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

Enhancing AdaBoost Performance: Comparative Analysis of CPU Parallel Processing on Breast Cancer Classification

Vaman Ashqi Saeed[1], Subhi R. M. Zeebaree[2]
vaman.saeed@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 : 3 Mar 2024
Reviewed: 12 Mar 2024
Accepted : 1 Apr 2024

Keywords

ADABoost, Parallel Processing, CPU Parallel time, Breast Cancer, Decision Tree.

Abstract

The implementation of time-sharing across processes in a real-time way has the potential to increase the execution efficiency of multiprocessor systems like the one described above. The system is able to carry out tasks that make use of a large number of processors in an effective way as a result of this. The aim of this research is to design a system with two primary goals: to enhance accuracy and to minimise the amount of time necessary with processing. This will be accomplished by integrating the ADABoost model with the decision tree algorithm. Furthermore, the statistics unambiguously demonstrate that the accuracy remains the same regardless of whether or not the central processing unit (CPU) makes use of parallel processing, which suggests that there is no variation in parallelization. As a consequence of this, there is a direct connection between the amount of time that is spent and an increase in the amount of parallel processing that is carried out by the central processing unit pertaining to the breast cancer dataset that is being investigated. This research was carried out using Python, which was the programming language that was used for the coding technique that was carried out during the course of its execution.

A. Introduction

Parallelization and parallel algorithms are traditional areas of computer science that are attracting an increasing number of scholars from a wide range of fundamental and applied scientific disciplines [1][2]. Undoubtedly, the increased accessibility and reduced cost of distributed computer resources have played a significant role in this restored triumph. Nevertheless, the parallel approach has demonstrated promise in addressing challenges that would otherwise be unmanageable, such as the statistical modeling of extremely massive data sets [3] [4]. One of the most basic methods for parallelizing evolutionary algorithms is to execute multiple parallel iterations of the same algorithm independently [5]. The configuration procedure is straightforward, and the results should only be processed once every run has been finished. The utilization of independent trials as a method to parallelize an evolutionary algorithm can be applied to various objectives[6]. When the computational time of the fitness evaluation is not particularly significant, for instance, the optimization algorithm itself must be quick [7]. In particular, when multitasking processing is involved, the behavior of complex real-time systems that comprise more than one processor can be characterized as complex [8]. Time-sharing among these processes in real-time can increase the execution productivity of multiprocessor systems, which are capable of generating effective process execution when dealing with multiple processes [9] [10]. By implementing multiprocessor systems, it becomes feasible to effectively handle newly initiated processes by ensuring a constant supply of adequate memory space. Monitoring the activities of processes and threads is one rationale for providing monitoring software [11].

To enhance the efficiency of machine learning model parallelism, encompassing design, implementation, and execution, numerous researchers have suggested CPU parallel processing methods [12]. These methods enable end-to-end adaptive distributed training solutions for machine learning algorithms and automatically search for and tune distributed parallel strategies[13]. Conventional learning-based single processing methods necessitate the iterative pursuit of learning and feedback, which is resource-intensive and time-consuming [14]. On the contrary, CPU parallel processing methods that are founded upon graph algorithms locate the optimal strategy by utilizing the graph search algorithm [15]. While this approach demands more information regarding the model structure and device topology, it necessitates fewer resources and time compared to the single processing method that is founded upon classical learning strategies [16]. To increase the efficiency of the AdaBoost algorithm, parallel and distributed processing can be utilized. Methods for a CPU-based comparison of the AdaBoost algorithm, one of the most effective classification techniques in use, are presented in the current paper, both with and without parallelization. After that, the paper is structured as follows: The context of the research is presented in Section B. A detailed description of our implementation of the AdaBoost model can be found in Section C. The results and experimental conditions are detailed in Section D. Conclusions are provided in Section E.

B. Background Theory

The Concept of Parallel Processing

Consecutive algorithms, also known as serial algorithms, are those that are executed in computer science in an orderly manner, devoid of any execution of intermediate processing. In contrast, parallel processing merely denotes the concurrent execution of numerous operations or calculations [17]. Parallel processing can also be defined as the process of accelerating the execution of a program by dividing it into smaller components that can be implemented simultaneously on multiple processors, with each component typically having its own processor. The completion duration of a program conducted on Q processors is potentially Q times that of a program executed on a single processor [18].

