samples (such as Backdoor_Malware, PingSweep, and Uploading_Attack), whereas high-volume DDoS variants such as DDoS-ICMP_Flood, DDoS-UDP_Flood, and DDoS-TCP_Flood exceed hundreds of thousands of instances. Although Edge-IIoT is comparatively more balanced, notable disparities persist between frequent classes such as benign and DDoS traffic and low-frequency attacks such as fingerprinting and man-in-the-middle. To mitigate majority-class bias, a hybrid resampling strategy combining SMOTE-based oversampling and controlled undersampling was adopted.

We explicitly applied SMOTE (Synthetic Minority Oversampling Technique) and Random Under Sampler from the imbalanced-learn library [45] in Python, before SecureFormer training, to bring all the classes to the same number of samples each. SMOTE [46] addresses the minority class imbalance by generating synthetic samples through interpolation in the feature space rather than through simple duplication. For each minority class instance, SMOTE identifies its k-nearest neighbours and creates new synthetic samples by linear interpolation [47].

$$x_{syn} = x_i + \lambda \cdot (x_{nn} - x_i)\,.$$

Where:

- X_i: denotes a minority-class sample
- X_{nn}: represents one of its k-nearest neighbours
- λ: is a random number in the range (0–1)
- $(x_{nn} - x_i)$ is the difference vector, which defines the direction and distance between the two points.

Following SMOTE, the majority classes were undersampled to achieve uniform class representation. This ensures that the minority classes benefit from synthetic augmentation and the majority classes retain their original samples up to the target threshold.

3.4.1.2 *Column-aware feature encoding*

Unlike NLP tasks where semantics arise from word order, network traffic consists of structured features (e.g., protocol, packet size, flags), therefore, instead of operating on natural language tokens, SecureFormer adopts a feature-aware tokenisation strategy, where structural network traffic data are transformed into pseudo-statements that preserve temporal ordering and protocol semantics while enabling direct compatibility with transformer-based sequence models.

To achieve this, SecureFormer utilises a custom vocabulary of structural tokens—namely [CLS], [VAL], [NULL], and </s>—to guide the model's contextual understanding. The pseudo-sentence is anchored by a global classification marker ([CLS]), followed by sequential feature names explicitly paired with their corresponding values via a value delimiter ([VAL]). Furthermore, to prevent downstream attention mechanisms from learning spurious correlations from empty or protocol-incompatible fields, missing or inapplicable data is explicitly masked using the [NULL] identifier. The resulting serialised network flow takes the following structural format before entering the tokenisation pipeline:

$$[CLS]\ <feature_1>\ [VAL]\ <value><feature_2>\ [VAL]\ [NULL]\ \ldots\ <feature_n>\ [VAL]\ <value></s>$$

The text-to-sequence workflow is summarised in Algorithm 1.

Algorithm 1 Column-Aware Feature Encoding for SecureFormer

```
Input: A network flow instance x with ordered feature set F = {f1, f2, ..., fn}
Output: A serialised pseudo-text sequence S
Initialise S ← [CLS]
for each feature fi in F do
    Append feature name fi to S
    Append [VAL] to S
    if value(fi) is missing or not applicable then then
        Append [NULL] to S
    else
        Append value(fi) to S
    end if
end for
Append </s> to S
Return S
```

3.4.1.3 *Byte-level BPE tokenisation*

SecureFormer employs a byte-level Byte Pair Encoding (BPE) tokeniser, trained from scratch on the encoded network features derived from the column-aware feature transformation of the training set. This approach enables the construction of a compact, domain-specific vocabulary that effectively manages both numerical and categorical values [13]. Notably, this design mitigates Out-of-Vocabulary (OoV) challenges in network traffic data, which do include arbitrary binary patterns absent from conventional natural language training corpora. By avoiding general-purpose pre-trained NLP tokenisers—which often have large vocabularies [48] – SecureFormer ensures domain specificity and maintains a compact vocabulary size. This design choice optimises the model for network protocol semantics while significantly reducing the memory footprint required for edge deployment. After the training, the tokeniser is used to tokenise the entire training set. Its outputs, input_ids and attention_mask, are then fed into the model for training. The input_ids is a sequence of integers (token ID) that maps directly to a token (which could be a word, subword, or character) in the vocabulary that was trained exclusively on this network traffic dataset using the tokeniser. Table 4 presents the tokenisation configuration for the byte-level BPE tokeniser used in this study.

Table 4. Tokenisation configuration for byte-level BPE tokeniser

Setting	Value	Design intent
Tokenizer type	Byte-level BPE	robust to mixed alphanumeric tokens and numeric strings
Vocabulary size	5000	trade-off between coverage and compact embedding table
Min frequency	2	filters rare/noisy substrings
Special tokens	<s>, <pad>, </s>, <unk>, <mask>,	BERT-style training/inference compatibility
Additional special tokens	[CLS], [VAL], [NULL]	Structural feature formatting and sequence-level flow classification
Max sequence length	≤ 512 (dynamically set based on the maximum possible datapoint in the encoded data, but capped at 512 to avoid excessively long sequences)	covers worst-case flows while allowing truncation

3.4.2 *Model training*

The SecureFormer training workflow follows a standard supervised learning pipeline with learning rate scheduling, batch-wise training using the Adam optimiser, Validation with accuracy and F1-score, and Early stopping based on the validation F1-score. Training uses cross-entropy loss, appropriate for multi-class classification tasks. To mitigate gradient explosion and improve training stability, gradient clipping was applied, constraining the global gradient norm to a threshold of 1.0 [49]. This technique is particularly vital for ensuring convergence when training deep neural architectures over long horizons, as it prevents large weight updates from destabilising the learning process [50]

