Al-Rafidain Journal of Computer Sciences and Mathematics (RJCSM)

www.csmj.uomosul.edu.iq

Reducing Execution Time of Pixel-Based Machine Learning Classification Algorithms Using Parallel Processing Concept

Aliaa Shaker Mahmoud[1] and Mohammed Chachan Younis[2]

[1] Department of Computer Science, College of Computer Science and Mathematics, University of Mosul, Iraq
[2] Department of Artificial Intelligence, College of Computer Science and Mathematics, University of Mosul, Iraq
Email: aliaa.23csp42@student.uomosul.edu.iq[1] , mohammed.c.y@uomosul.edu.iq[2]

Article information

Article history:
Received 22 December ,2024
Revised 10 January ,2025
Accepted 30 January ,2025
Published 26 June ,2025

Keywords:
Decision Tree, Machine Learning, Parallel Processing, Random Forest

Correspondence:
Aliaa Shaker Mahmoud
Email:
aliaa.23csp42@student.uomosul.edu.iq

Abstract

Parallel processing is essential in machine learning to meet the computational requirements resulting from the complexity of algorithms and the size of the dataset, by taking advantage of the computational resources of parallel processing that can distribute computational operations across multiple processors. Which contributes to significant improvements in performance and time efficiency. This research demonstrated the impact of parallel processing on the performance and time efficiency of machine learning for pixel-based image classification techniques. The methodology includes pre-processing the Oxford IIIT Pet dataset, from which 4 cat images were selected. The performance of two supervised machine learning classifiers, decision tree, and random forest (10, 100, 500, and 1000 trees) were compared and implemented in two ways with and without parallel processing. The data is split in two ways: the first is by splitting the data by 70% for training data and 30% for testing data and the second is by cross-validation by splitting the data into four folds. The research aims to compare the accuracy and timely scales of machine learning models with and without parallel processing. The results showed a strong predictive power of the algorithms with an accuracy of 97.5%, while the training times were significantly reduced in parallel from 88.83 to 15.88 seconds for the RF100 model. This reflects the effectiveness of parallel processing in improving the performance of machine-learning models for pixel-based image classification. The proposed system was programmed using MATLAB 2021 language tools. The work was carried out on a computer running Microsoft Windows 11 operating system with an Intel(R) Core (TM) i5-1135G7@2.40GHz 2.42GHz processor with 8 GB of RAM.

DOI: 10.33899/csmj.2025.156050.115,

1. Introduction

Most manufacturers have started producing multi-core processors since 2005 using parallel computing methods to maintain increased speed and performance [1][2]. The parallel processing approach derives from the philosophy of solving a problem by dividing complex problems into smaller tasks more efficiently, which can be controlled and managed [3][4][5]. Executing many parallel iterations of a specific algorithm separately is one of the most basic approaches to parallelizing algorithm evaluation [6]. Several researchers have proposed parallel processing approaches to improve the efficiency of machine learning algorithms, including design and implementation [7]. In machine learning, parallel processing becomes more important as difficult models and big datasets need a lot of accounting resources. Sequential training can hinder the development and deployment of machine learning models in terms of time consumption in real-world scenarios. By computational task allocation to several processors or even multiple computers, parallel processing expedites model training and lowers the overall time needed [8][9]. According to primary factors that convert user and programmer concerns into practical solutions, parallel processing takes less time than sequential processing [10]. Multicore processors are becoming more common,

although their compact parallel processing capacity cannot be well exploit until the software being developed is enhanced [11][12]. To improve the performance of multi-core computers, a program has to be executed in several parallels on a greater number of cores. Many processes execute on multicores, and the technological limitations of a single core, like issues related to throughput, high efficiency of energy, and extended battery life, have made multi-core processors more important [13]. For systems that process huge amounts of data for analysis, processing time decreases are necessary for enhancing efficiency. Big amounts of data lead to longer processing times. Therefore, it is important to reduce the duration of these processes [7].

The objectives of this research are to apply the concept of parallel processing to reduce the time of executing machine learning algorithms as follows:

1. Load the original images with their ground truth and data pre-processing by drowning the ground truth colours in RGB to be closer to the programming concept using the three basic colours (red, green, and blue).
2. Training classification models of machine learning DT and RF (10, 100, 500, 1000 trees). Then the implementation is done in two different ways, one by sequential processing and the other by parallel central processing.
3. The data was divided into 70% for training and 30% for testing for learning models with 10 rounds to get rid of problems related to random data selection models, which may result in differences when retraining.
4. The cross-validation method was also programmed and implemented, where the data was divided into four groups for training and testing.
5. The performance of the models trained on the test data was evaluated and compared through the accuracy measure and the time taken measure by displaying confusion matrices and predicted images for both methods followed in this research (sequential processing and parallel processing).

2. Literature Review

This section discusses the literature related to parallel processing and machine learning. The research [14] Presents an earthquake prediction model using parallel CPU to improve learning performance. Instead of relying on traditional CPU, GPU is used by CUDA framework, The outstanding hybrid state machine (H-SVM) algorithm is implemented on parallel CPU, and the results indicate that the use of GPU which speeds up by 3-70 times compared to CPU processing.

Boukhalfa et al. proposed [15] An approach based on storing and analyzing network data using machine learning algorithms (KNN, SVM, and DT) in a distributed and parallel manner. The research showed high efficiency of the KNN algorithm accuracy of up to 99.9%, with a decrease in processing time from 1792.8; to 1659.4 seconds for classification of 23 classes.

The research [16], a GPU coupled with a deep convolutional neural network (DCNN) was used to detect ultra-short-period planets (USP). It was trained on 2 million samples. The results showed an accuracy of up to 99.5%. This method is 1000 times faster in processing the optical light curve compared to the conventional least squares method without compromising accuracy.

The research [17] Examined the effect of parallelism in tuning hyperparameters on a fake news detection dataset using CV from sci-kit-learn to tune a random forest classifier. The results showed a slight change in model accuracy from 99.26% to 99.15%. The CPU times were five times faster compared to sequential processing.

The research [18] Parallel architecture using multi-core CPUs to accelerate the performance of machine learning models. It focused on RF, XGBoost, AdaBoost, and KNN models. The results showed a 1.7x and 3.8x faster performance improvement for both small and large datasets on quad-core CPUs, without compromising accuracy.

This research [19] Aims to use parallel processing using Python on breast cancer dataset, AdaBoost model and DT algorithm were combined to improve performance and reduce processing time. The results showed the accuracy was not affected by parallel processing, which is 97.37%. While the training time was reduced by 7.04% when implementing parallel processing compared to sequential processing.