Parallel Processing Benefits

Early computers had a single-programmer limitation in terms of concurrent program execution. They would require a combined total of two hours to complete their duties, including a one-hour intensive operations program and a one-hour tape-recording program. Both programs are initially executed concurrently at the outset of parallel processing. Before initiating an in/out operation, the computer would execute the Intensive operations program while it awaited the completion of the operation. A total of under an hour would be required to finish the two duties [19].

Parallel Processing Applications

In order to ensure the security and reliability of the United States' remaining nuclear arsenal, parallel processing devices are implemented. The analysis and prediction of potential issues arising from the extended storage of nuclear devices necessitate the use of extremely precise numerical simulations, in lieu of the nuclear testing that was previously conducted for research purposes, whether above or below ground [20].

• For the purpose of regulating the strength and durability of handrails in the event of a collision, parallel processing is applied to the development of computer-generated models of vehicles and grips. One model can require up to five days to execute on a single-processing computer, whereas it can be completed in several hours on a parallel machine.

• Airlines employ parallel processing to handle consumer data, predict requests, and determine pricing guidelines.

• Medical parallel processing devices are employed for the analysis of MRI images and models of bone implantation systems.

• Further applications include decrypting encrypted codes, conducting structural analysis, exploring geological information, utilizing animated graphics, computational fluid dynamics, studying chemistry and physical science, electronic designing, and engaging in climatology.

Boosting Algorithms Fundamentals

Boasting algorithms aim to increase prediction power by training a series of feeble models, with each model compensating for the shortcomings of its antecedent, as opposed to the approach taken by many ML models that emphasize high-quality predictions made by a single model. In ensemble learning, boosting is a crucial technique. In terms of learning algorithm design, it introduces a novel approach and concept. By integrating several weak learners, whose accuracies marginally surpass those of random guesswork, the objective is to generate a robust predictor with arbitrary accuracy. When the direct development of a robust learning algorithm becomes an impracticable undertaking, this becomes a critical consideration. Almost all prevalent machine learning algorithms can have their prediction accuracy enhanced through the utilization of Boosting as a meta-learning framework. As a result, it exhibits significant influence in the domains of machine learning and attains pervasive application. Beyond all other boosting algorithms, specifically.

Gradient Boosting is an additional well-known boosting method, whose algorithms include the increasingly popular XGBoost, LightGBM, and CatBoost. Gradient Boosting constructs a series of decision trees incrementally, fitting the residuals of the preceding tree to inform the construction of each subsequent tree. By reducing the overall prediction error, this methodology enhances the robustness of gradient boosting and renders it applicable to a diverse array of tasks, such as classification and regression. Boosting algorithms provide numerous benefits, such as enhanced predictive precision, the ability to generalize to a wide range of datasets, and resilience against overfitting. They may, nevertheless, be susceptible to chaotic data and outliers. In the realm of machine learning, boosting algorithms are of paramount importance, as they facilitate the construction of models that are not only precise but also adaptable across a wide range of domains and applications. s [21].

AdaBoost is the most successful representative algorithm and is ranked among the top ten data mining algorithms. Numerous renowned researchers have devoted considerable effort to various theoretical subjects associated with AdaBoost since its inception, thereby establishing a robust theoretical framework that may ultimately facilitate the effective implementation of AdaBoost. AdaBoost has achieved success not solely due to its efficacy as a learning algorithm, but also in consideration of the subsequent factors. Initially, it materializes Boosting, which was previously a mere speculation. Furthermore, the algorithm's implementation of certain techniques, including the modification of the initial distribution of training samples, sheds light on the design considerations of numerous other learning algorithms. Thirdly, AdaBoost-related developments have stimulated advancements in ensemble learning research.

There are numerous iterations of the AdaBoost algorithm, each specifically developed to address binary classification, regression, or multi-class classification challenges. The AdaBoost algorithm for multi-class classification is depicted in Figure 2. The AdaBoost pseudocode is characterized as follows:

Figure 1. AdaBoost algorithm exemplified [22]

Figure 1 illustrates that samples with greater associated weight (w) as a result of misclassification in the preceding phase (indicated by X) are represented by circles of varying sizes.

Machine Learning Concept

The scientific investigation of algorithms and statistical models that induce unprogrammed behavior in computer systems is referred to as machine learning (ML) [23]. There are numerous applications in which we utilize learning algorithms on a daily basis. One of the reasons a learning algorithm such as the one utilized by Google to rank web pages is that it improves with each use of the search engine [24][25]. A multitude of applications employ these algorithms, including but not limited to data mining, image processing, and predictive analytics [26][27]. One primary benefit of employing machine learning is that it enables algorithms to execute their tasks autonomously once they have acquired knowledge of how to process data[28][29]. A concise overview and prospective outlook on the extensive range of machine learning algorithm applications are presented in this paper [30][31].

