Indonesian Journal of Computer Science
ISSN 2549-7286 (*online*)
Jln. Khatib Sulaiman Dalam No. 1, Padang, Indonesia
Website: ijcs.stmikindonesia.ac.id | E-mail: ijcs@stmikindonesia.ac.id

Parallel Processing Impact on Random Forest Classifier Performance: A CIFAR-10 Dataset Study

Bareen Haval Sadiq[1], Subhi R. M. Zeebaree[2]
bareen.haval@dpu.edu.krd, subhi.rafeeq@dpu.edu.krd
[1]IT department, Technical College of Duhok, Duhok Polytechnic University, Duhok, Iraq
[2]Energy Eng. Dept., Technical College of Engineering, Duhok Polytechnic University, Duhok, Iraq

Article Information

Submitted : 5 Mar 2024
Reviewed: 12 Mar 2024
Accepted : 1 Apr 2024

Keywords

Parallel Processing,
Distributed Systems,
Cloud Computing,
Machine Learning,
Random Forest

Abstract

Using the CIFAR-10 dataset, this research investigates how parallel processing affects the Random Forest method's machine learning performance. Accuracy and training time are highlighted in the study as critical performance indicators. Two cases were studied, one with and one without parallel processing. The results show the strong prediction powers of the Random Forest algorithm, which continues to analyze data in parallel while retaining a high accuracy of 97.50%. In addition, training times are notably shortened by parallelization, going from 0.6187 to 0.4753 seconds. The noted increase in time efficiency highlights the importance of parallelization in carrying out activities simultaneously, which enhances the training process's computational efficiency. These results provide important new information about how to optimize machine learning algorithms using parallel processing approaches.

A. Introduction

Multiple processors are present in the behavior of complicated real-time systems, which is a complicated design when handling multitasking processing [1]. When dealing with several processes, multiprocessor systems can generate efficient process execution [2]. Additionally, time-sharing across these processes in a real-time manner can enhance execution productivity [3]. Thus, the capacity to control the process of newly entered processes with adequate memory space to be accessible continually will be provided by employing multiprocessor systems [4].

Serial processing takes longer to complete than parallel processing, based on the essential factors that convert user and programmer concerns into workable solutions[5]. Modern computer systems' architecture is built on various processor combinations. The capacity of multiprocessors to run several threads in parallel allows threads for the same resource to be processed on many processors at once [6]. The compact parallel processing capacity of multicore processors, which have gained popularity, cannot be fully utilized unless the software being worked on is designed for it. It's really difficult to write a parallel program that works and scales [7].

Many parallels are required to execute a program on a larger number of cores efficiently in order to maximize the performance of multi-core computers. several processes running concurrently on many cores Although multi-core processors have been present for a while, their significance increased as a result of the technological constraints single-core processors now face, such as those relating to high throughput, extended battery life, and great energy efficiency [8]. Building a Uniprocessor (UP) system with a faster (and more expensive) processor is the first method of increasing a computer's processing capability; building a system with many processors is the second [9]. Parallel processing is the broad term used to describe the second method. While intelligent multi-core computers are capable of parallel processing by task, the development of image processing applications typically necessitates multi-threaded coding [10]. Reducing processing time is crucial for increasing efficiency in systems that handle massive amounts of data for analysis. Long processing times are the result of large data quantities. As a result, it is frequently necessary to shorten the time that these operations execute [11].

In various types of research settings, random forest classification is a widely used machine learning technique for creating prediction models [12]. Reducing the number of variables required to produce a forecast is frequently the aim of prediction modeling, which aims to increase efficiency and lessen the workload associated with data collecting. There are several techniques for selecting variables when using random forest classification [13]. Random forest algorithms are useful for parallel implementation since each decision tree is independent, which makes them one of the hotspots for current large data research [14] However, huge data and feature redundancy lead standard RF algorithms to suffer from low accuracy and computational efficiency due to memory, time, and data complexity limits [15].

A type of operating system called a cloud is made to function in virtualization and cloud computing networks. Virtual machines, virtual servers, virtual infrastructure, hardware, and software backbone are all managed by a cloud operating system. Particle physics, data retrieval, and other fields employ a number of technologies that are employed in cloud computing technology. To

increase cloud computing performance, however, many strategies are applied. While the term "cloud" is frequently used in certain businesses, it is not entirely complete or helpful [16].