The research [20] Parallel processing was used in training SVM and RF models to improve classification accuracy and reduce training time. The results showed that the 100-tree RF model increased classification accuracy from 58.87% to 63.59%. The training time was also significantly reduced using the MPI4Py parallel processing interface from 1725.8 to 396.5ms.

The research [21] Focused on the effect of parallel processing on the performance of the Random Forest algorithm using the CIFAR-10 dataset. The results showed an accuracy of 97.50% for both sequential and parallel processing. While the training time was reduced from 0.6187 to 0.4753 seconds.

The research [22], the impact of parallel processing on the performance of the Extra Tree classifier was evaluated. The results showed a slight improvement in accuracy from 88.23% to 88.43% when parallel processing was applied. Also, parallel processing significantly reduced the computation time, from 37.463 to 4.837 seconds.

This research [23] investigated the effect of parallel processing on the performance of the LightGBM algorithm

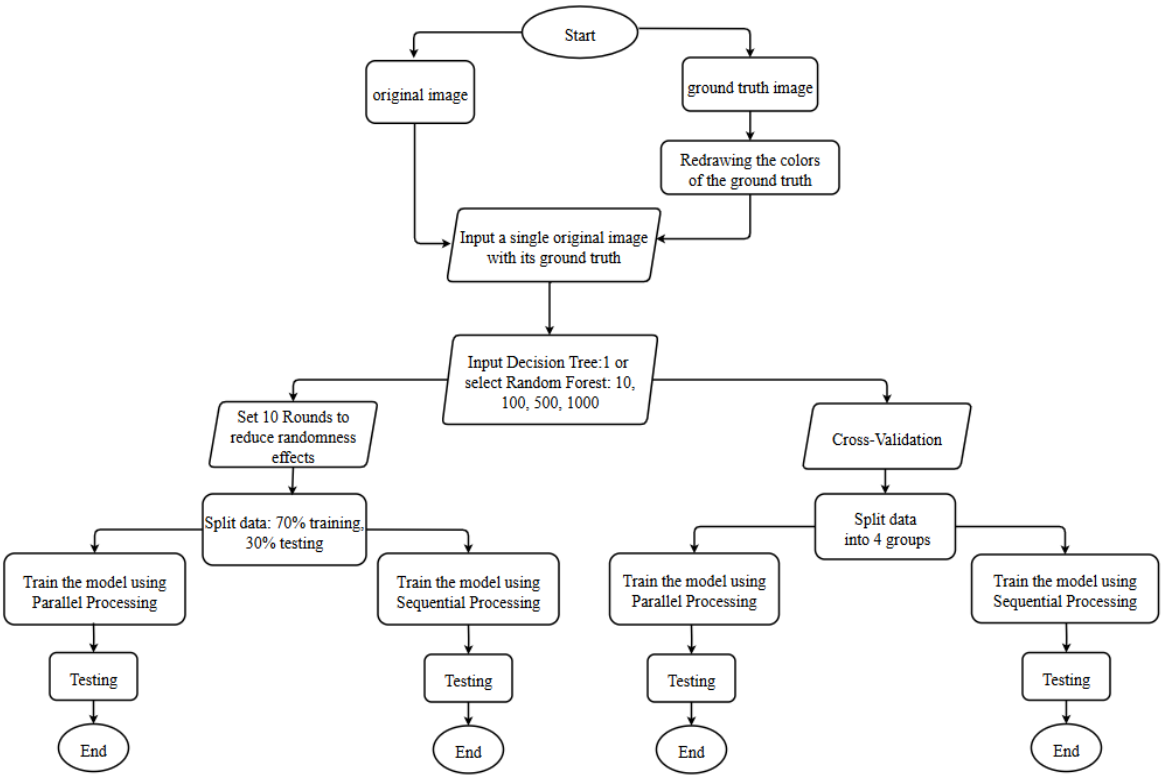

Figure 1. Process flow of the proposed approach

using the IRIS dataset. The results showed that the training efficiency was significantly improved while maintaining 100% classification accuracy for both parallel and non-parallel processing. While the training time was reduced from 0.2316 to 0.1921 seconds when using parallel processing.

The research [24] Investigated the classification of newspaper articles using the XGBoost algorithm. The results showed an accuracy of 79.83% in both parallel and non-parallel processing, with parallel processing contributing to a significant reduction in training time from 501,319 to 264,978 seconds.

The research [25] Presented an analysis of the effect of parallel processing on a Random Forest model using the Apple M1 chip. The results showed a high accuracy of 100% without the effect of parallel processing. The training time was significantly reduced from 1.4956 to 0.3758 seconds for parallel processing.

The research [26] Used parallel processing of the Bison algorithm using the PySpark framework to address classification problems. The results showed a high accuracy of 97%, with a significant improvement in execution time efficiency; the processing time was reduced from 25946.03 seconds to 270.63 seconds.

3. Methodology

In this research, the utilization of the parallel processing concept to reduce the execution time of machine learning algorithms for pixel-based image classification as in the methodology of the workflow plan shown in **Figure 1**.

1. Start.

2. Load the original images with their ground truth and data pre-processing by drowning the ground truth colours in RGB to be closer to the programming concept using the three basic colours (red, green, and blue).

3. Input a single image with its ground truth to the program.

4. Choose an algorithm: a decision tree or a random forest as a (10, 100, 500, and 1000) tree.

5. Set 10 rounds to reduce the randomness effect.

6. Data splitting to the training set (70%) and the testing set (30%).

- Training The model using Parallel Processing.
- Training The model using Sequential Processing.

7. Cross-validation: split data into four sets for training and testing.

- Training The model using Parallel Processing.
- Training The model using Sequential Processing.

8. Testing data.

9. End of the classification process.

3.1 ParFor Function in Matlab

ParFor is a robust tool that lets users divide loops through parallel processes, significantly accelerating code performance [27]. ParFor is a parallel variant of the standard for function, enabling the loop iterations to be conducted more quickly on computers with multiple cores by distributing them among several parallel processes rather than implementing them sequentially [28]. ParFor can be used in the following cases If the loop contains computations that take a long time to complete. Also, calculating large matrices or repeating computations, ParFor is useful for distributing computations across multiple processors or cores to speed up the calculation. When the loop operations are independent of each other, meaning that the result of one iteration does not depend on the result of the previous iteration [29]. The ParFor function in MATLAB speeds up the execution of loops by dividing the computation into smaller, parallel tasks [27]. Imagine that you have a long list of tasks. Instead of executing these tasks one by one, ParFor divides the list into smaller groups and assigns each group to a different "worker". Each worker performs its assigned tasks independently and at the same time. After all the workers have completed their tasks, the final results from each worker are summed to form the overall result of the loop. **Figure 2**: shows Pool of MATLAB Workers.