Breast Cancer Classification (BCC)

BCC attempts to ascertain the most appropriate course of action, which, contingent upon the cancer's classification, may be more or less aggressive. For an accurate prognosis, the classification of breast cancer requires the following nine characteristics: 1. ascertain the stratified structures (thickness of the pile); 2. Assess the consistency and magnitude of the sample (Uniformity of Cell Size); 3. Determine the degree of cell shape uniformity and detect minor deviations, as the morphology of cancer cells is prone to variation (Uniformity of Cell Shape); 4. Marginal adhesion indicates that normal cells are interconnected while cancer cells are dispersed throughout the organ; 5. Single epithelial cell size indicates that malignancy is indicated by enlarged epithelial cells, which are a measure of uniformity; 6. Nuclei of benign tumors are not encircled by cytoplasm; 7. The nucleus texture is characterized by a uniform shape in benign cells. The chromatin content of malignancies is typically more coarse; 8. The nucleolus is typically negligible and quite diminutive in healthy cells. There are numerous nucleoli in cancer cells, and each one becomes considerably more prominent. 9. Compute an approximation of the number of mitoses that have occurred. A higher magnitude corresponds to an increased probability of developing cancer (Mitoses). Pathologists assigned numerical values ranging from 1 to 10 to each of these characteristics in order to classify BC. Malignancy probability requires all nine criteria, even if one is extremely high. [32].

C. Related Works

The Authors here [33] the concept of integrating parallel and distributed computation . That In contrast to parallel computing on a single computer, which utilizes multiple processors to execute tasks in parallel, distributed parallel computing utilizes numerous computing devices to process tasks in parallel. The design of distributed systems is distinct from that of the primary network. Distributed systems include groups, grids, peer-to-peer (P2P) networks, and distributed storage systems, among others. Two distinct varieties of multicore processors exist: heterogeneous and homogeneous. This article examines the effects of multicore distributed-memory parallel processing systems. Furthermore, a variety of techniques have been proposed for implementation in distributed-memory systems, with an emphasis on determining which technique improves the efficacy of multiple cores in distributed systems. Optimal approaches utilized the Python programming language, an Intel Xeon 2.5 processor, and the gun/linux 4.8.0-36 operating system.

[34] demonstrates the potential of parallelism in this context through the application of an innovative algorithm to the statistical problem of density estimation and a traditional numerical approach to solve a differential equation. It is a challenging endeavor to resolve a complex problem using a solitary process and processing. Mainframe computers, for instance, execute a large number of processes within a single unit of time by employing multiple parallel processing units. This paper presents a method for parallel processing that divides a process into ever-smaller threads and portions, which can then be executed concurrently across multiple CPUs. Advanced Java forms the foundation of their emulator.

The algorithms used in [35] are able to compute theCPU and total execution durations, as well as started, terminated, consumed, and consumed while using the CPU of both servers and clients. Their research develops adaptable algorithms to facilitate efficient communication between the client and server sides, thereby addressing the challenges posed by hardware networking components and message transmission issues. Additionally, they address an enhanced method for problem subdivision in balanced form. To illustrate the impact of balance load division on their methodology, they utilized a matrix algebra case study. Special programming-checking subroutines examine and monitor the obtained results throughout numerous testing iterations; this ensures that the results are accurate to a significant degree.

[36] An architecture is described for a system comprising two main elements: monitoring and managing programs that are executed on distributed-multi-core architectures featuring 2, 4, and 8 CPUs with the purpose of achieving a designated objective. Regarding the resolution of issues, the network possesses the capability to accommodate a single client in addition to multiple servers. In the phase of implementation, it is critical to take into account three distinct scenarios that represent the vast majority of design alternatives. In addition to calculating the Total-Task-Time (TTT) on the client side, the proposed system is capable of determining the Startted, Elapsed, CPU, Kernel, User, Waiting, and Finish times of all pertinent servers. The subsequent creation scenario is meticulously deliberated upon during the process of designing User Programs (UPs): A single process is executed by multiple threads under the "single-process-multi-thread"(SPMT) computing paradigm. An increase in processing capacity is unambiguously associated with a corresponding increase in the rate at which problems are resolved, as demonstrated by the results. This specifically concerns the quantity of servers and the allocation of processors per server. As a result, the amount of time necessary to complete the task was multiplied by 9.156, contingent on three unique situations involving SPMT UPs. The engineering procedure for the implementation of their system is conducted using the C# programming language.[37] Implements Parallel Processing utilizing an application-oriented strategy for shared memory systems. The applications of such systems are capable of handling numerous duties through the utilization of Application-programming, which is based on the principles of shared memory parallel processing. Assuming a parallel processing approach, the effects of forcing processes among processes of a shared memory system are described. Computed total and CPU execution durations are impacted by these factors. In addition to the load size and the quantity of participating CPUs, the CPU utilization is also ascertained by its fluctuating manner.

