A Lightweight AI-Based Intrusion Detection System (IDS) For Resource-Constrained Networks

Sian Caine
The University of Cape Town
Cape Town, South Africa
cnxsia001@myuct.ac.za

ABSTRACT

Intrusion detection systems (IDSs) have become crucial to networks, particularly with the growth of IoT, rising cybercrime and increasingly sophisticated threats. Traditional intrusion detection systems struggle to scale effectively with these increases and to detect zero-day attacks. AI-based IDSs offer improved detection accuracy and adaptability, but are often too resource-intensive. The **LAIDS** (Lightweight AI-based Intrusion Detection System) project aims to address this gap by designing and evaluating lightweight AI-IDS models that balance strong detection capabilities with low resource usage. Three core model architectures are explored: a baseline CNN, a PCA-CNN and an AE-MLP, which are trained and evaluated with the CICIDS2017 dataset. These models, along with their quantised variants, are evaluated on their classification performance, resource efficiency and their ability to generalise to 'zero-day attacks'. The results indicate that a dynamic INT8 quantised PCA-CNN is the most promising model for deployment as a lightweight intrusion detection system. The model achieved some of the lowest CPU processing times, inference latency and false negative rates. Furthermore, the model showed the highest detection rate of DoS samples when given as zero-day attacks. The paper advances AI-based IDS solutions and demonstrates their suitability for resource-constrained environments.

1 INTRODUCTION AND MOTIVATION

In recent years, there has been exponential growth in low-resource device networks such as Internet of Things (IoT) and mobile edge computing networks [19, 23]. Low-resource networks can be defined as networks built with limited capacity links and computing devices [21]. The rapid growth of these networks has led to increased network traffic, distributed topographies and more diverse device connections [4, 10]. These expansions, which often lacks robust security, create a larger attack surface for malicious actors to exploit [19, 28]. To mitigate security risks, intrusion detection systems (IDSs) are often implemented in networks to monitor traffic and identify suspicious activity [15, 34]. However, the increasing number and sophistication of cyber-attacks, particularly 'zero-day attacks', have made it difficult for traditional systems, such as signature and anomaly-based IDSs, to keep pace and identify these unknown threats. Furthermore, they often require a great amount of computational power to be efficient [9, 26]. These limitations in traditional IDSs have led to the development of more advanced Artificial Intelligence (AI) based intrusion detection systems. Reddy et al.'s study [16] suggests AI-based IDSs offer improved accuracy over traditional IDSs in detecting known and zero-day attacks. However, their high computational costs present a significant barrier to deployment in resource-constrained environments [16, 26]. Therefore, there is a need for lightweight AI-based IDSs that can balance strong detection performance with efficient resource usage. The LAIDS project evaluates several lightweight AI-based models designed for low-resource environments. These include a Convolutional Neural Network (CNN), a Principal Component Analysis enhanced CNN (PCA-CNN) and an Autoencoder (AE), Multilayer Perceptron (MLP) ensemble model. It is crucial to continuously research new solutions in this field as IoT, mobile edge computing networks and their threats advance [15]. Without adequate protection for these low-resource networks, there is a risk of device exploitation, exposure of personal information and privacy violations [19].

1.1 Problem Statement

Low-resource networks, such as IoT, and the sophistication of the threats targeting them, are continuously evolving. Traditional IDSs have become inefficient and AI-based approaches are impractical due to their high computational demands [16]. Therefore, a critical gap has developed: the absence of lightweight AI-based intrusion detection systems that can deliver high detection accuracy while remaining feasible for deployment in resource-constrained networks.

1.2 Research Question

To address this gap, this study investigates how a series of lightweight AI-IDS models can be optimised for resource-constrained networks while still maintaining strong detection performance. The paper specifically examines the trade-off between minimising resource usage, such as CPU and memory, and sustaining key performance metrics, including accuracy, precision and recall. Furthermore, the models' ability to detect unseen attacks is assessed. The investigation is guided by the following research questions:

Questions

(1) Can a PCA-CNN or AE-MLP IDS achieve higher resource efficiency than a baseline 1D-CNN IDS, while maintaining comparable or superior performance in a resource-constrained environment?
(2) Can a quantised version of a PCA-CNN or AE-MLP IDS achieve higher resource efficiency than a quantised version of the 1D-CNN IDS baseline, while maintaining comparable or superior performance in a resource-constrained environment?
(3) Can PCA-CNN or AE-MLP IDS generalise to unseen data more effectively than a baseline 1D-CNN?

2 BACKGROUND AND RELATED WORK

This section reviews the three deep learning approaches that form the bases of this study: a Convolutional Neural Network (CNN), a

Principal Component Analysis combined with a CNN (PCA-CNN) and Autoencoder–Multilayer Perceptron (AE-MLP) model. For each model, their underlying theories are introduced, followed by a discussion of the methods and findings of related work that have investigated their intrusion detection capabilities.

2.1 Convolutional Neural Network

Convolutional Neural Networks are supervised deep learning models that consist of an input layer, several hidden layers and an output layer [30]. Within these hidden layers are convolution and pooling layers [18]. Convolutional layers identify significant features within the input data, while the pooling layers reduce dimensionality of the data [28, 29]. CNNs are able to evaluate large amounts of high-dimensional data and uncover underlying patterns [16]. Studies by Reddy et al [16] and Sowmya et al. [25] have demonstrated that CNNs achieve high accuracy with intrusion detection datasets and can outperform most other deep learning models. Due to CNNs' efficiency in intrusion detection, many researchers are exploring more lightweight CNN models specifically designed for deployment in low-resource network.

Sun and Zhao [28] proposed TinyNIDS, a 2D-CNN–based intrusion detection system optimised for edge devices in 6G networks using post-training quantisation and pruning to reduce latency. Their evaluation showed that TinyNIDS achieved an accuracy of 96.5%, outperforming the full precision deep learning models they had developed, while also maintaining the lowest latency. Their full precision CNN only achieved a 93.8% accuracy rate, while their RNN achieved 92.1% [28]. The study primarily focuses on the performance of the models, with limited evaluation of their resource efficiency in terms of CPU and RAM usage. Additionally, the paper explores only the models' binary classification capabilities and does not assess their ability to identify zero-day attacks.

Qazi et al. [14] employed a 1D-CNN for network intrusion detection using the CICIDS2017 dataset for training and evaluation. The model consisted of six convolutional layers with 32 to 16 filters each, a single max-pooling layer, a dropout layer, a flattening layer and five dense layers. It achieved an accuracy of 98.96%, a precision of 98.7% and a recall of 99.2%. However, the model was evaluated on only three of the six attack types present in the CICIDS2017 dataset. The study also does not analyse the CNN model's resource usage nor does it evaluate the CNN's ability to detect unseen attack types.

2.2 Convolutional Neural Network with Principal Component Analysis

Principal Component Analysis is an unsupervised dimensionality reduction technique that transforms high-dimensional data into a low-dimensional form. This is done by using linear combinations of the original data features to generate a new subset of features [8]. These new features, known as principal components, contain the most variance from the original data [3, 27]. High computational complexity is a common problem deep learning-based intrusion detection systems face [34]. Researchers such as Zhao et al. [34] have utilised PCA as a pre-processing step so that their deep learning models can become more lightweight, while still achieving high performance.

Awotunde et al. [3] proposed a hybrid Kernel PCA (KPCA) and CNN model for intrusion detection in IoT environments. Their model achieved an accuracy of 99.35%, a precision of 98.57% and a recall of 99.71%, outperforming their baseline CNN that was developed for comparison. While the study demonstrated the strong detection capabilities of the KPCA-CNN, it did not directly address the model's resource usage or its ability to detect zero-day attacks. Additionally, the study focused solely on the model's binary classification capabilities. It was also trained and evaluated on the IoTID20 dataset, a relatively small dataset with 626,000 samples and four attack classes.

Abed et al. [1] developed two PCA-CNN models, trained and evaluated on the UNSW-NB15 dataset. One model reduced the input data's features from 49 to 15 principal components (PCA15-CNN), while the other reduced the feature set to 10 principal components (PCA10-CNN). The PCA15-CNN and PCA10-CNN achieved accuracies of 93.81% and 97.91% respectively. While Abed et al. [1] demonstrate that a PCA-CNN can be an effective intrusion detection system, their study does not include a baseline CNN model for comparison, making it difficult to isolate the effect of PCA on the CNN's performance. Additionally, the study considers only the models' binary classification capabilities and does not address resource usage or evaluate the model's ability to detect unknown attack types.

2.3 Autoencoder-Multilayer Perceptron

Autoencoders are unsupervised deep learning models composed of an encoder and a decoder [25, 30]. AEs can be trained exclusively on unlabelled normal network traffic, allowing them to identify deviations from the normal traffic patterns they expect to encounter during deployment[13]. The encoder component compresses input data into a lower-dimensional representation, while the decoder tries to reconstruct the input data from this compressed form [12]. During the reconstruction process, the decoder aims to minimise the reconstruction error, which is a measure of how accurately the model can replicate its input [12]. When anomalous data is fed into the AE, the reconstruction error is significantly high, signalling that the data is a potential threat [22].

