International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

Efficient Ransomware Detection for Resource-Constrained IoT Devices using Genetic Algorithm and Random Forest

Aziz Fandi
Master in Web Science, Syrian Virtual University.

Dr. Sira Astour
Ph.D., Eng. Faculty of Information and Communication Engineering, Arab International University (AIU), Syria.
ORCID 0000-0003-3976-9258

Abstract - Over the past years, with the great development in technology and communications, the dependance on edge computing/Internet of Things devices to operate many services has increased. With this increase, many attacks targeting IoT have emerged, exploiting the limited resources of this environment, which makes it difficult to install large and complex detection models and difficult to perform intensive processing on edge nodes. One of the most prominent threats was ransomware targeting edge devices to disrupt critical and sensitive services. Numerous studies have focused on finding solutions to detect ransomware, but most of these solutions are costly in terms of memory consumption and response time. This limitation makes these solutions unsuitable for use on edge devices. To solve this problem, this paper proposes a hybrid detection model that optimizes the random forest algorithm using a genetic algorithm which acts as a filter, eliminating weak, low-performing decision trees and retaining only the most efficient, high-performing ones. This reduces the number and complexity of trees without negatively impacting prediction accuracy. The proposed model achieved 98% accuracy. With regards to the efficiency in the utilization of resources, the new model was compared to the base model, which was the random forest model, with a reduction in the number of decision trees in the new model from 100 to 10 (a reduction of 90%), and a reduction in the size of the model from 144.412 MB to 4.272 MB (a reduction of 97%). With regards to the reduction in time, the avg. inference time, in the new model, reduced from 1281.731 ms to 274.240 ms (improved by 78.61%), and the inference time/samples reduced from 3.524 ms to 3.187 ms (improved by 9.57%). These results show that the proposed hybrid model represents a practical solution for enhancing security in distributed and dynamic environments and is suitable for operation on devices with limited resources.

1. INTRODUCTION:

In this digital world, ransomware has been identified as one of the most dangerous and rapidly increasing threats on the internet today [1] which attacks the system by silently infiltrating the system, whereby once inside, they lock or encrypt critical files, giving the cybercriminal a ransom, usually in a cryptocurrency, before access can be granted to the already encrypted or locked files. Thus, the effects of ransomware attacks can be felt immediately and can be very severe [2] [3]. It's not only that access has been denied to a particular file; rather, the effects can be devastating to businesses and organizations that get affected by ransomware attacks, as they end up resulting in considerable financial losses due to system shutdowns and downtimes [4]. The effects of ransomware attacks on IoT networks can be very devastating, as they end up not only resulting in the loss of digital data but can completely cripple smart systems by resulting in substantial financial losses due to system downtimes, process shut-downs, or interruption of continuity of medical systems and critical infrastructure by ransomware attacks on IoT networks that implement edge computing, whereby control or processing decisions of the IoT system reside near the data source.

The world has in the past witnessed many ransomware attacks, such as WannaCry, Ryuk and NotPetya, which resulted losses in billions of dollars. These attacks showed the weaknesses in the security methods used by companies and organizations, and proved the need to develop more robust systems capable of detecting these viruses before the attack is occured [5]. With the rising trend of

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

smart systems, these attacks not only pose a threat to traditional computer systems, but they have also started targeting IoT devices, as they do not require sophisticated protection measures owing to their resource constraints. Sensors, smart controllers, as well as edge nodes, have been favored by attackers for their insecure authentication, absent security updates, as well as simple communication protocols.

Traditional protection techniques depended on signatures which means that the system had to have specific patterns for each type of virus. When testing a new program, it would match its patterns with the stored patterns. Therefore, if this technique want to detect a specific threat, it had to have a matching pattern. This cannot be guaranteed, especially with the rapid development of technology in general [6]. We can say that this method is very good at identifying viruses similar to patterns stored in the system, but when new viruses appear or the attacker makes modifications to the virus's features to appear identical to normal patterns, this method fails to detect them [7]. This problem is exacerbated in dynamic edge environments because the behavior of the operating patterns and device are constantly changing.

This gap has led many researchers to turn to machine learning, which can learn patterns from data and detect abnormal behavior that doesn't fit the usual pattern or normal state [8]. However, even machine learning has its limitations. It requires large, clean, and balanced datasets to operate reliably and can be susceptible to overfitting [9]. Furthermore, the strict limitations on IoT device resources necessitate the design of models that are both lightweight and efficient.

Therefore, This approach aims to build a lightweight model, which can works on edge nodes, for detecting ransomware in IoT before they cause significant damage to physical and digital systems.

2. LITERATURE REVIEWS:

The reviewed studies reflect the significant evolution of ransomware detection techniques with each bringing a unique perspective and contribution but the strategies, data reliance, and outcomes vary notably across the papers.

In short, a study [10], presented the XRAN system, an intelligent system that uses deep learning but also allows you to understand its decisions (through SHAP and LIME). This is crucial because any "black box" making decisions raises suspicion. However, the problem with this system is that it's computationally intensive, meaning it can't run quickly and instantly on any weak machine. On the other hand, the study [11] presented another system based on a multi-layered neural network that attempts to balance examining static code with the dynamic behavior of the virus. This system performs excellently and is "skillful" in many tasks, but it comes at a price in terms of high training costs, and, more importantly, we still don't know exactly how it will be able to handle the new "zero-day" attacks that appear suddenly.

In study [12] , researchers attempted to overcome the problem of outdated data by using GANs to simulate sophisticated and fake ransomware behavior. However, the risk here is that they might become overly reliant on this artificial data and deviate from the actual behavior of real-world viruses. Meanwhile, study [13] took a rather unusual "technical" approach, converting the virus's binary data into images and analyzing them using image recognition. This is an innovative idea, but the problem is that this additional "conversion" step could cause them to lose important details and information. This may partly explain the moderate F1-score reported.