This study demonstrates how CPU parallelism impacts Using the random forest approach, CIFAR-10 is a multi-class dataset. Using CPU parallel and non-parallel processing, the major objectives are to demonstrate how parallelism affects algorithm performance. Determine the computational efficiency in terms of processing time for both model training as well as forecasting with and without CPU parallelization.

B. Background Theory

The design philosophy of microprocessors on the usage of many processing cores changed as a result of advancements in the computer industry. This modification enables the simultaneous execution of several instructions by a single device [17]. These evolutions show a maximum core frequency and reveal a tendency toward parallel computing as an alternative to serial computing in order to surpass this limit. Modern processors are made up of several processing components (multicores), as opposed to one powerful processing unit like those found in older processors. The design of multiprocessors with several cores has opened up new possibilities for enhancing computer simulation performance. Prototypes of parallel programming are a growing and difficult problem in the era of parallel computing. When computer applications with high processing demands are identified, challenges in the form of effective programming models to design are faced. With the help of these programming models, the software can assist the performance of such apps while the hardware can handle the computations. Improved programming is therefore required to make development easier and, at the same time, to port a high performance [18].

Parallel Processing

Sequential or serial algorithms are those that need steps to be taken to complete an operation. Parallel algorithms are those that allow for the simultaneous execution of many operations. For a parallel computer, a parallel algorithm is a collection of processes that may be run concurrently and can communicate with one another to solve a particular issue. One definition of the term "process" is a component of a program that a processor may execute. The effectiveness of a parallel algorithm's usage of resources must be considered when creating it [19]. After an algorithm in parallel has been created, its efficiency (or performance) on a parallel computer should be assessed using a measurement. Run time is a frequently used common measurement. Run time, which is sometimes called elapsed time or completion time, is the amount of time an algorithm needs to run on a parallel computer to solve a given issue. To be more precise, it is the amount of time that passes between the initial processor starting and the last processor (or last group of processors) ending. Run time, as opposed to processor usage, is the most accurate way to gauge efficiency in a distributed processing context. This is typically the case since the main objective of parallel processing is to complete the calculation as quickly as feasible rather than to make

effective use of processors. To reduce the run time, it appears that multiplying the number of processes involved in issue-solving might be a solution[20].

The idea of parallel processing, is by no means new. Performance research has spent decades trying to find ways to speed up floating-point and other processes associated with solving numerically demanding algorithms used in domains like fluid dynamics and structural mechanics. The three main subfields of parallel processing are server-side functions, server-process client-side functions, and client-process object rendering. Parallel processing has two distinct uses or applications. On the one hand, high-performance systems are used to accelerate computationally intensive tasks. Large workstation clusters or conventional supercomputer systems can be used to run these. However, some embedded control systems operate on sequential hardware and need principles from parallel programming in order to handle concurrent external actuators or internal operations. Parallel software design is becoming more and more significant, and parallel processing is widely used nowadays under mainstream operating systems like Windows and Linux [21].

Image Processing Fundamentals

The idea of a digital picture was initially presented in the early 20th century when digital images were sent over undersea cable systems [22][23]. Furthermore, the development of contemporary digital image processing techniques was facilitated by advancements in computer hardware and processing units. In particular, the field of remote sensing applications gave rise to digital image processing. The image signal processing of a contemporary image capture system is crucial to producing digital images with a high quality [24]. Light travels via the color filter array (CFA) and lens. We are unable to get color information because an image sensor without a CFA absorbs light throughout the whole spectrum. Due to the heightened sensitivity of the human visual system to light at the green wavelength, digital cameras often employ a standard CFA known as the Bayer pattern, which consists of two green (G), one red (R), and one blue (B) filter to create color images. To boost the quantity of light, the advanced CFA swaps out the single green filter for a white filter [25].

Machine Learning

The fusion of distributed frameworks and artificial intelligence reveals itself as a potent catalyst in the dynamic cloud computing ecosystem, transforming the nature of services and apps [26]. The intersection of distributed frameworks and artificial intelligence emerges as a fundamental hub for managing the barriers manifest in contemporary computing standards in an era of unmatched information volumes, elevated preparation requirements, and the unavoidability of deft inventions. This innovative collaborative energy, which provides creative solutions to a variety of intricate problems pertaining to asset coordination, security, and adaptability in distributed frameworks, is the key to unlocking cloud computing's otherwise untapped potential [27].

The goal of efficient machine learning model creation has led to the standardization of the use of parallel processing to shorten training times. The effects of employing parallel processing and the Random Forest approach are

examined in this paper. Two key performance parameters are evaluated: training time and accuracy, in order to detect any significant differences in the model's output. The results provide a comprehensive insight of how the Random Forest method's prediction accuracy and computational efficiency are impacted by parallel processing [28].