Multilayer Perceptrons are a type of feed-forward neural network consisting of an input layer, several hidden layers and an output layer for classification. The hidden layers are primarily fully connected, or 'dense' layers, that process their input data with an activation function before passing the output to the next layer [2]. MLPs are widely used for classification tasks as they capture complex, non-linear relationships in input data. This makes them well-suited for detecting malicious network traffic, which is often high-dimensional with intricate non-linear patterns [33].

Sharmila and Nagapadma [22] proposed two lightweight Quantised Autoencoders: the QAE-fp16 and QAE-u8, designed to overcome the limitations of traditional AEs and provide a practical solution for intrusion detection in resource-constrained networks. Post-quantisation techniques, including clustering and pruning, were applied to the baseline autoencoder to produce these quantised models. Experiments with a Raspberry Pi demonstrated that the quantised models could achieve a high detection accuracy and precision rate, while requiring less CPU power, memory and processing

time than the baseline AE. However, the study also showed that these efficiency gains of the models came with a slight trade-off in performance. The baseline accuracy, precision and recall rates were around 98.40% while the QAE-fp16 and QAE-u8 had rates around 97.25% and 96.35% respectively. While the study details the performance and resource usage of the models, it does not test the models ability to detect unknown attacks. Furthermore, only binary classification was possible for the AE models.

Wei et al. [31] developed a hybrid AE-MLP model for DDoS attack detection using the CICDDoS2019 dataset. In their approach, the AE is first trained in an unsupervised manner to learn compressed representations of the network traffic. These compressed features are extracted from the AE's bottleneck layer and used to train the MLP, with corresponding labels, in a supervised manner. During evaluation of the model, new network traffic data is passed through the AE to have its features compress and then classified by the MLP. The AE-MLP model achieved an accuracy of 98.34%, a precision of 97.91% and a recall of 98.48%. However, the model was designed specifically to detect DDoS attacks, which limits its applicability in networks exposed to a variety of threats. Moreover, although the model was intended for deployment in IoT environments, the study does not discuss its resource usage. The model's ability to detect unseen attacks was also not evaluated.

3 THE CICIDS2017 NETWORK TRAFFIC DATASET

The CICIDS2017 dataset was developed by the Canadian Institute of Cybersecurity (CIC) to support the training and evaluation of machine learning models for intrusion detection [20]. The dataset includes the most common attack scenarios in networks. The benign traffic was generated using the CIC B-Profile system, which simulates realistic traffic patterns of normal user activity on a network. A controlled, isolated network was used to run a network traffic simulation, with both benign and attack traffic present. The CICIDS2017 dataset was then extracted from this simulation. The dataset's samples are labelled network flows, extracted using CICFlowMeter, which contain over 80 data features [20]. The dataset is spread across multiple CSV files, each representing network activity of a specific weekday. The CICIDS2017 dataset was chosen for this study due to its large set of features and diverse range of common attack types. Furthermore, the dataset contains a substantial amount of benign traffic, which is essential for training the unsupervised model in this study.

3.1 Preprocessing the Dataset

A preprocessed version of the CICIDS2017 from Kaggle [17], containing 52 features, was used for this study. The preprocessed CICIDS2017 dataset combines the multiple CICIDS2017 CSV files into a single dataset and removes duplicates, infinite values and missing entries. Furthermore, columns with only a single unique value or those highly correlated with other features columns were removed to reduce redundancy. Statistically irrelevant features, identified through the use of a Kruskal–Wallis H-test and Random Forest–based feature selection, were also discarded. Lastly, rare attack types such as *Infiltration* and *Heartbleed*, which had only thirty-six and eleven samples respectively, were excluded. This was a necessary step as the classes contained insufficient samples to support effective model training or to allow for meaningful oversampling.

The preprocessed CICIDS2017 dataset is divided into five subsets as seen in Table 1. The classifier training and validation sets were utilised by the baseline CNN, PCA-CNN and MLP. The sets containing malware samples were stratified to ensure that the proportions of each malware class were consistent across all sets. Additionally, a fixed random seed was used during the random assignment of samples to datasets to guarantee reproducibility. The AE was trained solely on benign samples so that it could learn normal traffic patterns and accurately reconstruct them. As AEs are deep learning models, they require large amounts of data to be adequately trained [25]. For this reason, around 70% of the benign data was allocated to the autoencoder for its training. As the CICIDS2017 dataset was highly imbalanced, the benign data in the baseline CNN and PCA-CNN datasets would have been undersampled regardless. Furthermore, evaluations of the models' during their development indicated that increasing the amount of benign data did not improve the classifiers' performance.

Table 1: CICIDS2017 Dataset Split Before ADASYN

Class Type	AE Training Set	AE Validation Set	Classifier's Training Set	Classifier's Validation Set	Test Set
Normal Traffic	1,357,596	150,844	301,688	75,423	209,506
DoS	-	-	123,997	30,999	38,749
DDoS	-	-	81,929	20,482	25,603
Port Scanning	-	-	58,044	14,511	18,139
Brute Force	-	-	5,856	1,464	1,830
Web Attacks	-	-	1,371	343	429
Bots	-	-	1,247	312	389

After the initial split in Table 1, the oversampling technique, ADASYN, is applied to the malicious classes in the classifiers' training set. ADASYN is an adaptive synthetic sampling method that enables a model to learn under-represented classes in a dataset more effectively. It achieves this by adaptively generating synthetic samples for minority classes, producing more samples for classes that are harder to learn. Furthermore, the harder a sample is to learn, which is typically the case for samples close to a classifier's decision boundary, the more synthetic samples are created in its vicinity. This approach reduces a classifier model's bias toward the majority class in a dataset [7]. The Imbalanced-learn Python library's *adasyn* function was used to oversample the minority classes in the CICIDS2017 dataset. Synthetic samples were generated based on the five nearest neighbours within each minority class, with the number of synthetic samples generated determined by the ADASYN algorithm. The new classifier training set can be seen in Table 2. The datasets were then shuffled, re-indexed and class labels for each dataset were integer-encoded and placed in separate lists.

Table 2: CICIDS2017 Dataset Split After ADASYN

Class Type	AE Training Set	AE Validation Set	Classifiers' Training Set	Classifiers' Validation Set	Test Set
Normal Traffic	1,357,596	150,844	301,688	75,423	209,506
DoS	-	-	301,889	30,999	38,749
DDoS	-	-	301,715	20,482	25,603
Port Scanning	-	-	301,212	14,511	18,139
Brute Force	-	-	301,697	1,464	1,830
Web Attacks	-	-	301,702	343	429
Bots	-	-	301,702	312	389

4 MODEL DESIGN AND IMPLEMENTATION

This section presents the design and implementation of the baseline CNN, PCA-CNN and AE-MLP models used in this study. It details the architecture of each model and the rationale behind the chosen layers and hyperparameters values. The data processing pipeline for each model is also described, along with their quantised variants.

4.1 Baseline CNN Model

In contrast to previous CNN-based IDSs discussed in Section 2, the baseline model developed in this study is a 1D-CNN. A 1D-CNN architecture was selected due to its suitability for processing one-dimensional data, such as network traffic. Moreover, given that this study aims to develop lightweight AI-based intrusion detection systems, a 1D-CNN offers considerably lower computational complexity and reduced resource requirements compared to a 2D-CNN [14]. The baseline's architecture seen in Figure 1 was built using the TensorFlow Python library and consists of eleven layers. The *BayesianOptimization* tuner from the KerasTuner Python library was employed to perform hyperparameter tuning on the baseline CNN model. Bayesian optimisation was selected as it is the most efficient tuning method for models with many hyperparameters. It constructs a probabilistic model of an objective function and uses this model to guide the tuning process. In the baseline model's case, the objective function was to maximise its validation accuracy. This approach reduces the computational cost of hyperparameter tuning by making informed decisions about which hyperparameter settings to evaluate next [11]. The optimisation was run over 10 trials, after which the best performing set of parameters was outputted. The hyperparameter choices for each layer, together with their results, are shown in Table 3. After the Bayesian optimisation tuning, additional manual tuning was performed to further refine the model's hyperparameters. This process identified improvements in the model's performance when the dropout rate was reduced from 0.5 to 0.3 and the learning rate was lowered from 0.005 to 0.003.

Table 3: The Baseline CNN's Bayesian Optimisation Hyperparameter Tuning Results.

