Scientific Research Publishing

Journal of Computer and Communications, 2025, 13(9), 121-140
https://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

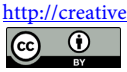
Cybersecurity Modeling: Application of Optimization in Machine Learning-Based Detection Systems

Gloria A. Odiaga*, Newton Masinde, Castro Yoga

School of Informatics and Innovative Systems, Jaramogi Oginga Odinga University of Science and Technology, Bondo, Kenya
Email: *gloriaodiaga@gmail.com, nmasinde@jooust.ac.ke, cyoga@jooust.ac.ke

How to cite this paper: Odiaga, G.A., Masinde, N. and Yoga, C. (2025) Cybersecurity Modeling: Application of Optimization in Machine Learning-Based Detection Systems. *Journal of Computer and Communications*, **13**, 121-140.
https://doi.org/10.4236/jcc.2025.139007

Received: July 31, 2025
Accepted: September 25, 2025
Published: September 28, 2025



Abstract

Mathematical optimization is a fundamental aspect of machine learning (ML). An ML task can be conceptualized as optimizing a specific objective using the training dataset to discern patterns and to predict and generalize on new, unseen data through testing. The reliability of optimization in addressing these tasks is fundamental to the efficacy of ML solutions, which has prompted the exploration of complex ML tasks with advanced mathematical approaches and substantially larger datasets. As dataset sizes grow, training complex models can take longer, even with high-performance hardware, qualifying the need for efficient optimization techniques and the performance enhancement of existing optimization methods. This study discusses optimization techniques and their application in ML and deep learning (DL). By framing the detection task as an optimization problem, the study proposes a systematic framework that includes mathematical modeling of the problem. The study also emphasizes the importance of selecting appropriate optimization methods across model development in architectural design, training, and tuning procedures, grounded in the mathematical modeling of a cyberattack detection task to ensure optimal performance in securing web-based systems. The experimental results on a Long Short-Term Memory (LSTM) model show an accuracy of 0.947, a False Negative (FN) rate of 0.053, and a False Positive (FP) rate of 0.011, hence demonstrating that integrating the proposed framework in cybersecurity model design can enhance the attack detection performance.

Keywords

Optimization, Cybersecurity, Machine Learning, Deep Learning, Constraint, Domain Set, Objective Function, Variable, Cyberattack

1. Introduction

Historically, optimization theory was introduced by Kantorovich in 1960, who advocated obtaining an optimum solution to a presented problem by applying a definite, scientifically based method. Optimization is a discipline that studies how actions are chosen to achieve objectives optimally [1]. Optimization delineates the methodology or process of improving a system's or network's efficacy or functionality [2]. Optimization generally entails the modification of variables to maximize or minimize a specific objective function while simultaneously balancing the constraints of the problem. DL is a specialized branch of ML that harnesses neural networks to autonomously analyze datasets, allowing computer systems to achieve refined performance through experience and continuous data exposure [3]. DL algorithms outperform traditional ML in cyberattack detection, leveraging their multi-layered structure and powerful extraction techniques to efficiently derive meaningful insights from training data [4]. Challenges with most deep learning algorithms include lengthy training periods, manual parameter optimization, and accuracy deficiencies in detection [4] [5].

Optimization is crucial in refining deep learning algorithms for improved efficiency and reliability for optimal performance. In cybersecurity, optimization techniques ensure that detection models can accurately identify threats while reducing false positives and computational overheads [6]. Optimization is valuable not because it seeks to understand every system detail, but because it identifies the most efficient way to refine and adjust it with minimal effort [7]. Optimization, in its simplest form, is about selecting inputs that will result in the best possible outputs of a given system. Optimization techniques help analyze the operational space and forecast the necessary adjustments to system parameters, ensuring peak performance [7]. The choice between stochastic and deterministic optimization methods relies on factors such as the demand for computational efficiency, the complexity of a problem, and the precision of available system data [8].

In stochastic optimization, the techniques applied lack predictability in the search process, such as through approximations. In contrast, deterministic optimization methods follow a fixed, predictable path to find a solution, relying on precise calculations of gradients or other problem-specific derivatives [9]. This integration of randomness generalizes stochastic optimization, as randomness can be incorporated using various mechanisms [8]. There are different classes of stochastic optimization algorithms, as indicated in the following.

1) *Simulated Annealing* (*SA*): This is a probabilistic technique modeled according to the annealing process in metallurgy, symbolizing the physical process of heating and cooling metals [10]. After identifying a search space, the SA algorithm comprehensively explores different combinations to find an optimal solution.

2) *Particle Swarm Optimization* (*PSO*): This draws inspiration from the collective dynamics of natural swarms, such as the flocking of birds, where the movement of particles is simulated within a search space to find an optimal solution

[11] [12].

3) *Game Theory-based optimization* (*GT*): GT-based methods seek to simulate the dynamics of a game with players pursuing individual goals, grounded in the principles of Nash's game theory [13] [14].

4) *Evolutionary Algorithms* (*EA*): EAs are techniques inspired by the replication of the evolutionary process as articulated in Darwin's theory of natural selection [15] [16]. Genetic Algorithms (GA) are a subset of EA that mimic natural evolution to achieve optimal solutions to complex problems through successive iterations of multiple potential solutions [17].

A key benefit of stochastic optimization is its ability to navigate beyond local minima and thoroughly explore the design space due to its inherent randomness [18]. The challenge is reconciling the comprehensive exploration of the solution space with the timely convergence to an optimal solution [19].

Algorithm tuning involves continuous adjustment of various control parameters [20], significantly influencing an algorithm's performance [21]. Stochastic optimization algorithms mainly draw inspiration from natural processes and adaptations to solve optimization [22]. The formulation of a specific optimization problem and its environment influence the performance of the applied optimization technique [23]. Stochastic optimization techniques generally offer advanced methods for solving optimization problems [24].

The optimization techniques proposed in our framework are well-grounded practices. However, their strategic integration for cyberattack detection, by aligning architectural design, training procedures, and tuning dynamics, presents a unified approach to cybersecurity modeling. In contrast to previous studies (discussed in Section 2), which concentrate exclusively on enhancing a single phase of the model life cycle or utilize generic parameter optimization, the framework in this study establishes a unified, mathematically defined optimization pipeline that fluidly incorporates feature selection, hyperparameter optimization, gradient-based training, and regularization for the CI-CIDS-2017 dataset and web-based attack profiles.