In machine learning, the efficiency of algorithms has a major impact on the creation and use of models. Parallel processing is now a potent tool for speeding up the training of complex models, allowing computers to handle the demands of complex activities.

Random Forest

Random Forest is a bagging classifier that uses two stochastic decision levels in its learning process. Each decision tree in the ensemble chooses a subset of samples and features for training. Random Forest decision tree models are very well-liked for application in various machine-learning situations due to their simplicity of usage and interpretation. Each decision tree operates poorly when used alone because it is vulnerable to overfitting [29].

Decision Tree

Decision tree approach is a widely used and easily comprehensible machine learning methodology. A decision tree of options, as its name suggests, partitions the data space into smaller subspaces, each of which is given a label or a probability. As the tree is constructed during training, the algorithm examines every possible split along every axis to make sure that every split is carried out as efficiently as is practical. A number of metrics, including as entropy, information gain, and Gini, may be used to quantify the impurity of the resulting two partitions, and the optimal split point is identified as having the lowest impurity among them.

C. Introduction

In this research [30] describe the two primary issues that parallel algorithms face: they shorten execution times at the expense of decreased prediction accuracy. They also provide solutions to these issues. (1) To make the process of data parallelism easier, data is partitioned vertically along feature space and horizontally along samples. (2) Optimal and significant features are chosen using the Parallel Multilevel Feature Selection (M-FS) method to enhance the categorization of cancer subtypes. The chosen characteristics are assessed using Spark's parallel Random Forest, and the outcomes are contrasted with both the sequential implementation of the identical methods and findings from earlier reports.

In this research [31]suggests a novel approach to HCS that is based on a cloud environment and optimizes the choice of virtual machines (VMs) using Parallel Particle Swarm Optimization (PPSO). Furthermore, a novel model for the diagnosis and prognosis of chronic kidney disease (CKD) is suggested to assess the effectiveness of our VMs model. Two successive approaches are used to create the CKD prediction model: neural networks (NN) and linear regression (LR). LR is used to identify important variables that affect CKD. NN is employed in CKD

prediction. According to the results, the suggested model performs 50% faster overall during execution than the state-of-the-art models.

In this research [32] provide a development approach for a multi-computer, multicore-processor distributed memory system. It is possible to use this technique on distributed-shared memory systems, applying client/server architectural concepts. The two main parts of the system that is being described are programs that are run on distributed multi-core architectures with 2, 4, and 8 CPUs to do certain tasks, and programs that are managed. Three main scenarios covering most of the design options must be taken into account during the implementation process. The suggested system may calculate all key server timings, including Started, Elapsed, CPU, Kernel, User, Waiting, and Finish, in addition to the Total-Task-Time (TTT) on the client side.

In this research [33] discussed a high-performance computing (HPC)-capable version of Iterative Random Forest (iRF). Explainable-AI eQTL analysis of SNP sets including more than a million SNPs is made possible by this new implementation. Additionally, it offers a cutting-edge method called iRF Leave One Out Prediction (iRF-LOOP), which enables the development of Predictive Expression Networks with a minimum of 40,000 genes through its use. It examines the time taken to complete the task on Summit and Titan, the two fastest supercomputers in the world, and compares the new implementation of iRF with the earlier R version.

The Parallel Random Forest (PRF) approach is available on the Apache Spark platform for large data sets is discussed in. The PRF method is optimized using a hybrid technique that combines task-parallel and data-parallel optimization. While a vertical data-partitioning strategy is used to effectively reduce the cost of data transmission, a data-multiplexing technique is used in data-parallel optimization to limit the quantity of data and allow the training dataset to be reused. In order to mitigate the issue of imbalanced data, proposed a novel weighted classes classification strategy to protect the network from malicious nodes. We include a supervised machine learning technique with a specially designed best effort iterative methodology, utilizing historical network node data, into the proposed system to enhance the accuracy of seldom detected assaults.

In this research [34] showed a distance-weighted optimal strategy for a parallel random forest algorithm is presented to address the issues with the current methods' lengthy time to execution and limited parallelism. The experimental findings demonstrate that the parallel random forest algorithm's optimization reduces the algorithm's execution time by 110 000 ms and significantly boosts its operational efficiency, effectively resolving the issues with the conventional random forest algorithm.

