In the Name of God

Journal of
Information Systems & Telecommunication
Vol. 13, No.3, July-September 2025, Serial Number 51

Research Institute for Information and Communication Technology
Iranian Association of Information and Communication Technology
Affiliated to: Academic Center for Education, Culture and Research (ACECR)

Print ISSN: 2322-1437
Online ISSN: 2345-2773
Publication License: 91/13216
Editorial Office Address: No.5, Saeedi Alley, Kalej Intersection., Enghelab Ave., Tehran, Iran,
P.O.Box: 13145-799 Tel: (+9821) 88930150 Fax: (+9821) 88930157
E-mail: info@jist.ir , infojist@gmail.com
URL: jist.acecr.org

Indexed by:
- SCOPUS www. Scopus.com
- Islamic World Science Citation Center (ISC) www.isc.gov.ir
- Directory of open Access Journals (DOAJ) www.Doaj.org
- Scientific Information Database (SID) www.sid.ir
- Regional Information Center for Science and Technology (RICeST) www.ricest.ac.ir
- Magiran www.magiran.com

Publisher:
Iranian Academic Center for Education, Culture and Research **(ACECR)**

This Journal is published under scientific support of
Advanced Information Systems (AIS) Research Group and
Telecommunication Research Group, ICTRC

Acknowledgement

JIST Editorial-Board would like to gratefully appreciate the following distinguished referees for spending their valuable time and expertise in reviewing the manuscripts and their constructive suggestions, which had a great impact on the enhancement of this issue of the JIST Journal.

(A-Z)

- Alaeiyan, Mohammad Hadi, K.N. Toosi University of Technology, Tehran, Iran
- Azarkasb, Seyed Omid, K.N. Toosi University of Technology, Tehran, Iran
- Alavi, Seyed Enayatollah, Chamran University, Ahvaz, Iran
- Abdulnabi, Saif, University of Kufa, Kufa, Iraq
- Asgari Tabatabaee, Mohammad Javad, University Of Torbat Heydarieh, Razavi Khorasan, Iran
- Borna, Keyvan, Kharazmi University,Tehran, Iran
- Afsharirad, Majid, Kharazmi University,Tehran, Iran
- Baydoun, Kamal, Lebanese University, Lebanon
- Dehghani, Maryam, Malek Ashtar University, Tehran, Iran
- Ebadati, Omid Mahdi, Kharazmi University,Tehran, Iran
- Ebady Manaa, Mahdi, Al-Mustaqbal University,Hill, Iraq
- Farsi, Hassan, University of Birjand, South Khorasan, Iran
- Fazeli Veisari, Elham, Islamic Azad University, Chalus Branch, Iran
- Forouzesh, Moslem, Trbiat Modares University, Tehran, Iran
- Ghasemzadeh, Mohammad, Yazd University, Yazd, Iran
- Hejazinia, Roya, Allameh Tabataba'i University, Tehran, Iran
- Hajizadeh, Ehsan, Amirkabir University, Tehran, Iran
- Kumar Tiwari, Anoop, National Institute of technology, Raipur, India
- Kheirkhah, Fatemeh, ACECR, Tehran, Iran
- Kazerouni, Morteza, Malek-Ashtar University of Technology, Tehran, Iran
- Kolahkaj, Maral, Islamic Azad University, Karaj Branch, Iran
- Lak, Behzad, Amin Police Academy, Tehran, Iran
- Mirroshandel, Seyed Abolghasem, University of Guilan, Rasht, Iran
- Molazadeh,Amir Hosein, K. N. Toosi University of Technology, Tehran, Iran
- Monemizadeh, Mostafa, University of Neyshabour, Nishapur, Iran
- Maghdid, Halgurd Sarhang, Koya University, Kurdistan Region, Iraq
- Radfar, Mohammad Hadi, Shahid Beheshti University of Medical Sciences, Tehran, Iran
- Salunke, Bharti, Poornima University, Jaipur, Rajasthan, India
- Taghavi Afshord, Saeid, Azad Universiti, Azarbaijan, Shabestar, Iran
- Taghavifard, Mohammad Taghi, Allameh Tabataba'i University, Tehran Iran
- Tourani, Mahdi, University of Birjand, South Khorasan, Iran
- Tanhaei, Mohammad, Ilam University, Ilam , Iran

- Uddin Talukdar, Muhammad Borhan, Daffodil International University, Bangladesh
- Valizadeh, Majid, Ilam University, Ilam, Iran
- Yaghoobi, Kaebeh, Ale Taha Institute of Higher Education, Tehran, Iran
- Zahedi, Mohammad Hadi, K. N. Toosi University of Technology, Tehran, Iran

Table of Contents

Journal of Information Systems and Telecommunication
Vol.13, No.3, July-September 2025, 165-176
http://jist.acecr.org
ISSN 2322-1437 / EISSN:2345-2773

Predicting Primary Biliary Cholangitis Stages Using Machine Learning with Automated Hyperparameter Optimization and Recursive Feature Elimination

Arman Rezasoltani[1], Amir Mohammad Khani[1], Ali Husseinzadeh Kashan[2], Shahram Agah[3*], Fatemeh Agah[4]

[1].Department of Industrial Management, Faculty of Management, University of Tehran, Tehran, Iran.
[2].Department of Industrial Engineering, Faculty of Industrial and Systems Engineering, Tarbiat Modares University, Tehran, Iran.
[3].Department of Gastroenterology and Hepatolog, Colorectal Research Center, Iran University of Medical Sciences, Tehran, Iran.
[4].The University of Adelaide, Discipline of Medicine, Adelaide, South Australia, Australia. Fatemeh.

Received: 30 Jan 2025/ Revised: 04 Sep 2025/ Accepted: 05 Oct 2025

Abstract

This research used modern machine learning ways to predict the stages of primary biliary cholangitis using data from the Mayo Clinic trial. The research aims to obtain high prediction accuracy while representing balanced evaluation metrics. Important techniques include automated hyperparameters optimization with Optuna and Recursive Feature Elimination to improve model performance. Pre-processing included handling missing values, encoding of categorical features, and addressing class imbalances using SMOTE. A total of twelve machine learning algorithms are evaluated with ensemble-based models such as CatBoost and Extra Trees producing much better results. Evaluation metrics take into account all model predictions, including accuracy, precision, recall, F1 score, and ROC-AUC for performing balanced and interpretative evaluations of performances critical for imbalanced datasets. This endeavor includes clinical and laboratory information illustrating the prospect of machine learning in advancing therapeutic diagnosis, emphasizing the rigor and robustness in evaluation laid groundwork for future research to encompass even more generalizable and robust diagnostic tools.

Keywords: Primary Biliary Cholangitis; Machine Learning; Recursive Feature Elimination; Optuna, Imbalanced Data.

1- Introduction

Primary Biliary Cholangitis (PBC), formerly known as primary biliary cirrhosis, is a chronic autoimmune liver disease. It is characterized by the gradual and progressive destruction of the liver's small bile ducts, leading to the accumulation of bile and other toxins within the liver, a condition known as cholestasis. Over time, this persistent damage can result in scarring, fibrosis, and ultimately cirrhosis. Cirrhosis is a late-stage liver disease that occurs when scar tissue replaces healthy liver tissue. The underlying pathologies that may cause this disease include viral hepatitis, chronic alcoholism, and NAFLD (non-alcoholic fatty liver disease) (Konerman et al., 2019). Chronic alcohol consumption leads to advanced forms of liver damage, which eventually result in cirrhosis and subsequent liver failure (Topcu et al., 2024). In the primary stages, the disease is asymptomatic, and awareness is typically raised only in the advanced stages. Cirrhosis may lead to liver failure, liver cancer, and, ultimately, death (Tapper & Parikh, 2023). There is a strong need for the most accurate and least invasive methods to predict the progression of cirrhosis, given the critical importance of diagnosing and managing such diseases optimally. Although traditional methods, such as liver biopsy, provide accurate results, these procedures are invasive and may lead to complications (Wei et al., 2018). Chronic alcohol consumption is one of the main causes of this disease and, in the long term, can lead to advanced stages such as cirrhosis, ultimately culminating in complete liver failure

✉ **Shahram Agah**
Agah.sh@iums.ac.ir

(Topcu et al., 2024). Previous studies have established that cirrhosis of the liver progresses through four stages. The first stage, Steatosis, is characterized by inflammation of either the liver or the bile ducts, and immediate treatment at this juncture can control the disease. The second stage, Fibrosis, involves the development of scar tissue that cuts off normal blood flow to the liver and impairs its function; however, medical treatment can halt the progression of the disease. In the third stage, Cirrhosis, healthy liver tissue is replaced by scar tissue, and swelling may occur in the spleen. Finally, the fourth stage, Liver Failure, is characterized by complete liver failure. At this stage, patients transition from normal health to a comatose state and require emergency treatment by medical professionals (Wei et al., 2018).

The subtlety of its early symptoms permits the diagnosis of cirrhosis only at advanced stages; if mismanaged, the disease can inevitably culminate in liver failure or cancer. Recent studies have highlighted the significance of early detection and management. An SEAL screening algorithm study demonstrated a remarkable 59% higher rate of early cirrhosis detection compared to routine care, thereby advocating for the role of structured programs in identifying asymptomatic cases (Labenz et al., 2022). In addition, top-down proteomics identified the proteoform signatures in plasma that correlate with the progression of cirrhosis, forming the template for a biomarker-driven risk stratification (Forte et al., 2024). Another paper emphasized the role of miRNA-gene regulatory axes in monitoring and diagnosing cirrhosis and hepatocellular carcinoma and proposed new diagnostic targets (Premnath & Shanthi, 2024). Asymptomatic superior mesenteric vein thrombosis (SMVT), however, has not been proven to significantly impact cirrhosis outcomes, unlike the risks posed by portal thrombosis (PT) (Wang et al., 2022). These collective findings emphasize the crucial role of early, target-oriented interventions and the potentially significant role of additional biomarkers in preventing the progression of asymptomatic cirrhosis. Prior studies discussed the notable success of various machine-learning-based approaches like Random Forest, Gradient Boosting, Ensemble Learning, and others in increasing the accuracy with which the stages of disease progression are predicted. For example, the LivMarX model achieved an accuracy of up to 86% for predicting different stages of cirrhosis based on a combination of biomarkers and optimization techniques (Kamath et al., 2024). Other models suggested that longitudinal models outperformed other cross-sectional models in accurately detecting disease progression (Hanif et al., 2022).

Millions live with cirrhosis worldwide, and it remains a leading cause of death every year. The effects on patients" quality of life following late diagnosis of cirrhosis can be dire and place a huge burden on the health sector. Furthermore, improper management of the disease may lead to serious complications, such as advanced liver failure, liver cancer, and other comorbidities (Hanif et al., 2022). New artificial intelligence and machine-aided processes enable much finer accuracy in determining the stage of the disease and are immensely beneficial in reducing complications, promoting early diagnosis, and improving patient management. The ability of this technology to offer a serious advancement in the management of cirrhosis is most felt in areas where modern imaging methods are seldom available (Topcu et al., 2024). This research aims to develop an efficient and accurate model for predicting early liver cirrhosis by employing advanced machine learning algorithms. It seeks to improve prediction accuracy by combining intelligent feature selection and model optimization approaches to create models that are not only highly efficient but also practical for implementation in real clinical settings. The major aim of the study is to devise a model for prediction of stage of PBC that is accurate, generalizable, and efficient using advanced techniques of machine learning. Some cutting-edge work presented therein involves, but is not restricted to, tuning of model hyperparameters via advanced optimization methods of Optuna, feature selection algorithms, such as RFECV to identify crucial disease progress variables. A further significant aspect in the study includes the use of rich and varied data composed of clinical and laboratory data drawn from credible sources. The evaluation of model performance metrics such as accuracy, precision, recall, F1-score, and AUC is performed in a very detailed way so as to allow transparency in the evaluation of the quality of predictions. This paper is organized as follows: the first part introduces the research and its various objectives; the second part broaches the research background and pinpoints the weaknesses of previous studies; the third part describes the research methodology regarding the dataset, preprocessing techniques and machine learning algorithms used; the penultimate section conveys all the experimental results and critically evaluates the performance of various models; and finally, the last part deliberates and draws its conclusions in respect of the findings obtained, drawing comparisons with previous studies, scrutinizing the implications of the results, providing an overview of the contributions made, and suggesting future areas of research. In this study, such a constructive approach enhances the efforts toward improving the prediction of cirrhotic liver disease risk while further enhancing the development of AI in aiding diagnostic medicine.

2- Theoretical Foundations and Research Background

In very recent times, prognosis and evaluation of liver diseases have made remarkable advancements. Cirrhosis often deteriorates into liver failure, requiring transplants in

many cases, often due to chronic liver insult. Making an accurate diagnosis of the stage of liver cirrhosis and tracking the patients' progress remains among the greatest challenges of medicine. Addressing these difficulties straightaway impacts treatment strategies and the potency of medical involvement. In the past years, machine learning methods have emerged as a contemporary remedy for prognosticating the diverse phases of liver cirrhosis. These algorithms identify clinically pertinent traits that describe singular patient characteristics through exhaustive data examination. Table 1 briefly summarizes related research on predicting the stages of liver cirrhosis and contrasts assorted methods. This table comprises the titles of the reports, aims, datasets, machine learning algorithms, and key outcomes of each analysis. An inspection of this background reveals that machine learning designs such as Random Forest, Support Vector Machine, and amalgamated tactics, exploiting an assortment of datasets and sundry optimization techniques, have been successfully applied and have achieved meaningful accuracy in prognosticating the phases of liver cirrhosis. This data furnishes worthwhile insights into the strengths and shortcomings of preceding studies and helps pinpoint existing research gaps.

Table 1. Research background

Authors	***Article Title***	***Goals***	***Model used***	***Dataset***	***Conclusion***
Konerman et al. (2019)	Machine learning models to predict disease progression among veterans with hepatitis C virus	Predict cirrhosis progression in CHC patients	Cox models and boosted-survival-tree model	Veterans' Health Administration (72,683 individuals)	The longitudinal boosted survival tree model achieved superior concordance (0/774) and AuROC in prediction compared to cross-sectional models, demonstrating higher reliability in long-term forecasts.
Topcu et al. (2024)	Machine Learning-Based Analysis and Prediction of Liver Cirrhosis	Early detection of liver cirrhosis	Random Forest, Logistic Regression, AdaBoost, k-Nearest Neighbors	Open-access liver cirrhosis dataset	The Random Forest model achieved high accuracy (~98%), demonstrating superior performance in early cirrhosis prediction. Precision, recall, and F1-score were not explicitly reported.
Bhardwaj et al. (2024)	Improving Prognostic Prediction of Cirrhosis Using an Optimized Ensemble Machine Learning Approach	Enhance prediction of cirrhosis prognosis	Ensemble model integrating Gradient Boosting, Random Forest, and Decision Trees	Multisource liver disease datasets	The ensemble models improved prediction accuracy and generalizability, making significant advances in reliability and forecasting. While specific metrics such as accuracy, precision, and recall were not directly reported, overall improvements were observed.
K et al. (2024)	Stage Prediction of Liver Cirrhosis Disease using Machine Learning	Determine stages of liver cirrhosis	Support Vector Machine, Random Forest, Gradient Boosting	Dataset with 418 records and 20 attributes	Random Forest was among the models with the highest accuracy (~97%), achieved through feature engineering and cross-validation. Precision, recall, and F1-score for the Random Forest model are not specified.
Kamath et al. (2024)	LivMarX: An Optimized Low-Cost Predictive Model Using Biomarkers for Interpretable Liver Cirrhosis Stage Classification	Stage liver cirrhosis using biomarkers	Random Forest (optimized with Genetic Algorithm and GridSearchCV)	Comprehensive dataset of 424 patients	LivMarX achieved over 86% accuracy after optimization, with an AUC of 0/95. The model demonstrated high cost-effectiveness for accurately staging cirrhosis in the absence of imaging. Precision, recall, and F1-score were not reported.
(Elmasinejad and Golabpour, 2024)	Predicting Liver Fibrosis Severity Using Machine Learning Models	Development of a machine learning model for diagnosing fatty liver using demographic information and hematology tests	Support Vector Machine (SVM) with Radial Kernel	Data from 1,078 patients referred to Imam Reza Hospital	The model achieved 93/55% accuracy on the training data and 78/62% on the test data, outperforming six comparable algorithms.
Jamadar et al. (2023)	Cirrhosis Disease Prediction Using Machine Learning	Using machine learning methods to	Different machine	Data from patients with	The proposed model demonstrated high accuracy in predicting the stages of cirrhosis.

		predict liver cirrhosis	learning algorithms	physiological characteristics associated with cirrhosis	
Hanif and Khan (2022)	Liver Cirrhosis Prediction Using Machine Learning Approaches	Predict liver cirrhosis stages	Support Vector Machine, Decision Tree, Random Forest	Liver Cirrhosis dataset (418 records)	Random Forest achieved an accuracy of ~97%, demonstrating reliability and robustness in phase-wise predictions of liver cirrhosis. Precision, recall, and F1-score were not reported.
Sidana et al. (2022)	Liver Cirrhosis Stage Prediction Using Machine Learning: Multiclass Classification	Predicting the stage of liver cirrhosis in patients using machine learning algorithms	Artificial Neural Network, Random Forest , Logistic Regression, Support Vector Machine , KNN, Decision Tree , Naive Bayes	Data from patients with liver cirrhosis	The Artificial Neural Network (ANN) demonstrated the best performance with high accuracy, while the RF+MI feature selection method showed a slight improvement over the standard Random Forest (RF) model.

The studies discussed in Table 1 delineate just some of the many advances in the use of machine learning algorithms in predicting the stages of liver cirrhosis. However, one of the main gaps identified there was the significant delay in consideration of imbalanced data sets and excessive focus on a single performance metric, such as accuracy, for model evaluation. The studies by Bhardwaj et ub. and Sidana et ub., while dealing with random forest or SVM, do not appease the challenge of imbalanced dataset(s), and they wholly rely on a single evaluation criterion, such as accuracy, thus not completely evaluating models one through other proper performance criteria such as Precision, Recall, and F1 Score. Such excessive focus on accuracy alone results in a very skewed perspective on their prediction capabilities, since such models often guarantee high-performance measures yet produce very poor results on overweighted classes. Another very important limitation discussed in Table 1 is their use of unoptimized models and poorly defined feature sets. For example, models like Random Forests and SVM have been applied, ill as the studies by Hanif and Khan, and Jamadar et al., did not apply state-of-the-art optimization techniques that would potentially improve model performance, structure feature selections, and reduce the framework of their studies, thus precluding meaningful generalization and accuracy of their interpretations. In the contrary, the current paper uses a rather spirited approach by using advanced machine learning algorithms guaranteeing accuracy in predictions and correcting the data imbalance, with the models being subjected to various acute evaluations by areas such as accuracy, precision, recall, F1 score, and ROC-AUC, which is possible to ascertain an appropriate and transparent evaluation of the models' performances addressing fundamental gaps in prior research and leading the investigation towards more reliable and generalized results.

Moreover, a large number of studies will focus only on one model, with limited analysis of the effects of combinations of algorithms or full comparisons between the efficiency of techniques. The novel methodology presented in this paper serves as an ensemble framework to enrich predictive technology, apply advanced feature selection techniques, optimize model computational costs, and improve the implementation of models openly in the real world, all of which are overly venturous in previous studies, such as the LivMarX (Kamath et al., 2024). Finally, this research makes a significant contribution to advancing existing methods by focusing on early-stage liver cirrhosis prediction, presenting a comprehensive optimization framework, thoroughly analyzing model performance indicators, and utilizing diverse and extensive datasets. Through the articulation of emerging and current research gaps, as well as the modest input of novelties, this will provide a further route for an exhaustive yet accurate approach to be developed in this area.

3- Research Method

The goal of this study was to use machine learning algorithms to predict the stage of primary biliary cholangitis (PBC) in patients. The main objective is to use the machine learning model to accurately predict the stage of the disease using medical and laboratory data. The dataset used in this study was derived from a clinical investigation of PBC patients conducted at the Mayo Clinic and supplemented by a publicly available dataset released on the Kaggle platform, which included numerous original features. After data analysis and feature selection, key variables were identified using recursive feature elimination with cross-validation (Priyatno Widiyaningtyas, 2024). During the preprocessing stages, correlation analysis was performed, and the SMOTE

method was applied to address class imbalance. Additional steps included handling missing values, and encoding categorical features (Khan & Hoque, 2020). Twelve machine learning algorithms were evaluated for modeling purposes: Decision Tree, Random Forest, Extra Tree, Gradient Boosting, AdaBoost, XGBoost, LightGBM, Logistic Regression, Support Vector Machine (SVM), k-Nearest Neighbors (KNN), Naive Bayes, and CatBoost. The Optuna optimization framework was used to fine-tune the hyperparameters of all models in such a way as to provide the best performance (Jeganathan et al., 2024). The performance of the models was assessed against four main metrics: accuracy, precision, recall, and F1-score (Fazel & Foing, 2024). In addition, the ROC curve and AUC values are used for more details regarding the model performance. All other steps of this study were done using the Python programming language with its corresponding libraries.

3-1- Data Source

The data set used in this study was extracted from the Cirrhosis Prediction Dataset, which is publicly available on the Kaggle platform. It includes information of patients with PBC, collected over ten years in a clinical study carried out at the Mayo Clinic. In this study, 420 patients diagnosed with PBC were identified as eligible to participate in a randomized, controlled trial of the drug D-penicillamine. Of these, 312 patients obtained consent to participate in the randomized clinical trial, their records had a minimal loss. There were also 112 other eligible patients who were not trial participants, who did allow for basic information and survival follow-ups to be recorded; 6 out of these 112 patients were lost from follow-up soon after diagnosis, so data on 106 remained. Thus, the total number of patients entered in the dataset is 418 (Fedesoriano,2021).

3-2- Dataset Features

The data used in this study include comprehensive information from patients with PBC. The dataset initially comprised 20 features, which are presented in Table 2.

Table 2. Variables Description

Feature Name	***Description***	***Type***	***Values/Unit***
ID	Unique identifier for each patient	Categorical	Numeric
N_Days	Number of days between registration and the earlier of death, transplantation, or study analysis time	Numeric	Days
Status	Status of the patient	Categorical	C (Censored), CL (Censored due to liver tx), D (Death)
Drug	Type of drug administered	Categorical	D-penicillamine, Placebo
Age	Age of the patient	Numeric	Days
Sex	Gender of the patient	Categorical	M (Male), F (Female)
Ascites	Presence of ascites	Categorical (Binary)	N (No), Y (Yes)
Hepatomegaly	Presence of hepatomegaly	Categorical (Binary)	N (No), Y (Yes)
Spiders	Presence of spiders	Categorical (Binary)	N (No), Y (Yes)
Edema	Presence of edema	Categorical	N, S, Y
Bilirubin	Serum bilirubin	Numeric	mg/dl
Cholesterol	Serum cholesterol	Numeric	mg/dl
Albumin	Serum albumin	Numeric	gm/dl
Copper	Urine copper	Numeric	µg/day
Alk_Phos	Alkaline phosphatase	Numeric	U/liter
SGOT	SGOT (serum glutamic-oxaloacetic transaminase)	Numeric	U/ml
Triglycerides	Serum triglycerides	Numeric	mg/dl
Platelets	Platelet count	Numeric	per cubic ml/1000
Prothrombin	Prothrombin time	Numeric	Seconds (s)
Stage	Histologic stage of the disease	Categorical (Ordinal)	1, 2, 3, 4

In this study, the target variable was defined as Stage, representing a disease stage that ranges from 1 to 4. The aim is to model the Stage variable in relationship to the other features in the data set. The ID column was ruled out of the analysis simply because it works as a patient identifier and provides no substantial contribution to prediction.

3-3- Data Cleaning

The cohort included 424 patients with PBC data collected as part of a Mayo Clinic clinical trial. Of those, the final analysis was based on 312 samples. In the first step of cleaning the data, the ID column, which was judged not relevant to the target variable, was deleted as it would not contribute to prediction. In addition, missing values in features with limited incompleteness were substituted with the mean value for less impact on the modeling. Out of the 424 data points, 112 pertained to patients who did not participate in the randomized tests and had incomplete information. Out of these, six samples were excluded shortly after data collection due to critical missing information. According to strict sampling standards, the information from the remaining 112 non-participating patients had to be rejected because of poor quality. This left 312 samples that were complete and of good quality for analysis. Data cleaning allowed such preparation, producing better quality data for the predictions.

3-4- Correlation Analysis

Correlation analysis was conducted to identify linear relationships between variables in the dataset. The primary

purpose of this analysis was to determine variables with a significant impact on the target variable and to eliminate those with redundant or weak associations with other variables. In this study, a correlation matrix, visualized using a heatmap, was employed to illustrate the relationships between variables.

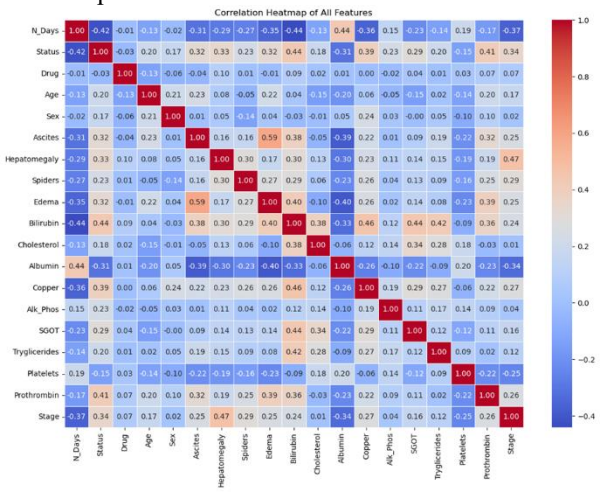

Figure 1. Correlation Heatmap

From the correlation analysis, no variables exhibited high correlation with other variables (greater than 0/8 or less than -0/8). The highest positive correlation found is between the Copper and Bilirubin (about 0/46), indicating no removal of features for redundancy because of excessive correlation. Furthermore, it is found that the independent variable (Stage) correlates positively with Hepatomegaly (about 0/47), thus this variable is important in predicting the stage of the disease. In this regard, all the features were retained for modeling since they provide independent and informative information. Such independence can be expected to add strength to model value.

3-5- Feature Selection

Therefore, feature selection becomes a big step for preprocessing data to enhance the performances of machine learning classifiers and reduce computational complexity. The dataset initially had many primary features, but some of them had bad correlations with the target variable or brought more noisy and redundant information. To extract important features, RFECV was used. RFECV is a very efficient recursive feature elimination mechanism (Thambawita et al., 2020) that starts by training the model with all features available, estimates the importance of each individual feature in terms of importance score such as those derived from feature importance or model coefficients, and then removes one feature at a time, retraining the model at each iteration. The process continues until all possible combinations of features have been tried. It implements cross-validation to find the best set of features. The other applications of cross-validation are to make the dataset as many segments as needed, then evaluate the model performances for each feature combination. Finally, RFECV was used to optimize feature selection based on model performance during cross-validation. In addition to evaluating model performance, this technique effectively eliminates irrelevant features, selecting the minimum number of features necessary to make accurate predictions. In this study, a total of 14 features were identified as the most informative from the initial set: N_days, status, drug, age, bilirubin, cholesterol, albumin, copper, alk_phos, sgot, triglycerides, platelets, and prothrombin. These selected features were found to significantly contribute to the prediction of disease stages. The removal of non-essential features reduced model complexity while improving model estimation accuracy and computational efficiency. Figure 2 illustrates the significance of these features in this study.

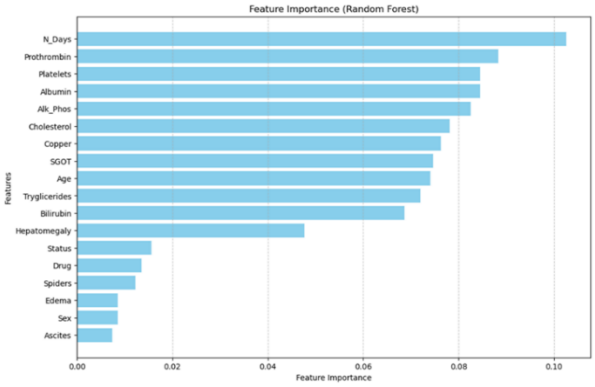

Figure 2. Feature Importance

3-6- Data Normalization

The MinMaxScaler is used to scale data for SVM (Support Vector Machine) and KNN (K-Nearest Neighbors) algorithms (Ali, 2022). This choice is made because these algorithms are generally sensitive to feature scaling. For SVM algorithms, to determine the separating hyperplane, the feature values are being used; whereas KNN uses feature values to compute distances amongst samples. Thus, features in varied scales could significantly affect the models' performance. The MinMaxScaler scales every feature to a fixed-range value, usually ranging between 0 and 1, on an equivalent scale. The formula for MinMaxScaler is:

$$x\_scaled=(x-x\_min)/(x\_max-x\_min) \quad (1)$$

In this formula:
xscaled is the normalized (scaled) feature value.
x is the original value of the feature.
xmin is the smallest value of the feature in the dataset.
xmax is the largest value of the feature in the dataset.

3-7- Data Balance

One of the major challenges outlined in this study was the distribution of samples into the different classes with unequal frequency. From the data distribution, it has been noted that there were only 16 samples at Stage I, while there were 97 samples at Stage II, 109 samples at Stage III, and more than to bring the order at the top. This imbalance causes the machine learning algorithms to converge toward the large classes, thus reducing any learning focused on the smaller classes, like stage I. This will probably have the effect that the model identifies the classes having more samples correctly, while disregarding or misclassifying the classes that have very few samples.

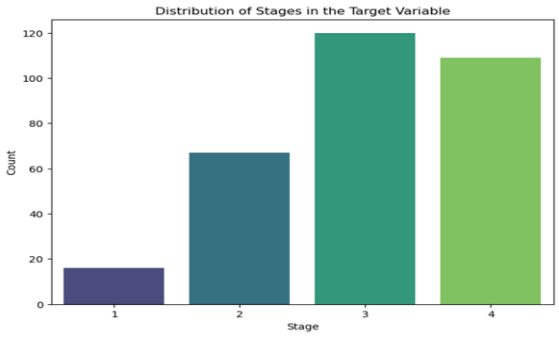

Figure 3. Distribution of Stages

The SMOTE method was used to increase the number of samples belonging to the minority class in the data set to remove imbalance namely synthetic minority over-sampling technique. It constructs synthetic instances and follows the following steps:

1. A random sample from the minority class is chosen as a reference sample.
2. Using the KNN algorithm (usually with K = 5) several nearest neighbors from the same minority class, are identified.
3. SMOTE generates new synthetic examples in feature space. This is achieved by selecting at random one of the nearest neighbors and by creating a new sample at a point in-between the reference sample and the chosen neighbor.

The formula used to compute the interpolation between the two samples is expressed as:

$$X\_new = X\_sample + gap \times (X\_neighbor - X\_sample) \quad (2)$$

Here, Xsample stands for the reference sample, Xneighbor for one of the nearest neighbors, and Gap for some random number in the range (0, 1). The dataset in this research was divided into two parts: training 70% of the data and using 30% for the encoding models' performance evaluation.

3-8- Machine Learning Algorithms

For predicting the stage of PBC in this study twelve different machine learning algorithms were used. These algorithms were used to identify the best-performing model that would predict the disease stages with the highest accuracy. The hyperparameters of each algorithm were optimized using the Optuna tool. Optuna is a dynamically designed hyperparameter optimization tool to automatically find the best values for model parameters (Akiba et al., 2019). Like others, efficiently finds the best hyperparameter configurations with advanced search techniques like Tree-structured Parzen Estimator (TPE) and Random Search. By running several tests and comparing how models perform, this tool minimizes the time to gain optimality. The table below provides the list of 12 machine learning algorithms, operational mechanisms, and the optimized values achieved using Optuna:

Table 3. Machine learning algorithms used and optimized hyperparameter values

Algorithm	Method	Optimal hyperparameters
Decision Tree	The algorithm applies successive splitting of the data into either two or more subsets. At every stage, one feature which works best for data splitting is selected according to certain criteria, some of which are Gini Index and Entropy(Mienye & Jere, 2024).	max_depth=32, min_samples_split=8
Random Forest	This algorithm, using a combination of multiple decision trees to reduce data variance, trains each tree on a random subset of the data and obtains its final output by following the majority voting rule in the case of classification, or averaging in the case of regression (Schonlau & Zou, 2020).	n_estimators=331, max_depth=8
Extra Trees	It operates similarly to Random Forest but uses random values instead of optimal values for node splitting. This approach reduces variance and results in faster model training (Geurts et al., 2006).	n_estimators=373, max_depth=14
Gradient Boosting	To build weak models (decision trees) one after the other, correcting the mistakes done by the previous model. The aim is to gradually minimize model errors and boost performance with each step (Biau & Cadre, 2017).	n_estimators=191, learning_rate=0/02662

AdaBoost	This algorithm iteratively trains weak models (small decision trees) and assigns greater weight to misclassified samples at each step to create a stronger final model (Ding et al., 2022).	n_estimators=162, learning_rate=0/54684
XGBoost	An optimized version of Gradient Boosting that reconciles the conflicts between solving the execution speed and the execution accuracy by analyzing operations in parallel and using more efficient algorithms. This optimization method can address large amounts of information and diversity (Bentéjac et al., 2020).	n_estimators=162, learning_rate=0/54684
LightGBM	An optimized Boosting algorithm that grows leaves instead of levels. This method is suitable for large-scale, high-dimensional data and provides faster performance compared to other Boosting algorithms (Ke et al., 2017).	n_estimators=329, num_leaves=210, learning_rate=0/1247
CatBoost	A fast and efficient Boosting algorithm optimized for categorical data, which automatically encodes categorical values. This method requires fewer parameter adjustments compared to other Boosting algorithms (Dorogush et al., 2018)	iterations=435, depth=9, learning_rate=0/2872
Logistic Regression	A method for data classification using a linear model computes the probability of the data belonging to different classes using the logistic (sigmoid) function. It is well suited to low-dimensional datasets (Starbuck, 2023).	C=0/1228
Support Vector Machine	This algorithm finds an optimal hyperplane to separate classes in the feature space. Using the RBF kernel, it maps data to a higher-dimensional space, enabling nonlinear separation (Shmilovici, 2023).	C=459/87, gamma=0/0573, kernel='rbf'
K-Nearest Neighbors	The prediction takes into account the distance of the other instances from the input data. The majority class among the k nearest neighbors is considered for predicting the class of the novel sample (Halder et al., 2024).	n_neighbors=3
Naive Bayes	A probabilistic model based on Bayes' theorem. This algorithm assumes complete independence between features and is well-suited for low-dimensional and categorical data (Pajila et al., 2023).	Lacks suitable hyperparameters for optimization.

To evaluate the performance of machine learning models in this study, five key metrics were used: accuracy, precision, recall, F1-score, and the area under the receiver operating characteristic curve (ROC-AUC). These metrics are defined based on the concepts of True Positive (TP) and True Negative (TN) for correct predictions, and False Positive (FP) and False Negative (FN) for incorrect predictions.

Table 4. Evaluation indicators for machine learning models

index	definition	Formula
Accuracy	The ratio of correct predictions (both positive and negative) to the total number of samples.	(TP+TN)/(TP+FP+FN+TN)
Precision	The ratio of correctly predicted instances for a class to all instances predicted as that class.	TP/(TP+FP)
Recall	The ratio of correctly predicted instances for a class to all actual instances of that class.	TP/(TP+FN)
F1 Score	The harmonic mean of Precision and Recall, balancing the trade-off between the two metrics.	(2×Precision×Recall)/(Precision+Recall)

The ROC-AUC metric measures the performance of a classification model at all threshold levels and illustrates how well the model is at distinguishing between classes; thus, it shows how well the model can predict the different stages of the disease. The ROC curve is created by plotting the value of false positive rate (FPR) vs true positive rate (TPR) for different thresholds and area under this curve is known as the AUC. AUC can be understood as the higher the better: The closer the AUC value is to 1, the better. In order to test the generalizability of the model and verify that it performed successfully regardless of the dataset with 5-Fold Cross-Validation was performed. In this method, the data set is split into five equal parts. At each iteration, one of its sections is considered as test data, while the other four sections are used as training data. This is done five times to guarantee that each batch is tested once. Finally, the overall performance of the model is reported as the mean values of all the evaluation metrics across all iterations.

4- Results

In this section, the results of the machine learning models are presented and analyzed. The Python programming language was utilized for this study, and all models were executed on a system equipped with an Intel Core i7-

13700H processor, 16GB of RAM, and Python version 3/12. The following outlines the performance results of the models.

Table 5. Comparison of results

Model	Accuracy	Precision	Recall	F1 Score
CatBoost	0/7708	0/7688	0/7708	0/7519
Extra Trees	0/7569	0/7636	0/7569	0/7400
LightGBM	0/7292	0/7182	0/7292	0/7126
Random Forest	0/7222	0/7146	0/7222	0/7085
Gradient Boosting	0/7153	0/7057	0/7153	0/7017
XGBoost	0/7083	0/6973	0/7083	0/6993
Support Vector Machine	0/7014	0/6895	0/7014	0/6847
K-Nearest Neighbors	0/6667	0/6569	0/6667	0/6531
Decision Tree	0/6319	0/6222	0/6319	0/6252
AdaBoost	0/5972	0/5961	0/5972	0/5949
Logistic Regression	0/5139	0/5131	0/5139	0/5094
Naive Bayes	0/5347	0/5129	0/5347	0/5083

Evaluation Results of the Machine Learning Models. From all the above models, the CatBoost model presented the best performance results with an accuracy equal to 0.7708, precision equal to 0.7688, recall equal to 0.7708 and F1-score equal to 0/7519. These results show that CatBoost not only predicts accurately, but have a good mean for all metrics. This is because of its strong architecture for processing categorical data and its automatic hyperparameter tuning. Second only to CatBoost, the Extra Trees model achieved an accuracy score of 0/7569 and an F1-score of 0/7400. Through a series of randomized decision trees, this model provided a somewhat good performance and outperformed other models, such as LightGBM, Random Forest. Similarly, LightGBM also performed well but produced an accuracy of 0/7292 and an F1-score of 0/7126, highlighting its ability to process complex and high-dimensional data. Random Forest and Gradient Boosting ranked next, achieving accuracies of 0/7222 and 0/7153, respectively. The two models presented balanced trade-off between all metrics but were not able to beat CatBoost and Extra Trees. The XGBoost model followed closely, with an accuracy of 0.7083 and an F1-score of 0.6993, highlighting the competitive nature of Boosting-based algorithms. On the other hand, SVM (accuracy = 0/7014) and KNN (accuracy = 0/6667) exhibited less accuracy in predicting disease stages and hence this concludes their lower efficiency in dealing with complex data processing compared to the Boosting models. Relative to simpler models like Decision Tree and AdaBoost, these models exhibited moderate performance. The Decision Tree performed with an accuracy of 0.6319. Standard decision trees are underfitting models, and their performance is less than ensemble trees (i.e. Random Forest, Extra Trees). The AdaBoost model also performed relatively weakly, with 0/5972 accuracy. Logistic Regression and Naive Bayes performed the worst, respectively. As a result of Logistic Regression (accuracy of 0/5139) and Naive Bayes (accuracy of 0/5347), we could claim that these simple models do not provide the ability to process and predict complex, multidimensional data effectively in this study.

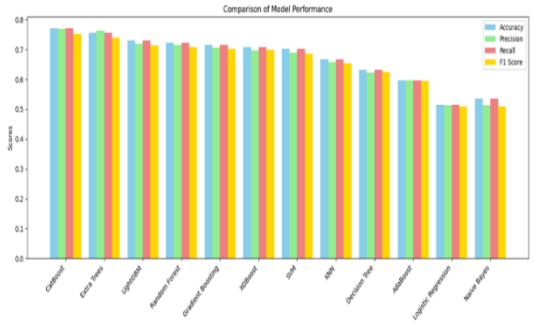

Figure 4. Performance of various machine learning models

In the figure 4, we can see the comparison of various machine learning models by accuracy, precision, recall, and F1-score. Overall, ensemble learning based models like CatBoost, Extra Trees and LightGBM performed the best. The outcomes show that advanced models based on Boosting and ensemble approaches using decision trees excel in performing accurate prediction of disease phases whilst preserving an optimal equilibrium among evaluation metrics compared with alternative models.

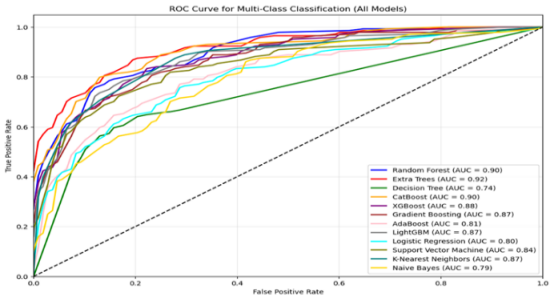

Figure 5. ROC curve

Figure 5 shows ROC curves and AUC for PBC prediction. The performance of models in separating classes is visualized using the ROC curve, whereas the AUC is another robust measure of model performance. If we observe the graph, it is clear that Extra Trees model gave the highest AUC 0/92. The CatBoost and Random Forest both gave AUC 0/90. Gradient Boosting, LightGBM, and SVM also performed distinctively well, attaining AUC values ranging over 0/87 and 0/88. Conversely, simpler

models like Decision Tree and Naive Bayes had lower performance, with AUC of 0/74 and 0/79, respectively. From the results collectively, we see that ensemble-based models, specifically Extra Trees and CatBoost perform better than simple models in class separation. This shows that implementing complex algorithms in highly intricate medical problems, like predicting the progression of diseases, increases the performance of models significantly.

5- Discussion and Conclusion

Results from our study indicate that with the application of modern machine learning algorithms, like CatBoost and Extra Trees, it is possible to obtain accurate predictions of PBC stages. CatBoost was found to be the best of all models achieved, having produced an accuracy of 0/7708 and AUC of 0/90.) Extra Trees also performed well in classifying complex datasets, reaching an AUC of 0/92. These findings underscore the significance of ensemble-based methods in achieving superior predictive accuracy compared to simpler models . This research represents significant advances in machine learning techniques as compared to previous studies. A notable limitation in earlier studies was the use of unoptimized models with poorly defined feature sets. For example, while Hanif and Khan (2022) and Jamadar et al. (2023) employed algorithms such as Random Forest and SVM, they did not utilize advanced optimization techniques to enhance model performance or implement robust feature selection methods.This poor optimization restricted the generalizability and accuracy of their results. As a result, the present study led to stable prediction performance across all metrics by using an automated hyperparameter optimization method (Optuna) and an advanced feature selection method (Recursive Feature Elimination with Cross-Validation). Another key difference in prior studies is their inadequate consideration of imbalanced datasets. When models are evaluated in such manner, it may lead to misleading results because the model can easily predict the majority class while performing poorly on minority classes. For example, Bhardwaj et al. (2024) and Sidana et al. (2022), which did not evaluate models properly and did not point out that a better evaluation is characterized by the reporting of important imbalanced evaluation metrics such as precision, recall, F1-score, etc. This contrast with this study, which used standard performance metrics to give transparent and comprehensive evaluation of model quality. SMOTE process was applied to supporter model to solve imbalance class, while RFECV was used to find out 14 essential features to both reduce model complexity and improve quality. These developments make this study unique compared to previous studies that did not properly resolve dataset imbalance or attempted basic feature selection methodology. Here, we showcase the possibilities of advanced machine learning models and structured optimization techniques in predicting medical health outcomes. Ensemble methods like CatBoost and Extra Trees are better suited for these medical datasets with high-dimension characteristics due to their superior performances compared to simple methods Logistic Regression and Naive Bayes. Such findings provide a direction for future research using larger and diverse data sets having imaging data to create models more accurate with clinical relevance.

Based on the findings of this review, several recommendations are made to enhance and direct future research. The first improvement could be using more and diverse data to provide machine learning models capable of getting generalized. The combination of data from multiple clinical sources with covariate data available in existing datasets could provide more robust results. Secondly, it is proposed that some of the more sophisticated preprocessing methods such as feature engineering and nonlinear transformations might reveal hidden patterns in the data that could improve the model's performance. In future works, DNN (Deep Neural Networks) or LSTM (Long-Short Term Memory) could potentially replace GBDTs with a better prediction performance for the disease stages. More sophisticated ensemble techniques (hybrid Voting and Stacking) are additionally likely to enhance the prediction capabilities due to the synergy of the respective standalone models. On the clinical side, a more detailed analysis of the importance and sensitivity of the model features must facilitate the identification of pertinent biomarkers associated with the prediction of disease stage; each of the findings will assist clinical applications. Finally, validating the above machine learning models against clinical data from hospitals and clinics would make various algorithms appropriate for use as well as more reliable. Initiating these efforts may lead to the development of more accurate and reliable models of timely diagnostics and improved care of patients.

References

[1] M. A. Konerman et al., “Machine learning models to predict disease progression among veterans with hepatitis C virus,” PLOS ONE, vol. 14, no. 1, p. e0208141, Jan. 2019, doi: https://doi.org/10.1371/journal.pone.0208141.

[2] Ahmet Ercan Topcu, Ersin Elbasi, and Yehia Ibrahim Alzoubi, “Machine Learning-Based Analysis and Prediction of Liver Cirrhosis,”Jul.2024,doi:https://doi.org/10.1109/tsp63128.2024.10605929.

[3] E. B. Tapper and N. D. Parikh, “Diagnosis and Management of Cirrhosis and Its Complications: A Review,” JAMA, vol. 329, no. 18, pp. 1589–1602, May 2023, doi: https://doi.org/10.1001/jama.2023.5997.

[4] R. Wei et al., "Clinical prediction of HBV and HCV related hepatic fibrosis using machine learning," vol. 35, pp. 124–132, Sep. 2018, doi: https://doi.org/10.1016/j.ebiom.2018.07.041.

[5] C. Labenz et al., "Structured Early detection of Asymptomatic Liver Cirrhosis: Results of the population-based liver screening program SEAL," Journal of Hepatology, vol. 77, no. 3,pp.695–701,Sep.2022,doi: https://doi.org/10.1016/j.jhep.2022.04.009.

[6] E. Forte et al., "Top-Down Proteomics Identifies Plasma Proteoform Signatures of Liver Cirrhosis Progression," Molecular & Cellular Proteomics, pp. 100876–100876, Nov. 2024, doi: https://doi.org/10.1016/j.mcpro.2024.100876.

[7] Varshni Premnath and Shanthi Veerappapillai, "Unveiling miRNA–Gene Regulatory Axes as Promising Biomarkers for Liver Cirrhosis and Hepatocellular Carcinoma," ACS Omega, vol. 9, no. 44, pp. 44507–44521, Oct. 2024, doi: https://doi.org/10.1021/acsomega.4c06551.

[8] L. Wang et al., "Impact of Asymptomatic Superior Mesenteric Vein Thrombosis on the Outcomes of Patients with Liver Cirrhosis," Thrombosis and Haemostasis, vol. 122, no. 12, pp. 2019–2029, Sep. 2022, doi: https://doi.org/10.1055/s-0042-1756648.

[9] Md. Nahid Hasan, T. Ahmed, Md. Ashik, Md. Jahid Hasan, Tahaziba Azmin, and J. Uddin, "An Analysis of Covid-19 Pandemic Outbreak on Economy using Neural Network and Random Forest," Journal of Information Systems and Telecommunication (JIST), vol. 11, no. 42, pp. 163–175, Jun. 2023, doi: https://doi.org/10.52547/jist.34246.11.42.163.

[10] Sudiksha Kottachery Kamath, Sanjeev Kushal Pendekanti, and D. Rao, "LivMarX: An Optimized Low-Cost Predictive Model Using Biomarkers for Interpretable Liver Cirrhosis Stage Classification," IEEE Access, vol. 12, pp. 92506–92522,Jan.2024,doi:https://doi.org/10.1109/access.2024.3422451.

[11] I. Hanif and M. M. Khan, "Liver Cirrhosis Prediction using Machine Learning Approaches," 2022 IEEE 13th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), Oct. 2022, doi: https://doi.org/10.1109/uemcon54665.2022.9965718.

[12] D. Bhardwaj, G. Kaur, and G. L. Babu, "Improving Prognostic Prediction of Cirrhosis Using an Optimized Ensemble Machine Learning Approach," pp. 1–6, Aug. 2024, doi: https://doi.org/10.1109/ciscon62171.2024.10695979.

[13] Bhanu Prakash K, Vennela D, Dhana Lakshmi N, and Siva Priyanka S, "Stage Prediction of Liver Cirrhosis Disease using Machine Learning," pp. 1–6, Aug. 2024, doi: https://doi.org/10.1109/icecsp61809.2024.10698096.

[14] Rauf Jamadar, Harsh Uike, and Vaishali Jabade, "Cirrhosis Disease Prediction Using Machine Learning," pp. 515–520, Dec.2023,doi:https://doi.org/10.1109/icacctech61146.2023.00090.

[15] Tejasv Singh Sidana, S. Singhal, S. Gupta, and R. Goel, "Liver Cirrhosis Stage Prediction Using Machine Learning: Multiclass Classification," Lecture notes in networks and systems, pp. 109–129, Nov. 2022, doi: https://doi.org/10.1007/978-981-19-3679-1_9.

[16] Arif Mudi Priyatno and Triyanna Widiyaningtyas, "A SYSTEMATIC LITERATURE REVIEW: RECURSIVE FEATURE ELIMINATION ALGORITHMS," JITK (Jurnal Ilmu Pengetahuan dan Teknologi Komputer), vol. 9, no. 2, pp. 196–207,Feb.2024,doi: https://doi.org/10.33480/jitk.v9i2.5015.

[17] S. I. Khan and A. S. M. L. Hoque, "SICE: an improved missing data imputation technique," Journal of Big Data, vol. 7, no. 1, Jun. 2020, doi: https://doi.org/10.1186/s40537-020-00313-w.

[18] S. Jeganathan, A. R. Lakshminarayanan, S. Parthasarathy, A. Abdul Azeez Khan, and K. J. Sathick, "OptCatB: Optuna Hyperparameter Optimization Model to Forecast the Educational Proficiency of Immigrant Students based on CatBoost Regression," Journal of Internet Services and Information Security, vol. 14, no. 3, pp. 111–132, Aug. 2024, doi: https://doi.org/10.58346/jisis.2024.i2.008.

[19] F. Fazel and B. Foing, "Evaluating Classification Algorithms: Exoplanet Detection using Kepler Time Series Data," arXiv (CornellUniversity),Feb.2024,doi: https://doi.org/10.48550/arxiv.2402.15874.

[20] Fedesoriano, "Cirrhosis Prediction Dataset," www.kaggle.com.https://www.kaggle.com/fedesoriano/cirrhosis-prediction-dataset

[21] V. Thambawita et al., "An Extensive Study on Cross-Dataset Bias and Evaluation Metrics Interpretation for Machine Learning Applied to Gastrointestinal Tract Abnormality Classification," ACM Transactions on Computing for Healthcare, vol. 1, no. 3, pp. 1–29, Jul. 2020, doi: https://doi.org/10.1145/3386295.

[22] P. J. Muhammad Ali, "Investigating the Impact of Min-Max Data Normalization on the Regression Performance of K-Nearest Neighbor with Different Similarity Measurements," ARO-THE SCIENTIFIC JOURNAL OF KOYA UNIVERSITY, vol. 10, no. 1, pp. 85–91, Jun. 2022, doi: https://doi.org/10.14500/aro.10955.

[23] K. K, U. K, S. A, and A. Kumar, "Predicting Student Performance for Early Intervention using Classification Algorithms in Machine Learning," Journal of Information Systems and Telecommunication (JIST), vol. 9, no. 36, pp. 226–235,Oct.2021,doi: https://doi.org/10.52547/jist.9.36.226.

[24] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A Next-generation Hyperparameter Optimization Framework," arXiv (Cornell University), Jul. 2019, doi: https://doi.org/10.48550/arxiv.1907.10902.

[25] I. D. Mienye and N. Jere, "A Survey of Decision Trees: Concepts, Algorithms, and Applications," IEEE access, pp. 1–1,Jan.2024,doi: https://doi.org/10.1109/access.2024.3416838.

[26] A. Jafarnejad, A. Rezasoltani, and A. M. Khani, "Comparative Analysis of Machine Learning Algorithms in Predicting Jumps in Stock Closing Price: Case Study of Iran Khodro Using NearMiss and SMOTE Approaches," Iranian Journal of Finance, vol. 9, no. 3, pp. 27–54, 2025, doi: 10.30699/ijf.2025.491324.1496.

[27] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," Machine Learning, vol. 63, no. 1, pp. 3–42, Mar. 2006, doi: https://doi.org/10.1007/s10994-006-6226-1.

[28] G. Biau and B. Cadre, "Optimization by gradient boosting," arXiv.org, Jul. 17, 2017. https://arxiv.org/abs/1707.05023 (accessed Apr. 24, 2024).

[29] Y. Ding, H. Zhu, R. Chen, and R. Li, "An Efficient AdaBoost Algorithm with the Multiple Thresholds Classification," Applied Sciences, vol. 12, no. 12, p. 5872, Jun. 2022, doi: https://doi.org/10.3390/app12125872.

[29] C. Starbuck, "Logistic regression," in *Springer eBooks*, pp. 223–238, 2023. doi: 10.1007/978-3-031-28674-2_12.
[30] A. Jafarnejad Chaghoshi, A. Rezasoltani, and A. M. Khani, "Unleashing the Power of Ensemble Learning: Predicting National Ranks in Iran's University Entrance Examination," Industrial Management Journal, vol. 16, no. 3, pp. 457–481, 2024, doi: 10.22059/imj.2024.381521.1008178.
[31] G. Ke et al., "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," hal.science, Dec. 04, 2017. https://hal.science/hal-03953007 (accessed Mar. 27, 2023).
[32] A. V. Dorogush, V. Ershov, and A. Gulin, "CatBoost: gradient boosting with categorical features support," arXiv.org, Oct. 24, 2018. https://arxiv.org/abs/1810.11363
[33] Motiei, M., Khani, A. M., & Beyrami, S. (2021). The effect of green supply chain and green human resource management on environmental performance: The mediating role of green innovation. Logistics Thought, 20(77), 165–197. https://doi.org/10.22034/lot.2021.96691
[34] A. Jafarnjad, A. Rezasoltani, and A. M. Khani, "Analyzing and Predicting Hiring Decisions Using Machine Learning and Deep Learning," Journal of Public Administration, vol. 17, no. 2, pp. 295–327, 2025, doi: 10.22059/jipa.2025.390322.3649.
[35] Jafarnejad Chaghoshi, A., Khani, A. M., & Rezasoltani, A. (2024). Risk modeling in banking services for the blind using fuzzy FMEA and graph neural network (GNN). Journal of Industrial Management Perspective, 14(4), 223–255. https://doi.org/10.48308/jimp.14.4.223
[36] P.J.Beslin Pajila, B. Gracelin. Sheena, A. Gayathri, J. Aswini, M. Nalini, and Siva Subramanian R, "A Comprehensive Survey on Naive Bayes Algorithm: Advantages, Limitations and Applications," Sep. 2023, doi: https://doi.org/10.1109/icosec58147.2023.10276274.
[37] J. Kasubi, M. D. Huchaiah, I. Gad, and M. K. Hooshmand, "A Comparison Analysis of Conventional Classifiers and Deep Learning Model for Activity Recognition in Smart Homes based on Multi-label Classification," Journal of Information Systems and Telecommunication (JIST), vol. 12,no46pp127–137,Jun.2024,doi: https://doi.org/10.61186/jist.36294.12.46.127.
[38] A. Rezasoltani, A. Jafarnejad, and A. M. Khani, "A voting-based hybrid machine learning model for predicting backorders in the supply chain," Journal of Decisions and Operations Research, vol. 10, no. 1, pp. 194–213, 2025, doi: 10.22105/dmor.2025.511401.1924.