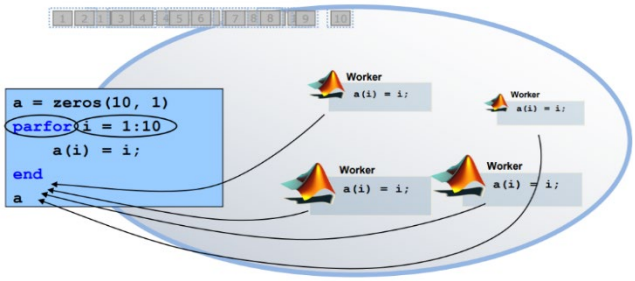

Figure 2. Pool of MATLAB Workers [29]

4. Dataset

The Oxford-IIIT Pet dataset, a dataset consisting of (7349) cat images of (37) different breeds, was used. It was obtained from the web and is an open-source data [30][31][32]. The original images included various cat categories, along with their ground truth images. However, one of the objectives of this research is to train and test pixel-based machine-learning models. Therefore, four cat images were carefully selected for quality and accuracy as training and testing models in this research. The ground truth images consist of three categories: cat, border, and background, which are characterized by different colours. To make the ground truth images closer to the software concept for this research, the primary colours (red, green, and blue) were used. **Figure 3** shows images used in this research.

4.1 Data Pre-processing

The ground truth images consist of three distinct classes: cat, border, and background. The background classes, cat border classes, and cat body classes were changed to be more consistent with the programming concept of using RGB colours. The ground truth images consist of three classes (red, green, and blue). **Figure 4** shows redrawing the colour of ground truth, where the left image is before redrawing, and right after redrawing.

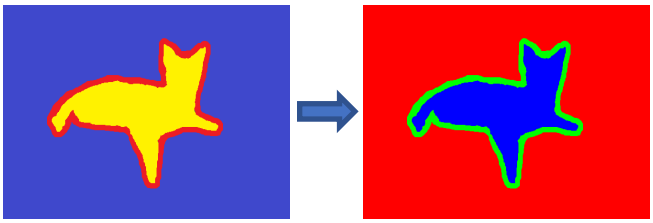

Figure 4. Redrawing the colour of ground truth, where the left image is before redrawing, and right after redrawing

There was no need for both the original image and ground truth and wisdom because pixel wisdom can come from ground truth to image. The images used in this section are represented using RGB during processing. To process each pixel, whether training or testing, concerning the ground truth, labels were predefined for each pixel class where the

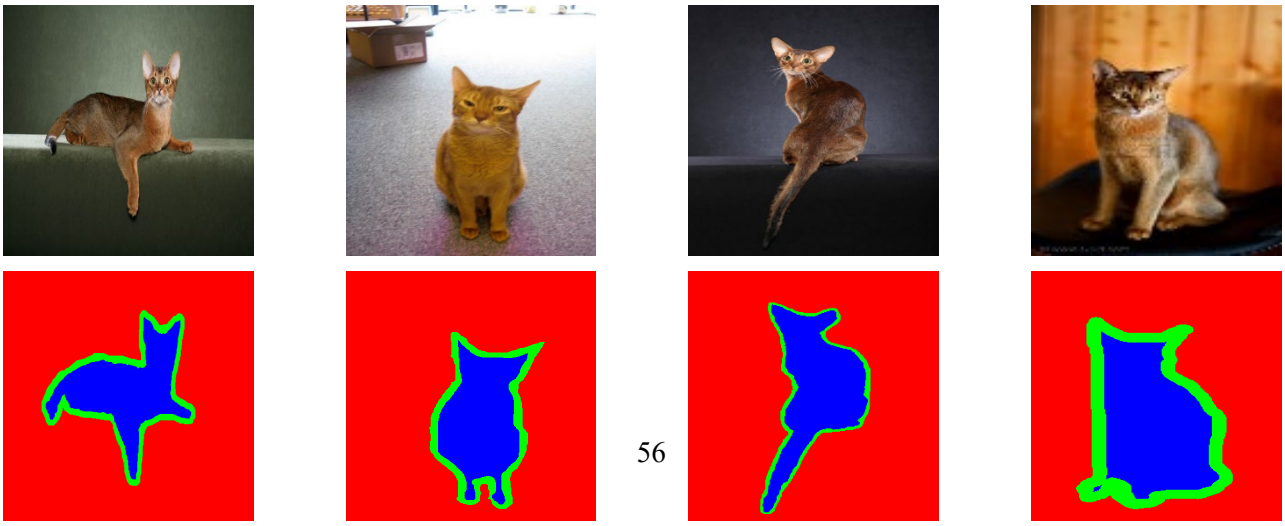

Figure 3. Samples of image used in this research

background was defined as red, the body as blue, and the shape boundaries as green.

5. Performance Metrics

A confusion matrix is a tool for evaluating the performance of classification models in machine learning, comparing the predicted results with the actual results [33][34]. The matrix consists of four cells in the case of binary classification [35][36]. **Figure 5**: shows confusion matrix.

The purpose of using accuracy in research is appropriate because it is a comprehensive measure for evaluating the performance of models and is a simple and effective indicator, especially for problems related to the balanced distribution of classes. It also highlights the impact of technical improvements on the quality of the results, showing that speeding up the processes does not affect the quantitative performance the models.

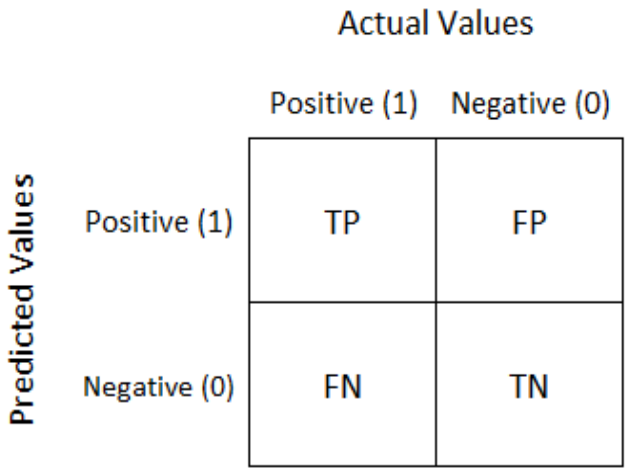

Figure 5. Confusion Matrix

Performance evaluation metrics using the confusion matrix include [37][38][39]:

1. Accuracy: The ratio of correct predictions to total.

$$\text{Accuracy} = \frac{tp + tn}{tp + fn + fp + tn} \quad (1)$$

2. Precision: The ratio of correctly classified positives to positives.

$$\text{Precision} = \frac{tp}{tp + fp} \quad (2)$$

3. Recall/Sensitivity: The ratio of correctly classified positives to actual positives.

$$\text{Recall} = \frac{tp}{tp + fn} \quad (3)$$

4. F1 Score: A weighted average of precision and recall that balances them.

$$\text{F1-Score} = \frac{2 * \text{precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \quad (4)$$

6. Decision Tree and Random Forest

DT and RF algorithms are machine learning algorithms. These can be used to solve classification and regression problems. DT structure consists of nodes, branches, and leaves. Nodes represent decision points that contain tests or conditions, branches represent the results of those tests, and leaves represent the final results [40][41][42]. **Figure 6** shows the structure of the DT algorithm.