In this research [35] suggested a strategy to do the prediction a specific amount of time ahead of the anticipated time point was proposed. This would enable a sufficient amount of time for scheduling tasks depending on the anticipated workload. We provide a workload prediction technique based on clustering, which first divides all the jobs into many groups and then trains a model for forecasting for each category independently, to further increase the forecast accuracy. The workload prediction techniques based on clustering outperform previous comparative approaches and enhance the forecast accuracy

to approximately 90 percent in both CPU and memory, as shown by the trace-driven studies conducted using Google cluster trace.

In this research [36] investigates the maintaining the classification and regression trees (CARTs) with superior classification effects and lowering the correlations between the CARTs make up the core notion. To be more precise, each CART was used to forecast three sets of reserved data in the classification impact assessment section, after which the mean accuracy of classification were attained, one by one. Compared to the five random forests used as a reference, the suggested enhanced random forest produced an average classification accuracy that was greater, and the lead was steady.

In this research [37] discusse the three optimizing elements of the suggested FastForest algorithm—Subsample Aggregating, or "Subbagging," Logarithmic Split-Point Sample collection, and Dynamic Limited Subspacing—combine to produce this outcome. Empirical testing on 45 datasets on PC and smartphone platforms reveals that Fast Forest maintains classification accuracy while offering an average 24% speed boost over Random Forest in model-training.

In this research [38] to successfully lower the cost of data transfer, a vertical data-partitioning approach is utilized, and a data-multiplexing technique is used to minimize the volume of data and enable the dataset used for training to be reused. When it comes to task-parallel optimization, RF is trained using a dual parallel technique, and a task Directed Acyclic Graph (DAG) is produced based on PRF's parallel training process and the reliance on the Resilient Distributed Datasets (RDD) objects. Comprehensive testing outcomes demonstrate the superiority and noteworthy benefits of the PRF algorithm in terms of classification accuracy, performance, and scalability when compared to the pertinent methods used by Spark MLlib and other research.

In this research [39] suggests using parallel computing to create the random forest approach using the R programming language. When random forest is implemented, one typical issue that frequently arises is high processing times since it takes a lot of data and builds a lot of tree models to construct random trees on a single processor. Among the instances utilized in this study are the Iris flower dataset, wine quality data, and Pima Indian woman's diabetes diagnostic data. The overall study findings demonstrate that utilizing parallel computing to run random forests reduces the computational time compared to using a single processor to run standard random forests.

In this research [40] focuses on creating a system with two primary stages: program monitoring and control. The program may operate on several multicore system architectures, such as those with 2, 4, and 8 CPUs. The work's algorithms are designed to be able to provide information about dependent computer systems, check the status of all processes that are currently running and provide pertinent information, and run all possible cases of processes and threads that make up the user program and may contain one of these scenarios (Single-Processes/Single-Threads, Single-Process/Multi-Thread, Multi-Process/single-Thread, Multi-Process/Multi-Thread and Multi-Process/ Single-Multi-Thread).

D. The Proposed Method

The proposed approach's general framework is depicted in Figure 1. First, sets for training and testing are created using the CIFAR-10 test dataset, with a preset allocation of 80% for training and 20% for assessment. Next, the random forest technique was used to utilize the training data first without doing any parallel processing, and subsequently it was used to use the same data using parallel processing. Main purpose of the system is to demonstrate the deference in performance and time execution of the random forest algorithm with parallel processing and without parallel processing. These experiments were conducted on a system equipped with an AMD Ryzen 5 Microsoft Surface (R) Edition 2.10 GHz processor (8 CPUs) and 8GB of RAM.

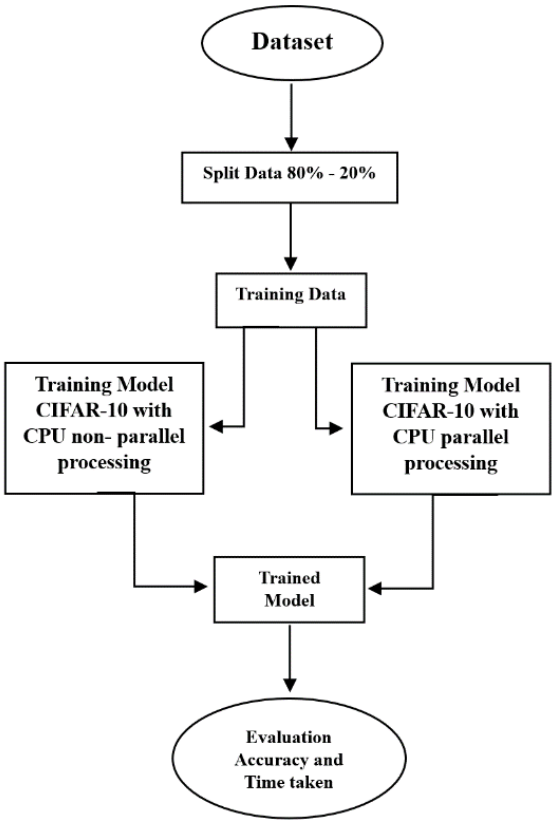

Figure 1. Flowchart for the proposed approach.