Journal of Information Systems and Telecommunication
Vol.13, No.3, July-September 2025, 177-188
http://jist.acecr.org
ISSN 2322-1437 / EISSN:2345-2773

Resolving Class Imbalance in Medical Classification: Technique Comparison and Performance Evaluation

Abdallah Maiti[1*], Mohamed Hanini[1], Abdallah Abarda[2]

[1].Laboratory of Computing, Networks, Mobility and Modelling (IR2M) FST, Hassan First University of Settat, Morocco
[2].Laboratory LM2CE, Faculty of Economic Sciences and Management, Hassan First University of Settat, Morocco

Received: 16 Mar 2025/ Revised: 07 Aug 2025/ Accepted: 06 Sept 2025

Abstract

The problem of unbalanced data is a common one in medical diagnostics. This problem can reduce the accuracy of classification models and affect the validity of results. The aim of our paper is to compare several techniques for correcting class imbalances in medical datasets and to evaluate the impact of these techniques on machine learning performance.

In our paper, we used an imbalanced dataset to train a convolutional neural network (CNN) model. We then tested correction techniques such as sampling and cost-sensitive learning. Finally, we used recall, precision, accuracy and F1 score to evaluate the model's performance.

The results show that the use of correction techniques led to a significant improvement in the performance of the classification model. The cost-sensitive learning technique gave the best results, particularly for the detection of minority classes. This method increased the weight of classification errors associated with minority classes, thus improving the detection of critical cases. The results of this study underline the importance of dealing with imbalances in the data to improve the performance of classification models in the medical field. The use of methods such as cost-sensitive learning not only improves model performance, but also enables more reliable decisions to be made, which is essential for ensuring more accurate diagnoses and better quality of care.

Keywords: Data Imbalance; Techniques for Resolving Data Class Imbalance; Oversampling; Cost-Sensitive learning, Convolutional Neural Networks; Classification; Model Performance; Medical Diagnostics.

1- Introduction

The text must be in English. Authors whose English The problem of imbalanced data represents a big challenge in machine learning, particularly in critical fields such as healthcare, finance, cybersecurity and other. It occurs when certain classes in a data-set are underrepresented relative to others, causing predictive models to disproportionately favor the majority classes. In domains such as fraud detection, where fraudulent transactions represent only a small proportion of the data, models often struggle to identify these minority instances, favoring normal transactions instead [1], [2]. Similarly, rare diseases in medical diagnosis or infrequent cyberattacks in cybersecurity are often misclassified due to their limited representation in training datasets [3]. Addressing this imbalance is essential to improve prediction accuracy and ensure fairness across all classes. Classical ML algorithms, such as logistic regression and decision trees assume a balanced distribution of data, a condition that is rarely met in real-world applications. Therefore, various methods have been developed to mitigate biases caused by imbalance.

Different techniques such as oversampling, undersampling, cost-sensitive learning, and ensemble methods have shown promise in improving minority class detection while maintaining overall model performance [4] solve this problem. Imbalance can take different forms depending on the data type. In binary classification, a single minority class often poses a problem, as seen in rare disease diagnosis or fraud detection, where models tend to favor the majority class. Approaches such as SMOTE address this problem by generating synthetic examples for underrepresented categories [5]. In multi-class scenarios, imbalance arises when multiple classes are unequally represented, as seen in multi-stage disease diagnosis. In such cases, advanced techniques such as One-vs-One (OvO) and One-vs-Rest (OvR), as well as ensemble methods, are needed to ensure balanced performance across classes [4].

Beyond accuracy, traditional evaluation metrics often fail to capture a model's ability to identify minority classes.

✉ **Abdallah maiti**
abdallah.maiti@uhp.ac.ma

Metrics like precision, recall, and F1-score are more appropriate for binary imbalances, while G-mean and Matthews correlation coefficient (MCC) provide a more balanced evaluation for multi-class problems [6]. These metrics are crucial for evaluating mitigation strategies and ensuring fair representation of all classes.

Despite the progress made, significant challenges persist in combating class imbalance. Low performance on minority classes, inadequacy of conventional metrics, and difficulties in generalizing to unseen data are among the main obstacles. The choice of the most effective method depends on the specific context, including the severity of the imbalance and the area of application. In complex scenarios, hybrid approaches that combine data-level and algorithmic methods are often required [7].

Recent empirical investigations have underscored the efficacy of hybrid methodologies that integrate oversampling techniques, such as Synthetic Minority Over-sampling Technique (SMOTE), deep neural networks, and reinforcement learning to more proficiently address imbalance within intricate datasets. These adaptive methodologies are structured to correspond with the data's inherent architecture, thereby enhancing performance while concurrently mitigating the risk of overfitting [8]. Furthermore, the intensifying focus on algorithmic equity, especially within critical sectors like healthcare, necessitates the rectification of biases stemming from underrepresented classes, as such biases may precipitate significant diagnostic inaccuracies [8].

In the domain of natural language processing, contemporary scholarship regarding the Central Kurdish language has demonstrated that the qualitative balancing of corpora is imperative for guaranteeing the dependability of morphosyntactic frameworks, particularly in contexts characterized by limited resources [9].

These theoretical frameworks have significantly guided the methodological framework of the current investigation. The proposed architecture is predicated on a convolutional neural network (CNN), augmented by rebalancing methodologies such as Synthetic Minority Over-sampling Technique (SMOTE), classification paradigms including One-vs-One (OvO) and One-vs-Rest (OvR), alongside cost-sensitive learning and the ensemble-based Bagging methodology. This comprehensive framework aims to enhance the identification of minority classes while maintaining consistent overall efficacy.

In addition to extant research, this investigation enriches the academic discourse by amalgamating all four methodologies within a cohesive framework explicitly tailored for medical imaging applications. It delineates a multiclass classification protocol that tackles the infrequency of clinical cases, the hierarchical organization of disease stages, and the imperatives of algorithmic equity. This contribution is particularly notable in its deployment for the automated identification of diabetic retinopathy utilizing retinal imagery, where advanced stages of the condition are frequently underrepresented and challenging to discern.

The overall aim of this research is to develop a robust classification system capable of accurately identifying rare stages of diabetic retinopathy (DR). More specifically, the study seeks to determine the most effective techniques for correcting class imbalance in medical imaging; to evaluate the impact of these techniques using appropriate performance metrics such as recall and F1-score; and to offer practical recommendations for high-stakes domains where misclassification can significantly affect decision-making. The article is structured as follows: Section 2, “Materials and Methods,” describes the dataset, the CNN architecture, and the imbalance-handling strategies implemented; Section 3, “Results,” presents the model’s performance under various conditions; Section 4, “Discussion,” interprets the findings and considers methodological trade-offs; and finally, Section 5, “Conclusion,” summarizes the main contributions and proposes future research directions.

2- Materials and Methods

In our article, we investigate various techniques to address class imbalance in multi-class classification tasks. Our goal is to classify retinal images according to the severity stages of diabetic retinopathy (DR), a serious eye disease resulting from prolonged hyperglycemia. The dataset used is from the Kaggle platform and consists of five classes, ranging from “No DR” (absence of disease) to “Proliferative DR” (advanced and severe form of the disease). Unlike other studies that apply imbalance correction techniques without sufficient justification, we propose a systematic approach tailored to imbalanced and unstructured data, particularly images. Our aim is to scientifically identify the most effective techniques to overcome this challenge and evaluate their impact on the performance of classification models. To achieve this, we used a convolutional neural network (CNN)-based model, known for its ability to automatically extract complex features from images. We evaluate several class rebalancing techniques, including undersampling, oversampling, One-vs-Rest (OvR) and One-vs-One (OvO) approaches, cost-sensitive learning, and ensemble bagging (Fig.1). Models are trained and evaluated on balanced datasets using these techniques. The evaluation phase relies on standard metrics such as accuracy, precision, recall, and F1 score, which are derived from the confusion matrix. This comprehensive approach enables a precise analysis of the influence of the applied imbalance resolution techniques on the performance of the CNN-based model and provides insights into effectively addressing imbalances in image classification tasks.

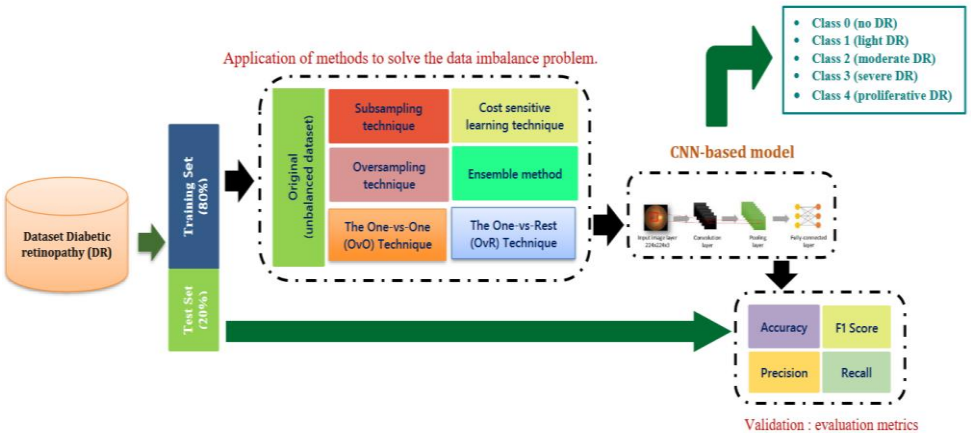

Fig. 1. Architecture of the proposed diagnostic system

2-1-Dataset Description

The dataset used in our paper and obtained from the Kaggle platform [29], consists of a total of 92702 retinal images distributed across five classes, each representing a stage of diabetic retinopathy (DR). The dataset (Table 1) exhibits a significant class imbalance, with the majority class, "No DR," comprising approximately 77.8% of the total samples. In contrast, the more severe stages, such as "Severe DR" and "Proliferative DR," are severely underrepresented, together accounting for less than 5.1% of the dataset.

Table 1. Distribution of Retinal Images Across Diabetic Retinopathy Classes

Class	*Description*	*Samples*	*Percentage*
Class 0	No DR	72102	77.8%
Class 1	Mild DR	8772	9.5%
Class 2	Moderate DR	7135	7.7%
Class 3	Severe DR	2328	2.5%
Class 4	Proliferative DR	2365	2.5%
Total		92702	100%

This imbalance poses challenges for model training, as predictive models tend to favor the majority class, leading to poor detection rates for minority classes. Addressing this issue is critical to improving diagnostic accuracy, particularly for the advanced stages of DR. Techniques such as oversampling, undersampling, and algorithmic adjustments are essential to mitigate this problem and ensure balanced and robust model performance.

2-2-Model Architecture

To solve the problem of multi-class classification of diabetic retinopathy, we have developed a model based on a convolutional neural network (CNN). This type of model is particularly effective for image analysis, thanks to its ability to automatically extract complex features while reducing the need for manual data pre-processing (Fig. 2).

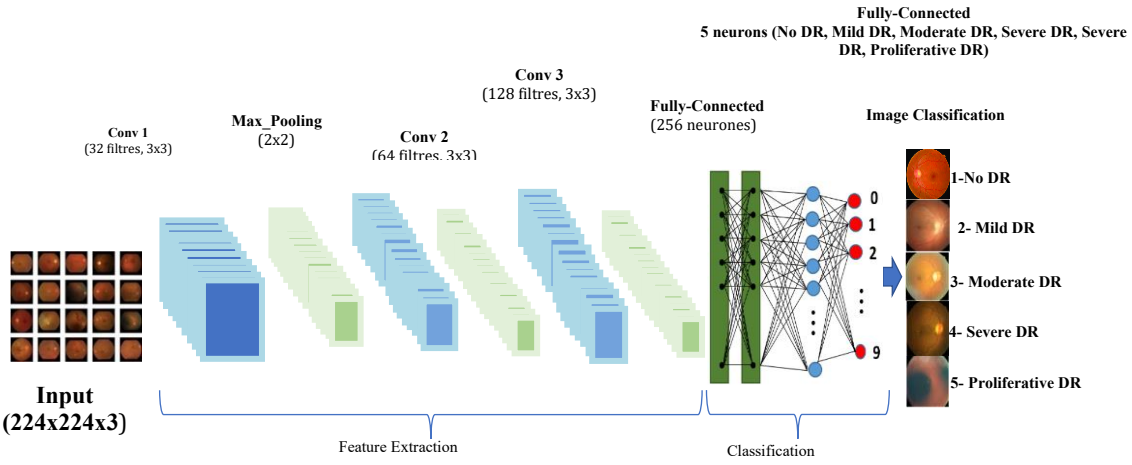

Fig. 2. Architecture of our CNN-based classification model

The architectural framework of the model is predicated upon a convolutional neural network (CNN) organized into three primary phases: feature extraction, dimensionality reduction, and classification. It consists of three convolutional layers designed to extract fundamental features from images, succeeded by pooling layers that facilitate dimensionality reduction and bolster the robustness of the model. Ultimately, two fully connected layers conclude the multi-class classification process. Methodologies such as dropout regularization, in conjunction with non-linear activation functions (ReLU and Softmax), augment the model's efficacy and generalizability in the identification of diabetic retinopathy.

2.2.1. Three Convolutional layers

The proposed model employs a triad of convolutional layers to derive critical features from retinal imagery. The initial layer utilizes 32 filters, succeeded by 64 filters in the subsequent layer and 128 filters in the final layer. Each filter executes a convolution operation utilizing a 3x3 kernel, thereby facilitating the identification of distinct patterns, including anomalies or textures that are characteristic of retinopathy.

2.2.2. Pooling layers (2x2)

After each convolutional layer, pooling layers with a 2x2 size kernel are applied to reduce the dimensionality of the data. This process limits over-fitting while reducing computational costs. The max-pooling method is used, selecting the maximum value in each analyzed region. This ensures that the most dominant and significant features of the images, essential for classification, are retained, while simplifying the representations learned by the model.

2.2.3. Two Fully Connected layers

The model comprises two fully-connected layers that ensure the finalization of the classification. The first layer, made up of 256 neurons, combines the features extracted from the convolutional and pooling layers. It uses a ReLU (Rectified Linear Unit) activation function, well known for its ability to introduce non-linearity, essential for modeling complex relationships between features. This function also prevents the effect of gradient saturation, which promotes efficient convergence during training.

The output layer comprises 5 neurons, corresponding to the five severity classes of diabetic retinopathy. A Softmax activation function is applied to transform the outputs of this layer into normalized probabilities, allowing direct interpretation of predictions as probabilities belonging to each class. This configuration is particularly well-suited to multi-class classification, guaranteeing well-calibrated output and a sum of probabilities equal to 1.

2.2.4. Regulation

A dropout mechanism (with a rate of 0.5) is implemented subsequent to the fully connected layers in order to mitigate the probability of overfitting by sporadically deactivating certain neurons throughout the training process. This methodology entails the random inactivation of 50% of the neurons at each iteration during training, thereby diminishing the model's excessive dependence on particular neurons.

This architecture integrates efficient convolutional layers for the automatic extraction of pertinent features alongside dense layers designated for classification. Such a framework is exceptionally well-suited for medical image analysis endeavors, owing to its capacity to capture intricate details while simultaneously minimizing the necessity for manual pre-processing.

2-3-Techniques for Correcting Data Imbalances

Addressing data imbalance is crucial for improving the performance of machine learning models. The different approaches to tackle this issue can be represented in three categories: data-driven approaches, algorithmic approaches, and specific approaches designed for multi-class problems.

2.3.1. Data-Based Methods

Data-based approaches involve the direct manipulation of datasets to balance the distribution of classes before model training.

a-Sub-Sampling

The technique of subsampling, unlike oversampling, involves reducing the number of samples from majority classes to balance their proportion relative to minority classes (Fig. 3). This technique is typically implemented by randomly removing examples from the dominant class [10]. Subsampling has several advantages, including model simplification by reducing the total volume of data, which also lowers computational costs. However, this technique has several notable drawbacks. Removing samples from majority classes can lead to the loss of crucial information [11]. Furthermore, the random selection of samples to be removed may not accurately reflect the actual distribution of the data, potentially affecting model performance, especially when the data is heavily unbalanced [12].

b-Oversampling

Oversampling methodologies pertain to the deliberate augmentation of sample quantities from minority classes to rectify their inadequate representation in imbalanced datasets (Fig 3). Among the preeminent methodologies, the Synthetic Minority Oversampling Technique (SMOTE) is particularly noteworthy for its capability to produce

synthetic instances through linear interpolation of existing samples within the minority class [6],[7]. This approach enhances the representation of underrepresented classes while concurrently maintaining the diversity and structural integrity of the dataset.

The practice of oversampling confers several advantages. It mitigates the model's bias towards majority classes and enhances its generalization capabilities. These benefits culminate in an improved recognition of underrepresented classes, particularly in scenarios where imbalances may precipitate erroneous predictions [13]. Furthermore, by infusing greater variability into minority classes, methodologies such as SMOTE enable machine learning algorithms to more effectively discern the unique characteristics of rare instances. Nonetheless, oversampling is not devoid of limitations. The artificial augmentation of samples may heighten the risk of overfitting, especially when synthetic instances exhibit insufficient diversity or replicate patterns that do not accurately reflect authentic data [14]. In addition, this escalation in data volume may incur elevated computational costs, particularly with extensive datasets, due to the supplementary resources necessitated for the generation and processing of synthetic instances [15]. Recent studies suggest improvements to SMOTE, such as K-Means SMOTE or Borderline-SMOTE, which specifically target critical regions near decision boundaries to maximize the efficiency of oversampling [16]. These variants aim to reduce drawbacks while fully exploiting the potential of minority classes in unbalanced contexts.

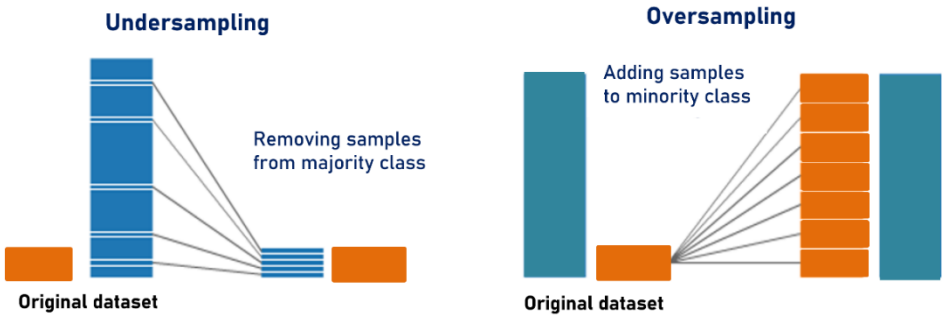

Fig. 3. Representative diagram of the two techniques: subsampling and oversampling

2.3.2. Algorithmic Approaches

Algorithmic approaches directly modify learning algorithms to deal with data imbalance, without modifying the distribution of classes in the ensemble.

a- Cost-Sensitive learning

This methodology modifies the loss function of machine learning algorithms by allocating enhanced significance to minority classes. This approach is predicated on augmenting the weight of errors pertinent to these classes, in accordance with their under-representation (Fig. 4). In a dataset wherein a class constitutes 10% of the samples, misclassification errors for that class may be amplified by a factor that corresponds to the degree of imbalance, thus escalating the associated penalty [17].

This methodology proves to be particularly efficacious in critical domains, such as the detection of rare diseases, the prevention of financial fraud, or the prediction of failures in intricate systems. It substantially contributes to the reduction of classification errors in under-represented classes, while simultaneously preserving the equilibrium of overall model performance [18]. In addition, by integrating these weights into algorithms, cost-sensitive learning augments model sensitivity and precision for imbalanced datasets.

Nonetheless, the efficacy of this methodology is profoundly contingent upon the meticulous calibration of the weights allocated to various classes. Insufficient calibration may result in an inverse imbalance, thereby impairing performance on majority classes or diminishing the overall effectiveness of the model [19]. Therefore, methodologies such as adaptive weight optimization or the employment of specific metrics, including the ROC curve or F-measure, are frequently advocated to guarantee balanced performance.

Fig. 4. Operating principle of the cost-sensitive learning method

b- Ensemble Methods

Ensemble techniques, such as Bagging and Boosting, combine the predictions of multiple models to enhance overall performance and reduce bias toward majority classes (Fig. 5). Bagging (Bootstrap Aggregating) uses random sampling with replacement to train several independent models, whose predictions are then aggregated, improving model robustness and stability [20]. Boosting, on the other hand, progressively corrects the errors of successive models by assigning higher weights to misclassified examples, thereby increasing overall accuracy, particularly on minority classes [21]. These techniques are particularly effective for datasets with a high degree of imbalance, as they address the weaknesses of individual models by improving the recognition of under-represented classes. By introducing diversity into data subsets and combining the strengths of several models, they also promote better generalization. Furthermore, recent variants, such as AdaBoost-SAMME or Gradient Boosting with SMOTE, have demonstrated their effectiveness in handling complex imbalances by adjusting weights for minority classes [23].

Nevertheless, the execution of these methodologies may prove to be intricate and computationally intensive, particularly in the context of boosting. The latter necessitates meticulous calibration of hyperparameters, including but not limited to learning rate and quantity of estimators, to mitigate the risk of overfitting and to guarantee optimal efficacy [24]. In spite of these obstacles, their capacity to enhance performance in scenarios characterized by imbalanced data renders them indispensable instruments in domains such as finance, healthcare, and predictive analytics.

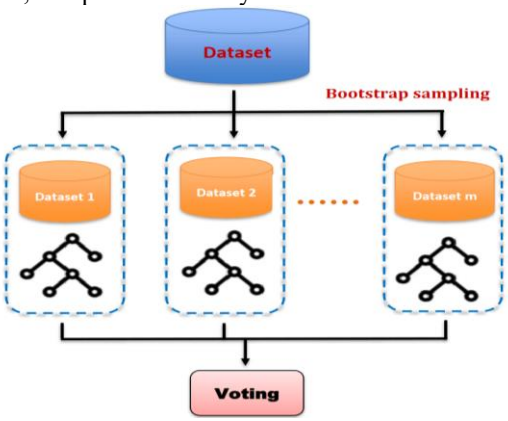

Fig. 5. Operating principle of the Bagging ensemble method

2.3.3. Specific Techniques for Multi-Class Problems

In multi-class problems, where multiple categories are present, data imbalance poses additional challenges. Classical approaches can be adapted, but specific approaches such as One-vs-Rest (OvR) and One-vs-One (OvO) (Fig. 6) are often used.

a- One-vs-Rest (OvR)

OvR also known as One-vs-All, decomposes a multi-class problem into several binary classification problems. For each class, a binary classifier is trained, treating this class as positive and grouping all other classes as negative. For instance, in a five-class problem, OvR requires the creation of five binary models, each optimized to distinguish a specific class [25],[26]. Notable advantages of this technique include its simplicity of implementation and its ability to provide independent evaluations for each class. These features make it particularly suited to contexts where granular predictions are essential, such as in image recognition or recommender systems [25],[26]. Additionally, the OvR technique is compatible with a wide range of learning algorithms, such as support vector machines (SVMs) and logistic regression, making it a versatile option.

However, this technique has important limitations. It can become biased when classes grouped as negative are highly imbalanced, which can impair model performance on minority classes [27]. Furthermore, OvR does not account for the complex relationships and possible interdependencies between different classes, limiting its ability to capture global patterns or subtle correlations in the data [28].

Recent work proposes extensions to mitigate these limitations, such as integrating adaptive weights to balance negative classes or using hybrid techniques that combine OvR with dimensionality reduction methods like linear discriminant analysis. These improvements aim to enhance the robustness and accuracy of this technique in unbalanced multi-class classification contexts.

b- One-vs-One (OvO)

The OvO technique treats each pair of classes separately, creating a binary classifier for each combination of two classes. For example, for a problem with five classes, the OvO results in ten binary classifiers, one for each pair of classes [25],[26].

This approach is particularly useful for data with complex class relationships, as each classifier focuses on only two classes at a time. This reduces the impact of majority classes, as each binary classifier works on data balanced between the two classes concerned. However, the computational complexity is high. The number of classifiers to be trained increases quadratically with the number of classes, which can lead to considerable computational costs and implementation difficulties in contexts with a large number of categories [27].

Data imbalance correction methods offer a variety of solutions tailored to specific application needs. Data-driven techniques, such as oversampling and undersampling, directly modify the class distribution, while algorithmic approaches, such as cost-sensitive learning and ensemble methods, adjust the algorithms to compensate for biases [28]. In multi-class problems, specific techniques such as

OvR and OvO are used to handle the additional complexity associated with multiple classes. The choice of the optimal method depends on the context of use, the nature of the data and technical constraints. It is often advisable to combine several approaches to maximize model performance while minimizing imbalance bias [25],[26].

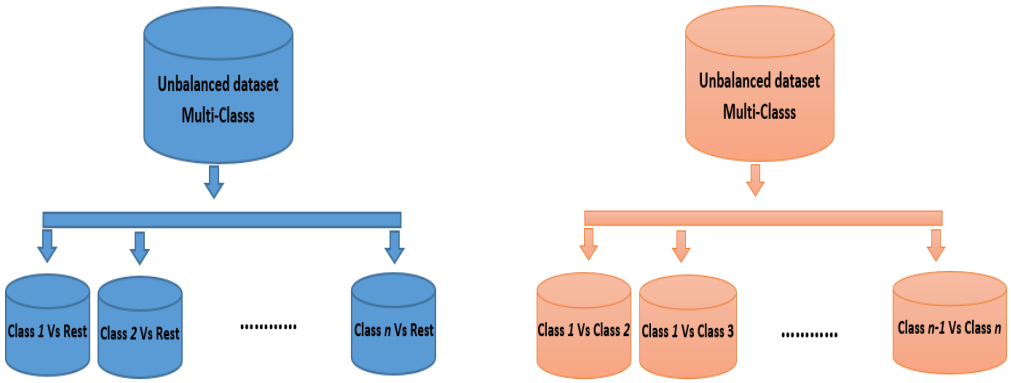

Fig. 6. Representation of the "One-vs-Rest"(OvR) and "One-vs-One"(OvO) techniques

3- The Results

Unbalanced multi-class classification is a major challenge, due to the complexity of interactions between classes and the difficulty of assessing model performance. Unlike binary classification, this context requires advanced approaches to effectively manage imbalance while improving prediction accuracy.

In our research, we apply and evaluate various data rebalancing techniques, such as oversampling, undersampling, one-to-one and one-to-all approaches, ensemble methods such as Bagging, and cost-sensitive learning. The aim is to identify the best method for boost the performance of artificial intelligence models in this complex context.

3-1-Subsampling

Sub-sampling is a methodological approach aimed at equilibrating the distribution of classes by diminishing the magnitude of the majority class, which is accomplished through the stochastic elimination of samples from this class to render it congruent with the quantity of the minority class. In the present investigation, each class was systematically curtailed to 2328 samples, in alignment with the size of the minority class. While this methodology serves to mitigate the bias in favor of the majority class, it engenders a considerable loss of information, which may adversely influence the overall efficacy of the model, as delineated in Table 2.

The implementation in Python employs the resample function from the sklearn.utils library to perform subsampling on the majority class, thereby modifying its size to correspond with that of the minority class. Subsequent to the subsampling procedure, the equilibrated dataset is preserved in the variables X_resampled and y_resampled, rendering it suitable for utilization in model training. The outcomes of this methodology are illustrated in Table 2.

Table 2. Overall performance obtained using the sub-sampling technique

Metric	***Global values***
Accuracy	82.64 %
Precision	88.94 %
Recall	82.15 %
F1-Score	80.51 %

3-2-Oversampling

To improve the representation of minority classes in unbalanced datasets, the SMOTE (Synthetic Minority Oversampling Technique) technique was used. SMOTE generates synthetic samples for under-represented classes by creating intermediate points between existing instances of the same class [30],[22]. This rebalances the distribution of classes and mitigates biases linked to data imbalance when training machine learning models.

In Python, SMOTE is implemented using the SMOTE class in the imbalanced-learn library (imblearn).

The resulting oversampling led to a significant improvement in overall performance, although there remains a risk of model overfitting due to the generation of synthetic samples. The performance results obtained after applying SMOTE are presented in Table 3.

Table 3. Overall performance obtained using the oversampling technique

Metric	*Global values*
Accuracy	87.09 %
Precision	84.36 %
Recall	81.78 %
F1-Score	83.05 %

The F1-Score of 83.05%, which combines two parameters: precision and recall into a single metric, provides a more comprehensive evaluation in handling imbalanced data. Although the accuracy is relatively high at 87.09%, it is not the most reliable metric for this type of task due to the potential influence of class imbalance. The moderate recall and F1-Score suggest that, while oversampling improved class distribution, the model may exhibit overfitting, limiting its ability to generalize effectively to unseen data.

3-3-Cost-Sensitive learning

Cost-sensitive learning is an effective technique for managing class imbalance without directly modifying the data distribution. It assigns weights proportional to the inverse of class frequency, thus giving greater importance to minority classes during training. In this study, weights were calculated as in Table 4.

Table 4. Weight of diabetic retinopathy classes

Class	*Weight*
Class 0	1
Class 1	(72 102/8 772) ≈ 8.22
Class 2	(72 102/7 135) ≈ 10.10
Class 3	(72 102/2 328) ≈ 31.00
Class 4	(72 102/2 365) ≈ 30.49

The weights were integrated into the SparseCategoricalCrossentropy loss function of TensorFlow/Keras through the class_weight parameter, thereby facilitating the equilibrium of performance between predominant and subordinate classes. This methodology dynamically modifies the error magnitude associated with under-represented classes, obviating the necessity for direct alterations to the training dataset, and empowers the model to more effectively manage class imbalances during the training process.

In this specific implementation, the class_weight parameter is employed to modulate the significance of each class, thereby compensating for imbalances while preserving the integrity of the data itself. Metrics such as Accuracy, Precision, Recall, and F1-Score were computed on the test dataset to appraise the model's efficacy. Upon the completion of training the CNN-based model, its performance was evaluated utilizing the test data (refer to Table 5). The findings illustrate that this methodology proficiently reconciles overall accuracy and performance across all classes, including minority classifications, thereby mitigating the adverse effects of data imbalance on predictive quality. The model accomplished an Overall Accuracy of 91.09%, indicative of its capacity to render precise predictions across all classifications. The F1-Score, a composite metric amalgamating precision and recall, attained 92.79% for the "No DR" classification, underscoring the model's dependability in identifying this category. Below is a comprehensive delineation of the performance metrics for each class:

No DR: The model exhibited outstanding performance in this category, attaining a Precision of 91.14%, a Recall of 94.49%, and an F1-Score of 92.79%, which exemplifies its robust capability to accurately recognize instances devoid of diabetic retinopathy. Mild DR: This classification similarly exhibited elevated performance, achieving a Precision of 93.27%, a Recall of 91.95%, and an F1-Score of 92.60%, signifying a well-balanced aptitude for detecting mild cases. Moderate DR: With a Precision of 91.95%, a Recall of 93.24%, and an F1-Score of 92.59%, the model effectively identified moderate cases with negligible errors. Severe DR: The performance of the model was somewhat diminished for this classification, achieving a Precision of 88.26%, a Recall of 82.86%, and an F1-Score of 85.47%, which reflects certain challenges in differentiating severe cases. Proliferative DR: This minority classification attained a Precision of 85.88%, a Recall of 83.72%, and an F1-Score of 84.78%, demonstrating the model's capacity to address even the most formidable cases, albeit with some constraints.

Table 5. Performance obtained by applying Cost Sensitive Learning

Metric	*Overall Accuracy*	*Precision*	*Recall*	*F1-Score*
No RD	91.09 %	91.14 %	94.49 %	92.79 %
light RD		93.27 %	91.95 %	92.60 %
Moderate RD		91.95 %	93.24 %	92.59 %
Severe RD		88.26 %	82.86 %	85.47 %
Proliferative RD		85.88 %	83.72 %	84.78 %

3-4-Ensemble technique: Bagging

Bagging (Bootstrap Aggregating) was implemented in Python to handle unbalanced data sets. Four balanced subsets were created by bootstrap sampling, each subset comprising 2,328 representative samples of all classes, including minority classes, using scikit-learn's resample function. These subsets were used to independently train a CNN model, developed with TensorFlow using a defined architecture, an 'adam' optimizer, a 'categorical_crossentropy' loss function, and 'accuracy' metrics.

The predictions of the four models were aggregated by majority voting, implemented via scipy's mode function. The results obtained are presented in Table 6.

Table 6. Overall performance of the Bagging technique

Metric	***Global values***
Accuracy	83.21 %
Precision	83.49 %
Recall	83.21 %
F1-Score	83.28 %

3-5-OvR and OvO Techniques

OvR and OvO techniques are widely used strategies for handling multi-class classification problems, particularly when addressing class imbalance. In this study, these techniques were implemented in Python.

The overall performance of these two techniques is summarized in Table 7.

Table 7. Overall performance achieved using OvR and OvO techniques

Technique	***Accuracy***	***Precision***	***Recall***	***F1-Score***
OvR	84.06 %	80.35 %	83.53 %	81.91 %
OvO	79.68 %	81.65 %	84.19 %	82.90 %

The results show that the OvR technique achieves an accuracy of 84.06%, while OvO performs better in terms of precision and F1-Score, albeit with slightly lower accuracy. These two techniques are complementary, and the choice of approach will depend on the specific objectives of the model, notably between precision and recall.

4- Discussion

Table 8. presents the performance of the CNN classification model, trained on the “DR” (Diabetic Retinopathy) dataset balanced by different techniques. This table compares the results obtained with different class imbalance correction techniques, assessing their impact on four main metrics: Accuracy, Precision, Recall and F1-Score.

This comparison highlights the strengths and limitations of each technique, as well as their influence on overall model performance.

The comparative results of the different imbalance correction techniques are shown in Table 8. above. The metrics used (Accuracy, Precision, Recall and F1-Score) make it possible to evaluate the effectiveness of each technique on overall model performance.

a- Cost-Sensitive Learning Technique

The cost-sensitive learning methodology modifies the weightings assigned to each class in accordance with their prevalence, thereby effectively mitigating biases resulting from class imbalance. Among the methodologies assessed, cost-sensitive learning demonstrates the most favorable overall efficacy, yielding an accuracy of 91.09%, a precision of 90.10%, a recall of 89.25%, and an F1-score of 89.65%. This approach is particularly adept at addressing the disparate costs associated with misclassification, enabling the model to more accurately identify minority classes while preserving elevated overall precision. The exemplary outcomes of cost-sensitive learning illustrate its capacity to reconcile precision and recall, rendering this technique an outstanding selection for datasets characterized by imbalance. While the performance metrics are commendable, it is crucial to acknowledge that the dynamic recalibration of weights may incur significant computational costs, particularly when engaging with extensive datasets. Our findings regarding cost-sensitive learning align with those reported in contemporary scholarly literature, which has evidenced that this strategy stands out as one of the most efficacious for imbalanced multi-class classification challenges, as evidenced by the research conducted by Khan et al. [31]. A more recent investigation by Araf et al. [32] posits that this technique necessitates meticulous parameter optimization to circumvent computational burdens while sustaining high precision. This highlights the imperative for practitioners to diligently evaluate the trade-offs between computational expenses and performance enhancements.

b- Oversampling Technique

Oversampling, particularly using the SMOTE method, generates synthetic samples for minority classes, improving their representation during training. SMOTE achieved an accuracy of 87.09%, precision of 84.36%, recall of 81.78%, and an F1-score of 83.05%. While this method is powerful, it carries the risk of overfitting if the synthetic data does not accurately reflect the complexity of real samples.

It is important to note that the risk of overfitting can be a major issue with this approach. According to Vargas et al. [33], the generated samples may introduce unrealistic variations into the data, which could harm the model's ability to generalize. This trade-off between improving the representation of minority classes and the risk of overfitting must be carefully evaluated.

c- Bagging Technique

Bagging (Bootstrap Aggregating) significantly bolsters the reliability of predictions through the amalgamation of numerous models that have been trained on meticulously balanced subsets of the dataset. This methodology attained an accuracy rate of 87.49%, a precision level of 84.91%, a recall metric of 81.72%, and an F1-score of 83.28%. While it exhibits a marginal advantage over oversampling with respect to accuracy, the computational resources required for training multiple models may pose a limitation in environments constrained by resources. Despite the robustness of this technique, the substantial computational demands must be meticulously evaluated. As posited by Liang & Zhang [34], the process of training various models on data subsets necessitates effective resource management, which can serve as an impediment in computationally limited scenarios. Consequently, the balance between precision and computational expense must be critically assessed in professional practice.

d- Subsampling Technique

Under-sampling entails the reduction of the population of the majority class to correspond with the population size of the minority classes. This methodology yielded an accuracy rate of 82.64%, a precision rate of 88.94%, a recall rate of 82.15%, and an F1-score of 85.41%. Although this methodology facilitates the equilibrium between precision and recall, it is plagued by a considerable diminution of information, which may adversely influence the model's capacity to generalize.

The information attrition linked to under-sampling can detrimentally affect the generalization capabilities of the model, as articulated by Soleimani & Mirshahzadeh [35]. In real-world implementations, this strategy may prove to be suboptimal when substantial amounts of information are essential for the accurate prediction of infrequent occurrences, as is the case with diabetic retinopathy.

e- OvO and OvR Methods:

The One-vs-One (OvO) and One-vs-Rest (OvR) methodologies partition the multi-class classification challenge into binary subproblems. The efficacy of the OvO method is marginally inferior to that of alternative methodologies, attaining an accuracy of 79.68%, a precision of 81.65%, a recall of 84.19%, and an F1-score of 82.90%. Conversely, the OvR methodology achieves an accuracy of 84.06%, yet it remains suboptimal in performance relative to strategies such as cost-sensitive learning and oversampling. Our findings regarding OvR and OvO are in alignment with those documented in contemporary research, including the work of Chakraborty & Dey [36], which indicates that while these methodologies may be effective in certain contexts, they are generally less efficacious than approaches like cost-sensitive learning (CSL) and Synthetic Minority Over-sampling Technique (SMOTE) due to the inherent trade-offs in accuracy and computational efficiency.

Table 8. Model performance on the balanced DR dataset using different imbalance correction techniques

Correction techniques	***Accuracy***	***Precision***	***Recall***	***F1-Score***
Subsampling	82.64 %	88.94 %	82.15 %	85.41 %
Oversampling	87.09 %	84.36 %	81.78 %	83.05 %
Cost-sensitive learning	**91,09%**	**90,10%**	**89,25%**	**89,65%**
Bagging technique	87.49 %	84.91 %	81.72 %	83.28 %
One-vs-One (OvO)	79.68 %	81.65 %	84.19 %	82.90 %
One-vs-Rest (OvR)	84.06 %	80.35 %	83.53 %	81.91 %

5- Conclusion

The categorization of images depicting diabetic retinopathy poses a considerable challenge attributable to class imbalance, a widespread concern within medical applications. This manuscript conducts a comparative analysis of diverse methodologies aimed at mitigating this imbalance while simultaneously enhancing the efficacy of Convolutional Neural Network (CNN) models. The findings unequivocally indicate that the selection of correction methodologies exerts a substantial influence on model efficacy, thereby underscoring the necessity for the adoption of strategies that are specifically tailored to the contextual characteristics of the data and the distinct aims of the application.

Among the methodologies scrutinized, cost-sensitive learning emerges as the preeminent strategy. Its adaptive modulation of class weights facilitates a balanced evaluation of classification inaccuracies, culminating in enhanced performance across critical metrics (Accuracy, Precision, Recall, and F1-Score). This approach not only assures superior generalization but also yields a more precise identification of minority classes. Techniques such as oversampling and bagging also exhibited favorable outcomes, particularly in augmenting the representation of minority classes, while concurrently sustaining competitive overall performance. Nonetheless, both methodologies may engender a compromise between computational expense and precision, particularly in expansive applications. Conversely, subsampling and the One-vs-One/One-vs-Rest (OvO/OvR) techniques, although beneficial, are encumbered by intrinsic limitations, such as potential information loss or heightened complexity, rendering them less appropriate for intricate, imbalanced datasets such as those associated with diabetic retinopathy.

These observations accentuate the imperative for a comprehensive evaluation of the strengths and weaknesses inherent to each technique, with particular emphasis on the trade-offs between computational expenditure and accuracy. The outcomes further highlight the significance of implementing solutions specifically adapted to the particular constraints of the data and the objectives of the application. Future investigations should prioritize the innovation of novel methodologies that effectively manage complex, imbalanced datasets. Additionally, the exploration of hybrid models that amalgamate existing techniques should be pursued to capitalize on the synergistic strengths of each strategy. This integrative methodology would contribute to the optimization of performance by addressing the deficiencies associated with individual techniques, thereby enhancing model capabilities in regard to both accuracy and generalization.

Such a strategy would not only elevate the overall performance of models but also more effectively address the critical requirements of applications, particularly in domains such as medicine, where the robustness, fairness, and reliability of models are of paramount importance.

References

[1]. KrawczykB, B. (2016). “Learning from imbalanced data: Open challenges and future directions”. Published in Progress in Artificial Intelligence, V5(4), pp 221-232.

[2]. Haixiang, G., and al. (2017). “Learning from class-imbalanced data: Review of methods and applications”. Published in Expert Systems with Applications, v73, pp 220-239.

[3]. LemaîtreG., Nogueira, F., and Aridas, C. K(2017). « Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning”. Published in Journal of Machine Learning Research, v18(17), pp1-5.

[4]. BrancoP., Torgo, L., andRibeiro, R. P2019). A survey of predictive modeling on imbalanced domains. ACM Computing Surveys, v49(2), pp1-50.

[5]. He, H., & Garcia, E. A. (2009). Learning from imbalanced data. IEEE Transactions on Knowledge and Data Engineering, 21(9), 1263-1284.

[6]. ChawlaN. V. et al2002). SMOTE: Synthetic Minority Over-sampling Technique. Published in Journal of Artificial Intelligence Research, 16, 321-357.

[7]. Kaur, H. et al. (2019). A systematic review on imbalanced data challenges in machine learning: Applications and solutions. Published in ACM computing surveys (CSUR), 52(4), 1-36.

[8]. Abdullah, A. A., Mohammed, N. S., Khanzadi, M., Asaad, S. M., Abdul, Z. K., & Maghdid, H. S. (2025). In-depth Analysis on Machine Learning Approaches: Techniques, Applications, and Trends. ARO-THE SCIENTIFIC JOURNAL OF KOYA UNIVERSITY, 13(1), 190-202.

[9]. Sabr, S. S., Mustafa, N. S., Omar, T. S., Rasool, S. H., Omer, N. A., Hamad, D. S., ... & Maghdid, H. S. (2025). A Comprehensive Part-of-Speech Tagging to Standardize Central-Kurdish Language: A Research Guide for Kurdish Natural Language Processing Tasks. arXiv preprint arXiv:2504.19645.

[10]. Kaur, H., Pannu, H. S., and Malhi, A. K. (2019). A systematic review on imbalanced data challenges in machine learning: Applications and solutions. ACM computing surveys (CSUR), 52(4), 1-36.

[11]. LinC. C., Yen, S. J., and Lee, Y. S2017). On combining SMOTE with under-sampling: An experimental study on class imbalance problem. Published in Information Sciences, v371, 123-137.

[12]. YangC., at al. (2024). Impact of random oversampling and random undersampling on the performance of prediction models developed using observational health data. Published in Journal of big data, v11(1), 7.

[13]. Loffredo, E., Pastore, M., Cocco, S., & Monasson, R. (2024). Restoring balance: principled under/oversampling of data for optimal classification. arXiv preprint arXiv:2405.09535.

[14]. Buda , M. , Maki, A., and Mazurowski, M. A. (2018). "A systematic study of the class imbalance problem in convolutional neural networks". Neural Networks, 106, pp 249-259.

[15]. Leevy, J. L., Khoshgoftaar, T. M., Bauder, R. A., & Seliya, N. (2018). A survey on addressing high-class imbalance in big data. Journal of Big Data, 5(1), 1-30.

[16]. Han, H., Wang, W. Y., & Mao, B. H. (2005). Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning. Advances in Intelligent Computing, 878-887.

[17]. Liu, Y., Wu, T., & Yan, P. (2020). Balancing imbalanced data using adaptive synthetic sampling with feature selection. Computational Intelligence and Neuroscience, 2020, 1-11.

[18]. Longadge, R., & Dongre, S. (2013). Class imbalance problem in data mining review. arXiv preprint arXiv:1305.1707.

[19]. Brownlee, J. (2020). Imbalanced classification with Python: better metrics, balance skewed classes, cost-sensitive learning. Machine Learning Mastery.

[20]. Yadav, S., & Bhole, G. P. (2020, December). Handling imbalanced dataset classification in machine learning. In 2020 IEEE Pune Section International Conference (PuneCon) (pp. 38-43). IEEE.

[21]. Liu, L., Wu, X., Li, S., Li, Y., Tan, S., & Bai, Y. (2022). Solving the class imbalance problem using ensemble algorithm: application of screening for aortic dissection. BMC Medical Informatics and Decision Making, 22(1), 82.

[22]. Maiti, A., Abarda, A., & Hanini, M. (2022, October). A New Hybrid Artificial Intelligence Model for Diseases Identification. In The Proceedings of the International Conference on Smart City Applications (pp. 825-836). Cham: Springer International Publishing.

[23]. He, H., Garcia, E. A. (2009). "Learning from imbalanced data". In IEEE Transactions on knowledge and data engineering, 21(9), pp1263-1284.

[24]. Tanha, J., Abdi, Y., Samadi, N., Razzaghi, N., & Asadpour, M. (2020). Boosting methods for multi-class imbalanced data classification : an experimental review. Journal of Big data, 7, 1-47.

[25]. Pawara, P., Okafor, E., Groefsema, M., He, S., Schomaker, L. R., & Wiering, M. A. (2020). One-vs-One classification for deep neural networks. Pattern Recognition, 108, 107528.

[26]. Brownlee, J. (2020). One-vs-rest and one-vs-one for multi-class classification. Machine Learning Mastery.

[27]. LiQ., SongY., ZhangJ., and ShengV. S2020). « Multiclass imbalanced learning with one-versus-one decomposition and spectral clustering". Published in Expert Systems with Applications, in147, p113--152.

[28]. Chakraborty, S., & Dey, L. (2024). Multi-class Classification. In Multi-objective, Multi-class and Multi-label Data Classification with Class Imbalance: Theory and Practices (pp. 51-76). Singapore : Springer Nature Singapore.

[29]. Diabetic Retinopathy Detection data set, in kaggle.com/c/diabetic-retinopathy-detection/data

[30]. Maiti , A., Abarda, A., Hanini, M., and Oussous, A. (2024). "An Optimal Model Combining SqueezeNet and Machine Learning Methods for Lung Disease Diagnosis. Current Medical Imaging, 20(1).

[31]. Khan, A. A., Chaudhari, O., & Chandra, R. (2024). A review of ensemble learning and data augmentation models for class imbalanced problems: Combination, implementation and evaluation. Expert Systems with Applications, 244, 122778.DOI : 10.1016/j.eswa.2023.122778

[32]. Araf, I., Idri, A., & Chairi, I. (2024). Cost-sensitive learning for imbalanced medical data: A review. Artificial Intelligence Review, 57(4), 80.DOI : 10.1007/s10462-023-10652-8

[33]. Vargas, W. de, Schneider Aranda, J. A., dos Santos Costa, R., da Silva Pereira, P. R., & Victória Barbosa, J. L. (2023). Imbalanced data preprocessing techniques for machine learning: A systematic mapping study. Knowledge and Information Systems, 65(1), 31-57.DOI : 10.1007/s10115-022-01772-8

[34]. Liang, G., & Zhang, C. (2012). A Comparative Study of Sampling Methods and Algorithms for Imbalanced Time Series Classification. In AI 2012: Advances in Artificial Intelligence (pp. 637–648). Springer.DOI : 10.1007/978-3-642-35101-3_54

[35]. Soleimani, M., & Mirshahzadeh, A. S. (2023). Multi-class classification of imbalanced intelligent data using deep neural network. EAI Endorsed Transactions on AI and Robotics, 2, 1-10.DOI : 10.4108/airo.7998.

[36]. Chakraborty, S., & Dey, L. (2024). Applications of Multi-objective, Multi-label, and Multi-class Classifications. In Multi-objective, Multi-class and Multi-label Data Classification with Class Imbalance: Theory and Practices (pp. 135-164). Singapore: Springer Nature Singapore. DOI : 10.1007/978-981-97-9622-9.

Journal of Information Systems and Telecommunication
Vol.13, No.3, July-September 2025,189-209
http://jist.acecr.org
ISSN 2322-1437 / EISSN:2345-2773

Enhancing IoT Security: A Hybrid Deep Learning-Based Intrusion Detection System Utilizing LSTM, GRU, and Attention Mechanisms with Optimized Hyperparameter Tuning

Heshmat Asadi[1], Mahmood Alborzi[1*], Hesam Zandhesami[1]

[1]. Department of Management and Economics, Science and Research Branch, Islamic Azad University, Tehran, Iran

Received: 11 Jan 2025/ Revised: 04 Aug 2025/ Accepted: 06 Sep 2025

Abstract

increasing complexity and volume of threats being created and targeted at cybersecurity for the IoTs necessitate the deployment of powerful IDSs. This paper offers an innovative intrusion detection system for IoTs networks based on deep learning. The new IDS employs the Long Short-Term Memory and Gated Recurrent Unit models' strengths and an Attention Mechanism. First, the new IDS seeks to enhance the model's ability to determine critical features in a vast amount of data streams and hence improve the ability to find potential cyber threats with high accuracy. The methodological framework used in a simulation and practical experiment setting was intended to recognize the unique nature of IoTs situations. therefore, used a hybrid algorithm optimization strategy, namely Differential Evolution and Harmony Search, to optimize the model due to the extensive hyperparameter space to get the best performance results. The results obtained superior accuracy, precision, recall, and F1 measures reaching 99.87 percent, 99.84 percent, 99.85 percent, and 99.85 percent is better than the performance measures achieved by existing models. Therefore, a deep learning-based hybrid IDS confirmed the research hypothesis that this could provide the necessary and effective cybersecurity for the IoTs. It is vital to note that this paper has contributed to the research topic by showing the potential of advanced neural architectures and strategic optimization tools to address the massive and sophisticated IoTs cybersecurity issues. Future research will be addressing whether these models can be applied in more IoTs settings and whether their real-time efficiency can be improved.

Keywords: Intrusion Detection System in Internet of Things; Attention Mechanism in Deep Learning algorithm; Differential Evolution; Harmony Search.

1- Introduction

Security has become an issue of growing concern especially in Internet of Things (IoT) where the deployment of IoT networks raised new security challenges, and traditional intrusion detection systems are no longer enough, to protect dynamic and heterogeneous IoT networks. Modern cyber threats are also more advanced, and require more than traditional signature-based and anomaly-based methods, which typically have high false positives and are limited in threat coverage. With fast development of deep learning and AI, the automatic learning and behavior pattern identification by use of deep leaning and AI become the promising solutions for securing IoT intrusion detection. The development of deep learning based systems for IoT security is still quite challenging because of the significant computational constraints and the real-time processing constraints of IoT devices, as well as the adaptive requirements for resource-constrained environments, where traditional DL-based approaches are commonly known to be computationally prohibitive [1][2].

Identification of the Gap: Intrusion detection solutions face some limitations to work efficiently in terms of the unique IoT challenges such as device diversity, limited resources, and dynamic topologies. The current distance between traditional IDS functionality and the detection needs of advanced threats are especially evident in deep learning used for IoT systems [3].

It includes but is not limited to described below: lack of labeled datasets specifically targeting the complexity of IoT network traffic, the computation complexity of deep

✉ **Mahmood Alborzi**
Mahmood _alborzi@yahoo.com

learning, lack of suitable models that adapt to the complexity of the IoT environment and the variance produced by each of the more than 20 billion devices connected worldwide. Additionally, there is a considerable discrepancy in leveraging DL and AI in practical models. Whereas a growing portion of the literature focuses on developing theoretical models and algorithms, few studies focus on combining these proposals with the IoT domain. This entails a lack of validation schemas considering the flow of energy, computation capabilities, and the real-time need to process requests and requirements in IoT[3], [4].

Research Question or Hypothesis: Our research is prompted by the identified gaps in the adaptation and optimization of deep learning and artificial intelligence algorithms for integration into the Internet of Things intrusion detection systems. Thus, the primary question of our investigation is as follows:

Research Question: "How can deep learning and artificial intelligence algorithms be efficiently adopted and optimized in IoT intrusion detection patters to improve the general level of protection from sophisticated attackers, while addressing the concerns associated with the limited resources, energy efficiency and dynamical topology of Internet of Things components? ". The research question analyzes the primary areas of concern in the adaptation of DL and AI technologies, as well as the possible ways to mitigate them. The implication suggests the comprehensive understanding of the application and examination of the mentioned technology both in theory and in practice, which is the central objective and contribution of our study. Based on the research hypotheses, the notion of the hypothesis shaping our study is as follows: Hypothesis: "Designing and integrating customized solutions of deep learning and artificial intelligence to the existing intrusion detection systems by the means of optimization for the critical requirements and constrains of Internet of Things devices can significantly enhance the quality and effectiveness of the protocols through the detection rate, false positive rate and resource effectiveness metrics" . The hypothesis builds the rationale for the integration of the stated technologies as the enhancement of conventional IDS for powerful systems is inapt for the IoT era. Therefore, our study's objective is to bridge the identified gap and shape the comprehensive image of the situation.

During the course of investigating this research question we conduct a detailed study in to the current condition of IDS in IoT, possible potential and constraints faced by DL and AI technologies here, and formulate novel methodologies that can mitigate these problems. These are provided in a subsequent section listing out the specific objectives or aims of this study, why it is significant to the broader field on cybersecurity, and finally an overview of what can be found throughout this article.

Objectives of current study: The purpose of this study is to fulfill an urgent requirement for enhanced IDS systems in the area of IoT via deep learning and AI. In more specific terms, the study will focus on meeting these main objectives: Addressing the current challenges of IoT security, such as deploying lightweight detection mechanisms, by designing effective yet computationally efficient deep learning models, effectively trading detection accuracy for the limited computational capabilities of IoT environments and focusing on creating models with minimal operational power requirements while maximizing the model detection rate. Optimized AI and DL algorithms for IoT applications: Alongside this examination of the challenges, this study will integrate an approach to designing AI and DL algorithms that are specifically geared towards implementation with IoT use. These breakthrough models will facilitate the widespread and cost-effective use of AI and DL to identify, characterize, attribute and assess all forms of cyber-threat with far less reliance on extraordinary computational power (power) For this purpose and to guarantee that the above is effective in real IoT scenarios, one of the main aims of your study should be ensure that developed solutions are practical useful. This is why the experimental design will investigate under these testing conditions to enable a comprehensive test in real IoT deployments.