Layer	Hyperparameter Choices	Best Parameter
Conv1D 1	Filters:{8, 16, 32, 64, 128}	32
	Kernel Size:{2, 3, 5}	2
Conv1D 2	Filters:{8, 16, 32, 64, 128}	16
	Kernel Size:{2, 3, 5}	3
Dense	Units:{8, 12, 24, 64}	64
	L2 Regularizer:$\{0.0, 10^{-3}, 10^{-4}, 10^{-2}\}$	0.0
Dropout	Dropout Rate 1:{0.0, 0.2, 0.25, 0.3, 0.5}	0.0
	Dropout Rate 2:{0.0, 0.2, 0.25, 0.3, 0.5}	0.5
Optimiser	Learning Rate:$\{10^{-3}, 5 \times 10^{-3}, 10^{-2}\}$	5×10^{-3}

The baseline's first layer is the input layer, which requires the data to be reshaped into the form (features, channels), where channels = 1 since each feature only contains one value. This layer is followed by two convolutional layers. Each convolutional layer contains a Conv1D layer that performs feature extraction on its input data. The first Conv1D layer's filters learn complex non-linear local patterns through analysing two consecutive features in the input data at a time. The layer outputs 32 feature maps. The second Conv1D layer's filters create 16 new feature maps from the previous feature maps, while searching through three consecutive features at a time. Both Conv1D layers are followed by a BatchNormalization layer, which normalises the output activation values of the feature maps. The normalised feature maps are then downsized by the MaxPooling1D layers. This downsampling is performed by analysing two values at a time in the feature maps and discarding the smaller value. The feature maps from the second convolutional layer are passed into a GlobalAveragePooling1D layer. This layer reshapes the data into a 1D vector by taking the average values across each feature map. The resulting vector, which contains a single value for each of the 16 feature maps, is fed into a dense layer containing 64 fully connected neurons. A dropout layer is applied afterwards, which randomly resets 30% of the model's neurons during training to prevent overfitting. The final layer, the output layer, is a dense layer with 7 fully connected neurons, one for each class in the dataset. This layer uses a softmax activation to produce a probability distribution vector across the seven classes. The vector's values represent the likelihood of a sample belonging to each class. The model's final predication of a sample's class type is based on the highest probability value in its vector. The baseline model was compiled using the sparse categorical cross-entropy loss function and the Adam optimiser.

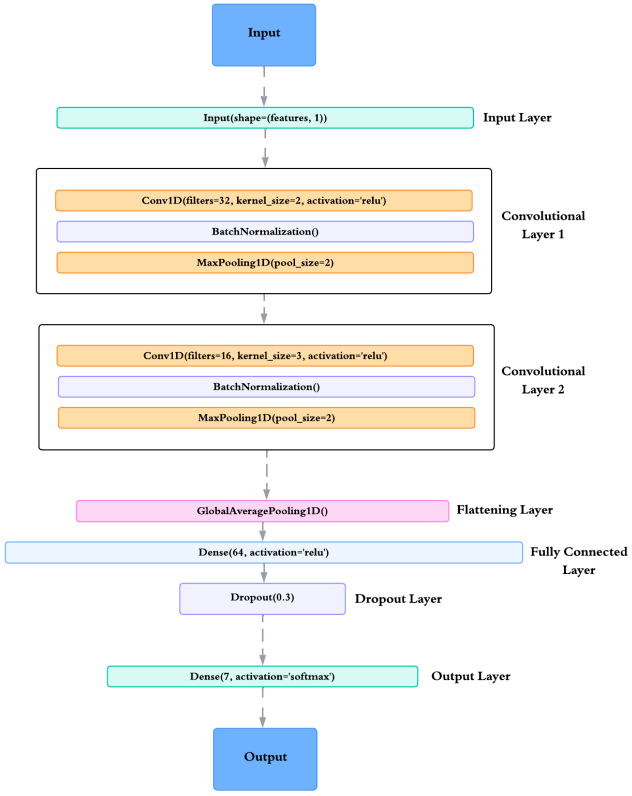

Figure 1: Baseline CNN Model Architecture

The baseline model is a full-precision (FP32) model, meaning its weights and activations are stored in a 32-bit floating-point format. The baseline was quantised into three variations for this study: an

FP16 model, a dynamic range INT8 model and a full INT8 model. The FP16 quantised model stores its weights in a 16-bit floating-point format, which is half-precision, while its activations remain in FP32. The dynamic range INT8 model has only its weights quantitated to an 8-bit integer format, while its activations remain in FP32. Finally, the full INT8 model has its weights and activations both converted into 8-bit integers during its quantisation. This process requires a representative dataset during its conversion to properly calibrate and quantise the full INT8 model's activation values.

4.2 Principle Components Anaylsis-CNN Model

High-dimensional data often contains unnecessary noise, as many features may be redundant or irrelevant. This noisy data can increase the computing time of a classification model as it tries to learn patterns from larger, more complex data samples. Consequently, this can reduce the model's ability to accurately detect malicious activity in network traffic [35]. By reducing the number of features in the input data, PCA enables a model to identify meaningful patterns more easily, improving intrusion detection accuracy while also reducing processing time [8, 27]. This study investigates how integrating PCA's feature reduction capabilities can enhance the effectiveness of a of the 1D-CNN baseline model. The Scikit-learn Python library was used to create a *PCA* instance with 21 principle components. The PCA instance made use of singular value decomposition (SVD) to perform linear dimensionality reduction on the CICIDS2017 data, projecting the data into a lower-dimensional space. The PCA-CNN model was evaluated using 10, 18 and 21 principal components, with 18 components capturing 95% of the variance and 21 components capturing 100% of the variance in the data. An assessment of the CNN model's performance across these configurations showed that it achieved the best results when the dataset features were reduced to 21 principal components. To ensure a fair comparison and to evaluate the effectiveness of PCA, the CNN component of this model is an exact replica of the baseline model's architecture. The only difference is that the PCA-CNN's input data contained only 21 features, whereas the baseline CNN's input had 52. The PCA enhanced CNN models are quantised into the same three variants as the baseline model.

4.3 Autoencoder-Multilayer Perceptron Model

Autoencoder Model

The autoencoder is a fully connected, symmetric deep neural network with 11 layers, built using the TensorFlow Python library. The AE in Figure 2 was inspired by similarly structured anomaly detection autoencoders and refined through hyperparameter tuning. Tuning results indicated that this architecture is the most effective for malware detection in network traffic. Bayesian optimisation was not applied to the autoencoder, as the resulting configurations did not preserve a symmetric encoder-decoder architecture. The first layer, the input layer, takes the number of features (52) of the input data as a parameter. The next five layers form the encoder, which compresses the input data into a lower-dimensional representation. The encoder layers make use of the ReLU activation function to capture complex, non-linear patterns in the data while it is compressed. The following four layers form the decoder, which also makes use of ReLU function, tries to reconstruct the compressed input data back to its original form. The final layer, the output layer, has the same number of neurons as the number of features in the original input data. It uses a sigmoid activation function when computing the final reconstructed input data values. The model is then compiled with the mean squared error loss function and the Adam optimiser.

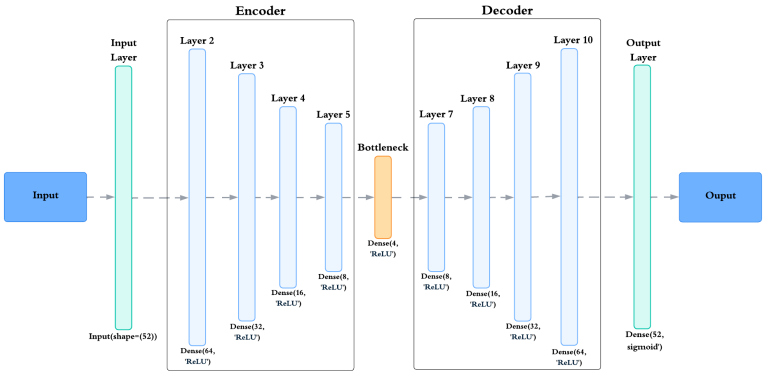

Figure 2: Autoencoder architecture

The reconstruction error values of each feature of a sample are then compared against a set of per-feature reconstruction error thresholds. If any feature reconstruction errors of a sample are above their specific threshold, the sample is labelled malicious. After hyperparameter tuning, it was found that creating thresholds for each feature worked best to detect malware samples. These thresholds are computed using Equation 1, which calculates the thresholds for a feature i as the sum of the mean absolute error of feature i and a single standard deviation from that mean. The mean absolute error of a feature i being the average reconstruction error value of the feature for all the samples in the validation dataset. This approach provided a realistic estimate of the reconstruction errors the AE produce for unseen benign traffic. By accounting for these errors, it allowed for more robust per-feature thresholds to be created and reduced the likelihood of incorrectly classifying unseen benign data as malicious.

$$\text{Threshold}_i = \text{MAE}_i + \sigma_i \tag{1}$$

$$\text{MAE}_i = \frac{1}{N}\sum_{k=1}^{N}\left|\hat{F}_{\text{val},i}^{(k)} - F_{\text{val},i}^{(k)}\right| \tag{2}$$

$$\sigma_i = \sqrt{\frac{1}{N}\sum_{k=1}^{N}\left(\left|\hat{F}_{\text{val},i}^{(k)} - F_{\text{val},i}^{(k)}\right| - \text{MAE}_i\right)^2} \tag{3}$$

Where:

- MAE_i = mean absolute reconstruction error for feature i
- σ_i = standard deviation of reconstruction error for feature i
- N = total number of validation samples
- $F_{\text{val},i}^{(k)}$ = original value of feature i in sample k
- $\hat{F}_{\text{val},i}^{(k)}$ = reconstructed value of feature i in sample k

- $|\hat{F}^{(k)}_{\text{val},i} - F^{(k)}_{\text{val},i}|$ = absolute reconstruction error for feature i in sample k

Multilayer Perceptron Model

The classifier model in Figure 3 is a fully connected Multilayer Perceptron (MLP) built using the Tensorflow Python library. The model takes the test samples flagged as malicious by the AE as input. Its primary role is to classify the specific types of malware present in its input samples. To address the AE's high false positive rate, the model also includes normal traffic as a classification category. This ensures that benign traffic mistakenly flagged as malicious by the AE can be correctly reclassified.