In this [38] As the subject of this study, the Wisconsin female breast cancer tumor data set was utilized. A breast cancer classification prediction model was proposed by integrating the Random Forest and AdaBoost algorithms, which can provide a benign or malignant diagnosis. Finally, the model was compared to decision tree, K-nearest Neighbor, Support Vector Machine, and logistic regression algorithms. In comparison to the single algorithm models, the ensemble model outperformed the single algorithm models by an average of 4.3% in prediction accuracy. The ensemble model achieved the greatest increase of 9.8%, establishing itself as a new benchmark for predicting breast cancer.

[39] A hybrid parallel and distributed AdaBoost algorithm was developed, which leverages the several processors of a CPU through the utilization of lightweight threads. Additionally, it employs multiple machines through the implementation of a web service software architecture, thereby attaining

exceptional scalability. Up to the number of processors at their disposal, they accomplish a virtually linear increase in speed using a novel distributed architecture based on hierarchical web services. They achieve a 95.1% speedup on 31 workstations, each equipped with a quad-core processor, resulting in a learning time of 4.8 seconds per feature, as opposed to the 2.66 percent speedup attainable on four nodes using a single-level master-slave parallel and distributed implementation in the previously published work.

[40] Effective Adaptive Boosting (eAdaBoost), a meta-classifier created by augmenting the AdaBoost algorithm to generate the highest classification accuracy and manage time complexity, is introduced as a novel ensemble method. Through the process of reweighing each feature, eAdaBoost achieves optimal accuracy and a reduction in error rate in comparison to existing methods. Using datasets from the UCI machine learning repository, the comparison outcomes of a comprehensive experimental evaluation of the proposed method are described. Various boosting algorithms are employed to assess the accuracy of the classifiers and conduct statistical test comparisons. Various decision tree classifiers, including C4.5, Decision Stump, NB Tree, and Random Forest, have also been integrated with the proposed eAdaBoost. A performance evaluation of the algorithm is provided after it has been executed on a variety of datasets using distinct weight thresholds. For a subset of datasets, the proposed method outperforms the decision stump and C4.5 classifiers when random forest and NB tree are used as base classifiers. In comparison to other classifiers, eAdaBoost provides improved accuracy in both classification and prediction, as well as a shorter execution time.

[41] analyzes and compares a dataset pertaining to breast cancer through the implementation of classification decision tree algorithms. The following decision tree algorithms are implemented: J48, Best First, Function Tree, Random Forest Tree, AD Alternating Decision Tree, and Decision Stump. These decision tree algorithms are categorized in a computationally efficient manner using the development program Waikato Environment for Knowledge Analysis (WEKA), which comprises a collection of machine learning algorithms. These masses comprised 569 cases, of which 357 were benign and 212 were malignant, each with 32 attributes. Their purpose was to demonstrate and evaluate the distinctions between the various classification methods or algorithms. These outcomes were obtained through a process that entails setting aside a specific sample of a medical dataset for model training purposes. The decision tree classification forms exhibit a reduced average error rate and a significantly higher precision of 97.7% in correctly classifying breast tumor cases. The model with the lowest predicted accuracy in correctly classifying instances according to the decision stump algorithm is 88.0%.

D. Research Methodology

Extensive effort has been devoted to perfecting medical diagnosis through the use of conventional techniques, which typically entail feature extraction and classification. On the contrary, there has been a growing trend among scientists to utilize machine learning algorithms for cancer classification in recent years, owing to their capacity for learning. Figure 1 illustrates the methodology employed in the proposed approach.