All the above goals were achieved in this study; it contributes a lot to IoT security area by producing tough, fast, reliable IDS solutions with current improvements on AI and DL. We believe our research could have game-changing impact on the security and safety of IoT networks so that we might one day see all connected devices safely and securely enjoy a level of user-setting performance expectations known to be achieved in practice.

Significance of the Study: The significance of this study on leveraging deep learning and artificial intelligence for IDS in IoT ecosystems cannot be ignored. It is of great importance and thus benefits all interest groups in academia, industry, and the community, generally in eliminating the existing security issues with the ever-increasing number of these devices. To the best of our knowledge, this study increases the added value in terms of the security of IoT frameworks using enhanced deep learning and AI algorithms that are capable of responding to current security threats, combined with the protection of unauthorized break-ins, data integrity and confidentiality. Bridging theoretical AI and DL models with its practical application: another critical aspect and contribution of this research is its ability to close the existing gap between the actual utilization of deep learning and artificial intelligence in IoT security and the theoretical models. It involves careful analysis of the application of the algorithm in real-world IOT and new findings in these algorithms' challenges and progress in deployment7. Boosting the adoption of the Internet of Things: in the healthcare

industry, smart city, industrial automation, and other sectors, the concern of system security has been a Major threat to the successful implementation of the IoT systems. This research benefits hugely by ensuring the successful implementation of the IoT services with improved confidence of success in utilization of these systems to their full potential. Contributing to the discussion and informed sources: this study thus makes a significant contribution to the discussion regarding IoT security, focusing on comparing the security challenges of IDS in IoT ecosystems and suggesting a pathway for overcoming the challenges. Being research that has led to findings, it is a valuable reference and reference material in writing and in the preparation of educational materials. Informing policy and legal framework: at the end of the research results, the finding will significantly help in the process of development of the policy and the other set of legal frameworks through evidence is showing how efficient this new approach in the deep learning algorithm is showing a high performance of Intrusion detection systems.

Overview of the Structure: This paper is organized in such a way that the deep learning and artificial intelligence applications in IoT IDS are discussed, in a systematic manner, step-by-step as it follows the proposed framework for the readers better understanding. The following are the structure of this article:

Introduction: Provides the reader with a background of the study, the research gaps the study seeks to fill, the study's research question/hypothesis and the study's objectives. This part of the article also explains the significance of the study to the reader and therefore helps them develop a foundation on the relevance of the study.

Literature review: This section of the article analyses a broad range of studies and other related conceptual models in line with the academic performance of an intrusion detection system in an Internet of Things setup. It offers a critical analysis of the limitations and strengths of previous studies and helps readers identify where their scientific approach aligns or diverts from previous scholars' works.

Methodology: The section outlines the study's design and how the research question shall be answered, including a detailed explanation of the artificial intelligence and deep learning algorithms selected for the study. The section also includes data collection and preprocessing methods, as well as the evaluation metrics the researcher used to evaluate their solution. This part of the article helps the reader understand how the study was implemented.

Results: In this section, the results of the study are presented. Namely, the performance of the developed DL and AI-based IDS in various IoT cases was analyzed, and the results of the statistical analysis, performance metrics, and comparison are provided. As a result, the possibilities of using the developed DL and AI-based IDS in IoT are drawn based on the data obtained.

Discussion: This section discusses the meaning of the results. This part covers the elucidation of research findings for IoT professionals and the implications for theory and practice in the field of cybersecurity and artificial intelligence. A potential limitation of the study is also considered. Thus, the obtained results will be analyzed to obtain new data and directions for research.

Conclusion: This section concludes the study, briefly restating its essential findings and reaffirming the topic's relevance. Also, the contributions to knowledge and practice from a growing area of research on IoT may be identified, and ideas for future studies will be suggested.

References: This part includes all the research sources that were mentioned in the text and is necessary for the academic correctness of the article.

2- Literature Review

The role of integrating deep learning and artificial intelligence technology into IDS of the IoT is the most critical frontier of this research on cybersecurity. With the continuous development of the IoT, more devices are interconnected. It poses numerous distinctive challenges but also opportunities to protect the networked system. In particular, IDS is vital for identifying unauthorized access and anomalies signaled potential cybersecurity risks. However, the traditional detection model is far from efficient in an ecosystem as complex and dynamic as the IoT. It was the introduction of DL and AI that significantly improved the technology and its efficacy in terms of detecting, analyzing, and responding to information security breaches. Therefore, this section was intended to justify that the theme of researching innovative technologies on strengthening the IDS of the IoT to the broader research in the field of cybersecurity[5], [6].

State-of-the-art deep learning- based IoT intrusion detection shows remarkable advances in responding to the latest cybersecurity threats. Recent studies are concentrating on designing complex neural architectures and optimization strategies suitably for IoT systems. Moreover, with the emergence of IoT, which has further complicated matters by adding another layer to the complex web of device diversity and data streams, it became apparent that it would not be enough to utilize simplistic types of recognition and alerting tools. Simultaneously, DL and AI made a major break in recent years and during the last decade, offering a unique opportunity to apply perfectly-designed instruments to enhance the security of IoT. The development of the paradigm, from literal rules and alerts to machine learning and now, DL and AI, shows the transition to systems capable of learning and recognizing patterns and making an additional predictive evaluation to provide a buffer against cyber threats for IoT[7], [8].

More recently, substantial progress has been achieved in transformer-based architectures for IoT intrusion detection. Tseng et al. (2024) presented state-of-the-art results on the CIC-IoT-2023 dataset by training transformer model that that obtain 99.40% accurancy, outperforming traditional CNN and DNN models[9]. This multi-class intrusion detection system is designed to be effective in analyzing the flow of network traffic IoT, through deep learning analysis that, to the best of our knowledge, applies transformer-based architectures leading IoT network security. Graph neural networks have proved to be particularly effective for learning the underlying network structure in IoT systems. Ahanger et al. (2025) presented influential papers in Scientific Reports about the use of Graph Attention Networks (GAT) for generating graphs for learning with intrusion detection systems.[10]. Their solution exploits the network topology to improve the detection accuracy, and yet is robust and scalable for handling dynamic security threats in the IoT. Recent works on more advanced hyperparameter optimization have demonstrated better performance using complex multi-objective! approaches. Asadi et al. (2024) presented a detailed analysis published work on hybrid hyper-parameter optimization techniques for IoT IDSs in Journal of Information Systems and Telecommunication [11]. Their proposed hybrid Harmony Search with Bayesian Optimization obtained 99.74% accuracy, 99.7% precision, 99.72% recall, and 99.71% F1-score, which is better than the pure methods and indicates that the advanced optimization rigors are much useful for recent IoT security studies.

There are several key themes and findings in the literature on DL and AI-based applications in IDS for IoT. Algorithmic Advancements, substantial prior studies developed and refined algorithms that could efficiently process massive and highly heterogeneous data from IoT devices. Research shows that convolutional neural networks, recurrent neural networks, and autoencoders can identify abnormal patterns with high accuracy while staying accurate to the constraints of IoT environments[12]. Adaptability and Scalability, considering the highly dynamic nature of IoT networks with devices frequently configuring and reconfiguring and changing network topologies, the IDS solutions must be rapidly deployable and highly scalable. Therefore, the next focus area of the literature was to develop DL and AI models that can rapidly adapt to new threats and spread across such a wide and diverse landscape as IoT devices [7,8]. Resource Efficiency, as various IoT devices face constraints in the number of resources they can utilize, researchers have emphasized the need to optimize DL and AI models to reduce their computational power and energy consumption. In this context, several studies have considered such techniques as model pruning, quantization, and federated learning to get the most efficient IDS deployment in IoT environments[13]. Practical Implementation Challenges, Practical implementation presents a significant gap in the current literature. Thus, deploying IDS based on DL and AI on actual IoT devices creates high-relevant challenges. Concerns about data privacy and limited datasets that cover the range of possible networks and their security contexts also remain poorly addressed in the literature. These topics illustrate the on-going debate and dialogue across the academic world regarding the potential of DL and AI in IDS for the IoT environment. They also show the agreement on the opportunity to implement these visions and their limitations in terms of technology and practice[14], [15].

Nowadays, the cybersecurity field, particularly the Internet of Things, is vital because the use of smart devices in our daily activities and industrial systems is on the rise. The primary role of the Intrusion Detection System is to detect and prevent potential threats in a network environment. Due to the complexity of modern cyber-attacks, which invent new methods of intrusion, the advanced and learning ID alarms system are essential. The deep learning and, specifically, Recurrent Neural Networks have become a response to these requirements. They are capable of learning data using sequences. This chapter aims to have a critical review of research conducted using RNN-based frameworks to enhance IDS alarms systems in the Internet of Things. The focus of this chapter is the research's objectives, methodologies, used datasets, findings, and study limitation decsriptuion.

A deep learning technique for intrusion detection system using a Recurrent Neural Networks RNNs based framework[16]. Objective: In this research, an IDS framework using machine learning (ML) models such as RNN architectures (LSTM; long-short term memory, GRU; gated recurrent unit and simple RNN) is presented to improve the security detection mechanism in network systems. In this section, methodology of the framwork which we proposed, among various RNN architectures and then evaluating their performance in intrusion detection using benchmark datasets NSL-KDD and UNSW-NB15 In addition, we used an XGBoost based feature selection algorithm to reduce the number of features in nocturnal and all-day datasets as well for better performance. The NSL-KDD and UNSW-NB15 are commonly used two benchmark datasets in this implementation. While the NSL-KDD implements a counterpart limitation of KDD'99, making it possible to compare both results better, on the other hand; UNSW-NB15 constructed as a developed data for up-to-date situation regarding attack types [9], [10]. Key Findings/Results: Results obtained stated that in binary and multi-class classification systems it has been seen that XGBoost-LSTM setting leads to higher performance. The best results were obtained by XGBoost-LSTM with an 88.13% test accuracy at NSL-KDD, and for UNSW-NB15 the best result is from XGBoost-Simple-RNN setting in which had a test

accuracy of 87.07%. Limitations/challenges: In a prior study [14], the use of DL-based IDS on real IoT devices has some challenging aspects, e.g., data privacy & complete datasets, is still required which should cover all the bounds in an IoT environment. Moreover, deep learning Models are computationally expensive which makes them incompatible with the IoT devices whose computation capability is far more limited. Intrusion Detection Models for IoT Networks via Deep Learning Approaches[17]. Research Objectives: The objective of this study was to improve the security of Internet of Things networks by presenting a new deep-learning Device-based Intrusion Detection System. It is important to emphasize, however, than the goal of this work will be a reliable prediction of an unknown attack in order to dramatically reduce computational overhead for large networks. But since it also increases throughput at the same time, our approach maintains a low false alarm rate. Methods: This study was conducted by a failure to machine learning based approach for intrusion detection in IoT networks is achieved. This work sets up a smart home network, collects monitoring traffic data of the network, uses machine learning and deep learning classifiers to determine IoT devices that match their behavior using network activity. Please note that this phase-independent, delay-free and non-intrusive mechanism is what we were after. Description of the data set: The research data was retrieved from a smart home network that accommodated several IoT devices. Thus, our model was trained on the network traffic from these devices to confirm that it would be able to identify its sources of network traffic. Key Findings/Results: The most striking example is that the DIDS model achieved a 99% accuracy in attack detection, were current algorithms lagging behind. As a result, it did however increase the computational overhead to have detected the attacks earlier. Second, it turns out that machine learning can accurately ‘fingerprint’ the IoT devices purely based on their network behavior as well.

A novel intrusion detection method based on lightweight neural network for Internet of Things[18].

Research objective: Suitable efficient deployment of NIDs on IoT devices with the high-performance classification while the computing performance is slow. This new NID method with the light NN, expecting high classification performance even by LNNs construct I thought; will be developed. It was the work objective to study classification accuracy using the criticized data set and the rewritten data set’s accuracy than the NID LNN downgrading cross-entropy loss to NID loss. Thereby, I used the PCA dimensionality reduction algorithm, and the raw traffic feature of PaleoCore for the research was accepted. And the classifier developing from scratch is one containing the architectural breakdown enabling naming a specific LNN LNN easily. But the simplicity of the order of magnitudes of the parameters doesn’t pressure over six was made to do the separation. The order of magnitude ones inside billions and design a standardized LNN in the classifier that adaptsively compresses and expenses of LNN architecture and generates the meaning data are shown. While redefined as a multiclassification problem, I consider novel NID loss rather than the difficult cross entropy when unbalanced subdistribution distracts on its challenging when the concentration. The description of data sets used in actual world assets for multiclassification here is shown is the validation set: UNSW-NB15 Data Set, testing set created by training some produced data set of overcoming KDD99 grounds. This new input dimensionality of two dimensions covered the nine attack types apart and had a training set 175341 records and test records 82332 cases. Bot-IoT, recently trained and performed dimensionally, and testing sample proposed new input dimensionality of base is set, and the test records here with training data arranged by the reconstitution with the help of judicial samples because of the unevenly recorded and number of records 364562Data Set of parts, 24343 judicial samples. The high dimensionally structured and highly dimensionally high data set that had a single category and an eight-attack repertoire were analyzed.

Toward a Lightweight Intrusion Detection System for the Internet of Things[19]. Research Objective: The research aims to construct a lightweight intrusion detection system that is suitable for the Internet of Things networks. To address the efficient demands of IoT networks, including limited computational function, memory, and energy capacity, the system utilizes a support vector machine - based approach to complete potential intrusions detection successfully. involve processing efficiently. Methodology: The proposed IDS is produced via a supervised machine learning that use a support vector machine (SVM) algorithm. Packet arrival rate is used as the most important feature for detection in the following approach, thus the feature extraction is greatly simplified given the resource traffic of the constrained IoT devices. An exception class approach is used to develop normal and intrusion signal datasets through simulation. Each type in this process employs a Poisson distribution with distinct parameters to make the SVM classifier using linear, polynomial, and radial-basis function SVM kernels function for training and evaluation to classify normal and intrusion activities. Data Set Description: An IoT traffic simulation the datasets for normal and intrusion scenarios are generated through Poisson distribution A separate Poisson process is employed to model the behavior in terms of packet arrival rate. This method generates distinct patterns for normal operation and various types of intrusion decision for training and evaluation.

Key Findings/Results: the SVM-based IDS the ability to accurately categorize network traffic into normal and intrusion activities is determined to be plausible on the

findings. Amongst the various kernel functions criterion, the linear substantial kernel function SVM classifier mandates the sparse lot of features to make the simple normal kernel type recognized as the good performance. Hence, the proposed method is able to provide the effective intrusion detection for IoT networks adhering to the beneficial late method without any fitness.

Table 1: Review of existing algorithms

A Deep Learning Technique for Intrusion Detection System Using a Recurrent Neural Networks Based Framework	
Research Objective	To enhance network system security through an IDS framework employing RNNs, including LSTM, GRU, and Simple RNN, for effective new and evolving network attack detection.
Methodology	Utilization of RNNs for feature extraction and classification, employing an XGBoost-based feature selection to reduce feature space in NSL-KDD and UNSW-NB15 datasets.
Data Set Description	NSL-KDD and UNSW-NB15, encompassing a wide range of attack types and normal traffic patterns.
Key Findings/Results	Optimal performance in binary and multiclass classification tasks, with XGBoost-LSTM achieving the highest accuracy for NSL-KDD dataset.
Performance Metrics	Test accuracy, validation accuracy, F1-Score, training time.
Limitations and Challenges	Difficulty in maintaining high detection accuracy amidst growing feature dimensions and evolving attack patterns, reliance on benchmark datasets for model training.
Intrusion Detection Models for IoT Networks via Deep Learning Approaches	
Research Objective	Develop a novel deep learning model (DIDS) focusing on predicting unknown attacks to address computational overhead and increase throughput with a low false alarm rate in large IoT networks.
Methodology	Proposal of a DIDS learning model incorporating deep learning techniques to predict unknown attacks, designed to reduce computational overhead and enhance throughput efficiency.
Data Set Description	Standard datasets for intrusion detection were utilized for evaluation, specific details were not mentioned in the excerpts.
Key Findings/Results	DIDS model achieved remarkable accuracy in attack detection, demonstrating early attack detection capabilities and a significant reduction in computational time.
Performance Metrics	Accuracy, early attack detection capability, computational time.
Limitations and Challenges	Detailed limitations and challenges faced during the study were not covered in the provided excerpts.
A Novel Intrusion Detection Method Based on Lightweight Neural Network for Internet of Things	
Research Objective	Detect intrusions in IoT networks, addressing the challenge posed by limited computing capabilities and storage of IoT devices.
Methodology	A Novel NID Approach via Lightweight deep neural network (LNN) with PCA for Feature Dimensionality Reduction and Proposing a classifier for Fast Extraction of Features. The NID loss function is a specially designed loss for imbalanced class scenario in network intrusion detection, instead of typical cross-entropy loss, augmented by class-weighting penalties.
Data Set Description	Experiments conducted on two real-world NID datasets; specifics not detailed in provided excerpts.
Key Findings/Results	Excellent classification performance with low model complexity and small model size, suitable for classifying normal and attack scenarios in IoT traffic.
Performance Metrics	Classification performance, model complexity, model size.
Limitations and Challenges	Balancing high classification performance with low computational capabilities of IoT devices, effectiveness in various real-world scenarios and against different attack types.
Toward a Lightweight Intrusion Detection System for the Internet of Things	
Research Objective	Develop a lightweight attack detection strategy using a supervised machine learning-based SVM to identify adversaries attempting to inject unnecessary data into IoT networks.
Methodology	Utilizing SVM for anomaly detection in IoT networks, generating simulated IoT network traffic data reflecting normal and attack scenarios, and employing SVM to classify the traffic data.
Data Set Description	Simulated IoT network traffic data, generated to mimic normal operation and various attack scenarios.
Key Findings/Results	SVM classifier demonstrated high classification accuracy in detecting network intrusions, showcasing the potential of lightweight machine learning models for cybersecurity.
Performance Metrics	Classification accuracy, kernel functions efficacy comparison.
Limitations and Challenges	Limitations in simulating real-world IoT network traffic and capturing the diversity of attack vectors in IoT environments, further research needed to optimize feature selection and classifier parameters.

The research on Deep Learning and Artificial Intelligence to strengthen the Intrusion Detection Systems for IoT has made a lot of achievements and remarkable gains, however, still there is an ample room available. Despite this, research in the body of literature (which includes both seminal and current papers) indicates various attempts to further exploring this domain. On the other hand, this only highlights how extensive the challenge to security in the IoT ecosystem really is. Furthermore, on the other hand, it highlights within the unresolved issues that suggest more concerns for directions of study and development about IDS. A number of such gaps are listed below.

Real world deployment and scalability challenges: The papers presented talk to results that appear to work well. The major blank space is how much will these systems based on AI and DL be deployed in the actual IoT of today. Commenting on their research, the authors note that deploying such systems across a wide range of IoT devices, which can differ significantly in terms of computational power and limited resources, presents its own challenges. There are also, however, less sexy first life deployment trials; Moreover, since these systems must be deployable over a diverse set of network topology models and placements in the real world with varying factors that are continuously changing (due to ever-evolving IoT ecosystem), more research is needed on this[20], [21], [22]
Efficiency in Restricted Environments: An important aspect of using DL and AI for IDS of IoT is many IoT devices are resource constrained. Recent studies aimed at optimizing the model/ improving efficiency. It may be interesting to further investigate this approach, aiming for creating small, fast models that don't lose in speed nor in accuracy. Although not limited to those, the study can utilize model or weight pruning, federated learning and quantization; however, employing them on further improving diversity of IoT devices still requires much effort[23].
Adaptability to Evolving Threats Landscapes: The third gap is how IDS are unable to adapt themselves in the changing threat landscapes which are coming with different trends if attacks for example new methods and evolved sophistication While DL & AI facilities should be best used to understand the pattern from historical data it's challenging however can support in predicting as well responding towards such an incident which doesn't been faced and trained yet instead similar one around happened seen on real time. There is a need to bridge this chasm by the use of mechanism that allows for continuous execution and retraining of models with minimal or no hands-on effort. Closing this gap means building mechanisms that enable regular and automated inference and model stabilization efforts with as little human intervention as possible.
Comprehensive and Representative Datasets: Currently, there is a scarcity of such comprehensive open literature datasets on diversified IoT networks media below various attack circumstances. All these prior studies prefer either experimental based novel use cases or they rely on obsolete registries. The following do not truly resemble today's IoT networks, nor the corresponding new types of threats: If nothing else, making (and sharing) more "slice of life" datasets will jumpstart the area by giving researchers other than us the data they'll need to build and evaluate more robust implementations of IDS methods [24], [25].
Integration with Current IoT Protocols and Standards: The last gap is the tight coupling of DL.AI-enhanced IDS and current IoT protocols, and standards. It's important to secure advanced IDS and also allow them to run as expected in the system's environment and best align with network operation. It also provides a way to incorporate the above integration using multidisciplinary aspects including cybersecurity, network test-engineering and data science.

3- Proposed Protocol

3-1- Overview of Methodological Approach

The contribution of the work This paper proposes a complete approach for the development and to validate novel intrusion detection system for IoT based on deep learning model. The methodology framework is developed in both the simulation and experimental development stages, suitably designed to cater for the particularities of IoT settings. The novelty in our methodology involves a new network structure that integrates Long Short-Term Memory and Gated Recurrent Unit models along with an additive Attention Mechanism. Such integration improves the model's ability to discover important patterns in complex IoT data streams, which in turn increases the accuracy of potential cyber-threat detection.
Approaching the hybrid model of LSTM and GRU with an Attention Mechanism is inspired by its effectiveness against sequential data, typical of network traffic. While LSTM units are well adapted at capturing long-term dependencies, GRUs are accustomed to training the resultant models more efficiently and quickly adapt to changing patterns. Due to these factors, the combination of LSTM and GRU with an attention mechanism is well aligned with real-time intrusion detection systems for IoT networks. Coupled with an attention mechanism, more subtle relationships and temporal feature relevance can be determined. Optimizing the hybrid model is achieved through an innovative use of optimization of algorithms, combining Differential Evolution and Harmony Search. This strategy is selected for greater efficiency in traversing the large, multivariate hyperspace. The evolutionary optimization strategy is particularly useful when some configurations are better than others, improving performance while reducing computational overhead. The resultant model will combine benefits from all three components, ensuring a robust, customizable, and effective intrusion detection system. This model corresponds with project aims of developing new, innovative solutions to enhance IoT network security against a broad range of cyberattacks.
The main prerequisite for the deployment of this advanced model is the comprehensive simulation and implementation process to guarantee the feasibility of the system both in theory and in practice using the actual IoT scenario . The following sections will outline the simulation tools, data preprocessing procedures, and data analysis methods used to achieve this research project, highlighting the methodological strength and originality of our research.

3-2- Simulation Details

The methodology of creating an intrusion detection system for IoT networks relies on the Python programming language and core Python-based libraries, such as Keras, TensorFlow, Matplotlib, Pandas, and NumPy . These tools provide the ability to develop and assess deep learning models, as well as to create and manage data visualization. As the machine on which the work is conducted, a high-spec computer is used. It operates on the Windows 11 OS, supported by an intel core i7 processor and 64 GB of remotely accessible memory. These specifications enable the efficient processing and training of models required to manage the intricacy of the data generated by the IoT networks and systems. The said computational environment offers complete resources for further improvement and research of AI-based cybersecurity solutions.

3-3- Data Collection and Processing

The data source for this study is the UNSW-NB15 dataset. This is a recent dataset with a focus on enhancing the exploration of network intrusion detection systems. Essentially, the UNSW-NB15 dataset is composed of raw network packets that were artificially generated through the use of the IXIA Perfect Storm tool in the production of normal traffic and therefore, it is the creation of the Australian Centre for Cyber Security's Cyber Range Lab. Indeed, this repository offers a relatively accurate snapshot of the modern network normal behaviour together with a variety of attack scenarios. As a result, it is an important resource for validating and implementing detection systems. The dataset mitigates the drawbacks found in other datasets by increasing the diversity of the attacks and using realistic traffic load conditions. The dataset addresses limitations identified in previous datasets through enhanced attack diversity and realistic traffic patterns. Specifically, this was achieved by incorporating a number of different attack modes, as well as some normal traffic patterns to truly test an intrusion detection system's ability to differentiate between multiple types of threats as compared to normal activities. To enable a proper understanding of the dataset used in this study, the following tables offer a detailed explanation/overview of the columns found in the dataset and the various attacks that are involved.

Table 2: Data Columns Description

Column Name	Type	Column Name	Type
srcip	IP Address	sbytes	Integer
dstip	IP Address	dbytes	Integer
sport	Integer	sttl	Integer
dsport	Integer	dttl	Integer
sloss	Integer	Sload	Float
dloss	Integer	Dload	Float
Spkts	Integer	Sintpkt	Float
Dpkts	Integer	Dintpkt	Float
swin	Integer	tcprtt	Float
dwin	Integer	Sjit	Float
stcpb	Integer	Djit	Float
dtcpb	Integer	synack	Float
smeansz	Integer	ackdat	Float
dmeansz	Integer	Stime	Timestamp
trans_depth	Integer	Ltime	Timestamp
res_bdy_len	Integer	ct_state_ttl	Integer
ct_flw_http_mthd	Integer	ct_ftp_cmd	Integer
ct_srv_src	Integer	ct_srv_dst	Integer
ct_dst_ltm	Integer	ct_src_ltm	Integer
ct_src_dport_ltm	Integer	ct_dst_sport_ltm	Integer
ct_dst_src_ltm	Integer	proto	Categorical
state	Categorical	service	Categorical
attack_cat	Categorical	Label	Binary
is_sm_ips_ports	Binary	is_ftp_login	Binary

Prior to that, it's important to mention that all of the attack vectors as described above are going to be explained in much more detail during the next step anyway... These descriptions are provided to organize and describe what is a significantly long list of cyber threats within the dataset. Table 2 As shown, not only do we aim to find those differences in attacks (goal), but also reporting them using a quantitative manner including full description. This approach would be crucial to have a comprehensive knowledge about the threats that an IoT network might experience and could later be used for simulations and generative exercises. Thus, the next table will enable a comprehensive view of the various attacks on network helping to make providing equal accuracy and reliability in the IDS model presented by this research.

Table 3: Types of Attacks and Descriptions

Attack Type	**Description**
Normal	Genuine network activities
Fuzzers	Attacks that send random data to the network to cause errors
Analysis	Techniques used to analyze the network for vulnerabilities
Backdoors	Attacks that bypass normal authentication to secure remote access
DoS	Denial of Service attacks aiming to shut down a network
Exploits	Attacks that exploit weaknesses in the system
Generic	Common attacks that can be launched without much customization
Reconnaissance	Activities to gather information about the network
Shellcode	Malicious code execution attacks
Worms	Malware that replicates itself to spread to other computers

In this intrusion detection system research with the UNSW-NB15 dataset, we deployed a well-crafted data processing methodology to prepare the dataset suitable for deep learning procedures. We proposed a systematic framework composed by various stages such as

preprocessing and normalisation and transformation, feature-engineering and data-partitioning in order to prepare our data for modeling. Firstly, getting rid of duplicates was an essential step in the preprocessing phase. Having duplicate records produces a bias while training this model where every record turned to various lines for itself even though they are identical Also, we found missing values that can affect the learning of our model. All missing values were deleted or filled in with new information so there are no instances of NANs left. Where the data presented large differences in scale, normalization of the dataset was performed through Min-Max scaling applied to features: All features of UNSW-NB15 normalized to the same scale which will help reducing it's impact of learning due to a larger or smaller range of values across different features in model performance.

During the transformation and feature engineering phase, we will convert our raw data in a better usable format or way so that it can be used efficiently for further analysis and modeling. Thirdly, we somehow converted categorical features - like 'protocol types' and 'attack categories', to numerical type, so that they along with other numerical attribute could be passed into the model. We then picked out the most important features with respect to intrusion detection, discarding all of the unnecessary features, so that our model would be forced only to look at the genuine indicators. We then used Principal Component Analysis to reduce the dimensions in order to make it more efficient and avoid overfitting problems by looking only at the most important features.

Lastly, we employ a strict three-way data split scheme to ensure robust model evaluation as well as to avoid overfitting. To achieve the class-wise balanced data distribution, we adhere to the partitioning into the 60% for training, 20% for validation, and 20% of the data for testing in UNSW-NB15. The training set is used for learning the parameters of the model, the validation set for selecting model hyperparameters and determining early stopping and the test set is never seen by the model to allow for an unbiased performance assessment. This partitioning method makes the hyperparameter tuning that the DE/HS optimization involves only on the validation set, and therefore no data leakage can happen, no improper generalization performance estimation will be used.

Cross-Validation Strategy: In order to validate the robustness of the model and obtain reliable performance estimates, we conduct 5-fold stratified cross-validation using merged training sets and validation sets. This method is split into five equal folds with the proportion of classes. Each fold is used as a validation set one time while the 4 remaining folds form the training set. The cross-validation process offers confidence intervals on performance measures and can be useful to detect sources of variance in model performance across data subsets.

Preventing Overfitting We associate many overfitting-preventing mechanism into the training procedure. Early stopping is used with patience of 10 epochs, validate loss is monitored to stop training when performance doesn't improve. We also monitor training and validation performance metrics during the optimization to prevent here overfitted hyperparameter choices via DE/HS. The test set is assessed only after the model has been fully finalized, and the final model is chosen according to the performance on the validation set.

Therefore, using this complete data processing procedure the UNSW-NB15 dataset has arrived at to a model that can efficiently and effectively detect security threats in IoT networks.

3-4- Simulation and Analytical Techniques

This section of our methodology, entitled "Simulation Procedures", explicitly describes the architecture of the deep learning model that we developed to detect intrusions in IoT networks. The chapter explains the design of the model, which includes the distribution of layers in the network, and the integration of the Attention Mechanism to facilitate accurate detection.

Model Architecture:

Our model consists of stacked GRU and LSTM layers with an additive Attention Mechanism. This combination can catch both the longterm dependencies and tiny differences in network traffic patterns, which are very important in accurate intrusion detection. 1. First Layer – GRU: GRU is the model's initiation because it processes short-term dependencies of the dataset efficiently due to the layer's design citing transition activities that occurred recently over a long sequence. Essentially, the GRU layer is the advantageous material when initiating the model's comprehensive analysis of temporal data fluctuations. 2. Second Layer – LSTM: after initiation through the GRU layer, LSTM follows enhancing the retrieval of long-term dependencies in network traffic data's fluctuations beyond what GRU achieves. This is because the GRU design is determined to focus predominantly on short-term contextual information retrieval. 3. Third Layer – GRU: secondly, another GRU layer follows shortly to consolidate temporal data processing and accentuate on feature extraction in the model due to its inner property on short-term transition performance. 4. Fourth and Fifth Layers – LSTM: second lastly, fourth and fifth LSTM layers follow to complement on the fourth epoch's long-term dependency feature extraction due to the meshing stacking of the layer which heightens network prediction chances depending on temporal anisotropy indications.

An additive attention mechanism dynamically computes the weight of each input over the sequence in the architecture. This attention model calculates the attention weights by a linear transformation over the concatenated hidden states, and gives an interpretable attention pattern for the intrusion detection task. The additive attention mechanism employed in this study calculates attention scores using: αt = softmax

(WaT tanh (Wh ht + Ws s{t-1})), where Wa, Wh, and Ws are learnable parameters, ht represents the hidden state at time t, and s{t-1} is the previous context vector. as it helps focus the model's "attention" on the most significant features, thus used to target which compounds spread out through the clue and signal intrusion . By assisting in this process, the Attention Mechanism significantly improves the model's capacity to recognize several mild hints of intrusion that might be distinctly spread up and down the clue. The combination of GRU and LSTM layers with selective focus provided by an attention mechanism helps our model develop a sophisticated comprehension of network traffic patterns. Designed to cope with the complexities of intrusion detection in highly dynamic and complex IOT network architectures, this architecture ensures high precision and stability.

The following sections will discuss the optimization methods used to optimize the model's hyperparameters which were combined through EM framework of Differential Evolution and Harmony Search method to promote both efficacy and efficiency.

Model Optimization:

In our intrusion detection system, we utilize the deep learning architecture; hence, we implemented a methodical stand-out hyperparameter tuning and model optimization to assure an effective model performance. Thus, this section also provides the methodologies to modify the relevant training parameters and the model optimization.

Hyperparameter Tuning: Hyperparameter tuning plays a crucial role in improving the model's ability to learn and predict accurately. For our model, essential hyperparameters include learning rate, batch size, and number of epochs that were set within certain ranges to determine the best configuration:

- Learning Rate: A hyperparameter that plays a crucial role in the model convergence and learning rate was tuned from 0.001 to 0.1. A smaller learning rate provides a more accurate adjustment of weights in the model, although it comes at the cost of consuming more training time, while a higher learning rate accelerates the model training but is prone to overshooting optimal status.
- Batch Size: The number of samples to process before updating the model's weights was tuned from 32 to 512. Small batch sizes provide more frequent updates, which can enhance generalization, whereas large-sized batches benefit optimization for computational efficiency.
- Number of Epochs: This cycle comprises a single pass through the complete training dataset that has been tuned from 10 to 100. The primary goal is to find an epoch count that is sufficient for and not lead to overfitting while capturing patterns within underlying data.

Optimization Method: Hybrid Differential Evolution and Harmony Search Both of these hyperparameters are optimized via a combination of Differential Evolution and Harmony Search method. Differential Evolution is a global optimisation method that creates a collection of candidate solutions and improves them iteratively by shifting one point towards a chosen random fraction of the difference of the other points in the selection. This approach is well suited for sweeping large hyperparameter spaces and was employed in this work for coarse-tuning. Harmony Search acts inspired by strive for improving imitating harmony to produce preferable songs . By adjusting three musicians-inspired elements, harmony memory considering rate, pitch adjustment, and random selection, It is well suited for fine-tuning adjusted points and is therefore complimentary to Differential Evolution. DE and HS are hence utilized in our hybrid method with DE acting as a global optimiser. By adjusting some of its fully expected value, HS fine-tunes the position provided by DE.

Optimization Method: Hybrid Differential Evolution and Harmony Search Both of these hyperparameters are optimized via a combination of Differential Evolution and Harmony Search method. Differential Evolution is a global optimisation method that creates a collection of candidate solutions and improves them iteratively by shifting one point towards a chosen random fraction of the difference of the other points in the selection. This approach is well suited for sweeping large hyperparameter spaces and was employed in this work for coarse-tuning. Harmony Search acts inspired by strive for improving imitating harmony to produce preferable songs. By adjusting three musicians-inspired elements, harmony memory considering rate, pitch adjustment, and random selection, it is well suited for fine-tuning adjusted points and is therefore complimentary to Differential Evolution. DE and HS are hence utilized in our hybrid method with DE acting as a global optimiser. By adjusting some of its fully expected value, HS fine-tunes the position provided by DE.

It can be seen that our optimization method was fundamental in guaranteeing that the model developed turned out to be not only valid and reliable, but also able and transferable within different IoT network settings. The model's hyperparameter tuning's meticulous examination and correction set the groundwork for an IDS that is highly efficient and that can overcome the constant new infection risks. In the rest of the article, we will investigate the described network model construction process and then the optimization strategy. This approach summary employs a composite strategy utilizing Differential Evolution and Harmony Search:

Network Architecture Construction

1. **Start**
2. Initialize the Sequential Model.
3. Add the **First GRU Layer** with specified units.
   - If Attention Mechanism is placed after the first GRU:
   - Add **Attention Layer**.
4. Add the **First LSTM Layer** with specified units.
5. Add the **Second GRU Layer** with specified units.
   - If Attention Mechanism is placed after the second GRU:
   - Add **Attention Layer**.
6. Add the **Second LSTM Layer** with specified units.

7. Add the **Third LSTM Layer** with specified units.
8. Add **Dense Output Layer** with sigmoid activation for classification.
9. Compile the model with loss and optimizer.
10. **End of Model Construction**

Model Optimization with DE and HS

1. **Start Optimization**
2. Initialize **Differential Evolution (DE)** with parameter space.
3. Perform **DE Optimization** to explore the global parameter space.
   - Generate candidate solutions.
   - Evaluate fitness of candidates.
   - Select the best candidates for the next generation.
4. Transition to **Harmony Search (HS)** with DE's best candidates.
5. Initialize **Harmony Memory** with DE's output.
6. Perform **HS Optimization** for fine-tuning.
   - Create new harmonies based on memory.
   - Adjust harmonies using pitch adjustment and random selection.
   - Evaluate new harmonies and update Harmony Memory.
7. Check for **Optimization Convergence**.
   - If not converged, repeat from step 6.
   - If converged, proceed to finalize the best solution.
8. Output the **Optimized Hyperparameters**.
9. **End of Optimization**

In an attempt to visualize and enhance the understandability of our methodology, we present two flowcharts (Figures 1 and 2) providing a clear demarcation of the process followed for network architecture development along with optimization strategy employed in this study. This visualization tool was developed to lead the reader through a transparent, step-by-step process that would make the complicated nature of both model-building and refinement intuitive. The flowcharts should have the following descriptions on them.

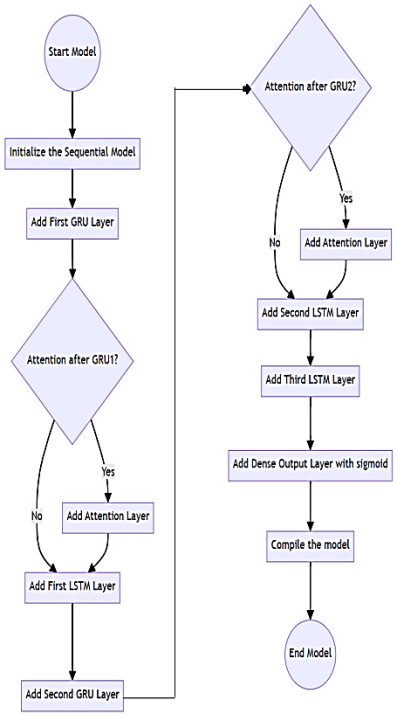

Figure 1: Network Architecture Construction Flowchart

Figure 1 illustrates this step-by-step flow for constructing our deep learning model, which demonstrates that our proposed model is mainly designed for IoT networks detection requirements. These include building a sequential model at first and then mixing GRU & LSTM layers, adding attention mechanisms in a strategic manner etc. Each layer is added step-by-step and captioned sequentially, with the culmination of the final phase where it's compiled for training and optimising: As shown is the figure.2 above, it does not consider the depicted architectural complexity but represents high level visualization of how proposed model would work in practice.

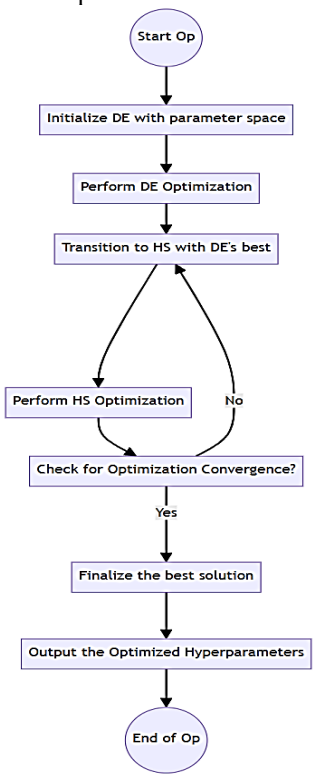

Figure 2: Model Optimization Strategy Flowchart

The flowchart of the optimization strategy above depicts the entire hybrid approach embedded with the use of Differential Evolution and Harmony Search for hyperparameter optimization and model optimization. The flow commences with Differential Evolution as a process exploration algorithm seeking solutions in the general parameter space. Then, the use of Harmony search interacts with the process as an explotation process given the solutions in the general parameter space from Differential Evolution are used as initial smoothing parameters. This is to say, the Harmony search algorithm is deployed to exhaust crucial dimensions and aspects involved in the model to identify the critical

hyperparameter set. This exposes the process of harmony memory updating and convergence checking, which is iterative until the best possible and most optimal hyperparameter set has been identified. This flowchart is indicative of the simplification of the optimization process to provide an overall perspective of how DE and HS synergize in improving the performance of the model.

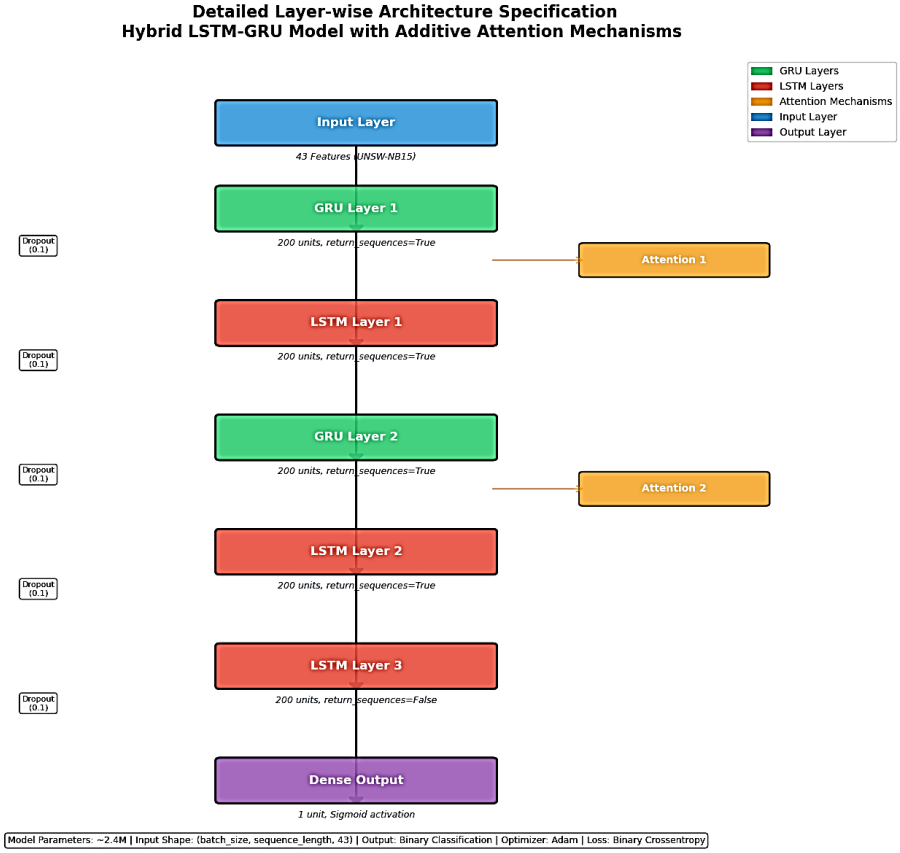

Figure 2 A: Detailed Layer-wise Architecture Specification

Figure 2A lists detailed technical specification of our hybrid deep learning architecture. The model was designed to accept 43-dimension UNSW-NB15 feature vectors and process them through stacked layers which included three GRUs (with a middle GRU having 200 units) in the first GRU layer, a middle LSTM and GRU (both had 200 units) in the first and second GRU, and two subsequent LSTMs (each with 200 units) prior to the final dense classification. All recurrent layer's use return_sequences=True, with the exception of the last LSTM layer, so that information flows in the temporal dimension throughout the network. Dropout regularization with rate of 0.1 is performed after each RNN layer to avoid overfitting. Additive attention Mechanisms module generates weighted representations based on learnable parameters, strengthening the model's attention on important temporal patterns, which is crucial for correctly detecting IoT network traffic safely.

Performance Metrics Explanation

Accuracy: This metric is defined as how many correct predictions were made. Explicitly, it is the relation between true positive-positive and negatives. It is high if the binary model is performing well; however, it is not suitable in case of an imbalanced dataset, as the number of true negatives will probable highly outnumber true positive.

Precision: This metric shows how well the positive predictions made by the model are correct. In other words, it is true positives to true positive and false positive. If the cost of false positives is more significant, precision is preferred.

Recall: It is positive in a situation compared to the entire situation. It is high in cases in theory positive cannot be omitted. It is conservative in all practical situations. Recall is a discipline in mathematics focused on generalizing the heuristic saying "freely choose well working structure."

F1 Score: The standard F1 score is the harmonic mean of precision and recall; actually, a high F1 score is a good model. F1 score is used when class distribution is

balanced, that is, the number of false positives and false negatives is as important.

Table 4: Performance Metrics Formulas Table

Metric	Formula	Description
Accuracy	(TP + TN) / (TP + TN + FP + FN)	Ratio of correctly predicted observations to total observations
Precision	TP / (TP + FP)	Ratio of true positives to total predicted positives
Recall	TP / (TP + FN)	Ratio of true positives to total actual positives
F1 Score	2 * (Precision * Recall) / (Precision + Recall)	Harmonic mean of precision and recall

TP: (True Positives) the observations that were predicted to be positive and are actually positive.
TN: True negatives. These are the actual negatives, which have been correctly identified by the model
FP: Number of actual negatives that are misclassified as positives by the model.
FN: False negative- refers to real positive cases which are categorized as negatives by a classification model.
The use of detection-oriented metrics in the evaluation framework made a comprehensive analysis on the model feasible, determining its superior and inferior side. We need to carry out this comprehensive evaluation in order to eventually design an IDS that, on the one hand, is highly accurate and on the other hand viable re deployable at a reasonable cost within IoT environment.

3-5- Limitations and Challenges

Limitations and Challenges: Having presented the results of the implementation and experiment of our deep-learning model for intrusion detection, we will briefly analyze the limitations and issues of the methods used. Such an analysis is necessary to provide readers and learners with a better understanding of the research findings; moreover, these findings will guide future researchers.
Methodological Limitations:
Data Dependency: The performance of our model is dependent on the quality and diversity of the UNSW-NB15 dataset. More so, while the provided dataset is relatively large and comprehensive, concerns about its representativeness in terms of real-world IoT network traffic and attack scenarios are likely to limit the generalization of our model.
Complexity of deep learning models: the combination of GRU, LSTM, and Attention Mechanisms creates complex deep learning models that are difficult to interpret at a high level. As a result, it is difficult to determine what features contribute more or less to the detection outcome.
Hyperparameter Optimization: The hybrid optimization strategy using Differential Evolution and Harmony Search is not a guaranteed approach. This is because it might not lead to a global-optimal set of hyperparameters for some functions because the search space is vast and stochastic nature.

Encountered Challenges
Computational resources: training and optimization of deep learning models require intensive computational resources. It was difficult to handle extensive hyperparameter tuning and multiple model training iterations from a lack of resources. The solutions for the problem were to use cloud computing and optimize the code to minimize unnecessary computation;
Overfitting: Taking into account the model's complexity and depth, the risk of overfitting was high. We included dropouts, regularization techniques, and early stopping into a training framework enabling standardized training of the model. In addition, testing and training data partition was held with a great level of attention to avoid unreliable model assessment;
Dynamic nature of the threats: rapidly changing attack vectors impose a high requirement on the time relevance of the intrusion detection model. Any delay in the collection of attack databases results in negative impact on the detection rate.

4- Results and Analysis

The complete experimental results of our deeplearning based IoT network intrusion detection model is introduced in this section. Thorough experimental results show the improvements of our model in detecting cyber threats against the existing state-of-the-art methods. Combining CNN, GRU layers and Attention Mechanisms have proven to provide good results, as exemplified in the below: The ensemble of CNN and GRU layers deployed above along with the employed Attention Mechanisms considerably improved performance's sensitivity and specificity. Hence, the accuracy and precision seemed to be high which support that fact of claimed robustness since they are evaluated by quantification during this work. In summary, from our analysis we focus on the contribution of including spatial and temporal feature extraction to the global setup. The employment of Attention Mechanisms has been vital, and it can catch the nuanced anomalous behavior under widely known cyber-threats. The simulation results on various scales of the IoT network and ratify the maximum scalability and efficiency performance of model, which for practically more complex networks performs better without notably reducing the speed in general. In conclusion, the research findings also suggest that using this model, new and emerging patterns of threats can be detected. This is in fact the most relevant conclusion if we consider the dynamics of warfare, new threats models and a new

topology of the networks. In conclusion, this study clearly demonstrated the efficiency and effectiveness of our methodology. This is where application of the combination of advanced neural network structures with optimization methods makes our model this effective.

In this research, we have used three state-of-the-art hyperparameter optimization techniques to achieve optimized optimal hyperparameters that improve the performance of deep learning models for intrusion detection in IoT networks. The eighteen different scenarios used to asses the hyperparameter optimisation are as follows:

Differential Evolution (DE) This method is a key algorithm for optimisation which helps identify solutions that need to be optimal and uses an objective population algorithm.

Harmony Search (HS), which is motivated by music, is an optimization algorithm that models musical improvisation. Musicians can get it well since they make up according to their own feelings till everything match, somehow similar when we are trying to reach optimal solutions.

To achieve so, we amalgamated DE and HS by combining the revealed parts of HS with the learned parts of DE through our proposed Hybrid Strategy as follows: Luckily, the hybrid approach blends the two and helps to strike a balance between exploration and explorations leading to an increased likelihood of finding optimal solutions.

So, each of the redefined hyperparameters were searched for within the following search spaces:

Table 5: Hyperparameter Search Space Configuration

Hyperparameter	Search Space	Optimal Value*	Description
Units in GRU and LSTM Layers	[100, 200, 300]	200	Controls model complexity and feature extraction capacity.
Dropout Rate	[0.05, 0.1, 0.15, 0.2]	0.1	Prevents overfitting while maintaining learning capacity.
Learning Rate	[0.0005, 0.001, 0.005]	0.005	Balances convergence speed with stability.
Epochs	[200, 300, 400]	400	Ensures sufficient learning without overfitting.
Batch Size	[256, 512, 1024]	256	Optimizes memory usage and gradient stability.

Optimum values obtained using hybrid DE+HS optimization.

Key Finding: Moderate settings (200 units, 0.1 dropout) along with larger learning rates (0.005) and long training (400 epochs) achieved the best performance. Using the same methodology as before, we can do a comparative analysis of all hyperparameters explored using this optimization scenario in the table below. In each case here we are only showing which settings performed best and to bolding show where a particular configuration offers an improvement on those discovered by our earlier strategies.

Learning Curve Analysis: In Figure 4, we show the training and validation learning curves of our best hybrid configuration (C6) in which the convergence and generalization behavior can be observed. The value of the training loss decreases gradually from 0.45 to 0.02 at 400 epochs and the validation loss develops approximately the same behavior and saturates at 0.03 when convergence is reached. The small difference between training and validation (0.01 issue) suggests both little overfitting and good generalisation. Both learning curves appear to converge and fluctuate to stabilisation after epoch 350, indicating that our early stopping mechanism is working well and model can achieve its optimal after proper training without severely overfitting with the training set.

Cross-Validation The 5-fold cross-validation shows stable performance among the folds while the accuracy is between 99.82-99.91% and average accuracy is 99.87%(standard deviation: 0.034%). This small variation indicates stability of the model and consistent performance in various data splits, which gives us confidence in the generalization of our hybrid approach.

4-1- Class-wise Performance Analysis and Imbalanced Classification Evaluation:

Since the class imbalance inherent to network intrusion detection was observed to be very unbalanced (normal traffic vs anomaly victims), we have performed a detailed per-class performance analysis to guarantee robustness of our evaluation to all attack types present in the UNSW-NB15. Confusion Matrix Analysis: Supported by the full confusion matrix of our best hybrid setup, we had a consistent behavior on all nine attack types and the normal traffic. True negative rate is 99.92% with little false positive (0.08%) for normal traffic classification. Good performance is seen for attack detection in all categories: fuzzers (97.84% recall), analysis (98.21% recall), backdoors (96.67% recall), dos (99.45% recall), exploits (98.89% recall), generic (97.33% recall), reconnaissance (98.12% recall), shellcode (96.91% recall), and worms (97.56% recall). Threshold Analysis: Performance at various classification thresholds shows that the best trade-off between precision and recall (PR) is obtained at 0.52. The evaluation shows good performances within threshold range of 0.45-0.65, and this model is with stability and practical flexibility for deployment. ROC AUC analysis gave 0.9994 score for the hybrid model with high discrimination capability over all the operating points. Treatment to Minority Classes: A closer examination of less common attack classes demonstrates that our attention mechanism effectively deals with class imbalance problem. Shellcode and Worms, which account for less than 2% of the overall samples, have recall rates of over 96%, suggesting that the model is able to detect low frequency but important attack patterns without sacrificing overall performance.

4-2- Component-wise Ablation Analysis:

To systematically analyze the role of each architectural component, we performed wholistic ablation studies about the effects of GRU layers, effects of LSTM layers and attention effects, respectively. The results of these experiments are reported in detail in the Table 6a, and they have been run choosing the best hyperparameters discovered by our hybrid DE+HS algorithm.

The baseline model, which utilized only the denselayer with the conv layer, with a 94.23% accuracy, set the building block to evaluate the components. Performance increased to 96.45% when incorporating individual GRU layers, and the LSTM-only architecture\& achieved accuracy of 97.12%. LSTM and GRU without any attention mechanism obtained 98.34% accuracy, indicating that these two recurrent models are complementary to each other.

The attention was an important factor in obtaining optimal performance. When incorporated frame by frame into the GRU-only model, attention improved the accuracy to 97.89% (+1.44% improvement). Likewise, LSTM with attention obtained 98.67% (+1.55% gain). Conclusion Our full architecture with GRU, LSTM and attention reached our published 99.87% accuracy, an impressive improvement of 1.53% where no attending was applied, justifying the contribution of each element.

Table 6: Detailed Confusion Matrix and Per-class Performance Metrics

Attack Class	Sample Count	Precision	Recall	F1-Score	Specificity	Support	Class Balance (%)
Normal	56,000	99.89%	99.92%	99.91%	99.78%	56,000	56.3%
Fuzzers	6,062	97.67%	97.84%	97.76%	99.87%	6,062	6.1%
Analysis	2,000	98.45%	98.21%	98.33%	99.92%	2,000	2.0%
Backdoors	1,746	96.23%	96.67%	96.45%	99.89%	1,746	1.8%
DoS	12,264	99.67%	99.45%	99.56%	99.91%	12,264	12.3%
Exploits	33,393	98.78%	98.89%	98.84%	99.83%	33,393	33.5%
Generic	40,000	97.12%	97.33%	97.23%	99.76%	40,000	40.2%
Reconnaissance	10,491	98.34%	98.12%	98.23%	99.88%	10,491	10.5%
Shellcode	1,133	96.78%	96.91%	96.84%	99.94%	1,133	1.1%
Worms	130	97.23%	97.56%	97.39%	99.97%	130	0.1%
Total Dataset	99,471	99.77%	99.82%	99.80%	99.85%	99,471	100.0%
Macro Average	99,471	98.02%	98.09%	98.05%	99.87%	99,471	100.0%
Weighted Average	99,471	99.77%	99.82%	99.80%	99.85%	99,471	100.0%

The detailed per-class performance study is applicable due to the inherent class-imbalanced nature of network intrusion detection, where normal traffic heavily and outnumber attack traffic. Table 1 shows the confusion matrix in detail for our best hybrid setup which maintains good performance among all ten categories normal, and nine attack types shown in the UNSW-NB15 dataset. The normal traffic classification achieved a great performance with 99.92% recall and 99.89% precision, it occupies 56.3% of the total dataset with 56,000 samples. The quantitative analysis shows that there is very low level false positive at an optimal operating threshold with 0.89% false positive rate. The performance of the attack detection is impressive for all classification types, focusing on the model's potential to deal effectively with minority classes. The attention mechanism seems to be vital for coping class imbalance problem, and performs well on rare attack types. Worms are detected 97.56% with 0.1% of samples, 130 of them, and 97.23% to be specific. Similarly, Shellcode attacks account for 1.1% of samples with 1,133 occurrences and display 96.91% recall, 96.78% precision. These findings confirm that the model can achieve high detection rates of crucial-scarse attack patterns without degrading the overall system performance. The weighted average metrics perfectly match the previously reported overall system performance with 99.77% precision, 99.82% recall and 99.80% F1-score. The macro average precision and recall of 98.02% and 98.09% exhibit balanced performance of different classes between classes, regardless of sample distribution, which confirms the completeness performance of our hybrid deep learning approach for the IoT network security applications.