A Reduce-on-Plateau learning rate scheduler was employed to adaptively adjust the learning rate based on validation performance. Specifically, the learning rate was reduced by a factor of 0.5 if the validation loss failed to improve over two consecutive epochs. This strategy encourages finer-grained optimisation once the model approaches convergence and helps avoid oscillation around suboptimal minima. Table 5 presents the Training parameters used in this study. The overall training employed an early stopping mechanism with a patience of 5 epochs to prevent overfitting and optimise training time. Although the maximum training limit was set to 100 epochs, the models achieved rapid convergence. On the Edge-IIoT dataset, the model reached within 1% of its peak F1-score by the 10th epoch and eventually triggered early stopping at epoch 53 as performance plateaued. For the CICIOT23 dataset, the model demonstrated consistent incremental gains, maintaining a steady reduction in loss and achieving a high validation accuracy of 89.21% by the 50th epoch. These observations suggest that the architecture is highly efficient at capturing relevant intrusion patterns early in the training process. The training pipeline used in this study is available on GitHub [39].

Table 5. Training parameters used

Parameter	Value	Description
	Data Configuration	
Batch Size	32	Samples per batch during training/validation
Max Token Length	512	Maximum sequence length (capped computed maximum)
Train/Val Split	80/20	Proportion of data for training vs validation
	Model Architecture	
Number of Classes	15 or 34	Attack types to classify
Embedding Dimension	128	Token embedding dimension
Number of Attention Heads	4	Multi-head attention configuration
Intermediate Dimension	512	Feedforward intermediate layer size
Tokeniser Vocab Size	5000	ByteLevelBPE tokeniser vocabulary
	Optimizer & Loss	
Optimizer	Adam	With learning rate 1e-4
Learning Rate	1e-4	Initial learning rate
Loss Function	CrossEntropyLoss	For multi-class classification
	Training Configuration	
Epochs	50	Maximum number of training epochs
Early Stopping Patience	5	Epochs without improvement before stopping
LR Scheduler	ReduceLROnPlateau	Reduce LR by 0.5 after 2 epochs without improvement
Gradient Clipping	max_norm=1.0	Prevent exploding gradients
Evaluation Metric	F1-Score (Weighted)	Used for early stopping and model selection

4. Results and discussion

This section presents the evaluation result from a baseline model, Random Forest, which was used for feature selection and baseline study. The discussion then provides a comprehensive analysis of SecureFormer's performance across two heterogeneous NIDS benchmark datasets: the large-scale CICIoT2023 and the Industrial-focused Edge-IIoTset. This discussion evaluates the results through three primary lenses: classification efficacy, computational efficiency, and edge-deployability.

4.1 *Baseline study/feature selection using RF*

As reported in Table 6, the experimental evaluation of the feature set selection demonstrates that substantial dimensionality reduction can be achieved without compromising classification performance. The feature set selection analysis reveals that the Best 15 feature subset on the CICIoT2023 dataset matches the full 46-feature model's accuracy (0.99) while improving macro precision (0.81 to 0.90) and macro F1 score (0.74 to 0.77), indicating enhanced minority-class discrimination and reduced class imbalance. For the EdgeIIoT dataset, while the full 61-feature model achieves perfect accuracy (1.00), the best15 subset maintains robust performance (accuracy = 0.96, macro F1 = 0.93) with a 75% reduction in dimensionality. Thus, the 15-feature subset offers an effective trade-off between performance and efficiency for real-world IoT intrusion detection applications.

Table 6. Impact of Feature Subset Selection on Model Performance across CICIoT2023 and EdgeIIoT Datasets

Dataset	Feature Configuration	Features	Acc	Macro avg (P/R/F1)	Weighted avg (P/R/F1)	Performance Profile
CICIoT2023	All Features	46	0.99	**0.81/0.72/0.74**	0.99/0.99/0.99	High accuracy, weak minority class detection
	Best-15 Features	15	0.99	**0.90/0.74/0.77**	0.99/0.99/0.99	11.1% improvement in macro precision
EdgeIIoT	All Features	61	1.00	**1.0 /1.00 /1.00**	1.0 /1.00 /1.00	Perfect performance - likely overfitting on dataset-specific patterns
	Best-15 Features	15	0.96	**0.97/0.91/0.93**	0.97/0.96/0.96	Removal of timing/flow IDs maintains strong erformance

4.2 *SecureFormer performance report*

Tables 7–10 presents the classification report and model performance analysis of SecureFormer on the two datasets evaluated in this study.

Table 7. SecureFormer model performance on the CICIoT2023 dataset

Metric Category	Metric	Measured Value	Analysis
		Model Size & Parameters	
Total Parameters	Parameter Count	**941,474**	Parameters from loaded state dict
Model Size (Disk)	File Size	**3.6 MB**	
		Computational Efficiency	
FLOPs per Inference	Computation	**0.09 GFLOPs**	Extremely lightweight; <0.1% of BERT-base [51]

Table 7. (cont.)