The justification for this integrated methodology is to address practical constraints related to class imbalance, convergence efficiency, and real-time detection rates in high-traffic scenarios. Comparative experiments (section 6.3) demonstrate that the fully integrated approach achieves a false positive rate of 1.1% and a false negative rate of 5.3%, representing measurable enhancements over a baseline LSTM model lacking staged optimization, which recorded false positive and false negative rates of 3.4% and 8.9%, respectively.

This paper is structured as follows: Section 2 presents the related studies on optimization in cyberattack detection. Section 3 discusses the general optimization problem modeling task. Section 4 highlights the mathematical modeling of the cyberattack detection task as an optimization problem. Section 5 draws attention to the experimental design, while Section 6 discusses the study results. Finally, Section 7 gives the conclusions of the study.

2. Related Studies

ML has been an optimal area for algorithm optimization, but it has also led to more research in optimization methods [25] [26]. ML focuses on making machines learn from experience. Learning involves using observations to search for an environment's model and predict new queries for a problem [26]. In the earlier days of ML, there was more emphasis on models catering to specifically identified input data structures due to low computational capabilities and limited data access, which allowed the design of models to specifically leverage data structures and tune them for tasks with small data amounts [27]. The merit was that it facilitated the models' comprehensive theoretical analysis and guaranteed the models' performance [28].

Using off-the-shelf algorithms to search for optimal models is possible, but new optimization models must be designed specifically for the particular problem, enhancing ML efficiency [29]. Alternatively, with improved computational capabilities and larger amounts of data, it has been possible to utilize more flexible and complex models, such as Deep Neural Networks (DNN). Regardless of complex models requiring more resources and data to fit appropriately to the data and problem, the advantage is that similar models may be adopted without modifying them for specific problems, thus they perform better in real-world problems.

Previous decades have witnessed a shift in how intelligent systems are designed [30]. Rather than mimicking human decision-making, machine design mimics the human learning process. Such a framework involves a model with variables from appropriate data specific to the problem or task. Popular optimization approaches are applicable in addressing the learning problem in the case of ML and DL [31]. Also, the adoption of generic optimization may be inefficient due to the problem's scale on the loss function and output space. To address a dataset's large size, optimization methods' stochastic variants that only work with a subset of the dataset at a time have been popularized [8]. Further, dealing with complexity in loss functions and output space associated with the learning problem calls for developing methods that leverage the structure respective to the task to enhance the optimization's efficiency. ML optimization encompasses a broader range of techniques, such as neural networks, support vector machines (SVM), linear and logistic regression [32].

In the context of cyberattack detection, several studies have utilized optimization, such as Brindha *et al.* [33], who applied optimized deep learning approaches; Do *et al.* [34], who studied optimization using ML; Injadat *et al.* [32], who researched a multi-stage optimized ML framework; Nayak *et al.* [35], who studied an ML and Bayesian optimization-driven intelligent framework for Internet of Medical Things (IoMT) cyberattack detection; and Zhang *et al.* [36], who focused on low-rate DoS attack detection using Power Spectral Density (PSD)-based entropy and ML. However, most of these works either lack a unified optimization framework or do not fully address practical constraints concerning hyper-parameter tuning, class imbalance, and real-time attack detection. This study, therefore,

aims to focus on these challenges by considering the optimization lifecycle of cyberattack detection under practical constraints.

3. The General Optimization Problem

The following discussion provides a general definition of the optimization problem. In a task requiring predicting a y variable that depends on x input, the application of ML essentially involves two computational problems. In the assumption as presented in Equation (1), that:

$$y = f(x;\theta) \tag{1}$$

where the f function maps input to output and has θ as the parameters, then the initial computation problem is the computation of the y output by evaluation of $f(x;\theta)$ at x input, which is called the inference or prediction problem.

Secondly, if given data D, parameters θ of function f may be estimated in Equation (2) as:

$$\theta^* = \arg\min_\theta L(f, D) \tag{2}$$

L is the loss function measuring the parameter configuration quality, which is called the learning problem. In general, the learning and inference problems may be seen as optimization problems in ML and DL. The inference problem involves choosing an output y from a set Y, maximizing a score estimating the prediction's quality as in Equation (3):

$$y^* = \arg\min_{y \in Y} \text{score}(x, y) \tag{3}$$

The Y domain differs for each task. For a classification problem, the set would be categorical, but for a regression task, it would be a continuous space. The domain set may be sophisticated, such as the possible rankings of a set in ranked retrieval tasks. Whereas deploying off-the-shelf optimization models to solve inference problems is possible, other cases require the development of new optimization models to leverage the problem's structure to solve it effectively.

For the learning problem in a standard setting, according to Equation (4), parameters θ of the model $f(x;\theta)$ are selected since the loss function's expected value $L(f(x;\theta), y)$ is minimized by the model over the data distribution P_{data}.

$$\theta^* = \arg\min_\theta E_{(x,y)\sim P_{data}} \left[L(f(x;\theta), y) \right] \tag{4}$$

The risk is the objective of the optimization problem [28], and in computing, this requires knowledge of the data distribution P_{data}, which is generally not known. Hence, an empirical estimate is optimized as the loss function expectation on the training data D_{train} known as empirical risk, as in Equation (5).

$$\theta^* = \arg\min_\theta \frac{1}{|D_{train}|} \sum_{(x,y)\in D_{train}} L(f(x;\theta), y) \tag{5}$$

Whereas the training set is only used for model training, it is surmised that it

performs efficiently on unseen data in the selected distribution. A model's overall performance may be evaluated using the testing data, D_{test}, to determine risk. A learning algorithm that overfits the training set model will result in negative overall performance. However, overfitting can be addressed by adding additional regularization to the equation's objective function.

The relationship between an independent and dependent variable, x and y, respectively, such as in linear regression, with n observations in a given dataset, $\{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$, aims to identify the parameters w as in Equation (6) where:

$$(w, x_i) = y_i, \forall i \tag{6}$$

The inner product of w and x is represented by (w, x). If the relationship between x and y is non-linear, then the problem may be unsolvable. This can be avoided by getting the parameters of w as close as possible to y (average), as in Equation (7):

$$\min_w f(w) = \frac{1}{n}\sum_{i=1}^{n}\left((w, x_i) - y_i\right)^2 \tag{7}$$

Designing ML tasks as optimization problems can be done, as in Equation (8), in the following form:

$$\min_w f(w) = \frac{1}{n}\sum_{i-1}^{n} f_i(w; x_i) \tag{8}$$

where $\{x_i\}$ is data from an unknown probability distribution p. In ML and DL systems, each term $f_i(w; x_i)$ shows how a model with parameters w fits a particular observation x_i. With a dataset D_a having n samples $\{x_i\}$, $f(w)$ identifies how well the model fits the entire corpus of data on average. This is an empirical risk minimization problem, which is an estimate of the true problem to be solved, *i.e.*, the expected risk minimization problem: $\min_w \mathbb{E}_{x_i \sim p}\left[f_i(w; x_i)\right]$. Generally, without information on the distribution p to solve a risk minimization problem, we can solve Equation (8) alternatively.