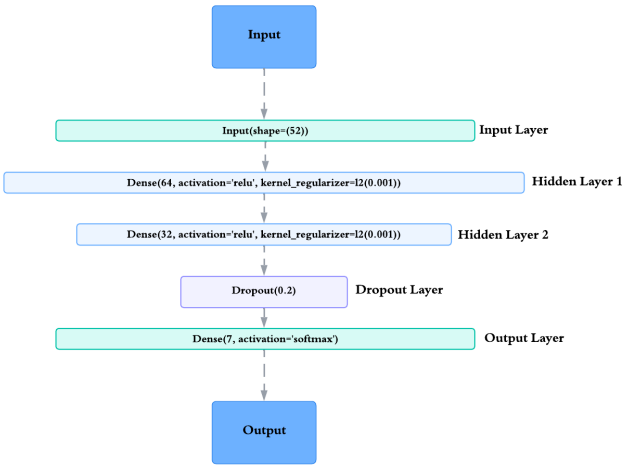

Figure 3: MultiLayer Perceptron architecture

The model architecture begins with an input layer that takes the features of the network traffic samples as a parameter. This layer is followed by two hidden dense layers that progressively compress and transform the input features. L2 regularisation functions with penalty values of 0.001 was employed within these layers to prevent the model from overfitting. These layers are followed by a dropout layer with a rate of 0.2. The final layer of the MLP is a dense layer with 7 fully connected neurons and a softmax function.

The AE and MLP models were each quantised into the same three variants as the baseline and PCA-CNN models: FP16, dynamic INT8 and full INT8. Each AE–MLP pair are the same quantised variant to ensure consistency.

5 EVALUATION METRICS

The performance metrics used in this study to evaluate the models and their quantised variants include accuracy, precision and recall. The study also considers the false positive and false negative rates of the models. Resource usage is primarily assessed in terms of model storage size, CPU time and RAM required to process a single sample.

5.1 Detection Performance Metrics

The detection performance metrics are computed using the confusion matrix from the Scikit-learn Python *metrics* module. True positives (TP) represent the number of malicious network traffic samples correctly identified (even if classified into the wrong malware class), while true negatives (TN) represent the number of normal traffic samples correctly identified by the model. False positives (FP) are normal traffic samples that are misclassified as malicious and false negatives (FN) are malicious traffic samples that are incorrectly reported as benign by the model [32]. Based on these definitions, the following performance metrics were recorded and evaluated:

Accuracy measures the proportion of samples correctly classified as benign or malicious.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}\,[32] \tag{4}$$

Precision is a measure of the amount of malware samples correctly identified as malware, out of all samples classified as malware by a model.

$$\text{Precision} = \frac{TP}{TP + FP}\,[15] \tag{5}$$

Recall or *'Detection Rate'* is the proportion of malware samples correctly identified as malware, out of the total number of malware samples in the dataset [15].

$$\text{Recall} = \frac{TP}{TP + FN}\,[15] \tag{6}$$

False Negative Rate is the percentage of malware samples incorrectly classified as benign samples.

$$\text{FNR}\ (\%) = \frac{FN}{FN + TP} \times 100\,[15] \tag{7}$$

False Positive Rate is the percentage of benign samples incorrectly classified as malware by a model.

$$\text{FPR}\ (\%) = \frac{FP}{FP + TN} \times 100\,[32] \tag{8}$$

5.2 Resource Efficiency Metrics

The resource efficiency of the models was evaluated using *psutil* to monitor resource usage while processing a single sample from the dataset. These measurements help determine if the quantised models are suitable for deployment in resource-constrained network environments. The target hardware constraints for this evaluation are: 4 GB RAM, 500 MB storage, no GPU acceleration and 2 logical CPU cores with a clock speed of 2.2 GHz. Based on these requirements, the following resource usage metrics were recorded:

Model Size (MB): The amount storage required by a model.

Average Memory Usage (KB): The runtime memory required by a model while it is processing a single sample.

Average CPU Usage (ms): The amount of CPU time, in milliseconds, a model consumes while processing a single sample.

Average Inference Latency (ms): The wall-clock time taken, in milliseconds, for a model to classify a single sample.

6 EXPERIMENT DESIGN

The primary aim of this study is to evaluate the performance, resource efficiency and generalisation capabilities of several lightweight AI-based intrusion detection system models. Specifically, the experiments investigate whether a PCA-CNN or AE-MLP models can achieve comparable or superior detection performance to a baseline CNN, while minimising resource usage. Additionally,

this study investigates how post-training quantisation affects these models and assesses their ability to generalise and detect unseen attacks.

6.1 Experiment 1: Benchmarking

The purpose of this experiment is to establish a performance and resource baseline by benchmarking the initial full precision (FP32) models against the baseline CNN. Benchmarking provides a fair comparison across the different model architectures and aids in answering the first research question of this study. Additionally, benchmarking determines whether the quantised versions of the models maintained comparable performance while improving resource efficiency.

6.1.1 ***Further Data Preprocessing.*** In addition to the dataset preprocessing detailed in Section 3.1, each model's dataset was further tailored to their specific needs. The datasets for the baseline and PCA-CNN models were standardised using the *StandardScaler* from Scikit-learn's preprocessing module. The scaler was fitted on the training set data and subsequently applied to the validation and testing sets. For the PCA-CNN model, an additional transformation was applied using Scikit-learn's *PCA* module. The PCA instance was fitted to the training data and then applied to the validation and test sets to project the features into principal components. Finally, the datasets for both models were reshaped to match the required CNN input format. The AE-MLP's datasets were normalised using the *MinMaxScaler* from the Scikit-learn preprocessing module. The scaler was fitted on the AE's training set and then applied to all other the AE-MLP datasets used in the experiment for consistency purposes and to reduce data leakage. It was found that the MinMaxScaler was the best scaler to use to normalise the AE-MLP's input data.

6.1.2 ***Model Training.*** The models were all trained over twenty epochs, with early stopping implemented to halt training if the validation dataset's loss started increasing consistently over four epochs. This allowed the models to effectively learn the training data without overfitting. After the models were tuned and trained, the best-performing version of each model was evaluated on their test set and their performance and resource efficiency metrics were recorded.

6.1.3 ***Performance and Resource Measurement*** . The performance of each model was calculated using the equations outlined in Section 5.1, with values obtained from their confusion matrices. The matrices are generated using the Scikit-learn library's *confusion_matrix* function, which compares the model's predictions on the test set against the true labels. To measure resource usage, each model was run in a dedicated Google Colab virtual environment with set CPU and RAM constraints. For each model, inference was performed on a single test sample 1000 times and CPU usage, RAM consumption and inference time were recorded after each run. The average values of these measurements was computed to obtain stable resource usage metrics. For each model, five independent resource usage measurements were taken. To ensure the accuracy and independence of each measurement, the Google Colab virtual environment was reset after each run. The resulting average resource usages are reported in Table 4.

6.2 Experiment 2: Quantisation

This experiment investigates the impact of quantisation on the performance and resource efficiency of the FP32 baseline CNN, PCA-CNN and AE-MLP models, addressing the second research question of this study. This experiment evaluates the performance and resource consumption of the quantised variants (FP16, dynamic INT8 and full INT8) of the full precision (FP32) PCA-CNN and AE-MLP models. These quantised models are subsequently compared to the corresponding quantised versions of the baseline CNN.

6.2.1 ***Post-training Quantisation.*** Each FP32 model underwent the same post-training quantisation process using the TensorFlow Lite (TFLite) Python library. The models were converted into each quantised version using the *TFLiteConverter* TensorFlow class. For the full INT8 quantised models, where both weights and activation values are quantised to 8-bit integers, a representative dataset was required to calibrate the activation values. For each full INT8 model, a subset of their corresponding FP32 model's preprocessed training set, as described in the benchmarking experiment, was selected to calibrate the model. The Full INT8 baseline CNN, PCA-CNN, MLP each had representative datasets comprised of 350 samples, made up of 50 samples drawn from each class in the dataset. A random seed was used to select these samples for reproducibility. The AE's full INT8 model used 1000 benign samples from its training set as the representative dataset.

6.2.2 ***Performance and Resource Evaluation.*** The quantised models are evaluated on the same test set as the full precision models, using the same performance and resource usage metrics. The same methods that were used to capture these metrics in the benchmarking experiment are also applied in this experiment. During the evaluation of the quantised models' performance, the Full INT8 models test data went through a slightly change in order to be preprocessed by the models. The test data must first be converted to int8-bits in order to be processed by the model. After the model has processed the test data, the models output values are converted into float32-bits in order to be interpreted.

6.3 Experiment 3: Generalisation To Unseen Data

This experiment evaluates the ability of the models and their quantised variants to generalise to previously unseen, zero-day attacks, addressing the third research question of this study. The objective of this experiment is to simulate realistic zero-day attack scenarios and assess whether the models can learn generalisable patterns of malicious behaviour rather than memorising known attack signatures. With the rapid evolution of cyberattacks, it is ideal for IDSs to be able to identify emerging threats without requiring continuous retraining on newly collected data [24]. As indicated in Section 2, many studies fail to evaluate their model's performance when faced with unseen attacks. Moreover, many studies assume that their models generalise to unseen attack data based on evaluation results, but never test their theory [6].

6.3.1 ***Zero-Day Attack Selection.*** Two attack classes were selected as the zero-day attacks for this experiment: DoS and Bots.