At the same time, the study [14] changed the focus and looked at performance indicators at the entire system level, and used hardware counters to detect the problem. The good thing is that DeepWare exploited hardware-level data, but the big problem is that this method is not easy to use everywhere, especially in cloud or virtual systems that do not always allow reaching this level.

On the other hand, more advanced, the study [15] presented a detection method that relies on “Transformers”, which analyzes “Assembly Language” in order to extract deeper and more understandable information. But use these converters, especially DistilBERT requires huge computing power and takes a long time to process.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

The paper [16] focused more on the topic of 'meaning' or 'semantic dimension', and relied on large language models (LLMs) to integrate information coming from experts and specialized laws through a module called LATAP. Combining human experience with machine capabilities is a solid and futuristic idea, but the problem is that the accuracy of their model was only 90%, which makes it weaker than the other models we talked about. Also, in order for this system to remain operational and suitable for new threats, it will need constant updates to the laws and specialized data, which of course is not easy or not always possible to do.
As for the research paper [17], it identified an important weakness in the old systems, which is that they cannot understand the subtle differences in meaning. It tried to solve this by merging 'BERT' with 'CNN' and 'LSTM'. This combination created a powerful model that understands context and order perfectly, but this structure remains very heavy, requires tremendous computing power, and is also very expensive.

The paper [18] went back to the older and more practical method, and used a set of machine learning classifiers. They relied on analyzing the importance of each characteristic or feature in order to increase performance in all these classifiers. But the problem or flaw here is that it is possible for a 'data leak' to occur between the training groups and the test groups after they have chosen these characteristics. This means that the accuracy they advertise may be slightly exaggerated.

3. METHODOLOGY:

IoT environments and edge devices are becoming more and more susceptible to security issues because of the high deployment of smart devices in those environments that are resource-constrained and are becoming more and more susceptible to ransomware attacks. The environments are also characterized by their distributed nature and the use of resources that are limited in their capabilities. This makes it inefficient to use traditional models or even deep models in those environments. There has been a significant increase in the application of machine learning in the detection of ransomware attacks. However, most of the traditional models were not designed to be used in environments that are highly resource-constrained.

The research gap identified in this area is filled by this current research, as it is required that the proposed methodology for ransomware detection is intelligent, light, and capable of adapting to the evolution of ransomware. This calls for ensuring that the proposed methodology retains its light nature and response time for ransomware detection, as discussed above. To achieve the proposed objectives in this research, it is suggested in the proposed methodology for ransomware detection that a hybrid model of a genetic algorithm [19] and a decision tree [20] will be used, in which a genetic algorithm will be used as a mechanism to develop the most efficient decision trees. This will help in developing a more efficient random forest model, as suggested in this research, for early ransomware detection.

The objective of this section is to describe how this optimization can be used to produce a concise and accurate ensemble of decision trees that can be used in a resource-constrained environment such as the Internet of Things and edge computing.

Unlike other forms of random forests that depend on a random selection of features and samples, this method will employ an evolution of parameters for hyper decision trees using a genetic algorithm (GA). The development process will aim to produce a reduced number of high-quality trees to improve a new forest. This section will discuss how dataset development, evolution design, learning a decision tree, an ensemble, and evaluation will be conducted, together with mathematical formulation and diagrams of the algorithm. Figure 1 shows the key steps of the proposed methodology.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