Metric Category	Metric	Measured Value	Analysis
Inference Latency (Measured on CPU)			
Full Pipeline Latency	Preprocessing + Inference	**15.71 ms**	Total end-to-end latency per sample
Inference-Only Latency	Model Forward Pass	**6.87 ms**	Pure model computation time
Latency Contribution - Preprocessing	Percentage	**56.33%**	Significant bottleneck in the pipeline
Latency Contribution - Inference	Percentage	**43.67%**	Primary contributor to total latency
Throughput (Estimated)			
Single Sample Throughput	Samples/sec	**~ 145 samples/sec**	1000 samples ÷ 6.87 ms

1GFLOPS = 1 billion floating-point operations.

Table 8. Classification report of SecureFormer on CIC-IoT 23 dataset

Attack Class	Precision	Recall	F1-Score	Support
DDoS-ACK_Fragmentation	0.98	1.00	0.99	3056
DDoS-SYN_Flood	1.00	1.00	1.00	43,356
DDoS-PSHACK_Flood	1.00	1.00	1.00	44,035
Mirai-greeth_flood	1.00	1.00	1.00	10,623
DoS-SYN_Flood	1.00	1.00	1.00	21,637
DDoS-ICMP_Flood	1.00	1.00	1.00	76,935
DDoS-TCP_Flood	1.00	1.00	1.00	48,207
DDoS-UDP_Flood	1.00	1.00	1.00	58,141
Mirai-greip_flood	1.00	1.00	1.00	8181
BenignTraffic	0.93	0.85	0.89	11,772
Mirai-udpplain	1.00	1.00	1.00	9583
DoS-TCP_Flood	1.00	1.00	1.00	28,498
DDoS-SynonymousIP_Flood	1.00	1.00	1.00	38,400
DoS-UDP_Flood	1.00	1.00	1.00	35,589
DDoS-RSTFINFlood	1.00	1.00	1.00	43,252
Recon-HostDiscovery	0.65	0.89	0.75	1460
DDoS-UDP_Fragmentation	0.99	1.00	0.99	3065
DDoS-ICMP_Fragmentation	0.99	1.00	0.99	4861
DDoS-HTTP_Flood	0.95	1.00	0.98	293
MITM-ArpSpoofing	0.84	0.79	0.81	3257
Recon-PortScan	0.70	0.65	0.67	885
DNS_Spoofing	0.57	0.76	0.65	1956
DoS-HTTP_Flood	0.99	1.00	0.99	773
DDoS-SlowLoris	0.99	0.99	0.99	267
DictionaryBruteForce	0.83	0.50	0.62	136
VulnerabilityScan	0.96	1.00	0.98	412
Recon-OSScan	0.49	0.62	0.55	1074
Recon-PingSweep	0.61	0.61	0.61	18
XSS	0.89	0.41	0.57	41

Table 8. (cont.)

Attack Class	Precision	Recall	F1-Score	Support
CommandInjection	0.90	0.62	0.73	60
Backdoor_Malware	1.00	0.60	0.75	43
SqlInjection	1.00	0.42	0.60	59
BrowserHijacking	0.87	0.58	0.69	57
Uploading_Attack	0.83	0.28	0.42	18
Accuracy			**0.99**	**500,000**
Macro Avg	**0.91**	**0.84**	**0.86**	**500,000**
Weighted Avg	**0.99**	**0.99**	**0.99**	**500,000**

Table 9. SecureFormer model performance on the Edge_IIoT dataset

Metric Category	Metric	Measured Value	Analysis
Model Size & Parameters			
Total Parameters	Parameter Count	**939,023**	Parameters from loaded state dict
Model Size (Disk)	File Size	**3.59 MB**	
Computational Efficiency			
FLOPs per Inference	Computation	**0.11 GFLOPs**	Extremely lightweight; <0.1% of BERT-base [51]
Inference Latency (Measured on CPU)			
Full Pipeline Latency	Preprocessing + Inference	**24.42 ms**	Total end-to-end latency per sample
Inference-Only Latency	Model Forward Pass	**10.25 ms**	Pure model computation time
Latency Contribution - Preprocessing	Percentage	**58.03%**	Significant bottleneck in the pipeline
Latency Contribution - Inference	Percentage	**41.97%**	Primary contributor to total latency
Throughput (Estimated)			
Single Sample Throughput	Samples/sec	**~ 98 samples/sec**	1000 samples ÷ 10.25 ms

Table 10. Classification report of SecureFormer on Edge_IIoT dataset

Attack Class	Precision	Recall	F1-Score	Support
Uploading	1.00	0.86	0.92	2054
DDoS_HTTP	1.00	1.00	1.00	2112
Password	1.00	0.99	1.00	1998
SQL_injection	1.00	1.00	1.00	2062
Backdoor	1.00	0.93	0.96	2039
Ransomware	1.00	0.84	0.91	2185
XSS	0.99	0.95	0.97	2011
Normal	1.00	0.99	1.00	4860
Vulnerability_scanner	1.00	1.00	1.00	2015
DDoS_ICMP	1.00	1.00	1.00	2818
Port_Scanning	0.75	0.96	0.84	2014

Table 10. (cont.)