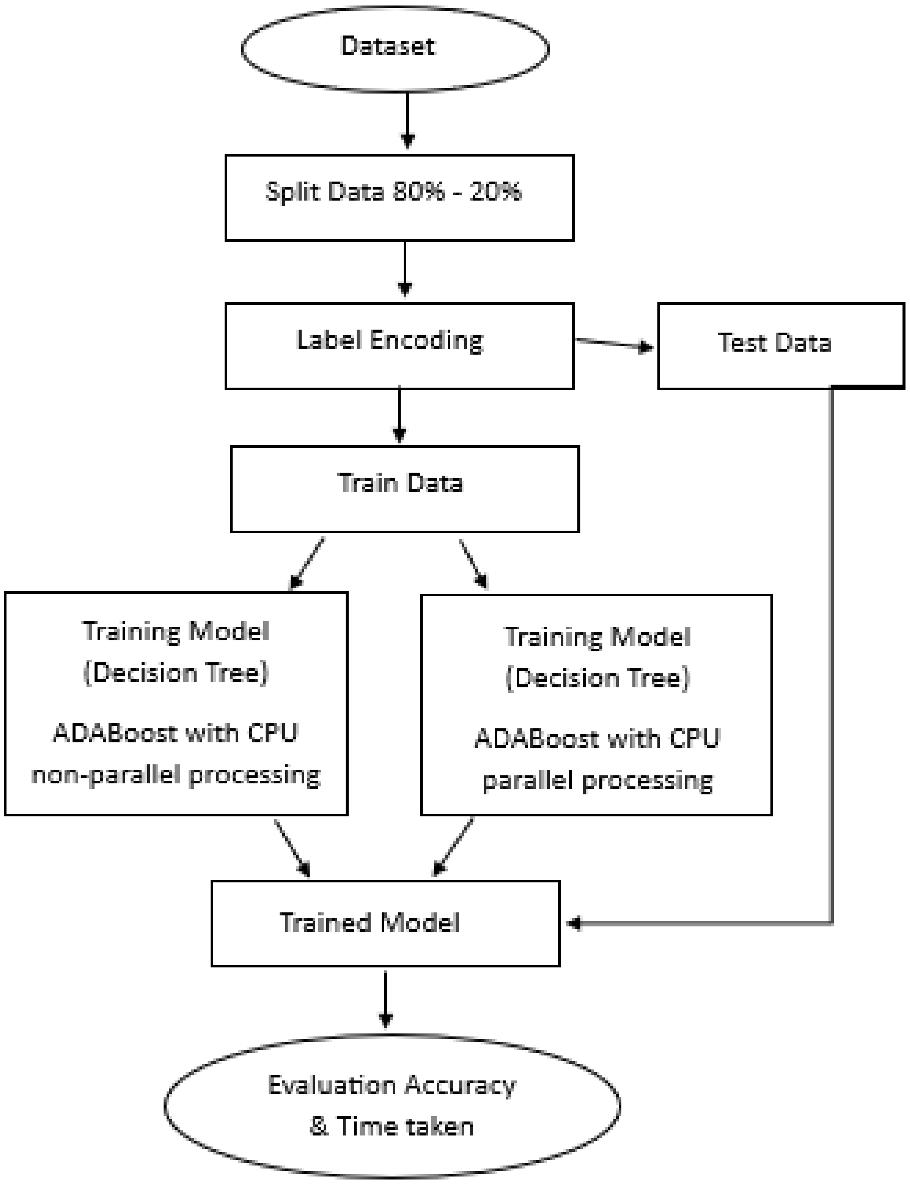
Dataset
Split Data 80% - 20%
Label Encoding
Test Data
Train Data
Training Model (Decision Tree) ADABoost with CPU non-parallel processing
Training Model (Decision Tree) ADABoost with CPU parallel processing
Trained Model
Evaluation Accuracy & Time taken

Figure 2. Flowchart for our proposed model.

Dataset Description

Broad-based, sci-kit-learn is utilized for a range of machine learning tasks. Its extensive community support and well-documented API make it a popular option among Python ecosystem machine learning novices and experts alike. There are numerous open-source studies that examine the classification of breast cancer. These studies utilize the WBCD database and implement a range of mathematical techniques, statistical models/algorithms, machine learning strategies, and data mining methods. The breast cancer dataset undergoes a straightforward and traditional binary classification, as detailed in Table 1. Predefined splitting's of "malignant" and "benign" are utilized to partition the dataset as follows: 80% is allocated for training purposes, while the remaining 20% is designated for testing.

Table 1. Dataset Description

Properties	Description
Samples per class	212(M), 357(B)
Samples total	569
Dimensionality	30
Features	Real, positive
Classes	2

Pre-Processing

The implementation of pre-processing is an essential phase when working with machine learning algorithms that demand numerical input. Dimensionality reduction, cleansing, encoding, and scaling are a few of the operations that comprise data preprocessing. Data cleansing is the procedure by which problems with data are resolved. Outliers, duplicate entries, and erroneous data types are typical components of raw data. The purpose of data cleansing is to eliminate issues such as absent values, inconsistencies, and chaotic data. While evaluating the model in its original (untransformed) space, this study implements the LabelEncoder pre-processing technique to transform the prediction target for learning. Encode target label values from 0 to n_classes-1 in this location. As opposed to the input X, this transducer ought to be utilized to encode target values, y.