If the objective is to make a prediction based on an input, each data sample x in the dataset D_a has a corresponding label $y = C(x)$, for an unknown labeling function C. In this case, a training pair refers to the tuple (x, y), and x is considered to be a d-dimensional vector with $x \in \mathbb{R}^d$, unless specified otherwise.

ML and DL models use objective functions formulated in Equation (8). Logistic regression, which is useful for binary classification, uses the objective function as in Equation (9):

$$f_i(w) = \log\left(1 + \exp\left(-y_i(w, x_i)\right)\right) \tag{9}$$

where y_i represents, across n observations, an averaged binary label (±1).

The framing of cyberattack detection systems as optimization problems targeting accuracy, efficiency, and robustness is necessary for optimal performance. Hence, by formulating the detection task as an optimization problem, the subse-

quent mathematical modelling enables the development of a structured and systematic framework for model development. The following is the definition of the optimization problem.

4. The Cyberattack Detection Optimization Problem

This section provides a mathematical definition of the cyberattack detection optimization problem. It begins by first defining key terms and variables that will be used in formulating the cyberattack optimization problem, before giving a formal definition of the problem.

4.1. Preliminary Definitions

The objective problem comprises variables, an objective function, and constraints defined over a particular domain. These elements are expressed in our cyberattack detection task as follows.

- *Constraints*, C_{ineq} *and* C_{eq}: The boundaries on hyperparameters, such as the learning rate, dropout rate, and batch size, ensure steady and effective training. These comprise the inequality and equality constraints, denoted as C_{ineq} and C_{eq}, respectively.
- *Dependent variable*, D_v: A binary attack label $D_v \in \{0,1\}$, that denotes the absence or presence of an attack.
- *Domain set*, D: The web-based system cyberattacks, including DoS/DDoS, brute force, SQL injection, CSRF, and XSS, are recorded in the Canadian Institute for Cybersecurity Intrusion Detection System (CIC-IDS) 2017 dataset [37].
- *Independent variables*, I_n: Seventeen network traffic characteristics were selected through Boruta on top of the CIC-IDS 2017 dataset, and are categorized based on feature importance into eight features: specifically, TCP flag counts, flow duration, total packets, flow bytes/s, down/up ratio, window bytes, mean packet length, and mean inter-arrival time (IAT).
- *Model parameters*, P: These are adjusted during training to influence how input is translated to output, which helps to make accurate predictions and minimize errors.
- *Objective functions*: To maximize attack detection rate, R_a, minimize FP (false alarms), and minimize FN (undetected attacks), with further conditions regarding computational efficiency.
- *Weight coefficients*, α *and* β: These are crucial in balancing the FP and FN values.

These are summarized in **Table 1**.

4.2. Formulation of Cyberattack Detection as an Optimization Problem

To optimize DL models, we present the mathematical formulation of the cyberattack detection task in Equations (10)-(13).

$$\max_{\{P \in D\}} = \left[R_a \left(P; D_v, I_1, I_2, \cdots, I_8 \right) - \alpha \cdot FP\left(P\right) - \beta \cdot FN\left(P\right) \right] \tag{10}$$

Table 1. Summary of terms used.

	Term	Description
1.	C_{ineq}	Inequality constraint
2.	C_{eq}	Equality constraint
3.	$D_v \in \{0, 1\}$	Dependent variable
4.	I_n	Independent variable ($n = 1, 2, 3, \cdots, 8$)
5.	D	Domain set
6.	P	Model parameters
7.	R_a	Attack detection rate
8.	FP	False Positives (false alarms)
9.	FN	False Negatives (undetected attacks)
10.	a, β	Weight coefficients

Subject to:

$$C_{eq}\left(D_v, I_1, I_2, \cdots, I_8; P\right) = 0 \tag{11}$$

$$C_{ineq}\left(D_v, I_1, I_2, \cdots, I_8; P\right) \le 0 \tag{12}$$

$$D_v \in \{0,1\} \tag{13}$$

Figure 1 is a visual showing the optimization function. The explicit formulation of the generic optimization theory for the intrusion detection problem guarantees that the framework is mathematically sound and, in addition, operationally sound.

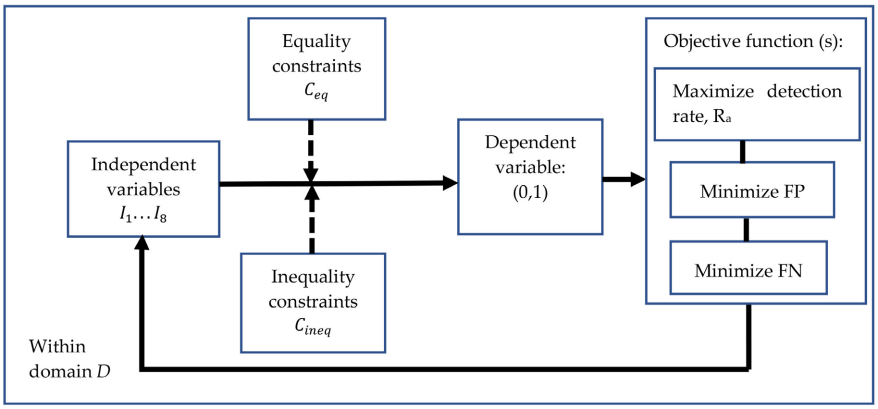


Figure 1. Formulation of the optimization problem.

5. Experimental Design

Mathematical optimization techniques are used to find optimal values for the deep learning-based model parameters, *P*.

5.1. Experimental Environment

The experimental environment specifications for testing the proposed framework for the LSTM model were as follows: a v2-8 TPU, an NVIDIA Tesla K80 GPU with 12 GB of RAM, and an Intel(R) Xeon(R) CPU running at 2.20 GHz. Boruta

is used for the feature selection task on the CICIDS-2017 dataset.

The rationale for the CICIDS-2017 dataset selection is the complex attacks it contains; hence, the dataset is representative of real-world network traffic [37]. The dataset comprises 79 features with 15 class labels spanning eight files of network traffic captured over five days. This ensures a complete network infrastructure crucial for cybersecurity model training for attack detection tasks.