Attack Class	Precision	Recall	F1-Score	Support
DDoS_TCP	1.00	1.00	1.00	2050
DDoS_UDP	0.88	1.00	0.93	2900
Fingerprinting	1.00	0.70	0.83	200
MITM	1.00	1.00	1.00	80
Accuracy			**0.97**	**31,398**
Macro Avg	**0.97**	**0.95**	**0.96**	**31,398**
Weighted Avg	**0.97**	**0.97**	**0.97**	**31,398**

4.2.1 *Analysis of classification performance*

SecureFormer demonstrates exceptional classification robustness, achieving a global accuracy of 99% on CICIoT2023 and 98% on Edge-IIoTset. On the CICIoT2023 dataset (Table 7), the model achieves a near-perfect F1 Score (1.00) for high-volume volumetric attacks such as DDoS-SYN, DDoS-ICMP, and Mirai-based floods. This indicates that the Transformer's self-attention mechanism effectively captures the predictable statistical patterns inherent in large-scale flow-based attacks. In contrast, the performance on the Edge-IIoTset (Table 9) is remarkably balanced, with a Macro Average F1-score of 0.97. The model successfully identifies specialised industrial threats, including SQL Injection (0.99 F1) and Ransomware (0.93 F1). Notably, even with smaller support sets, classes like MITM (1.0 F1) and Backdoor (0.96 F1) are detected with high precision, suggesting that the SecureFormer architecture is highly effective at learning the behavioural signatures of Industrial IoT (IIoT) protocols.

A closer inspection of the class-wise results highlights several important failure modes. On CICIoT2023, the weakest performance is concentrated in rare and subtle attack classes such as Uploading_Attack (0.40 F1), XSS (0.53 F1), SqlInjection (0.57 F1), Recon-OSScan (0.58 F1), and Recon-PortScan (0.61 F1). These classes are characterised by very low support and behavioural patterns that overlap substantially with benign or closely related traffic, making them more difficult to distinguish. Similar behaviour is observed for DictionaryBruteForce and MITM-ArpSpoofing, where recall remains comparatively limited despite strong global accuracy. On Edge-IIoTset, the failure modes are less severe but still visible. In particular, Fingerprinting exhibits lower recall (0.74) than the other classes, despite high precision, suggesting that the model captures the attack signature conservatively but does not generalise equally across all instances. These results indicate that, while SecureFormer is highly effective on dominant attack classes, rare, low-support, or semantically subtle attacks remain challenging and represent an important direction for future improvement.

The differing performance profiles on CICIoT2023 and Edge-IIoTset are partly explained by the distinct structure of the two datasets and the nature of their selected features. As shown in Table 2, the CICIoT2023 feature subset is primarily flow-based, whereas the Edge-IIoTset subset is dominated by packet-level descriptors. In addition, CICIoT2023 exhibits more severe class imbalance and includes several subtle, low-support attack classes, which depresses macro-level performance despite very high global accuracy. Taken together, these results suggest that the observed variation is not solely a property of the model, but also reflects differences in traffic representation, feature semantics, and attack distribution.

4.2.2 *Computational efficiency and structural analysis*

A core contribution of SecureFormer is its extreme structural efficiency. The model maintains a compact parameter count of approximately 941k, resulting in a storage footprint of only 3.6 MB. This is significantly lower than standard Transformer models (e.g., BERT-base), making it ideal for the limited non-volatile memory of edge gateways.

The computational demand, measured at 0.09 to 0.11 GFLOPs per inference, is less than 0.1% of that required by BERT-base. This confirms that SecureFormer successfully distills the predictive power of the Transformer architecture into

a lightweight framework without the prohibitive overhead of traditional Large Language Models (LLMs). This efficiency is vital for maintaining a low-energy profile in battery-powered IoT nodes.

4.2.3 *Edge-deployability and latency benchmarking*

The end-to-end latency analysis (Tables 7 and 9) provides critical insights into real-time deployment feasibility. SecureFormer achieves an inference-only latency of 6.87 ms on the CICIoT2023 dataset and 10.52 ms on the Edge-IIoTset using standard CPU processing. These sub-15 ms values satisfy the requirements for near-real-time intrusion detection in high-speed network environments.

However, the analysis highlights a significant bottleneck: preprocessing accounts for 56%–58% of the total pipeline latency. While the model forward pass is exceptionally fast, the transformation of raw network telemetry into semantic "input_ids" (tokenisation) introduces a delay of approximately 9ms to 14ms per sample. Despite this, the system maintains a throughput of 145 samples per second for CICIoT2023. In an edge deployment scenario, this throughput is sufficient for monitoring local IoT subnetworks where traffic is bursty but aggregated flow rates remain within the processing threshold of modern edge processors like the Raspberry Pi 4 or NVIDIA Jetson Nano. However, direct benchmarking on such hardware is left for future work.

4.2.4 *Synthesis and comparative advantage*

Compared to the baseline RF results in this work, SecureFormer addresses the critical "minority class" detection gap. While the RF baseline achieved high accuracy, it struggled with the semantic complexity of the 34-class CICIoT2023 task. SecureFormer's superior Macro average metrics demonstrate that the attention mechanism provides a better contextual understanding of flow sequences than the decision-boundary logic of classical ensemble learners. Furthermore, by outperforming the latency of the DL-BiLSTM [16] and the size of SecurityBERT [14], SecureFormer emerges as a highly optimised solution that bridges the gap between the high-performance requirements of modern NIDS and the severe resource limitations of the IoT edge.

4.3 *Comparison with state-of-the-art studies in IoT security*

Table 11 presents a comparative summary of SecureFormer against representative machine learning, deep learning, and Transformer-based IDS models evaluated on CICIoT2023 and/or Edge-IIoTset. NR indicate values not explicitly reported by the corresponding authors, and NA is used when the metric is not applicable.