Dataset

The CIFAR-10 dataset is used in this investigation. This multi-class dataset has 60,000 32x32 color image divided into 10 classes, with 6,000 images in each class. Ten thousand test images and fifty thousand training images in total. [41].

Random Forest

The Random Forest classifier is a group of tree-structured classifiers with evenly distributed random vectors that are independently distributed [42]. Stated differently, RF creates regression trees and then takes the average of the results once it receives the input vector that contains the characteristic values for a particular training set. By eliminating the association between distinct decision

trees, random forest reduces the variance in bagging by creating trees from several training data subsets. Through replacement sampling, the random forest model resamples the original dataset to provide training data using the bagging technique.

The enhanced consistency and predictability of the random forest method is aided by this bagging feature. To build a tree and improve generalization ability while lowering generalization error, random forest employs the best-split variables in a selected at random evidentiary feature subset. Samples designated as "out-of-bag" (OOB) were those that were not chosen for the bagging procedure training. Classification algorithms perform better and are more stable when there are comparatively less characteristics used in the classification process. Selecting pertinent characteristics for the building of classifiers has so drawn a lot of attention.

Figure 2 shows the Diagram for the Random Forest. By averaging the predictions of individual trees T, the Random Forest method provides forecast $\hat{Y}$ for a given input X:

$$\hat{Y} = \frac{1}{N} \sum_{i=1}^{N} T_i(X) \qquad (1)$$

N represents the number of trees in the forest. The goal of adjusting the Random Forest model's parameters, which involve the total number of trees, maximum depth, and minimum samples per leaf, is to either minimize the mean squared error (MSE) or increase the coefficient of determination.

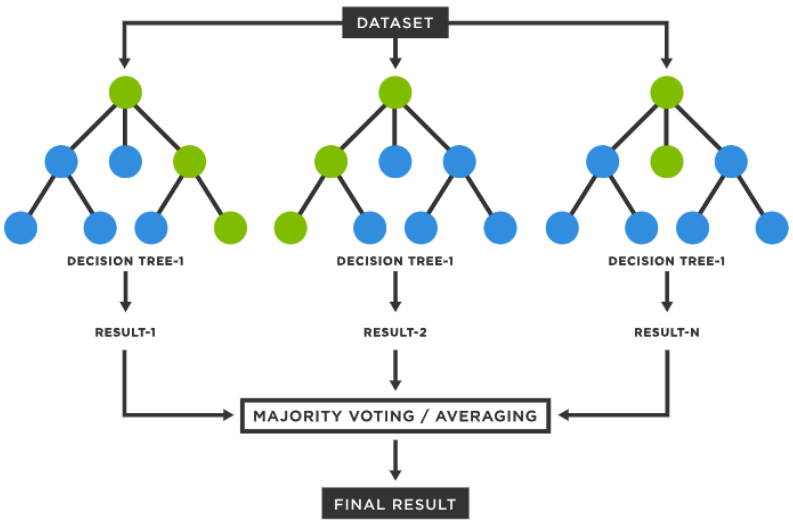

Figure 2. Random Forest diagram [42]

E. Results

In this study, the Random Forest approach is investigated with and without parallel processing using Python as the programming language. The predictive capacity of the model is measured by accuracy, and the computational effectiveness is measured by training time. These are the two main performance metrics that this study looks at. By means of a comparative analysis, the study

seeks to ascertain the impact of parallel processing on Random Forest method performance.

The Random Forest approach is trained using a CIFAR-10 dataset, and its performance was assessed in two separate scenarios: one with parallel processing and the other without. The key findings are as follows:

Even with parallel processing, the accuracy stayed at 97.50%.

The accuracy score shows that adding parallel processing has no effect on the model's ability to generate accurate predictions. Since any variation might indicate potential issues with the parallel processing approach, this accuracy consistency is crucial for evaluating the success of parallelization efforts.