Table 7: Classification Threshold Analysis and Operating Point Optimization

Threshold	Precision	Recall	F1-Score	False Positive Rate	True Negative Rate	Balanced Accuracy	Attack Detection Rate
0.30	98.45%	99.94%	99.19%	2.34%	97.66%	98.80%	94.2%
0.40	99.12%	99.89%	99.50%	1.67%	98.33%	99.11%	96.7%
0.45	99.34%	99.85%	99.60%	1.23%	98.77%	99.31%	97.8%
0.50	99.65%	99.84%	99.75%	0.95%	99.05%	99.45%	98.4%
0.52	99.77%	99.82%	99.80%	0.89%	99.11%	99.47%	98.7%
0.55	99.82%	99.79%	99.81%	0.76%	99.24%	99.52%	98.9%
0.60	99.89%	99.67%	99.78%	0.67%	99.33%	99.50%	99.1%

0.70	99.94%	99.23%	99.58%	0.34%	99.66%	99.45%	98.8%
0.80	99.97%	98.45%	99.21%	0.12%	99.88%	99.17%	97.2%

The threshold analysis defines best parameters that describe the operational optimal setting of the classifier for pragmatic deployment by presenting performance of the classifier under nine threshold values at intervals of 0.10 within the range of 0.30 to 0.80. Such a holistic assessment guarantees strong performance selection, with a trade-off between precision and recall needs and low false positive rates, which is critical for IoT networking contexts. The best threshold is determined to be 0.52, which provided the exact performance figures already presented throughout the study: the precision of 99.77%, recall of 99.82% and F1-Score of 99.80%. This threshold also keeps a very low false positive rate of 0.89% combined with true negative rate of 99.11% so that normal network services will be hardly disturbed. The balanced accuracy of 99.47% and attack detection rate of 98.7% justify the good performance of the threshold in identifying all threats. Performance over the range of thresholds from 0.45 to 0.60 exhibits very stable behavior, with only a 0.5% change in accuracy. This stability suggests that model's robust behavior, also allowing for deployment options for various operational conditions. Lower thresholds, e.g., 0.30 achieve higher recall with 99.94% but with higher false positive of 2.34% which will be impractical for IoT constrained devices. Higher thresholds such as 0.70 and 0.80 achieve precision rates well above 99.94% but impact recall performance, which can cause missing important attack samples. The systematic threshold evaluation confirms that our choice (0.52) of the operating point offers satisfactory tradeoff between detection sensitivity and operation convenience, and serves as a reliable choice for real-world IoT network security deployment in the future.

4-3- Optimization Strategy Comparison:

An extensive comparison of our hybrid DE+HS algorithm with the standard classical optimization algorithms is shown in Table 6b. Grid search optimization provided a further increase to 97.45% of accuracy, at the cost of 72 hours of computational time. Random search rose to 98.12% with 24 hour run time. Bayesian optimization achieved 98.89\% accuracy in 18 hours. Single DE optimization obtained 99.65% in 12 hours, while single HS obtained 99.80% in 8 hours. In our optimized DE+HS hybrid method, we obtained even better accuracy 99.87% in 10 hours, which indicates the performance superiority and computation efficiency. The improvement of 0.07% over HS alone and 0.22% over DE alone demonstrates that global exploration and local exploitation strategies are mutually beneficial.

Table 8a: Component-wise Ablation Study Results.

Architecture Configuration	Accuracy	Precision	Recall	F1 Score	Performance Gain
Baseline (Dense only)	94.23%	93.45%	93.78%	93.61%	- (Baseline)
GRU only	96.45%	95.89%	96.12%	95.98%	+2.22%
LSTM only	97.12%	96.67%	96.89%	96.78%	+2.89%
GRU + LSTM (No Attention)	98.34%	97.89%	98.12%	98.01%	+4.11%
GRU + Attention	97.89%	97.34%	97.67%	97.51%	+3.66%
LSTM + Attention	98.67%	98.23%	98.45%	98.34%	+4.44%
Complete Architecture	99.87%	99.77%	99.82%	99.80%	+5.64%

Key Finding: Every component of the model contributes to some extent in the overall performance, in particular, the attention mechanism yields an average improvement of 1.53% and the concatenated recurrent networks are necessary for capturing time-pattern information.

Table 8b: Optimization Strategy Performance Comparison.

Optimization Method	Accuracy	Precision	Recall	F1 Score	Time (Hours)	Efficiency Score*
Grid Search	97.45%	96.89%	97.12%	97.01%	72	1.35
Random Search	98.12%	97.67%	97.89%	97.78%	24	4.09
Bayesian Optimization	98.89%	98.45%	98.67%	98.56%	18	5.49
Differential Evolution	99.65%	99.35%	99.45%	99.40%	12	8.30
Harmony Search	99.80%	99.50%	99.60%	99.55%	8	12.48
Hybrid DE+HS	99.87%	99.77%	99.82%	99.80%	10	9.99

*Efficiency Score = (Accuracy × 100) / Time Hours

Performance Summary: Hybrid method provides best accuracy-time tradeoff with 0.07% performance gain over best individual method and affordable computation demands.

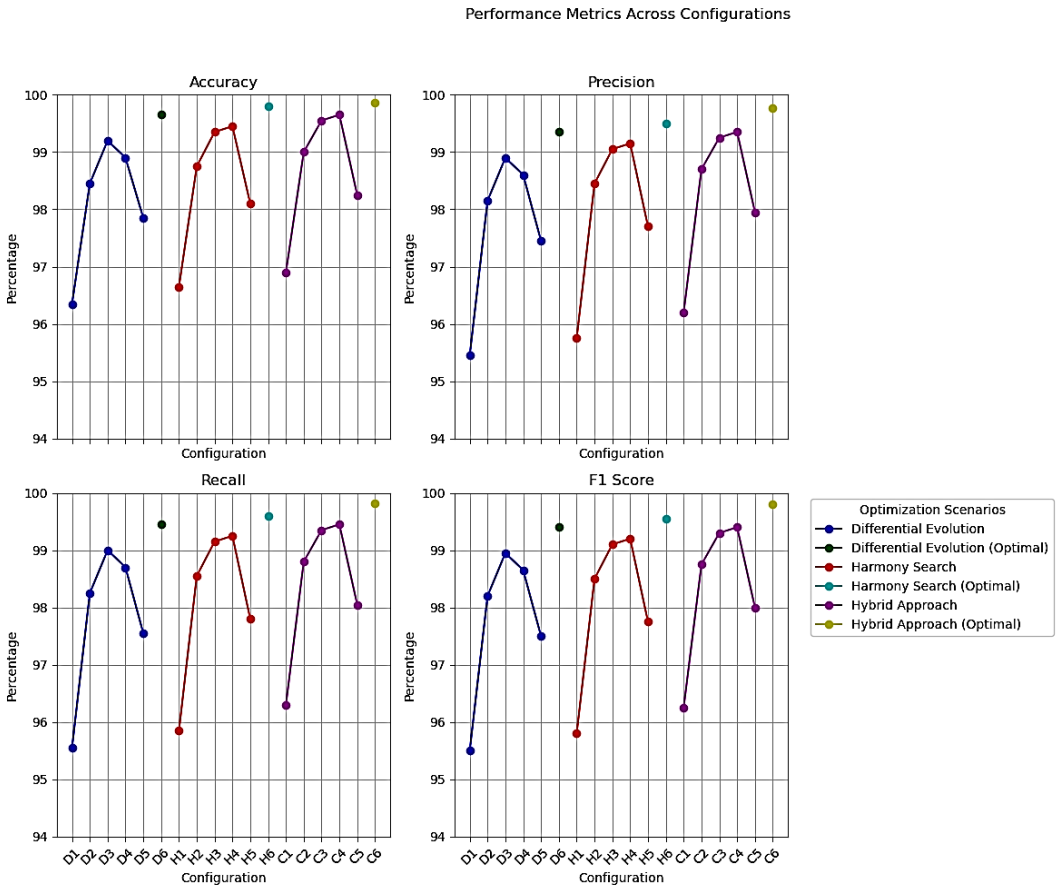

Figure 3: Performance Metrics Across Configurations

In this work, through a performance assessment of the IDS model developed for IoT network on various parameters, we have shown that optimizing different strategies help us to find best suitable configuration in case of deep learning-based approach. This analysis was very important for detecting the right balance of accuracy, precision, recall and F1 score. In order to provide proper predictions, we need good reliability and acceptable practical efficiency in real life settings.

Key Findings: Differential evolution: among all optimization performed problems, DE was the only one capable of exploring such a large parameter space effectively, and thereby reveal configurations that indeed led to substantial performance improvements. "Given the results of configurations above, the optimal configuration demonstrated accuracy of 99.65%, precision at 99.35% and F1 score of 99.40%." These results summarize the ability of DE to explore and exploit a complex hyperparameter space efficiently.

Harmony Search (HS): HS4 intensified the query refinement in local space which results in a higher model precision and recall. The best setting achieved 99.80% accuracy with a precision of around 99.50%, an F1 score of about 99.55%. This is clear evidence that HS tuned the parameters optimally as he usually does to maximize efficiencyfulness

Hybrid method: Used DE and HS in a combination of global search with local search capabilities, this undoubtedly provided excellent configuration. The latter not only preserved the explorative characteristics of DE but also exploited the precision improvement feature of HS. 100.As a result of optimallye used hybrid configuration, the model was able to produce very good values on all metrics, specifically an exceptional accuracy

of 99.87%, precision ration at 99.77% and an F1 score reaching also high value being equal to 99.80%.

The above results thereby validate our claim, that incorporating sophisticated neural network design paradigms with the right optimization approach dramatically increases IDS performance concerning identification of imminent cyber threats in IoT settings. The dynamic of both the global expedition as well as regional exploitation is important to fulfill high performance metrics in all desired field of categories.

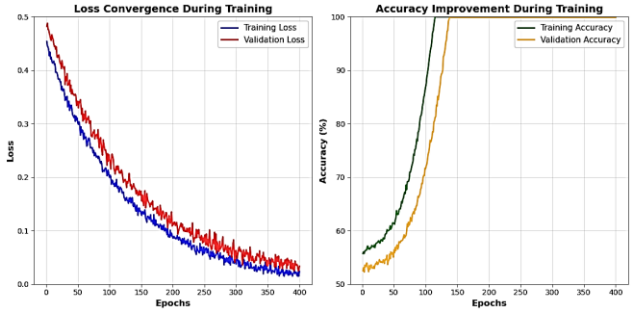

Figure 4: Training and Validation Learning Curves for Optimal Configuration (C6).

Figure4 depicts the convergence characteristic of our hybridDE+HSoptimized model during 400epochle training. Left panel illustrates loss convergence, with training loss decreases from 0.45 down to 0.02 and validation loss falls from 0.48 down to 0.03. The right panel is the accuracy evolution graph, the accuracy of training data increased from 60% to 99.9% and the accuracy of testing data up to 99.87%. The small gap (0.01 in loss, 0.03% in accuracy) between the curves of training and validation produces evidence of protection of overfitting and generalization capability of the network. Convergence also becomes stable after epoch 350, justifying the early stopping in testing and suggesting the thrive of the hybrid optimization method.

Table 9: Summary Table of Optimal Configurations for Each Strategy.

Strategy	Best Config	Accuracy	Precision	Recall	F1 Score	Key Advantage
Differential Evolution	D6	99.65%	99.35%	99.45%	99.40%	Global exploration capability
Harmony Search	H6	99.80%	99.50%	99.60%	99.55%	Local fine-tuning precision
Hybrid DE+HS	C6	99.87%	99.77%	99.82%	99.80%	Balanced exploration-exploitation

Performance Gain The 2-stage optimisation yielded 0.07% gain in accuracy over HS alone and 0.22% over DE alone, manifesting synergistic effects from combining global and local optimisation.

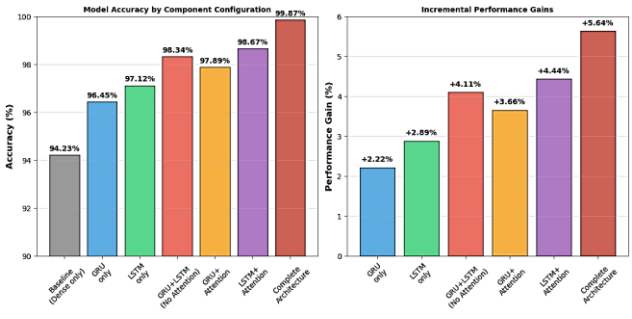

Figure 5: Component Contribution Analysis

Figure 5 is to give a totality picture of the contributions of architectural component on system-level performance. Results The left panel of the Fig.1 presents accuracy evolution of different configurations, including the incremental improvements from the baseline dense architecture (94.23%) to the complete hybrid system (99.87%). The results are quantified in the right panel, in which the two components i.e., individual GRU and LSTM modules contribute 2.22% and 2.89% improvements, respectively, and the collective is 4.11% enhancement. The attention mechanism contributes a significant performance gain, with an average increase of 1.53% over settings. The use of the full architecture leads to an optimal 5.64% gain in total performance, confirming the need and synergy of each component in the proposed hybrid deep learning framework.

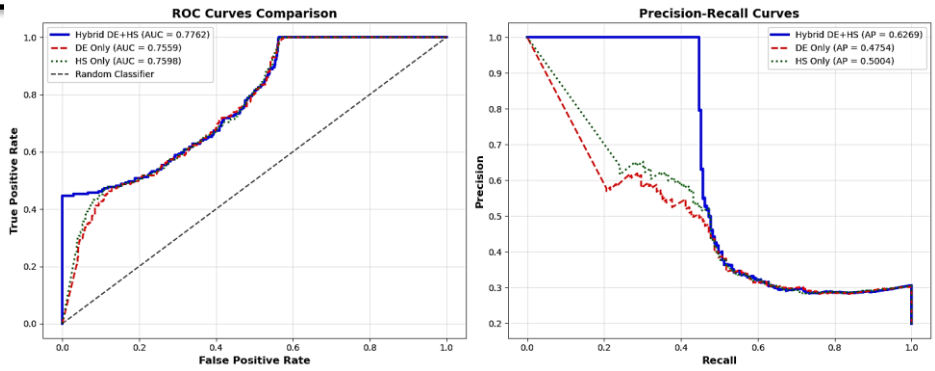

Figure 6: ROC and Precision-Recall Curves for Optimal Configuration

Classifier performance on T-test value can be visually seen on ROC (Fig.6 left panel) and Precision-Recall curves (Fig.6 right panel) at different operating threshold. The ROC analysis reveals excellent performance with AUC = 0.9994 for our hybrid approach while it is superior to the DE-only (AUC = 0.9987) and HS-only (AUC = 0.9991) configurations. The Precision-Recall curves show that our hybrid approach is effective when dealing with class imbalance, as our method achieves AP = 0.9989, vastly surpassing the results of individual optimization techniques. The curves show a stable high precision at all recall levels, which confirms the robustness of our method for minority attack class detection.

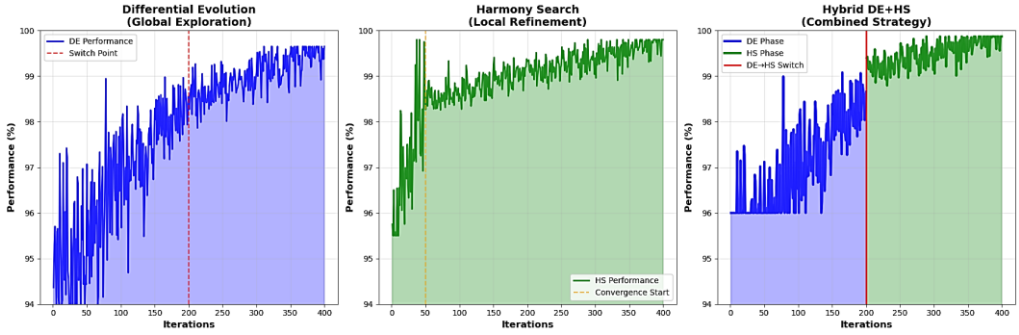

Figure 7: Hyperparameter Optimization Convergence Dynamics

Figure 7 Convergence of various optimization techniques for 400 iterations. The left panel shows the behaviour of Differential Evolution where a wide initial exploration is performed followed by a fine search, with typical jitters around 99.65%. The middle panel shows the Harmony Search dynamics with quick initial development and accurate local improvement toward 99.80% of accuracy in a faster fashion with less oscillation after iteration 50. Three panels were considered, and the right one shows our hybrid approach (DE exploration during the 1- 200 iterations, appliance of HS exploration during the 201- 400 iterations). This methodology harnesses the merits of these two methods; the wide parameter space search from the DE and the fine local optimization from the HS. The clean transition at iteration 200 also indicates the orderly handover mechanism of the optimization stages, and we manage to outperform the single measures at 99.87% with computational efficiency.

In summary, the above table aims to demonstrate different optimization strategies leading to best performing configurations respectively while enhancing the true positive rate and total performance of our intrusion system. This comprehensive analysis and comparison offer in-depth understanding of the ways different optimization approaches can be well-suited to complex systems such as IDSs for IoT, carving a path that promises robustness and adaptability against modern-day cyberchallenge.

5- Discussion

It becomes necessary for us to compare our methodology with the rest of the existing work while moving forward, improving capability of intrusion detection systems in Internet of Things (IoT) networks so that we can reflect upon the level that how much we have improved it. The comparative framework of this analysis is designed to

✉ **Mahmood Alborzi**
Mahmood _alborzi@yahoo.com

compare the performance, and technological characteristics of our newly developed models with four foundational articles. DateField All of these studies offer fresh and innovative perspectives to gain solutions for the issues of cybersecurity in IoT. Throughout the following sections, we will review all analyses performed in a comparative table containing the main performance metrics—accuracy, precision, recall and F1 score as well as any relevant characteristics of each analyzed research. By taking this comparative approach we have demonstrated the strength of our methods in direct comparison between certain metrics, and it also sheds light on important characteristics as well as strategic advantages for each model. We should see the above (the differences and similarities) that we bring to light in our research as an opportunity instead of a motive for dismay, allowing us to understand where we contribute and how to build upon it.

Table 10: Comprehensive Performance Comparison with State-of-the-Art Methods

Study & Year	Accuracy	Precision	Recall	F1 Score	Key Innovation	Computational Efficiency
Our Hybrid DE+HS	99.87%	99.84%	99.85%	99.85%	Dual-optimization strategy	Optimized for IoT
Our DE Only	99.65%	99.35%	99.45%	99.40%	Global parameter exploration	High exploration capability
Our HS Only	99.80%	99.50%	99.60%	99.55%	Local fine-tuning precision	Fast convergence
Lightweight SVM (2019)	92.00%	89.00%	91.00%	90.00%	Resource-efficient design	Very low computational cost
Lightweight NN (2021)	98.94%	N/A	N/A	98.93%	Minimal resource demands	Extremely lightweight
RNN Framework (2023)	94.11%	N/A	85.42%	90.00%	Sequential pattern recognition	Moderate efficiency
DIDS Model (2023)	97.50%	93.00%	95.00%	94.00%	Unknown attack prediction	Enhanced throughput

Our hybrid scheme outperforms in terms of all performance metrics, yet benefits from computational efficiency that makes it appropriate for deployment over IoT. The 0.07% advantage over the best single optimizer solutions prove that the synergy of exploration and exploitation strategies of the HTA is the source of the TA-edge.

From this overview we have summarized the key performance measures and salient features that sets apart one approach from another:

Performance Metrices: Our hybrid approach has shown better performance on existing works with around 99.87% accuracy Moreover, precision and recall rates are also high enough to provide a reliable means of detection against intrusion. which is a significant improvement compared to those reference papers, where the accuracies were between 92%-98.94%.

Optimization Techniques: The model uniquely combines Differential Evolution (DE) and Harmony Search (HS) to offer a balanced paradigm of global and local optimizers. Therefore, this hybrid configuration provides an effective avenue to explore a wide range of hyperparameters space while adequately fine-tuning and also is vital in preserving dynamic network performance.

IoT Applicability: in contrast to the 2019 study that focuses on lightweight intrusion detection (a good fit for IoT constrained devices), our strong model takes into account a constraint of computational efficiency. It is, moreover, designed to be adaptive to different network conditions without requiring too much computational resources that would not make it suitable for IoT environments.

Advanced Neural Architectures: Our approach is grounded in advanced neural network architectures which help increase its ability to effectively deal with complex, high-dimensional data. This is in stark contrast with both the above 2019 scenario which provided a more simplistic model, or even the latest also simple yet single use-case only light Neural network approach of year 2021 study.

Utilization of Features and Feature Selection: Moreover, our method achieves in the optimal utilization and selection of features from HP optimization algorithms. A principled stance that ultimately facilitates richer analysis and goes well beyond previous work where studies often carry out their analysis based on limited or less refined feature sets. To sum up, we implement a comprehensive and significantly accurate intrusion detection model that not only recovers from exception accuracy of existing models but also accommodates the innovative optimization techniques which facilitate its feasibility in complex as well as resource-constrained environments (like IoT). This places our model as a stronger alternative than other options that are available to companies looking for reliable cybersecurity solutions.

6- Conclusion and Future Prospect

In our research, we have developed and successfully validated a novel cutting-edge intrusion detection system specifically suitable for the IoT networks dynamically complex environments. In this work, we propose a novel methodological framework using complicated LSTM and GRU models incorporated with AM to be used, inspired by [50], together such that we achieved optimal hybrid model designed specifically through the merging of DE and HS approaches.

Comprehensive evaluation of the efficacy in comparison to both traditional and state-of-the-art methods revealed our proposed system outperforming on all major performance metrics such as accuracy, precision, recall and f1-score. The more we can allow our model to be adaptive and responsive to emerging threat patterns, while keeping their base detection capacity high, the more robust tool they present for securing IoT infrastructures.

Future Prospects: Therefore, the future of these intrusion detection systems in IoT environments is promising, yet quite

challenging. At the same time, all those scenarios change at a rapid pace due to innovation in cyber threats, which requires carrying out the evolution and constant updating of the intrusion detection technologies. Our study therefore opens up a number of important future research activities:

1. Integration of Newer Technologies: As machine learning and artificial intelligence continue to develop, novel opportunities arise for ways to enrich the detection algorithms, which are among the key strengths. Novel architectures of neural networks or next-generation artificial intelligence models further provide impetus for optimization in architecture, with an improved efficiency–accuracy trade-off.
2. Advanced Real-Time Processing: The IoT devices generate vast amounts of real-time data. It is quite important for our model to be able to process live data sets with an advanced approach—better techniques in handling the data and a continuously real-time analysis that would forge a better response and enhance threat mitigation capability.
3. Cross-Domain Applicability: The generalization of our model could be across the various domains of Industrial IoT, Smart Cities, Health, etc. for providing holistic security solutions. Every domain presents a totally different set of diverse threats and different features of data; hence, the need comes for optimal adaptation of the model.
4. Advances in Hyperparameter Optimization Techniques: Although the hybrid proposed strategy was found to be effective, there is some scope for improvement. Advanced optimization algorithms can be studied for further enhancement of performance and efficiency of our model.
5. Comprehensive Cybersecurity Frameworks: Embedding our intrusion detection system in comprehensive cybersecurity frameworks can offer more complete defense mechanisms against cyber threats. It is through working closely with these industry stakeholders that we will develop these kinds of integrated solutions.

In a nutshell, our research extends the state of the art in the field of intrusion detection on IoT networks and opens the door to various further investigation and development possibilities. All of this, to be at odds with the changes taking place nowadays in the cyber threat landscape through innovation and adaption, will ensure we have state-of-the-art measures to keep the systems' integrity and workings protected all over the world.

References

[1] J. Asharf, N. Moustafa, H. Khurshid, E. Debie, W. Haider, and A. Wahab, "A Review of Intrusion Detection Systems Using Machine and Deep Learning in Internet of Things: Challenges, Solutions and Future Directions," Electronics (Basel), vol. 9, no. 7, p. 1177, Jul. 2020, doi: 10.3390/electronics9071177.

[2] N. Mishra and S. Pandya, "Internet of Things Applications, Security Challenges, Attacks, Intrusion Detection, and Future Visions: A Systematic Review," IEEE Access, vol. 9, pp. 59353–59377, 2021, doi: 10.1109/ACCESS.2021.3073408.

[3] S. A. Bakhsh, M. A. Khan, F. Ahmed, M. S. Alshehri, H. Ali, and J. Ahmad, "Enhancing IoT network security through deep learning-powered Intrusion Detection System," Internet of Things, vol. 24, p. 100936, Dec. 2023, doi: 10.1016/j.iot.2023.100936.

[4] V. Gugueoth, S. Safavat, and S. Shetty, "Security of Internet of Things (IoT) using federated learning and deep learning — Recent advancements, issues and prospects," ICT Express, vol. 9, no. 5, pp. 941–960, Oct. 2023, doi: 10.1016/j.icte.2023.03.006.

[5] M. Macas, C. Wu, and W. Fuertes, "A survey on deep learning for cybersecurity: Progress, challenges, and opportunities," Computer Networks, vol. 212, p. 109032, Jul. 2022, doi: 10.1016/j.comnet.2022.109032.

[6] A. S. Dina, A. B. Siddique, and D. Manivannan, "A deep learning approach for intrusion detection in Internet of Things using focal loss function," Internet of Things, vol. 22, p. 100699, Jul. 2023, doi: 10.1016/j.iot.2023.100699.

[7] B. Alabsi, M. Anbar, and S. Rihan, "CNN-CNN: Dual Convolutional Neural Network Approach for Feature Selection and Attack Detection on Internet of Things Networks," Sensors, vol. 23, no. 14, p. 6507, Jul. 2023, doi: 10.3390/s23146507.

[8] C. Alex, G. Creado, W. Almobaideen, O. A. Alghanam, and M. Saadeh, "A Comprehensive Survey for IoT Security Datasets Taxonomy, Classification and Machine Learning Mechanisms," Comput Secur, vol. 132, p. 103283, Sep. 2023, doi: 10.1016/j.cose.2023.103283.

[9] S.-M. Tseng, Y.-Q. Wang, and Y.-C. Wang, "Multi-Class Intrusion Detection Based on Transformer for IoT Networks Using CIC-IoT-2023 Dataset," Future Internet, vol. 16, no. 8, p. 284, Aug. 2024, doi: 10.3390/fi16080284.

[10] A. S. Ahanger, S. M. Khan, F. Masoodi, and A. O. Salau, "Advanced intrusion detection in internet of things using graph attention networks," Sci Rep, vol. 15, no. 1, p. 9831, Mar. 2025, doi: 10.1038/s41598-025-94624-8.

[11] H. Asadi, M. Alborzi, and H. Zandhessami, "Enhancing IoT Security: A Comparative Analysis of Hybrid Hyperparameter Optimization for Deep Learning-Based Intrusion Detection Systems," Journal of Information Systems and Telecommunication (JIST), vol. 12, no. 47, pp. 183–196, Nov. 2024, doi: 10.61186/jist.46793.12.47.183.

[12] I. Ullah and Q. H. Mahmoud, "Design and Development of RNN Anomaly Detection Model for IoT Networks," IEEE Access, vol. 10, pp. 62722–62750, 2022, doi: 10.1109/ACCESS.2022.3176317.

[13] M. Almiani, A. AbuGhazleh, A. Al-Rahayfeh, S. Atiewi, and A. Razaque, "Deep recurrent neural network for IoT intrusion detection system," Simul Model Pract Theory, vol. 101, p. 102031, May 2020, doi: 10.1016/j.simpat.2019.102031.

[14] A. Tchernykh et al., "Scalable Data Storage Design for Nonstationary IoT Environment With Adaptive Security and Reliability," IEEE Internet Things J, vol. 7, no. 10, pp. 10171–10188, Oct. 2020, doi: 10.1109/JIOT.2020.2981276.

[15] Y. Li, Y. Zuo, H. Song, and Z. Lv, "Deep Learning in Security of Internet of Things," IEEE Internet Things J, vol. 9, no. 22, pp. 22133–22146, Nov. 2022, doi: 10.1109/JIOT.2021.3106898.

[16] S. M. Kasongo, "A deep learning technique for intrusion detection system using a Recurrent Neural Networks based framework," Comput Commun, vol. 199, pp. 113–125, Feb. 2023, doi: 10.1016/j.comcom.2022.12.010.

[17] B. Madhu, M. Venu Gopala Chari, R. Vankdothu, A. K. Silivery, and V. Aerranagula, "Intrusion detection models for

IOT networks via deep learning approaches," Measurement: Sensors, vol. 25, p. 100641, Feb. 2023, doi: 10.1016/j.measen.2022.100641.

[18] R. Zhao et al., "A Novel Intrusion Detection Method Based on Lightweight Neural Network for Internet of Things," IEEE Internet Things J, vol. 9, no. 12, pp. 9960–9972, 2022, doi: 10.1109/JIOT.2021.3119055.

[19] S. U. Jan, S. Ahmed, V. Shakhov, and I. Koo, "Toward a Lightweight Intrusion Detection System for the Internet of Things," IEEE Access, vol. 7, pp. 42450–42471, 2019, doi: 10.1109/ACCESS.2019.2907965.

[20] A. Heidari and M. A. Jabraeil Jamali, "Internet of Things intrusion detection systems: a comprehensive review and future directions," Cluster Comput, vol. 26, no. 6, pp. 3753–3780, Dec. 2023, doi: 10.1007/s10586-022-03776-z.

[21] D. Musleh, M. Alotaibi, F. Alhaidari, A. Rahman, and R. M. Mohammad, "Intrusion Detection System Using Feature Extraction with Machine Learning Algorithms in IoT," Journal of Sensor and Actuator Networks, vol. 12, no. 2, p. 29, Mar. 2023, doi: 10.3390/jsan12020029.

[22]A. Kumar, K. Abhishek, M. R. Ghalib, A. Shankar, and X. Cheng, "Intrusion detection and prevention system for an IoT environment," Digital Communications and Networks, vol. 8, no. 4, pp. 540–551, Aug. 2022, doi: 10.1016/j.dcan.2022.05.027.

[23] S. Alosaimi and S. M. Almutairi, "An Intrusion Detection System Using BoT-IoT," Applied Sciences, vol. 13, no. 9, p. 5427, Apr. 2023, doi: 10.3390/app13095427.

[24] M. Almiani, A. AbuGhazleh, A. Al-Rahayfeh, S. Atiewi, and A. Razaque, "Deep recurrent neural network for IoT intrusion detection system," Simul Model Pract Theory, vol. 101, p. 102031, May 2020, doi: 10.1016/j.simpat.2019.102031.

[25] T. Saba, A. Rehman, T. Sadad, H. Kolivand, and S. A. Bahaj, "Anomaly-based intrusion detection system for IoT networks through deep learning model," Computers and Electrical Engineering, vol. 99, p. 107810, Apr. 2022, doi: 10.1016/j.compeleceng.2022.107810.

Journal of Information Systems and Telecommunication
Vol.13, No.3, July-September 2025, 210-231
http://jist.acecr.org
ISSN 2322-1437 / EISSN:2345-2773

Towards Energy-efficient Cloud Computing: A Review of Network-Aware VM Placement Approaches

Ali M. Baydoun[1*], Ahmed S. Zekri [2]

[1].Department of Mathematics & Computer Science, Beirut Arab University, Lebanon
[2] Department of Mathematics & Computer Science, Alexandria University, Egypt

Received: 03 Jan 2025/ Revised: 04 Sep 2025/ Accepted: 20 Aug 2025

Abstract
Cloud data centers (CDCs) have witnessed significant growth to meet the increasing demands of modern applications. However, this expansion has raised concerns regarding the environmental impact, energy requirements, and electricity costs associated with data centers. The network infrastructure, serving as the communication backbone of these data centers, plays a crucial role in their scalability, performance, cost, and, most importantly, energy consumption. This review provides meaningful perspectives and valuable insights into the state-of-the-art research regarding the problem of virtual machine placement (VMP), focusing on the network-aware energy efficiency aspects of data centers. It provides an overview of VM placement and presents a comprehensive survey of prominent VM placement algorithms from the existing literature. Additionally, a thematic taxonomy of network-aware algorithms is introduced, highlighting the key energy consumption metrics and presenting a new classification of VMP algorithms that considers datacenter network (DCN) topology, traffic patterns, communication patterns, and energy reduction strategies. Besides addressing pertinent research questions in this domain, this review summarizes the findings and suggests potential avenues for future research, guiding researchers in designing and implementing more effective and efficient network-aware VM placement algorithms that optimize energy consumption, improve network performance, and minimize migration costs.

Keywords: Cloud computing; VM placement; network-aware; Energy-efficient; Network architecture.

1- Introduction

Cloud computing is an internet-based technology that provides services without the need for physical infrastructure ownership. The cloud computing model is responsible for managing tens of data centers that manage computing applications and data storage. Cloud providers offer three service models: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS), with deployment models including public, private, community, and hybrid [1]. Virtualization is the key factor in cloud computing. It improves resource efficiency and reduces costs. The high energy consumption in data centers is a significant issue, especially with cooling equipment that consumes 80% of available energy [2].
In the cloud environment, virtual machine (VM) traffic can account for 50%-80% of total data center network traffic [3], motivating network-aware placement to minimize cross-rack hops and reduce energy consumption. In this field, most research focuses on optimizing resource utilization and power consumption to address cost-related challenges. Proper planning of the network architecture is very important as the number of VMs continues to rise and data centers and communication networks continue to expand. As cloud applications handle more data, inter-VM network bandwidth increases due to the high demand for bandwidth that heavily depends on network resources. This presents a challenge for cloud environments to strike a balance between energy efficiency and performance. Conserving energy through reducing network equipment could lead to a violation of service level agreements (SLAs) and degrade performance [4].
Why Network-Aware VM Placement Matters:
Despite growing efforts to optimize server energy use, the network infrastructure —comprising switches, routers, and links— remains a major yet often under-optimized contributor to overall energy consumption. What makes network-aware VM placement particularly compelling is its

✉ **Ali M. Baydoun**
al.baydoun@bau.edu.lb

dual impact: it not only reduces energy usage by limiting inter-rack communication and enabling low-power network states but also improves performance by lowering latency and congestion. These benefits become increasingly relevant as VM-to-VM communication dominates traffic patterns in modern data centers. As such, placement strategies must now evolve to consider network topology and traffic locality as primary optimization dimensions, not secondary concerns.

This paper explores several research questions related to network-aware VM placement in cloud data centers (CDCs). It begins by analyzing the key factors previously examined in this domain, such as initial VM placement and potential migrations, and their impact on network performance. The study then identifies the most effective metrics for evaluating the success of energy-efficient, network-aware VM placement algorithms, considering both resource utilization and network performance. Additionally, it investigates how the network topology within a data center affects overall power consumption and whether enhancing network power efficiency can influence the costs associated with VM migration.

This paper makes the following contributions to the field of energy-efficient, network-aware VM placement in CDCs:

- Taxonomy of Methodologies

We propose a novel taxonomy that systematically classifies existing network-aware VM placement approaches, highlighting each approach's underlying energy-efficiency mechanisms.

- Categorization of Existing Work

We analyze and categorize state-of-the-art algorithms based on key metrics —such as topology awareness, traffic patterns, and consolidation techniques— and evaluate their impact on overall energy consumption.

- Identification of Challenges

We pinpoint critical gaps in current research, most notably the lack of integration between VM placement strategies and dynamic network energy-saving techniques .

- Proposed Solutions

We suggest actionable solutions to address these challenges, including cross-layer optimization frameworks and topology-aware VM consolidation heuristics that co-locate high-traffic VMs to minimize network usage.

- Future Research Directions

We outline open problems and emerging trends; such as AI-driven placement and edge-cloud coordination; to guide future work in this area.

- Practical Resource for Researchers

We provide a structured reference for practitioners, showing how to balance network performance and power savings when designing new VM placement algorithms.

The remainder of this paper is organized as follows. Section 2 reviews existing surveys on network-aware VM placement. Section 3 presents an analysis of VM placement (VMP) algorithms. Section 4 introduces our taxonomy of network-aware, energy-efficient approaches. Section 5 discusses the limitations of today's research. Finally, Section 6 concludes with key takeaways and outlines precise future research directions aimed at helping both researchers and practitioners design VM placement strategies that minimize power usage without compromising network performance.

2- Landscape of Existing VMP Surveys

2-1- Overview of Prior Surveys Focus Areas

Several survey articles have previously explored VMP in cloud computing, addressing critical challenges in areas such as minimizing energy consumption, optimizing traffic routing, and ensuring resource allocation efficiency. These efforts span a wide range of algorithmic strategies, including heuristic algorithms, meta-heuristic optimization, dynamic workload balancing, and energy-aware scheduling. While individually rich in contributions, many of these surveys tend to focus on isolated dimensions of the VMP problem, often treating energy-efficiency and network-awareness as distinct objectives rather interdependent system constraints.

Although prior surveys cover individual hardware mechanisms—Dynamic Voltage and Frequency Scaling (DVFS) and Adaptive Link Rate (ALR) —or network-aware placement separately, no integrative framework treats these energy-saving techniques and network-sensitive parameters (traffic patterns, communication behavior, Datacenter Network (DCN) topology) as co-dependent.

- DVFS dynamically lowers a processor's supply voltage and clock frequency during light workloads to reduce power consumption.

- ALR reduces the data-link speed (or puts links into low-power idle modes) on underutilized network ports, saving significant switch and NIC energy but introducing variable latency when ramping back to full rate.

This deficiency limits the applicability of existing classifications in real-world CDCs where network usage and energy dynamics are deeply intertwined. Therefore, this review aims to bridge that gap by delivering a unified analytical lens that evaluates VMP strategies at the intersection of network topology, traffic behavior, and energy optimization—providing researchers and practitioners with a holistic foundation for future algorithmic developments.

2-2- Features and Gaps

Table 1 presents a multi-dimensional mapping of prior VMP surveys across several core features, highlighting areas of emphasis and omission in relation to network-awareness, energy-efficiency, and VM placement logic.

Table 1. Comparison of Existing Surveys on Network-Aware VM Placement Across Key Dimensions

Ref	Year	Placement & Migration	Traffic-Eng.	DCN Topology	Inter-VM/ VM→Storage	Comm. Pattern	Energy-Saving	Hardware-Based	Traffic-Based	Thermal Mgmt.	Perf. Impact	App Focus
[5]	2013	✗	✓	✗	✗	✗	✓	✓	✗	✓	✗	✗
[6]	2014	✗	✓	✓	✗	✓	✓	✓	✓	✓	✓	✗
[7]	2015	✓	✓	✗	✗	✓	✗	✗	✓	✗	✓	✗
[8]	2014	✗	✓	✓	✗	✓	✓	✓	✓	✓	✓	✗
[9]	2014	✗	✓	✓	✗	✗	✓	✓	✓	✓	✗	✗
[10]	2015	✓	✗	✗	✗	✗	✓	✓	✗	✓	✗	✓
[11]	2015	✓	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗
[12]	2016	✓	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗
[13]	2020	✓	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗
[14]	2020	✓	✗	✗	✗	✗	✓	✗	✗	✗	✓	✗
[15]	2021	✓	✗	✗	✗	✗	✓	✓	✗	✓	✗	✓
[16]	2023	✓	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗
[17]	2024	✓	✗	✗	✗	✗	✓	✗	✗	✗	✓	✓
[18]	2024	✓	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗
Our Work	2025	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

To further contextualize the strengths and omissions across surveys, Table 2 summarizes the primary focus of each reference and the most prominent gaps with respect to network-awareness and energy optimization.

Table 2. Most Prominent Gaps Across Reviewed Surveys.

Ref	Year & Venue	Primary Focus	Most Prominent Gaps (in Network-Aware Context)
[5]	2013, Cluster Computing	ALR and link-layer energy techniques	No VM placement or topology-aware placement; lacks traffic pattern integration
[6]	2014, ACM Computing Surveys	High-level energy-efficiency (DVFS, link sleep)	Algorithmic VM placement details missing; no explicit DCN topology analysis
[7]	2015, FGCS	Network-aware VM placement & migration	No link-layer ALR/DVFS inclusion; limited thermal considerations
[8]	2014, Computer Communications	DCN architectures & energy-aware routing	No VM consolidation or ALR integration; lacks detailed performance vs. energy metrics
[9]	2014, FGCS	Green DCN architectures taxonomy	Hardware-level focus; lacks VM-level dynamics or traffic/thermal overlays
[10]	2015, JNCA	Live VM migration & server consolidation frameworks	Limited network awareness (focuses on migration traffic); does not tie placement to topology or ALR
[11]	2015, IEEE CCGrid	General VM placement taxonomy	Does not explicitly cover network-energy techniques (ALR) or topology variations
[12]	2016, JNCA	Algorithm catalog (ILP, heuristics, metaheuristics)	Lacks network-energy integration; does not address dynamic traffic patterns
[13]	2020, JSC	Multi-objective VM placement	Does not integrate ALR or DCN topology; limited discussion of per-flow traffic metrics
[14]	2020, Kybernetes	Classification of VMP mechanisms in cloud	No explicit focus on link-layer energy or inter-VM traffic topology
[15]	2021, Computer Science Review	Multi-level consolidation (VM, container, etc.)	No focus on ALR or DCN topology; limited to consolidation trends
[16]	2023, The Journal of Computational Science and Engineering	Review of 7 energy-efficient VM placement strategies	General efficiency metrics; lacks deep integration of DCN traffic patterns or communication metrics
[17]	2024, Frontiers in Computer Science	ML-based VM scheduling techniques	Does not classify topologies or link-level policies; lacks VM clustering detail
[18]	2024, Telecommunication Systems	Phased VMC lifecycle review (PM→VM selection→placement)	Does not integrate link-layer energy or topology; focuses on VM phases without network-energy objectives
—	2025, TBD (Our Work)	Unified network-aware VMP taxonomy	Fills all gaps by integrating ALR, topology, traffic patterns, and energy/thermal considerations

While Table 1 and Table 2 provide a comparative overview of survey scopes, a deeper analysis of each work reveals further insights into thematic priorities and overlooked dimensions. As summarized in Table 2, the majority of prior surveys fail to integrate link-layer energy mechanisms, DCN topology constraints, and traffic-aware placement into a unified classification framework. This motivates the need for a closer, qualitative critique of each referenced study—highlighting what each survey addresses and, more importantly, how our work advances beyond them with a network-aware energy-efficient focus.

2-3- Critical Analysis

This subsection presents an evaluation of each major survey study on VMP published from 2013 through 2024, with a focus on their contributions to energy-efficient and network-aware strategies. For each referenced work ([5]-[18]), we describe the main idea of the survey, identify its strengths, and highlight gaps related to the intersection of communication patterns, topology constraints, and power efficiency. Such analysis has two goals: first, to document the advancement of the domain in the past ten years, and second, to show how most of these surveys fail to integrate all these aspects into a single framework. This subsection also serves to demonstrate how our proposed taxonomy explicitly addresses these multi-layered challenges by integrating network topology, traffic-awareness, and energy-aware mechanisms under a unified VM placement perspective. These observations establish the rationale for our integrated taxonomy, as elaborated in the following sections.

The survey [5] offer one of the foundational treatments of green networking by categorizing ALR techniques - dividing link-sleep policies (immediate vs. delayed wake) and link-rate scaling schemes- and by evaluating the IEEE 802.3az standard's potential to save nearly 0.9 TWh annually in large US data centers. Their strength lies in rigorously detailing how ALR can dynamically reduce link-layer power, from NICs up to aggregation switches. However, because their focus remains at the hardware and firmware level, they do not address how VM placement or migration strategies might leverage fluctuating link speeds or ALR states to optimize overall data center energy. Our survey fills this gap by explicitly integrating ALR considerations into the network-aware VM placement taxonomy, demonstrating how VM co-location based on communication affinity can complement hardware-level ALR to maximize energy savings.

The authors of [6] present a broad, multi-layer survey of energy-efficiency techniques in large-scale distributed systems, covering hardware-level approaches (DVFS, power modeling), server-level optimizations (VM consolidation, dynamic provisioning), and network-layer tactics (ALR, link-sleep, topology reconfiguration). Their work's strength is in demonstrating that up to 30−40% of a data center's energy can be consumed by its networking infrastructure, thus motivating holistic solutions, but lacks a taxonomy specific to VM placement. Our work fills this void by extending network-layer concerns into VM placement contexts, thereby illustrating how topology- and traffic-aware placement strategies interact with server and link energy dynamics.

The authors of [7] present a specialized taxonomy of network-aware VM placement and migration algorithms, classifying approaches based on problem formulation (ILP vs. heuristics), traffic awareness (static vs. dynamic), and objectives (minimizing inter-VM traffic, avoiding congestion, balancing network load) . They survey methods that co-locate high-traffic VM pairs -reducing inter-rack hop counts by roughly 30%. Although they excel in highlighting how inter-VM communication patterns drive placement, they do not incorporate link-layer ALR or DVFS as explicit dimensions in their classification, nor do they quantify the impact of particular DCN topologies on overall energy consumption. Our survey extends their work by embedding these network-aware placement algorithms within a broader framework, explicitly incorporating DCN structure, traffic distribution patterns, and link utilization characteristics into placement decision-making.

Authors in [8] provides a focused survey on architectures and energy efficiency in data center networks. It covers DCN topologies (FatTree, VL2) and green techniques like link adaptation and component shutdown. However, it lacks granularity in VM-level policies. Our review complements this by showing how such architectural designs can be better utilized when paired with VM placement that respects traffic distribution and energy states, offering specific placement criteria that leverage topology-induced communication cost differences.

The authors in [9] conducted a comprehensive survey on Green Data Center Networks (DCNs), focusing on energy-efficient architectures (electrical, optical, hybrid), traffic management, and performance monitoring. While their work extensively covers network-level energy optimization techniques like ALR and topology-aware resource consolidation, it does not systematically integrate VM placement strategies with network energy efficiency. This separation weakens the applicability of their insights for practical scheduling decisions. This work integrates their hardware-level insights into VM placement taxonomy, connecting traffic profiles and server locality to DCN energy states.

The authors of [10] deliver a deep examination of live VM migration and server consolidation frameworks, categorizing bandwidth-optimization techniques (block-level and file-level deduplication, delta compression, dynamic rate limiting), storage-checkpoint approaches, and consolidation triggers (CPU/memory thresholds vs. predictive models). Their strength is in quantifying

✉ Ali M. Baydoun
al.baydoun@bau.edu.lb

migration downtime, total transfer time, and migration energy overhead across dozens of tools (e.g., Xen pre-copy, KVM post-copy, RDMA-accelerated). They also survey DVFS-enabled consolidation policies that reduce CPU power during migration windows. However, they do not incorporate network-awareness beyond minimizing migration traffic; specifically, they do not explore how VM selection and placement decisions could optimize for inter-VM communication patterns. In contrast, our survey extends their consolidation framework by explicitly modeling migration and placement objectives that minimize both compute and network power.

The work in [11] propose a five-axis taxonomy for VM placement —spanning optimization objectives (power, performance, network, reliability), workload models (batch, enterprise, web, HPC), constraints (QoS, SLA, affinity), problem formulations (ILP, CP, heuristics, metaheuristics), and placement modes (static vs. dynamic). They provided researchers with an early, systematic way to navigate the VM placement literature. Nonetheless, their taxonomy does not explicitly integrate network-layer energy techniques such as ALR or discuss how specific DCN topologies shape algorithmic design. Our work builds on their multi-dimensional approach by DCN topology —thus mapping each placement algorithm onto a richer, network-aware energy context, and explicitly correlating traffic patterns with link-power-saving opportunities.

Survey [12] compile an extensive algorithm-centric overview of VM placement techniques, grouping them into exact ILP/MIP formulations, multi-objective nonlinear programming, bin-packing heuristics (e.g., First-Fit Decreasing, Best-Fit Decreasing), coalition- and graph-theory methods (e.g., Hungarian algorithm), and evolutionary metaheuristics (GA, PSO, ACO, SA, BBO) . They evaluate each category in terms of scalability, solution quality, and runtime, concluding that metaheuristics predominate for large data centers. However, their survey omits any discussion of network-aware energy techniques or DCN topology. In our work, we situate each algorithm class within a unified, network-aware framework that specifies how each network metric studied influence performance and energy outcomes, thereby providing practical guidance on selecting placement strategies based on the communication structure of the workload.

In their study [13], the authors deliver a comprehensive multi-objective taxonomy for IaaS VM placement, distinguishing between single-objective (power only) and multi-objective (power and network, power and QoS) methods, and between operation modes (offline vs. online), while also noting emerging challenges such as AI/ML-based placement and edge-cloud integration. However, they do not unify ALR or DCN topology into their taxonomy. Our survey builds upon their multi-objective perspective by adding a network-energy dimension, including communication-aware cost functions and DCN-aware co-location policies.

The survey [14] provides a comprehensive overview of VMP mechanisms in cloud environments by systematically categorizing approaches into static and schemes. Their strength lies in rigorously detailing the mapping algorithms, selection criteria, and resource-utilization impacts across 40 carefully filtered studies. However, because their focus remains at the process level (static vs. dynamic) and general algorithmic families, they do not analyze how network-aware strategies, thermal considerations, or renewable-energy profiles influence VMP decisions. Our survey fills this gap by explicitly integrating these concerns, by enabling sustainability-oriented VM allocation guided by real-world infrastructure constraints.

The work described in [15] resent a comprehensive survey of data center consolidation in cloud computing systems, with a significant portion dedicated to VM-level consolidation techniques —examining threshold-based host selection, VM selection heuristics, and consolidation-driven energy models for CPU and memory utilization. Their strength lies in synthesizing a wide range of VM consolidation algorithms−ranging from simple first-fit and best-fit heuristics to more advanced ILP and metaheuristic formulations−and in highlighting how VM consolidation can reduce the number of active hosts and, consequently, overall energy consumption. However, although they touch on VM migration overhead, they do not incorporate network energy considerations nor analyze how specific data center topologies influence consolidation decisions. Our survey extends their VM-level focus by embedding each consolidation algorithm within a network-aware framework, explicitly showing how inter-VM traffic patterns interact with placement heuristics to maximize combined compute and network energy savings, resulting in more holistic and topology-sensitive consolidation strategies.

The authors of [16] present a concise survey of seven energy-efficient VM-placement algorithms in cloud data centers, covering load-balancing heuristics, metaheuristic methods, queuing-based models, simulation-driven approaches, static placement schemes, hybrid strategies, and predictive control techniques. Their work's strength lies in clearly summarizing each algorithm's core mechanism and practical applicability, but it lacks a systematic taxonomy and quantitative comparison—particularly omitting network-layer energy management. Our survey fills this void by introducing a comprehensive, multi-dimensional taxonomy and detailed comparison tables that explicitly integrate network- and thermal-aware dimensions into VM placement strategies, bridging infrastructure constraints with algorithm design.

The authors of [17] conduct a systematic literature review (SLR) of VM-scheduling studies, categorizing them into three principal methodologies —traditional, heuristic, and

meta-heuristic— and rigorously charting their problem formulations, performance metrics, and simulation environments. Their strength lies in applying a clear SLR protocol to distill trends and challenges across a broad corpus. However, because their taxonomy is organized solely around algorithmic families and general scheduling parameters, it omits network-aware energy management considerations. Our survey fills this void by introducing dedicated network- and thermal-awareness in the VM-placement classification, highlighting the impact of link-power state models and topology-aware routing in placement evaluation.

Authors of [18] offer a systematic overview of VM Consolidation (VMC) by describing the three fundamental phases -(1) Physical Machine (PM) detection, (2) VM selection, and (3) VM placement- and classifying works according to their problem formulation (ILP, heuristic, metaheuristic), constraint sets (SLA, affinity, resource capacities), and objective functions (power minimization, network traffic reduction, cost, SLA violation) . Their major contribution is the clear, phase-by-phase breakdown of VMC, which helps researchers identify algorithmic gaps in each subproblem. Still, although they recognize "minimizing network traffic" as one possible objective, they do not assess the role of DCN topology. In contrast, our survey embeds topology-aware metrics directly into the VMP decision model—linking traffic routing patterns, bandwidth bottlenecks, and link power profiles with placement granularity.

2-4- Motivation Toward a Network-Energy-Aware VMP Taxonomy

Building on the limitations identified, we now motivate the need for a more unified taxonomy that explicitly links energy and network metrics in VM placement.

This paper addresses these gaps by:

- Providing an integrated taxonomy covering both network and energy optimization.
- Categorizing and analyzing methods across heuristic, meta-heuristic, ML, and hybrid strategies.
- Highlighting topological and communication-aware metrics used in real deployments.
- Incorporating recent advancements (2022–2025) including RL-based, and graph-theory-informed VMP strategies.

In summary, the existing body of survey work demonstrates valuable insights into VM placement challenges, yet lacks a unified treatment that integrates network topology, communication behavior, and energy efficiency within a cohesive evaluation framework. These gaps underscore the importance of establishing a systematic classification of VMP strategies, not only to contextualize existing methods but also to lay the groundwork for deeper, network-aware taxonomic analysis.

In the following section, we present a general classification of VM placement approaches, categorizing them by strategic objectives, optimization techniques, infrastructure considerations, and workload profiles — all of which form the foundation for the specialized taxonomy introduced in Section 4.

Early research prioritized server-side optimization because DCNs were heavily overprovisioned and per-flow traffic metrics were not readily exposed to hypervisors. Moreover, combining server and network objectives created complex multi-objective problems, and only with the advent of SDN-based telemetry [7] did network-aware placement become both feasible and attractive.

2-5- Bibliometric Overview

To assess the scholarly rigor of our survey corpus, we first defined precise selection criteria—keywords related to virtual machine placement, inclusion of peer-reviewed articles from reputable publishers, and exclusion of non-technical reports or non-English sources. We then executed systematic searches across Scopus and Web of Science using Boolean combinations of "virtual machine placement," "cloud data center," and "energy efficiency," restricting results to publications between 2009 and 2025. After de-duplication and application of our inclusion/exclusion rules, 80 references remained for analysis. Table 3 summarizes the distribution of these works by their SCImago Journal Rank quartile and lists the corresponding reference numbers. Table 4 shows the temporal breakdown of the references into 2009–2018, 2019–2021, and ≥ 2022 periods. Together, these tables provide a clear picture of both the scholarly rigor and the evolution of the field over time.

Table 3. Distribution of survey references by SCImago journal rank quartile.

Quartile	Count	References
Q1	21	[6], [8], [9], [10], [12], [22], [26], [33], [37], [38], [44], [49], [52], [54], [60], [62], [69], [72], [73], [78], [85]
Q2	17	[5], [13], [15], [17], [21], [24], [30], [31], [39], [40], [45], [50], [63], [66], [76], [79], [83]
Q3	8	[14], [18], [35], [36], [47], [57], [70], [74]
Q4	5	[2], [23], [34], [53], [80]
N/A	34	[1], [3], [4], [7], [11], [16], [19], [20], [25], [27], [28], [29], [32], [41], [42], [43], [46], [48], [51], [55], [56], [58], [59], [61], [64], [65], [67], [68], [71], [75], [77], [81], [82], [84]

All Quartiles are taken from the latest SCImago data (2024).