From the table, we can see that while classical machine learning baselines, such as the Random Forest implementations in this work and the studies by [21] and [20] demonstrate strong global accuracy—often exceeding 99% on the CICIoT2023 dataset—they consistently exhibit high variance and poor performance on minority class detection, as evidenced by significantly lower macro-average metrics and/or F1 score. In contrast, SecureFormer achieves a more robust equilibrium, maintaining a 99% accuracy on CICIoT2023 while delivering a superior macro F1-score of 0.86, indicating its enhanced ability to detect rare but sophisticated adversarial vectors that traditional learners overlook.

More advanced deep learning architectures, such as the CNN-LSTM [52], demonstrate strong detection capability on Edge-IIoTset, achieving 97.14% accuracy. However, their reliance on hybrid sequential architectures and omission of deployability metrics leave uncertainty regarding real-time execution on edge devices. Moreover, compared to these deep learning hybrids, CNN-LSTM [52] and DL-BiLSTM [16], SecureFormer demonstrates superior computational efficiency. Specifically, while the DL-BiLSTM model requires approximately 6400 ms for inference on the CICIoT2023 dataset, SecureFormer reduces this latency to a mere 6.87 ms, representing a several-fold improvement in real-time responsiveness. Though hardware-specific differences may play a part in this. Although KronNet [38] offers sub-millisecond inference, it relies on a Kronecker-enhanced feed-forward structure that lacks the self-attention mechanism fundamental to SecureFormer, which is vital for capturing the long-range temporal dependencies and semantic relationships inherent in complex network flow sequences.

Table 11. Comparative Evaluation with State-of-the-Art models

Study (Year)	Model Category	Dataset	Task	Accuracy (%)	Weighted avg (P/R/F1)	Macro avg (P/R/F1)	Edge Deployability (Inference latency on CPU / model size / #params)	Remarks
[21]	ML models	Edge-IIoTset	Multi-class	80.83		NR	NR	Dataset baseline results – RF is best among ML
[21]	DNN	Edge-IIoTset	Multi-class	94.67		NR	NR	Dataset baseline results for DL
[53]	Fine-tuned LLM (DeepSeek-R1-Distil-Llama-8 B)	Edge-IIoTset	15 classes	73.2	0.75/0.73/0.71	NR	NR/8B	Compares LLM fine-tuning against zero-shot/few-shot for anomaly mitigation and mitigation recommendation
[52]	CNN-LSTM	Edge-IIoTset	Multi-class	97.14	0.82/0.72/0.74	—	—	Excludes application-layer features to promote privacy and generalisability across other environments.
[14]	SecurityBERT (BERT-like Transformer)	Edge-IIoTset	Multi-class	98.2	0.98/0.98/0.98	0.87/0.84/0.84	158 ms/16.7 MB/11M	PPFLE encoding + tokenisation + BERT-like pre-training with more focus on improving accuracy.
[20]	Benchmark ML/DL	CICIoT2023	34-class	99.1	0.7/0.83/0.71	NR	NR	Dataset baseline results – RF is best among ML/DNN
[16]	DL-BiLSTM	CICIoT2023	8-classes	93.13	0.918/0.931/0.919	NR	6400 ms/0.036 MB/2K	Uses IPCA and dynamic quantisation to achieve a lightweight status
[38]	KronNet (Kronecker-enhanced FFNN)	CICIoT2023	34-class	99.01	0.99 /0.99/0.99	NR	0.2 ms/0.019 MB/5K	Sub-millisecond inference; evaluated for edge and microcontroller feasibility.
Random Forest (this work)	RF	Edge-IIoTset	15-classes	96	0.97/0.96/0.96	0.97/0.91/0.93	NA	This study baseline model. High accuracy, weak minority class detection
Random Forest (this work)	RF	CICIoT2023	34-classes	99	0.99/0.99/0.99	0.90/0.74/0.77	NA	This study's baseline model
SecureFormer (this work)	Transformer	Edge-IIoTset	15-classes	97	0.97/0.97/0.97	0.97/0.95/0.96	10.25 ms/3.50 MB/939k	This study's model designed for edge feasibility
SecureFormer (this work)	Transformer	CICIoT2023	34-classes	99	0.99/0.99/0.99	0.91/0.84/0.86	6.87 ms/3.6 MB/941k	This study's model designed for edge feasibility

NR indicate values not explicitly reported by the corresponding authors, and NA is used when the metric is not applicable.

Furthermore, Transformer-based approaches represent the current performance frontier in NIDS research, with studies such as SecurityBERT [14] achieving high accuracy and weighted performance on Edge-IIoTset through privacy-preserving encoding and BERT-style pretraining. However, compared to SecureFormer, SecurityBERT comes at the cost of a slightly larger model size (11 million parameters and 16 MB) and higher inference latency (158 ms on CPU). SecureFormer achieves comparable or superior detection rates—specifically 0.94 macro F1-score on the Edge-IIoTset—with a parameter count under 1 million (939k–941k) and a model size of approximately 3.6 MB. This represents a nearly 15-fold reduction in inference time compared to SecurityBERT, facilitating seamless deployment on resource-constrained edge gateways.

Tejero-Fernández and Sánchez-Macián [53], demonstrate the feasibility of applying LLMs to IoT security tasks but incur substantial computational overhead. Their fine-tuned DeepSeek-R1-Distil-Llama-8B achieved only 73.2% accuracy on the Edge-IIoT set despite its 8-billion-parameter scale. This discrepancy highlights that while massive models excel in general reasoning, they often lack the domain-specific efficiency and precision required for high-speed network telemetry.