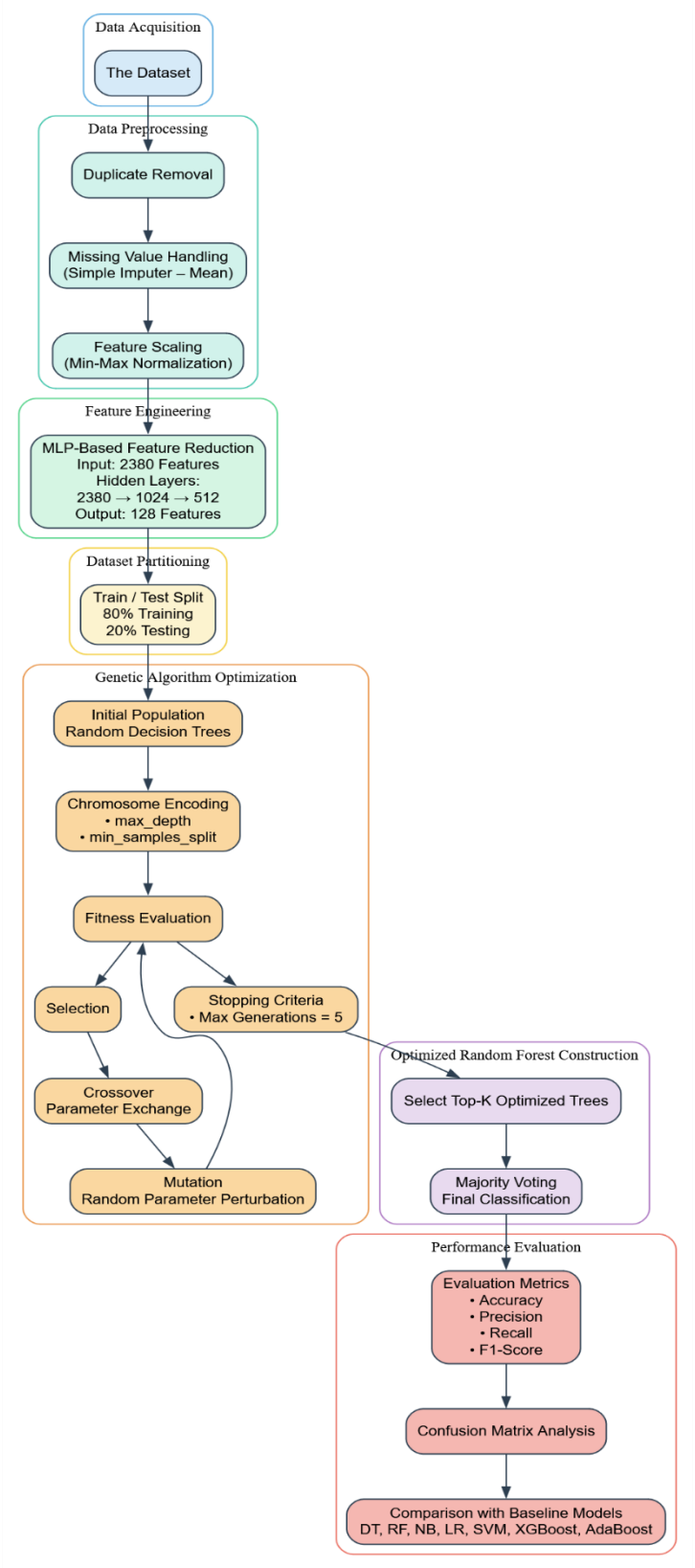

Figure 1. The key steps of the proposed methodology

3.1 Data set used:

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

We used the EMBER dataset which is a collection of features from million Portable Executable (PE) files [21]. Each PE file contains 2,381 features. In our research, due to limited physical resources, we used 400,000 samples, divided into 80% for training and 20% for testing.

Suppose the data set is defined as follows:

$$D = \{(x_i, y_i)\}_{i=1}^{N} \quad (1)$$

Where:

x_i: The learned embedding vector with dimension d.

y_i: Indicates the class classification where C=2 (binary classification).

N: The samples count.

The data set is randomly divided as follows:

$$D_{train} \cup D_{test} = D, D_{train} \cap D_{test} = \emptyset \quad (2)$$

3.2 Data preprocessing:

Duplicate records are removed from the dataset because duplicate records will consume more memory space and slow down the training process, making it useless [22].

The next step of preprocessing process is imputation which includes replace missing values with a statistic (mean). Datasets typically contain observations with missing values, the main options for handling this problem are [23]: (1) Exclude rows that contain any missing values, but this process leads to loss information that could be important. (2) Exclude any variable (column) that contains missing values, but this procedure may also result in the loss of predictive variables that significantly affect model performance. (3) Fill in missing entries with estimated values. This process represents the best procedure for filling in missing values because it avoids the loss of information that could affect performance results. The proposed methodology uses the SimpleImputer() function to fill in missing values with the arithmetic mean of the values of the variable in which the missing value appears [24].

Normalization is the process of resetting a data value range so that all values fall within a defined range, typically between 0 and 1 [25]. In this study, we used a Min-Max Scaler [26]. The formula for the Min-Max Scaler is as follows:

$$x_{normalized} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (3)$$

Where:

x: The original value of the feature.

x_{min}: The minimum value of that feature in the dataset.

x_{max}: The maximum value of that feature in the dataset.

3.2.1 Reducing the dimensions of the features:

The features we have are enormous, totaling 2,381. This quantity leads to computational inefficiencies and may also contain irrelevant or redundant elements (noise) that can compromise classification accuracy [27]. Therefore, the optimal solution is to "simplify" this data and reduce its size, but of course, without losing essential and important information. To achieve this, we will use a neural network (MLP) [28]. This network starts with an input layer containing 2,381 "cells" (corresponding to the number of features we have), then it passes through two "hidden" layers of 1,024 and 512 cells respectively, and finishes with an output layer containing only 128 cells, essentially taking all the information and extracting

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

the useful summary. The MLP network undergoes a new training process, where each batch is divided into equal-sized training and test sets from each class. The benchmark scores for each class are calculated based on the arithmetic mean of each feature, and the Euclidean distance is used to classify the test samples, followed by a backpropagation error process to adjust the network weights. The goal of this network is to reduce the feature space from 2380 to 128 features. Figure 2 illustrates how the artificial neural network is trained.