Efficiency in Terms of Time without Using Parallel Processing The model took 0.6187 seconds to train. The task was completed in 0.4753 seconds due to parallel processing, indicating a significant reduction in training time. It is evident that time efficiency is increased by parallel processing. The shortened training period implies that the parallelized Random Forest algorithm's execution benefited from simultaneously performing tasks, making the model's training process more time-efficient.

In this study the dataset that was used is relatively small in size that's why it does not show clearly the difference in time when executing the random forest algorithm in parallel and without parallelism. However; if we use a large size dataset the time required to execute the algorithm in parallel mode would be significantly less that executing it without parallelism.

F. Discussion and Comparison

When used in the context of parallel processing, the Random Forest method showed a reduction in training time while maintaining the high level of prediction accuracy displayed by the non-parallelized model. This demonstrates that increasing the Random Forest algorithm's computational efficiency through parallel processing is a workable strategy that maintains the functionality of the model.

The same accuracy ratings of the two situations indicate that the decision-making processes were properly synchronized across parallel threads or cores by the parallel processing solution.

Particularly noteworthy is the observed reduction in training time, which highlights the practical advantages of parallel processing when processing capacity is scarce. When tasks are successfully finished in parallel, machine learning becomes more productive overall. Faster model installation, modification, and creation are all part of this.

However, it is important to consider the scalability of these results. It may be feasible to gain a better grasp of the adaptability and potential benefits of parallel processing by experimenting with larger datasets or more intricate models. Moreover, the hardware architecture is crucial, and more powerful parallel processing systems may yield even greater efficiency gains.

G. Conclusion

This paper investigates the impact of parallel processing on the machine learning performance of the Random Forest method. The Random Forest

technique is developed using the CIFAR-10 dataset, and the study emphasizes accuracy and training time as key performance metrics. Two scenarios were assessed, one with parallel processing and the other without. The algorithm's strong potential for forecasting is proved by the data, which indicate that accuracy remains constantly high at 97.50% with parallel processing. Furthermore, the research indicates that training times utilizing parallel processing are much lower (0.4753 seconds) compared to those that do not (0.6187 seconds). This increase in time efficiency demonstrates the value of parallelization by showing how concurrent task execution improves the computational efficiency of the training process. The findings offer valuable new insights into the application of parallel processing techniques for machine learning algorithm optimization.

H. References

[1] I. M. I. Zebari, S. R. M. Zeebaree, and H. M. Yasin, "Real time video streaming from multi-source using client-server for video distribution," in *2019 4th Scientific International Conference Najaf (SICN)*, IEEE, 2019, pp. 109–114.

[2] Z. S. Ageed *et al.*, "A state of art survey for intelligent energy monitoring systems," *Asian Journal of Research in Computer Science*, vol. 8, no. 1, pp. 46–61, 2021.

[3] K. Jacksi, N. Dimililer, and S. R. Zeebaree, "State of the art exploration systems for linked data: a review," *Int. J. Adv. Comput. Sci. Appl. IJACSA*, vol. 7, no. 11, pp. 155–164, 2016.

[4] A. Salih, S. T. Zeebaree, S. Ameen, A. Alkhyyat, and H. M. Shukur, "A survey on the role of artificial intelligence, machine learning and deep learning for cybersecurity attack detection," in *2021 7th International Engineering Conference "Research & Innovation amid Global Pandemic"(IEC)*, IEEE, 2021, pp. 61–66.

[5] D. A. Hasan, B. K. Hussan, S. R. M. Zeebaree, D. M. Ahmed, O. S. Kareem, and M. A. M. Sadeeq, "The impact of test case generation methods on the software performance: A review," *International Journal of Science and Business*, vol. 5, no. 6, pp. 33–44, 2021.

[6] S. R. Zeebaree, R. R. Zebari, K. Jacksi, and D. A. Hasan, "Security approaches for integrated enterprise systems performance: A Review," *Int. J. Sci. Technol. Res*, vol. 8, no. 12, pp. 2485–2489, 2019.

[7] P. Y. Abdullah, S. R. Zeebaree, H. M. Shukur, and K. Jacksi, "HRM system using cloud computing for Small and Medium Enterprises (SMEs)," *Technology Reports of Kansai University*, vol. 62, no. 04, p. 04, 2020.

[8] M. A. Omer, S. R. M. Zeebaree, M. A. M. Sadeeq, B. W. Salim, Z. N. Rashid, and L. M. Haji, "Efficiency of malware detection in android system: A survey," *Asian Journal of Research in Computer Science*, vol. 7, no. 4, pp. 59–69, 2021.