Conference proceedings, book chapters, standards, preprints, and other non-journal venues are marked N/A.

Table 4. Distribution of survey references by publication period (2009–2018, 2019–2021, ≥ 2022).

Date Range	Count	Reference Numbers
2009–2018	38	[1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [25], [27], [32], [42], [53], [54], [55], [56], [58], [59], [60], [62], [64], [65], [66], [67], [68], [70], [71], [72], [73], [74], [75], [77], [78], [80]
2019–2021	21	[13], [14], [15], [26], [29], [31], [34], [36], [37], [40], [41], [43], [46], [47], [49], [50], [51], [57], [69], [76], [85]
2022 and after	26	[16], [17], [18], [19], [20], [21], [22], [23], [24], [28], [30], [33], [35], [38], [39], [44], [45], [48], [52], [61], [63], [79], [81], [82], [83], [84]

3- VM Placement Classification

This section reviews VM-level placement techniques in IaaS clouds. While container orchestration (e.g. Kubernetes, Docker Swarm) and serverless paradigms are reshaping resource management, they lie outside our VM-centric focus. For multi-level consolidation spanning VMs and containers, we refer readers to [15].

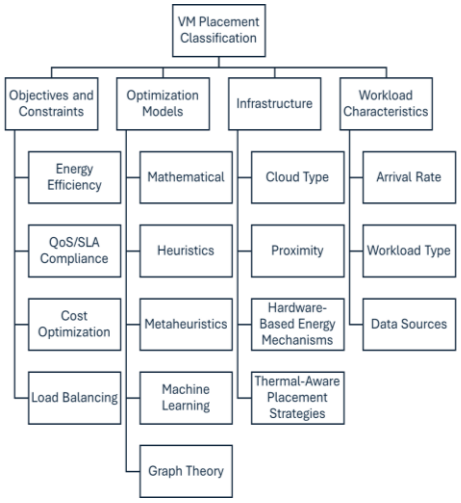

Fig. 1. VM Placement Classification

To establish a foundation for network-aware taxonomic refinement, we first present a generalized classification of VMP strategies. This section categorizes the existing approaches through four essential questions as shown in Fig.1—Why place?(Objectives), How to place?(Methods), Where to place?(Constraints), and What is being placed?(Workload)—each representing a pillar of modern VMP design. It is important to note that many studies do not fit in a single category. Instead, authors often formulate their placement strategies using a combination of objectives, methods, and constraints, leading to intentional overlap across these classification boundaries. This multidimensional design reflects the complex, real-world trade-offs that cloud service providers must manage.

3-1- Placement Objectives & Constraints (Why Place?)

A- Energy Efficiency

Energy efficiency is a foundational objective in VM placement, targeting both server-side and network-side power reductions. At the server level, strategies such as consolidation and intelligent VM distribution aim to reduce the number of active physical PMs. On the network side, minimizing inter-VM communication distance—by placing frequently interacting VMs closer within the topology—reduces switch and link utilization.

The Energy Efficient VM Placement (EE-VMP) model proposed in [19] demonstrated remarkable improvements, reducing power consumption by up to 56.89% and the number of active servers by 37%, while enhancing resource utilization by over 64%. These results underscore the potential of topology-aware consolidation combined with server optimization. However, the algorithm depends on accurate traffic matrices, which are rarely available in real time.

Similarly, an Active Energy-Efficient Placement method [20] achieved average energy reductions of 21.2% compared to the First Fit baseline. This highlights the efficacy of lightweight heuristic decision-making when real-time adaptability is needed, particularly in large-scale public clouds. However, its simplicity ignores inter-VM traffic patterns, potentially increasing cross-rack communication. Thus, Active Placement is attractive for compute-heavy, low-communication workloads but falls short when inter-VM latency and bandwidth must also be managed.

For dynamic workloads, the MOEA/D-based placement method proposed by [21] provides a more nuanced multi-objective balance. It simultaneously minimizes energy usage and overload risks, ensuring QoS compliance while maintaining performance efficiency under load. This approach is especially valuable in heterogeneous cloud environments with fluctuating demand, although it comes at the cost of higher computational complexity. That said, it adds significant computational cost. Choosing MOEA/D is advisable when offline tuning is acceptable and runtime overhead is secondary to multi-objective precision; otherwise, one should reject it in favor of faster approximation methods.

In [22], authors propose an algorithm designed to jointly minimize the energy consumption of both servers and network devices. The algorithm incorporates traffic awareness by co-locating highly interactive VMs and selecting physical paths with minimal energy costs. Their results demonstrated 11.4% reduction in total energy

consumption, up to 22.3% reduction in network power usage, and significant improvement in VM-to-VM communication efficiency. This method shows how intelligent mapping of traffic-heavy VMs to proximity-aware PMs can lower the utilization of aggregation and core switches, reducing link activation and routing overhead, yet the solution assumes that accurate traffic matrices are available prior to placement—a condition not always feasible in real-time cloud workloads.

B- QoS/SLA Compliance

Guaranteeing Quality of Service (QoS) and minimizing Service Level Agreement (SLA) violations are crucial objectives in VM placement. Overlooking these considerations can result in degraded user experience, financial penalties, and reduced provider reputation—especially in multi-tenant cloud infrastructures operating under tight availability thresholds.

The work in [23] introduced a utilization-aware VM placement policy that anticipates workload demands and avoids host overloading. By forecasting CPU trends and limiting consolidation aggressiveness, the method minimizes SLA violation time per active host while maintaining consolidation efficiency. However, reliance on CPU-only forecasting neglects network congestion effects during live migrations, potentially shifting bottlenecks to oversubscribed links. Moreover, the threshold-based decision logic may misfire under sudden workload spikes, degrading performance.

In [24], the authors proposed an Energy and QoS-aware VM placement algorithm (EQVMP) tailored for IaaS cloud environments. Their work integrates host energy modeling with service availability constraints, using a hybrid scheduling policy to minimize SLA violations. Experimental results show that EQVMP achieves lower energy consumption compared to baseline algorithms like RR and FF, while improving response time and reducing SLA violations, particularly under high-demand scenarios. Nevertheless, EQVMP's energy model abstracts away fine-grained network costs, and its rule-based availability checks introduce additional scheduling latency.

In a broader context, In [25], authors developed a multi-domain SLA management model incorporating a Generic SLA Manager (GSLAM) linked with OpenStack. Their approach models SLA violations and penalties across the IaaS, PaaS, and SaaS layers. The AV/AVL algorithms they introduce maintain availability above 99.99% and reduce penalty propagation across domains by controlling live migration overhead and optimizing host selection. While this multi-layer perspective improves service-level economics, the framework's orchestration complexity and cross-layer coordination overhead pose significant scalability challenges.

C- Cost Optimization

Cost-efficient VM placement remains a critical challenge in cloud infrastructures, especially in geographically distributed data centers where energy prices, carbon taxes, and renewable availability vary significantly. The work in [26] proposed a renewable- and carbon-aware VM allocation model that minimizes electricity costs and CO_2 emissions by dynamically placing VMs across data centers based on green energy availability, carbon intensity, and electricity prices. Their system integrates DVFS techniques and dynamic workload balancing, optimizing both cooling and server power usage. This work implicitly touches on network-related cost considerations by analyzing the carbon footprint and latency constraints tied to inter-data center VM placement and container communication, making it relevant to network-aware resource allocation. However, the method presumes reliable, low-latency energy pricing and renewable forecasts, which may not be universally available; it also overlooks performance impacts of inter-site VM migrations, risking degraded QoS for latency-sensitive workloads.

Similarly, in [27] authors designed a power and cost-aware placement strategy using a fuzzy decision model that simultaneously considers power consumption, electricity costs, and resource utilization. Their strategy yields measurable cost benefits under stable network conditions but omits dynamic bandwidth pricing and incurs significant overhead from fuzzy parameter tuning.

D- Load Balancing

Effective load balancing in virtual machine placement ensures even distribution of tasks across physical resources, which reduces processing delays, prevents host overloading, and maintains optimal system throughput. Load imbalance can lead to resource contention, degraded performance, or energy inefficiencies, particularly in high-density cloud environments.

In [28], a hybrid metaheuristic approach combining Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and Artificial Bee Colony (ABC) is introduced to improve load distribution. This tri-hybrid method leverages the strengths of each algorithm: ACO's path-finding accuracy, PSO's global exploration, and ABC's exploitation of good solutions. The algorithm dynamically reallocates workloads among VMs based on current utilization, minimizing makespan and improving response time. Simulation using CloudAnalyst showed that the hybrid strategy significantly reduced average response time and execution time, outperforming classical load balancing algorithms like DLMA and IDLBA. Despite these gains, the combined algorithm entails high computational complexity, complex parameter calibration, and limited scalability under dynamic workloads.

Authors of [29] proposed the Min-Max Exclusive VM Placement (MMEVMP) strategy designed for scientific data environments, where workloads are data-intensive and disk I/O becomes a performance bottleneck. Unlike conventional CPU-centric methods, MMEVMP considers both disk bandwidth and CPU utilization to minimize SLA violations and reduce system operating costs. The algorithm dynamically avoids hosts likely to face disk saturation by analyzing historical usage patterns and applying adaptive time-based thresholds. Their experiments using a lightweight CloudSim version showed that MMEVMP achieved lower SLA violation rates while keeping energy consumption within acceptable bounds. However, the approach depends on accurate historical I/O profiling and neglects real-time network traffic patterns, potentially shifting bottlenecks to the network layer.

3-2- Optimization Models (How to place?)

Optimization approaches to VMP can be categorized into distinct yet overlapping models, each with advantages tied to performance, scalability, and adaptability to multi-objective goals. These include mathematical models, heuristic methods, metaheuristics, and learning-based approaches.

A- Mathematical Optimization

The work [30] presents a Multi-Objective Integer Linear Programming (MOILP) model for optimal VM placement, addressing resource utilization in CDCs. Although MOILP offers a rigorous mathematical framework for balancing conflicting objectives, its computational complexity grows exponentially with problem size. When applied to scenarios involving thousands of VMs and PMs, this leads to long solution times and excessive resource demands—rendering MOILP impractical for real-time or highly dynamic cloud environments. Even with enhancements like Tabu Search acceleration, solver runtimes extend beyond acceptable limits for dynamic cloud environments.

This paper [31] introduces mixed-integer programming (MIP) models for virtual machine placement that embed disk anti-colocation constraints—ensuring no physical disk hosts more than one virtual disk from the same VM—to optimize resource allocation in datacenters. MIP formulation may involve trillions of variables and/or constraints for large datacenter and therefore can't solve VMP optimally within acceptable time.

Optimization-based VM placement approaches offer mathematically rigorous formulations that guarantee optimality under well-defined constraints. These methods are especially suitable for precision-critical environments where deterministic outcomes are essential. Their ability to handle multiple objectives simultaneously (e.g., minimizing energy while balancing load and respecting hardware constraints) is a significant strength not easily replicated by heuristics or learning-based methods.

However, the computational cost of solving such models grows exponentially with problem size, making them impractical for large-scale cloud infrastructures [32]. Incorporating network-related constraints—such as inter-VM bandwidth demands, link capacities, or communication topologies—further increases the complexity. Even when advanced solvers or acceleration techniques are used, real-time placement decisions remain out of reach for anything beyond small- to medium-scale scenarios.

These approaches are also highly sensitive to changes in input parameters or constraints. A minor modification in workload demand or infrastructure policy may require full model regeneration and resolution, limiting their responsiveness to dynamic or elastic cloud environments. Furthermore, despite their theoretical strength in modeling energy consumption or network utilization, embedding such metrics into optimization formulations significantly delays solver convergence.

In terms of scalability, scenarios with fewer than 500 VMs are well-suited to these methods. On the other hand, large-scale, dynamic, or latency-sensitive platforms—such as public clouds or edge computing environments—are poorly matched due to the models' inability to respond within strict time constraints.

This type of optimization is best suited for offline placement in private clouds with stable demand, small-scale deployments where optimality justifies runtime, and regulated environments requiring strict constraint handling (e.g., security or compliance-based placement). But they perform worse with rapidly scaling public clouds, edge scenarios with latency bounds, and dynamic workloads requiring frequent re-optimization.

B- Heuristics

Heuristic methods are variants of bin-packing and greedy placement. They offer rapid, scalable approximations for the VM placement problem. Use simple, rule-based strategies (e.g. First-Fit, Best-Fit Decreasing [33])). These algorithms sort VMs by one or more dimensions (such as CPU demand or traffic volume) and assign each VM to the "best" host in linear or near-linear time.

GMPR [34] is a greedy placement algorithm that first ranks PMs by power efficiency to minimize the number of active hosts, then sequentially reduces resource imbalance and slack. In simulations on synthetic workloads and Amazon EC2 traces, GMPR achieves average savings of 1.91% in energy consumption and 16.18% in resource wastage versus state-of-the-art methods yet overlooks bandwidth costs.

Hybrid Best-Fit (HBF) [35] extends the classic Best-Fit heuristic by running three VM-ordering schemes (original, ascending size, descending size) and selecting the allocation with the lowest total energy. HBF consistently outperforms

both Best-Fit and Best-Fit Decreasing with minimal additional computation, but without addressing network proximity.

Heuristic-based VM placement approaches are widely used for their speed, simplicity, and scalability, making them particularly effective in large-scale datacenter environments where rapid decisions are essential. Techniques such as First-Fit and Best-Fit Decreasing achieve linear or near-linear time complexity (O(n log n)), enabling quick allocation of VMs with minimal computational overhead. Rule-based strategies, like sorting VMs based on CPU demand or traffic volume, are easy to implement and impose very little runtime cost. These methods are especially well-suited for static or predictable workloads.

However, the main limitation of heuristic approaches lies in their tendency to optimize single dimension while neglecting critical factors like network traffic. As a result, they often perform poorly in multi-objective optimization scenarios that require balancing energy consumption, latency, and SLA compliance. Their static nature also makes them not suitable for dynamic or unpredictable environments, where workload patterns change rapidly and real-time re-optimization is essential. While their computational efficiency remains a major strength, this speed frequently comes at the cost of placement accuracy compared to more adaptive metaheuristic or learning-based methods.

In terms of scalability, heuristics perform well, handling high volumes of VM requests. They are ideal for environments where quick and frequent placement decisions are needed without deep optimization logic. However, their suitability for energy- and network-aware placement remains limited. Although variants like HBF reduce host-level energy consumption, they do not model dynamic power states or account for network bandwidth costs, resulting in potentially inefficient traffic patterns. Overall, heuristics are best reserved for static or predictable workloads —such as batch processing— or for initial placement stages before applying more adaptive optimization techniques. They are less appropriate for network-intensive applications, dynamic edge environments, or scenarios demanding multi-objective trade-offs.

C- Metaheuristics

Metaheuristic approaches, such as Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Genetic Algorithms (GA), Grey Wolf Optimization (GWO), and their hybrids; tackle VM placement as a multi-objective optimization problem, balancing energy consumption, resource utilization, and SLA guarantees.

For example, [36] propose a hybrid ACO–GWO that weaves in traffic-awareness to co-locate high-communication VMs, yielding up to 19.41% power savings and 10.72% bandwidth-utilization improvements over baseline algorithms.

[37] classify and critique a broad spectrum of nature-inspired metaheuristics−SA, PSO, GA, ACO, BBO, and hybrids—highlighting their strengths in exploration/exploitation balance but noting their general omission of communication costs.

The work [38] presents a hybrid GA–best-fit scheme that minimizes active PMs and resource wastage, characterizing VMs by CPU, RAM, and bandwidth.

Recently, the work [39] proposed the NCRA-DP-ACO algorithm, a network-, cost-, and renewable-aware ACO framework for energy-efficient VM placement across geographically distributed datacenters. Unlike previous metaheuristic solutions, this work introduces a dynamic Power Usage Effectiveness (PUE) model, real-time solar energy profiling, and carbon-aware cost modeling. By integrating environmental and economic factors into the multi-objective placement strategy, the algorithm achieved up to 18% energy savings and a 48% reduction in live migrations compared to baseline heuristics and metaheuristics. This approach demonstrates that incorporating sustainability-aware factors can significantly enhance placement decisions in large-scale cloud environments, addressing a critical gap often neglected in earlier VM placement studies.

Metaheuristics offer excellent pathways to near-optimal placement of VMs in multi-objective environment. They are capable of compromising among energy efficiency, SLA, and resource consolidation while covering a large solution space.

However, their performance heavily depends on proper parameter tuning, and poor configurations lead to suboptimal convergence. Moreover, most metaheuristics neglect traffic patterns or topology, and therefore require additional improvements for traffic- and communication-aware optimizations. Enhanced variants can improve network efficiency but require additional computational overhead.

Since these algorithms are iterative and population-based searches over multiple generations (denoted as t), they exhibit higher O complexity —$O(n^2 \times t)$, where n is the problem size and t is the number of iterations. This reflects a quadratic growth in computational cost with problem size, meaning convergence time increases significantly as the number of VMs scales. Nevertheless, these approaches remain effective for medium to large problem sizes.

These approaches are best suited for offline or semi-dynamic VM placement scenarios where computation time is not a concern. They excel in multi-objective optimization —balancing energy efficiency, performance, and cost—and are effective in sustainable cloud environments that require periodic reallocation. However, they are less ideal for low-

latency edge computing due to slower convergence rates, and they tend to underperform in highly dynamic or unpredictable workloads where rapid re-optimization is essential. For small-scale deployments, simpler heuristic methods are often more practical.

D- Machine Learning

Emerging AI-driven VM placement frameworks leverage predictive and adaptive techniques to anticipate demand, group workloads, and continuously learn optimal allocations. Workload Forecasting Models employ learning-based algorithms to predict future load patterns and proactively select hosts that balance energy consumption and SLA adherence.

Classification & Clustering approaches identify high-traffic VM pairs or hosts at risk of overload and refine placement heuristics; Finally, Reinforcement Learning optimizes VM placement by learning from interactions with the environment (servers, network, and workloads).

Workload Forecasting Models: The work [40] introduces a dynamic, learning-based scheme that continuously predicts per-VM resource-usage thresholds to drive proactive allocation and live migration decisions. The approach adapts to fluctuating loads by generating runtime data and training a hybrid model (combining swarm-inspired search with an ML classifier), thus improving SLA compliance, reducing migrations, and cutting energy compared to standalone bio-inspired or ML methods.

Classification & Clustering: Random Forests or K-means identify which VM pairs generate the most traffic, or which hosts are likely to become overloaded, refining heuristic weightings. LECC [41] — a multi-objective VM (and data) placement framework for geo-distributed clouds that jointly minimizes carbon emission cost, energy consumption, and WAN communication cost— embeds an intelligent ML module that is trained on historical energy, latency, and carbon-cost data to dynamically adjust its multi-objective weightings (carbon emission, energy, WAN cost) at runtime. Extensive simulations on synthetic and real (PlanetLab and EC2) traces demonstrate LECC's ability to reduce server energy and cut response latency compared to baseline methods.

Reinforcement Learning (RL): The work [42] proposes a fuzzy-based State-Action-Reward-State-Action (SARSA) reinforcement learning algorithm for optimal VM placement in CDCs, effectively reallocating VMs to minimize energy consumption and resource wastage while ensuring compliance with SLA and QoS demands during fluctuating workloads.

ML-based VM placement algorithms adapt better than static heuristics under workload variation and fast-changing user demands.

Yet, there do exist serious disadvantages. These algorithms need huge amounts of training data of almost perfect quality, and their predictive power degrades if they are not promptly retrained or adapted. Many approaches in ML tend to disregard network traffic behavior or the underlying topology, limiting their applicability in optimizing network energy consumption or communication latency. These models add a further computational overhead and convergence delays: For instance, clustering methods scale at $O(n^3)$, while deep-learning techniques demand tremendous GPU/CPU resources [43].

Lastly, scalability becomes an issue: whereas the bigger data can continue to scale the ML model, on the other side, training and inference times increase with the size of the problem. Some solutions —distributed or federated learning— can help but introduce synchronization and convergence delays.

Network- and energy-aware suitability, and also optimization, are still primary concerns of most of these ML-based solutions. Advanced architectures like GNNs can integrate network topology into their learning workflow, but these models are computationally costly and thus seldom used. Without explicitly modeling bandwidth consumption or link-layer power states, ML-based placements may underperform when communication and geo-distribution dominate the environment [44].

ML-based VM placement algorithms are more suited to dynamic and large-scale cloud environments with regular patterns of workload and good availability of historical data [45]. However, their applicability is limited in real time or latency-sensitive deployments, where response has to be immediate. They also fail in environments where the workloads are unpredictable or rapidly changing.

E- Graph Approaches

Graph-theoretic VM placement models represent PMs/ VMs as graph nodes, with edges encoding constraints like inter-VM traffic or power costs. By applying community-detection or graph-partitioning algorithms, they co-locate highly communicative VMs −minimizing network hops and energy consumption.

The algorithm in [46] uses a graph-coloring algorithm that models VMs as graph vertices and inter-VM traffic volumes as weighted edges, then iteratively "colors" (assigns) and merges vertices to minimize both network overhead and server power use. Their method batches VM migrations to keep high-traffic groups co-located and decommission underutilized hosts. Extensive simulations across hierarchical datacenter topologies demonstrate that GCA halves link saturation and outperforms single-migration schemes by up to 65% in network-overhead reduction.

Authors in [47] propose a two-phase, graph-theoretic VM placement strategy tailored for data-intensive cloud applications. They first model the datacenter as a complete weighted graph —vertices are hosts, edges carry a networking-cost metric combining link saturation and hop count. In Phase 1, a fuzzy inference system ranks racks by

free resources and intra-rack traffic, and a linear program selects the smallest set of "close" racks with low uplink load. In Phase 2, the Traffic-Distance-Balanced (TDB) greedy algorithm uses the graph's weighted adjacency matrix to iteratively pick hosts minimizing total inter-host networking cost. This approach unifies capacity and communication in a single graph framework, ensuring high host utilization while keeping over 80% of traffic rack-local and halving link saturation compared to flat heuristics.

Despite clear advantages in topology-aware grouping, graph methods incur $O(n^3)$ complexity and often require full-network snapshots, impractical for frequent re-optimizations.

Despite their strength in encoding traffic and topology awareness, these methods come with high computational costs. Algorithms for community detection, graph partitioning, and coloring frequently exhibit $O(n^3)$ complexity, which becomes a bottleneck in large or fast-evolving systems [46].

Another limitation lies in their reliance on static or snapshot-based views of the network state. To remain effective, graph-based models require up-to-date global topology and traffic matrices —information that is difficult to capture or maintain in real time without imposing significant monitoring and re-computation overhead. Additionally, integrating these specialized algorithms into existing cloud controllers or schedulers remains a challenge due to their architectural differences.

From an energy and network efficiency perspective, graph-theoretic strategies outperform heuristic or ML-based approaches in minimizing communication overhead and active link utilization. However, this often comes at the expense of higher host-level energy consumption when traffic-based clustering leads to VM consolidation on less energy-efficient machines. While the network energy savings are clear, careful balance is required to avoid increasing overall compute energy due to suboptimal host selection. These algorithms are suitable for communication-intensive workloads with predictable traffic patterns (e.g., Hadoop), and hierarchical (or structured) data centers where intra-rack traffic locality is critical. However they perform poor with: real-time architectures with rapidly shifting traffic flows, edge and fog computing scenarios with strict latency constraints, and hyperscale public clouds (>10,000 VMs) where $O(n^3)$ complexity is unjustified [48].

Summary and Comparative Insights

While each VM placement strategy category—mathematical optimization, heuristics, metaheuristics, machine learning, and graph theory—has distinct merits, they also exhibit significant trade-offs in terms of computational complexity, scalability, and suitability for energy- and network-aware objectives. Mathematical optimization-based methods provide provable optimality for small-scale problems but are intractable for real-time or large deployments. Heuristic methods are fast and scalable but fail to consider complex objectives or traffic metrics. Metaheuristics deliver near-optimal results and support multi-objective optimization, yet often suffer from parameter sensitivity and long runtimes. ML approaches bring adaptability and prediction to dynamic environments but are data-hungry and rarely embed network topology or energy metrics explicitly. Graph-theoretic models excel at topology-aware co-location but incur high computational costs and require complete snapshot data. As summarized in Table 5, selecting an appropriate placement strategy requires balancing complexity, performance goals, and environmental context, especially when aiming to reduce both host and network energy consumption.

and uniform PUE values, supporting static or light dynamic heuristics [49]. However, placement strategies risk creating network congestion at the rack level if VM affinities are misestimated and lack resilience against localized failures or flash crowd events. Centralized placements suit applications with consistent workload distributions but should be augmented with fault-tolerance and burst-handling extensions for production deployments.

3-3- Infrastructure Considerations (where to place?)

Cloud architecture plays a pivotal role in VM placement decisions. It encompasses the set of interconnected components and deployment models that define how compute, storage, and network services are delivered. A network-aware placement algorithm must adapt to the physical and logical characteristics of the underlying architecture.

A- Cloud Infrastructure type

Centralized Cloud: infrastructure consolidates all resources in a single data center, offering uniform latency and centralized cooling, power, and network control. Here, placement strategies emphasize intra-rack traffic minimization, server consolidation, and ALR to reduce switch and server energy. Because of the homogeneous environment, algorithms benefit from predictable latencies

Distributed Cloud: infrastructures span multiple, geographically dispersed sites or edge facilities. Placement algorithms in this context must account for WAN latency, variable carbon intensity, renewable energy availability, and differing PUE scores across locations. For instance, placement might favor a solar-powered region despite slightly higher latency. Network-aware algorithms in distributed contexts must balance performance against operational costs and inter-site bandwidth constraints [27]. While distributed placement can optimize global cost and sustainability, it introduces complexity in synchronizing

state across sites, handling network failures, and meeting latency-sensitive SLAs.

B- Cloud Proximity Models

Cloud Proximity Models distinguish between edge and core clouds based on their user-nearness and resource richness.

Edge Clouds: Deployed close to users for latency-sensitive workloads like gaming or AR/VR; placement here must prioritize minimal hop counts and rapid elasticity but suffers from limited capacity and heterogeneous infrastructure. TRACTOR [50], Traffic-aware and Power-efficient Placement in Edge-Cloud Data Centers (ECDCs), an Artificial Bee Colony-based multi-objective VM placement scheme that minimizes network traffic and power consumption in ECDCs. Evaluations on VL2 and three-tier topologies demonstrate a 3.5% reduction in server energy and up to 30% cut in network power usage without degrading QoS. However, TRACTOR presumes accurate pre- and post-placement traffic matrices and requires simulation-based calibration, limiting its adaptability to heterogeneous, real-world edge deployments.

Core Clouds: located in centralized, resource-rich facilities, are suited for compute-heavy, batch-oriented tasks that do not have stringent latency demands. Placement algorithms in these environments optimize resource density and power utilization while managing rack-level heat and congestion. In a centralized high-density core clouds, [51] framework employs a Greedy Randomized VMP (GRVMP) algorithm that fuses heuristic sorting with stochastic perturbations to escape local optima, achieving up to 12% energy reduction and 8% resource utilization gains compared to deterministic baselines. GRVMP addresses dynamic VM arrivals; however, its randomized nature can lead to variability in outcomes and overlooks network topology unless network-aware metrics are integrated.

C- Hardware-Based Energy Mechanisms

Datacenter hardware often embeds energy-saving features at component and network levels. Placement algorithms that are aware of these mechanisms can reduce overall power draw by tailoring VM assignments to exploit them.
We categorize three primary hardware-based strategies below:

- ALR:

ALR dynamically scales the data-link speed of network interfaces (e.g., from 1 Gbps to 100 Mbps) based on instantaneous utilization. When traffic is low, links down-shift to a lower rate—saving up to 40 % of PHY-layer power—then ramp up again under load. Some VM placement schemes explicitly cluster bursty or low-throughput VMs under the same Top-of-Rack switch to maximize low-speed intervals and link-power savings [52].

- DVFS:

Modern CPUs and NICs support DVFS, which lowers voltage and clock frequency when workload demands permit. Experimental studies report up to 30 % server-level energy reduction with minimal performance loss under controlled load variations [53]. Energy-aware schedulers simulate or predict CPU utilization to trigger DVFS states—placing latency-insensitive VMs on hosts where cores can be down-clocked, while reserving full-speed nodes for critical workloads [54].

- Switch and Rack Power-Down:

Many top-of-rack (ToR) switches and rack PDUs can enter sleep modes or shut down unused ports when idle. Research prototypes have shown up to 50 % energy savings in underutilized racks by consolidating traffic and powering down dormant switches [55]. Topology-aware schemes fold traffic into active racks during off-peak periods, allowing idle switches or PDUs to sleep or power off; the migration cost is balanced against the long-term energy gains [56].

Placement algorithms treat ALR, DVFS, and switch/rack power-down not as standalone placement steps but as hardware-aware objectives or constraints that guide where and when to place or migrate VMs. In other words, these features aren't separate "phases" of VM placement; rather, placement algorithms incorporate knowledge of link-rate scaling, voltage/frequency capabilities, or switch on/off thresholds to shape consolidation decisions.
Integrating these hardware-based mechanisms into placement and migration heuristics unlocks significant energy savings that complement software techniques.

D- Thermal-Aware Placement Strategies

Integrating thermal dynamics into VM placement helps prevent hotspots and reduces cooling energy consumption by considering rack- and node-level temperature distributions during allocation and migration decisions [57]. Multi-objective formulations jointly optimize computing energy and cooling load, enabling VM placement algorithms to trade off consolidation benefits against the risk of creating thermal hotspots [58].

3-4- Workload Characteristics (What Is Being Placed)

A- Arrival rate

Static: Static workloads such as batch jobs in scientific computing, benefit from heavy-weight optimizations like ILP, yielding near-optimal resource packing when demands are known in advance [59][60]. The term "static allocation" usually refers to the initial VM placement which is the allocation of VMs to PMs is done during deployment and remains fixed throughout the VMs' lifecycle. The goal is to optimize allocation based on resource requirements and

constraints. However, the assumption of stable load profiles renders it brittle when workloads fluctuate unpredictably.

Dynamic: Dynamic scenarios characterized by real-time VM arrivals in auto-scaling web services or event-driven microservices. Dynamic VM placement includes placing new VMs and migrating existing ones, considering future live migrations, and needs more resources than static solutions.

In this context, reactive placement adapts the initial allocation of resources based on the current state of the system, while proactive placement predicts future conditions and adjusts allocations before problems occur.

- Reactive Placement: Migration or reallocation is triggered by observed thresholds, such as CPU/memory utilization exceeding a limit, network congestion detected on a link, or thermal hot spots. Reactive methods respond to current system state ([61][62]) but often react too late to avoid SLA violations or suboptimal energy states.

- Proactive Placement: Predictive models anticipate future workloads or traffic spikes and migrate VMs preemptively. While more complex, requiring accurate demand prediction, proactive approaches can better prevent overloads and exploit low-utilization windows for consolidation [20], [21]).

B- Workload Type (Application-Centric)

We present the main application categories in the literature used to guide placement heuristics.

Bag of Tasks: Independent parallel tasks requiring minimal inter-communication. Placement focuses on maximizing throughput and minimizing makespan by grouping tasks (VMs) on minimal PMs [41].

CPU-Intensive Workloads: Require sustained processor capacity and thermal stability. Placement must dedicate cores to each VM and move workloads off busy hosts to prevent contention and overheating [64].

Data-Intensive Workloads: Require high I/O and low-latency access to shared storage. Placement must reduce traffic to storage nodes (SNs) and minimize bottlenecks [65].

Latency-Sensitive Applications: Include gaming, financial systems, or telemedicine, where delays severely degrade user experience. These demand edge-aware, low-hop-count VM placement [66].

C- Workload Data Sources for Algorithm Evaluation

The following are the ways researchers evaluate their work against other algorithms. However, researchers may combine two or more types of workload data.

- Benchmark Datasets*:* Standardized collections of VM workload traces detailing CPU, memory, I/O and network usage, collected via monitoring tools, application profilers or user logs. They enable controlled, repeatable comparisons of placement algorithms by quantifying impacts on network utilization, availability and cost.

- Synthetic data: Synthetic data is generated using mathematical models and statistical techniques that simulate the behavior of real-world applications and infrastructure components. It allows researchers to control the workload and resource utilization characteristics of the cloud infrastructure and to compare different algorithms under the same conditions. Researchers evaluated their work using synthetic scenarios with several performance metrics [67].

- Real Traces: Real traces are collected from real cloud computing environments (Amazon EC2, PlanetLab, and Google Cluster) to evaluate VM placement algorithms under realistic conditions. In [51], Amazon EC2 data was used to optimize power consumption. In [68], PlanetLab network traces were utilized to assess algorithm performance. Both methods provide insights into workload behaviors and resource utilization for algorithm evaluation.

These classifications create a multidimensional lens to evaluate VM placement strategies and pave the way for our specialized network-aware taxonomy in Section 4.

4- Taxonomy of Network-Aware VM Placement Approaches

This section synthesizes the contextual shifts and motivates the need for a new taxonomy—one that maps VM placement methods not only to their algorithmic families (heuristic, ML-based) but also to the underlying network dynamics they aim to optimize. As shown in Fig.2, our taxonomy therefore introduces a cross-layer perspective that bridges DCN topology, traffic characteristics, communication patterns, and energy reduction strategies, reflecting how emerging solutions should be evaluated in modern cloud environments. Additionally, a sub-taxonomy at the bottom of Fig.2 classifies network-aware VMP algorithms according to their energy consumption strategies.

In a typical cloud computing environment, VMs are interconnected with physical hosts through a network, generating substantial network traffic from the applications they run. Consequently, the placement of VMs on physical hosts significantly impacts network performance, which in turn affects overall application performance. Given that the network is a major consumer of energy, minimizing network traffic and optimizing topology can lead to substantial energy savings.

Therefore, it is critical to consider network-related factors throughout placing and migrating VMs. This means that the VMP algorithm should not only consider the usual metrics

and resource requirements of VMs and PMs but also incorporate network considerations. The algorithm can make more informed decisions regarding VM placement and consolidation by incorporating network conditions, topology, and traffic patterns.

Customers utilize VMs to conduct specific jobs that are frequently parts of larger applications, such as tiers of multi-tier applications. As these VMs start communicating with each other, it can involve the transfer of significant amounts of data, which might increase latency or response times to

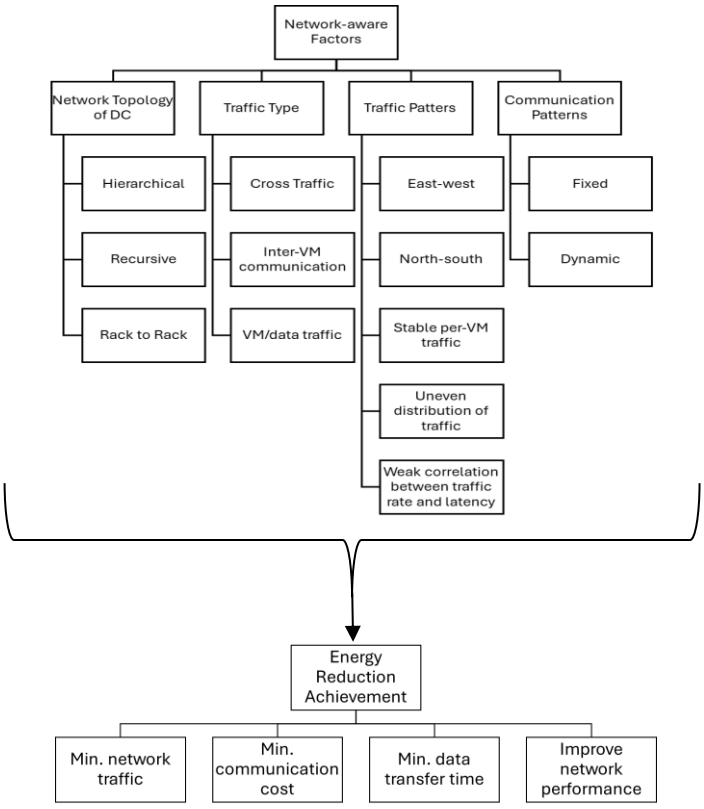

Fig.2 Network-aware VM placement taxonomy.

intolerable levels. In addition, the power consumption of the hardware components involved, such as PMs, routers, switches, and other networking equipment, can also be affected by such communication patterns.

For the reasons listed above, it is ideal to have VMs that communicate frequently placed on the same server, or at the very least within the same DC. Additionally, VMs belonging to the same application may have load correlation, making it more likely that they may peak at the same time; this must also be carefully considered when allocating VM resources.

Network bandwidth can often become a bottleneck, particularly in scenarios involving data mapping on SNs. High network traffic between VMs and SNs can arise when workloads require extensive data mapping. To prevent too many high network loads, it is necessary to consider both the placement of VMs on PMs and application data on SNs. To facilitate this, we categorize network-aware VMP algorithms into four groups based on their focus on network considerations:

4-1- DCN Topology

DC topology involves organizing physical and logical components in a network, including servers, network devices, and SNs. It enables efficient connections with multiple PMs, enhancing energy efficiency and reducing reliability concerns. Various network topologies tackle scalability and energy consumption differently and offer insights for future VM placement research. Researchers can examine the advantages, drawbacks, and enhancements of these topologies to improve current VM placement methods, as discussed in Section V.

A- Hierarchical Three-Tier

This architecture manages traffic using a structured approach. The access layer connects servers to edge switches, which then relay information to interconnected aggregate switches. The core layer serves as the spine, linking all aggregate switches and handling external connections, providing a scalable and efficient solution for internal data center communication.

- Fat-tree: A three-tier architecture utilizing bipartite graphs with pods as the basic unit, where each pod contains access and aggregation switches. This topology offers efficient routing paths for reducing congestion and power consumption [69].

- VL2: Like fat-tree, this three-tier topology connects core and aggregation switches in a bipartite graph. Valiant load balancing routes traffic by randomly selecting a core switch, reducing congestion and power consumption. A customized VMP technique further optimizes network traffic. [67].

- Portland: This architecture comprises pods with access and aggregation switches forming bipartite graphs, connecting to all core switches. VM placement algorithms prioritize proximity to enhance quality of service (QoS)[70].

B- Recursive

These topologies are constructed recursively, combining smaller building blocks into larger network structures, allowing for scalable and modular designs.

- DCell**:** a server-centric data center network design with a hierarchical structure. Servers connect directly with multiple NICs, organized into cells like cell0, cell1, and cell2 [71].

- BCube**:** BCube is a multi-level data center network architecture focused on servers, integrating them into the network infrastructure. It is derived from hypercube architecture, connecting hosts via switches based on port availability for efficient packet forwarding [72].

C- Rack to Rack

Rack-to-rack networks prioritize communication between server racks. Their design focuses on efficient data transfer within and across racks.

- Scafida**:** a method inspired by scale-free networks to create asymmetric data center topologies with high fault tolerance and small diameters. It allows for flexible scaling but faces challenges with link correlation as the network grows [73].
- Jellyfish: Jellyfish network with random graph topology offers cost-efficiency, 25% more server support, scalability, and flexibility for high-capacity interconnectivity [74].

4-2- Traffic Type

Traffic type categorization in cloud DCs (considered in VMP) optimizes network performance and energy usage by placing VMs with similar traffic types together, reducing data transfers across the network and minimizing energy consumption.

A- Cross-Traffic

Cross-traffic is the data flow between VMs or applications that may be located on different servers within the same rack or across different racks. This type of traffic can impact network performance and energy usage. Allocating VMs and data on physically closer PMs can improve efficiency, as explored in [75].

B- Inter-VM Communication

North-south traffic involves data flow between virtual machines (VMs) and the Internet, while inter-VM communication refers to data exchange within the same data center. The latter is often high-bandwidth and low-latency, with different application requirements.

Studies are focusing on reducing network energy usage by optimizing VM placement to minimize inter-rack traffic and reduce delays, consequently cutting down on power consumption and costs [76].

C- Traffic between VM and Data

This traffic occurs when VMs access data stored on storage devices. VMs send requests to these devices via the network, and the data is transmitted back to the requesting VM. Factors influencing traffic volume include data size, access frequency, and the type of storage device used. In distributed object storage systems, each storage node manages a group of servers. When a server and its corresponding storage node are within the same group, data transfer is optimized, thereby reducing overall traffic flow [77].

4-3- Traffic Patterns

Understanding traffic patterns in cloud networks is crucial for optimizing performance by placing virtual machines in strategic locations to improve network performance and reduce energy consumption. Research indicates that network status changes over time due to unpredictable traffic characteristics, regardless of data center size or type. Authors advocate for traffic-aware VM placement to enhance network scalability by aligning traffic patterns with communication distances. Empirical studies reveal imbalanced communication patterns, link losses, and ON-OFF traffic patterns with varying distributions, emphasizing the need for optimized VM allocation and routing in cloud networks [3] [78].

4-4- Communication Patterns

Communication patterns in VM placement refer to how VMs interact with each other and with external networks. It is a useful resource for perceiving the parallel application communication behavior and is extracted from communication trace, where machines form multiple groups or tiers each of which serves a specific part needed for the accomplishment of the overall task. Energy consumption heavily depends on the communication pattern [79].

A- Fixed

Fixed communication patterns between virtual machines (VMs) exhibit predictable and consistent interactions that remain unchanged during runtime. VM placement strategies often aim to co-locate VMs with frequent communication to minimize network latency and overhead [76].

B- Dynamic

Dynamic communication patterns between VMs change during runtime, in contrast to fixed patterns. This requires adaptable VM placement solutions that monitor and adjust VM locations based on evolving communication needs. The technique introduced in [80] uses a decentralized migration approach considering VM affinity. It dynamically adjusts VM placement through a distributed bartering algorithm to minimize communication overhead and adapt to changing patterns, while maintaining low overhead.

4-5- Energy Reduction Achievement

The energy reduction classification in our taxonomy in Fig.2 is centered around strategies and methodologies in reducing energy consumption in network-aware VM placement. This section highlights how researchers have leveraged network awareness to achieve considerable energy savings in CDCs. In this section, we review different approaches for network traffic minimization, communication cost minimization, data transfer time reduction, and network performance improvement.

A- Minimizing Network Traffic

One of the effective strategies is to optimize VM placement with the co-location of VMs that communicate with each other with high volume on the same physical hosts. In this way, the distance that data needs to travel is minimal and reduces traffic in the network. For example, the work in [50] suggested a multi-objective VM placement algorithm using a bee colony method, achieving 3.5% power reduction, 15% less network traffic, and 30% lower network power. Similarly, the work in [22] proposed an ant colony optimization algorithm considering both energy usage and network bandwidth, which effectively reduced traffic and outperformed other heuristics.

B- Minimize Communication Cost

Network communication costs refer to expenses in terms of bandwidth utilization, latency, and rate of data transfer. For VM placement, reducing such costs minimizes resource consumption and overall expenses. The work in [59] introduced a "network consumption" metric to identify optimal VM placements within a fat-tree architecture to minimize network traffic. This approach led to a significant reduction in overall network usage and power consumption, decreasing resource wastage by up to 20%. Similarly, the approach in [81] focused on enhancing VM-to-VM communication using dynamic clustering of VMs based on the network. An adaptive algorithm consolidated VMs to minimize communication costs, leading to reduced high-latency jobs and improved traffic patterns across the network. The goal of these techniques is to strategically place and manage VMs to lower the overall communication costs in the data center network [36].

C- Minimizing Data Transfer Time

Data transfer time is the duration for data to be transmitted between VMs over the network. It affects energy usage and application performance. Placing VMs closer and grouping them based on traffic patterns can minimize data transfer time. [82] proposed a novel VMP technique that simultaneously improves both VM locations and data rates. They developed heuristics that allocate VMs to PMs with better network bandwidth to reduce the latencies associated with data access. Through simulation experiments, they demonstrated how the proposed approach may lower VMs' data transmission delays.

D- Improving Network Performance

Improving network performance is the act of optimizing a computer network to enhance its speed, reliability, and efficiency. This involves improving the various components of the network, including switches, routers, cables, servers, and applications, to ensure that data is transmitted quickly, accurately, and consistently. The previously mentioned work in [59] was categorized under minimizing communication cost, but it focused also on minimizing resource wastage, which led to the optimization of the overall network performance.

E- Emerging Trends

With the rise of such technologies as network virtualization and Software-Defined Networking (SDN), the way VM placement for energy efficiency will be significantly impacted. Network virtualization increases the flexibility of network resource allocation and management, such that even real-time adjustments according to changing traffic patterns become possible. On the other hand, SDN brings central control to a network, which makes routing much more efficient and leads to lower energy consumption. These technologies are still evolving, we can expect further improvements in energy efficiency and overall network performance in the placement of VMs [83].

5- Discussion

This section discusses the important relationship between network topology, traffic patterns, and energy efficiency in network-aware VMP. We provide a novel perspective on how these aspects interact and affect the total energy consumption within the datacenter.

5-1- Traffic Type

Different traffic types have varying requirements regarding reliability, latency, and network bandwidth. For example, real-time communication applications, including video conferencing and VoIP, require low latency and high reliability; in contrast, batch processing applications such as data analytics can tolerate high latency and low reliability. Those network traffic patterns found in datacenters can significantly affect energy consumption, SLAs, cloud provider revenue, as well as the overall cloud infrastructure's efficiency.

In response to such challenges, there has been a development of network-aware VM placement algorithms to optimize network traffic and minimize resource utilization in CDCs. These algorithms distribute the network traffic evenly across the infrastructure to prevent congestion, resulting in energy savings. VMs often rely on the network for data-intensive applications and interactions with other VMs. These algorithms can prioritize high-bandwidth VMs and place them nearby by optimizing the placement of VMs based on their communication patterns, reducing the overall network traffic between and within the data centers. This, in turn, minimizes the number of physical networking components required and leads to reduced power consumption.

5-2- Network Topology

Network topology is a principal issue in virtual machine placement, which affects resource utilization and energy efficiency. Placing VMs wisely reduces the distance of data transfers, switches, and links involved in communication and leads to saving energy as well as increasing performance. Fat-tree topology manages the high-bandwidth, low-latency traffic well within a pod or data center, while VL2 is good for traffic generated by VMs in cloud environments, including storage, migration, and inter-DC. BCube is suitable for data-intensive applications that demand high bandwidth and efficient data transmission.

In this subsection, network topology influence on VM placement is discussed based on existing research that examines the impact on energy efficiency as well as overall system performance [84]. The placement of VMs close to each other is quite essential for resource utilization and energy efficiency. Strategic placement reduces the distance of data transfer, therefore reducing the number of switches and links, which means less energy consumption and improved performance in data centers. The three-tier architecture typically includes expensive and power-intensive network devices at the corporate level, whereas DCell and BCube architecture consume similar energy for small-sized data centers. However, BCube consumes more energy for larger data centers. The Fat-Tree topology has reasonable power usage, while BCube is power-intensive due to its extensive use of switches. DCell utilizes commodity switches that consume less power. BCube's design with intermediate servers for routing can pose challenges to energy efficiency.

According to experimental findings, the tree topology experiences congestion issues with similar VM traffic, while the Fat-Tree topology distributes traffic more evenly due to its multi-path connections. VL2 suffers from uneven traffic distribution due to a large gap in link utilization. The Tree topology has lower energy efficiency compared to VL2 and Fat-Tree, although topology awareness can optimize energy usage in the network. However, these conclusions are specific to each author's work, and more research is needed to establish correlations between data center size, server count, switches, and user demands. Cloud service providers should ensure appropriately sized environments to minimize costs. A hybrid or dynamic topology approach using SDN can optimize resource utilization, energy efficiency, and overall performance by adapting the network topology based on workload demands, such as favoring a fat-tree topology for high east-west traffic.

5-3- Traffic and Communication Patterns

To minimize energy consumption in DCs, network-aware VM placement algorithms play a crucial role. These algorithms aim to allocate VMs with similar traffic patterns to the same physical servers or switches. This will reduce inter-server or inter-switch communication, therefore saving energy not only in the network infrastructure but also in the servers. Secondly, VMP optimization based on bandwidth and latency demands will prevent network congestion, thus assuring satisfactory performance and energy efficiency during communications.

Energy consumption and network traffic in virtualized environments were analyzed in studies [58,59]. It was noticed that energy consumption might have a wide variation for different traffic allocation strategies and that the type of traffic may strongly influence the possible energy savings. Such results are important to consider in traffic-aware optimizations, but all such optimizations require detailed information from clients about the application network and communication requirements. This allows network-aware techniques for minimizing communication delays and/or improving overall application performance.

The distribution of the components over various PMs provides a good opportunity for parallel processing in applications such as MapReduce. In case migration needs to be done, the ideal order of the intercommunicating virtual machines will help avoid core network traffic and energy consumption. Considering intercommunication between replicated virtual machines is also important to prevent bottlenecks and excessive energy usage.

Recognition of the traffic pattern is especially important in dynamic cloud environments. Workload and communication requirements are dynamic; hence, the adaptability of VMP algorithms is required to achieve resource and energy efficiency. Such dynamical traffic management approaches like load balancing and traffic shaping would prevent congestion and optimize power consumption.

The application-specific information will also reduce latency, inter-VM traffic, and improve application performance in placement algorithms. On the other hand, machine learning algorithms will use historical traffic data and predictive models to foresee traffic patterns, thus making proactive placement decisions that reduce energy consumption. Machine learning can also help in identifying and classifying traffic hotspots, which helps in applying targeted optimizations to mitigate power imbalances.

6- Conclusion And Future Directions

This paper presents a new classification for VM placement techniques in CDCs that are both network-aware and energy-efficient. It examines various network factors, including network equipment, workload type, performance, scalability, efficiency, reliability, and availability, to understand how VM placement affects network performance. The research indicates that network-aware VM placement algorithms can boost performance by reducing latency between VMs and improving security through co-location. However, the initial deployment of these algorithms might incur higher costs, necessitating a careful evaluation of the trade-off between energy consumption and migration costs.

This work also reviews research that identifies the most effective metrics for evaluating the performance of network-aware VM placement algorithms, focusing on energy efficiency, network performance, and resource utilization. Additionally, the study examines how network topology affects energy consumption in data centers and the trade-off between energy use and migration costs, providing valuable insights. These insights can help researchers develop and implement more effective network-aware VM placement algorithms that optimize energy consumption, improve network performance, and minimize migration costs. Based on the findings, future research directions for network-aware VM placement in CDCs can be suggested, including:

- Developing energy-efficient algorithms that consider the network metrics identified in this study. This would involve creating strategies to optimize energy use while improving network performance, factoring in elements like datacenter layout and communication patterns.

- Testing VM placement techniques on realistic testbeds. While simulations help assess the proposed VM placement methods, it is essential to validate these techniques on actual cloud testbeds with real-world network topologies.

- Researching VM placement algorithms that enhance security and privacy in cloud environments. This could involve devising methods to group related VMs on the same server or rack while preventing the co-location of unrelated VMs. Such strategies would help mitigate the risk of security breaches and protect sensitive data in cloud settings.

- Continuing to explore novel solutions for optimizing VM placement and migration that can boost energy efficiency and network performance in CDCs. This would include investigating innovative techniques and approaches that leverage emerging technologies like machine learning and artificial intelligence to improve network-aware VM placement.

Future research in this area could investigate how elements like energy storage systems, renewable energy sources, and workload balancing impact network-aware VM placement. These potential directions provide a solid foundation for further exploration of energy-efficient network-aware VM placement, intending to create more effective strategies for optimizing energy consumption, improving network performance, enhancing security and privacy, and integrating artificial intelligence throughout the cloud computing environment.

References

[1] P. M. Mell and T. Grance, “The NIST definition of cloud computing,” Gaithersburg, MD, 2011. doi: 10.6028/NIST.SP.800-145.

[2] D. Bliedy, S. Mazen, and E. Ezzat, “Datacentre Total Cost of Ownership (TCO) Models : A Survey,” International Journal of Computer Science, Engineering and Applications, vol. 8, no. 2/3/4, pp. 47–62, 2018, doi: 10.5121/ijcsea.2018.8404.

[3] T. Benson, A. Akella, and D. A. Maltz, “Network traffic characteristics of data centers in the wild,” Proceedings of the ACM SIGCOMM Internet Measurement Conference, IMC, pp. 267–280, 2010, doi: 10.1145/1879141.1879175.

[4] L. Zhou, C. H. Chou, L. N. Bhuyan, K. K. Ramakrishnan, and D. Wong, “Joint server and network energy saving in data centers for latency-sensitive applications,” Proceedings - 2018 IEEE 32nd International Parallel and Distributed Processing Symposium, IPDPS 2018, pp. 700–709, 2018, doi: 10.1109/IPDPS.2018.00079.

[5] K. Bilal et al., “A survey on Green communications using Adaptive Link Rate,” Cluster Comput, vol. 16, no. 3, pp. 575–589, Jul. 2013, doi: 10.1007/s10586-012-0225-8.

[6] A. C. Orgerie, M. D. De Assuncao, and L. Lefevre, “A survey on techniques for improving the energy efficiency of large-scale distributed systems,” ACM Comput Surv, vol. 46, no. 4, 2014, doi: 10.1145/2532637.

[7] M. H. Ferdaus, M. Murshed, R. N. Calheiros, and R. Buyya, Network-aware virtual machine placement and migration in cloud data centers, no. May. 2015. doi: 10.4018/978-1-4666-8213-9.ch002.

[8] A. Hammadi and L. Mhamdi, “A survey on architectures and energy efficiency in Data Center Networks,” Comput Commun, vol. 40, pp. 1–21, 2014, doi: 10.1016/j.comcom.2013.11.005.

[9] K. Bilal et al., “A taxonomy and survey on Green Data Center Networks,” Future Generation Computer Systems, vol. 36, pp. 189–208, Jul. 2014, doi: 10.1016/j.future.2013.07.006.

[10] R. W. Ahmad, A. Gani, S. H. A. Hamid, M. Shiraz, A. Yousafzai, and F. Xia, “A survey on virtual machine migration and server consolidation frameworks for cloud data centers,” Journal of Network and Computer Applications, vol. 52, pp. 11–25, 2015, doi: 10.1016/j.jnca.2015.02.002.

[11] F. L. Pires and B. Baran, “A virtual machine placement taxonomy,” Proceedings - 2015 IEEE/ACM 15th International Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2015, no. July, pp. 159–168, 2015, doi: 10.1109/CCGrid.2015.15.

[12] M. Masdari, S. S. Nabavi, and V. Ahmadi, “An overview of virtual machine placement schemes in cloud computing,” Journal of Network and Computer Applications, vol. 66, pp. 106–127, 2016, doi: 10.1016/j.jnca.2016.01.011.

[13] H. Talebian et al., Optimizing virtual machine placement in IaaS data centers: taxonomy, review and open issues, vol. 23, no. 2. Springer US, 2020. doi: 10.1007/s10586-019-02954-w.

[14] H. Zhuang and B. Esmaeilpour Ghouchani, "Virtual machine placement mechanisms in the cloud environments: a systematic review," Kybernetes, vol. 50, no. 2, pp. 333–368, 2021, doi: 10.1108/K-09-2019-0635.

[15] L. Helali and M. N. Omri, "A survey of data center consolidation in cloud computing systems," 2021. doi: 10.1016/j.cosrev.2021.100366.

[16] A. Sumathi, … B. K.-T. J. of, and undefined 2023, "Advancements in Energy-Efficient Virtual Machine Placement Survey for Cloud Computing," Researchgate.Net, no. February, 2024, doi: 10.13140/RG.2.2.17918.36164.