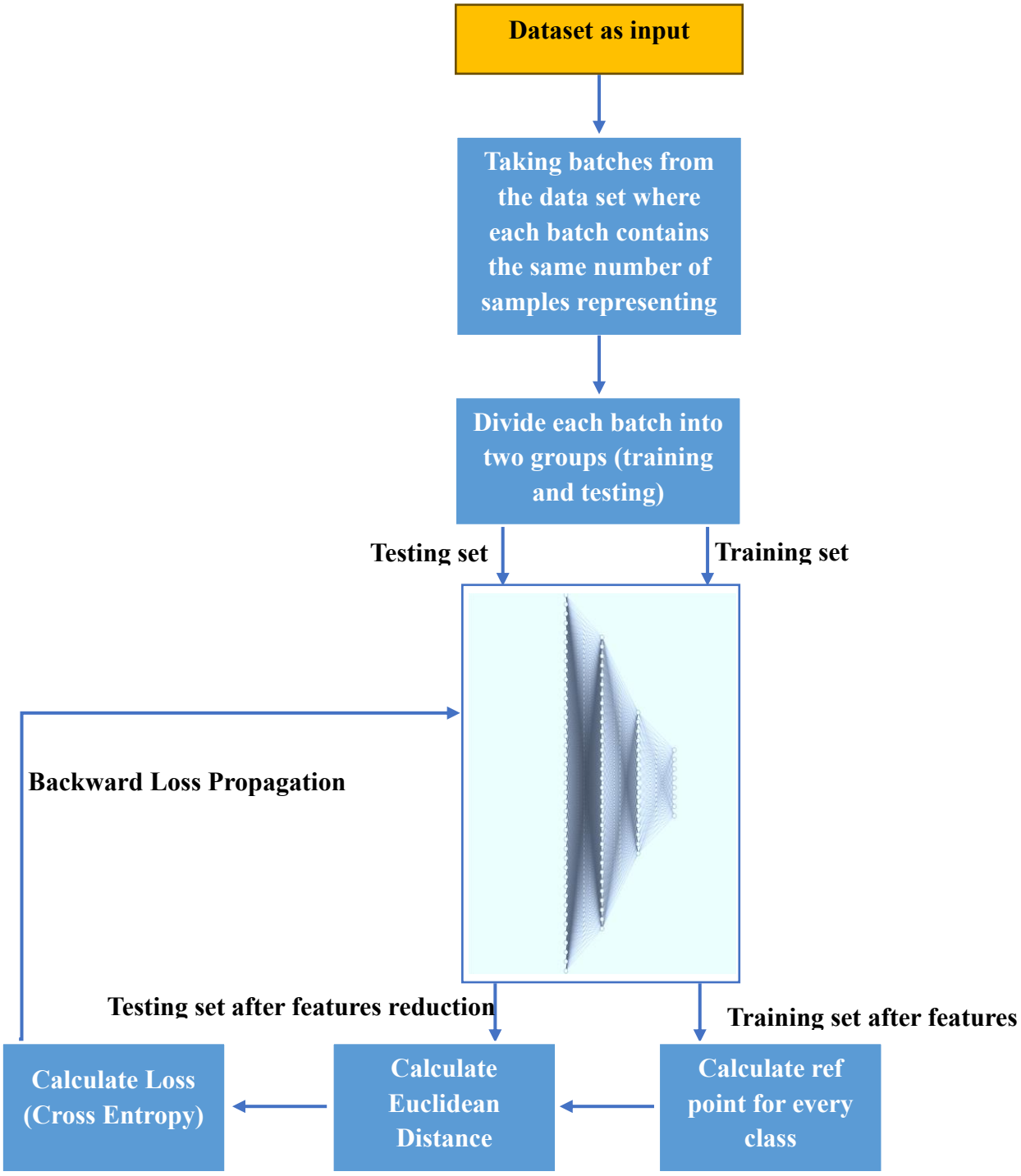

Figure 2. How the proposed ANN is trained

The architecture showed in figure 2 is designed to transform high-dimensional inputs into a compact representation while preserving the distinctive features of the classes. The network is trained to minimize classification errors using a novel distance-based loss approach.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

The proposed ANN includes a set of layers containing artificial neurons. Each neuron in any layer has connections to all neurons in the next layer, which is why it is called dense due to the density of connections between neurons within each layer [29]. Each of these connections has a weight, and the goal of the training process is to adjust these weights to achieve the lowest error in the output. These weights are adjusted at each iteration based on the error between the actual value and the predicted value, represented here by the Euclidean distance between a point representing the sample in a multidimensional space and a reference point representing each class [29].

In the calculate loss in figure 2, the cross_entropy function plays a central role in computing the classification loss [30], specifically, the model predicts class labels based on the minimum distance between a test sample and class ref point. The distances tensor contains the pairwise distances between each input and the class ref point and instead of traditional logits, these distances are passed directly into the cross_entropy function. Although cross-entropy typically expects raw class scores (logits), where higher values indicate greater confidence in a class prediction [31], this approach leverages the inverse relationship, where the resulting of Euclidean distance is passed to the cross_entropy function in reverse (-distance),so, smaller distances as indicative of higher similarity. Consequently, the function effectively learns to minimize the distance between each input and its corresponding class prototype. By computing the cross-entropy loss over distance-based pseudo-logits, the model is guided to bring embeddings of the same class closer together, thereby improving its ability to distinguish between classes. The loss is calculated as shown in formula (2).

$$loss = cross\ entropy\ (-distance, label) \qquad (4)$$

Formula 2 is the cross-entropy loss, but here we're passing distances as logits. Normally, the logits are passed to cross_entropy (higher = more likely), but here, we're passing distances, so lower is better.
So, we invert the distances by passing them directly into cross_entropy function and system assumes higher logits = more likely= the lowest distance. So, classes with lower distances (more similar) will result in lower loss, because cross-entropy rewards higher values at the true label index. This makes the model learn to minimize the distance to the correct class ref point.

3.3 Proposed Model:

The proposed methodology is based on a method for building a random forest using a combination of genetic algorithms and decision trees. The principle of inference is used to build the random forest, where the best performing trees are kept using the genetic algorithm, and poorly performing decision trees are eliminated. This results in a forest of highly efficient trees. This improves the performance and reduces the memory space required to store a large number of decision trees. The basic principle of this method is that decision making in the random forest is based on majority voting, and the efficiency of the decision trees generated by the random forest algorithm is not individually tested. This implies that poorly performing decision trees may affect the final decision. Therefore, to improve the efficiency of the classification process, weak trees are eliminated. This results in a reduced number of trees, which reduces the size of the model and, consequently, the memory space required to store the model. Figure 3 shows the development process of a genetic algorithm.

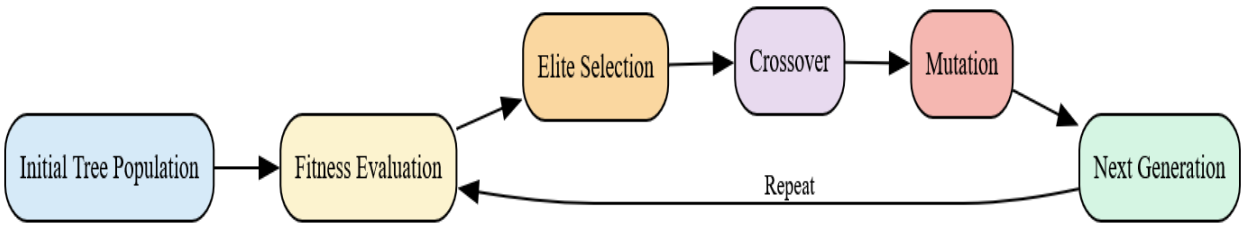

Figure 3. The development process of a genetic algorithm

3.3.1 Individual representation and population preparation:

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

A genetic algorithm is used to develop an ensemble of decision trees, which improves prediction accuracy and performance robustness. Each individual tree T_k in the ensemble is subject to a small set of hyperparameters θ_k. The process of evolution iteratively improves the population through elite selection, hybridization, and mutation, guided by a fitness function.

$$\theta_k = \{\text{Max}Depth, MinSampleSplit\} \quad (5)$$

where $k \in \{1, 2, \dots, P\}$ and P is the population size in each generation.