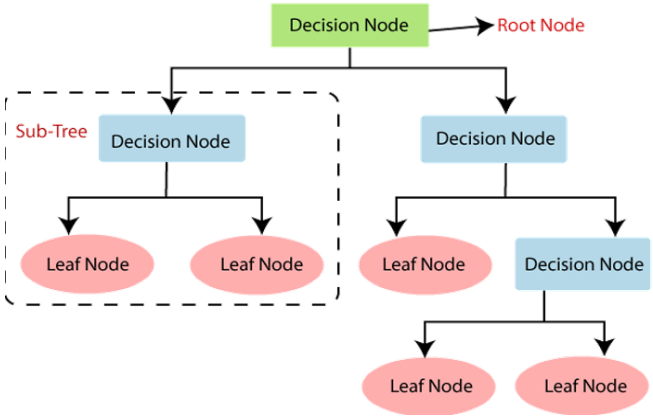

Figure 6. Structure of DT algorithm [43]

Algorithm 1: Pseudo code for the DT algorithm

1. **BuildTree (N):**
2. **If** N contains instances of only one class **then**
3. **return**
4. **else**
5. Randomly select x% of the possible splitting features in N
6. Select the feature F with the Gini index to split on
7. Create f child nodes of N, N_1, …., N_f, where F has f possible values (F_1, …, F_f)
8. **For** i = 1 to f **do**
9. Set the contents of N_i to D_i, where D_i is all instances in N that match
10. F_i
11. Call BuildTree (N_i)
12. **end for**
13. **end if**

Random Forest is based on the Ensemble Learning method, which improves the performance and accuracy, and controls the overfitting of models by merging multiple decision trees into a single model and merging their results to achieve more stable results. [44][45][46] **Figure 7** shows the structure of the RF algorithm.

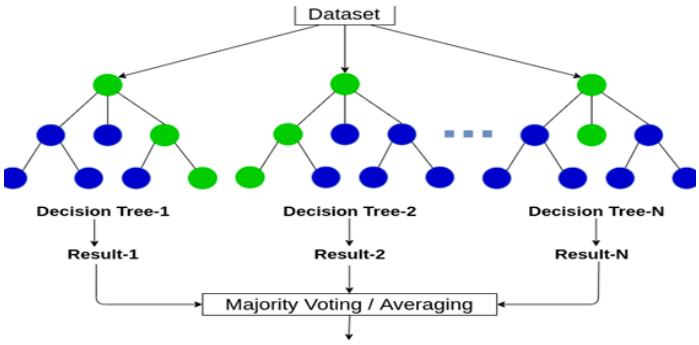

Figure 7. Structure of RF algorithm [40]

Algorithm 1: Pseudo code for the RF algorithm

1. To generate c classifiers:
2. **For** i = 1 to c **do**
3. Randomly sample the training data D with replacement to produce D_i
4. Create a root node, N_i containing D_i
5. Call BuildTree (N_i)
6. **end for**

7. **BuildTree (N):**
8. **If** N contains instances of only one class **then**
9. **return**
10. **else**
11. Randomly select x% of the possible splitting features in N
12. Select the feature F with the Gini index to split on
13. Create f child nodes of N, N_1, …., N_f, where F has f possible values (F_1, …, F_f)
14. **For** i = 1 to f **do**
15. Set the contents of N_i to D_i, where D_i is all instances in N that match
16. F_i
17. Call BuildTree (N_i)
18. **end for**
19. **end if**

7. Performance Evaluation

The tree trained on the training data samples is used to classify the pixels in the original image during the testing phase. Each pixel is classified as belonging to one of three classes: background, border, or shape, which are denoted by the numbers 1, 2, and 3, respectively. The prediction is based on the colour characteristics of each pixel, and the trained tree is used to evaluate only the test data pixels. After evaluation, the percentage of correctly classified pixels with their ground truth is calculated, which is an estimate of the classification accuracy. The accuracy percentage and the time taken to classify are important performance metrics used to evaluate classification models, making them essential criteria in this research.

Where the ($counter\ total$) is the total number of pixels that

$$Percentage\ \text{accuracy} = \frac{counter\ matched}{counter\ total} \text{x}100 \quad (6)$$

used in the test (the number of test data pixels), and the matching counter ($counter\ matched$) is the number of test data pixels that was evaluated correctly.

7.1 Results and Discussion

In the results section, we will analyze and present the outcomes of testing various models. Additionally, we will evaluate the models' performance on unseen data by generating confusion matrices, and accuracy scores using the testing dataset.

A standardized evaluation scheme was used in which the same test conditions and calculations were applied to multiple runs of the Decision Tree (DT) and Random Forest (RF) algorithms to compare the effectiveness of different classification methods. The results of all runs were averaged to reflect the most accurate statistical values, as the results can vary slightly with each run due to the random selection of pixels for training and testing. In the case of RF, the effect of the number of trees on classification performance was evaluated using four different forest sizes: 10 trees, 100 trees, 500 trees, and 1000 trees. The cases were randomly selected to represent low and high values of the number of trees. Since tree generation and testing are time-consuming and depend on the number of features and the size of the test data, the processing time for both DT and RF was improved by implementing parallel processing using the ParFor function in MATLAB. Two processing models were compared: the regular model (with a single core) and the parallel model (with multiple cores). In both models, the same training and testing conditions and image data were used, with only the processing mode differing. The results calculated include: classification accuracy (percentage of pixels correctly classified), average time per run, and confusion matrix (which compares the predicted data to the target). 70% of the image data was used for training and 30% for testing, with no overlap between training and testing data.