[9] S. R. M. Zeebaree, A. B. Sallow, B. K. Hussan, and S. M. Ali, "Design and simulation of high-speed parallel/sequential simplified DES code breaking based on FPGA," in *2019 International Conference on Advanced Science and Engineering (ICOASE)*, IEEE, 2019, pp. 76–81.

[10] J. Saeed and S. Zeebaree, "Skin lesion classification based on deep convolutional neural networks architectures," *Journal of Applied Science and Technology Trends*, vol. 2, no. 01, pp. 41–51, 2021.

[11] Z. M. Khalid and S. R. M. Zeebaree, "Big data analysis for data visualization: A review," *International Journal of Science and Business*, vol. 5, no. 2, pp. 64–75, 2021.
[12] M. R. Mahmood, M. B. Abdulrazzaq, S. Zeebaree, A. K. Ibrahim, R. R. Zebari, and H. I. Dino, "Classification techniques' performance evaluation for facial expression recognition," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 21, no. 2, pp. 176–1184, 2021.
[13] J. L. Speiser, M. E. Miller, J. Tooze, and E. Ip, "A comparison of random forest variable selection methods for classification prediction modeling," *Expert Syst Appl*, vol. 134, pp. 93–101, 2019.
[14] M. A. S. Ali *et al.*, "A Novel Method for Survival Prediction of Hepatocellular Carcinoma Using Feature-Selection Techniques," *Applied Sciences*, vol. 12, no. 13, p. 6427, 2022.
[15] L. Yin, K. Chen, Z. Jiang, and X. Xu, "A Fast Parallel Random Forest Algorithm Based on Spark," *Applied Sciences*, vol. 13, no. 10, p. 6121, 2023.
[16] H. Malallah *et al.*, "A comprehensive study of kernel (issues and concepts) in different operating systems," *Asian Journal of Research in Computer Science*, vol. 8, no. 3, pp. 16–31, 2021.
[17] D. A. Zebari, H. Haron, S. R. M. Zeebaree, and D. Q. Zeebaree, "Multi-level of DNA encryption technique based on DNA arithmetic and biological operations," in *2018 International Conference on Advanced Science and Engineering (ICOASE)*, IEEE, 2018, pp. 312–317.
[18] K. Jacksi, S. R. M. Zeebaree, and N. Dimililer, "Lod explorer: Presenting the web of data," *Int. J. Adv. Comput. Sci. Appl. IJACSA*, vol. 9, no. 1, pp. 1–7, 2018.
[19] A. AL-Zebari, S. Zeebaree, K. Jacksi, and A. Selamat, "ELMS–DPU ontology visualization with Protégé VOWL and Web VOWL," *Journal of Advanced Research in Dynamic and Control Systems*, vol. 11, pp. 478–485, 2019.
[20] S. R. Zeebaree, "DES encryption and decryption algorithm implementation based on FPGA," *Indones. J. Electr. Eng. Comput. Sci*, vol. 18, no. 2, pp. 774–781, 2020.
[21] H. Shukur, S. Zeebaree, R. Zebari, O. Ahmed, L. Haji, and D. Abdulqader, "Cache coherence protocols in distributed systems," *Journal of Applied Science and Technology Trends*, vol. 1, no. 3, pp. 92–97, 2020.
[22] N. O. M. Salim, S. R. M. Zeebaree, M. A. M. Sadeeq, A. H. Radie, H. M. Shukur, and Z. N. Rashid, "Study for food recognition system using deep learning," in *Journal of Physics: Conference Series*, IOP Publishing, 2021, p. 012014.
[23] H. Dino *et al.*, "Facial expression recognition based on hybrid feature extraction techniques with different classifiers," *TEST Engineering & Management*, vol. 83, pp. 22319–22329, 2020.
[24] D. A. Zebari, H. Haron, S. R. M. Zeebaree, and D. Q. Zeebaree, "Enhance the mammogram images for both segmentation and feature extraction using wavelet transform," in *2019 International Conference on Advanced Science and Engineering (ICOASE)*, IEEE, 2019, pp. 100–105.
[25] M. B. Abdulrazaq, M. R. Mahmood, S. R. M. Zeebaree, M. H. Abdulwahab, R. R. Zebari, and A. B. Sallow, "An analytical appraisal for supervised classifiers' performance on facial expression recognition based on relief-F feature

selection," in *Journal of Physics: Conference Series*, IOP Publishing, 2021, p. 012055.