5.2. Test Model

In DL, neural networks such as Long Short-Term Memory (LSTM), Recurrent Neural Network (RNN), or DNN are trained for the classification task. This study employs the LSTM algorithm for training the optimized cyber-attack detection system and for the baseline experiment. An inherent characteristic of LSTM is its superior ability to memorize and comprehend long-term sequences, which is particularly crucial given network data's sequential and time-based nature. Conversely, LSTM is a specialized architectural variation of RNN [38] developed to address challenges such as vanishing and exploding gradients. It exhibits exceptional sensitivity in detecting recurring patterns over brief and extended periods, making it stand out in detecting intricate and time-consuming multi-step cyber threats.

5.3. Model Setup

The key stages where optimal measures are essential are feature selection, hyperparameter tuning, model training, and validation. Optimization helps address various challenges in cyberattack detection where data is continuously generated in real-time environments. These include long training times, poor generalization speeds, model size complexities, and achieving continuous model learning over time. By optimizing the training algorithm, DL models can learn more effectively from the data and better capture the underlying patterns of attacks.

Figure 2 provides a summary of the optimization techniques applied in the study.

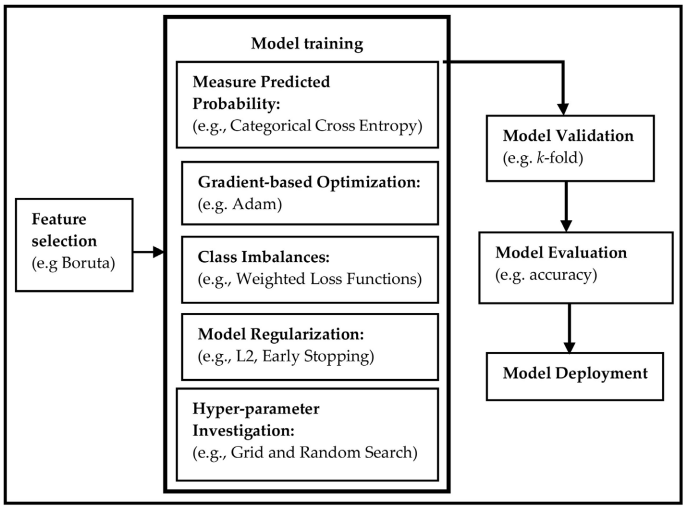


Figure 2. Framework for application of the optimization techniques.

5.3.1. Feature Selection

The model's performance will be enhanced by selecting pertinent features and excluding irrelevant or superfluous ones [13]. Optimization iteratively evaluates different feature subsets based on predefined criteria until an optimal solution is found, such as with the Boruta algorithm. Boruta reduces the features to 17 from 79 in the complete feature set, offering dimensionality reduction.

5.3.2. Hyperparameter Tuning

The model does not learn hyperparameters during training; attributes are set beforehand and influence learning [20]. Grid and random search techniques thoroughly explore the hyper-parameter space and identify the configurations with the best outcomes [28] [32]. Grid search is a brute-force method that tests various combinations of hyperparameters, while random search uses random combinations from predefined hyperparameter ranges [18] [20] [28].

The model can achieve higher accuracy and generalization ability following hyper-parameter tuning, where the best values for different model parameters, including batch size, number of neural network layers, learning rate, and number of units per layer, are identified. We classify the hyperparameters into compilation, architectural, and training categories. The compilation category defines how the model is compiled, the architectural category defines the structure of the DL-model, and the training category affects the model's training process, as summarized in Table 2.

Table 2. Hyperparameter configurations.

No	Hyperparameter	Value	Category
1	Batch size	64	Training
2	Learning rate	0.001	Training
3	Epochs	250	Training
4	L2 penalty factor	0.01	Training
5	Early stopping patience	10	Training
6	Optimizer	Adam	Compilation
7	Loss function	Categorical cross entropy	Compilation
8	Metric	Accuracy	Compilation
9	Output layer activation	Softmax	Architectural
10	LSTM layers	2 + 1 dense layers for output	Architectural
11	Units per layer	50	Architectural

5.3.3. Model Training

During the training phase, optimization is leveraged to improve the training speed and stability of the model. Gradient-based optimization algorithms, such as stochastic gradient descent (SGD), Adaptive Moment Estimation (Adam), or Root

Mean Square Propagation (RMSprop), are used to update the model parameters iteratively [32]. Learning rate scheduling and momentum adaptation also fine-tune the optimization process and prevent vanishing or exploding gradients.

Activation functions control how the output of each layer is transformed before passing it to the next layer in neural networks [32] [39]. These include the Rectified Linear Unit (ReLU), Softmax, and the hyperbolic tangent function (Tanh). Softmax is used in the output layer for multi-class classification tasks to convert raw output scores from the model into probabilities, which sum to 1 [32]. This makes it suitable for interpreting the model's predictions and enables the model to express its confidence in each class. Tanh maps outputs between -1 and 1 to ensure zero-centered outputs and adds non-linearity for efficient model training [32] [39]. ReLU is simple, fast, and can minimize the vanishing gradient problems in DL models [39].

This study employs the categorical cross-entropy loss function, ensuring the model appropriately differentiates between normal and attack data over time. The loss function measures how well the predictions from the model align with the actual data values [6]. It is a guiding metric during training, allowing the model to update its parameters or weights to reduce differences between predicted and actual values. It is appropriate for classification problems like intrusion detection systems [6]. It affects the learning process and the model's performance through error reduction.

The Adam optimizer blends the advantages of the adaptive learning rate and momentum techniques during training to ensure correct model generalization [32]. It uses estimates of the first (mean) and second gradient (variance) moments, ensuring robustness in training DL algorithms. Adam ensures that the neural network learns efficiently [38].

Another optimization task is class imbalance handling. Class imbalance is common with cybersecurity datasets, as most datasets have benign data and a smaller portion consists of malicious data [33]. The class-weighted loss function is applied to counter class imbalance problems. It ensures that optimizing the loss function for imbalanced data is addressed using class weights [33]. Assigning larger weight values to the marginal class (attacks) is crucial to ensure attack misclassifications are heavily penalized.

Regularization techniques are applied to avoid model overfitting and generalization problems, specifically early stopping and L2 regularization. In the early stopping technique, model training is stopped as soon as the performance weakens. This is significant as it prevents overfitting the model to the training data. In L2 regularization, large weights are penalized by attaching a penalty factor to the loss function. Consequently, smaller weights are used, which avoids model overfitting. Optimizers, activation functions, and regularization techniques collectively affect the performance of a model [40]. **Table 3** shows a summary of the concepts applied in the optimization of the DL-based model development for cyberattack detection in web-based systems.