Against this landscape, SecureFormer demonstrates a balanced and deployment-oriented design. On Edge-IIoTset, SecureFormer delivers competitive multi-class performance with strong weighted and macro-averaged metrics while maintaining a sub-4 MB footprint and under-11 ms CPU inference latency. On CICIoT2023, SecureFormer achieves 99% accuracy across 34 classes with fewer than one million parameters, a result that matches or exceeds larger deep learning and Transformer-based models while remaining edge-deployable. Compared to the Random Forest baselines from this study, SecureFormer exhibits more stable macro-averaged performance, indicating improved robustness to class imbalance. Overall, these results confirm that SecureFormer effectively bridges the gap between detection performance and deployability, offering a practical Transformer-based solution tailored to real-world edge IoT intrusion detection.

5. Conclusion

In this paper, we introduced SecureFormer, a lightweight, edge-optimised transformer architecture designed to address the critical challenges of network intrusion detection in modern IoT and IIoT environments. By leveraging a streamlined self-attention mechanism, we demonstrated that the predictive power of Transformer models—typically associated with high computational overhead—can be effectively distilled into a compact framework suitable for resource-constrained edge devices.

Experimental evaluation on CICIoT2023 and Edge-IIoTset confirms that SecureFormer achieves state-of-the-art classification performance. With a global accuracy of 99% on the 34-class CICIoT2023 task and 98% on the Edge-IIoTset, the model significantly outperforms classical machine learning baselines and traditional deep learning hybrids in detecting complex, multi-stage attack vectors. Crucially, the model maintains a remarkably small footprint, fewer than 1 million parameters and approximately 3.6 MB in size, and an inference-only latency of 6.87 ms, representing a nearly 15-fold improvement in efficiency over existing BERT-based security models. This balance of high-fidelity detection and low computational cost (0.09 GFLOPs) underscores its viability for real-time deployment on edge gateways.

Beyond performance, this study provides insight into the relationship between model complexity, feature dimensionality, and detection capability. The findings show that compact attention mechanisms can retain strong representational power even when trained on a reduced feature set, supporting the feasibility of deploying Transformer-based intrusion detection systems at the network edge.

However, the study also highlights several practical considerations, including preprocessing overhead and sensitivity to feature representation, which must be addressed for real-world deployment. Future work will focus on further model optimisation through compression techniques such as quantisation and pruning, as well as extending the system to support end-to-end processing from raw network traffic to prediction. This will enable a more comprehensive evaluation of SecureFormer in real-world smart home environments.

Code availability

The code related to this project is on GitHub: https://github.com/asegun-cod/SecureFormer.

Conflicts of interest

The authors declare no competing financial interest.

References

[1] U. Ahmed, M. Nazir, A. Sarwar, T. Ali, E. M. Aggoune, T. Shahzad, et al., "Signature-based intrusion detection using machine learning and deep learning approaches empowered with fuzzy clustering," *Scientific Reports*, vol. 15, p. 1726, 2025. https://doi.org/10.1038/s41598-025-85866-7.

[2] P. Malhotra, Y. Singh, P. Anand, D. K. Bangotra, P. K. Singh, and W. Hong, "Internet of things: evolution, concerns and security challenges," *Sensors*, vol. 21, no. 5, 2021. https://doi.org/10.3390/s21051809.

[3] J. Ferdous, R. Islam, A. Mahboubi, and M. Z. Islam, "A survey on ML techniques for multi-platform malware detection: securing PC, mobile devices, IoT, and cloud environments," *Sensors*, vol. 25, no. 4, 2025. https://doi.org/10.3390/s25041153.

[4] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41-50, 2018. https://doi.org/10.1109/TETCI.2017.2772792.

[5] O. Apejoye, N. Ajienka, J. He, and X. Ma, "Critical review of network intrusion detection benchmark datasets for practical IoT security," *Computer Networks and Communications*, vol. 3, no. 2, pp. 182-208, 2025. https://doi.org/10.37256/cnc.3220257228.

[6] M. A. Ferrag, F. Alwahedi, A. Battah, B. Cherif, A. Mechri, N. Tihanyi, et al., "Generative AI in cybersecurity: a comprehensive review of LLM applications and vulnerabilities," *Internet of Things and Cyber-Physical Systems*, vol. 5, pp. 1-46, 2025. https://doi.org/10.1016/j.iotcps.2025.01.001.

[7] B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in internet of things," *Journal of Network and Computer Applications*, vol. 84, pp. 25-37, 2017. https://doi.org/10.1016/j.jnca.2017.02.009.

[8] P. M. S. Sánchez, J. M. J. Valero, A. H. Celdrán, G. Bovet, M. G. Pérez, and G. M. Pérez, "A survey on device behavior fingerprinting: data sources, techniques, application scenarios, and datasets," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 1148-1179, 2021. https://doi.org/10.1109/COMST.2021.3064259.

[9] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525-41550, 2019. https://doi.org/10.1109/ACCESS.2019.2895334.

[10] L. D. Manocchio, S. Layeghy, W. W. Lo, G. K. Kulatilleke, M. Sarhan, and M. Portmann, "FlowTransformer: a transformer framework for flow-based network intrusion detection systems," *Expert Systems with Applications*, vol. 241, 2024. https://doi.org/10.1016/j.eswa.2023.122564.