These classes were chosen because of their contrast in size and feature richness. As seen in the original data split of the CICIDS2017 dataset in Table 1, DoS attacks are among the most prevalent malicious classes in CICIDS2017. This allows the models to be tested on their ability to detect an abundant unseen attack type. In contrast, Bots are the least represented class, providing insight into model detection performance when the unseen attack rarely occurs. Moreover, Sharafaldin et al.'s [20] study on the creation of the CICIDS2017 dataset detailed the best feature sets for detecting each class type in the dataset. It was found that the dataset contains up to ten best features for DoS detection, providing models substantial information to support accurate classification of the class. The Bots class's best feature set however, only contained four relevant features, three of which also rank among the most important features for identifying benign traffic. Furthermore, Bots is the only class that shares key features with benign traffic.

6.3.2 ***Model Retraining and Evaluation.*** The experiment was conducted twice for each model and its quantised variants. In each iteration, one of the selected attack classes was excluded from the training set, after which the models were retrained and quantised. The resulting models were then evaluated on the test dataset, which still included the omitted attack class. The models' abilities to detect this unseen class were then recorded. Each iteration was repeated five times to ensure reliability and the models achieving the strongest zero-day detection performance were selected for analysis. The training and quantisation procedures as well as the methods used to capture models' performance were the same as those applied in the previous two experiments. Since the autoencoder model is unsupervised and trained exclusively on benign data, it was not retrained for this experiment. The MLP classifier, however, underwent the same retraining process for this experiment as the baseline CNN and PCA-CNN models. To prevent data leakage, each zero-day attack was omitted from the representative datasets used to calibrate the full INT8 quantised models.
The results of this experiment provides insight into whether this study's models can adapt to the evolving cyber threat landscape when deployed, or if they will require frequent retraining and updates to maintain effective attack detection.

6.4 Ethical, Professional and Legal Issues

6.4.1 ***Legal Considerations.*** The LAIDS project is not expected to encounter any direct legal issues, as research does not involve the collection or processing of human personal data during the AI model development. Compliance with privacy regulations, such as the Protection of Personal Information Act (POPIA), is maintained by using the publicly available CICIDS2017 dataset, which contains no sensitive or personal information [20]. All project outputs, including code, trained models and documentation, will be released as open source to promote transparency.

6.4.2 ***Ethics clearance.*** Ethical clearance was not required for this project, as it does not involve human participants or the use of sensitive personal data. The primary ethical consideration, dataset usage, is addressed by employing the CICIDS2017 dataset. As the dataset is synthetic and does not contain any real-world network traffic or personal information [20]. All experimentation present in this study were conducted in closed, isolated environments, meaning no real-world systems were put at risk. The AI models developed in this study, by design, cannot form biases to real-world network traffic origins, as they were only exposed to simulated data. This ensures that the models cannot inadvertently associate specific behaviours with real-world network addresses or users.

7 RESULTS AND DISCUSSION

This section presents the results the experiment conducted in this study, along with a detailed discussion of their implications. The reliability of the results is evaluated and the findings are used to address the research questions.

7.1 Benchmarking

Table 4 provides the benchmarking experiment results of the full precision (FP32) baseline CNN, PCA-CNN and AE-MLP models. It provides a comprehensive overview of the models' performance and resource consumption.

The baseline CNN achieved the highest accuracy rate (95.39%), meaning it could distinguish between benign and malicious network traffic the best. Additionally, the baseline CNN exhibited the lowest false positive rate (5.89%), an important attribute for deployed intrusion detection systems as it minimises wasted resources and time investigating false alarms. The baseline model is resource-efficient within the prescribed hardware constraints of this study. Its size is only 0.0194 MB, occupying a negligible portion of the 500 MB storage available on the target device. Furthermore, processing a single sample requires just 3.6 KB of RAM, a fraction of the device's 4 GB (4,194,304 KB) memory. The model also exhibits low latency, processing a single sample in 0.0648 ms, which corresponds to approximately 15,432 samples per second. Evaluating the CPU usage of the model, processing a single sample requires only 0.068 ms of CPU time on a 2.2 GHz CPU. This corresponds to approximately 149,600 CPU cycles per sample out of the 2.2 billion cycles available per second. This demonstrates that only a small fraction of the CPU's capacity is needed to process the network traffic samples. In terms of multiclass classification capabilities, the results in Table 5 shows that the baseline model detects port scan, web attacks, and bots most effectively, with rates of 99.74–99.81%. However, its confusion matrix (Figure 4, Appendix A) reveals that while it detects these attacks well, it generates many false alarms in doing so. In comparison to Sun and Zhao's [28] baseline 2D-CNN, the 1D-CNN used in this study achieved higher accuracy and recall, as well as a significantly lower false negative rate. Against Qazi et al.'s [14] 1D-CNN, this study's baseline CNN displays lower precision, recall, and accuracy rates. This performance difference may be due to the baseline model's more lightweight architecture.

Despite having lower accuracy and precision rates than the baseline, the PCA-CNN model achieved the lowest false negative rate (0.29%) and highest recall rate (99.71%) out of the three FP32 models. Only 244 malware samples were misclassified as benign out of 85,139 malware samples in the test set. This high detection rate comes at the cost of a false positive rate exceeding 10%, twice the amount of false positives than the baseline mode. This increased

Table 4: FP32 Models' Performance and Resource Usage Results

Model	Accuracy (%)	Precision (%)	Recall (TPR) (%)	FPR (%)	FNR (%)	Size (MB)	RAM Usage (KB)	CPU Usage (ms)	Inference Latency (ms)
Baseline CNN	**95.39**	87.17	98.53	**5.89**	1.47	0.0194	3.5992	0.068	0.0648
PCA-CNN	92.32	79.14	**99.71**	10.68	**0.29**	0.0194	3.694	**0.048**	**0.0493**
AE-MLP	94.93	**90.53**	92.07	9.47	7.94	0.0756	**2.627**	0.1240	0.0912

Table 5: Percentage of Each Malware Class Correctly Identified

Model	Port Scanning (%)	Web Attacks (%)	Brute Force (%)	DDoS (%)	Bots (%)	DoS (%)
Baseline CNN	**99.81**	**99.77**	96.34	99.23	**99.74**	97.04
PCA-CNN	98.65	95.34	**99.51**	**99.92**	**99.74**	**98.98**
AE-MLP	92.50	9.09	76.12	95.97	78.15	90.61

false positive rate may be due to the extreme dimensionality reduction of the CICIDS2017 dataset features from 52 to 21. Some of the low-variance features that were removed could have been important for correctly identifying certain network traffic as benign [5]. In terms of resource efficiency, the PCA-CNN was the most time-efficient, exhibiting the lowest CPU and inference time per sample. This efficiency is likely due to the reduction in features, enabling faster processing of the data [8]. The RAM usage of the PCA-CNN is an anomaly, as it is slightly higher than the baseline's. This contradicts literature that suggests the addition of PCA and the generation of highly distinct principal components simplifies model computations and reduces resource consumption [8]. The PCA-CNN's storage size is the same as the baseline model, this is expected, as both models share the same architecture. In terms of multiclass classification capabilities, the results in Table 5 indicate that the PCA-CNN can correctly classify the largest number of attack types. It performs best at identifying brute force, DDoS, bots and DoS attacks, with detection rates ranging from 98.98% to 99.92%. Table 5 indicates that the PCA-CNN faces the same issue as the baseline in classifying these classes with very high recall. In achieving such strong detection rates, it generates many false positives, particularly for the DoS and bots classes (Figure 5, Appendix A). Compared to the prior work discussion in Section 2, the PCA-CNN exhibits lower performance. However, this may be due to the prior work being trained and evaluated on different datasets, which may have been more suitable for the application of PCA.

The AE-MLP achieves the second highest accuracy (94.93%) and the highest precision rate (90.53%). Examining Table 8 in Appendix A, which displays the extended AE-MLP results, the strong precision, accuracy and recall rates of the AE-MLP can be attributed to the MLP's classification abilities. The AE alone only achieved a 68% accuracy and 47% precision. This is confirmed by the AE's confusion matrix (Figure 6, Appendix A) which shows many benign samples misclassified as malicious, leading to a high false positive rate - a common issue in unsupervised models [28]. The AE's frequent misclassification of benign traffic as malicious is likely the result of its strict anomaly detection threshold. The MLP exhibits strong performance results (Table 8) with the exception of a relatively high false positive rate of 9.47%, a common trend amongst this study's models. It is important to note that the MLP was evaluated only on samples flagged as malicious by the AE, resulting in a smaller, filtered test set. Its strong performance may be due to the exclusion of subtler malware the AE missed. In terms of multiclass classification abilities, The MLP shows highly precise classification abilities across the classes but this could also be the result of its smaller test set. However, the AE-MLP ensemble model does not achieve comparable or more precise attack classifications than the other models. Regarding resource usage, the AE-MLP requires more storage, CPU time and latency than the single models of this study, likely due to its ensemble nature. However, it demonstrated the lowest RAM usage, suggesting some efficiency advantages. A possible explanation is that, unlike CNNs, the AE-MLP does not generate multiple feature maps during processing. Although the AE-MLP contains more parameters overall, it is primarily composed of small dense layers, which likely results in lower memory requirements.