Table 3. Optimization concepts employed.

	Concept	Technique
1	Hyper-parameter exploration	Grid search, random search
2	Activation functions	ReLU, Softmax, Tanh
3	Regularization	L2, early stopping
4	Gradient-based optimization	Adam
5	Loss functions	Categorical cross-entropy, Weighted

5.3.4. Model Validation

The data validation set is used to tune the hyperparameters, hence avoiding generalization and overfitting issues. Validation ensures the model reliably functions on training and validation data [41] [42]. Training loss is calculated on the training data after each epoch, while the validation loss is calculated on the validation data after each epoch [41]. If the validation loss is higher than the training loss, it may be overfitting to the training data, signifying that it memorizes patterns rather than learning generalization.

Cross-validation methods are also extensively used in intrusion detection system research as they lower computing time while maintaining the performance of the algorithms [41] [42]. Generally, the validation loss and accuracy results (discussed in Section 6) reiterate the optimization process's success (or lack thereof) and subsequent model performance. Table 4 provides a summary of techniques used and their relevance to the optimization process of the models.

Table 4. Summary of techniques and their relevance to optimization.

	Technique	Relevance
1.	Early stopping, L2 regularization	Enhance model generalization.
2.	Boruta	Feature selection.
3.	Grid search, random search	Hyper-parameter optimization.
4.	Hyper-parameter tuning	Identify the best values for different model parameters.
5.	Adam optimizer	Adjust the learning rate for efficient model learning.
6.	ReLU activation function	Handles vanishing gradient problems.
7.	Tanh activation function	Adds non-linearity, which is essential for learning patterns.
8.	Softmax activation function	Generates probability distributions for efficient training.
9.	Weighted loss function	Handling class imbalances.
10.	Categorical cross-entropy loss function	Error reduction.

5.4. Baseline Experiment

A baseline experiment without integrated optimization (WIO) was carried out us-

ing the same LSTM architecture without Boruta feature selection and class imbalance handling, and with manual hyper-parameter tuning, to measure the advantages of the suggested integrated optimization pipeline. The baseline employed default Adam optimizer settings, all 78 features, and 1 for the label, from the CICIDS-2017 dataset, and no regularization methods other than a dropout rate of 0.2.

6. Results and Discussion

The results of the experimental work follow.

6.1. Results for the Baseline LSTM Experiment WIO

The accuracy and loss values over 390 training epochs indicate a gradual increase in the training accuracy to 87.4%, and the validation accuracy peaks at 85%. The training loss decreases by 0.4 from 1.7, while the validation loss reduces to 0.5 from 1.8. The recurring increase in validation loss with reducing training loss over the epochs indicates overfitting. Without optimization, the model will eventually start memorizing the training data at a high rate, causing it to fail in generalization. These results depict poor model learning, increased overfitting instances, a lack of convergence control, and unstable training. Figure 3 shows the accuracy and loss values over the training cycle.

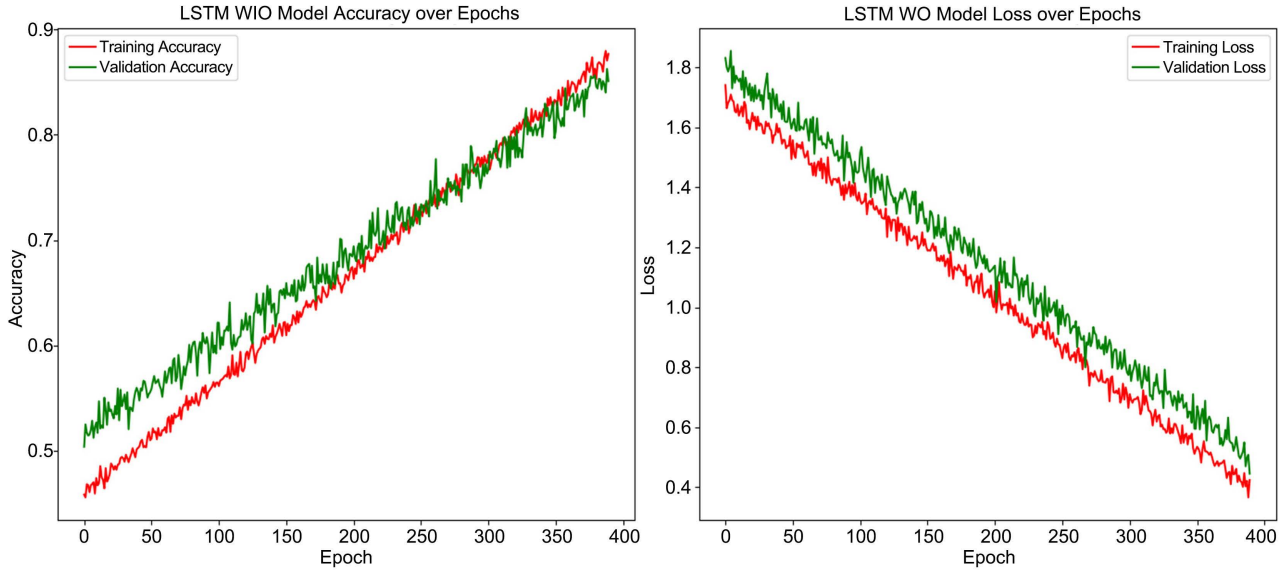


Figure 3. LSTM baseline WIO accuracy and loss over epochs.

Table 5. LSTM baseline WIO performance metrics.

Metric	Percentage	Relevance
FP	3.4	High benign misclassifications
FN	8.9	High rates of missed attacks
Accuracy	87.8	Overall correctness for all predictions

As presented in Table 5, the model achieves accuracy, FP, and FN scores of 87.8%, 3.4%, and 8.9%, respectively. These FP and FN values demonstrate higher cases of misclassifications and missed attacks, which are not ideal for cybersecurity applications.

6.2. Results for LSTM with Optimization

The accuracy and loss values are tracked over 250 training epochs. The training accuracy gradually improves to 94.8%, while the validation accuracy peaks at 93%. The training loss decreases from 1.2 to 0.2, while the validation loss starts at 1.3 and reduces to 0.3. This demonstrates effective model learning, as depicted in Figure 4.