These standards directly control:

- Model capacity (tree depth).
- Degree of generalization (minimum number of samples needed to split a node).

The initial population is:

$$\mathcal{P}_0 = \{\mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_P\} \quad (6)$$

Each tree T_k is randomly generated within finite ranges of hyperparameters θ_k:

$$MaxDepth \in \{5, \dots, 15\}, MinSample\ Split \in \{2, \dots, 20\} \quad (7)$$

This random initialization ensures diversity in the search space to prevent premature convergence.

An ensemble of optimized decision trees $\mathcal{P} = \{T_1, T_2, \dots, T_N\}$ is trained using genetic algorithm (GA). The fitness function is determined by accuracy, selecting only decision trees that achieve high accuracy, and excluding weak trees because they will affect the quality of the collective decisions made for the trees within the final random forest.

$$Fitness\ (Y_i) = Accuracy(T_i) = \frac{1}{N}\sum_{i=1}^{N} T(x_i) = y_i \quad (8)$$

3.3.2 Mechanism for selecting individuals:

Trees are ranked according to their fitness, maintaining the top 20% of individuals generated across all generations and this ensures non-decreasing fitness across generations. The selection mechanism depends on the tournament selection. The tournament selection mechanism is based on selecting one individual from a randomly selected set of k chromosomes, focusing on the individual with the best fitness. This mechanism maintains the strength of selection by monitoring k chromosomes, and also selects the fittest chromosomes. This mechanism reduces the risk of losing high-performance trees due to random changes. Figure 4 shows the tournament selection.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

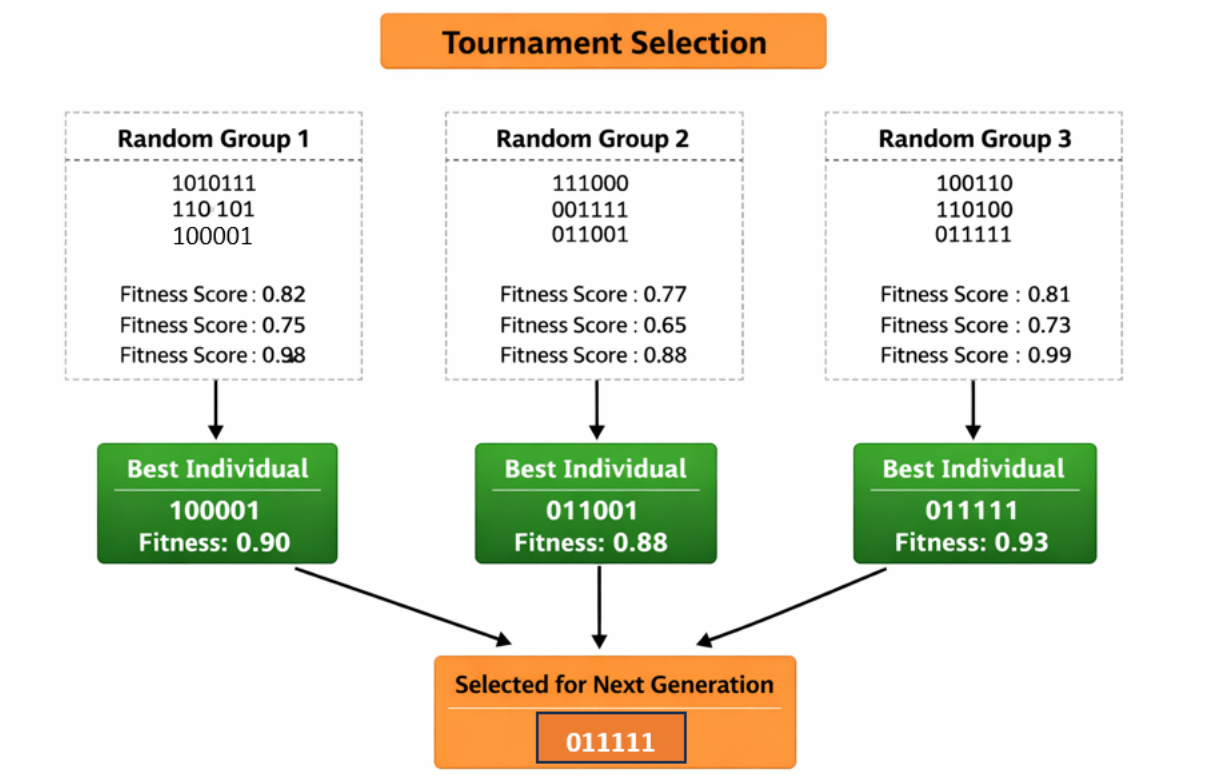

Figure 4. The tournament selection

3.3.3 Crossover:

To explore new regions in hyperparameter space, we apply the crossover process to pairs of selected parents trees from the previous generation population. Assuming three parents individuals:

$$\mathcal{T}_p^{(1)} = \{MaxDepth_1, MinSamplesSplit_1\}$$
$$\mathcal{T}_p^{(2)} = \{MaxDepth_2, MinSamplesSplit_2\} \quad (9)$$
$$\mathcal{T}_p^{(3)} = \{MaxDepth_3, MinSamplesSplit_3\}$$

An individual child T_c is created by inheritance at the parameter level:

$$MaxDepth_c = random\ choice\{MaxDepth_1, MaxDepth_2, MaxDepth_3\} \quad (10)$$

$$MinSamplesSplit_c = random\ choice\{MinSamplesSplit_1, MinSamplesSplit_2, MinSamplesSplit_3\} \quad (11)$$

3.3.4 Mutation:

To preserve genetic diversity and prevent convergence toward suboptimal solutions, we apply mutation to the resulting trees with a predefined probability p_m. Given a tree $\mathcal{T} = \{MaxDepth, MinSamplesSplit\}$, we apply mutation to introduce finite random perturbations.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

$$MaxDepth' = clip(MaxDepth + \Delta_d) \quad (12)$$

$$MinSamplesSplit' = clip(MinSamplesSplit + \Delta_s) \quad (13)$$

Where Δ_d and Δ_s are small discrete sets {-2,-1,0,1,2}. The Clip() function is a function used to enforce valid ranges so that they do not exceed the ranges that were specified at the beginning of the work ($MaxDepth \in \{5, \dots, 15\}$ $MinSample\ Split \in \{2, \dots, 20\}$).