7.2 Performance Measurements

Table 1 shows a comprehensive comparison of the performance of pixel-based classification algorithms using two methods, sequential processing, and parallel processing using the ParFor function. By dividing the data into 70% for training data and 30% for testing data, the results showed the highest average accuracy of 96.4 for both sequential and parallel processing methods, while the average execution

time of the algorithms was 83.86 seconds and 791.50 seconds using sequential parallel processing. The highest accuracy achieved among the models is the random forest model with 100 trees for the images (1, 2, 3, 4) (96.7, 97.5, 97.2, 96.1) respectively. The lowest accuracy was for the decision tree model (96.2, 95.7, 96.9, 90.5) for images 1, 2, 3, and 4 respectively. While parallel processing significantly outperformed sequential processing in terms of execution time, the execution time decreased from (1.85, 1.99, 3.70, 0.46) to (1.43, 0.82, 2.28, 0.35) for images (1, 2, 3, 4) respectively. The results confirm the effectiveness and impact of parallel processing on the efficiency of the algorithms without compromising accuracy.

These results confirm that the random forest with 100 trees is the best choice in terms of balance between accuracy and execution time, and highlight the effectiveness of parallel processing in improving time efficiency without affecting the accuracy of classification, which makes it a practical solution for dealing with large data and complex models.

7.3 Confusion Matrix

Table 2 shows the confusion matrices of the RF100 classification model performance for both sequential and parallel processing methods. This classification parameter value gives the best performance among all settings according to **Table 1**, Confusion Matrices for images 1, 2, 3, and 4. Where classes 0, 1, and 2 in the table represent background, border, and shape, respectively.

For the first confusion matrix, the background class was correctly predicted 1,953,690 times and was misclassified as a border or shape 55,461 times. The border class was correctly predicted 69,274 times and was misclassified as a background or shape 12,608 times. The shape class was correctly predicted 298,688 times and was misclassified as both background and border 8,794 times. The overall accuracy of the RF classifier is 100 for both serial and parallel processing, similarly for the rest of the matrices.

Table 1. Comparison of accuracy and time taken between models to predict test images

Dataset		Normal		Parallel (par for) function	
	classifier	*Accuracy (%)*	*Avg _time[s]*	*Accuracy (%)*	*Avg_time[s]*
Image 1	DT	96.2	1.859	96.2	**1.03**
	RF (10)	96.6	6.086	96.6	**3.33**
	RF (100)	**96.7**	43.87	**96.7**	**29.28**
	RF (500)	96.7	174.72	96.7	**111.29**
	RF (1000)	96.7	447.43	96.7	**231.09**
Image 2	DT	95.7	1.99	95.7	**0.82**
	RF (10)	97.2	5.53	97.2	**2.11**
	RF (100)	**97.5**	88.83	**97.5**	**15.88**
	RF (500)	97.5	221.39	97.5	**74.57**
	RF (1000)	97.5	365.56	97.5	**174.3**
Image 3	DT	96.9	3.70	96.9	**2.28**
	RF (10)	97.1	13.41	97.1	**7.79**
	RF (100)	**97.2**	89.66	**97.2**	**54.45**
	RF (500)	97.2	422.89	97.2	**262.23**
	RF (1000)	97.2	13458.7	97.2	**550.30**
Image 4	DT	90.5	0.46	90.5	**0.35**
	RF (10)	95.1	2.66	95.1	**2.25**
	RF (100)	**96.1**	29.82	**96.1**	**9.73**
	RF (500)	96.1	185.63	96.1	**52.81**
	RF (1000)	96.1	265.84	96.1	**91.38**
Mean		96.4	791.50	96.4	83.86

Table 2. Percentages of normal and parallel avsa confusion matrix.

		Image 1 Normal				Image 1 Parallel			
		predicted value				predicted value			
	RF 100	Class0	Classl	Class2	**Total**	Class0	Classl	Class2	**Total**
Real value	Class0	1953690 81.40%	55461 2.30%	1485 0.10%	2010636 97.20% 2.80%	1953722 81.40%	55582 2.30%	1492 0.10%	2010796 97.20% 2.80%
	Classl	8851 0.40%	69274 2.90%	3757 0.20%	81882 84.60% 15.40%	8813 0.40%	69157 2.90%	3669 0.20%	81639 84.70% 15.30%
	Class2	49 0.00%	8745 0.40%	298687 12.40%	307482 97.10% 2.90%	55 0.00%	8741 0.40%	298768 12.40%	307565 97.10% 2.90%
	Total	1962590 99.50% 0.50%	133480 51.90% 48.10%	303930 98.30% 1.70%	1321652 / 2400000 96.70% 3.30%	1962590 99.50% 0.50%	133480 51.80% 48.20%	303930 98.30% 1.70%	1321648 / 2400000 96.70% 3.30%

	RF 100	Image 2 Normal				Image 2 Parallel			
		Class0	Classl	Class2	**Total**	Class0	Classl	Class2	**Total**
Real value	Class0	1499332 80.00%	19005 1.00%	2095 0.10%	1520432 98.60% 1.40%	1499115 80.00%	12385 0.70%	2195 0.10%	1513695 99.00% 1.00%
	Classl	7006 0.40%	86390 4.60%	1170 0.10%	94566 91.40% 8.60%	7006 0.40%	69479 3.70%	1177 0.10%	77662 89.50% 10.50%
	Class2	1090 0.10%	16043 0.90%	242868 13.00%	260002 93.40% 6.60%	1190 0.10%	23445 1.30%	259008 13.80%	283643 91.30% 8.70%
	Total	1507428 99.50% 0.50%	121438 71.10% 28.90%	246134 98.70% 1.30%	1828591 / 1875000 97.50% 2.50%	1507311 99.50% 0.50%	105309 66.00% 34.00%	262380 98.70% 1.30%	1827602 / 1875001 97.50% 2.50%

	RF 100	Image 3 Normal				Image 3 Parallel			
		Class0	Classl	Class2	**Total**	Class0	Classl	Class2	**Total**
Real value	Class0	4273023 82.00%	103243 2.00%	12023 0.20%	4388289 97.40% 2.60%	4271799 81.90%	105043 2.00%	11023 0.20%	4387865 97.40% 2.60%
	Classl	4034 0.10%	59987 1.20%	10779 0.20%	74800 80.20% 19.80%	4030 0.10%	59491 1.10%	10778 0.20%	74299 80.10% 19.90%
	Class2	1073 0.00%	16710 0.30%	733127 14.10%	750911 97.60% 2.40%	1033 0.00%	16610 0.30%	734192 14.10%	751836 97.70% 2.30%
	Total	4278130 99.90% 0.10%	179940 33.30% 66.70%	755930 97.00% 3.00%	5066138 / 5214000 97.20% 2.80%	4276862 99.90% 0.10%	181144 32.80% 67.20%	755994 97.10% 2.90%	5065483 / 5214000 97.20% 2.80%