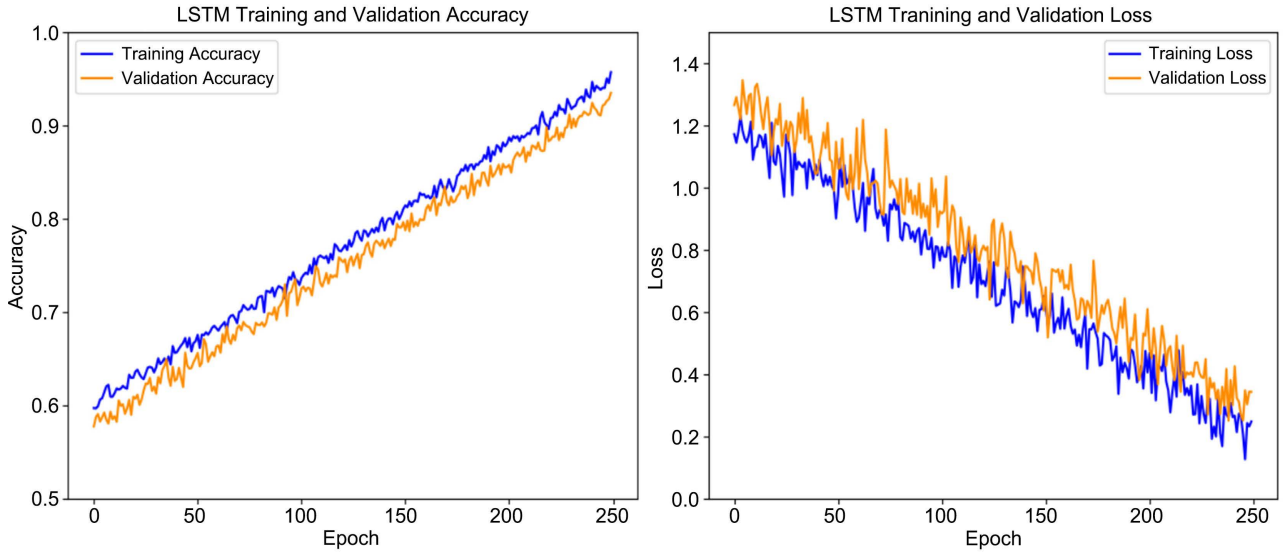


Figure 4. LSTM training and validation accuracy and loss.

The results of the experiment, presented in Table 6, show an accuracy score of 94.7%, an FP rate of 1.1%, and an FN rate of 5.3%. These values are optimal and crucial in cybersecurity to ensure reliable attack detection while minimizing false alarms and missed detections.

Table 6. Performance metrics for the optimized LSTM model.

Metric	Percentage	Relevance
FP	1.1	Low benign misclassifications
FN	5.3	Low missed attacks
Accuracy	94.7	Overall correctness for all predictions

6.3. Performance Comparison of the Baseline WIO and the Optimized Framework

The optimized LSTM framework outperforms the Baseline LSTM without integrated optimization (WIO). The FP rate of 1.1% indicates that only 1.1% of the

benign instances are incorrectly flagged as malicious. This low FP rate demonstrates the high specificity of the optimized LSTM model. A small FP rate can still be disruptive in high-traffic networks; therefore, continuous tuning is necessary. A high FP rate, such as 3.4% in the LSTM WIO model, results in excessive false alerts that can lead to system analyst fatigue, wasted resources, ignored alerts, disruption of operations, and diminished trust in the detection models.

Conversely, the FN rate indicates malicious instances are wrongly classified as benign. FN is critical as it demonstrates the ability to spot malicious traffic. The achieved FN rate in the optimized LSTM model signifies that only 5.3% of the actual attacks are missed. This relatively low rate of missed attacks demonstrates robust performance in identifying malicious traffic. A higher FN value, as in the LSTM WIO model at 8.9%, can potentially lead to data breaches. A severe missed attack, even with a low FN rate, can be detrimental in cybersecurity. A low FN rate ensures resiliency against different malicious vectors.

An accuracy of 94.7% implies that the optimized LSTM model correctly predicted 94.7% of all instances, whether normal or malicious, showing strong overall performance in identifying patterns in varied network scenarios compared to 87.8% for the LSTM WIO model.

Overall, there is an improvement in the accuracy, FP, and FN scores by 6.9%, 67.6%, and 40.4%, respectively. The comparison of the results is shown in Table 7.

Table 7. Results comparison.

Metric	Baseline LSTM (WIO)	Optimized LSTM	Improvement
Accuracy (%)	87.8	94.7	+6.9%
FP Rate (%)	3.4	1.1	+67.6%
FN Rate (%)	8.9	5.3	+40.4%
Training Epochs to Converge	390	250	+35.9%
Avg. Training Time/Epoch(s)	42	26	+38.1%

6.4. Scalability and Computational Performance

Beyond detection accuracy, a cyberattack detection system's practicality rests on its capacity to handle high traffic volumes with low latency.

- *Training Performance*: The average training time per epoch dropped from 42 seconds (complete feature set) to 26 seconds, a 38.1% improvement as presented in Table 7. Boruta-based dimensionality reduction decreases the feature set from 78 to 17. The Adam optimizer and early stopping criteria are instrumental in shortening the total training time. These further reduced the number of training epochs needed for convergence from 390 (WIO baseline) to 250, indicating a 35.9% improvement.
- *Inference Performance*: The optimized model processed about 137,000 flows per minute on the evaluation hardware, with an average inference latency of 7.3 milliseconds per network flow.

- *Memory Footprint*: The final trained LSTM model took up 22 MB, which allows for deployment in edge devices with limited resources.

Component contributions to scalability include:

1) Boruta feature selection, which reduces input dimensionality and speeds up preprocessing and training, is one component that contributes to scalability.

2) The Adam optimizer with momentum adaptation converges quickly without sacrificing accuracy.

3) Early stopping, after optimal generalization is attained, avoids wasting training cycles.

4) To address class imbalance, weighted loss functions prevent the need for repeated retraining.

These improvements allow for highly accurate, near-real-time threat detection in production networks.

7. Conclusion

This paper presents a comprehensive optimization-driven framework for cyberattack detection, offering extensible theoretical and practical insights for cybersecurity modelling and a foundation for future research and actual deployment in security environments. The framework transforms standard optimization components into a domain-specific, DL-based system, validated across multiple threat scenarios in the CIC-IDS 2017 dataset, offering an intelligent solution for cyber defense. Deploying the optimization framework across architectural design, training techniques, and hyper-parameter tuning highlights its efficiency and robustness in model development. Extensions of this framework to other detection systems and the application of ensemble and reinforcement learning for dynamic defense strategies lay the groundwork for future research.