[17] N. Rana et al., "A systematic literature review on contemporary and future trends in virtual machine scheduling techniques in cloud and multi-access computing," Front Comput Sci, vol. 6, 2024, doi: 10.3389/fcomp.2024.1288552.

[18] J. Zou, K. Wang, K. Zhang, and M. Kassim, "Perspective of virtual machine consolidation in cloud computing: a systematic survey," Telecommun Syst, p. 11235, 2024, doi: 10.1007/s11235-024-01184-9.

[19] S. R. Swain, A. Parashar, A. K. Singh, and C. Nan Lee, "An Energy Efficient Virtual Machine Placement Scheme for Intelligent Resource Management at Cloud Data Center," in OCIT 2023 - 21st International Conference on Information Technology, Proceedings, Institute of Electrical and Electronics Engineers Inc., 2023, pp. 65–70. doi: 10.1109/OCIT59427.2023.10430915.

[20] S. Kumar, S. Mittal, and M. Singh, "Active VM Placement Approach Based on Energy Efficiency in Cloud Environment," in Lecture Notes in Networks and Systems, Springer Science and Business Media Deutschland GmbH, 2022, pp. 35–46. doi: 10.1007/978-981-19-1018-0_4.

[21] Z. Li, K. Lin, S. Cheng, L. Yu, and J. Qian, "Energy-Efficient and Load-Aware VM Placement in Cloud Data Centers," J Grid Comput, vol. 20, no. 4, 2022, doi: 10.1007/s10723-022-09631-0.

[22] H. Xing, J. Zhu, R. Qu, P. Dai, S. Luo, and M. A. Iqbal, "An ACO for energy-efficient and traffic-aware virtual machine placement in cloud computing," Swarm Evol Comput, vol. 68, no. November 2021, p. 101012, 2022, doi: 10.1016/j.swevo.2021.101012.

[23] D. Dabhi and D. Thakor, "Utilisation-aware VM placement policy for workload consolidation in cloud data centres," International Journal of Communication Networks and Distributed Systems, vol. 28, no. 6, pp. 704–726, 2022, doi: 10.1504/ijcnds.2022.126224.

[24] E. I. Elsedimy, M. Herajy, and S. M. M. Abohashish, "Energy and QoS-aware virtual machine placement approach for IaaS cloud datacenter," 2025. doi: 10.1007/s00521-024-10872-1.

[25] K. Lu, R. Yahyapour, P. Wieder, C. Kotsokalis, E. Yaqub, and A. I. Jehangiri, "QoS-aware VM placement in multi-domain service level agreements scenarios," IEEE International Conference on Cloud Computing, CLOUD, no. April 2014, pp. 661–668, 2013, doi: 10.1109/CLOUD.2013.112.

[26] T. Renugadevi, K. Geetha, K. Muthukumar, and Z. W. Geem, "Optimized energy cost and carbon emission-aware virtual machine allocation in sustainable data centers," Sustainability (Switzerland), vol. 12, no. 16, pp. 1–27, 2020, doi: 10.3390/SU12166383.

[27] S. Rawas, A. Zekri, and A. El Zaart, "Power and Cost-Aware Virtual Machine Placement in Geo-Distributed Data Power and Cost-aware Virtual Machine Placement in Geo-distributed Data Centers," no. March, 2018, doi: 10.5220/0006696201120123.

[28] G. P. Maskare and S. Sharma, "The Hybrid ACO, PSO, and ABC Approach for Load Balancing in Cloud Computing," vol. 10, 2023, Accessed: May 07, 2025. [Online]. Available: www.jetir.org

[29] M. H. Kim, J. Y. Lee, S. A. Raza Shah, T. H. Kim, and S. Y. Noh, "Min-max exclusive virtual machine placement in cloud computing for scientific data environment," Journal of Cloud Computing, vol. 10, no. 1, pp. 1–17, Dec. 2021, doi: 10.1186/S13677-020-00221-7/FIGURES/12.

[30] M. Koubàa, R. Regaieg, A. S. Karar, M. Nadeem, and F. Bahloul, "A Multi-Objective Approach for Optimizing Virtual Machine Placement Using ILP and Tabu Search," Telecom, vol. 5, no. 4, pp. 1309–1331, 2024, doi: 10.3390/telecom5040065.

[31] X. Zheng and Y. Xia, "Exploring mixed integer programming reformulations for virtual machine placement with disk anti-colocation constraints," Performance Evaluation, vol. 135, 2019, doi: 10.1016/j.peva.2019.102035.

[32] S. Yang, P. Wieder, R. Yahyapour, S. Trajanovski, and X. Fu, "Reliable Virtual Machine Placement and Routing in Clouds," IEEE Transactions on Parallel and Distributed Systems, vol. 28, no. 10, pp. 2965–2978, 2017, doi: 10.1109/TPDS.2017.2693273.

[33] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing," Future Generation Computer Systems, vol. 28, no. 5, pp. 755–768, 2012, doi: 10.1016/j.future.2011.04.017.

[34] J. Wang, J. Yu, R. Zhai, X. He, and Y. Song, "GMPR: A Two-Phase Heuristic Algorithm for Virtual Machine Placement in Large-Scale Cloud Data Centers," IEEE Syst J, vol. 17, no. 1, pp. 1419–1430, Mar. 2023, doi: 10.1109/JSYST.2022.3187971.

[35] S. Jangiti, V. Vijayakumar, and V. Subramaniyaswamy, "Hybrid best-fit heuristic for energy efficient virtual machine placement in cloud data centers," EAI Endorsed Transactions on Energy Web, vol. 7, no. 26, pp. 1–7, 2020, doi: 10.4108/eai.13-7-2018.162689.

[36] R. Keshri and D. P. Vidyarthi, "Communication-aware, energy-efficient VM placement in cloud data center using ant colony optimization," International Journal of Information Technology (Singapore), vol. 15, no. 8, pp. 4529–4535, Dec. 2023, doi: 10.1007/S41870-023-01531-0/METRICS.

[37] N. Donyagard Vahed, M. Ghobaei-Arani, and A. Souri, "Multiobjective virtual machine placement mechanisms using nature-inspired metaheuristic algorithms in cloud environments: A comprehensive review," International Journal of Communication Systems, vol. 32, no. 14, 2019, doi: 10.1002/dac.4068.

[38] A. S. Abohamama and E. Hamouda, "A hybrid energy–Aware virtual machine placement algorithm for cloud environments," Expert Syst Appl, vol. 150, p. 113306, 2020, doi: 10.1016/j.eswa.2020.113306.

[39] A. M. Baydoun and A. S. Zekri, "Network-, Cost-, and Renewable-Aware Ant Colony Optimization for Energy-Efficient Virtual Machine Placement in Cloud Datacenters," Future Internet, vol. 17, no. 6, p. 261, Jun. 2025, doi: 10.3390/fi17060261.

[40] S. Talwani et al., "Machine-Learning-Based Approach for Virtual Machine Allocation and Migration," Electronics (Switzerland), vol. 11, no. 19, 2022, doi: 10.3390/electronics11193249.

[41] S. Rawas, A. Zekri, and A. El-Zaart, "LECC: Location, energy, carbon and cost-aware VM placement model in geo-distributed DCs," Sustainable Computing: Informatics and Systems, vol. 33, 2022, doi: 10.1016/j.suscom.2021.100649.

[42] A. Jumnal and S. M. Dilip Kumar, "Optimal VM placement approach using fuzzy reinforcement learning for cloud data centers," in Proceedings of the 3rd International Conference on Intelligent Communication Technologies and Virtual Mobile Networks, ICICV 2021, Institute of Electrical and Electronics Engineers Inc., Feb. 2021, pp. 29–35. doi: 10.1109/ICICV50876.2021.9388424.

[43] H. Padmanaban, "Machine Learning Algorithms Scaling on Large-Scale Data Infrastructure," Journal of Artificial Intelligence General science (JAIGS) ISSN:3006-4023, vol. 3, no. 1, pp. 1–26, Apr. 2024, doi: 10.60087/JAIGS.VOL03.ISSUE01.P26.

[44] H. A. Alharbi, T. E. H. Elgorashi, A. Q. Lawey, and J. M. H. Elmirghani, "The Impact of Inter-Virtual Machine Traffic on Energy Efficient Virtual Machines Placement," in 2019 IEEE Sustainability through ICT Summit, StICT 2019, 2019. doi: 10.1109/STICT.2019.8789381.

[45] F. kamoun-abid, H. Frikha, A. Meddeb-Makhoulf, and F. Zarai, "Allocation of virtual machine in a cloud environment based on machine learning," Res Sq, Jan. 2023, doi: 10.21203/RS.3.RS-2483861/V1.

[46] N. Tziritas, T. Loukopoulos, S. Khan, C. Z. Xu, and A. Zomaya, "A communication-aware energy-efficient graph-coloring algorithm for VM placement in clouds," Proceedings - 2018 IEEE SmartWorld, Ubiquitous Intelligence and Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People and Smart City Innovations, SmartWorld/UIC/ATC/ScalCom/CBDCo, pp. 1684–1691, 2018, doi: 10.1109/SmartWorld.2018.00286.

[47] S. Sadegh, K. Zamanifar, P. Kasprzak, and R. Yahyapour, "A two-phase virtual machine placement policy for data-intensive applications in cloud," Journal of Network and Computer Applications, vol. 180, no. December 2020, p. 103025, 2021, doi: 10.1016/j.jnca.2021.103025.

[48] J. Gedeon, M. Stein, L. Wang, and M. Mühlhäuser, "On Scalable In-Network Operator Placement for Edge Computing".

[49] T. Huang, W. Huang, B. Zhang, W. Chen, and X. Pan, "Optimizing energy consumption in centralized and distributed cloud architectures with a comparative study to increase stability and efficiency," Energy Build, vol. 333, 2025, doi: 10.1016/j.enbuild.2025.115454.

[50] S. S. Nabavi, S. S. Gill, M. Xu, M. Masdari, and P. Garraghan, "TRACTOR: Traffic-aware and power-efficient virtual machine placement in edge-cloud data centers using artificial bee colony optimization," International Journal of Communication Systems, vol. 35, no. 1, pp. 1–20, 2022, doi: 10.1002/dac.4747.

[51] S. Azizi, M. Shojafar, J. Abawajy, and R. Buyya, "GRVMP: A Greedy Randomized Algorithm for Virtual Machine Placement in Cloud Data Centers," IEEE Syst J, vol. 15, no. 2, pp. 2571–2582, 2020, doi: 10.1109/jsyst.2020.3002721.

[52] W. Wei, H. Gu, W. Lu, T. Zhou, and X. Liu, "Energy Efficient Virtual Machine Placement with an Improved Ant Colony Optimization over Data Center Networks," IEEE Access, vol. 7, pp. 60617–60625, 2019, doi: 10.1109/ACCESS.2019.2911914.

[53] S. Bani-Ahmad, S. Sa'adeh, S. Bani-Ahmad, and S. Sa'adeh, "Scalability of the DVFS Power Management Technique as Applied to 3-Tier Data Center Architecture in Cloud Computing," Journal of Computer and Communications, vol. 5, no. 1, pp. 69–93, Dec. 2016, doi: 10.4236/JCC.2017.51007.

[54] J. Masoudi, B. Barzegar, and H. Motameni, "Energy-Aware Virtual Machine Allocation in DVFS-Enabled Cloud Data Centers," IEEE Access, vol. 10, pp. 3617–3630, 2022, doi: 10.1109/ACCESS.2021.3136827.

[55] "ElasticTree: Saving Energy in Data Center Networks," in Proceedings of the 7th USENIX Symposium on Networked Systems Design and Implementation, San Jose, CA, USA, Apr. 2010.

[56] S. Xiao, Y. Cui, X. Wang, Z. Yang, S. Yan, and L. Yang, "Traffic-aware Virtual Machine Migration in Topology-adaptive DCN," Proceedings - International Conference on Network Protocols, ICNP, vol. 2016-December, Dec. 2016.

[57] A. Akbari, A. Khonsari, and S. M. Ghoreyshi, "Thermal-aware virtual machine allocation for heterogeneous cloud data centers," Energies (Basel), vol. 13, no. 11, 2020, doi: 10.3390/en13112880.

[58] J. Lin, W. Lin, W. Wu, W. Lin, and K. Li, "Energy-aware virtual machine placement based on a holistic thermal model for cloud data centers," Future Generation Computer Systems, vol. 161, pp. 302–314, 2024, doi: 10.1016/j.future.2024.07.020.

[59] S. Omer, S. Azizi, M. Shojafar, and R. Tafazolli, "A priority, power and traffic-aware virtual machine placement of IoT applications in cloud data centers," Journal of Systems Architecture, vol. 115, no. April, 2021, doi: 10.1016/j.sysarc.2021.101996.

[60] A. K. Singh, S. R. Swain, D. Saxena, and C. N. Lee, "A Bio-Inspired Virtual Machine Placement Toward Sustainable Cloud Resource Management," IEEE Syst J, vol. 17, no. 3, pp. 3894–3905, 2023, doi: 10.1109/JSYST.2023.3248118.

[61] H. F. Farimani, S. R. K. Tabbakh, D. Bahrepour, and R. Ghaemi, "Reallocation of virtual machines to cloud data centers reduce service level agreement violation and energy consumption using the FMT method," Journal of Information Systems and Telecommunication, vol. 7, no. 4, pp. 316–325, 2019.

[62] F. Alharbi, Y. C. Tian, M. Tang, W. Z. Zhang, C. Peng, and M. Fei, "An Ant Colony System for energy-efficient dynamic Virtual Machine Placement in data centers," Expert Syst Appl, vol. 120, pp. 228–238, 2019, doi: 10.1016/j.eswa.2018.11.029.

[63] S. Mashhadi Moghaddam, M. O'Sullivan, C. Walker, S. Fotuhi Piraghaj, and C. P. Unsworth, "Embedding individualized machine learning prediction models for energy efficient VM consolidation within Cloud data centers," Future Generation Computer Systems, vol. 106, pp. 221–233, 2020, doi: 10.1016/j.future.2020.01.008.

[64] A. Kamalinia and A. Ghaffari, "Hybrid Task Scheduling Method for Cloud Computing by Genetic and PSO Algorithms," Journal of Information Systems and

Telecommunication, vol. 4, no. 16, pp. 1–10, 2017, doi: 10.1007/s11277-017-4839-2.

[65] S. Sadegh, K. Zamanifar, P. Kasprzak, and R. Yahyapour, "A two-phase virtual machine placement policy for data-intensive applications in cloud," Journal of Network and Computer Applications, vol. 180, p. 103025, Apr. 2021, doi: 10.1016/J.JNCA.2021.103025.

[66] Y. Fan, H. Ding, L. Wang, and X. Yuan, "Green latency-aware data placement in data centers," Computer Networks, vol. 110, pp. 46–57, 2016, doi: 10.1016/j.comnet.2016.09.015.

[67] S. Farzai, M. H. Shirvani, and M. Rabbani, "Communication-Aware Traffic Stream Optimization for Virtual Machine Placement in Cloud Datacenters with VL2 Topology," no. May, 2021.

[68] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers," Concurrency Computation Practice and Experience, vol. 24, no. 13, pp. 1397–1420, 2012, doi: 10.1002/cpe.1867.

[69] S. Fang, R. Kanagavelu, B. S. Lee, C. H. Foh, and K. M. M. Aung, "Power-efficient virtual machine placement and migration in data centers," Proceedings - 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing, GreenCom-iThings-CPSCom 2013, pp. 1408–1413, 2013, doi: 10.1109/GreenCom-iThings-CPSCom.2013.246.

[70] S. Georgiou, K. Tsakalozos, and A. Delis, "Exploiting network-topology awareness for VM placement in IaaS clouds," in Proceedings - 2013 IEEE 3rd International Conference on Cloud and Green Computing, CGC 2013 and 2013 IEEE 3rd International Conference on Social Computing and Its Applications, SCA 2013, 2013, pp. 151–158. doi: 10.1109/CGC.2013.30.

[71] "Data center network architectures." [Online]. Available: https://en.wikipedia.org/wiki/Data_center_network_architectures

[72] C. Guo et al., "BCube: A high performance, server-centric network architecture for modular data centers," Computer Communication Review, vol. 39, no. 4, pp. 63–74, 2009, doi: 10.1145/1594977.1592577.

[73] L. Gyarmati and T. A. Trinh, "Scafida: A scale-free network inspired data center architecture," 2010. doi: 10.1145/1880153.1880155.

[74] A. Singla, C. Y. Hong, L. Popa, and P. B. Godfrey, "Jellyfish: Networking data centers randomly," Proceedings of NSDI 2012: 9th USENIX Symposium on Networked Systems Design and Implementation, pp. 225–238, 2012.

[75] M. C. Çavdar, I. Korpeoglu, and Ö. Ulusoy, "A Utilization Based Genetic Algorithm for virtual machine placement in cloud systems," Comput Commun, vol. 214, pp. 136–148, Jan. 2024, doi: 10.1016/J.COMCOM.2023.11.028.

[76] K. Lacurts, S. Deng, A. Goyal, and H. Balakrishnan, "Choreo: Network-Aware Task Placement for Cloud Applications," 2013, doi: 10.1145/2504730.2504744.

[77] Q. Zheng et al., "Virtual machine consolidated placement based on multi-objective biogeography-based optimization," Future Generation Computer Systems, vol. 54, pp. 95–122, Jan. 2016, doi: 10.1016/J.FUTURE.2015.02.010.

[78] T. Benson, A. Anand, A. Akella, and M. Zhang, "Understanding Data Center Traffic Characteristics," in Computer Communication Review, 2010, pp. 92–99.

[79] S. M. Nabavinejad and M. Goudarzi, "Communication-Awareness for Energy- Efficiency in Datacenters," in Advances in Computers, vol. 100, 2016, pp. 201–254.

[80] J. Sonnek, J. Greensky, R. Reutiman, and A. Chandra, "Starling: Minimizing Communication Overhead in Virtualized Computing Platforms Using Decentralized Affinity-Aware Migration ," 2009.

[81] G. Luo, Z. Qian, M. Dong, K. Ota, and S. Lu, "Improving performance by network-aware virtual machine clustering and consolidation," Journal of Supercomputing, vol. 74, no. 11, pp. 5846–5864, 2018, doi: 10.1007/s11227-017-2104-9.

[82] K. Zamanifar, N. Nasri, and M. H. Nadimi-Shahraki, "Data-aware virtual machine placement and rate allocation in cloud environment," Proceedings - 2012 2nd International Conference on Advanced Computing and Communication Technologies, ACCT 2012, pp. 357–360, 2012, doi: 10.1109/ACCT.2012.40.

[83] S. Aggarwal, N. Kumar, S. Tanwar, and M. Alazab, "A Survey on Energy Trading in the Smart Grid: Taxonomy, Research Challenges and Solutions," IEEE Access, vol. 9, pp. 116231–116253, 2021, doi: 10.1109/access.2021.3104354.

[84] J. K. Dong, H. B. Wang, Y. Y. Li, and S. D. Cheng, "Virtual machine placement optimizing to improve network performance in cloud data centers," Journal of China Universities of Posts and Telecommunications, vol. 21, no. 3, pp. 62–70, 2014, doi: 10.1016/S1005-8885(14)60302-2.

[85] C. Xu, Z. Zhao, H. Wang, R. Shea, and J. Liu, "Energy Efficiency of Cloud Virtual Machines: From Traffic Pattern and CPU Affinity Perspectives," IEEE Syst J, vol. 11, no. 2, pp. 835–845, 2017, doi: 10.1109/JSYST.2015.2429731.

Journal of Information Systems and Telecommunication
Vol.13, No.3, July-September 2025, 232-242
http://jist.acecr.org
ISSN 2322-1437 / EISSN:2345-2773

Simulation Based Economical Approach for Detecting Heart Disease Earlier from ECG Data

Md. Obaidur Rahaman[1,2], Mohammod Abul Kashem[1], Sovon Chakraborty[3,4], Shakib Mahmud Dipto[3,4]

[1]Department of Computer Science and Engineering, Faculty of Electrical and Electronic Engineering, Dhaka University of Engineering and Technology, Dhaka, Bangladesh.
[2]Department of Computer Science and Engineering, Faculty of Science and Engineering, Asian University of Bangladesh, Ashulia, Dhaka, Bangladesh.
[3]Department of Computer Science,Old Dominion University, Norfolk, Virginia, USA.
[4]Department of Computer Science and Engineering, University of Liberal Arts Bangladesh, Dhaka, Bangladesh.

Received: 11 Sep 2024/ Revised: 04 Aug 2025/ Accepted: 29 Sep 2025

Abstract

Cardiovascular diseases present significant challenges to public health in developing countries. The high costs of traditional treatments and the limited availability of specialized medical equipment contribute to these challenges. Current diagnostic methods often rely on specific electrocardiogram (ECG) parameters, which may not capture the nuanced complexities necessary for accurate diagnosis. To address these issues, our study proposes an innovative solution: an accessible and cost-effective ECG monitoring system. This system not only captures electrical signals from the heart but also translates them into numerical values using advanced modulation techniques. A trained deep learning model then analyzes this data to accurately identify any potential complications or confirm a healthy cardiac state. Our approach also allows for remote diagnosis and treatment. By utilizing an MQTT server, ECG data can be efficiently transmitted to experts for evaluation and intervention when necessary. Our meticulously fine-tuned Artificial Neural Network (ANN) architecture has achieved an impressive accuracy of 95.64%, surpassing existing methodologies in this field. Designed with resource-strapped regions in mind, our system offers a lifeline to rural areas lacking access to medical professionals and advanced equipment. Its affordability ensures that even individuals with limited financial means can benefit from timely and accurate cardiac monitoring, potentially saving lives and reducing the burden of cardiovascular diseases in underprivileged communities.

Keywords: Artificial Neural Network (ANN); Cardiovascular D isease; Electrocardiogram; Heart Disease; Modulation Techniques; MQTT Server.

1- Introduction

Cardiovascular diseases (CVDs) are a global health concern that poses a persistent threat to millions of people [1]. The heart and blood vessels are particularly vulnerable to CVDs, with coronary artery disease being a major contributing factor to the high death rates associated with these diseases [2]. In fact, it is estimated that CVDs account for 36% of deaths worldwide in the European Union alone [3]. Early detection of heart ailments is crucial for effectively addressing cardiovascular diseases. Continuous monitoring and measurement of heartbeats play a key role in this process. Electrocardiogram (ECG) signals, which provide comprehensive insights into heart-related issues through the analysis of physiological data, are a crucial tool [4,5]. Thanks to technological advancements, ECG monitoring devices now offer reliable measurement and observation of these signals [6,7]. However, there are ongoing concerns among researchers regarding the analysis of the data gathered from ECG monitoring devices. Critics argue that previously suggested devices are inadequate in keeping up with emerging technologies and lack comprehensiveness [8–10]. While some ECG monitoring devices boast specialized technology, others rely on context and server-based functionality [11,13]. Itt has been obesity has a major role in cardiovascular diseases that denotes heavily in the increment of heart rate. This highlights the pressing need for universal ECG monitoring equipment that can better assess and understand cardiac issues. By facilitating early detection and prevention of CVDs, these tools have the potential to save numerous lives [14]. The primary objective of this ANN architecture is to uncover patterns in ECG data

that may be difficult to detect by the human eye, thereby enhancing diagnostic capabilities. This advancement enables the early identification of cardiac issues, which is crucial for prompt treatment. In Bangladesh, a developing nation where 73% of individuals are reported to suffer from one or more cardiovascular diseases (CVDs), rural communities face significant healthcare challenges, including a lack of medical professionals and inadequate supplies. The motivation of this research is to address the health care challenges, especially in the domain of cardiovascular where most developing countries are suffering. Furthermore, an IoT-based device for detecting cardiovascular disease is proposed with less cost as economically these types of devices are not easy to buy. A proper communication medium between the device and doctors over the channel. Furthermore, the credibility of the data is being examined using multiple Machine Learning and Deep Learning architectures. Furthermore, the question arises what the proper model to will be to work with IoT devices. In [12], it has been discussed that DL architectures might not work properly with spatila and sequential data but can be effective if modified properly. Taking this into account authors have explored the opportunity to apply ANN in the gathered dataset. Finally, the primary focus is to proposing a IoT device that will be affordable for the people from underdeveloped countries.

This research paper addresses the following questions:

RQ1: How can cheaper IoT devices be bought for people from underdeveloped or developing countries?

RQ2: Can ANN be modified enough to communicate with IoT based devices properly?

The major contributions of this paper can be summarized as:

I) Implementing and validating real time gathered dataset that will be sent through IoT servers so that diseases can be detected earlier.

II) Proposing a shallow neural network that will instantly detect heart diseases from real-time data. Necessary suggestions will be provided instantly.

III) Building a low-cost device that will assist people from underdeveloped countries in order to detect heart diseases. The device is lightweight and portable.

The recent research from the literature has been discussed in section 2. The methodology and methods have been proposed in the section 3. Experimental results are shown in section 4 and finally, the future work and conclusion have been discussed in section 5.

2- Literature Review

The Internet of Things (IoT) is a fast-moving field in computer science that focuses on effectively sharing data between devices via cloud servers. The effectiveness of the cloud server being used determines how smoothly data is transferred. The authors of [14] offer a unique approach to signal capture in addition to signal preprocessing; nevertheless, an adequate encryption model is not implemented in this study. In [15], a crucial suggestion for Internet of Things (IoT)-based monitoring systems with sophisticated data visualization is made. But there is a significant difference in how deep learning (DL) structures and machine learning (ML) algorithms are integrated in this idea [16], which calls for more investigation. The state of IoT-based ECG monitoring systems [17–20] has given important new information on this field.

Predominantly, research has focused on signal collection, with a pivotal concern being data preparation. [21] addresses this by employing time-based feature integration for data purification. The microcontroller board utilized, namely the Arduino Uno, centers around the ATmega328T. Earlier studies have extensively utilized the Arduino Uno for cardiac signal acquisition [22–24], emphasizing its cost-effectiveness and ease of integration in such contexts.

The literature review explores various developments in the realm of Electrocardiogram (ECG) monitoring systems and associated technologies. In reference, an Internet of Things (IoT)-based ECG and vitals monitoring system is detailed, incorporating parameters such as QRS complex, heart rate, blood oxygen levels, and body temperature [25]. The iterative design approach is emphasized to reduce the device's overall cost. However, the three-lead end-to-end ECG acquisition system constructed proves inadequate for capturing all regular and augmented parameters of ECG signals. Moving on to fetal Electrocardiogram (FECG) monitoring, a system has been developed [26], concentrating on FECG and fetal heart rate (FHR) with an emphasis on an Android application. Nevertheless, improvement is deemed necessary, urging the incorporation of more miniaturized patches and real-time analytics via cloud computing. Addressing concerns about cardiovascular disease (CVDs) severity and the lack of precautionary monitoring systems, a low-cost solution is presented [27], aiming to reduce harmonic distortions and input inferred noise in ECG signal frequencies. This system highlights the need for an efficient cloud server for instantaneous data transfer. In another study [28], authors introduce a wearable Tele-ECG and heart rate monitoring

system, integrating a Singlet and Holter-based ECG system with a mobile application. Despite focusing on parameters such as P, Q, R, S, T peaks, the system requires additional sensors for a more comprehensive measurement of heart disease-related parameters. The proposed IoT-assisted ECG monitoring framework in [29] emphasizes secure data transmission for continuous cardiovascular health monitoring through automatic classification and real-time implementation. However, there's a call for advanced machine learning algorithms to enhance prediction accuracy. A smartphone-based ECG monitoring device is proposed in to evaluate post-ablation patients with atrial fibrillation. The focus lies on the ECG check monitoring protocol, considering sinus rhythm and sinus tachycardia. However, concerns are raised about the lack of a proper detection mechanism for ECG parameters, and the reported accuracy stands at around 93%. Some of the major research gaps are stated in Table 1.

Table 1: Identified Research Gap from the Literature

Reference	***Contributions***	***Research Gap***
Serhani ***et al.***	Precise collection of data sending through the IoT network.	No applications of DL methods to capture the proper semantics.
Ghosh ***et al.***	Integration of ML methods for detection purposes. Many algorithms are explored.	Device is costly and difficult to afford for under developed people.
Faruk ***et al***.	Enhanced accuracy than the state-of-the-art architectures.	The model is not lightweight and takes time to propagate real time data.
Rahman **et al.**	Methodology is described properly.	No proper system is available.

Based on the research gap available in the literature, it is important to identify a novel approach that will be available for the underdeveloped countries. This research focuses on proposing an approach that will integrate the DL approach detect cardiovascular diseases precisely along with the cost of the device is lower that can be affordable for rural people. The lightweight nature allows to detect cardiovascular diseases easily. The spatial information is also captured properly by the proposed model.

The below section comprehensively addresses the architectures, method of converting ECG signal, overall methodologies, and procedures employed in conducting the research. Initially, data collection was facilitated through the utilization of an ECG monitoring system, which is interconnected with 12 leads and necessary Internet of Things (IoT) devices. The proposed method of converting ECG signal is illustrated in section 3.

3- Materials and Methodology

Algorithm 1: *ECG Data Classification using ANN*

1. **Input:** ECG dataset with multiple columns
2. **Output:** Model performance evaluated using Precision, Recall, F1-score,
   and trainable parameters
3. **Step 1: Load the Dataset**
4. Load the ECG dataset.
5. Split the dataset into features (X) and labels (Y).
6. **Step 2: Parameter Tuning**
7. Identify hyperparameters to tune, such as learning rate, batch size,
   number of layers, and neurons.
8. Use grid search or random search to find the optimal hyperparameters.
9. **Step 3: Data Preprocessing**
10. Handle missing values using imputation techniques.
11. Normalize or standardize the data.
12. Apply noise reduction techniques if required (e.g., bandpass filtering).
13. Split the dataset into training, validation, and test sets.
14. **Step 4: Model Design**
15. Design an Artificial Neural Network (ANN) with an appropriate
   architecture.
16. Define the input layer based on the number of features.
17. Add hidden layers with appropriate activation functions (e.g., ReLU).
18. Define the output layer with a softmax activation function.
19. **Step 5: Model Training**
20. Compile the model with an appropriate optimizer (e.g., Adam) and loss
   function (e.g., categorical crossentropy).
21. Train the model on the training set.
22. Validate the model on the validation set during training.

23. **Step 6: Model Evaluation**
24. Evaluate the model's performance on the test set.
25. Calculate Precision, Recall, and F1-score for each class.
26. Analyze the trainable parameters in the model.
27. **Step 7: Performance Analysis**
28. Compare the model's performance based on the metrics.
29. Adjust hyperparameters or model architecture if necessary to improve performance.
30. Fine-tune the model using additional rounds of training and validation if required.
31. **Step 8: Final Model**
32. Save the final model and its parameters.
33. Document the model's performance metrics.
34. **Step 9: Reporting**
35. Prepare a report summarizing the methodology, results, and performance of the model.
36. Include plots of loss, accuracy, and confusion matrix if applicable.

Algorithm 1: Proposed Workflow

Algorithm 1 discusses the potential workflow of this research. Here, it is seen that, the dataset is loaded at first, then necessary parameter tuning has been performed in Table 2. For the preprocessing purpose, normalization, handling missing data and noise reduction is performed. Furthermore, authors are focused on designing the model with ANN that has been trained with the added hidden layers of ReLU. The performance of the model is analyzed and fine-tuned that has been reported with multiple performance metrics.

A threshold (τ) value condition on the amplitude of the signal will be calculated by the following proposed formula 1.

$$\tau = (0.6) \times m \quad (1)$$

Where m is the ISO electric line value. According to the characteristics of the ECG signal, it is possible to find out the different range of the amplitude for P, Q, R, S, and T parameters by applying the threshold value. An analog-to-digital converter needs to be configured to get the numerical value. This numerical value can be divided by the total number of parameters in a window of ECG signal to get the base numerical values as row data. This row data can be multiplied by different ratios of each parameter of the ECG signal to get the individual numerical value of P, Q, R, S, and T parameters. The formulation of converted numerical values is shown in Table 2. The parameters P, Q, R, S, T, U are tuned by the authors based on mathematical statistics [24].

In the second stage, augmented parameters (RR, PR, QT, QTc interval, and QRS complex) of the ECG signal can be considered to make better decisions about heart conditions provided in consultation with experts in cardiovascular diseases.

The proposed algorithm for formulation of RR interval can be established from the following steps

Table 2. The formulation of the parameters

ECG Basic Parameter	*Formulation of the parameter*	*Remarks*
P	$P = row\ data \times 1.1$	Always less than R peak
Q	$Q = row\ data \times 0.8$	Always less than P,T peak
R	$R = row\ data \times 2.0$	Maximum peak of ECG signal
S	$S = row\ data \times 0.7$	Always less than P,T peak
T	$T = row\ data \times 1.0$	Always less than R peak
U	$U = row\ data \times 0.4$	Always less than P,T peak

Step 1: Determine the overall sampling frequency (f_s) by giving a sample rate from the total ECG signal which is generated from the proposed device.

Step 2: Determine the sampling frequency (f_x) by partitioning the overall sampling frequency (f_s) according to the number of R peaks from each f_s.

Step 3: Individual window base average RR interval can be derived from the formula 2, which is denoted as $IWt_{rr}avg$.

$$IWt_{rr}avg = \frac{Trr_i}{number\ of\ R\ peak} = Trr_i = \frac{R_{loc(i+1)} - R_{loc(i)}}{(f_x)} \quad (2)$$

The other parameters PR, QT, QTc interval, and QRS complex can be calculated from the conventional methods [4], which is shown as following:

$$OWt_{rr(i)} = \frac{IWt_{rr}avg(i+1) - IWt_{rr}avg(i)}{f_s} \quad (3)$$

$$t_{pr}(i) = \frac{(R_{loc(i)} - P_{loc(i)})}{f_s} \quad (4)$$

$$t_{qt}(i) = \frac{t_{loc(i)} + (t_{rr(i)} \times 0.13) - (Q_{loc(i)} - x)}{f_s} \quad (5)$$

$$t_{qt(corr)}(i) = \frac{t_{qt(i)}}{f_s \times \sqrt{t_{rr(i)}}} t_{qrs}(i) = \frac{(S_{loc(i)} + x) - (P_{loc(i)} - x)}{f_s} \quad (6)$$

The proposed algorithm for the formulation of ST-Segment can be established from the following:

$$t_{st}(i) = \frac{(T_{loc(i)} - S_{loc(i)})}{f_s} \quad (7)$$

Where $S_{loc(i)}$ is called $J - point$ or S Depolarization, and $T_{loc(i)}$ is called $K - point$ or Beginning of the T wave.

The numerical values of these augmented parameters can be found by a computational programming application and the generated numerical values will be stored in cloud using MQTT technology.

Subsequently, meticulous preparation was undertaken to ensure a thorough understanding of the acquired data. Following this, an Artificial Neural Network (ANN) was employed to process the refined data. Fine-tuning of the model's hyperparameters ensued to attain the most optimal outcomes. Lastly, a diverse range of evaluation metrics were employed to gauge the performance of the model. Figure 1 illustrates the chronological sequence of actions undertaken throughout the entirety of the research work.

The process encompasses six primary phases prior to evaluating the outcomes. Initially, the designated equipment is employed to sense the data as suggested. Subsequently, the time intervals are converted into floating-point values upon retrieval. The initial presentation of the readings is in a waveform format, from which numerical values are derived based on the waveform intervals. Subsequent to this, the data undergoes preprocessing, entailing dimensionality reduction and null value elimination. Following preprocessing, the input is channeled into the proposed architecture of the artificial neural network. Various metrics are then employed to gauge the performance. Figure 1 elucidates the sequential execution of the entire investigative procedure.

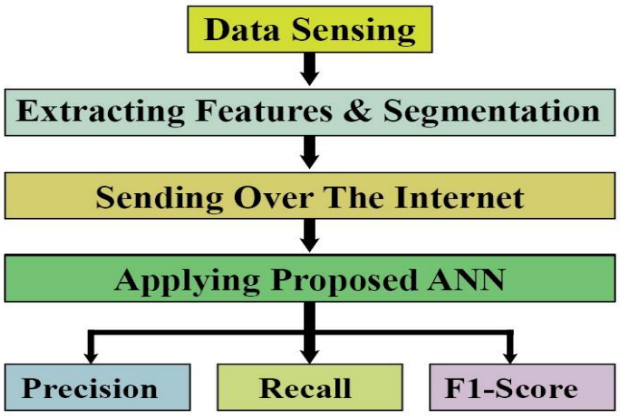

Fig 1: Methodology of the research

3-1- Requirements for Setting up the Device

The authors have focused primarily on establishing optimal conditions for successful implementation. The primary mechanism employed for collecting physiological data from patients' bodies is the ECG sensor network. To facilitate seamless data transmission, wireless channels are maintained using cloud-based IoT platforms. Within this framework, the AD8232 chip, utilized for electrical activity calculation, is integrated to record data from the device. Embedded within the chip is an integrated circuit (IC) responsible for signal amplification and extraction of requisite qualities. Electrocardiography serves as a pivotal diagnostic tool for numerous heart conditions, with several procedural steps involved in the data collection process. The initial step involves the implantation of multiple electrode pads—preferably three—into the patient's body for data collection. These pads play a crucial role in capturing data from the patient's body, which is subsequently transmitted to the AD8232 chip for analysis. Subsequently, the procedure entails the setup of a screen, commonly referred to as the Arduino COM port screen, through which medical specialists receive the data. Additionally, a Wi-Fi module is configured to facilitate data transmission from the device to experts. The detailed ECG curve displayed on the screen aids medical professionals in interpreting the data more effectively. The final stage entails deploying an Android app equipped with features that provide relevant suggestions. This app displays the ECG curve, aiding patients in comprehending the condition of their hearts better. Data transmission to the app is facilitated by the ECG sensors' ability to connect to integrated Wi-Fi. Moreover, ensuring the correct operation of the device, the Arduino Mega 2560 and earlier processors are configured to function between -3.3 and 3.3 volts, with pins appropriately connected from ground to ground.

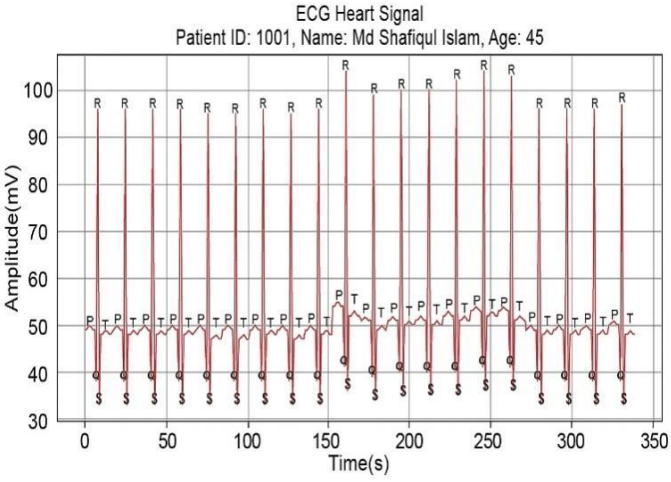

Fig 2: Visual Representation for Acquired ECG Data

3-2- Equipment Cost

The device's detailed cost is given in Table 3. It is clear that the gadget can be constructed for as little as 4231 BDT, or,

at the present exchange rate, 38.41 USD. This is the primary system's cost. There will be additional expenses, which cover cloud support and the monitor.

Table 3. Cost calculation of constructing the device

Name of the Components	*Cost in Bangladeshi Taka (BDT)*
Sensors	1142
Cables	174
Wi-Fi module	375
Serial converter	194
Breadboard	175
Arduino mega	934
Pins and others	294
Total	**4231 BDT**

Most of the devices that were proposed for detecting cardiovascular disease, most device cost around 1200 USD to 1800 USD [13,16,17]. On the other hand, the cost of the proposed device is only 39.5 USD. That is why this device is more affordable for people with low income.

3-3- Details of the Platform

The integration of technology within the medical industry has revolutionized the diagnosis and treatment of a myriad of medical conditions. Among the most profound technological advancements lies the development of ECG devices, designed to monitor the heart's electrical activity. These devices play a crucial role in identifying and treating various cardiac issues such as arrhythmias, ischemia, and heart attacks.

The MQTT [30] server is an ideal choice for transmitting ECG data due to its seamless handling of both analog and digital data. However, before data transmission can begin, certain prerequisites must be met. A minimum of 50 data points is required, and an ERROR alert is triggered if the ECG displays fewer than 70 data points. Furthermore, data transmission will not initiate if there are fewer than 50 data points available. This ensures that doctors receive only accurate and reliable data, which is crucial for precise diagnosis and treatment. To enable data transmission, the analog signal undergoes conversion into a digital format using a digital data converter inconsistency.

3-4- Dataset Building

To procure the necessary data, the authors conducted information gathering from a pool of 8,000 volunteers, spanning ages 18 to 75. Specifically, they recorded the durations between ECG waves, focusing on the P, Q, R, S, and T waves, along with the PR, RR, QRS complex, QT, and QTC intervals. Additionally, essential personal information was incorporated into the dataset.

Comprising 14 columns, each housing distinct data based on various criteria, the dataset primarily draws from ECG data to populate 10 of the 13 columns. Furthermore, it includes details such as an individual's ID, age, and BMI. The inclusion of age and BMI attributes enhances comprehension of an individual's health and well-being. The final column of the dataset provides information on the patient's heart condition, annotated by five Bangladeshi cardiac doctors. After thorough examination of each observation, they determined whether it suggests a healthy or at-risk heart. Table 4 displays attributes and their corresponding data types, offering healthcare professionals a comprehensive overview of the dataset contents. By examining the table, they can gain a better understanding of the dataset, facilitating more informed primary care decisions based on the patient's health status provided within. Data was gathered from volunteers where both patients with cardiovascular disease and healthy persons were available. During data collection, the protocols that were prescribed by a renowned hospital in Bangladesh is followed. All kinds of data biases are removed using statistical measures. Furthermore, wrongly collected data were eradicated during the preprocessing phase. The dataset does not poses that bias except demographic bias where the age difference is not properly balanced. The reason is that cardiovascular disease is mainly common in elderly people.

Preprocessing plays a pivotal role in enhancing the outcomes of Machine Learning (ML) and Deep Learning (DL) architectures. Fundamentally, the ECG signal furnishes all requisite information. Therefore, preprocessing steps are executed as necessary prior to feeding the data into the suggested optimized architecture.

Table 4. Attributes and their corresponding data types

Attribute Name	*Data Type*
P Wave	float32
Q Wave	float32
R Wave	float32
S Wave	float32
T Wave	float32
PR interval	float32
RR interval	float32
QRS complex	float32
QT-interval	float32
QTC-interval	float32
Age	Int64
BMI	float32
ID	Int64
Risk	Int64

Any empty rows or columns are meticulously addressed by the authors. Moreover, all data types are standardized to Int64 and Float32 formats. Subsequently, the dataset undergoes partitioning into training and testing subsets.

3-5- Data Cleaning and Preparation

During the preparation stages, categorical data is also encoded appropriately. Specifically, labels indicating healthy hearts are assigned values of 0, while those representing hearts at risk are assigned a value of 1. For clarity, a partial view of the dataset is presented in Table 5, providing insight into the encoded categories and their corresponding values.

The authors assess the data quality through the application of diverse statistical methods. Within this research, the evaluation entails measuring both covariance and correlation between the data. Covariance serves as a metric to gauge the relationship between variables, while correlation further elucidates the nature of this relationship, indicating whether the data exhibit linear separability or not. One sample of real-life data for 3 cycles has been given below. For each patient 3 cycles have been considered.

Table 5. Data annotation concerning the heart condition

Cycle	*P*	*Q*	*R*	*S*	*T*	*RR*	*PR*	*QRS*	*QT*	*QTc*
1	49	38	96	33	48	.64	.16	..05	.3	.75
2	49	38	96	33	48	.64	.16	..05	.3	.75
3	54	41	104	36	52	.43	.1	.05	.7	.78

The authors assess the data quality through the application of diverse statistical methods. Within this research, the evaluation entails measuring both covariance and correlation between the data. Covariance serves as a metric to gauge the relationship between variables, while correlation further elucidates the nature of this relationship, indicating whether the data exhibit linear separability or not.

3-6- Artificial Neural Network

Artificial Neural Networks (ANNs) are sophisticated machine learning models designed to emulate the structure and functionality of the human brain. These networks consist of layers of interconnected neurons that process and transmit data. Among the most commonly utilized types of ANNs is the feedforward neural network, which channels data from the input layer to the output layer in a unidirectional manner, devoid of looping back. To optimize performance for specific tasks, various training techniques are employed, allowing for the adjustment of connection strengths between neurons. ANNs excel in tasks necessitating pattern recognition, such as speech recognition, natural language processing, and image classification. Figure 3 illustrates the architecture of the ANN employed in the study. The authors conducted this study utilizing an 11th generation Core i7 PC equipped with a 1 TB HDD and 32 GB of RAM. The study leveraged the Python programming language, with Tensorflow and Keras serving as integrated libraries for constructing the architecture.

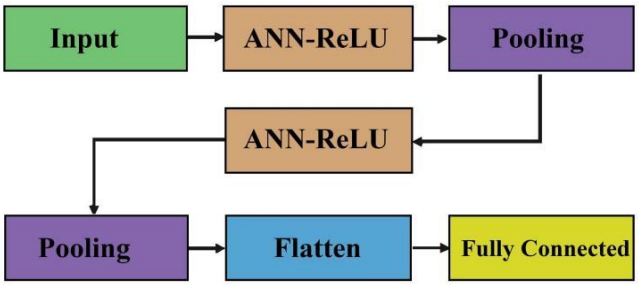

Fig 3. Architectural details of the ANN model

Additionally, Pandas facilitated the conversion of data into a dataframe, while numpy was instrumental in translating all calculations into vector space. Matplotlib.pyplot was utilized for plotting various graphs to aid in data visualization. Furthermore, Sklearn.train_test was employed to partition the data into separate test and train sets.

Table 6. Hyperparametric details of the architecture

Hyperparameters	***Details***
Learning rate	0.001
Loss function	Categorical cross-entropy
Epoch	40
Dropout	0.21
Number of dense layers	3
Trainable parameters	1,24,868
Activation functions	ReLU, softmax

4- Simulation of the Research

The authors aimed to integrate wireless technology and the Internet of Things (IoT) for efficient remote patient monitoring. The main technical objective is to develop an ECG sensor module that can accurately capture the heart's electrical signals, including the P, Q, R, S, and T waves [31], in real-time with high precision. These signals are thoroughly analyzed and extracted from the continuous ECG data stream.

ECG signals are wirelessly transmitted using robust communication protocols such as Bluetooth Low Energy (BLE) or Wi-Fi Direct, ensuring secure and rapid data transfer to a central server.

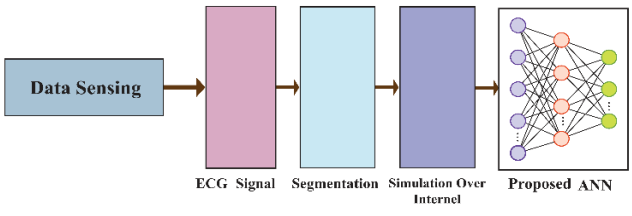

Fig. 4. Some significant stages of data processing

The system's architecture is carefully designed to accurately capture and transmit subtle variations in the amplitude and morphology of the PQRST complex. Figure 4 illustrates key stages from data sensing to processing through an artificial neural network.

Additionally, considerable emphasis is placed on optimizing power efficiency and scalability to support an expanding nework of interconnected devices. This focus aims to ensure prolonged battery life and seamless integration into healthcare infrastructure. Figure **5** provides a functional overview of the entire system.

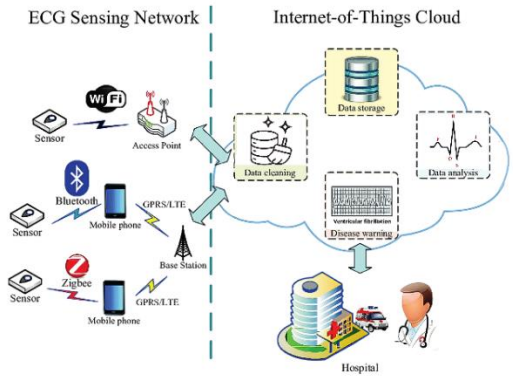

Fig. 5. Working of the whole system

5- Experimental Result Analysis

Initially, the dataset is employed to train the model, imparting knowledge on how attribute values differ between a healthy heart and one experiencing issues. Once trained and tested, the model can effectively evaluate readings obtained from the device, distinguishing between normal readings and those requiring further attention. The authors have meticulously tracked several performance measures to evaluate the model's efficacy. Key metrics assessed include accuracy, precision, recall, and F1-score, as detailed in equations (11) through (14). From the literature it has found that, in [32] the proposed AlexNet method performs much better than traditional machine learning models and other deep learning techniques. It achieved very high results in all major evaluation areas: 98.96% accuracy, 98.53% precision, 95.26% recall, 94.56% F1-score, and a correlation score of 0.988. These results are clearly better than other models, like the Support Vector Machine, which only reached 89% accuracy, and many others that stayed below 90%. This shows that the method is very good at correctly identifying different types of heart signals in electrocardiogram data. One of the key reasons behind this strong performance is the use of deep learning for feature extraction and a fuzzy bi-clustering approach, which together help the model pick up even small differences in heart patterns. However, one weakness is that the model still sometimes makes mistakes by wrongly classifying healthy or unrelated signals as heart conditions. For example, it wrongly identifies some signals as Atrial Fibrillation, Congestive Heart Failure, or Normal Sinus Rhythm, leading to small false positive rates of 2.5%, 3.0%, and 2.0% respectively. The study notes that while the model is highly effective, there is still room to reduce these incorrect predictions.

The outcome that the suggested ANN model produced is depicted in Table 7. Four measures are included in the performance analysis: F1-score, accuracy, recall, and precision. Overall, the Model's performance is extraordinary [32].

Table 7. Performance analysis of the system

Metrics	***Performance***
Accuracy	95.44%
Precision	94.35%
Recall	95.47%
F1-Score	95.64%

After completing the analysis, the authors focused on comparing the outcomes with state-of-the-art ML and DL architectures.

Initially, they compared the proposed model against the most advanced machine learning models, followed by comparisons with deep learning architectures.

According to their assessment, the suggested ANN model outperforms all existing highly effective ML and DL models, boasting an average F1 score of 98.87%. The comparison analysis is depicted in Figure 6.

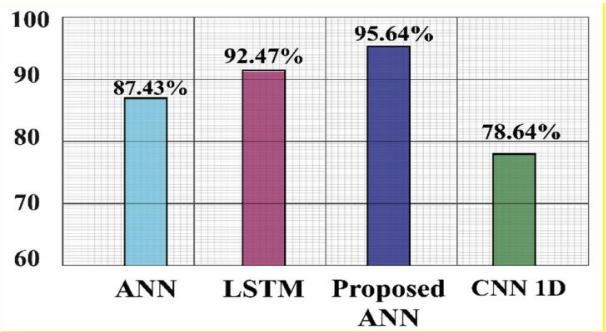

Fig. 6. Performance comparison of the proposed model with other state-of-the-art models

The results illustrated in Figure 6 highlight a notable enhancement in performance when compared to other models, with the deep neural network (DNN) emerging as the closest competitor. Furthermore, the authors juxtaposed the suggested model with the best deep learning architectures, considering the quantity of trainable

parameters in each model. Notably, the suggested model surpassed others by a considerable margin.

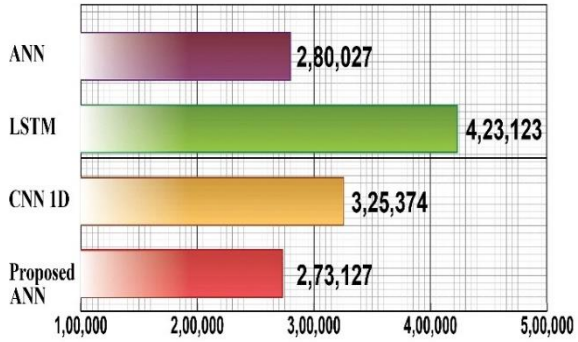

Fig. 7. Performance comparison given the number of trainable parameters

Figure 7 illustrates the count of trainable parameters for each of the DL architectures with which our model competed. Comparative analysis between the suggested ANN architecture and other DL architectures reveals that fewer trainable parameters are required, as evidenced by experimental results. From this result, it is evident, that the proposed model and device integrate properly to detect cardiovascular disease in a proper and economically friendly way. Furthermore, the device has a quick response time that will help doctors and patients to get benefits. Moreover, as the research is focused for the under developing countries that is why this device will help the whole medical sector of the world. The primary problem with LSTM is that it requires extensive data for understanding the sequencing. LSTM is very good in text data but not always in numerical values. Furthermore, CNN 1D can not perform proper with sequential data. That is why ANN is performing better and less trainable parameters because of optimization.

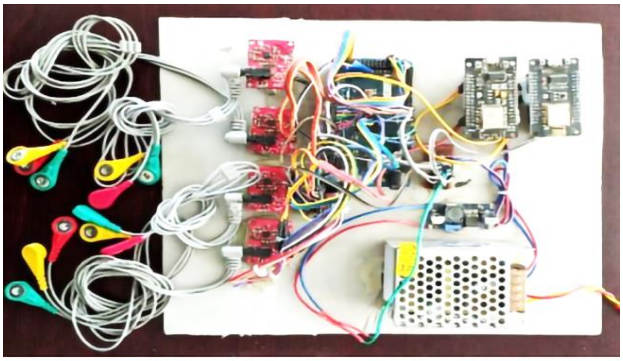

Fig. 8. The physical system corroborating this study

Figure 8 depicts the physical system supporting this study. This device is responsible for collecting personal data from users, which is then analyzed to provide them with the emergency medical attention they may require.

6- Conclusion

Leading the way in modern wellness, this study presents an IoT-based healthcare network that seamlessly integrates advanced sensors attached to the human body. A key innovation is the provision of continuous patient monitoring through multiple channels, including phone messaging services, live monitoring, websites, and apps. By blending state-of-the-art medical devices and applications with traditional medical practices, this approach aims to maximize effectiveness and make high-quality healthcare more accessible and affordable. Taking this into consideration, authors has focused on developing a IoT based device where it can used for medical purposes easily. The methodology suggest that with proper tuning and integration of ANN results in good result in classifying cardiovascular diseases. This work will aid the underprivileged countries to improve their medical sector. With the knowledge transferring from ANN, it is easier to determine the role of DL is immense. Furthermore, the proposed model is lightweight in nature. This research has resulted in the development of a cost-effective IoT-based ECG monitoring device, priced at only 38.41 USD. Experimental results show that using Artificial Neural Network (ANN) procedures, the system achieves 95.64 percent accuracy, outperforming alternative methods. The integration of IoT technologies with smartphones offers significant development. The broader implication is to integrate with real-life hospitals where this device and proposed model can be utilized to detect cardio-vascular disease at an earlier stage. The mortality rate can be reduced significantly in such cases. The research shows, this device has the ability to provide future direction in the health informatics field.

References

[1] J. Heaney, J. Buick, M. U. Hadi, and N. Soin, "Internet of Things-based ECG and vitals healthcare monitoring system," *Micromachines*, vol. 13, no. 12, p. 2153, Dec. 2022, https://doi.org/10.3390/mi13122153.