3.3.5 Generation update and stop standard:

The next generation $\mathcal{P}_{g+1}$ contains of :

$$\mathcal{P}_{g+1} = \varepsilon_g \cup \mathcal{C}_g \cup \mathcal{M}_g \quad (14)$$

Where:

ε_g: Trees transferred to the next generation via tournament selection mechanism.

$\mathcal{C}_g$: The trees resulted from crossover process.

$\mathcal{M}_g$: The trees resulted from mutation process.

The evolutionary process continues for a specified number of generations or until convergence in fitness is observed, and the best performing trees across all generations are retained to build the final decision forest.

After G generations, the K most efficient trees are selected to form a random forest F, and the final majority prediction is performed as shown in formula (5.5):

$$\hat{y} = argmax_{c \in \{0,1\}} \sum_{T \in F} T(x) = c \quad (15)$$

3.4 Evaluation Metrices:

The confusion matrix (CM) is a basic tool used to assess the accuracy of a classification [32]. Figure 5 shows the confusion matrix in general.

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

Figure 5. The confusion matrix in general

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

It also helps calculate key measures like accuracy, precision, recall and F1 score.

$$Pr\,e\,cision = \frac{TP}{TP + FP} \qquad (16)$$

$$Re\,c\,all = \frac{TP}{TP + FN} \qquad (17)$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \qquad (18)$$

$$F1_{Score} = 2 \times \frac{Pr\,e\,cision \times Re\,c\,all}{Pr\,e\,cision + Re\,c\,all} \qquad (19)$$

4. RESULTS AND DISCUSSIONS:

In this section, we present the performance metrics achieved by applying the basic machine learning algorithms, which will be compared with the results of the proposed model. Table 1 shows a comparison between the proposed model with machine learning algorithms.

Table 1. A comparison between the proposed model with machine learning algorithms

Algorithm	Precision	Recall	F1_Score
Decision Tree	0.913	0.912	0.912
Naïve Bayes	0.798	0.801	0.799
Logistic Regression	0.842	0.837	0.839
SVM	0.863	0.855	0.858
RF	0.955	0.945	0.945
AdaBoost	0.86	0.85	0.85
XGBoost	0.93	0.93	0.92
Proposed Model	0.98	0.98	0.98

The Decision Tree model achieved an F1-Score of approximately 0.912, which is considered acceptable. However, its reliance on a single tree makes it more susceptible to being overridden, which limits its ability to generalize.

The Naïve Bayes model performed the worst. It achieved 0.799 F1-Score. This is due to its basic assumption of the independence of properties, an assumption that is inconsistent with the complex and interconnected nature of ransomware properties.

The Logistic Regression and Supporting Vector Machine (SVM) models showed moderate performance. They achieved F1-Scores 0.839 and 0.858 respectively. This suggests that these two models have difficulty representing the complex boundaries between benign behaviors and ransomware behaviors.

The traditional random forest model showed an improvement over individual models. It achieved 0.945 F1-Score. Its improvement was owing to the inclusion of several decision trees in the model, thereby reducing variance and improving overall generalization abilities. Although it performs well with low variance, it generally has the drawback of increased memory usage and computation time, which would be observed in the upcoming results. It poses a tremendous problem in an IoT/Edge scenario with resource limitations.

Furthermore, we note the superiority of the proposed model over other ensemble algorithms such as Adaboost and XGBoost in terms of all performance metrics, where the Adaboost model achieved an F1 value of 0.85 and the XGBoost model achieved an F1 value of 0.92.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

In contrast, the proposed model achieved the highest performance among all models, recording balanced and high values for Precision, Recall, and an F1-Score of 0.98. This superiority reflects the ability of the proposed hybrid approach, which combines a genetic algorithm with decision trees, to select the most efficient and discriminating structures within the random forest. The genetic algorithm helps eliminate underperforming trees and reduces structural complexity while maintaining high ransomware detection accuracy and minimizing false positive rates.

Figure 6 shows a comparison between the basic algorithms and the proposed model in terms of accuracy.