Decision Tree

The decision tree method, also referred to as the Classification and Regression Tree (GINI), is a widely preferred and easily comprehensible machine learning algorithm. A decision tree of choices, as its nomenclature suggests, partitions the data space into more manageable subspaces, attributing labels or probabilities to each one. In order to optimize the execution of each split, the algorithm evaluates every possible split along every axis while the tree is being trained. A number of metrics, including entropy, information gain, and Gini, can be used to quantify the impurity of the two resulting partitions; the optimal split point is identified as the one with the lowest impurity among them. As one of the most precise and time-efficient classifiers, the technique is known for its tree-like design and configuration. In reality, numerous academics have widely implemented it; the decision tree is considered to be among the most straightforward classifiers to utilize. It is created and designed on the basis of data entropy.

AdaBoost Algorithm

Viola and Jones [42] Implemented a variant of the AdaBoost algorithm for both feature selection and classifier training in an effort to develop a rapid and dependable method for object detection. AdaBoost is one supervised learning scheme utilized to improve a basic learning algorithms classification performance. Their approach incorporates a feature extraction technique known as theHaar-like feature to support feeble classifiers. [43].

A classification algorithm based on AdaBoost can attain precise results while minimizing computational effort, provided that suitable features are utilized. By utilizing the AdaBoost learning algorithm, the classification performance of a basic learning algorithm is enhanced. By integrating numerous weak classifiers, this algorithm generates a robust classifier. The strong classifier is computed by linearly combining the weighted outputs of these weak classifiers. A large number of positive and negative example images are used to train the weights of the weak classifiers. [44].

The basic idea introduced by [45] is that a combination of single rules or "weak classifiers" gives a "strong classifier." Each sample is defined by a feature

vector $x = (x_1, x_2, \dots x_D)^T$ in a D-dimensional space and its corresponding class: C(x) = y ∈ {−1, +1} in the binary case. We define the weighted learning set *S* of *P* samples as:

$$S = \{(x_1, y_1, w_1), (x_2, y_2, w_2), \dots (x_p, y_p, w_p)\} \tag{1}$$

where *wi* is the weight of the *ith* sample.

In each iteration of the procedure, the optimal weak classifier is identified, denoted by the classifier with the smallest error. When the weak classifier consists of a single threshold, each threshold is evaluated.

Following each iteration, the incorrectly categorized samples are assigned greater weights, while the correctly classified samples are assigned lighter weights.

The final class *y* is given by

$$y(x) = sgn(\textstyle\sum_{t=1}^{T} a_t\, h_t\,(x)), \tag{2}$$

where both *αt* and *ht* are to be learned by the boosting procedure

The characteristics of the classifier we have to encode in the architecture are the coefficients *αt* for *t* = 1, . . . ,*T*, and the intrinsic constants of each weak classifier *ht* [46].

In order to determine how CPU parallel processing affects model performance, two AdaBoost classifiers are trained and evaluated: Indicated by the value of n_jobs to 1, the initial AdaBoost classifier is configured to function in the absence of CPU parallelism. This parameter limits the model's utilization of the CPU core to a single instance for both training and prediction. The configuration of the second AdaBoost classifier makes use of all available CPU cores in order to optimize computational efficiency through the utilization of CPU parallel processing. This is accomplished by configuring AdaBoost to utilize multiple CPU cores for parallelized computation via the n_jobs parameter, which is set to -1.

Evaluation Metrics

Two primary metrics (accuracy and training time) were employed to assess the performance of the model in this approach. In order to evaluate two implementations, one that utilized Python's time module for precise time measurements and the other that utilized parallel processing on the CPU were contrasted. The parallelized iteration sought to leverage the computational capabilities of multiple CPU processors, whereas the absence of parallel processing served as a benchmark for comparison.

E. Results

The experimental test results for the proposed model are described in this section. The evaluation of the predictive capabilities of the adaptive boosting algorithm and decision tree is conducted using performance metrics including accuracy and time required. The investigation was carried out on a system equipped with a 2.30 GHz Intel Core i7-10510U processor and 16GB of RAM.