[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, et al., "Attention is all you need," In Proc. 31st Conference on Neural Information Processing Systems, Long Beach, CA, USA, Dec. 1-9, 2017.

[12] Z. Wu, H. Zhang, P. Wang, and Z. Sun, "RTIDS: a robust transformer-based approach for intrusion detection system," *IEEE Access*, vol. 10, pp. 64375-64387, 2022. https://doi.org/10.1109/ACCESS.2022.3182333.

[13] X. Lin, G. Xiong, G. Gou, Z. Li, J. Shi, and J. Yu, "ET-BERT: a contextualized datagram representation with pre-training transformers for encrypted traffic classification," In Proc. WWW '22: The ACM Web Conference, Virtual, Apr. 25-29, 2022, pp. 633-642. https://doi.org/10.1145/3485447.3512217.

[14] M. A. Ferrag, M. Ndhlovu, N. Tihanyi, L. C. Cordeiro, M. Debbah, T. Lestable, et al., "Revolutionizing cyber threat detection with large language models: a privacy-preserving BERT-based lightweight model for IoT/IIoT devices," *IEEE Access*, vol. 12, pp. 23733-23750, 2024. https://doi.org/10.1109/ACCESS.2024.3363469.

[15] T. Suwannaphong, F. Jovan, I. Craddock, and R. McConville, "Optimising TinyML with quantization and distillation of transformer and mamba models for indoor localisation on edge devices," *Scientific Reports*, vol. 15, 2025. https://doi.org/10.1038/s41598-025-94205-9.

[16] Z. Wang, H. Chen, S. Yang, X. Luo, D. Li, and J. Wang, "A lightweight intrusion detection method for IoT based on deep learning and dynamic quantization," *PeerJ Computer Science*, vol. 9, p. e1569, 2023. https://doi.org/10.7717/peerj-cs.1569.

[17] V. Hnamte and J. Hussain, "DCNNBiLSTM: an efficient hybrid deep learning-based intrusion detection system," *Telematics and Informatics Reports*, vol. 10, 2023. https://doi.org/10.1016/j.teler.2023.100053.

[18] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153-1176, 2015. https://doi.org/10.1109/COMST.2015.2494502.

[19] M. Sarhan, S. Layeghy, and M. Portmann, "Towards a standard feature set for network intrusion detection system datasets," *Mobile Networks and Applications*, vol. 27, no. 1, pp. 357-370, 2022. https://doi.org/10.1007/s11036-021-01843-0.

[20] E. C. P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu and A. A. Ghorbani, "CICIoT2023: a real-time dataset and benchmark for large-scale attacks in IoT environment," *Sensors*, vol. 23, no. 13, p. 5941, 2023. https://doi.org/10.3390/s23135941.

[21] M. A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, and H. Janicke, "Edge-IIoTset: a new comprehensive realistic cyber security dataset of IoT and IIoT applications for centralized and federated learning," *IEEE Access*, vol. 10, pp. 40281-40306, 2022. https://doi.org/10.1109/ACCESS.2022.3165809.

[22] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki, "Network intrusion detection for IoT security based on learning techniques," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2671-2701, 2019. https://doi.org/10.1109/COMST.2019.2896380.

[23] A. Khraisat and A. Alazab, "A critical review of intrusion detection systems in the internet of things: techniques, deployment strategy, validation strategy, attacks, public datasets and challenges," *Cybersecurity*, vol. 4, pp. 1-27, 2021. https://doi.org/10.1186/s42400-021-00077-7.

[24] G. R. Kumar, A. D. Kulkarni, B. S. Kumar, N. Singh, V. Revathi and T. C. A. Kumar, "Machine learning approaches for anomaly detection in IoT networks," In Proc. 2024 International Conference on Advances in Computing, Communication and Applied Informatics, Chennai, India, May 9-10, 2024, pp. 1-5. https://doi.org/10.1109/ACCAI61061.2024.10601954.

[25] M. A. Ali and S. A. H. Al-Sharafi, "Intrusion detection in IoT networks using machine learning and deep learning approaches for MitM attack mitigation," *Discover Internet of Things*, vol. 5, 2025. https://doi.org/10.1007/s43926-025-00104-w.

[26] A. H. Ali, M. Charfeddine, B. Ammar, B. B. Hamed, F. Albalwy, A. Alqarafi, et al., "Unveiling machine learning strategies and considerations in intrusion detection systems: a comprehensive survey," *Frontiers in Computer Science*, vol. 6, 2024. https://doi.org/10.3389/fcomp.2024.1387354.

[27] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: approaches, datasets, and comparative study," *Journal of Information Security and Applications*, vol. 50, 2020. https://doi.org/10.1016/j.jisa.2019.102419.

[28] A. M. Alashjaee, "Deep learning for network security: an Attention-CNN-LSTM model for accurate intrusion detection," *Scientific Reports*, vol. 15, 2025. https://doi.org/10.1038/s41598-025-07706-y.

[29] L. Mohammadpour, T. C. Ling, C. S. Liew, and A. Aryanfar, "A survey of CNN-based network intrusion detection," *Applied Sciences*, vol. 12, no. 16, 2022. https://doi.org/10.3390/app12168162.

[30] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: an ensemble of autoencoders for online network intrusion detection," *arXiv*, 2018. https://doi.org/10.48550/arXiv.1802.09089.