Although neither the PCA-CNN nor the AE-MLP IDS demonstrated outright superiority over the baseline 1D-CNN in terms of overall performance and resource efficiency, both models exhibited specific advantages. The PCA-CNN's lower processing time and higher recall, along with the AE-MLP's improved precision and reduced RAM usage, show that while the baseline 1D-CNN remains competitive, both models offer targeted advantages.

7.2 Quantisation

Table 6 presents the quantisation experiment results for the baseline CNN, PCA-CNN and AE-MLP models. It provides an overview of the performance and resource consumption across their FP16, Dynamic INT8 and Full INT8 quantised variants.

Several trends can be observed from the quantisation results. As the models' precision are reduced, their accuracy, precision and recall generally decline, while their false positive and false negative rates increase. FP16 and Dynamic INT8 variants, show only minor performance reductions compared to their full-precision counterparts. The deterioration of performance becomes more pronounced in the Full INT8 models, reflecting the impact of quantising both the weights and activations of the models. The Full INT8 AE-MLP preserves its performance more effectively than the other Full INT8 models. However, an examination of the extended AE-MLP results in Table 8, Appendix A, shows that this is largely attributable to the MLP component. Analysing the AE on its own, the Full INT8 AE experiences performance degradation comparable to the other fully quantised models in this study.

As 16-bit floating-point (FP16) models show no significant reduction in performance relative to their FP32 counterparts, the FP16 baseline CNN achieved the highest accuracy (95.36%). The FP16 AE-MLP would have achieved the highest precision of 90.66%, if not for the Full INT8 AE-MLP which shows a precision of 95.41%. A closer look at Table 8 reveals that this high precision is due to the MLP.

Table 6: Quantised Models' Performance and Resource Usage Results

Model	Quant Variant	Accuracy (%)	Precision (%)	Recall (TPR) (%)	FPR (%)	FNR (%)	Size (MB)	RAM Usage (KB)	CPU Usage (ms)	Inference Latency(ms)
Baseline CNN	FP16	**95.36**	86.96	98.74	6.01	1.26	0.0145	3.596	0.084	0.0778
	Dynamic INT8	95.12	86.74	98.11	6.10	1.89	**0.0131**	3.4408	0.09	0.0762
	Full INT8	64.59	43.17	71.26	38.12	28.74	0.0141	2.2672	0.116	0.256
PCA-CNN	FP16	92.32	79.12	**99.71**	10.69	**0.29**	0.0145	3.615	0.07	0.0667
	Dynamic INT8	91.99	78.44	99.67	11.13	0.33	**0.0131**	3.496	**0.066**	**0.0537**
	Full INT8	58.06	33.71	46.72	37.33	53.28	0.0141	2.2464	0.0820	0.0933
AE-MLP	FP16	94.97	90.66	92.06	9.34	7.94	0.0440	**1.946**	0.1320	0.1226
	Dynamic INT8	93.86	88.10	91.04	11.90	8.96	0.0331	3.55	0.082	0.0926
	Full INT8	90.72	**95.41**	71.31	**4.59**	28.69	0.0353	2.662	0.13	0.1290

The Full INT8 AE model alone has only a precision of 37%, consistent with its high false positive and false negative rates. Overall, the MLP's performance does not degrade as rapidly as the other models in this study when quantised. However, it does experience an increase in false negative rates as its size is reduced, reaching a 10% FNR when quantised to a Full INT8 model. As mentioned in Section 7.1, these performance results could be a product of the smaller set of samples the MLP is tested on. The FP16 PCA-CNN achieves the highest recall (99.71%). The Full INT8 AE-MLP achieves the lowest false positive rate (4.59%), but this comes at the cost of a 28.69% FNR.

The baseline CNN and PCA-CNN exhibit a general trend of decreasing resource usage as the models are reduced in size however, these decreases are modest. Their Full INT8 quantised models use the least amount of memory out of all their quantised variants but show increases in CPU time and inference latency. The reduction in memory usage is likely due to smaller weights and activation sizes of the models. The increased processing time may result from quantising their inputs to INT8 and converting their outputs back to FP32 for predictions. The PCA-CNN quantised variants consistently use the same amount or slightly more RAM than the baseline CNN quantised models, contradicting the literature again which suggests that PCA reduces computational load [8]. However, the PCA-CNN variants do consistently require less CPU time and latency, with the Dynamic INT8 PCA-CNN achieving the fastest processing times in Table 6. Although in comparison to the FP32 PCA-CNN in Table 4, the Dynamic INT8 PCA-CNN possesses a slightly higher CPU and inference latency time. The majority of AE-MLP models used less memory than the CNNs with the FP16 AE-MLP producing the lowest memory usage of (1.946 KB per sample). This is likely because the CNNs generate and store multiple feature maps during runtime, causing them to require additional memory.

Overall, although the Full INT8 models showed reduced resource usage, their performance degradation, especially their high false negative rates, makes them unsuitable for reliable intrusion detection. Among the quantised models, the PCA-CNN Dynamic INT8 stood out as the most promising candidate for a reliable low-resource IDS with its low FNR, fast processing times and reduced storage demands. While its memory usage was not the lowest, it remained below that of the FP32 CNN and PCA-CNN. The AE-MLP and its quantised variants showed inconsistent performance and resource usage. While the FP16 version performed well, other quantised versions varied. This highlights the AE-MLP's sensitivity to quantisation, making its performance less predictable than the baseline CNN or PCA-CNN.

7.3 Generalisation To Unseen Data

Table 7 presents the detection rates of the baseline CNN, PCA-CNN and AE-MLP models when evaluated on unseen or 'zero-day' attacks. Both the full-precision models and their quantised variants are included. The table reports the percentage of successful detections for DoS and Bot attacks when these attack types are treated as previously unseen.

Table 7: Results of Models' Generalisation to Unseen Attack

Model	Version	Dos Detection Rate (%)	Bots Detection Rate (%)
Baseline CNN	FP32	60.87	4.88
	FP16	60.93	4.88
	Dynamic INT8	58.35	4.88
	Full INT8	**82.96**	0.26
PCA-CNN	FP32	63.56	7.46
	FP16	63.54	7.46
	Dynamic INT8	**64.15**	7.71
	Full INT8	5.66	0.26
AE-MLP	FP32	64	60.41
	FP16	61.62	60.41
	Dynamic INT8	61.67	**60.67**
	Full INT8	58.50	49.1

Overall, the models detected unseen DoS attacks more effectively than unseen Bot attacks, with detection rates ranging from 58% to 64%. This is likely due to the similarity between DoS attack samples and DDoS samples still present in the models' training sets. Furthermore, Sharafaldin et al.'s [20] study on the CICIDS2017 dataset indicated that many of the features used by models to identity DoS samples are shared with the DDoS class. The models' confusion matrices support this notion as the majority of DoS samples were classified as DDoS across all the architectures. Moreover, the test dataset contains significantly more DoS than Bots samples, which could have aided in its identification. Comparing the models in Table 7, the PCA-CNN and AE-MLP models outperformed the baseline CNN models in detecting the unseen DoS attacks. The 82% DoS detection rate of the Full INT8 baseline CNN appears driven by a high false positive rate, reflecting poor precision rather than true identification. For this reason, the anomaly is considered unreliable and excluded from further discussion. The majority of the PCA-CNN models showed the strongest detection performance on DoS samples, with the Dynamic INT8 variant achieving a detection rate of 64%. However, the Full INT8's detection abilities are inadequate, supporting the evidence throughout this study that the Full INT8 models are unable to retain sufficient performance while minimising resource usage. The AE-MLP models showed the second strongest performance in detecting the DoS samples, with detection rates ranging from 58% to 64%. This suggests that ensemble semi-supervised models could be more effective at detecting

previously unseen attacks than singular supervised models. Moreover, it indicates that MLP-based models might be better suited than CNN-based models for detecting unknown attacks. Notably, the full INT8 AE-MLP demonstrated the most stable detection rate while also reducing resource usage. Investigating the extended generalisation results of the AE-MLP in Table 10, this strong performance can be attributed to the detection rates of 91% and 64% achieved by the full INT8 AE and MLP components, respectively. It should be taken into account that these detection rates could have been influenced by their high false postive rate.

The baseline and PCA-CNN models showed poor performance in detecting unseen Bots samples in the test set, with the PCA-CNN models performing only slightly better than the baseline. This may be due to the limited number of Bot samples in the test set, the absence of similar attacks in the training set, or the fact that the benign traffic class shares similar identification features, as discussed in Section 6.3. The AE-MLP models achieved the strongest Bot detection rates, ranging from 49% to over 60%, further supporting the superior generalisation capabilities of semi-supervised models compared to single supervised models.

The change in detection performance when models are reduced from full precision to Dynamic INT8 quantisation is negligible, indicating that reduced precision does not negatively affect their ability to detect unseen attacks. However, reducing models to Full INT8 significantly impairs their generalisation and detection capabilities. Notably, the PCA-CNN's Dynamic INT8 model showed higher detection rates compared to its FP32 and FP16 counterparts, suggesting that reduced weight precision may have enhanced its ability to generalise to unseen attacks. Both the PCA-CNN and AE-MLP outperformed the baseline CNN in detecting unseen attacks. The PCA-CNN was particularly effective at identifying attack types most similar to data in the training set, while the AE-MLP detected Bots most efficiently. Overall, the AE-MLP demonstrated the most stable detection rates across both unseen attack classes.