Accuracy

On the breast cancer dataset, both AdaBoost models were evaluated; one model utilized parallel processing on the CPU and the other did not. The findings revealed noteworthy variations in their levels of performance. Both models performed admirably when it came to classifying cancer samples into premade categories. Both models exhibit efficacy in the precise classification of cancer samples, as evidenced by their respective Accuracy values of approximately (97.37%). In contrast to the non-parallel model, it is unexpected that the accuracy remains unaffected by the utilization of CPU parallel processing. This indicates that parallelism does not undermine classification accuracy for the given dataset and model configuration.

Computational Efficiency

Significant differences can be observed in the training and prediction times of models when contrasting the utilization of CPU parallel processing with and without the models. Computational efficacy refers to this phenomenon. In terms of the time required to train and predict the model, it is evident that the two AdaBoost configurations differ significantly. In contrast to the non-parallel model, which required approximately (0.3117 seconds) to complete the same duties, the parallelized model generated equivalent results in approximately (0.2898 seconds). The information provided indicates that the observed disparities in training and prediction timeframes between the two AdaBoost configurations can be attributed to two primary factors. To begin with, the duration of training and prediction may be significantly impacted by the dataset's size. In general, the duration needed for model training and prediction is longer for larger datasets in comparison to smaller ones. Additionally, an imbalanced data set, characterized by substantially fewer samples for some classes compared to others, may have an effect on the training procedure. Resolving class imbalances frequently necessitates supplementary calculations in order to guarantee equitable representation of minority classes, which may have an impact on the total time investment.

F. Discussion

The results of this research shed light on an intriguing circumstance wherein the two AdaBoost models achieve equivalent accuracy in classifying breast cancer samples despite significant disparities in computational efficiency. While both models attain an accuracy rate of 97.37%, the model that implements CPU parallel processing accomplishes a substantial reduction in processing time. This approximately 7.04% reduction in training time demonstrates how parallelization techniques enhance computational efficiency. An additional factor that could be observed in our comparative analysis is whether the dimensionality of the feature space has an effect on the efficacy of algorithms or whether specific models are more sensitive to the quantity of features.

Further Analysis and Considerations

While the primary focus of the study is to enhance performance and accuracy, it could be even more enlightening to investigate alternative parallel processing methodologies or hyperparameters in order to optimize model performance and resource consumption. Furthermore, further investigation is necessary to assess the scalability of parallelism in the context of larger datasets characterized by a greater quantity of samples or variations in dimension.

G. Conclusion

A breast cancer dataset obtained from the Sklearn Data Repositories was utilized to develop a predictive model for breast cancer in this study. The algorithm utilized is a decision tree and adaptive boosting model. By employing two AdaBoost configurations—one with and one without CPU parallelism—our principal objective in this study is to enhance the performance of the AdaBoost algorithm in breast cancer. Comparative analysis was conducted using the evaluation metrics of training duration and accuracy. The effective classification of breast cancer samples was achieved by both AdaBoost models, as evidenced by the 97.37% Sklearn.Datasets.Load_breast_canceracure rate of the data they generated.

Unexpectedly, the addition of CPU parallel processing did not impair AdaBoost's predictive capability, demonstrating its durability. Conversely, the most significant disparity was noted with regard to computational efficiency. When testing time was compared between the parallelized and non-parallel models, the parallelized model exhibited a reduction of 7.04%. Our discovery underscores the potential benefits that practitioners applying similar classification tasks may derive from parallelization as a means to improve the computational efficiency of AdaBoost.

In order to optimize model performance and resource utilization, additional research is necessary to determine how parallelism can be made to function more effectively with more complex data sets and to determine its scalability. In summary, this research contributes to the existing body of knowledge regarding the performance of AdaBoost in different processing conditions. It illuminates the intricate correlation between accuracy and computational efficiency in the context of breast cancer classification. Subsequently, we intend to develop a learning framework that is both adaptable to various machine learning models and has the capability to reduce processing time.

H. References

[1] 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.

[2] 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.

[3] S. Merler, B. Caprile, and C. Furlanello, “Parallelizing AdaBoost by weights dynamics,” *Comput Stat Data Anal*, vol. 51, no. 5, pp. 2487–2498, 2007.

[4] 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.

[5] 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.

[6] 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.

[7] P. Collet, “Why GPGPUs for evolutionary computation?,” in *Massively Parallel Evolutionary Computation on GPGPUs*, Springer, 2013, pp. 3–14.

[8] 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.

[9] 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.

[10] 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.