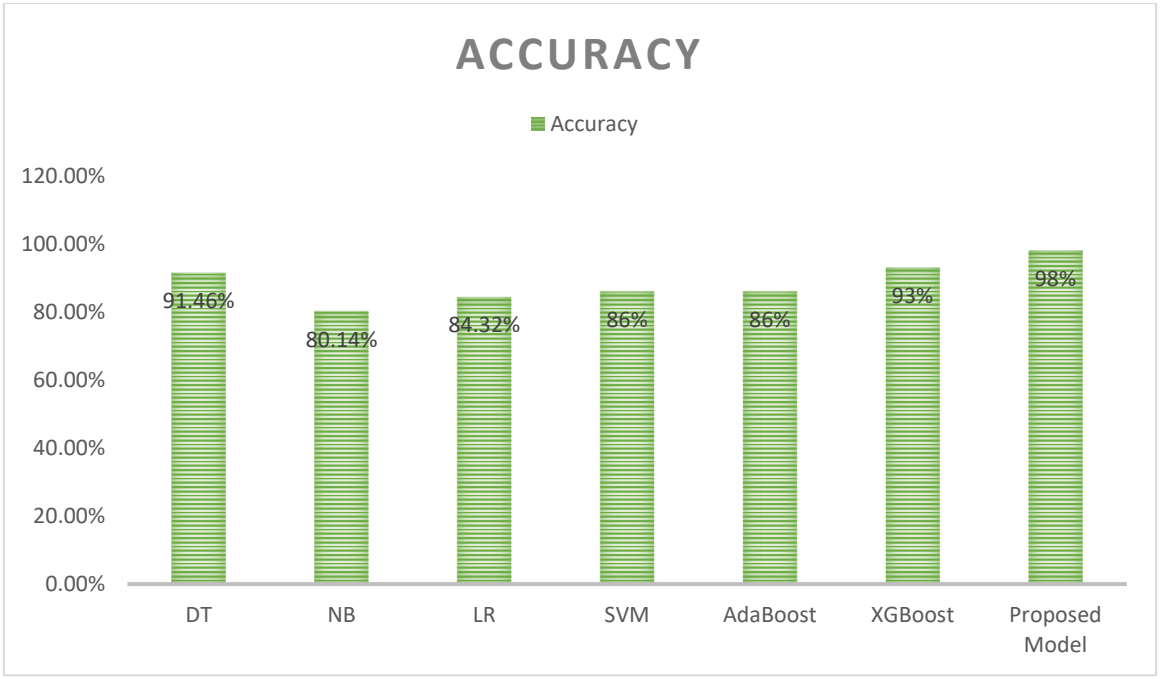

Figure 6. A comparison between the basic algorithms and the proposed model in terms of accuracy

Table 2 shows a comparison between RF and the proposed model in terms of hyperparameter values.

Table 2. A comparison between RF and the proposed model in terms of hyperparameter values

Hyperparameter	Description	Baseline RF	Proposed Model
n_estimators	Number of decision trees created in the forest	100	50
criterion	Function of measuring split quality	gini	gini
min_samples_split	Minimum number of samples required to split an internal node	2	**Random between 2, 20**
min_samples_leaf	The minimum number of samples required per leaf node. The split point at any depth will only be considered if at least `min_samples_leaf` leaves training samples in both the right and left branches.	1	1

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

max_features	Number of features to consider when looking for the best partition	sqrt	sqrt
max_depth	Maximum depth per tree. If this parameter is set to None, the nodes will be expanded until all leaves are pure (all samples within a leaf belong to the same class) or until all leaves contain fewer samples than min_samples_split.	None	**15**
Parameters Resulted After Training			
Min depth resulted	Minimum depth between resulting trees	37	**5**
Max depth resulted	Maximum depth of the tree between the resulting trees	70	**15**
Number of saved trees	The number of trees in the final decision in the forest that are stored in memory	100	**10**

The comparison in Table 3 shows that the proposed model achieves higher accuracy than the base random forest model while consuming fewer resources, as it stores only 10 decision trees with a maximum depth of 15. This 80% reduction in model size directly translates to lower memory consumption for storage, a crucial advantage for resource-constrained environments. Table 4 shows a quantitative comparison of efficiency metrics between the base random forest model and the proposed model.

Table 4. A quantitative comparison of efficiency metrics between the base random forest model and the proposed model

Metric	Baseline RF	Proposed model	Improvement
Number of Trees	100	10	90% Reduction
Model Storage Size	144.412 MB	4.272 MB	97% Reduction
Average Inference Time (Test data)	1281.731 ms ± 33.683	274.240 ms ± 30.843	78.61% Reduction
Avg. Inference Time per Sample	3.524 ms	3.187 ms	9.57% Reduction

5. CONCLUSIONS AND FUTURE WORKS:

This research addresses the problem of traditional models failing to keep pace with the rapid changes in ransomware threats, particularly in IoT environments. The contribution to this work was the design of a hybrid detection model that combines the power of genetic algorithms with the flexibility of a random forest algorithm. The proposed model achieved 98% accuracy. With regards to the efficiency in the utilization of resources, the new model was compared to the base model, which was the random forest model, with a reduction in the number of decision trees in the new model from 100 to 10 (a reduction of 90%), and a reduction in the size of the model from 144.412 MB to 4.272 MB (a reduction of 97%). With regards to the reduction in time, the avg. inference time, in

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

the new model, reduced from 1281.731 ms to 274.240 ms (improved by 78.61%), and the inference time/samples reduced from 3.524 ms to 3.187 ms (improved by 9.57%).

Despite the outstanding results of this research, some limitations should be noted. The research relied on the EMBER dataset, which focuses on PE files. The model may face challenges in adapting immediately to threats that do not rely on PE files, such as script-based ransomware. Future work should focus on developing a mechanism for continuously and automatically training the model to update its knowledge in real time as new threats emerge. Furthermore, the features used should be expanded to include data such as network traffic, in addition to file-specific characteristics.

REFERENCES:

[1] O. Vermesan *et al.*, "Internet of things beyond the hype: Research, innovation and deployment," in *Building the Hyperconnected Society-Internet of Things Research and Innovation Value Chains, Ecosystems and Markets*: River Publishers, 2022, pp. 15-118.
[2] A. N. Lone, S. Mustajab, and M. Alam, "A comprehensive study on cybersecurity challenges and opportunities in the IoT world," *Security and Privacy,* vol. 6, no. 6, p. e318, 2023.
[3] D. Canavese, L. Mannella, L. Regano, and C. Basile, "Security at the edge for resource-limited IoT devices," *Sensors,* vol. 24, no. 2, p. 590, 2024.
[4] O. Kuznetsov, S. Adilzhanova, S. Florov, V. Bushkov, and D. Peremetchyk, "Privacy-Preserving Federated Learning for Distributed Financial IoT: A Blockchain-Based Framework for Secure Cryptocurrency Market Analytics," *IoT,* vol. 6, no. 4, p. 78, 2025.
[5] E. Fazeldehkordi and T.-M. Grønli, "A survey of security architectures for edge computing-based IoT," *IoT,* vol. 3, no. 3, pp. 332-365, 2022.
[6] Lalhriatpuii, Ruchi, and V. Wasson, "Comprehensive exploration of IoT communication protocol: coap, MQTT, HTTP, LoRaWAN and AMQP," in *International Conference on Machine Learning Algorithms*, 2024: Springer, pp. 261-274.
[7] M. Neupane, "A Comprehensive Survey on Dynamic Software Updating Techniques in IoTs," *arXiv preprint arXiv:2412.02450,* 2024.
[8] M. Shafiq, Z. Gu, O. Cheikhrouhou, W. Alhakami, and H. Hamam, "The Rise of "Internet of Things": Review and Open Research Issues Related to Detection and Prevention of IoT-Based Security Attacks," *Wireless Communications and Mobile Computing,* vol. 2022, no. 1, p. 8669348, 2022.
[9] K. B. Adedeji, A. M. Abu-Mahfouz, and A. M. Kurien, "DDoS attack and detection methods in internet-enabled networks: Concept, research perspectives, and challenges," *Journal of Sensor and Actuator Networks,* vol. 12, no. 4, p. 51, 2023.
[10] S. Gulmez, A. G. Kakisim, and I. Sogukpinar, "XRan: Explainable deep learning-based ransomware detection using dynamic analysis," *Computers & Security,* vol. 139, p. 103703, 2024.
[11] E. B. Karbab, M. Debbabi, and A. Derhab, "SwiftR: Cross-platform ransomware fingerprinting using hierarchical neural networks on hybrid features," *Expert Systems with Applications,* vol. 225, p. 120017, 2023.
[12] U. Urooj, B. A. S. Al-Rimy, A. B. Zainal, F. Saeed, A. Abdelmaboud, and W. Nagmeldin, "Addressing behavioral drift in ransomware early detection through weighted generative adversarial networks," *Ieee Access,* vol. 12, pp. 3910-3925, 2023.
[13] J. Zhu, J. Jang-Jaccard, A. Singh, I. Welch, H. Al-Sahaf, and S. Camtepe, "A few-shot meta-learning based siamese neural network using entropy features for ransomware classification," *Computers & Security,* vol. 117, p. 102691, 2022.
[14] G. O. Ganfure, C.-F. Wu, Y.-H. Chang, and W.-K. Shih, "Deepware: Imaging performance counters with deep learning to detect ransomware," *IEEE Transactions on Computers,* vol. 72, no. 3, pp. 600-613, 2022.
[15] M. Gaber, M. Ahmed, and H. Janicke, "Zero day ransomware detection with Pulse: Function classification with Transformer models and assembly language," *Computers & Security,* vol. 148, p. 104167, 2025.
[16] C. Zhou *et al.*, "SRDC: Semantics-based Ransomware Detection and Classification with LLM-assisted Pre-training," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025, vol. 39, no. 27, pp. 28566-28574.
[17] J. Liu, Y. Zhao, Y. Feng, Y. Hu, and X. Ma, "Semalbert: Semantic-based malware detection with bidirectional encoder representations from transformers," *Journal of Information Security and Applications,* vol. 80, p. 103690, 2024.
[18] M. A. Hossain, T. Hasan, F. Ahmed, S. H. Cheragee, M. H. Kanchan, and M. A. Haque, "Towards superior android ransomware detection: An ensemble machine learning perspective," *Cyber Security and Applications,* vol. 3, p. 100076, 2025.
[19] S. J. Rigatti, "Random forest," *Journal of insurance medicine,* vol. 47, no. 1, pp. 31-39, 2017.
[20] M. Ayara, J. Timmis, R. de Lemos, L. N. de Castro, and R. Duncan, "Negative selection: How to generate detectors," in *Proceedings of the 1st international conference on artificial immune systems (ICARIS)*, 2002, vol. 1: University of Kent at Canterbury Printing Unit Canterbury, UK, pp. 89-98.
[21] H. S. Anderson and P. Roth, "Ember: an open dataset for training static pe malware machine learning models," *arXiv preprint arXiv:1804.04637,* 2018.
[22] D. Bruschi, L. Cavallaro, and A. Lanzi, "Diversified process replicæ for defeating memory error exploits," in *2007 IEEE International Performance, Computing, and Communications Conference*, 2007: IEEE, pp. 434-441.
[23] T. Emmanuel, T. Maupong, D. Mpoeleng, T. Semong, B. Mphago, and O. Tabona, "A survey on missing data in machine learning," *Journal of Big data,* vol. 8, no. 1, p. 140, 2021.
[24] S. Tripathi, L. Singh, and J. Sermanraja, "Complete Data Using Exploratory Data Analysis and ML Algorithms," in *2024 4th International Conference on Technological Advancements in Computational Sciences (ICTACS)*, 2024: IEEE, pp. 1293-1296.
[25] D. Singh and B. Singh, "Feature wise normalization: An effective way of normalizing data," *Pattern Recognition,* vol. 122, p. 108307, 2022.
[26] B. Deepa and K. Ramesh, "Epileptic seizure detection using deep learning through min max scaler normalization," *International journal of health sciences,* no. I, pp. 10981-10996, 2022.
[27] D. A. A. Gnana, S. A. A. Balamurugan, and E. J. Leavline, "Literature review on feature selection methods for high-dimensional data," *International Journal of Computer Applications,* vol. 136, no. 1, pp. 9-17, 2016.
[28] S.-J. Lee *et al.*, "A dimension-reduction based multilayer perception method for supporting the medical decision making," *Pattern Recognition Letters,* vol. 131, pp. 15-22, 2020.
[29] G. Huang, Z. Liu, G. Pleiss, L. Van Der Maaten, and K. Q. Weinberger, "Convolutional networks with dense connectivity," *IEEE transactions on pattern analysis and machine intelligence,* vol. 44, no. 12, pp. 8704-8716, 2019.
[30] L. Li, M. Doroslovački, and M. H. Loew, "Approximating the gradient of cross-entropy loss function," *IEEE access,* vol. 8, pp. 111626-111635, 2020.
[31] Z. Zhang and Y. Boykov, "Collision Cross-entropy for Soft Class Labels and Entropy-based Clustering," 2024.
[32] S. Sathyanarayanan and B. R. Tantri, "Confusion matrix-based performance evaluation metrics," *African Journal of Biomedical Research,* vol. 27, no. 4S, pp. 4023-4031, 2024.