8 CONCLUSION

This study evaluated three core model architectures, a baseline 1D-CNN, a PCA-CNN and an AE-MLP, focusing on quantisation, resource efficiency and the ability to generalise to unseen attacks. No single model emerged as the definitive best, instead each model demonstrated distinct strengths. The full-precision baseline CNN and its FP16 quantised variant achieved the highest accuracy and the lowest false positive rates when detecting known attacks. The PCA-CNN, while less accurate and precise, produced the lowest false negative rate and delivered the fastest processing times. Among the quantised variants, the Dynamic INT8 PCA-CNN was the most promising, offering a reduced CPU time and inference latency without a major drop in performance. In contrast, the AE-MLP and its quantised versions showed more unpredictable performance and resource usage results, but generalised best to the DoS and Bots when they were posed as unseen attacks. Overall, the findings suggest that developing a lightweight, effective IDS for resource-constrained networks may require combining feature reduction techniques, such as PCA, with semi-supervised learning approaches. This balance may help achieve strong detection accuracy, efficient resource usage and better detection of novel threats.

9 LIMITATIONS AND FUTURE WORK

This study faced several limitations that shaped both the results and the scope of the investigation. First, training was constrained by limited computational resources, which restricted experimentation with larger architectures, longer training times and more extensive hyperparameter tuning. Resource usage was also measured on Google Colab, a platform where resource constraint implementations are only suggested and not strictly enforced by the system. Consequently, the reported values may have been influenced by Colab's own system overheads and the lack of guaranteed isolation within its shared runtime environments. This means the metrics collected should be viewed as an approximation rather than a definitive, isolated measurement. Limitations also lay in the CICIDS2017 dataset, which was heavily imbalanced and included only five common categories of cyber-attacks. Furthermore, while oversamples improved model performance, it also raises the risk that the models learned synthetic patterns rather than true characteristics of malicious traffic. The experimental design limited the assessment of model generalisation to only two attack types within a single dataset. As a result, the models' full generalisation capabilities were not comprehensively evaluated.

Future research can build on this study in several ways. The models' performance and resource usage could be tested on actual resource-constrained devices or within network simulations that incorporate real-world traffic conditions to obtain more reliable results. In terms of model design, future research could explore quantisation-aware training techniques, which may allow models to be compressed to Full INT8 format while retaining stronger performance. Further studies could also investigate combining feature reduction with semi-supervised learning to improve both resource efficiency and generalisation to unseen attacks. Finally, future evaluations could expand the generalisation assessment by testing multiple attack types and datasets, providing a more comprehensive understanding of the models' ability to detect new threats.

REFERENCES

[1] Ruqaya Abdulhasan Abed, Ekhlas Kadhum Hamza, and Amjad J Humaidi. 2024. A modified CNN-IDS Model for Enhancing the Efficacy of Intrusion Detection System. *Measurement Sensors* 35 (09 2024), 101299–101299. https://doi.org/10.1016/j.measen.2024.101299

[2] Layth Almahadeen, Ghayth ALMahadin, Kathari Santosh, Mohd Aarif, Pinak Deb, Maganti Syamala, and B Kiran Bala. 2024. Enhancing Threat Detection in Financial Cyber Security Through Auto Encoder-MLP Hybrid Models. *International Journal of Advanced Computer Science and Applications* 15 (2024). https://doi.org/10.14569/ijacsa.2024.0150495

[3] Joseph Awotunde, Ranjit Panigrahi, Biswajit Brahma, and Akash Bhoi. 2023. CNN-KPCA: A hybrid Convolutional Neural Network with Kernel Principal Component Analysis for Intrusion Detection System for the Internet of Things Environments. https://ceur-ws.org/Vol-3584/Paper-7.pdf

[4] Shahid Allah Bakhsh, Muhammad Almas Khan, Fawad Ahmed, Mohammed S. Alshehri, Hisham Ali, and Jawad Ahmad. 2023. Enhancing IoT network security through deep learning-powered Intrusion Detection System. *Internet of Things* 24 (12 2023), 100936. https://doi.org/10.1016/j.iot.2023.100936

[5] Bo Cao, Chenghai Li, Yafei Song, and Xiaoshi Fan. 2022. Network Intrusion Detection Technology Based on Convolutional Neural Network and BiGRU. *Computational Intelligence and Neuroscience* 2022 (04 2022), 1–20. https://doi.org/10.1155/2022/1942847

[6] Laurens D'hooge, Miel Verkerken, Tim Wauters, Filip De Turck, and Bruno Volckaert. 2023. Investigating Generalized Performance of Data-Constrained Supervised Machine Learning Models on Novel, Related Samples in Intrusion Detection. *Sensors* 23 (02 2023), 1846. https://doi.org/10.3390/s23041846

[7] Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. 2008. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*. Ieee, 1322–1328.

[8] K. Keerthi Vasan and B. Surendiran. 2016. Dimensionality reduction using Principal Component Analysis for network intrusion detection. *Perspectives in Science* 8 (09 2016), 510–512. https://doi.org/10.1016/j.pisc.2016.05.010

[9] Ansam Khraisat, Iqbal Gondal, Peter Vamplew, and Joarder Kamruzzaman. 2019. Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity* 2 (07 2019), 1–22. https://doi.org/10.1186/s42400-019-0038-7

[10] Rakesh Kumar, Mayank Swarnkar, Gaurav Singal, and Neeraj Kumar. 2021. IoT Network Traffic Classification using Machine Learning Algorithms: An Experimental Analysis. *IEEE Internet of Things Journal* (2021), 1–1. https://doi.org/10.1109/jiot.2021.3121517

[11] Mohammad Masum, Hossain Shahriar, Hisham Haddad, Md Jobair Hossain Faruk, Maria Valero, Md Abdullah Khan, Mohammad A. Rahman, Muhaiminul I. Adnan, Alfredo Cuzzocrea, and Fan Wu. 2021. Bayesian Hyperparameter Optimization for Deep Neural Network-Based Network Intrusion Detection. *2021 IEEE International Conference on Big Data (Big Data)* (12 2021). https://doi.org/10.1109/bigdata52589.2021.9671576

[12] Ghada AL Mukhaini, Mohammed Anbar, Selvakumar Manickam, Taief Alaa Al-Amiedy, and Ammar Al Momani. 2023. A systematic literature review of recent lightweight detection approaches leveraging machine and deep learning mechanisms in Internet of Things networks. *Journal of King Saud University - Computer and Information Sciences* 36 (11 2023), 101866–101866. https://doi.org/10.1016/j.jksuci.2023.101866

[13] Salman Muneer, Umer Farooq, Atifa Athar, Muhammad Ahsan Raza, Taher M Ghazal, and Shadman Sakib. 2024. A Critical Review of Artificial Intelligence Based Approaches in Intrusion Detection: A Comprehensive Analysis. *Journal of engineering* 2024 (04 2024), 1–16. https://doi.org/10.1155/2024/3909173

[14] Emad Ul Haq Qazi, Abdulrazaq Almorjan, and Tanveer Zia. 2022. A One-Dimensional Convolutional Neural Network (1D-CNN) Based Deep Learning System for Network Intrusion Detection. *Applied Sciences* 12 (08 2022), 7986. https://doi.org/10.3390/app12167986

[15] Mahbubur Rahman, Shaharia Al Shakil, and Mizanur Rahman Mustakim. 2025. A Survey on Intrusion Detection System in IoT Networks. *Cyber Security and Applications* 3 (12 2025), 100082. https://doi.org/10.1016/j.csa.2024.100082

[16] Vignesh Reddy, Sunitha R, M. Anusha, S Chaitra, and Abhilasha P Kumar. 2024. Artificial Intelligence Based Intrusion Detection Systems. *2024 4th International Conference on Mobile Networks and Wireless Communications (ICMNWC)* (12 2024), 1–6. https://doi.org/10.1109/icmnwc63764.2024.10872055

[17] Eric Anacleto Ribeiro. 2017. CICIDS2017: Cleaned Preprocessed. *Kaggle.com* (2017). https://doi.org/10412325/ab7509a66752b465911bf33c66d9b305

[18] Aya H. Salem, Safaa M. Azzam, O. E. Emam, and Amr A. Abohany. 2024. Advancing cybersecurity: a comprehensive review of AI-driven detection techniques. *Journal Of Big Data* 11 (08 2024), 1–38. https://doi.org/10.1186/s40537-024-00957-y

[19] Hannelore Sebestyen, Daniela Elena Popescu, and Rodica Doina Zmaranda. 2025. A Literature Review on Security in the Internet of Things: Identifying and Analysing Critical Categories. *Computers* 14 (02 2025), 61–61. https://doi.org/10.3390/computers14020061

[20] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. 2018. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. *Proceedings of the 4th International Conference on Information Systems Security and Privacy* (2018). https://doi.org/10.5220/0006639801080116

[21] Taveesh Sharma. 2023. Investigating optimal internet data collection in low resource networks. http://hdl.handle.net/11427/38141

[22] B S Sharmila and Rohini Nagapadma. 2023. Quantized autoencoder (QAE) intrusion detection system for anomaly detection in resource-constrained IoT devices using RT-IoT2022 dataset. *Cybersecurity* 6 (09 2023). https://doi.org/10.1186/s42400-023-00178-5

[23] Abdul Manan Sheikh, Md Rafiqul Islam, Mohamed Hadi Habaebi, Suriza Ahmad Zabidi, Athaur Rahman, and Adnan Kabbani. 2025. A survey on edge computing (EC) security challenges: Classification, threats, and mitigation strategies. *Future Internet* 17 (04 2025), 175–175. https://doi.org/10.3390/fi17040175

[24] Mahdi Soltani, Behzad Ousat, Mahdi Jafari Siavoshani, and Amir Hossein Jahangir. 2023. An adaptable deep learning-based intrusion detection system to zero-day attacks. *Journal of Information Security and Applications* 76 (2023), 103516.