	RF 100	Image 4 Normal				Image 4 Parallel			
		Class0	Classl	Class2	**Total**	Class0	Classl	Class2	**Total**
Real value	Class0	569994 66.00%	13134 1.50%	1021 0.10%	584149 97.60% 2.40%	569994 66.00%	13134 1.50%	1021 0.10%	584149 97.60% 2.40%
	Classl	1206 0.10%	79984 9.30%	5332 0.60%	86522 92.40% 7.60%	1206 0.10%	79984 9.30%	5332 0.60%	86522 92.40% 7.60%
	Class2	1990 0.20%	11402 1.30%	179937 20.80%	193329 93.10% 6.90%	1990 0.20%	11402 1.30%	179937 20.80%	193329 93.10% 6.90%
	Total	573190 99.40% 0.60%	104520 76.50% 23.50%	186290 96.60% 3.40%	829915 / 864000 96.10% 3.90%	573190 99.40% 0.60%	104520 76.50% 23.50%	186290 96.60% 3.40%	829915 / 864000 96.10% 3.90%

7.4 Predicted Images

In **Figure 8** the predicted images on the top row represents the sequential processing and the bottom row represents the parallel processing of the random forest model with 100 trees. The Far left represents image number one. The second image from the left represents image number two. The third image from the left represents image number three. The farthest right represents image number four. The classification accuracy values shown in Table 1 for the RF 100 model produce the predicted image that is most similar to the actual real image after classification shown in Figure 4. The RF 100 model achieves the highest classification accuracy of (96.7%, 97.5%, 97.2%, and 96.1%) for the four images (1, 2, 3, 4) respectively.

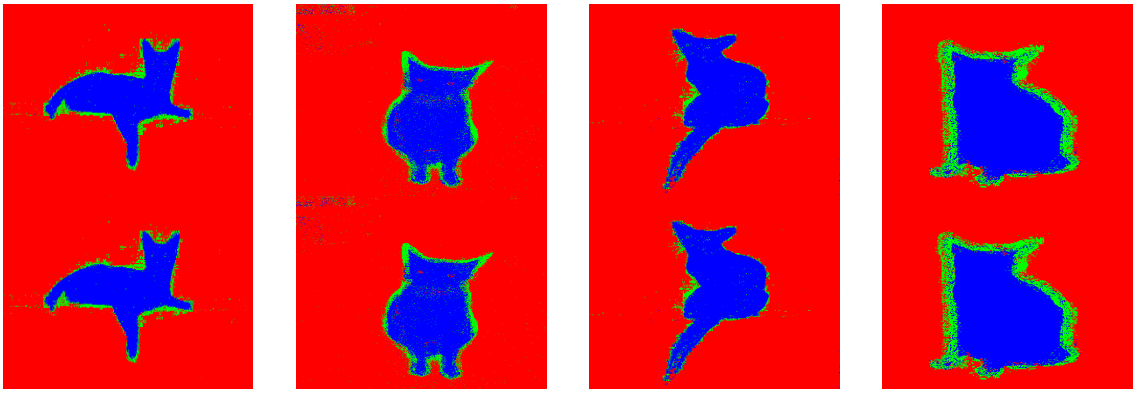

Figure 8. Predicted images for Normal and Parallel methods.

8. Cross Validation

In this research, 4-fold cross-validation was used as a mechanism to evaluate the performance of the classifiers.[47] [48]

In this method, the data is divided into 4 approximately equal folds, where each fold is used once as the test data, while the remaining three folds are used as the training data. In each iteration, the classification model is trained on the training data and then tested on the test fold, ensuring that the test data is completely unknown to the classifier to ensure an objective evaluation. The results of each test run are saved, then the next fold is moved and the process is repeated until all folds are tested. The results are then saved for four folds and summed to calculate the overall performance of the model. This method achieves an accurate and reliable evaluation of the performance of the models by taking advantage of the available data.

8.1 Performance Measurements of Cross-Validation

Table 3 compares the performance of the algorithms using the cross-validation split for the two methods, sequential and parallel processing. The table shows that the average accuracy remains constant for both methods, while the average time decreases significantly using parallel processing by 399.01 seconds compared to 801.05 seconds for sequential processing. The highest accuracy achieved in the four images (1, 2, 3, 4) is (96.7, 97.4, 97.2, 96.1) respectively. The lowest accuracy of the decision tree model was (96.3, 95.9, 96.9, 90.5) for images 1, 2, 3, and 4 respectively. While parallel processing significantly outperformed sequential processing in terms of execution time, execution time decreased from (10.17, 8.28, 45.18, 4.02) to (7.92, 4.48, 12.73, 2.56) for images (1, 2, 3, 4) respectively. The results confirm the effectiveness and impact of parallel processing on the efficiency of algorithms without compromising accuracy.

8.2 Confusion Matrix Cross-Validation

Table 4 shows the confusion matrices for cross-validation (CV) of the classification model performance for both sequential and parallel processing methods. This classification parameter value gives the best performance among all settings according to **Table 3**, Confusion Matrices for images 1, 2, 3, and 4. Where classes 0, 1, and 2 in the table represent background, border, and shape, respectively.

For the first confusion matrix, the background class was correctly predicted 770510 times and was misclassified as a border or shape 10739 times. The border class was correctly predicted 39413 times and was misclassified as a background or shape 14929 times. The shape class was correctly predicted 118225 times and was misclassified as both background and border 6184 times. The overall accuracy of the RF classifier is 100 for both serial and parallel processing, similarly for the rest of the matrices.