[11] Y. Zeng, W. Wang, Y. Ding, J. Zhang, Y. Ren, and G. Yi, “Adaptive Distributed Parallel Training Method for a Deep Learning Model Based on Dynamic Critical Paths of DAG,” *Mathematics*, vol. 10, no. 24, p. 4788, 2022.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] C. Ying, M. Qi-Guang, L. Jia-Chen, and G. Lin, "Advance and prospects of AdaBoost algorithm," *Acta Automatica Sinica*, vol. 39, no. 6, pp. 745–758, 2013.

[22] S. Salcedo-Sanz *et al.*, "Analysis, characterization, prediction and attribution of extreme atmospheric events with machine learning: a review," *arXiv preprint arXiv:2207.07580*, 2022.

[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] 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.

[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] 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.

[27] K. Jacksi, N. Dimililer, and S. R. M. Zeebaree, "A survey of exploratory search systems based on LOD resources," 2015.

[28] 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.

[29] 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.

[30] 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.

[31] B. Mahesh, "Machine learning algorithms-a review," *International Journal of Science and Research (IJSR).[Internet]*, vol. 9, no. 1, pp. 381–386, 2020.

[32] M. Amrane, S. Oukid, I. Gagaoua, and T. Ensari, "Breast cancer classification using machine learning," in *2018 electric electronics, computer science, biomedical engineerings' meeting (EBBT)*, IEEE, 2018, pp. 1–4.

[33] D. M. Abdulqader and S. R. M. Zeebaree, "Impact of Distributed-Memory Parallel Processing Approach on Performance Enhancing of Multicomputer-Multicore Systems: A Review," *QALAAI ZANIST JOURNAL*, vol. 6, no. 4, pp. 1137–1140, 2021.

[34] M. Kumar, M. S. Jain, and M. Snehalata, "Parallel Processing using multiple CPU," *International Journal of Engineering and Technical Research (IJETR)*, vol. 2, 2014.

[35] S. R. Zebari and N. O. Yaseen, "Effects of parallel processing implementation on balanced load-division depending on distributed memory systems," *J. Univ. Anbar Pure Sci*, vol. 5, no. 3, pp. 50–56, 2011.

[36] 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.

[37] S. R. Zeebaree and K. Jacksi, "Effects of processes forcing on CPU and total execution-time using multiprocessor shared memory system," *Int. J. Comput. Eng. Res. Trends*, vol. 2, no. 4, pp. 275–279, 2015.

[38] D. Yifan, L. Jialin, and F. Boxi, "Forecast model of breast cancer diagnosis based on RF-AdaBoost," in *2021 International Conference on Communications, Information System and Computer Engineering (CISCE)*, IEEE, 2021, pp. 716–719.

[39] M. Abualkibash, A. ElSayed, and A. Mahmood, "Highly scalable, parallel and distributed adaboost algorithm using light weight threads and web services on a network of multi-core machines," *arXiv preprint arXiv:1306.1467*, 2013.

[40] S. Dinakaran and P. R. J. Thangaiah, "Ensemble method of effective AdaBoost algorithm for decision tree classifiers," *International Journal on Artificial Intelligence Tools*, vol. 26, no. 03, p. 1750007, 2017.

[41] N. K. Al-Salihy and T. Ibrikci, "Classifying breast cancer by using decision tree algorithms," in *Proceedings of the 6th International Conference on Software and Computer Applications*, 2017, pp. 144–148.

[42] M. Hambali, Y. Saheed, T. Oladele, and M. Gbolagade, "ADABOOST ensemble algorithms for breast cancer classification," *Journal of Advances in Computer Research*, vol. 10, no. 2, pp. 31–52, 2019.

[43] M. Hiromoto, K. Nakahara, H. Sugano, Y. Nakamura, and R. Miyamoto, "A specialized processor suitable for AdaBoost-based detection with Haar-like features," in *2007 IEEE conference on computer vision and pattern recognition*, IEEE, 2007, pp. 1–8.

[44] M. Hiromoto, H. Sugano, and R. Miyamoto, “Partially parallel architecture for adaboost-based detection with haar-like features,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 1, pp. 41–52, 2008.

[45] Y. Freund and R. E. Schapire, “Experiments with a new boosting algorithm,” in *icml*, Bari, Italy, 1996, pp. 148–156.

[46] J. Mitéran, J. Matas, E. Bourennane, M. Paindavoine, and J. Dubois, “Automatic hardware implementation tool for a discrete adaboost-based decision algorithm,” *EURASIP J Adv Signal Process*, vol. 2005, pp. 1–12, 2005.