[31] Z. Liu, "A review of advancements and applications of pre-trained language models in cybersecurity," In Proc. 24th International Symposium on Digital Forensics and Security, San Antonio, TX, USA, Apr. 29-30, 2024, pp. 1-10. https://doi.org/10.1109/ISDFS60797.2024.10527236.

[32] H. Xu, S. Wang, N. Li, K. Wang, Y. Zhao, K. Chen, et al., "Large language models for cyber security: a systematic literature review," *ACM Transactions on Software Engineering and Methodology*, 2024. https://doi.org/10.1145/3769676.

[33] Z. Long, H. Yan, G. Shen, X. Zhang, H. He, and L. Cheng, "A transformer-based network intrusion detection approach for cloud security," *Journal of Cloud Computing*, vol. 13, 2024. https://doi.org/10.1186/s13677-023-00574-9.

[34] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," In Proc. Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis, MN, USA, Jun. 2-7, 2019, pp. 4171-4186. https://doi.org/10.18653/v1/N19-1423.

[35] Y. Yang and X. Peng, "BERT-based network for intrusion detection system," *EURASIP Journal on Information Security*, vol. 2025, 2025. https://doi.org/10.1186/s13635-025-00191-w.

[36] H. H. Samson, "Lightweight transformer architectures for edge devices in real-time applications," *arXiv*, 2026. https://doi.org/10.48550/arXiv.2601.03290.

[37] Q. Shi, Z. Yang, and S. Shao, "TransEdge: a lightweight transformer-based anomaly detection framework for edge devices," *Journal of Intelligent Information Systems*, 2026. https://doi.org/10.1007/s10844-026-01043-w.

[38] S. Ullah, J. Wu, M. M. Kamal, and A. K. J. Saudagar, "KronNet a lightweight Kronecker enhanced feed forward neural network for efficient IoT intrusion detection," *Scientific Reports*, vol. 15, 2025. https://doi.org/10.1038/s41598-025-08921-3.

[39] O. Apejoye, "SecureFormer," 2026. [Online]. Available: https://github.com/asegun-cod/SecureFormer. [Accessed Jun. 16, 2026].

[40] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, et al., "Roberta: a robustly optimized bert pretraining approach," *arXiv*, 2019. https://doi.org/10.48550/arXiv.1907.11692.

[41] L. D. Manocchio, S. Layeghy, and M. Portmann, "FlowTransformer: a flexible python framework for flow-based network data analysis," *Software Impacts*, vol. 22, 2024. https://doi.org/10.1016/j.simpa.2024.100702.

[42] R. Xiong, Y. Yang, D. He, K. Zheng, S. Zheng, C. Xing, et al., "On layer normalization in the transformer architecture," In Proc. International Conference on Machine Learning, Virtual, Jul. 13-18, 2020, pp. 10524-10533.

[43] D. Arp, E. Quiring, F. Pendlebury, A. Warnecke, F. Pierazzi, C. Wressnegger, et al., "Dos and don'ts of machine learning in computer security," In Proc. 31st USENIX Security Symposium, Boston, MA, USA, Aug. 10-12, 2022, pp. 3971-3988.

[44] F. Hussain, R. Hussain, S. A. Hassan, and E. Hossain, "Machine learning in IoT security: current solutions and future challenges," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1686-1721, 2020. https://doi.org/10.1109/COMST.2020.2986444.

[45] G. Lemaître, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: a python toolbox to tackle the curse of imbalanced datasets in machine learning," *Journal of Machine Learning Research*, vol. 18, pp. 1-5, 2017.

[46] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321-357, 2002. https://doi.org/10.1613/jair.953.

[47] X. Tan, S. Su, Z. Huang, X. Guo, Z. Zuo, X. Sun, et al., "Wireless sensor networks intrusion detection based on SMOTE and the random forest algorithm," *Sensors*, vol. 19, no. 1, 2019. https://doi.org/10.3390/s19010203.

[48] M. J. Bommarito, D. M. Katz, and J. Bommarito, "KL3M tokenizers: a family of domain-specific and character-level tokenizers for legal, financial, and preprocessing applications," *arXiv*, 2025. https://doi.org/10.48550/arXiv.2503.17247.

[49] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," In Proc. 30th International Conference on Machine Learning, Atlanta, GA, USA, Jun. 16-21, 2013, pp. 1310-1318.

[50] J. Zhang, T. He, S. Sra, and A. Jadbabaie, "Why gradient clipping accelerates training: a theoretical justification for adaptivity," *arXiv*, 2019. https://doi.org/10.48550/arXiv.1905.11881.

[51] C. Fu, H. Huang, X. Chen, Y. Tian, and J. Zhao, "Learn-to-share: a hardware-friendly transfer learning framework exploiting computation and parameter sharing," In Proc. 38th International Conference on Machine Learning, Virtual, Jul. 18-24, 2021, pp. 3469-3479.

[52] E. M. de Elias, V. S. Carriel, G. W. De Oliveira, A. L. Dos Santos, M. Nogueira, R. H. Junior, et al., "A hybrid CNN-LSTM model for IIoT edge privacy-aware intrusion detection," In Proc. IEEE Latin-American Conference on Communications, Rio de Janeiro, Brazil, Nov. 30-Dec. 2, 2022, pp. 1-6. https://doi.org/10.1109/LATINCOM56090.2022.10000468.

[53] J. J. Tejero-Fernández and A. Sánchez-Macián, "Evaluating language models for threat detection in IoT security logs," *arXiv*, 2025. https://doi.org/10.48550/arXiv.2507.02390.