Table 3. Comparison Of Accuracy And Time Taken Between Models To Predict Test Images Cross-Validation

Dataset		Normal		Parallel (par for) function	
	classifier	*Accuracy (%)*	*Avg_time[s]*	*Accuracy (%)*	*Avg_time[s]*
Image 1	DT	96.3	10.17	96.3	**7.92**
	RF (10)	**96.7**	34.56	**96.7**	**26.15**
	RF (100)	96.7	177.60	96.7	**136.21**
	RF (500)	96.7	1258.9	96.7	**582.56**
	RF (1000)	96.7	1700.0	96.7	**1188.7**
Image 2	DT	95.9	8.28	95.9	**4.48**
	RF (10)	97.2	26.02	97.2	**9.78**
	RF (100)	**97.4**	129.87	**97.4**	**70.25**
	RF (500)	97.4	638.30	97.4	**510.09**
	RF (1000)	97.4	1633.9	97.4	**833.67**
Image 3	DT	96.9	45.18	96.9	**12.73**
	RF (10)	97.1	197.73	97.1	**71.61**
	RF (100)	**97.2**	749.03	**97.2**	**234.27**
	RF (500)	97.2	2581.61	97.2	**1160.6**
	RF (1000)	97.2	5399.41	97.2	**2494.1**
Image 4	DT	91.1	4.02	91.1	**2.56**
	RF (10)	95.2	12.81	95.1	**5.17**
	RF (100)	**96.1**	81.89	**96.1**	**35.62**
	RF (500)	96.1	501.83	96.1	**185.75**
	RF (1000)	96.1	829.94	96.1	**407.97**

Table 4. Percentages Of Normal And Parallel Avsa Confusion Matrix Cross-Validation

		Image 1 Normal				**Image 1 Parallel**			
		predicted value				predicted value			
RF 10		Class0	Classl	Class2	**Total**	Class0	Classl	Class2	**Total**
Real value	Class0	770510 80.30%	10181 1.10%	558 0.10%	781249 98.60% 1.40%	770593 80.30%	10181 1.10%	558 0.10%	781332 98.60% 1.40%
	Classl	12140 1.30%	39413 4.10%	2789 0.30%	54342 72.50% 27.50%	12140 1.30%	46444 4.80%	2703 0.30%	61287 75.80% 24.20%
	Class2	2386 0.20%	3798 0.40%	118225 12.30%	124409 95.00% 5.00%	2386 0.20%	3788 0.40%	111207 11.60%	117381 94.70% 5.30%
	Total	785036 98.10% 1.90%	53392 73.80% 26.20%	121572 97.20% 2.80%	928148 / 960000 96.70% 3.30%	785119 98.10% 1.90%	60413 76.90% 23.10%	114468 97.20% 2.80%	928244 / 960000 96.70% 3.30%

		Image 2 Normal				**Image 2 Parallel**			
RF 100		Class0	Classl	Class2	**Total**	Class0	Classl	Class2	**Total**
Real value	Class0	599997 80.00%	10022 1.30%	1034 0.10%	611053 98.20% 1.80%	595271 79.40%	10022 1.30%	1034 0.10%	606327 98.20% 1.80%
	Classl	2003 0.30%	30209 4.00%	3204 0.40%	35416 85.30% 14.70%	2003 0.30%	39027 5.20%	3104 0.40%	44134 88.40% 11.60%
	Class2	1136 0.20%	2400 0.30%	99995 13.30%	103531 96.60% 3.40%	1133 0.20%	2400 0.30%	96006 12.80%	99539 96.50% 3.50%
	Total	603136 99.50% 0.50%	42631 70.90% 29.10%	104233 95.90% 4.10%	730201 / 750000 97.40% 2.60%	598407 99.50% 0.50%	51449 75.90% 24.10%	100144 95.90% 4.10%	730304 / 750000 97.40% 2.60%

RF 100		Image 3 Normal					Image 3 Parallel			
		Class0	Classl	Class2	**Total**		Class0	Classl	Class2	**Total**
Real value	Class0	212189 61.40%	3387 1.00%	2470 0.70%	218046 97.30% 2.70%		222285 64.30%	3387 1.00%	2470 0.70%	228142 97.40% 2.60%
	Classl	3351 1.00%	24821 7.20%	1118 0.30%	29290 84.70% 15.30%		3351 1.00%	43714 12.60%	1118 0.30%	48183 90.70% 9.30%
	Class2	1312 0.40%	1744 0.50%	95208 27.50%	98264 96.90% 3.10%		1312 0.40%	1744 0.50%	66219 19.20%	69275 95.60% 4.40%
	Total	216852 97.80% 2.20%	29952 82.90% 17.10%	98796 96.40% 3.60%	332218 / 345600 96.10% 3.90%		226948 97.90% 2.10%	48845 89.50% 10.50%	69807 94.90% 5.10%	332218 / 345600 96.10% 3.90%

RF 100		Image 4 Normal					Image 4 Parallel			
		Class0	Classl	Class2	**Total**		Class0	Classl	Class2	**Total**
Real value	Class0	569994 66.00%	13134 1.50%	1021 0.10%	584149 97.60% 2.40%		569994 66.00%	13134 1.50%	1021 0.10%	584149 97.60% 2.40%
	Classl	1206 0.10%	79984 9.30%	5332 0.60%	86522 92.40% 7.60%		1206 0.10%	79984 9.30%	5332 0.60%	86522 92.40% 7.60%
	Class2	1990 0.20%	11402 1.30%	179937 20.80%	193329 93.10% 6.90%		1990 0.20%	11402 1.30%	179937 20.80%	193329 93.10% 6.90%
	Total	573190 99.40% 0.60%	104520 76.50% 23.50%	186290 96.60% 3.40%	829915 / 864000 96.10% 3.90%		573190 99.40% 0.60%	104520 76.50% 23.50%	186290 96.60% 3.40%	829915 / 864000 96.10% 3.90%

Predicted Images Cross-Validation

In **Figure 9** the predicted images on the top row represents the sequential processing and the bottom row represents the parallel processing of the random forest model with 100 trees. The Far left represents image number one. The second image from the left represents image number two. The third image from the left represents image number three.

The farthest right represents image number four. The classification accuracy values shown in Table 3 for the RF 100 model produced the predicted images that are most similar to the actual/ real image in Fig. 3. The RF 100 model achieves the highest classification accuracy of (96.7%, 97.4%, 97.2%, and 96.1%) for the four images (1, 2, 3, and 4) respectively.

Table 5 confirms the results obtained previously, the performance of the proposed method was compared with the methods mentioned in [14], [19], [20], [21], [23], and [22] before and after applying parallel processing, as shown in **Table 5**. The comparison shows that the proposed model (using parallel processing of the ParFor function achieves a significant time reduction of 24% while maintaining the same accuracy. In [22], [20], and [14], we observe a slight time reduction, while in [19], [21], and [23], the time decreased by 4%, 7%, and 1%, respectively.