[25] T. Sowmya and E.A. Mary Anita. 2023. A comprehensive review of AI based intrusion detection system. *ScienceDirect* 28 (06 2023), 100827–100827. https://doi.org/10.1016/j.measen.2023.100827

[26] Charles Stolz, Fuhao Li, and Jielun Zhang. 2024. Implementing Lightweight Intrusion Detection System on Resource Constrained Devices. *IEEE Internet of Things Journal* 9 (10 2024), 1–6. https://doi.org/10.1109/cars61786.2024.10778716

[27] Basant Subba, Santosh Biswas, and Sushanta Karmakar. 2016. Enhancing performance of anomaly based intrusion detection systems through dimensionality reduction using principal component analysis. *2016 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)* (11 2016). https://doi.org/10.1109/ants.2016.7947776

[28] Bin Sun and Yu Zhao. 2024. TinyNIDS: CNN-Based Network Intrusion Detection System on TinyML Models in 6G Environments. *Internet Technology Letters* (12 2024). https://doi.org/10.1002/itl2.629

[29] Miracle Udurume, Vladimir Shakhov, and Insoo Koo. 2024. Comparative Evaluation of Network-Based Intrusion Detection: Deep Learning vs Traditional Machine Learning Approach. *2024 Fifteenth International Conference on Ubiquitous and Future Networks (ICUFN)* (07 2024), 520–525. https://doi.org/10.1109/icufn61752.2024.10625037

[30] Patrick Vanin, Thomas Newe, Lubna Luxmi Dhirani, Eoin O'Connell, Donna O'Shea, Brian Lee, and Muzaffar Rao. 2022. A Study of Network Intrusion Detection Systems Using Artificial Intelligence/Machine Learning. *Applied Sciences* 12 (01 2022), 11752. https://doi.org/10.3390/app122211752

[31] Yuanyuan Wei, Julian Jang-Jaccard, Fariza Sabrina, Amardeep Singh, Wen Xu, and Seyit Camtepe. 2021. AE-MLP: A Hybrid Deep Learning Approach for DDoS Detection and Classification. *IEEE Access* 9 (2021), 146810–146821. https://doi.org/10.1109/ACCESS.2021.3123791

[32] Chuanlong Yin, Yuefei Zhu, Jinlong Fei, and Xinzheng He. 2017. A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks. *IEEE Access* 5 (10 2017), 21954–21961. https://doi.org/10.1109/access.2017.2762418

[33] Qihao Zhao, Fuwei Wang, Weimin Wang, Tianxin Zhang, Haodong Wu, and Weijun Ning. 2025. Research on intrusion detection model based on improved MLP algorithm. *Scientific Reports* 15 (02 2025). https://doi.org/10.1038/s41598-025-89798-0

[34] Ruijie Zhao, Guan Gui, Zhi Xue, Jie Yin, Tomoaki Ohtsuki, Bamidele Adebisi, and Haris Gacanin. 2021. A Novel Intrusion Detection Method Based on Lightweight Neural Network for Internet of Things. *IEEE Internet of Things Journal* 9 (2021), 9960–9972. https://doi.org/10.1109/jiot.2021.3119055

[35] Shengchu Zhao, Wei Li, Tanveer Zia, and Albert Y. Zomaya. 2017. A Dimension Reduction Model and Classifier for Anomaly-Based Intrusion Detection in Internet of Things. *2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech)* (11 2017). https://doi.org/10.1109/dasc-picom-datacom-cyberscitec.2017.141

A SUPPLEMENTARY INFORMATION

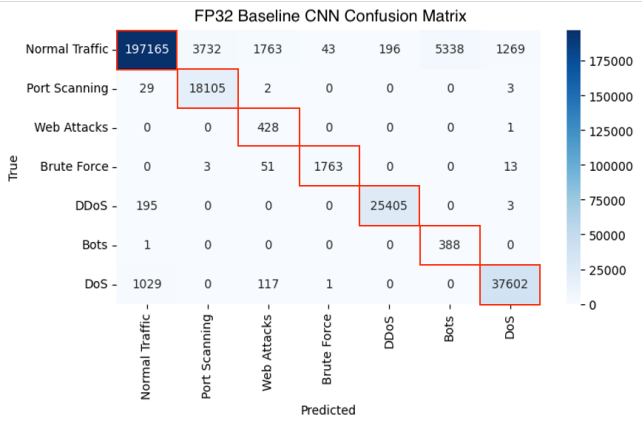

Figure 4: FP32 Baseline CNN CM

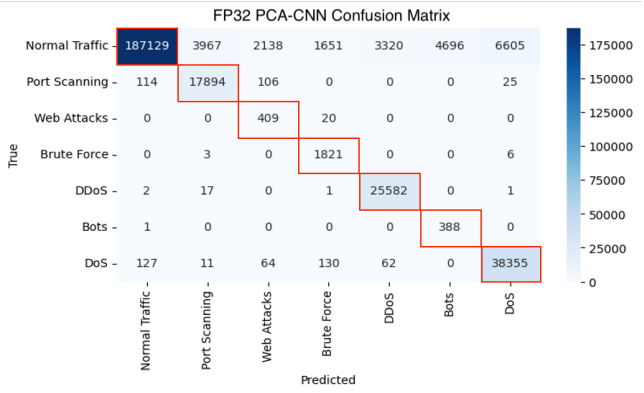

Figure 5: FP32 PCA-CNN CM

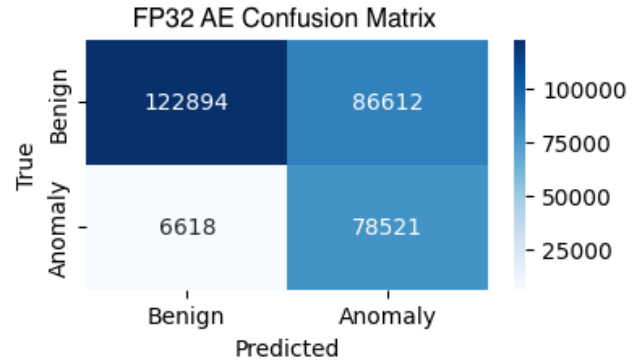

Figure 6: FP32 AE CM

Table 8: Extended AE-MLP Performance Results

Model	Accuracy (%)	Precision (%)	Recall (TPR) (%)	FPR (%)	FNR (%)
Autoencoder FP32	68.36	47.55	92.23	41.34	7.77
MLP FP32	94.95	90.53	99.83	9.47	0.17
AE-MLP FP32	**94.93**	**90.53**	**92.07**	**9.47**	**7.94**
Autoencoder FP16	68.36	47.55	92.22	41.33	7.78
MLP FP16	95.03	90.66	99.83	9.32	0.17
AE-MLP FP16	**94.97**	**90.66**	**92.06**	**9.34**	**7.94**
Autoencoder Dynamic INT8	51.25	36.71	94.91	66.50	5.09
MLP Dynamic INT8	93.75	88.10	95.92	7.51	4.08
AE-MLP Dynamic INT8	**93.86**	**88.10**	**91.04**	**11.90**	**8.96**
Autoencoder Full INT8	55.56	37.98	84.98	56.39	15.02
MLP Full INT8	92.36	95.41	83.92	2.47	16.08
AE-MLP Full INT8	**90.72**	**95.41**	**71.31**	**4.59**	**28.69**

Table 9: Extended Percentage of Each Malware Class Correctly Identified by the AE-MLP

Model	Port Scanning (%)	Web Attacks (%)	Brute Force (%)	DDoS (%)	Bots (%)	DoS (%)
Autoencoder	92.89	9.09	76.23	95.99	78.15	91.17
MLP	99.41	100.00	99.86	99.98	100.00	99.39
AE-MLP	**92.50**	**9.09**	**76.12**	**95.97**	**78.15**	**90.61**

Table 10: Extended Results of AE-MLP Model's Generalisation to Unseen Attack

Model	Dos Detection Rate (%)	Bots Detection Rate (%)
AE FP32	91.17	78.15
MLP FP32	70.20	77.30
AE-MLP FP32	**64.00**	**60.41**
AE FP16	91.17	78.15
MLP FP16	67.59	77.30
AE-MLP FP16	**61.62**	**60.41**
AE Dynamic INT8	95.33	95.37
MLP Dynamic INT8	64.69	63.61
AE-MLP Dynamic INT8	**61.67**	**60.67**
AE Full INT8	91.16	83.29
MLP Full INT8	64.17	58.95
AE-MLP Full INT8	**58.50**	**49.10**