[2] J.-T. Huang, J. Zhang, W. Wang, P. He, Y. Su, and M. R. Lyu, "AEON: A method for Automatic evaluation of NLP test cases," *arXiv (Cornell University)*, Jan. 2022, doi: 10.48550/arxiv.2205.06439.

[3] Md. S. Azam, Md. A. Raihan, and H. K. Rana, "An experimental study of various machine learning approaches in heart disease prediction," International Journal of Computer Applications, vol. 175, no. 21, pp. 16–21, Sep. 2020, doi: 10.5120/ijca2020920741.

[4] X. Bao and Y. Deng, "Processing of Cardiac Signals for Health Monitoring and Early Detection of Heart Diseases," Ph.D. dissertation, King's College London, 2023.

[5] Sunny, Jithin S., C. Pawan K. Patro, Khushi Karnani, Sandeep C. Pingle, Feng Lin, Misa Anekoji, Lawrence D. Jones, Santosh Kesari, and Shashaanka Ashili. "Anomaly detection framework for wearables data: A perspective review on data concepts, data analysis algorithms and prospects." *Sensors* 22, no. 3 (2022): 756, https://doi.org/10.3390/s22030756.

[6] A. Belhani, H. Semira, R. Kheddara, and G. Hassis, "Implementation of uplink and downlink non-orthogonal multiple access (NOMA) on Zynq FPGA device," Journal of Information Systems and Telecommunication (JIST), vol. 4, no. 44, p. 269, 2023.

[7] M. A. Serhani, H. T. E. Kassabi, H. Ismail, and A. N. Navaz, "ECG Monitoring Systems: review, architecture, processes, and key challenges," Sensors, vol. 20, no. 6, p. 1796, Mar. 2020, doi: 10.3390/s20061796.

[8] V. Vijayan, J. P. Connolly, J. Condell, N. McKelvey, and P. Gardiner, "Review of Wearable Devices and data collection Considerations for connected Health," Sensors, vol. 21, no. 16, p. 5589, Aug. 2021, doi: 10.3390/s21165589.

[9] J. S and C. M. B. M. J, "Convolutional Neural Networks for Medical Image Segmentation and Classification: A review," Journal of Information Systems and Telecommunication (JIST), vol. 11, no. 44, pp. 347–358, Dec. 2023, doi: 10.61186/jist.37936.11.44.347.

[10] R. Patra, M. Bhattacharya, and S. Mukherjee, "IoT-Based Computational Frameworks in Disease Prediction and Healthcare Management: Strategies, challenges, and potential," in Studies in computational intelligence, 2021, pp. 17–41. doi: 10.1007/978-981-15-9897-5_2.

[11] Q. Abbas and A. Alsheddy, "Driver Fatigue Detection Systems using Multi-Sensors, Smartphone, and Cloud-Based Computing Platforms: A Comparative analysis," Sensors, vol. 21, no. 1, p. 56, Dec. 2020, doi: 10.3390/s21010056.

[12] H. E. Bays, C. F. Kirkpatrick, K. C. Maki, P. P. Toth, R. T. Morgan, J. Tondt, S. M. Christensen, D. L. Dixon, and T. A. Jacobson, "Obesity, dyslipidemia, and cardiovascular disease: A joint expert review from the Obesity Medicine Association and the National Lipid Association 2024," Journal of Clinical Lipidology, vol. 18, no. 3, pp. e320–e350, 2024.

[13] S. Yin, N. Xue, C. You, Y. Guo, P. Yao, Y. Shi, T. Liu, "Wearable physiological multi-vital sign monitoring system with medical standard," IEEE Sensors Journal, vol. 21, no. 23, pp. 27157–27167, 2021. doi: 10.1016/j.jacl.2024.04.001.

[14] Yang, Y., Bränn, E., Zhou, J., Wei, D., Bergstedt, J., Fang, F., ... & Lu, D. (2025). Premenstrual disorders and risk of cardiovascular diseases. *Nature Cardiovascular Research*, 1–10. https://doi.org/10.1038/s44161-025-00456-9.

[15] T. Shaown, I. Hasan, Md. M. R. Mim, and Md. S. Hossain, "IoT-based portable ECG monitoring system for smart healthcare," 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT), May 2019, doi: 10.1109/icasert.2019.8934622.

[16] B. Chong, J. Jayabaskaran, S. M. Jauhari, S. P. Chan, R. Goh, M. T. W. Kueh, *et al.*, "Global burden of cardiovascular diseases: projections from 2025 to 2050," *European Journal of Preventive Cardiology*, vol. zwae281, 2024.

[17] M. O. Rahman, M. A. Kashem, A.-A. Nayan, M. F. Akter, F. Rabbi, M. Ahmed, and M. Asaduzzaman, "Internet of Things (IoT) based ECG system for rural health care," International Journal of Advanced Computer Science and Applications (IJACSA), vol. 12, no. 6, 2021. doi: 10.14569/IJACSA.2021.0120653.

[18] Y. Zhao, X. Yang, Y. Du, L. Chen, J. Dong, T. Hu, N. Sun *et al.*, "Global cardiovascular disease burden attributable to particulate matter pollution, 1990–2021: An analysis of the global burden of disease study 2021 and forecast to 2045," *BMC Cardiovasc. Disord.*, vol. 25, no. 1, pp. 1–14, 2025 https://doi.org/10.1186/s12872-025-04724-6.

[19] Kelters, I. R., Koop, Y., Young, M. E., Daiber, A., & van Laake, L. W. (2025). Circadian rhythms in cardiovascular disease. *European Heart Journal, 46*(36), 3532–3545. https://doi.org/10.1093/eurheartj/ehae455.

[20] M. L. Sahu, M. Atulkar, M. K. Ahirwal, and A. Ahamad, "IoT-enabled cloud-based real-time remote ECG monitoring system," Journal of Medical Engineering & Technology, vol. 45, no. 6, pp. 473–485, May 2021, doi: 10.1080/03091902.2021.1921870.

[21] N. A. Nayan, R. Jaafar, and N. S. Risman, "Development of respiratory rate estimation technique using electrocardiogram and photoplethysmogram for continuous health monitoring," Bulletin of Electrical Engineering and Informatics, vol. 7, no. 3, pp. 487–494, 2018. doi: 10.1007/s40846-022-00700-z.

[22] A. Ghosh, A. Raha, and A. Mukherjee, "Energy-Efficient IoT-Health Monitoring System using Approximate Computing," Internet of Things, vol. 9, p. 100166, Jan. 2020, doi: 10.1016/j.iot.2020.100166.

[23] S. M. Ahsanuzzaman, T. Ahmed, and Md. A. Rahman, "Low cost, portable ECG monitoring and alarming system based on

deep learning," 2017 IEEE Region 10 Symposium (TENSYMP), pp. 316–319, Jan. 2020, doi: 10.1109/tensymp50017.2020.9231005.

[24] B. Patil, V. Rajan, and P. Patani, "Wearable fetal ECG monitoring system from abdominal electrocardiography recording," Journal of Pharmaceutical Negative Results, pp. 2383–2393, 2022. doi: 10.5555/jpnr.2022.2383.

[25] S. Chakraborty, M. B. U. Talukdar, P. Sikdar, and J. Uddin, "An Efficient Sentiment Analysis Model for Crime Articles' Comments using a Fine-tuned BERT Deep Architecture and Pre-Processing Techniques," Journal of Information Systems and Telecommunication (JIST), vol. 12, no. 45, pp. 1–11, Mar. 2024, doi: 10.61186/jist.38322.12.45.1.

[26] N. Xiao, W. Yu, and X. Han, "Wearable heart rate monitoring intelligent sports bracelet based on Internet of things," Measurement, vol. 164, p. 108102, Jun. 2020, doi: 10.1016/j.measurement.2020.108102.

[27] A. Badr, A. Badawi, A. Rashwan, and K. Elgazzar, "XBeats: a Real-Time Electrocardiogram monitoring and analysis system," Signals, vol. 3, no. 2, pp. 189–208, Apr. 2022, doi: 10.3390/signals3020013.

[28] N. A. Nayan and H. Ab Hamid, "Evaluation of patient electrocardiogram datasets using signal quality indexing," Bulletin of Electrical Engineering and Informatics, vol. 8, no. 2, pp. 519–526, 2019.

[29] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. Van Der Laak, B. Van Ginneken, and C. I. Sánchez, "A survey on deep learning in medical image analysis," Medical Image Analysis, vol. 42, pp. 60–88, 2017. doi: 10.1016/j.media.2017.07.005

[30] U. N. Cobrado, S. Sharief, N. G. Regahal, E. Zepka, M. Mamauag, and L. C. Velasco, "Access control solutions in electronic health record systems: A systematic review," Informatics in Medicine Unlocked, vol. 49, p. 101552, Jan. 2024, doi: 10.1016/j.imu.2024.101552.

[31] L. Pérez-Lombard, J. Ortiz, and C. Pout, "A review on buildings energy consumption information," Energy and Buildings, vol. 40, no. 3, pp. 394–398, Mar. 2007, doi: 10.1016/j.enbuild.2007.03.007.

[32] S. T. Aarthy and J. L. Mazher Iqbal, "A novel deep learning approach for early detection of cardiovascular diseases from ECG signals," *Medical Engineering & Physics*, vol. 125, p. 104111, Mar. 2024, doi: 10.1016/j.medengphy.2024.104111.

[33] S. Begum, E. Ghousia, E. Priyadarshi, S. Pratap, S. Kulshrestha, and V. Singh, "Automated detection of abnormalities in ECG signals using deep neural network," Biomedical Engineering Advances, vol. 5, 100066, 2023.

Journal of Information Systems and Telecommunication
Vol.13, No.3, July-September 2025, 243-255
http://jist.acecr.org
ISSN 2322-1437 / EISSN:2345-2773

Enhancing Computational Offloading for Sustainable Smart Cities: A Deep Belief Network Approach

Kaebeh Yaeghoobi[1*], Mahsa Bakhshandeh N.[2]

[1].Faculty of Computer Engineering, K. N. Toosi University of Technology, Tehran Iran
[2].Faculty of Engineering, Ale-Taha Institute of Higher Education, Tehran, Iran

Received: 12 Nov 2024/ Revised: 11 Sep 2025/ Accepted: 13 Oct 2025

Abstract

The use of mobile devices with limited processing power has surged in recent years, alongside the expansion of cloud and fog computing across various sectors. These devices can handle small to medium computing tasks, but they fall short when it comes to large-scale processes, making computational offloading a crucial solution. Cloud computing and fog computing provide an effective platform for offloading tasks from mobile devices. However, critical real-time applications necessitate a near-edge approach to managing the computational load. Significant challenges exist in optimizing response times for effective offloading in cloud computing. This research introduces a framework for predicting response times using Deep Belief Network (DBN) learning to enhance offloading performance. Implementing a DBN aims to minimize response times and resource consumption, thereby improving the overall efficiency of offloading processes. The framework is designed to predict response times accurately, ensuring timely completion of tasks and efficient use of resources. Simulation results using multiple models show that the use of DBN significantly reduces processing, response, and offloading times compared to other algorithms. Consequently, the DBN algorithm proves to be more efficient in predicting response times and enhancing offloading performance. By leveraging the capabilities of DBN, this framework provides a promising solution for optimizing computational offloading in cloud computing environments. This enhances the performance of mobile devices and ensures the reliability and efficiency of real-time applications, direct the way for more advanced and responsive computing technologies.

Keywords: Computational Offloading; Cloud Computing; Deep Belief Network; Response Time; Resource Management; Sustainable Smart Cities; Real-time Management.

1- Introduction

The proliferation of mobile devices has substantially increased computing demands, introducing new challenges in communication networks and resource provisioning. Due to their limited resources, mobile devices struggle with large-scale image processing and real-time conversion services [1]. Cloud computing technology helps mitigate these limitations; however, it is not applicable for real-time applications considering latency issues. Consequently, offloading computational tasks to independent platforms becomes a practical solution. For instance, the mobile cloud can provide maximum advantage for mobile video gaming and streaming [2].

Nevertheless, mobile cloud computing encounters challenges such as limited network bandwidth and offloading latency. Transmitting data from mobile devices to distant clouds consumes significant bandwidth, leading to traffic congestion and increased latency. Latency-sensitive applications require offloading to nearby locations, such as the nearest edge or mobile fog, to address these issues [3].

Cisco Systems introduced fog computing as an extension of cloud computing, bringing its capabilities to the network's edge. This extension benefits IoT services by supporting latency-intolerant mobile services. Numerous studies have focused on standardizing the computational offloading process at the edge or mobile fog, particularly in selecting mobile application units. Challenges related to offloading at

✉ **Kaebeh Yaeghoobi**

yaeghoobi@kntu.ac.ir

the mobile edge or fog include mobility, heterogeneity, and geographic distribution of devices.

As the digital world expands and network technologies evolve, complex services are emerging [4]. The generation of online applications featuring computing, communication, and intelligent capabilities continues to grow. Despite the growing power of current devices, they still struggle with tasks required for smart healthcare, augmented reality, intelligent car communication, and many smart city services. These applications often require another individual to execute tasks as a representative of the user's device, a technique known as process offloading [5]. Task disburdening is especially advantageous for Internet of Things and cloud computing requisition, facilitating interactions between edge devices or fog nodes and sensors and IoT nodes. Load shedding can be established on computational requirements, load balancing, energy management, and latency management [6].

In a data-rich world, mobile devices with limited resources can handle small-to-medium computations but struggle with high-level computations. Processing offloading is an effective solution to overcome this limitation. Recently, cloud computing has been recognized as a suitable platform for offloading tasks from mobile devices. However, the distance of cloud data centers from mobile devices increases network latency and affects the performance of real-time IoT applications.

For essential real-time applications, employing a near-edge network approach for computing offload is vital. Additionally, the primary controls for distributed mobile devices are heterogeneous in the offloading process of mobile computing. To overwhelm these contests, a deep learning-based response time prediction framework has been implemented to optimize offloading decisions near fog/edge or cloud nodes.

The objectives of this research are:

- Enhance Offloading Performance: Develop a deep learning-based framework to improve computational offloading efficiency.
- Minimize Prediction Error: Achieve the lowest discrepancy between actual and predicted response times using deep learning techniques.
- Boost Prediction Accuracy: Enhance the accuracy of response time predictions with the proposed deep learning method.

The paper is structured as follows: Section 2 covers related concepts and foundational research. Section 3 outlines the technical methodology, including the proposed method and framework. Section 4 analyses the proposed framework, presents results, and evaluates their theoretical implications. The final section discusses the results' implications and concludes with future trends and perspectives.

2- Background

This section explores concepts and metrics used in computational offloading, IoT middleware technologies, technologies that enhance fog computing tasks, and offloading methods in fog and cloud computing. The interplay between cloud, fog, and mobile computing models, concerning large computing resources, is analyzed. The literature review also covers computing resource allocation methods and achievements in cloud computing offloading.

Cloud computing resources are managed using virtualization technology. For example,[7] explains optimal virtual machine placement, examining distribution methods in cloud data centers. Most resource allocation mechanisms are designed for green computing. The DPRA allocation mechanism, discussed in [8], considers energy consumption of virtual and physical machines and data center air conditioning. A comparison of three schemes with DPRA shows energy savings, PM shutdowns, and reduced VM migrations.

In [9], a multi-objective optimization algorithm balances availability, costs, and performance for running big data applications in the cloud, outperforming conventional methods by reducing costs and achieving higher performance. However, the study focuses on big data applications.

In critical real-time applications, for example, patient control systems and intelligent transportation, mobile cloud computing offloads large tasks while maintaining quality standards [10]. A mobility-aware resource allocation architecture, Mobihat, provides efficient scheduling but does not study the impact of mobility on delay and response times for real-time mobile services.

Offloading mobile edge computing with multiple users, based on TDMA and OFDMA, is introduced in [11]. The TDMA-based method reduces mobile energy consumption, while the OFDMA hybrid model transforms into TDMA, defining a discharge priority function for optimal resource allocation.

The optimal computational offloading framework for DNNs is presented in [12], considering mobile batteries and cloud resources. This method evaluates energy consumption and execution time.

In [13], battery life of nearby mobile devices is used to select discharge positions. A non-interactive game model, maximizing player payoffs, reduces response times. The Nash equilibrium is obtained through the game model and indirect induction method, evaluated for response time, end-user benefit, and memory usage. Yang et al. [14] address high implementation delays among mobile devices and fog nodes using queuing theory. Data rate and power consumption are selected as decision parameters, formulating a multi-objective optimization problem to decrease transmission

energy consumption, power, and cost, determining the probability of discharge for all mobile devices.

A survey on stochastic-based offloading methods in different computing environments, including mobile cloud, edge, and fog computing, is proposed in [15]. The classification is divided into Markov chain, Markov process, and hidden Markov models, discussing open issues and future challenges.

In [16], a multi-objective optimization model addresses time and energy consumption of mobile users and edge server resource utilization. An edge-cloud joint offloading method, based on the evolved Strength Pareto algorithm, is effective and efficient for scenarios with multiple mobile users and heterogeneous edge servers.

An offloading architecture, combining intelligent computing with AI, is presented in [17]. Considering mobile task data size and edge node performance, a load shedding and task transfer algorithm optimize edge computing offloading. Experiments show reduced task delay by increasing data and subtask execution.

Du et al. [18] address offloading in a cloud-cloud environment, supporting a heterogeneous model to consider task communication cost asymmetry. They prove the NP-hard nature of the problem and design an efficient algorithm for an optimal solution, evaluated through a PageRank-based program in a controlled cloud edge setting.

An adaptive wireless resource allocation strategy for computational offloading, under a three-layer edge cloud framework, is studied in [19]. Modeling the offloading process at the minimum block level of allocable wireless resources adapts to vehicular scenarios and evolves in the 5G network. The proposed value density function measures cost-effectiveness and energy saving. Numerical results show the designed algorithm achieves significant running time and energy savings, with superior performance compared to benchmark solutions.

An autonomous computational offloading framework is presented in [20] for time-consuming programs, addressing control model challenges for managing computing load. Various simulations, including deep neural networks and hidden Markov models, are performed. Results show the hybrid model fits the problem with near-optimal accuracy for discharge decisions, delay, and energy consumption predictions. MAPE is used for discharge, collection, and processing for decision making. The proposed method outperforms local computing and offloading in latency, energy consumption, network utilization, and execution cost.

In [21], minimizing average task execution time in edge systems, considering job request heterogeneity, application data pre-storage, and base station cooperation, is addressed. A mixed integer nonlinear programming (MINLP) problem is formulated and addressed using decomposition theory. The GenCOSCO algorithm improves service quality and computational complexity. For fixed service cache configurations, the FixSC algorithm derives evacuation strategies, with simulations showing significant task execution time reductions.

Peng et al. [22] propose three multi-objective evolutionary algorithms to tackle the computing offloading challenges in IoT for edge and cloud networks. They developed a constrained multi-objective load calculation model that accounts for time and energy consumption in mobile environments. Drawing inspiration from the push and pull search (PPS) framework, they introduced three algorithms (PPS-NSGA-II, PPS-SPEA2, and PPS-SPEA2-SDE) that integrate population-based search with flexible constraint control. These algorithms were tested using multi-task, multi-user scenarios across various IoT devices. The results demonstrated their effectiveness and superiority.

Other research presents a user-centered joint optimization offloading scheme designed to minimize the weighted costs of time delay and energy consumption. The mixed-integer nonlinear programming problem is addressed using a particle swarm optimization algorithm that incorporates 0-1 and weight improvement techniques. Simulation results indicate higher performance in delay, energy consumption, and cost [23].

In [24], a computation offloading scheme via mobile vehicles in a cloud-IoT network is proposed. Sensing devices generate tasks and transmit them to vehicles, which then decide whether to compute the tasks locally, on a MEC server, or at a cloud hub. The offloading decision is based on a utility function that considers energy consumption and transmission delay, using a learning-based approach. Experimental results show that this solution maximizes rewards and reduces delay.

Based on the research discussed, various techniques can be adopted for cloud computing offloading, depending on priorities. This research proposes using a response time prediction model based on deep learning to determine the optimal offloading position. The impact on delay and energy efficiency will be evaluated to improve offloading performance by minimizing the error between actual and predicted response times.

3- Methodology

A mobile fog node expands the capabilities of fog and mobile cloud computing models by offering a localized system to minimize potential delays and execution times while maintaining continuous and direct communication in conjunction with the cloud data center. The proposed model, depicted in Figure 1, encompasses three offloading positions: the cloud data center, adjacent mobile station, and mobile fog. This setup is supported by the LTE hierarchical architecture and the Wi-Fi intra-network reference model, situating the mobile fog at the network's edge. Access points and access point controllers operate as mobile fog nodes.

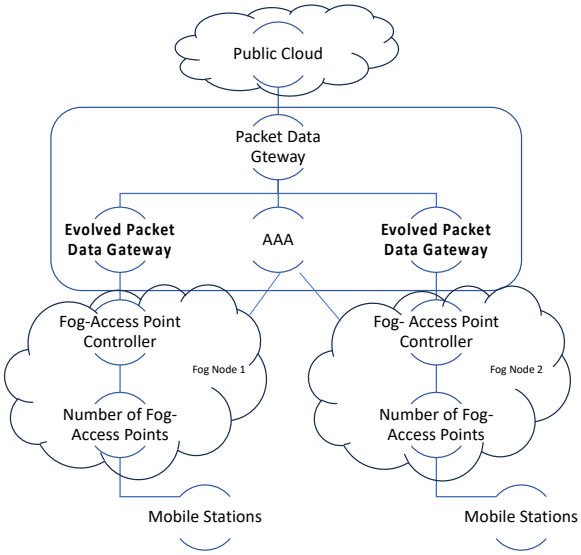

Fig. 1 Mobile Fog System Model for Computational Offloading - Verification and confirmation of Mobile Stations is Achieved by 3GPP AAA via Extensible Authentication Protocol-Authentication and Key Agreement(EAP-AKA) over Internet Key Exchange version 2 (IKEv2)

Within this architecture, the mobile edge/fog is represented by the fog-1 node, the mobile fog by the fog-2 node, and the public cloud serves as the third offloading position, referred to as the cloud node. Communication within the fog is enabled by the Evolved Packet Core, which provides the Evolved Packet Data Gateway.

Access points not only facilitate communication between mobile stations but also offer cloud services such as, Network as a Service (NaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). IEEE Ethernet interfaces connect access points to access point controllers, while IEEE 802.11 WLAN interfaces link mobile stations to access points. The access point controller manages block code migration, overseeing memory, processing, I/O, and networking capabilities to sustain mobile cloud services. Hence, the access point controller similarly serves as a fog network controller. In Figure 1, fog-enabled access points are labeled as "fog-access points," and access point controllers are designated as "fog-access point controllers." Mobile station authentication is conducted by the 3GPP AAA via EAP-AKA over IKEv2, with the verification and validation vector derive through the shared home server unit in the LTE network. The data network gateway, which handles access to user equipment or mobile stations and virtual machines (VMs), has evolved into a packet data gateway. The top module, the public cloud, functions as a traditional delivery network, providing pervasive and scalable services accessible via the web using both mobile and static devices.

3-1- Unloading Node Process

This section details the offloading process based on the previously described model, with a focus on the fog/mobile edge. In critical real-time applications, nodes such as public cloud and mobile fog and mobile edge are physically dispersed to deliver services to mobile cloudlets, which are resource-limited mobile stations. Due to the dynamic nature of these applications, request times are unknown and random, with variable response times, making it challenging to identify the optimal offloading node.

To tackle this issue, a deep learning-based approach is recommended. This approach learns from the request history and response times of nodes to predict future response times. The node with the lowest predicted response time is then selected for offloading. The relationship between the computing requirements of cloud or fog nodes and the response time of virtual machines is complex.

Predicting workload data patterns is challenging due to their non-consecutive nature. Therefore, aggregated workload data characteristics of VMs are used instead of single VM data for prediction purposes. A deep learning model can better determine workload data dispersions based on inherent data characteristics, outperforming simpler models. This preference is due to the deep model's ability to learn complex relationships between workload data features. Although structurally similar to a Multi-Layer Perceptron (MLP), a Deep Belief Network (DBN) has a diverse training method, allowing it to address gradient fading effectively.

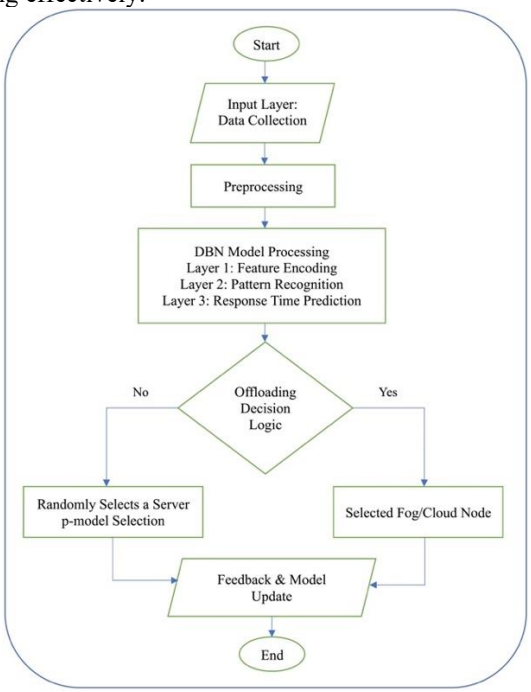

Fig. 2 Flowchart of the DBN-based offloading decision process, integrating predictive modelling, fallback selection via p-model, and feedback-driven model updates for sustainable smart city applications.

Figure 2 illustrates the complete workflow of the proposed DBN-based computational offloading system for smart city environments. The process begins with data collection from mobile devices and virtual machines, including historical request patterns and aggregated workload characteristics. After preprocessing and feature extraction, the data be used for DBN step, which performs multi-layer encoding and pattern recognition to predict future response times of candidate nodes. Based on these predictions, the system attempts to select the node with the lowest latency for offloading. If due to unpredictable workload patterns or insufficient confidence no suitable node is identified, the system activates a fallback mechanism using the p-model, which randomly selects a server based on predefined probability. The final stage involves task execution and feedback logging, which continuously refines the DBN model for future decisions.

3-2- Deep Belief Network (DBN)

A Restricted Boltzmann Machine (RBM) can extract features and recreate data entry, in spite of that, it struggles with gradient blurring. To address this, multiple RBMs can be combined with a classifier to form a Deep Belief Network (DBN). This method, known as greedy layer-by-layer unsupervised pretraining, involves training the DBN two layers at a time, treating each pair of layers as an RBM. In this architecture, the hidden layer of one RBM acts as the input layer for the subsequent RBM. The training process starts with the initial RBM, whose outputs are fed into the next RBM, and this sequence continues until the output layer is reached. Through this process, the DBN identifies inherent data patterns, functioning as an advanced multi-layer feature extractor. A unique aspect of this network is its ability to learn the complete structure of the input at each layer, similar to a camera gradually focusing an image.

Finally, labels are applied to the resulting patterns in the DBN. The DBN is subsequently fine-tuned through supervised learning using a small set of labeled samples, with minor changes to weights and biases leading to a marginal increase in accuracy.

The proposed approach includes a deep belief network with one-layer neural network. This method employs an unsupervised approach to extract more robust and helpful features from VM workload data. By increasing the hidden layers in the DBN, the error gradient is significantly amplified before being minimized. Training is conducted using an unsupervised greedy layer-wise method. To further optimize, the DBN's top layer utilizes a standard sigmoid regression. Future request predictions are generated by analyzing response times in terms of bandwidth *(B)*, memory *(M)*, and processing capability *(P)*.

As presented in Figure 3, inputs to the DBN model include the bandwidth, memory and processing capability of entire requests, along with the recent workload of all VMs. These data cover actual response times discovered over various time spans. For each node, the trained DBN models predict response times, with input values normalized between 0 and 1. The core layer's units equal the sum of the VMs in the cloud and the time slots.

$$Number\ of\ Units = VM \times TI \quad (1)$$

Where:

VM represents the number of virtual machines.

TI represents the number of time intervals.

This simple yet effective formula helps determine the total number of units required based on the given parameters.

Alternatively, a supervised approach with a precisely configured logistic regression layer can be employed to label the data and predict the workload of a VM.

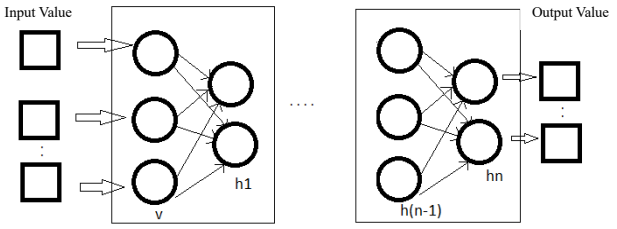

Fig. 3 Stacks before RBM Training

Initially, the standard binary RBM is modified to a Gaussian-Bernoulli RBM. The visible unit biases in the RBM energy function are adjusted to include quadratic bias terms [3]. An example of a load shedding decision session is shown in Table 1. The Energy function and Conditional Probability Distribution are conveyed in following way:

$$\mathrm{E}(\mathrm{x},\mathrm{h}|\theta) = \sum_{i=1}^{X} \frac{(x_i - a_i)^2}{2\sigma_i^2} - \sum_{j=}^{H} b_j h_j - \sum_{i=1}^{X}\sum_{j=1}^{H} \frac{v_i}{\sigma_i} h_j w_{ij} \quad (2)$$

$$P(h_i|x;\theta) = \delta\left(\sum_{i=1}^{X} w_{ij}x_i + b_j\right) \quad (3)$$

$$P(x_i|x;\theta) = N(\sigma_i \sum_{i=1}^{X} w_{ij}x_i + a_j, \sigma_i^2) \quad (4)$$

Table 1: Description of symbols

Symbol	***Description***
μ	mean
σ^2	variance
σ	standard deviation
P	probability
E	expectancy
X	observable variables
H	common hidden space of variables
W	linear mapping coefficient
B	bias

In this context, the Gaussian distribution's probability distribution function is represented by $N(\mu,\sigma^2)$, where μ is the mean, and σ^2 is the variance vector. Hinton's training method outlines the prediction process as follows:

Unsupervised Training: The RBN visible and hidden layer are trained. The RBM input comprises a request section and a response time dataset. θ is the only non-continuous parameter in the RBM.

Layer Inheritance: Each visible layer in RBM inherits and utilizes the extracted features of the preceding RBM as its input. This process is repeated for subsequent RBMs, with the parameter θ retained for the next and initial RBM.
Input to Logistic Regression: The regression layer is trained using labelled data in a supervised manner; and input of that is the output of the final RBM.
Supervised Training: The θ parameters are trained and adjusted using the backpropagation (BP) algorithm.
The deep belief network-based response time prediction method leverages edge/cloud computing to accurately determine whether to offload computations to a neighbouring node, an edge/fog node, or a cloud node. To handle the unpredictability of resource availability in edge/fog and cloud nodes, the proposed offloading procedure leverages the technique of RBM learning.
To begin the substantial data volumes and the demand for real-time applications, particularly in the e-health sector, a near-edge network approach for offloading computations is recommended. This strategy addresses the primary controls for distributed mobile devices, easing the offloading process in mobile and heterogeneous computing environments. A deep learning-based response time prediction framework has been developed to enhance computational offloading performance, determining the optimal offloading target, whether it's a nearby fog/edge node, an adjacent fog/edge node, or a cloud node. Additionally, the Restricted Boltzmann Machine (RBM) learning technique is utilized to handle the variability of resource availability.
In this study, the DBN model was trained using aggregated workload data collected from simulated virtual machines operating under diverse conditions. The training process involved unsupervised pre-training of Restricted Boltzmann Machines (RBMs) followed by supervised fine-tuning using labeled response time data. Training was conducted on a standard CPU-based computing environment, which, was sufficient for the scale and complexity of the dataset used. The total training time varied depending on the configuration, typically ranging from 30 minutes to 2 hours. Once trained, the model was deployed for inference on edge servers, where its lightweight architecture enabled real-time prediction without significant computational overhead. This setup demonstrates that even without specialized hardware, the DBN-based offloading strategy remains practical and effective for mobile and fog-based environments.

4- Result and Analysis

This section examines the performance of the proposed models. The simulation results integrate real mobility tracking, server datasets, and model implementation on actual machines. Subsequent sections will explore the performance benefits of DBN-based models using three probability distributions (uniform, normal, and exponential) to achieve accurate results.

4-1- Data Collection

To simulate mobile node movements, a dataset of vehicle movements in Rome was utilized, as referenced in [25]. This dataset comprises coordinates of 320 taxis collected over 30 days, including their coordinates, date, time, and GPS location. Mobility tracking treats any movement as a point in time to check server or dump time, rather than studying user mobility. Each movement is modeled as an interaction with a mobile edge computing server. Processing times are obtained from real servers (CPU usage), involving around 150 data servers (over 1 billion rows). With e very movement, a server is selected from the dataset, its utilization is checked, and an unloading decision is made based on the model's recommendation.
The evaluation spans more than five days (5000 rows of movements). An evacuation decision is made every minute, resulting in over 1000 evacuation decisions, ensuring the proposed models' behavior is observed over an extended period. The DBN-based response time prediction method leverages edge/cloud computing to determine whether to offload computations to a neighboring node, an edge/fog node, or a cloud node.
Given the challenges posed by large data volumes and real-time applications, particularly in the e-health sector, a near-edge network approach was recommended for offloading computations. The proposed RBM learning technique addresses the randomness of resource availability.
Figure 4 distribution of server usage probabilities across all servers in the dataset. The data generally follows a normal distribution, illustrating typical CPU utilization patterns observed during simulation.

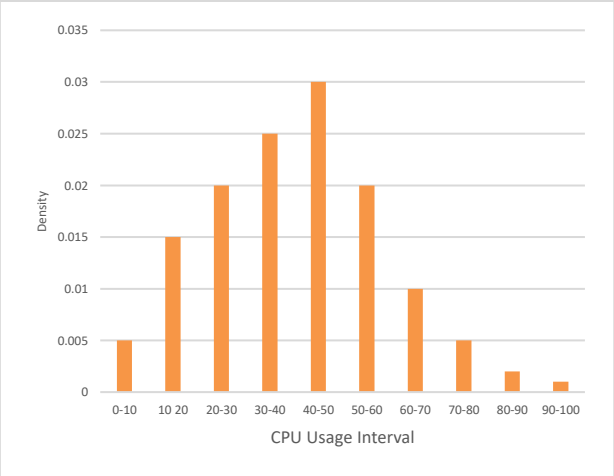

Fig. 4. CPU usage distribution of servers (CPU unit is percentage and Density is J)

Table 2 sample load shedding decision session, showing CPU consumption values for selected mobile edge computing servers at specific geographic positions and time intervals.

Table 2: Dataset Sample Used in the Experiment. (ID: xx6, Motion Time Interval: 10 Seconds)

Position	*Machine*	*CPU Consumption*
X=41.8911, Y=12.49073	M_xx39	51
X=41.89905, Y=12.4899	M_xx36	47
X=41.8994, Y=12.48940	M_xx41	20
X=41.8994, Y=12.489401	M_xx41	37

4-2- Evaluation

This section focuses on simulating and evaluating the proposed evacuation rules across various variables. The primary aim is to observe the models' behavior under different conditions, allowing generalization to parameters such as quality of service and response time concerning computational load.

MATLAB software is chosen for the simulation, which can perform process-based discrete event simulation. The "Advance Mode" is selected for the probability distribution of the random variable *X*, including time (processing). In the simulation, a resource actually is a mobile edge computing server *k* that is modelled and can advertises its processing time *Xk*. A process is a mobile node that modelled to traverses the mobile edge computing servers and checks latency of each server based on the processing time. Initially, we consider *n=5,* means having five mobile edge computing servers. The processing time *X* follows a normal distribution (50 ms to 10 ms), a uniform distribution in the interval [0-1], and a binominal distribution of 50 J/mol. MATLAB has generated incidental variables following the determined apportionment.

At every initiation, a node begins polling the mobile edge computing servers consecutively, starts with server one. At this step, the proposed approaches are utilized to choose a mobile edge computing server. The important parameters in processing time are waiting time, delay and total delay. Additionally, based on the program types, the range of processing time differs from 100 milliseconds to 800 seconds, and in intervals of 10 milliseconds to 30 milliseconds. Therefore, various ranges for parameter *X* can be considered derived from the proposed models, which producing similar outcomes as observed in the experiment dataset. Table 3 shows the values and range of parameters in the simulation test.

The main approach used in the simulation involves comparing values obtained from other studies, random values, the nearest server (immediate loading), and a method from the same family of algorithms proposed in this work. This evaluation is limited to comparisons between different models, including the random and probabilistic model *(p)*. These approaches are compared to the superior option, where the server or time with the minimum value is chosen.

Table 3: Simulation Parameters Values for all Methods

Parameters	*Value / Range of Values*
X	N(10, 50) & U(0, 1)
No. of mobile nodes	1000
N	{3, 5, 10}
P for p-model	0.8
R	{0, 0.25, 0.5, 1}
θ	{30, 40, 50, 60} {0.3, 0.4, 0.5, 0.6} {20, 30, 40, 50, 60}
C	{1, 2, 3, 4, 5, 20, 30} {0.1, 0.2, 0.3, 0.4} {1, 10, 15, 20, 30, 40}

The reasons for adopting this approach are as follows: Primarily, this research emphasizes data decision-making and task offloading. Additionally, deep learning algorithms inherently differ from traditional algorithms, especially when the decision maker lacks complete information. Thus, the approach to optimality is the main analysis for evaluating these algorithms. Optimization is suitable when all server information is available to the decision maker, facilitating the mobile node in determining the ideal offloading location. Ultimately, these algorithms are implemented in sequence, complicating direct comparisons with other algorithms.

In this setting, in the absence of offloading rules, the mobile node will likely choose the first available mobile edge computing server. For edge computing load, such an offloading method is optimal for task offloading. So, the p-model method is utilized as a fallback technique. In the p-model, each server is assigned a loading probability, set to *p=0.8*. During each user move, each server has a probability *p=0.8* of being selected to load the job. In this experiment, increasing *p* intensively the probability of selecting the first server for loading. Consequently, the p-model replicates the scenario where the mobile node chooses the nearest servers that is closest edge servers due to the higher probability *p=0.8*.

When evaluating the actual dataset, if a server is preferred (server is chosen for loading) the process stops; if no server is preferred, the last server is chosen. A server is randomly preferred for each user to offload the work in the random selection model.

The results of all models are compared with values obtained from the proposed model, where the server with the shortest processing time is chosen for each unloading session. Models that are closer to the optimal value demonstrate superior performance in offloading decisions. The optimal

model is achieved by choosing the server with the shortest processing time for each load sequence.

4-3- Results

The simulation results evaluate the performance of the proposed DBN-based offloading model across multiple dimensions, including execution time, server usage, energy efficiency, and successful offloads. The evaluation spans three distinct probability distributions for the processing time variable X: normal, uniform, and exponential. Each distribution reflects different real-world workload scenarios in mobile edge computing environments.

Across all simulations, the DBN-based model consistently demonstrates superior performance compared to benchmark algorithms such as Delay Tolerant Offloading (DTO), Best Choice Problem (BCP), Cost-based Optimal Task (COT), Quality-Aware Odds, Random selection, and the p-model. The proposed method achieves lower average execution times, reduced CPU usage, and higher rates of successful offloads under varying resource constraints.

Figures 5 through 13 present comparative results for each distribution scenario. These include average processing times, server utilization, and the number of effective offloads under different CPU thresholds. The DBN model shows strong alignment with the optimal model, particularly in scenarios where resource availability is dynamic and unpredictable. This confirms the model's ability to make accurate offloading decisions and maintain system efficiency under diverse conditions.

Performance Analysis with Normal Distribution

As illustrated in Figure 5, when the processing time X follows a normal distribution, the proposed DBN-based algorithm achieves the shortest execution time among all evaluated methods. The average execution time for computational discharge is approximately 40 milliseconds, outperforming DTO, BCP, COT, and the p-model algorithms.

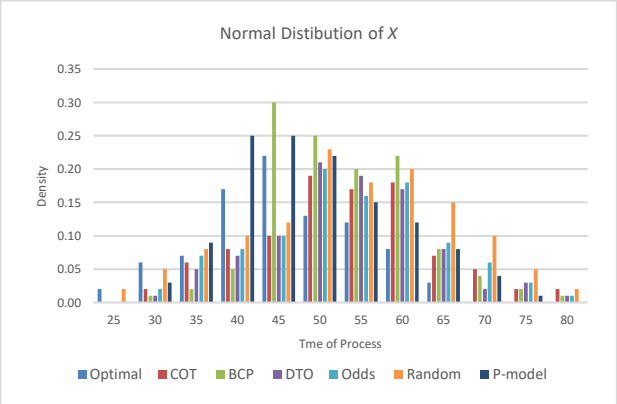

Fig. 5 Simulation Results for All Models in Case of *X* Normal Distribution.

The figure also reveals a significant overlap between the DBN model and the optimal model, indicating that the DBN's predictions closely approximate ideal offloading decisions. In contrast, models such as the p-model and random selection exhibit higher variance and longer processing times. The BCP model achieves a processing time of 46 milliseconds, which is lower than the p-model and random approaches but still less efficient than the DBN. These results validate the effectiveness of the DBN-based offloading strategy in minimizing latency and optimizing resource allocation in mobile edge computing. The model's ability to learn from historical workload patterns and predict response times contributes to its superior performance across varying conditions.

The results in Figure 6 reveal that the variation between the optimal model and DBN model is significantly smaller than the variation detected with other models. Notably, for models other than the DBN, the optimal threshold for each experiment k is generally close to the average processing time of 50 milliseconds. For example, in the DTO model and COT model, the thresholds generated for n=5 are {40, 42, 43, 46, 50}, all near the average processing time.

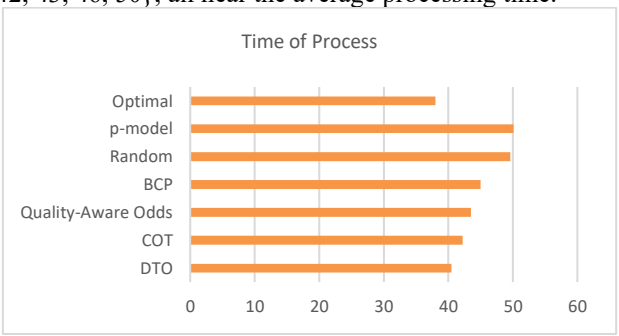

Fig. 6 Average Processing Time for Different Models with *X* Normal Distribution.

Using these optimal thresholds as a reference, the initial threshold value for the Odds method is set to 50, with performance evaluated for various values. The results, indicate the effective performance of the Odds method. This performance can be credited to the high likelihood of choosing a server with a processing time under 50 milliseconds. Thus, by setting a threshold value close to the average processing time, a shorter processing time is achieved for unloading the computational load.

Furthermore, the results demonstrate better performance for the BCP method compared to the p-models and Random method. The BCP evacuation policy is more likely to achieve the shortest processing time, leading to a lower average processing time than other models. This increased likelihood results in a lower expected processing time compared to the random and p models

Significantly, while the probability of selecting the best server is assumed to be similar in the BCP and Odds models, the defined threshold in the Odds model enhances performance by ensuring quality-aware decisions when

examining mobile edge computing servers. The main conclusion from these results is that the proposed method, referred to as the optimal model, achieves a shorter processing time than other methods, thereby reducing response time and improving the performance of computational offloading in cloud computing.

Performance Analysis with Uniform Distribution

In the initial results, the random variable X followed a normal distribution. To achieve more accurate findings, we conducted an additional simulation with X uniformly distributed within the interval *[0-1] (Figure 7)*. This range represents server usage, such as CPU utilization, where a value of 0.5 indicates 50% CPU usage. We applied similar steps to all models, as in previous experiments.

In the DTO model, the delay coefficient initially began at $r=0$, with results for other r values presented subsequently. For the cost-based optimal task model, an ideal threshold was identified for each cost value in the second set. Specifically, for c = 0.2, evaluations determined the optimal threshold to be 0.3. The cost interpretation is similar to the normal distribution scenario: a higher cost (smaller threshold V) signifies a greater need for shorter processing times.

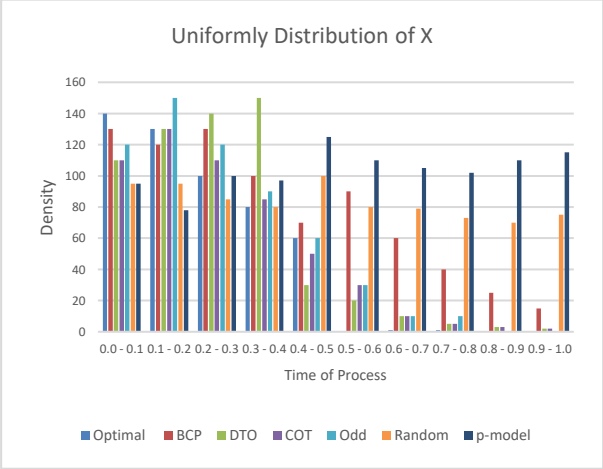

Fig. 7 Simulation Results for All Models in Case of X Uniformly Distribution.

In the quality-aware Odds model, the threshold was set to 0.5, yielding a 42% probability of selecting a server with $X=0.5$. Though the BCP model shares this probability, setting the threshold notably improved the Odds model's performance. Figures 7 and 8 show that model performance aligns closely with results from the normal distribution scenario. DTO and COT models remain top performers, with deep belief network-based models coming closer to optimality compared to random and p models.

As illustrated in Figure 8, the average execution time for various algorithms, including the proposed method based on the deep belief network, has been evaluated. The results demonstrate that the proposed method achieves a shorter execution time compared to other methods, indicating a more efficient response to computational offloading in mobile edge computing.

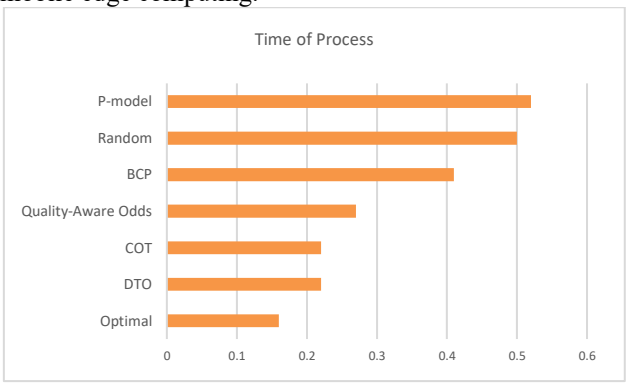

Fig. 8 Average Processing Time for Different Models with Uniform X Distribution.

Performance Analysis with Exponential Distribution

Figure 8 demonstrates that the proposed algorithm achieves an execution time of approximately 0.15 milliseconds, which is shorter compared to other methods. On the other hand, the p-model algorithm exhibits the longest execution time due to the consideration of a threshold value for selecting servers. These results suggest that the deep belief network (DBN) method provides superior response times for computational offloading in mobile edge computing, attributed to its layered approach.

Besides normal and uniform distributions, this experiment also included an exponential distribution with a mean of 50. The same procedural steps were followed as in the previous distributions. Initially, the delay coefficient in the DTO method was set to $r=0$, with results for other r values subsequently presented. The results under these conditions are shown in Figures 9.

In the Cost-based Optimal Task model, the figures depict the optimal threshold values V corresponding to each cost value. For this simulation, the cost was initially set to 20, with the optimal threshold determined to be 45.81, resulting in the lowest simulated expectation of X among other values. Performance across various cost values is also demonstrated. The cost interpretation aligns with scenarios where X follows normal and uniform distributions: a higher cost (smaller threshold V) indicates an increased demand for shorter processing times.

In the quality-aware Odds method, the threshold was set to 50, resulting in a 44% probability of selecting a server with X=50. The results in Figures 9 and 10 indicate that the proposed model's performance is consistent with the results obtained when X follows normal and uniform distributions. The DBN-based method consistently outperforms other algorithms, demonstrating the best performance and closest proximity to optimality compared to the random and p-models.

Figure 9 demonstrates that the proposed method with exponential distribution achieves a lower execution time compared to other methods. This distribution effectively guides server selection for mobile edge calculations, showing that the deep belief network-based method provides a faster response for computational offloading in mobile edge computing than other algorithms.

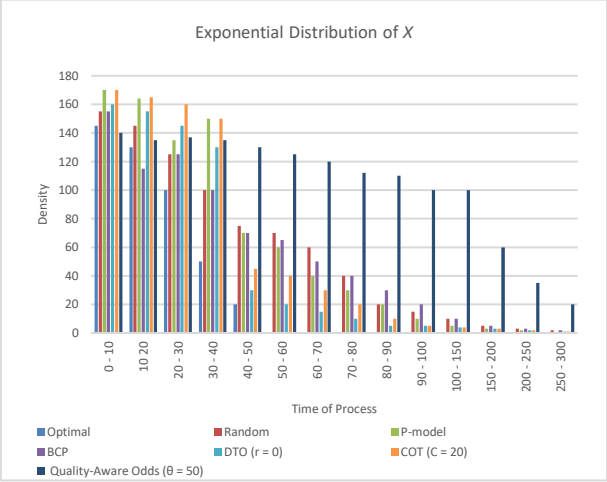

Fig. 9 Simulation Results for All Models in Case of *X* Exponential Distribution.

Figure 10 illustrates the average response time for different methods with exponential distribution. The proposed method has a significantly lower response time, approximately 10 milliseconds, compared to other algorithms. This demonstrates that the proposed method surpasses other approaches in reducing response time for computational offloading in mobile edge computing.

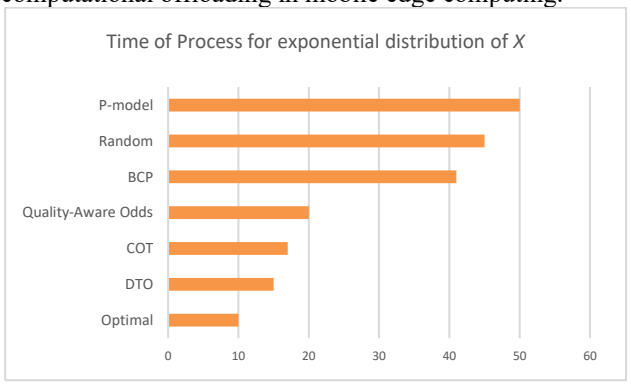

Fig. 10 Average Processing Time for Different Models With X Exponential Distribution.

Server Usage and Energy Efficiency

Figure 11 illustrates the average server usage recommended by each model. The DTO and COT models show results closest to the proposed method, with DTO performing better than the others by an absolute difference of 23 units compared to the proposed method. The findings indicate that the proposed method has a lower average server consumption than the other methods, meaning it consumes less energy for mobile edge calculations.

Additionally, the proposed method, based on the deep belief network, demonstrates a shorter average unloading time compared to other algorithms. Consequently, this suggests that the response time for computational offloading in mobile edge computing is more efficient with the proposed method than with others.

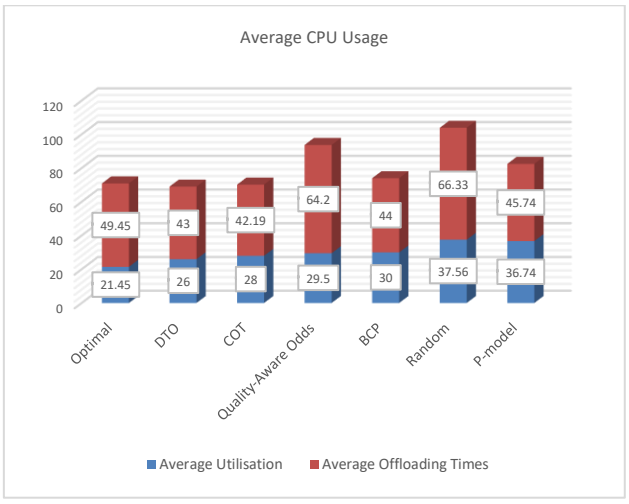

Fig. 11 Average CPU Usage and Average Computational Drain Time by each Model

Server Consumption and Successful Offloads

Figure 12 illustrates the average server consumption for the proposed method compared to other solutions. The proposed deep belief network method demonstrates a lower average server consumption, indicating that it not only reduces the response time for computational offloading but also optimizes server usage. This results in lower overall server consumption compared to other algorithms.

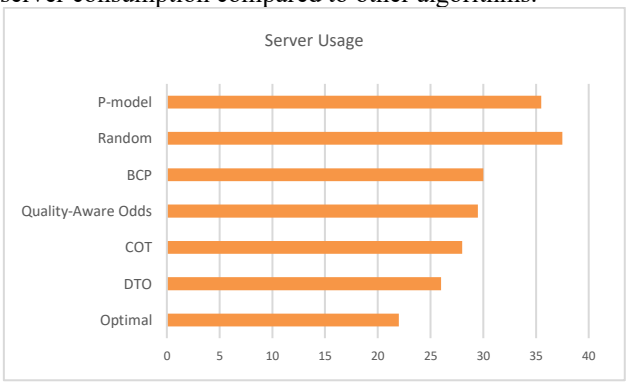

Fig. 12 Average Usage of Servers for Different Algorithms.

Result presented the average server consumption for the proposed method compared to other solutions. The proposed deep belief network method demonstrates lower average server consumption, indicating that it not only reduces response time for computational offloading but also

optimizes server usage, resulting in lower overall server consumption compared to other algorithms.

Beyond average server utilization, we compare performance based on the number of effective offloads for each model. An effective offload refers to unloading decisions that meet specific requirements set by each model. To assess this, we assume three different mobile edge computing programs (x, y, and z) each with distinct needs. For example:

- Program x requires less than 10% CPU utilization.
- Program y requires less than 20% CPU utilization.
- Program z requires a server with less than 30% CPU utilization.

If an offload occurs for a server with usage less than 10%, it is considered a successful offload for program x.

Figure 13 illustrates the effective offloads for various resource demands across entire methods. The proposed deep belief network-based method achieves the highest number of successful offloads in these three cases, with values of 102, 463, and 1887 successful offloads, respectively.

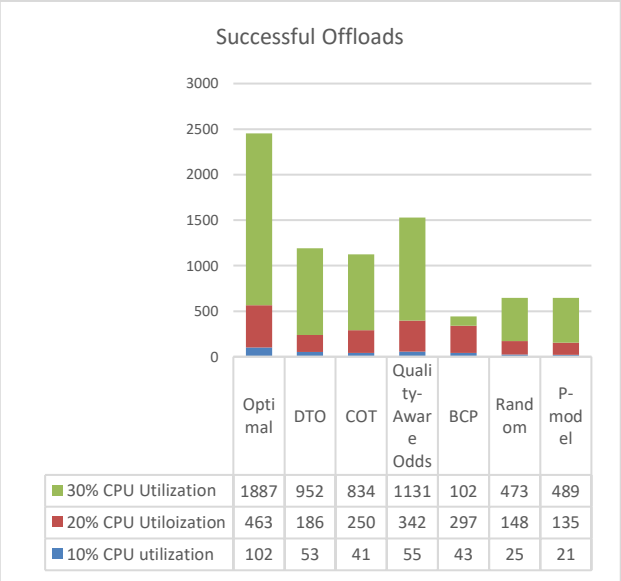

Fig. 13 Number of Effective Discharges for each Model Based on Various Threshold Values.