Author Contributions

Conceptualization, G.A.O. and N.M.; methodology, G.A.O. and C.Y.; software, G.A.O. and C.Y.; validation, G.A.O., N.M. and C.Y.; formal analysis, G.A.O. and N.M.; investigation, G.A.O. and N.M.; resources, G.A.O.; data curation, G.A.O. and C.Y.; writing—original draft preparation, G.A.O.; writing—review and editing, G.A.O., N.M. and C.Y.; visualization, G.A.O. and C.Y.; supervision, N.M. and C.Y.; project administration, N.M. and C.Y. All authors have read and agreed to the published version of the manuscript.

Data Availability Statement

The CIC-IDS 2017 dataset used in the research is available at: https://www.kaggle.com/datasets/amanverma1999/a-comprehensive-dataset-for-ddos-attack.

Acknowledgements

We thank the members of the School of Informatics and Innovative Systems at

Jaramogi Oginga Odinga University of Science and Technology for their valuable insights during the research.

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] Kutateladze, S.S. (2011) Mathematics and Economics of Leonid Kantorovich.

[2] Mohammadi, A. and Sheikholeslam, F. (2023) Intelligent Optimization: Literature Review and State-of-the-Art Algorithms (1965-2022). *Engineering Applications of Artificial Intelligence*, **126**, Article 106959. https://doi.org/10.1016/j.engappai.2023.106959

[3] Zhang, J., Pan, L., Han, Q., Chen, C., Wen, S. and Xiang, Y. (2022) Deep Learning Based Attack Detection for Cyber-Physical System Cybersecurity: A Survey. *IEEE/CAA Journal of Automatica Sinica*, **9**, 377-391. https://doi.org/10.1109/jas.2021.1004261

[4] Zieni, R., Massari, L. and Calzarossa, M.C. (2023) Phishing or Not Phishing? A Survey on the Detection of Phishing Websites. *IEEE Access*, **11**, 18499-18519. https://doi.org/10.1109/access.2023.3247135

[5] Do, N.Q., Selamat, A., Krejcar, O., Herrera-Viedma, E. and Fujita, H. (2022) Deep Learning for Phishing Detection: Taxonomy, Current Challenges and Future Directions. *IEEE Access*, **10**, 36429-36463. https://doi.org/10.1109/access.2022.3151903

[6] Mao, A.Q., Mohri, M. and Zhong, Y.T. (2023) Cross-Entropy Loss Functions: Theoretical Analysis and Applications. *Proceedings of the* 40*th International Conference on Machine Learning* (*ICML* 2023), PMLR, Honolulu 23803-23828. https://doi.org/10.48550/arXiv.2304.07288

[7] Adby, P. (2013) Introduction to Optimization Methods. Springer Science & Business Media.

[8] Lan, G. (2020) Convex Optimization Theory. In: *First-Order and Stochastic Optimization Methods for Machine Learning*, Springer, 21-51.

[9] Erten, H.I., Deveci, H.A. and Artem, H.S. (2020) Stochastic Optimization Methods. In: *Designing Engineering Structures Using Stochastic Optimization Methods*, CRC Press, 10-23. https://doi.org/10.1201/9780429289576-2

[10] Guilmeau, T., Chouzenoux, E. and Elvira, V. (2021) Simulated Annealing: A Review and a New Scheme. 2021 *IEEE Statistical Signal Processing Workshop* (*SSP*), Rio de Janeiro, 11-14 July 2021, 101-105. https://doi.org/10.1109/ssp49050.2021.9513782

[11] Gad, A.G. (2022) Particle Swarm Optimization Algorithm and Its Applications: A Systematic Review. *Archives of Computational Methods in Engineering*, **29**, 2531-2561. https://doi.org/10.1007/s11831-021-09694-4

[12] Shami, T.M., El-Saleh, A.A., Alswaitti, M., Al-Tashi, Q., Summakieh, M.A. and Mirjalili, S. (2022) Particle Swarm Optimization: A Comprehensive Survey. *IEEE Access*, **10**, 10031-10061. https://doi.org/10.1109/access.2022.3142859

[13] Sohrabi, M.K. and Azgomi, H. (2020) A Survey on the Combined Use of Optimization Methods and Game Theory. *Archives of Computational Methods in Engineering*, **27**, 59-80. https://doi.org/10.1007/s11831-018-9300-5

[14] Rajeswaran, A., Mordatch, I. and Kumar, V. (2020) A Game Theoretic Framework for Model Based Reinforcement Learning.

[15] Slowik, A. and Kwasnicka, H. (2020) Evolutionary Algorithms and Their Applications to Engineering Problems. *Neural Computing and Applications*, **32**, 12363-12379. https://doi.org/10.1007/s00521-020-04832-8

[16] Deng, W., Shang, S., Cai, X., Zhao, H., Song, Y. and Xu, J. (2021) An Improved Differential Evolution Algorithm and Its Application in Optimization Problem. *Soft Computing*, **25**, 5277-5298. https://doi.org/10.1007/s00500-020-05527-x

[17] Sun, R. (2020) Optimization for Deep Learning: An Overview. *Journal of the Operations Research Society of China*, **8**, 249-294. https://doi.org/10.1007/s40305-020-00309-6

[18] Mendler-Dünner, C., Perdomo, J., Zrnic, T. and Hardt, M. (2020) Stochastic Optimization for Performative Prediction. In: *Advances in Neural Information Processing Systems*, Curran Associates, 4929-4939.

[19] Drusvyatskiy, D. and Xiao, L. (2023) Stochastic Optimization with Decision-Dependent Distributions. *Mathematics of Operations Research*, **48**, 954-998. https://doi.org/10.1287/moor.2022.1287

[20] Curtis, F.E. and Scheinberg, K. (2020) Adaptive Stochastic Optimization: A Framework for Analyzing Stochastic Optimization Algorithms. *IEEE Signal Processing Magazine*, **37**, 32-42. https://doi.org/10.1109/msp.2020.3003539

[21] Li, B. and Huang, T. (2022) Control Variable Parameterization and Optimization Method for Stochastic Linear Quadratic Models. *Chaos, Solitons & Fractals*, **154**, Article 111638. https://doi.org/10.1016/j.chaos.2021.111638

[22] Piotrowski, R., Wonia, M. and Wonia, A. (2023) Stochastic Optimisation Algorithm for Optimisation of Controller Parameters for Control of Dissolved Oxygen in Wastewater Treatment Plant. *Journal of Water Process Engineering*, **51**, Article 103357. https://doi.org/10.1016/j.jwpe.2022.103357