[26] S. M. Mohammed, K. Jacksi, and S. Zeebaree, "A state-of-the-art survey on semantic similarity for document clustering using GloVe and density-based algorithms," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 22, no. 1, pp. 552–562, 2021.

[27] D. A. Hasan, S. R. M. Zeebaree, M. A. M. Sadeeq, H. M. Shukur, R. R. Zebari, and A. H. Alkhayyat, "Machine Learning-based Diabetic Retinopathy Early Detection and Classification Systems-A Survey," in *2021 1st Babylon International Conference on Information Technology and Science (BICITS)*, IEEE, 2021, pp. 16–21.

[28] K. Jacksi, R. K. Ibrahim, S. R. M. Zeebaree, R. R. Zebari, and M. A. M. Sadeeq, "Clustering documents based on semantic similarity using HAC and K-mean algorithms," in *2020 International Conference on Advanced Science and Engineering (ICOASE)*, IEEE, 2020, pp. 205–210.

[29] X. Zhou, P. Lu, Z. Zheng, D. Tolliver, and A. Keramati, "Accident prediction accuracy assessment for highway-rail grade crossings using random forest algorithm compared with decision tree," *Reliab Eng Syst Saf*, vol. 200, p. 106931, 2020.

[30] L. Venkataramana, S. G. Jacob, and R. Ramadoss, "A parallel multilevel feature selection algorithm for improved cancer classification," *J Parallel Distrib Comput*, vol. 138, pp. 78–98, 2020.

[31] A. Abdelaziz, M. Elhoseny, A. S. Salama, and A. M. Riad, "A machine learning model for improving healthcare services on cloud computing environment," *Measurement*, vol. 119, pp. 117–128, 2018.

[32] D. M. ABDULQADER, S. R. M. ZEEBAREE, R. R. ZEBARI, S. A. L. I. SALEH, Z. N. RASHID, and M. A. M. SADEEQ, "SINGLE-THREADING BASED DISTRIBUTED-MULTIPROCESSOR-MACHINES AFFECTING BY DISTRIBUTED-PARALLEL-COMPUTING TECHNOLOGY," *Journal of Duhok University*, vol. 26, no. 2, pp. 416–426, 2023.

[33] A. Cliff, J. Romero, D. Kainer, A. Walker, A. Furches, and D. Jacobson, "A high-performance computing implementation of iterative random forest for the creation of predictive expression networks," *Genes (Basel)*, vol. 10, no. 12, p. 996, 2019.

[34] Q. Wang and H. Chen, "Optimization of parallel random forest algorithm based on distance weight," *Journal of Intelligent & Fuzzy Systems*, vol. 39, no. 2, pp. 1951–1963, 2020.

[35] J. Gao, H. Wang, and H. Shen, "Machine learning based workload prediction in cloud computing," in *2020 29th international conference on computer communications and networks (ICCCN)*, IEEE, 2020, pp. 1–9.

[36] Z. Sun, G. Wang, P. Li, H. Wang, M. Zhang, and X. Liang, "An improved random forest based on the classification accuracy and correlation measurement of decision trees," *Expert Syst Appl*, vol. 237, p. 121549, 2024.

[37] D. Yates and M. Z. Islam, "FastForest: Increasing random forest processing speed while maintaining accuracy," *Inf Sci (N Y)*, vol. 557, pp. 130–152, 2021.

[38] J. Chen *et al.*, "A parallel random forest algorithm for big data in a spark cloud computing environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 4, pp. 919–933, 2016.
[39] N. Azizah, L. S. Riza, and Y. Wihardi, "Implementation of random forest algorithm with parallel computing in R," in *Journal of Physics: Conference Series*, IOP Publishing, 2019, p. 022028.
[40] L. M. Haji, S. R. M. Zeebaree, O. M. Ahmed, M. A. M. Sadeeq, H. M. Shukur, and A. Alkhavvat, "Performance Monitoring for Processes and Threads Execution-Controlling," in *2021 International Conference on Communication & Information Technology (ICICT)*, IEEE, 2021, pp. 161–166.
[41] Y. Abouelnaga, O. S. Ali, H. Rady, and M. Moustafa, "Cifar-10: Knn-based ensemble of classifiers," in *2016 International Conference on Computational Science and Computational Intelligence (CSCI)*, IEEE, 2016, pp. 1192–1195.
[42] D. Wang and A.-X. Zhu, "Soil mapping based on the integration of the similarity-based approach and random forests," *Land (Basel)*, vol. 9, no. 6, p. 174, 2020.