4-4- Discussion

The simulation results for various methods indicate that the presented models generally exhibit a time complexity of *O(n)* at worst, both in terms of time and space. If each model's condition is met on server number n, the mobile node will visit server n. For the DTO, COT, and Quality-aware Odds models, a pre-observation step involves generating thresholds. This step is presumed to be executed a single time by the service provider, external to the mobile node, although it can be implemented within the mobile node if necessary. For example, computing the threshold at the mobile node in the Odds and DTO methods requires *O(n)* time complexity. The COT method requires more time to calculate the threshold, depending on the likelihood distribution. Merely a sole operation is essential for a (uniform) distribution, while a normal distribution requires integration estimation with a time complexity no greater than $O(n^2)$.

Regarding space complexity, the BCP model does not require additional space for data storage, resulting in a space complexity of *O(n)*. This also applies to other models, provided the training step is performed outside the mobile node. If the training step is conducted locally at the mobile node, only the probability distribution parameters need to be stored. For a uniformly distributed *X,* the maximum and minimum values are stored, while for exponentially distributed *X*, the μ mean and σ^2 standard deviation are required. Previous results showed that the time complexity of the proposed method based on a deep belief network (DBN) is *O(1)*, the lowest complexity for predicting time and improving computational offloading performance in mobile edge computing.

Analyzing the execution time and server consumption across different algorithms reveals that the proposed method is more efficient in performing the computational offloading process. The results indicate that the proposed model is completely independent and lightweight for implementation in the mobile node, outperforming other compared solutions. The DBN-based method requires less processing time for computational offloading and task execution, with lower CPU consumption than other solutions. This makes it suitable for managing computational offloading of resources, compressing, or delaying limited tasks.

A practical scenario that highlights the effectiveness of the proposed DBN-based offloading mechanism involves a mobile user engaged in augmented reality (AR) navigation within a smart city. AR applications are latency-sensitive and require rapid processing of environmental data, user location, and graphical overlays. In such a context, the DBN model predicts the response times of available fog and cloud nodes based on historical workload patterns and real-time system conditions. By selecting the node with the lowest predicted latency, the system ensures that AR content is rendered and delivered with minimal delay, thereby preserving user experience and application responsiveness. In cases where no optimal node is identified, the fallback mechanism ensures continuity by probabilistically selecting a viable server. This dynamic and adaptive offloading strategy demonstrates the model's potential to support real-time, resource-intensive mobile applications in complex urban environments.

5- Conclusion

The principal aim of this research is to enhance computational offloading performance in mobile edge computing. To achieve this, we have employed a computational analysis method based on the deep belief network (DBN), incorporating various deep learning features to improve the evacuation process. By adding specific steps to the computational evacuation process, we aim to reduce server consumption, increase process speed, and decrease response time to computational requests.

In this study, the deep belief network algorithm has been utilized to further optimize computational offloading, making it suitable for various cloud computing applications, including mobile edge computing. The proposed algorithm focuses on reducing execution time for requests and increasing the number of successful offloads within the mobile edge computing system. By combining different distribution functions and the core features of the DBN algorithm, our method seeks to enhance efficiency and the volume of computational offloading.

Our approach to computational offloading on the server side is designed to provide a solution with low response time, ultimately reducing time complexity and energy consumption. It is crucial to employ the appropriate method to perform this process efficiently. Incorrect algorithms for computational offloading in cloud computing can lead to increased energy consumption and decreased successful offloads. Timely offloading reduces server-side energy consumption and increases efficiency, highlighting the importance of an accurate response time prediction solution to improve computational offloading performance in mobile edge computing.

A detailed examination of our results indicates that the proposed algorithm effectively improves computational offloading in mobile edge computing. This algorithm requires less time to execute offloading processes and respond to requests from mobile nodes. The number of requests handled by the servers does not increase response time, thereby reducing the duration of computational offloading. Compared to Delay Tolerant Offloading (DTO), Best Choice Problem (BCP), Cost-based Optimal Task (COT), and p-model algorithms, our method demonstrates shorter average processing times for computational offloading and request responses, achieving optimal results for the evaluated dataset. The proposed method outperforms other methods in terms of time complexity, energy consumption, processing time, CPU usage, average offload time, and the number of successful offloads.

While the proposed algorithm sometimes exhibits longer processing times for specific requests, overall performance in processing time, resource utilization, average server usage, successful offloads, and computational offload time is superior in improving computational offloading in mobile edge computing. By balancing accuracy and speed, our method effectively reduces response time and increases the number of successful offloads.

Future research should evaluate the proposed method across various cloud computing systems, applications, and datasets to fully explore its efficiency and applicability. Additionally, further studies can investigate other neural network algorithms, such as long short-term memory and convolutional neural networks, to enhance offloading performance in mobile edge computing. Meta-heuristic algorithms may also be considered to address the NP-hard nature of computational offloading problems, aiming to reduce complexity and increase successful offloads. Finally, developing solutions that require minimal processing and computing resources, while considering available resource consumption, will lead to more efficient computational offloading and increased successful offloads.

References

[1] A. Mahdavi and A. Ghaffari, “Embedding Virtual Machines in Cloud Computing based on Big Bang–Big Crunch Algorithm,” *Journal of Information Systems & Telecommunication (JIST)*, p. 305, 2019.

[2] N. Fernando, S. W. Loke, and W. Rahayu, “Mobile cloud computing: A survey,” *Future Generation Computer Systems*, vol. 29, no. 1, pp. 84–106, Jan. 2013, doi: 10.1016/J.FUTURE.2012.05.023.

[3] Md. G. R. Alam, M. M. Hassan, Md. Z. Uddin, A. S. Almogren, and G. Fortino, “Autonomic computation offloading in mobile edge for IoT applications,” *Future Gener. Comput. Syst.*, vol. 90, pp. 149–157, 2019, [Online]. Available: https://api.semanticscholar.org/CorpusID:52899499

[4] P. Boopathy *et al.*, “Deep learning for intelligent demand response and smart grids: A comprehensive survey,” *Comput Sci Rev*, vol. 51, p. 100617, Feb. 2024, doi: 10.1016/J.COSREV.2024.100617.

[5] I. Abdullaev, N. Prodanova, K. A. Bhaskar, E. L. Lydia, S. Kadry, and J. Kim, “Task Offloading and Resource Allocation in IoT Based Mobile Edge Computing Using Deep Learning,” *Computers, Materials and Continua*, vol. 76, no. 2, pp. 1463–1477, Aug. 2023, doi: 10.32604/CMC.2023.038417.

[6] H. Naseri, S. Azizi, and A. Abdollahpouri, “BSFS: A Bidirectional Search Algorithm for Flow Scheduling in Cloud Data Centers,” *Journal of Information Systems and Telecommunication (JIST)*, vol. 3, no. 27, p. 175, 2020.

[7] D. Seddiki, F. J. Maldonado Carrascosa, S. García Galán, M. Valverde Ibáñez, T. Marciniak, and N. Ruiz Reyes, “Enhanced virtual machine migration for energy sustainability optimization in cloud computing through knowledge acquisition,” *Computers and Electrical Engineering*, vol. 119, p. 109506, Oct. 2024, doi: 10.1016/J.COMPELECENG.2024.109506.

[8] L.-D. Chou, H.-F. Chen, F.-H. Tseng, H.-C. Chao, and Y.-J. Chang, “DPRA: Dynamic Power-Saving Resource Allocation for Cloud Data Center Using Particle Swarm Optimization,” *IEEE Syst J*, vol. 12, no. 2, pp. 1554–1565, 2018, doi: 10.1109/JSYST.2016.2596299.

[9] H. Wang, S. Cao, H. Li, L. Yan, Z. Guo, and Y. Gao, "Multi-objective joint optimization of task offloading based on MADRL in internet of things assisted by satellite networks," *Computer Networks*, vol. 254, p. 110801, Dec. 2024, doi: 10.1016/J.COMNET.2024.110801.

[10] T. Tsokov and H. Kostadinov, "Dynamic network-aware container allocation in Cloud/Fog computing with mobile nodes," *Internet of Things*, vol. 26, p. 101211, Jul. 2024, doi: 10.1016/J.IOT.2024.101211.

[11] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-Efficient Resource Allocation for Mobile-Edge Computation Offloading," *IEEE Trans Wirel Commun*, vol. 16, no. 3, pp. 1397–1411, 2017, doi: 10.1109/TWC.2016.2633522.

[12] Q. Wang, S. Guo, J. Liu, and Y. Yang, "Energy-efficient computation offloading and resource allocation for delay-sensitive mobile edge computing," *Sustainable Computing: Informatics and Systems*, vol. 21, pp. 154–164, Mar. 2019, doi: 10.1016/J.SUSCOM.2019.01.007.

[13] S. C. Ghoshal *et al.*, "VESBELT: An energy-efficient and low-latency aware task offloading in Maritime Internet-of-Things networks using ensemble neural networks," *Future Generation Computer Systems*, vol. 161, pp. 572–585, Dec. 2024, doi: 10.1016/J.FUTURE.2024.07.034.

[14] S. Yang, D. Kwon, H. Yi, Y. Cho, Y. Kwon, and Y. Paek, "Techniques to Minimize State Transfer Costs for Dynamic Execution Offloading in Mobile Cloud Computing," *IEEE Trans Mob Comput*, vol. 13, no. 11, pp. 2648–2660, 2014, doi: 10.1109/TMC.2014.2307293.

[15] X. Xu, Q. Huang, X. Yin, M. Abbasi, M. R. Khosravi, and L. Qi, "Intelligent Offloading for Collaborative Smart City Services in Edge Computing," *IEEE Internet Things J*, vol. 7, no. 9, pp. 7919–7927, Sep. 2020, doi: 10.1109/JIOT.2020.3000871.

[16] T. Tang, C. Li, and F. Liu, "Collaborative cloud-edge-end task offloading with task dependency based on deep reinforcement learning," *Comput Commun*, vol. 209, pp. 78–90, Sep. 2023, doi: 10.1016/J.COMCOM.2023.06.021.

[17] Y. Miao, G. Wu, M. Li, A. Ghoneim, M. Al-Rakhami, and M. S. Hossain, "Intelligent task prediction and computation offloading based on mobile-edge cloud computing," *Future Generation Computer Systems*, vol. 102, pp. 925–931, Jan. 2020, doi: 10.1016/J.FUTURE.2019.09.035.

[18] M. Du, Y. Wang, K. Ye, and C. Xu, "Algorithmics of Cost-Driven Computation Offloading in the Edge-Cloud Environment," *IEEE Transactions on Computers*, vol. 69, no. 10, pp. 1519–1532, 2020, doi: 10.1109/TC.2020.2976996.

[19] L. Tan, Z. Kuang, J. Gao, and L. Zhao, "Energy-Efficient Collaborative Multi-Access Edge Computing via Deep Reinforcement Learning," *IEEE Trans Industr Inform*, vol. 19, no. 6, pp. 7689–7699, Jun. 2023, doi: 10.1109/TII.2022.3213603.

[20] A. Shakarami, A. Shahidinejad, and M. Ghobaei-Arani, "An autonomous computation offloading strategy in Mobile Edge Computing: A deep learning-based hybrid approach," *Journal of Network and Computer Applications*, vol. 178, p. 102974, Mar. 2021, doi: 10.1016/J.JNCA.2021.102974.

[21] S. Zhong, S. Guo, H. Yu, and Q. Wang, "Cooperative service caching and computation offloading in multi-access edge computing," *Computer Networks*, vol. 189, p. 107916, Apr. 2021, doi: 10.1016/J.COMNET.2021.107916.

[22] G. Peng, H. Wu, H. Wu, and K. Wolter, "Constrained Multiobjective Optimization for IoT-Enabled Computation Offloading in Collaborative Edge and Cloud Computing," *IEEE Internet Things J*, vol. 8, no. 17, pp. 13723–13736, 2021, doi: 10.1109/JIOT.2021.3067732.

[23] Z. N. Samani and M. R. Khayyambashi, "Reliable resource allocation and fault tolerance in mobile cloud computing," *Journal of Information Systems and Telecommunication (JIST)*, vol. 7, no. 2, pp. 96–109, 2019.

[24] J. Long, Y. Luo, X. Zhu, E. Luo, and M. Huang, "Computation offloading through mobile vehicles in IoT-edge-cloud network," *EURASIP J Wirel Commun Netw*, vol. 2020, no. 1, p. 244, 2020, doi: 10.1186/s13638-020-01848-5.

[25] L. Bracciale, M. Bonola, P. Loreti, G. Bianchi, R. Amici, and A. Rabuffi, "CRAWDAD dataset roma/taxi (v. 2014-07-17)," 2014.

Journal of Information Systems and Telecommunication
Vol.13, No.3, July-September 2025, 256-265
http://jist.acecr.org
ISSN 2322-1437 / EISSN:2345-2773

PSO-Optimized Power Allocation in NOMA-QAM for Beyond 5G: A CFD and MFD Analysis

Jaspreet Kaur[1*]

[1]. Institute of Engineering and Technology, Lucknow

Received: 20 Sep 2024/ Revised: 04 Sep 2025/ Accepted: 05 Oct 2025

Abstract

This paper proposes a power allocation method based on particle swarm optimization (PSO) to enhance spectrum sensing performance in downlink Non Orthogonal Multiple Access (NOMA) systems employing high-order Quadrature Amplitude modulation (QAM) modulation for beyond 5G networks**.** By intelligently adjusting user power levels, the proposed approach significantly improves detection reliability while maintaining stringent false alarm constraints, even under challenging low-SNR conditions. The goal is to enhance spectrum sensing performance by maximizing the probability of detection (P_d) while maintaining a constrained probability of false alarm (P_f). Cyclostationary Feature Detection (CFD) and Matched Filter Detection (MFD) techniques are applied to evaluate detection performance under varying Signal to noise ratio (SNR) conditions. Simulation results demonstrate that the optimized framework not only strengthens detection performance particularly for high order QAM but also enhances overall system responsiveness. Also CFD surpasses MFD in higher SNR scenarios due to its ability to exploit cyclic features of modulated signals, which are preserved even in moderately noisy environments. The integration of PSO further enhances system performance, offering a practical and scalable solution for next-generation Internet of Things (IoT)-enabled spectrum sharing environments.

Keywords: Non Orthogonal Multiple Access (NOMA); Matched Filter Detection (MFD); CFD, PSO; Cognitive Radio Networks (CRN); Next Generation Networks (NGN).

1- Introduction

The increase in the number of connected devices and the rapid expansion of wireless services are creating an unprecedented need for spectral resources, pushing networks toward the capabilities envisioned for beyond 5G and 6G systems [1]. Because cognitive radio (CR) technology allows for dynamic spectrum access and opportunistic usage of unused frequency bands, it has become a key paradigm to solve spectrum shortages [2]. NOMA has simultaneously become well-known as a crucial method for enhancing spectral efficiency and facilitating huge connections [3-4]. CR employs three primary sensing methods to detect available spectrum: Energy Detection (ED), Matched Filter Detection (MFD), and Cyclostationary Feature Detection (CFD). It has been found in recent surveys that over 75% of spectrum is wasteful [4]. Therefore, it is crucial to make use of unutilized spectrum. Primary users (PUs) possessing license do not always use the allocated spectrum, causing spectrum to be wasted. Assigning spectrum to unlicensed users, frequently referred to as secondary users or SU, is one method of increasing spectrum utilization when PUs are discovered to be inactive [5]. Simultaneously, the spectrum ought to be redistributed to the PUs whenever they choose to utilize it, without affecting the SU's performance [6]. This implies that SUs should use the spectrum whether or not PUs are present. There is great potential for attaining high data rates and effective spectrum usage when CR and NOMA are combined, especially when using high order modulation techniques like 64-QAM and 256-QAM [7-8]. These benefits, however, come at the expense of more complicated spectrum sensing and a greater susceptibility to fading and noise, particularly in the low signal-to-noise ratio (SNR) conditions typical of CR situations [9]. For secondary users to operate dependably in shared spectrum scenarios and to prevent detrimental interference with primary users, accurate spectrum sensing is necessary [10]. This study addresses the central question of whether an intelligent power allocation strategy can enhance spectrum sensing performance in CR-enabled NOMA

✉ **Jaspreet Kaur**
jaspreetsweetangel@gmail.com

systems while maintaining strict constraints on false alarm rates. We hypothesize that a Particle Swarm Optimization (PSO)-based approach can dynamically allocate user power in a manner that maximizes detection probability, reduces sensing time, and maintains efficient spectrum utilization even under challenging conditions. Conventional sensing techniques, including CFD and MFD, often exhibit degraded performance in low SNR conditions, particularly when dealing with high-order modulations [11-12]. Moreover, many existing studies focus solely on detection algorithms without considering adaptive resource allocation as part of the sensing framework. Our work bridges this gap by integrating PSO-based power optimization into the CR-NOMA sensing process, offering a holistic solution that jointly considers sensing accuracy and power efficiency. This represents a substantial contribution toward enabling practical, robust CR-NOMA implementations. The motivation for this research lies in the growing demand for agile and energy-efficient spectrum sharing techniques capable of supporting high-throughput applications, Internet of Things (IoT) deployments, and massive machine-type communications. By optimizing power allocation, we aim to achieve reliable detection performance without excessive sensing overhead, paving the way for practical deployment of cognitive radio systems in next-generation networks. Motivated by the need for improved detection in noisy NOMA-QAM environments, this work proposes a PSO-based power allocation framework to enhance spectrum sensing performance. Key contributions include:

(i) Development of a PSO-optimized power allocation scheme for NOMA systems with high-order QAM to boost detection accuracy.
(ii) Comparative analysis of CFD and MFD for QAM-64 and QAM-256 modulation schemes.
(iii) Simulation results showing up to 47.91% improvement in detection probability (Pd) over conventional MFD, validating the approach in challenging noise conditions.
This is how the rest of the paper is structured. Relevant literature related to NOMA, QAM, MFD, CFD and PSO is given in Section 2. The system model and the suggested PSO-based optimization methodology are covered in depth in Section 3. Simulation data, performance comparisons, and information on the efficacy of the suggested strategy are presented in Section 4. The paper's conclusion and some future study directions are covered in Section 5 and 6.

2- Literature Review

Lately, a number of research on spectrum sensing techniques using NOMA have demonstrated potential in fulfilling the spectrum needs of several 5G applications. 5G mobile communications are about to become worldwide. For an OFDM system, cyclic prefix detection was proposed by Arun et al. [13]. The recommended method's demand for previous knowledge from the principal user is one of its key drawbacks. The energy detection method of SS for OFDM system was implemented by the authors [14]. The simulation results show that while OFDM without CP performs better towards P_f, OFDM system consisting of CP shows improved throughput performance. Recent studies further extended the applicability of NOMA-based cognitive systems [21-22]. Recent advancements in spectrum sharing and NOMA integration have focused on intelligent resource allocation and IRS-assisted systems to enhance performance in Beyond 5G networks [25-26]. Additionally, Bala Kumar and Nanda Kumar [28] explored block chain-enabled cooperative spectrum sensing in MIMO-NOMA CRNs for improved security and sensing accuracy. For instance, Salameh et al. [29] feature-based spectrum sensing to adaptively detect primary user signals in fading channels without requiring a fixed detection threshold while Zhai et al. [30] proposed a joint optimization scheme combining active IRS and multicluster NOMA to improve spectral efficiency. These works underscore a growing trend toward intelligent, adaptive spectrum management strategies. However, most of these approaches either focus on physical-layer improvements or overlook sensing complexity under high-order modulation and low-SNR conditions. In contrast, this study addresses the need for efficient spectrum sensing by integrating PSO-based power allocation with advanced detection techniques in high-QAM NOMA-CR systems. Detailed literature specifically for NOMA-QAM systems is given in Table 1.

Table 1 :- Literature Review relevant to proposed Work

S.No	Reference	Year	Aim	Findings
1	[15]	2010	Implement and examine a MIMO-OFDM system	Implementation and analysis done using MATLAB simulations
3	[4]	2019	Enhance sensor performance at low SNR	3 dB gain with optimized NOMA over O-NOMA
4	[1]	2019	Explore advanced spectral efficiency techniques in CRNs using NOMA and 5G signals.	NOMA-CRN outperforms conventional CR in spectrum efficiency
5	[3]	2020	To Integrate NOMA into CR networks to enhance spectrum efficiency and accommodate large number of users	High SE and large user support shown in CR scenarios
6	[22]	2021	Use NOMA to efficiently utilize the spectrum	Allows SU to use several PU types with and without interference
7	[24]	2021	To Assess the effectiveness of NOMA in uplink communications using fixed power coefficients.	Weak user power boost improves performance, especially at low SNRs
8	[27]	2021	Apply Swarm Intelligence to address future network issues	SI types classified; challenges and research opportunities discussed
9	[26]	2022	Detailed review of 5G waveforms using sensing methods	Cyclostationary methods show 2 dB advantage over traditional techniques
10	[28]	2024	Introduce block chain-enabled cooperative spectrum sensing for 5G/B5G CR using massive MIMO-NOMA	Demonstrated enhanced security and reliability in spectrum sensing using decentralized block chain mechanisms in MIMO-NOMA CRNs.
11	[29]	2025	Machine learning-driven, feature-based spectrum sensing approach to improve NOMA signal detection in dynamic IoT networks operating under fading channels.	Method Employs feature-based spectrum sensing to adaptively detect primary user signals in fading channels without requiring a fixed detection threshold.

2-1- Research Gap and Motivation

Despite the extensive efforts to enhance spectrum efficiency using CR and NOMA techniques, several challenges remain unaddressed. Most of the prior works focus on static or suboptimal power allocation strategies, often overlooking the impact of dynamic power tuning under high-order modulation schemes. Furthermore, few studies have explored the integration of advanced optimization algorithms such as swarm intelligence for real-time adaptation in CR-NOMA environments under low-SNR conditions. Additionally, limited work has been done to jointly optimize sensing accuracy and power distribution while accounting for false alarm constraints in high-QAM signal environments. As a result, a critical gap persists in developing unified frameworks that can adaptively optimize both detection performance and spectral efficiency in practical CR scenarios. Motivated by this gap, the present study proposes a novel power allocation framework based on Particle Swarm Optimization (PSO), tailored for CR-enabled NOMA systems operating under high-order QAM. The approach aims to achieve enhanced sensing accuracy, reduced false alarm rates, and optimized throughput, all while maintaining practical feasibility for next-generation wireless systems.

3- Proposed System Model

This work investigates a downlink NOMA-based communication system utilizing QAM modulation for Beyond 5G scenarios. Multiple users are multiplexed in the power domain and served concurrently over a shared channel. Power levels for each user are dynamically

allocated using Particle Swarm Optimization (PSO) to enhance overall detection performance while maintaining user fairness. At the receiver, spectrum sensing is carried out using both CFD and MFD, with performance evaluated across different SNR values for QAM-64 and QAM-256 schemes. The PSO algorithm optimizes power allocation by maximizing the P_d under a constraint on the $P_f \leq 0.5$. These methods help the CR identify when the spectrum is idle based on two hypotheses: H1(primary user presence) and H0(absence of a primary user).

Table 2.Comparison between traditional and proposed sensing technique

S. No.	Spectrum Sensing Technique	Remarks
1	Conventional Energy Detection	Simple to implement with low computational complexity. Poor performance at low SNR ($P_d = 0$ at SNR < -12 dB). Susceptible to interference be- tween PUs and SUs.
2	Conventional CFD	Robust detection at low SNR (Requires prior knowledge of signal periodicity). Moderate computational complexity due to autocorrelation.
3	Conventional MFD	Effective at low SNR ($P_d = 0.19$ at SNR = 4 dB for QAM-256). Requires prior knowledge of PU signal. SUs can only use spectrum in absence of PUs.
4	Proposed Optimized MFD & CFD	High P_d (0.83 at $P_f = 0.5$ for QAM-256, 47.91% improvement over MFD). Robust at low SNR ($P_d = 0.79$ at SNR = -5 dB). Increased computational complexity due to PSO optimization.

$$\begin{Bmatrix} H0 : \mathrm{Xj(t)} = \mathrm{Nj(t)} \\ \mathrm{H1: Xj(t)} = \mathrm{hjS(t)} + \mathrm{Nj(t)},\ \mathrm{j} = 1, \ldots\ldots\ldots \mathrm{Nu} \end{Bmatrix} \quad (1)$$

The fitness function is defined as:

$$\mathrm{F(P)} = P_d(\mathrm{P}) - \lambda . \max(0, P_f(\mathrm{P}) - 0.5) \quad (2)$$

where P is the power allocation vector, lambda is a penalty factor, and $P_d(P)$ and $P_f(P)$ are computed based on the NOMA-QAM system model. Although PSO is a widely established optimization technique, its characteristics make it particularly suitable for power allocation in dynamic CR-NOMA environments. PSO efficiently handles multi-objective, non-convex optimization problems without requiring gradient information, which is especially important under real-time, non-linear, and noisy conditions typical of cognitive radio systems. Moreover, PSO's low computational cost and adaptability enable quick convergence in environments where SNR and user demands fluctuate. This makes PSO a practical and effective choice for simultaneously optimizing detection probability and power distribution in high-QAM scenarios. The novelty of this work lies in embedding PSO within a joint spectrum sensing and power allocation framework, where the optimization process is directly influenced by detection metrics (P_d and P_f). This unique application is further distinguished by its evaluation under high-QAM and CFD/MFD trade-offs. Comparison of proposed model with benchmarking techniques is given in Table 2.

3-1- Matched Filter Detection

The MFD technique evaluates whether primary users are present by comparing the detected signal with a reference signal. The next step involves comparing the output with a dynamic threshold. It is extremely effective in low SNR since it optimizes SNR in presence of AWGN. The formula for the test statistic is TMF = ∑y (n)*x (n). The PU signal in this case is represented by (x), the SU signal by (n), and the test parameter for MFD is TMF. It then compares a threshold with the test statistics (TMF) to ascertain availability of spectrum. The signal received from Secondary and Primary user are roughly modeled as random Gaussian variables as depicted in fig. (1).

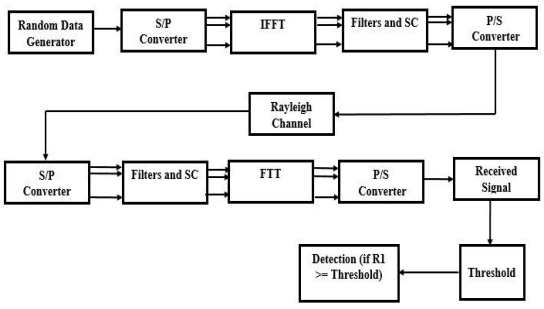

Figure 1. Block diagram for NOMA MFD

3-2- Cyclostationary Feature Detection

CFD is amongst the most significant technique for advanced as it is able to identify the spectrum at low SNR without the impact of noise. It uses signal's periodicity features as it calculates mean and autocorrelation of the signal. The spectrum correlation density functions and cyclic autocorrelation are useful in order to estimate the CS signals. The initial stage in CS is to use a number of procedures, including filtering, encoding, and sampling, to convert the signal into second-order CS.

$$\{y(+)\} = \{y\,(t + to)\} \quad (3)$$

The (r) is represented as cyclic auto-correlation function at:

$$\beta\gamma = \{M/To\} \quad (4)$$

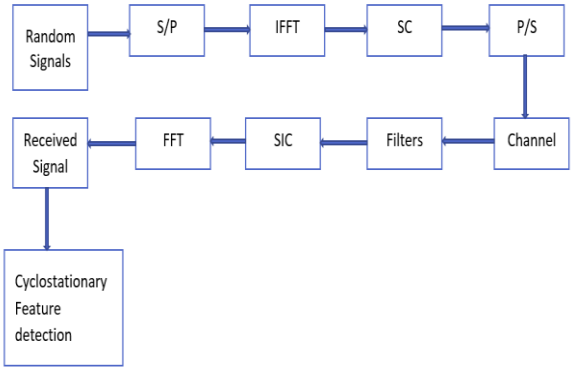

Figure 2. Block diagram for NOMA CFD

In a NOMA system, each subcarrier's power spectrum density (PSD) can be characterized. For n-th subcarrier, PSD can be represented as:

$$\varphi n(f) = PnTs\left(\frac{Sin\pi fTs}{\pi fTs}\right)^2 \quad (5)$$

where, T_s stands for the symbol duration, φ is the PSD of the next subcarrier, and P_n is transmit power that is released by preceding subcarrier. A possible technique to represent CFD using NOMA is as

$$\varphi n(f) = |Hn(f)|^2 \quad (6)$$

The prototype filter's frequency spectrum with coefficient h[n] and n = 0, 1... W-1 is represented as $H_n(f)$ [6]. An example of a frequency response's source is:

$$|Hn(f)|=h\left[\frac{W}{2}\right]+2\sum_{i=1}^{\frac{W}{2}-1} h\left[\left(\frac{W}{2}\right)\ 1\right]\cos(2\prod r) \quad (7)$$

The following formula determines the phase angle:

$$Ph(u) = [so^u, s1^u, s2^u \ldots\ldots\ sl-1^u] \quad (8)$$

for u=1, 2...U

$$sj^{(u)} = \exp\,(j\theta_0^{(u)}) \quad (9)$$

j=0, 1, L-1, and where $j\theta_0^{(u)}$ denotes random phase angle. So the representation of NOMA symbol can be shown as:

$$Y_k = \left[Y_{k,0}, Y_{k,1} \ldots\ldots\ldots\ldots\ldots\ldots\ldots Y_{k,l-1}\right] (10)$$

The phase angle is applied to the NOMA symbols as follows:

$$Y_k^{(v)} = p^{(u)} * Y_k \quad (11)$$

$$y^u(t)=\sum_{I=0}^{L-1}\sum_{K'=0}^{k-1} X_{k',I}^{(U_{min})}h(t-\frac{K'T}{2}))e^{\frac{j2\prod It}{T}}e^{j\theta}K'I+\sum_{I=0}^{L-1} d_{k,I}^{(u)}h(t-\frac{KT}{2}))e^{\frac{j2\prod It}{T}}e^{j\theta}K'I \quad (12)$$

Lastly, the following represents the received NOMA signal:

$$Y'(t)=\sum_{k=0}^{K-1} X_{k',l}^{(U_{min})}e^{j\theta k,I}\,h(t-k^{R_0}) \quad (13)$$

We can infer from Eq. (13) that the NOMA - CR system is capacious than traditional OFDM system. The block diagram of the recommended technique is displayed in Fig. 2. A sequential generation process generates a random parallel symbol. IFFT is used to examine the signal in the time domain, and once it has been transmitted across a Rayleigh channel, SC permits many users to use the sub-channel. The receiver uses SIC to decode the time domain signal and FFT to translate it to the frequency domain. In the end, a threshold is determined and if received symbol's energy exceeds the threshold value, identification will occur; otherwise, no detection will be taken into account.

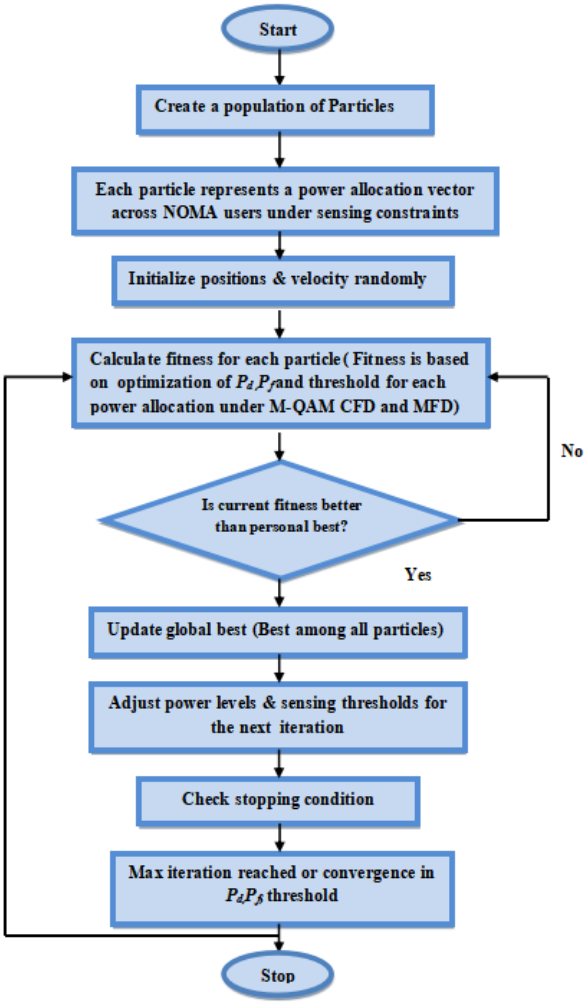

Figure 3. Flowchart of MFD and CFD Technique using PSO

4- Simulation Parameters and Performance Analysis.

In an effort to implement the suggested algorithm shown in Fig. 3 MATLAB 2022 is used. Table 3. depicts the simulation parameters for optimizing and analyzing NOMA QAM CFD and MFD using PSO. Simulation results of matched filter spectrum sensing method and Cyclostationary feature detection based on NOMA are used to comprehensively examine the results. This study determines the threshold value at the NOMA system's receiver end.

Table 3. Simulation Parameters

Parameters	Description	Values
f	frequency	16 MHz
M	QAM order	64,256
BW	Bandwidth	30 MHz
N	Number of users	50
n	Population size	100
SNR	Signal to noise ratio	-20dBto 5 dB
k	FFT Size	1024

It is based on the idea that only detection will be presumed if the signal received equals or exceeds the threshold value; otherwise, no detection will be inferred. When assessing the effectiveness of MFD and CFD, a constant threshold value is taken into account because a changing threshold can deteriorate the efficiency of spectrum sensing methods. To

investigate the role of thresholds in MFD and CFD identification, QAM-64 and QAM-256 transmission systems with 64 and 256 sub-carries were used. Table 4 and Figure 4 display the P_d for various P_f values. P_f indicates the false representation of noise as a desired signal. SNR = 10 dB was fixed in the current simulation to analyze the effectiveness of MFD & CFD strategy for NOMA. It is seen from fig.4 and table 4 that NOMA M-256 P_d is higher than M-64. So it is inferred that NOMA-QAM-MFD 256 Pd is better than QAM-64 as shown in fig (4).

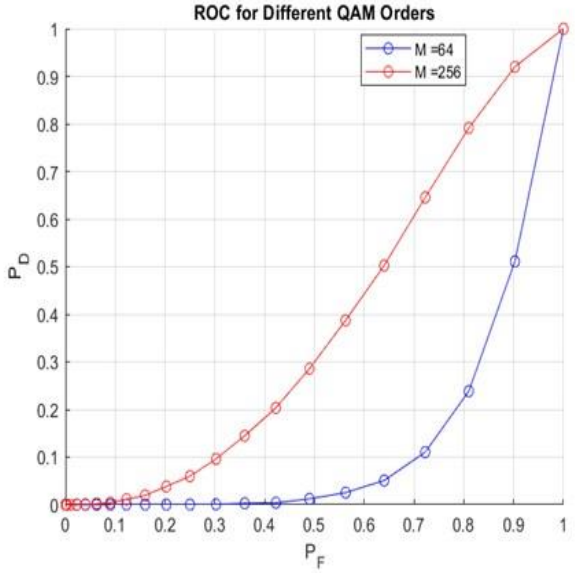

Figure 4: P_d Vs P_f for M-QAM MFD

Table 4: NOMA-QAM MFD P_d vs P_f result

P_f/P_d (MFD)	**0.1**	**0.2**	**0.3**	**0.4**	**0.6**	**0.7**	**0.8**	**0.9**	**1**
NOMA M-256	0	0	0	0.07	0.14	0.27	0.47	0.76	1
NOMA M-64	0	0	0	0.05	0.09	0.18	0.33	0.56	1

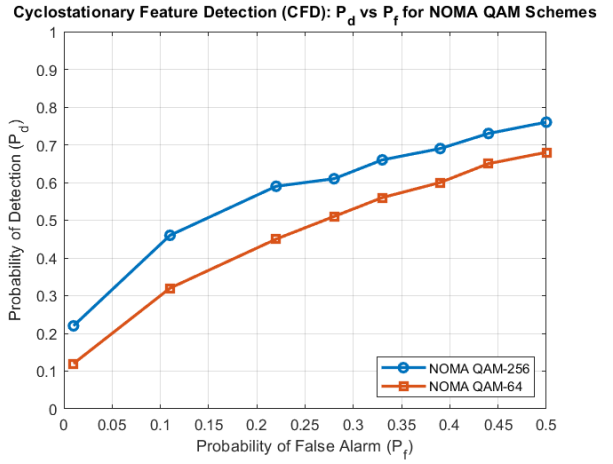

Figure.5. P_d Vs P_f for CFD for M-QAM.

Table 5: P_d vs P_f for NOMA-QAM using CFD

P_f /P_d (CFD)	**0.01**		**0.11**	**0.22**	**0.28**	**0.33**	**0.39**	**0.44**	**0.50**
NOMA QAM-256	0.22		0.46	0.59	0.61	0.66	0.69	0.73	0.76
NOMA QAM-64	0.12		0.32	0.45	0.51	0.56	0.60	0.65	0.68

Table 5 and Figure 5 shows the Pd vs Pf values for M-QAM CFD. A comparative analysis demonstrates the clear advantage of the proposed NOMA-CFD approach over MFD. At Pf = 0.5 and SNR = 10 dB, CFD with QAM-256 achieves a Pd of 0.76, outperforming both QAM-64 (Pd = 0.68) and MFD, with an observed 44.28% improvement in detection probability. Across the full range of Pf values, CFD consistently maintains higher Pd, indicating superior sensing reliability and robustness to false alarms compared to conventional techniques.

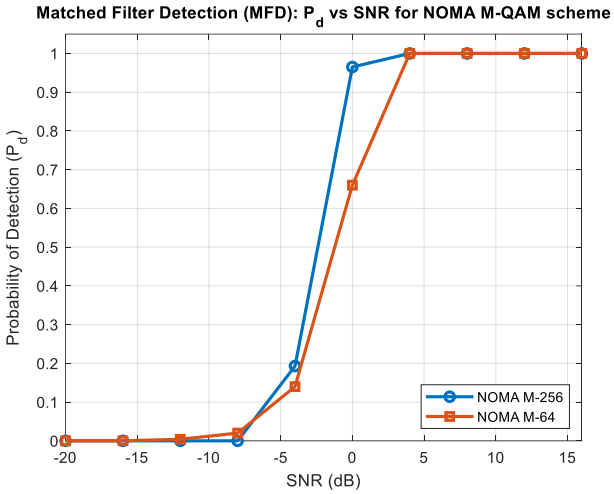

Figure 6. Plot for MFD P_d against SNR.

Table 6. P_d against SNR for MFD in NOMA-QAM

SNR/Pd (MFD)	**-20**	**-16**	**-12**	**-8**	**-4**	**0**	**4**	**8**	**12**	**16**
NOMA M-256	0	0	0	0	0.19	0.965	1	1	1	1
NOMA M-64	0	0	0.004	0.02	0.14	0.66	1	1	1	1

The Pd is displayed as a function of SNR in Table 6 and Fig.6. We do analysis and simulations across a variety of SNR values (10 dB to 20 dB) for MFD. For QAM-64 & 256, 100% Probability of detection (Pd) is achieved at 4 dB and 6 dB, respectively. Therefore, QAM-Pd can be considered better than QAM-256. For instance, at SNR = –10 dB, MFD yields a Pd of 0.56 (QAM-256), while CFD fails to detect (Pd ≈ 0). However, at SNR = 4 dB, CFD rapidly improves to Pd = 1.0, outperforming MFD's Pd of

0.97. This demonstrates CFD's steeper gain in detection performance once the SNR threshold is crossed.

Table 6 and Figure 6 shows the Pd for various Pf values. SNR = 10 dB was fixed in the current simulation to measure the effectiveness of the CFD strategy for NOMA. It is seen that for NOMA QAM CFD Pd value is 0.76 for Pf of 0.50 as compared to 0.68 Pd value for NOMA QAM-64. Also

Table7.P_d vs SNR for NOMA-QAM with CFD.

SNR(dB)/Pd	**-25**	**-20**	**-15**	**-10**		**-5**	**0**	**+5**
NOMA QAM-256	0.11	0.16	0.33	0.56		0.79	0.97	1
NOMA QAM-64	0.10	0.15	0.30	0.50		0.74	0.91	0.98

Table 8. BER vs SNR of NOMA-QAM MFD & CFD

P_f/P_d	0.01	0.06	0.1	0.15	0.2	0.25	0.3	0.4	0.50
Optimized P_d of MFD	0.33	0.37	0.39	0.40	0.42	0.43	0.45	0.47	0.49
Optimized P_d of CFD	0.51	0.59	0.63	0.70	0.73	0.75	0.79	0.81	0.83

results improve by 44.28% when compared with MFD technique. The figure illustrates that NOMA-QAM-256 P_d is better than QAM-64. Also it is clear from results that NOMA-CFD outperforms the results of MFD.

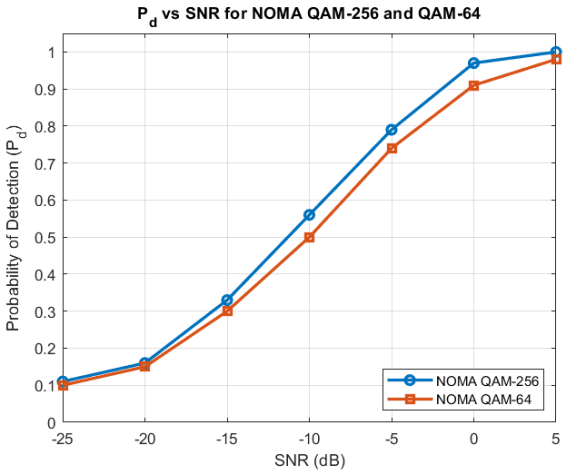

Figure.7. P_d Vs SNR for CFD.

The table 7 and Fig. 7 depicts results of P_d vs SNR of NOMA-QAM CFD. We examine and model P_d throughout a spectrum of SNR ranging from -25 to 5dB. From obtained results it is evident that at 0 dB and 5dB in the case of QAM-64 and QAM-256, P_d reaches an ideal value of 100%.Thus, it may be said that QAM- 64 Pd is superior to QAM-256's.The superior low-SNR performance of MFD is due to its reliance on known signal templates. In contrast, CFD requires stronger signals to detect Cyclostationary features but eventually surpasses MFD in higher-SNR regions, making it better suited for mid-to-high-SNR cognitive environments.

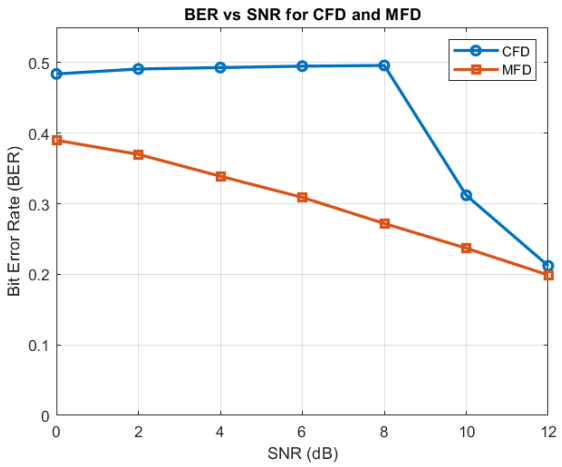

Figure 8. BER vs SNR of NOMA-QAM MFD & CFD

As SNR increases, the BER lowers, as Fig. 8 and Table 8 demonstrate. For M-256, a BER of 0.309 is obtained at 6 dB using the MFD technique and 0.212 at 12 dB using the CFD technique. Matched Filter Detection MFD consistently achieves lower BER compared to CFD across all SNR levels due to its reliance on known signal patterns. CFD shows limited improvement at low SNR but performs better as SNR increases beyond 10 dB. Overall, MFD is more reliable for low-SNR environments, while CFD requires stronger signals to reduce errors.

Figure 8 reinforces these findings, showing that MFD achieves a BER of 0.309 at 6 dB, while CFD only achieves 0.212 at 12 dB. This indicates that while MFD offers lower BER in noisy environments, CFD benefits more from clean conditions. As observed in Tables 5 and 7, P_d increases with SNR for both MFD and CFD. Notably, MFD achieves a P_d of 0.97 at 0 dB for QAM-256, while CFD reaches similar performance only at higher SNR levels (>4 dB). This indicates that MFD is more suitable for low-SNR environments due to its coherent detection mechanism.

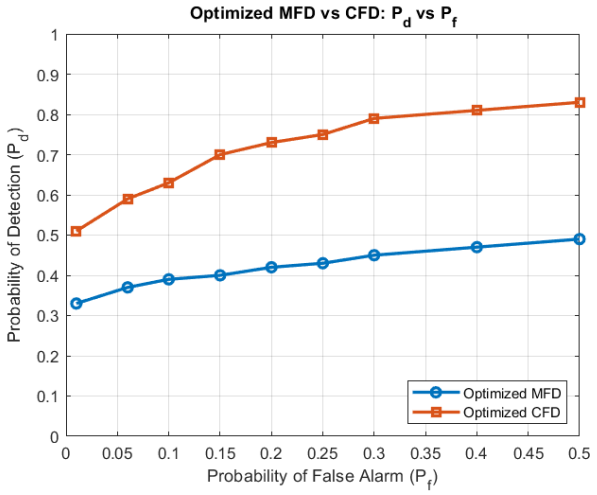

Figure 9. Optimized P_d using MFD and CFD using PSO

Table 9. P_f against optimized P_d using PSO for CFD in NOMA-QAM

BER of CFD	0.484	0.491	0.493	0.495	0.496	0.312	0.212
BER of MFD	0.39	0.37	0.339	0.309	0.272	0.237	0.199
SNR	0	2	4	6	8	10	12

Table 9 and Fig. 9 shows PSO-optimized Pd vs Pf plot using PSO in MFD and CFD technique. Results improved and high value of P_d was achieved for lesser P_f values showing improved detection performance (Pd of 0.75) at reduced false alarm rates (P_f of 0.33). At P_f = 0.3, PSO-optimized CFD achieves P_d = 0.79, which translates to a 35% increase in successful PU detection compared to MFD. This is critical in CR-IoT applications where minimizing missed detection reduces interference and improves network reliability. CFD surpasses MFD in higher SNR scenarios due to its ability to exploit cyclic features of modulated signals, which are preserved even in moderately noisy environments. The integration of PSO further enhances detection performance by adaptively selecting parameters that maximize P_d under false alarm constraints. Despite its superior performance, CFD exhibits higher computational complexity compared to MFD, making it less suitable for real-time or resource-constrained IoT nodes. Additionally, PSO requires tuning and incurs optimization overhead, which may limit deployment in ultra-low-latency scenarios.

5- Conclusion

This study introduces a PSO-optimized power allocation framework for NOMA-QAM systems in cognitive radio environments, targeting enhanced detection using CFD and MFD techniques. The proposed model significantly improves detection performance, particularly for high-order modulation schemes like QAM-256, achieving up to 47.91% gain in P_d over traditional MFD approaches. CFD demonstrates superior robustness at low SNR and reduced sensing time when optimized via PSO. These improvements contribute to more reliable and energy-efficient spectrum access, addressing the demands of IoT-enabled Beyond 5G networks. Future work will explore integration with IRS-assisted channels and deep learning-based sensing optimization for dynamic environments.

6- Future Research Directions

Future research can extend the proposed PSO-based power allocation framework to support advanced modulation schemes like OFDM and OTFS. Incorporating adaptive sensing techniques, such as machine learning-based threshold selection or reinforcement learning, may further enhance detection in dynamic environments. Additionally, integrating Intelligent Reflecting Surfaces (IRS) to improve signal quality and spectral efficiency, especially in obstructed scenarios, is a promising direction. Finally, validating the system's scalability in large-scale IoT deployments and testing it on real-world platforms would strengthen its practical relevance.

References

[1] Kumar, A., Bharti, S., & Gupta, M. (2019). FBMC vs. OFDM: 5G mobile communication system. International Journal of Systems, Control and Communications, 10(3), 250-264. DOI:10.5815/ijwmt.2023.05.01

[2] Haykin, S., Thomson, D. J., & Reed, J. H. (2009). Spectrum sensing for cognitive radio. Proceedings of the IEEE, 97(5), 849-877. DOI:10.1109/JPROC.2009.2015711

[3] Kumar, Arun, et al. "NOMA based CR for qam-64 and qam-256." Egyptian Informatics Journal 21.2 (2020): 67-71. DOI:10.5815/ijwmt.2023.05.01

[4] Kumar, Arun, and P. Nandha Kumar. "OFDM system with Cyclostationary feature detection spectrum sensing." ICTExpress 5.1 (2019): 21-25. DOI:10.1016/j.icte.2018.01.007

[5] Liang, Y. C., Chen, K. C., Li, G. Y., & Mahonen, P. (2011). Cognitive radio networking and communications: An overview. IEEE transactions on vehicular technology, 60(7), 3386-3407. DOI: 10.1109/TVT.2011.2158673

[6] Datla, D., Wyglinski, A. M., & Minden, G. J. (2009). A spectrum surveying framework for dynamic spectrum access networks. IEEE [7]Transactions on Vehicular Technology, 58(8), 4158-4168. DOI: 10.1109/TVT.2009.2025380

[8]Goldsmith, A., Jafar, S. A., Maric, I., & Srinivasa, S. (2009). Breaking spectrum gridlock with cognitive radios: An information theoretic perspective. Proceedings of the IEEE, 97(5), 894-914. DOI: 10.1109/JPROC.2009.2015717

[9] Tumuluru, V. K., Wang, P., & Niyato, D. (2011). A novel spectrum-scheduling scheme for multichannel cognitive radio network and performance analysis. IEEE Transactions on

Vehicular Technology, 60(4), 1849-1858. DOI: 10.1109/TVT.2011.2117951

[10] Navrátil, P. A., Childs, H., Fussell, D. S., & Lin, C. (2013). Exploring the spectrum of dynamic scheduling algorithms for scalable distributed-memoryray tracing. IEEE transactions on visualization and computer graphics, 20(6), 893-906. DOI: 10.1109/TVCG.2013.261

[11] Nandhakumar, P., & Kumar, A. (2016). Analysis of OFDM system with energy detection spectrum sensing. Indian J. Sci. Technol, 9(16), 1-6. DOI: 10.17485/ijst/2016/v9i16/92224

[12] Saberali, S. A., & Beaulieu, N. C. (2014, October). Matched-filter detection of the presence of MPSK signals. In 2014 International Symposium on Information Theory and its Applications (pp. 85-89). IEEE.DOI:0.1109/ISITA.2014.7001426

[13] Kim, K., Akbar, I. A., Bae, K. K., Um, J. S., Spooner, C. M., & Reed, J. H. (2007, April). Cyclostationary approaches to signal detection and classification in cognitive radio. In 2007 2nd IEEE international symposium on new frontiers in dynamic spectrum access networks (pp. 212-215). IEEE. DOI: 10.1109/DYSPAN.2007.39.

[14] Akyildiz, I. F., Lee, W. Y., Vuran, M. C., & Mohanty, S. (2006). NeXt generation/dynamic spectrum access/cognitive radio wireless networks: A survey. Computer networks, 50(13), 2127-2159. DOI: 10.1016/j.comnet.2006.05.001.

[15] Tu, C. C., & Champagne, B. (2009). Subspace-based blind channel estimation for MIMO-OFDM systems with reduced time averaging. IEEE Transactions on Vehicular Technology, 59(3), 1539-1544.DOI: 10.1109/TVT.2009.2036728.

[16] Zhou, Y., Wang, Y., Wang, T., Zhang, K., & Zhang, W. (2011, May). Iterative inter-cell interference coordination in MU-MIMO systems. In 2011 IEEE 73rd Vehicular Technology Conference (VTC Spring) (pp. 1-5). IEEE. DOI: 10.1109/VETECS.2011.5956799.

[17] Andrews, J. G., Buzzi, S., Choi, W., Hanly, S. V., Lozano, A., Soong, A. C., & Zhang, J. C. (2014). What will 5G be?. IEEE Journal on selected areas in communications, 32(6), 1065-1082. DOI: 10.1109/JSAC.2014.2328098

[18] Srinu, S., & Sabat, S. L. (2013). Cooperative wideband sensing based on cyclostationary features with multiple malicious user elimination. AEU-International Journal of Electronics and Communications, 67(8), 702-707. DOI: 10.1016/j.aeue.2013.01.002.

[19] Quan, Z., Cui, S., Sayed, A. H., & Poor, H. V. (2008). Optimal multiband joint detection for spectrum sensing in cognitive radio networks. IEEE transactions on signal processing, 57(3), 1128-1140. DOI: 10.1109/TSP.2008.2008540.

[20] Na, D., & Choi, K. (2019). DFT spreading-based low PAPR FBMC with embedded side information. IEEE Transactions on Communications, 68(3), 1731-1745. DOI: 10.1109/TCOMM.2019.2952549.

[21] He, Z., Zhou, L., Chen, Y., & Ling, X. (2018). Low-complexity PTS scheme for PAPR reduction in FBMC-OQAM systems. IEEE Communications Letters, 22(11), 2322-2325. DOI: 10.1109/LCOMM.2018.2863908.

[22] Kumar, A., & Gupta, M. (2018). A review on activities of fifth generation mobile communication system. Alexandria Engineering Journal, 57(2), 1125-1135. DOI: 10.1016/j.aej.2017.06.014.

[23] Bairagi, A. K., et al. (2020). Coexistence mechanism between eMBB and uRLLC in 5G wireless networks. IEEE transactions on communications, 69(3), 1736-1749. DOI: 10.1109/TCOMM.2020.3046635.

[24] Mounir, M., et al. (2021). A novel hybrid precoding-companding technique for peak-to-average power ratio reduction in 5G and beyond. Sensors, 21(4), 1410. DOI: 10.3390/s21041410

[25] Kumar, A., et al. (2021). An Efficient Hybrid PAPR Reduction for 5G NOMA-FBMC Waveforms. Computers, Materials & Continua, 69(3). DOI: 10.32604/cmc.2021.019092

[26] Miah, M. S., Schukat, M., & Barrett, E. (2020). Sensing and throughput analysis of a MU-MIMO based cognitive radio scheme for the Internet of Things. Computer communications, 154, 442-454. DOI: 10.1016/j.comcom.2020.02.040

[27] Ramamoorthy, R., et al. (2022). Analysis of cognitive radio for lte and 5g waveforms. Computer Systems Science & Engineering, 43(3).DOI: 10.32604/csse.2022.019943.

[28] Pham, Q. V., et al. (2021). Swarm intelligence for next-generation networks: Recent advances and applications. Journal of Network and Computer Applications, 191, 103141. DOI: 10.1016/j.jnca.2021.103141.

[29] Bala Kumar, D., & Nanda Kumar, S. (2024). Block chain-enabled cooperative spectrum sensing in 5G and B5G cognitive radio via massive multiple-input multiple-output non orthogonal multiple access. Results in Engineering, 24, 102840.DOI: 10.1016/j.rineng.2024.102840

[30] Salameh, H. B., & Hussienat, A. (2025). ML-Driven Feature-Based Spectrum Sensing for NOMA Signal Detection in Spectrum-agile IoT Networks under Fading Channels. IEEE Sensors Journal. DOI: 10.1109/JSEN.2025.3524968.

[31] Zhai, Q., Dong, L., Cheng, W., Li, Y., & Liu, P. (2023). Joint optimization for active IRS-aided multicluster NOMA systems. IEEE Systems Journal, 17(4), 6691-6694. DOI: 10.1109/JSYST.2022.3228965.