[23] Li, C. and Grossmann, I.E. (2021) A Review of Stochastic Programming Methods for Optimization of Process Systems under Uncertainty. *Frontiers in Chemical Engineering*, **2**, Article 622241. https://doi.org/10.3389/fceng.2020.622241

[24] Khaled, A. and Jin, C. (2024) Tuning-Free Stochastic Optimization. https://doi.org/10.48550/arXiv.2402.07793

[25] Karimi-Mamaghan, M., Mohammadi, M., Meyer, P., Karimi-Mamaghan, A.M. and Talbi, E. (2022) Machine Learning at the Service of Meta-Heuristics for Solving Combinatorial Optimization Problems: A State-of-the-Art. *European Journal of Operational Research*, **296**, 393-422. https://doi.org/10.1016/j.ejor.2021.04.032

[26] Gambella, C., Ghaddar, B. and Naoum-Sawaya, J. (2021) Optimization Problems for Machine Learning: A Survey. *European Journal of Operational Research*, **290**, 807-828. https://doi.org/10.1016/j.ejor.2020.08.045

[27] Pang, G., Shen, C., Cao, L. and Hengel, A.V.D. (2022) Deep Learning for Anomaly Detection: A Review. *ACM Computing Surveys*, **54**, 1-38. https://doi.org/10.1145/3439950

[28] Brownlee, J. (2021) Optimization for Machine Learning. In: *Machine Learning Mastery*, Crater Publication, 8-25.

[29] Chen, J., Touati, C. and Zhu, Q. (2021) A Dynamic Game Approach to Designing Secure Interdependent IoT-Enabled Infrastructure Network. *IEEE Transactions on Network Science and Engineering*, **8**, 2601-2612. https://doi.org/10.1109/tnse.2021.3100801

[30] Knobelreiter, P., Sormann, C., Shekhovtsov, A., Fraundorfer, F. and Pock, T. (2020) Belief Propagation Reloaded: Learning Bp-Layers for Labeling Problems. 2020 *IEEE/CVF*

Conference on Computer Vision and Pattern Recognition (*CVPR*), Seattle, 13-19 June 2020, 7897-7906. https://doi.org/10.1109/cvpr42600.2020.00792

[31] Yong, J. (2018) Optimization Theory: A Concise Introduction. World Scientific.

[32] Injadat, M., Moubayed, A., Nassif, A.B. and Shami, A. (2021) Multi-Stage Optimized Machine Learning Framework for Network Intrusion Detection. *IEEE Transactions on Network and Service Management*, **18**, 1803-1816. https://doi.org/10.1109/tnsm.2020.3014929

[33] Brindha Devi, V., Ranjan, N.M. and Sharma, H. (2022) IoT Attack Detection and Mitigation with Optimized Deep Learning Techniques. *Cybernetics and Systems*, **55**, 1702-1728. https://doi.org/10.1080/01969722.2022.2145660

[34] Do, C., Dam, N.Q. and Lam, N.T. (2021) Optimization of Network Traffic Anomaly Detection Using Machine Learning. *International Journal of Electrical and Computer Engineering*, **11**, 2360-2370. https://doi.org/10.11591/ijece.v11i3.pp2360-2370

[35] Nayak, J., Meher, S.K., Souri, A., Naik, B. and Vimal, S. (2022) Extreme Learning Machine and Bayesian Optimization-Driven Intelligent Framework for IoMT Cyber-Attack Detection. *The Journal of Supercomputing*, **78**, 14866-14891. https://doi.org/10.1007/s11227-022-04453-z

[36] Zhang, Y., Li, P. and Wang, X. (2019) Intrusion Detection for IoT Based on Improved Genetic Algorithm and Deep Belief Network. *IEEE Access*, **7**, 31711-31722. https://doi.org/10.1109/access.2019.2903723

[37] Boukhamla, A. and Gaviro, J.C. (2021) CICIDS2017 Dataset: Performance Improvements and Validation as a Robust Intrusion Detection System Testbed. *International Journal of Information and Computer Security*, **16**, 20-32. https://doi.org/10.1504/ijics.2021.117392

[38] Ahsan, M., Rifat, N., Chowdhury, M. and Gomes, R. (2022). Intrusion Detection for IoT Network Security with Deep Neural Network. 2022 *IEEE International Conference on Electro Information Technology* (*eIT*), Mankat, 19-21 May 2022, 467-472. https://doi.org/10.1109/eit53891.2022.9814006

[39] Karthik, M.G. and Krishnan, M.B.M. (2021) Hybrid Random Forest and Synthetic Minority over Sampling Technique for Detecting Internet of Things Attacks. *Journal of Ambient Intelligence and Humanized Computing*. https://doi.org/10.1007/s12652-021-03082-3

[40] Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A. and Talwalkar, A. (2020) Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. *Journal of Machine Learning Research*, **18**, 1-52.

[41] Marcot, B.G. and Hanea, A.M. (2021) What Is an Optimal Value of K in K-Fold Cross-Validation in Discrete Bayesian Network Analysis? *Computational Statistics*, **36**, 2009-2031. https://doi.org/10.1007/s00180-020-00999-9

[42] Stiawan, D., Idris, M.Y.B., Bamhdi, A.M. and Budiarto, R. (2020) CICIDS-2017 Dataset Feature Analysis with Information Gain for Anomaly Detection. *IEEE Access*, **8**, 132911-132921. https://doi.org/10.1109/access.2020.3009843

Abbreviations

The following abbreviations are used in this manuscript:

ML	Machine Learning
DL	Deep Learning
SA	Simulated Annealing
PSO	Particle Swarm Optimization
GT	Game Theory
EA	Evolutionary Algorithm
GA	Genetic Algorithms
DNN	Deep Neural Network
IoMT	Internet of Medical Things
PSD	Power Spectral Density
CICIDS	Canadian Institute for Cybersecurity Intrusion Detection System 2017
TCP	Transmission Control Protocol
IAT	Inter-Arrival Time
FP	False Positive
FN	False Negative
DoS	Denial of Service
DDoS	Distributed Denial of Service
SQL	Structured Query Language
CSRF	Cross-Request Site Forgery
XSS	Cross-Site Scripting
LSTM	Long Short-Term Memory
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
Adam	Adaptive Moment Estimation
RMSProp	Root Mean Square Propagation
ReLU	Rectified Linear Unit
Tanh	Hyperbolic Tangent
TPU	Tensor Processing Unit
GPU	Graphics Processing Unit
CPU	Central Processing Unit
WIO	Without Integrated Optimization