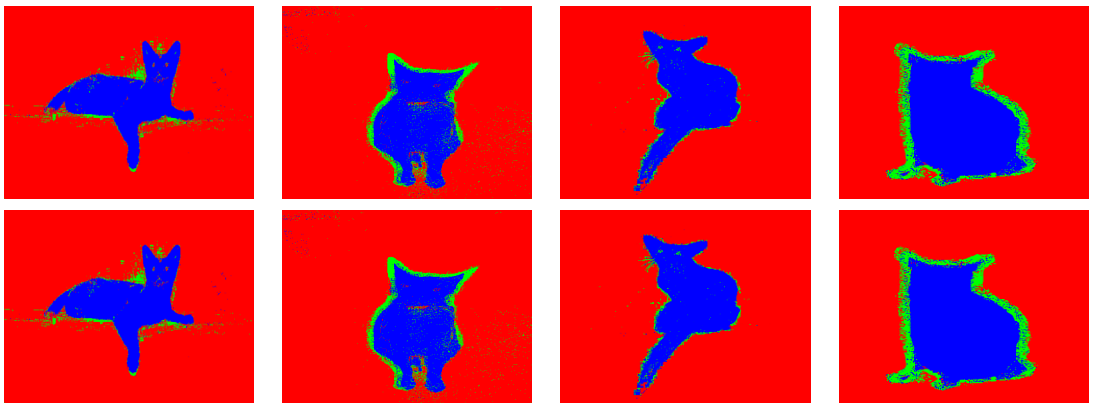

Figure 9. Predicted images of Cross-Validation

Table 5. Comparison With Previous Research

NO.	Model	Year	Dataset	Accuracy	Time before applying parallel	Time after applying parallel
1	[14] KNN, SVM, Tree	2023	NSL KDD	99.90%	from (1826, 1856.5, and 1792.8) for 2, 5, and 23-class classifications, respectively.	To (1667.6, 1611.9, and 1659.4) s for 2, 5, and 23-class classifications, respectively.
2	[19] SVM and RF	2024	imSitu	63.59%	from 1725.86ms	to 396.54ms
3	[20] RF	2024	CIFAR-10	97.50%	from 0.6187 seconds	To 0.4753 seconds
4	[21] Extra Trees Classifier	2024	Fashion MNIST	88.43%	from 37.463 seconds	to 4.837 seconds
5	[22] LightGBM	2024	IRIS plant	100%	From 0.2316 seconds	to 0.1921 seconds
6	[23] XGBoost	2024	Keras	79.83%	from 501.319 seconds	to about 264.978 seconds
7	Proposed system	2024	Oxford-IIIT Pet	97.2%	from 13458.7 seconds	to 550.30 seconds

Conclusion

This research analyzed the effect of parallel processing using the Parfor function on the performance of machine learning algorithms for pixel-based image classification, focusing on decision trees and random forest algorithms (10, 100, 500, and 1000 trees). The models were tested in two ways: 70% data split for training and 30% for testing, and using four-fold cross-validation. The results showed that parallel processing outperformed in reducing execution time while maintaining accuracy. Parallel processing proved effective in improving performance with large data and complex models without compromising accuracy, while cross-validation provided a comprehensive assessment of the models. Parallel processing using the Parfor function is an effective option for improving the efficiency of machine learning-based classification applications. In future work, experiments could be on larger and more diverse datasets to evaluate the effectiveness of parallel processing. Additionally, parallel processing could be applied to other machine learning algorithms, such as deep neural networks or unsupervised learning algorithms, to measure the effectiveness of this approach with more complex algorithms and high computational requirements.

Acknowledgement

The authors would express they're thanks to the College of Computer Sciences and Math. - University of Mosul.

Conflict of interest

None.

References

[1] A. Alseqyani and A. Almutairi, "History and Future Trends of Multicore Computer Architecture," International Journal of Computer Graphics & Animation, vol. 13, no. 2, pp. 01–08, Apr. 2023. https://doi: 10.5121/ijcga.2023.13201

[2] Mingjun Li., "A Parallel Algorithm and Implementation to Compute Spatial Autocorrelation (Hotspot) Using MATLAB Autocorrelation (Hotspot) Using MATLAB," 2020. https://epublications.marquette.edu/theses_open/584

[3] K. Jacksi, "A SURVEY OF EXPLORATORY SEARCH SYSTEMS BASED ON LOD RESOURCES," 2015. www.w3.org/TR/rdf-sparql-query

[4] D. A. Zebari, H. Haron, S. R. M. Zeebaree, and D. Qader Zeebaree, "Multi-Level of DNA Encryption Technique Based on DNA Arithmetic and Biological Operations," ICOASE 2018 - International Conference on Advanced Science and Engineering, pp. 312–317, Nov. 2018. https://doi: 10.1109/ICOASE.2018.8548824

[5] A. N. Younis and F. M. Ramo, "A new parallel bat algorithm for musical note recognition," International Journal of Electrical and Computer Engineering, vol. 11, no. 1, pp. 558–566, Feb. 2021. https://doi: 10.11591/IJECE.V11I1.PP558-566.

[6] 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," 2019 International Conference on Advanced Science and Engineering, ICOASE 2019, pp. 100–105, Apr. 2019. https://doi: 10.1109/ICOASE.2019.8723779

[7] 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, Mar. 2021. https://doi: 10.38094/jastt20189

[8] H. Shukur et al., "Characteristics and Analysis of Hadoop Distributed Systems." https://www.researchgate.net/publication/341775003

[9] P. Y. Abdullah, S. R. M. Zeebaree, K. Jacksi, and R. R. Zeabri, "AN HRM SYSTEM FOR SMALL AND MEDIUM ENTERPRISES (SME)S BASED ON CLOUD COMPUTING TECHNOLOGY," International Journal of Research -GRANTHAALAYAH, vol. 8, no. 8, pp. 56–64, Aug. 2020. https://doi: 10.29121/granthaalayah.v8.i8.2020.926

[10] Dathar A. Hasan, Bzar Kh. Hassan, Subhi R. M. Zeebaree, Dindar M. Ahmed, Omar S. Kareem &, and Mohammed A. M. Sadeeq, "The impact of test case generation methods on the software," International Journal of Science and Business, 2021.

[11] D. Abas Hasan, R. R. Zebari, S. R. M Zeebaree, and K. Jacksi, "Security Approaches For Integrated Enterprise Systems Performance:

A Review," Article in International Journal of Scientific & Technology Research, vol. 8, 2019. www.ijstr.org

[12] P. Abdullah, H. Shukur, P. Y. Abdullah, S. R. M. Zeebaree, H. M. Shukur, and K. Jacksi, "HRM System using Cloud Computing for Small and Medium Enterprises (SMEs)," 2020. https://www.researchgate.net/publication/341883552

[13] M. A. Omer et al., "Efficiency of Malware Detection in Android System: A Survey," Asian Journal of Research in Computer Science, pp. 59–69, Apr. 2021. https://doi: 10.9734/ajrcos/2021/v7i430189

[14] M. Kollam and A. Joshi, "A MACHINE LEARNING MODEL FOR AN EARTHQUAKE FORECASTING USING PARALLEL PROCESSING," Faculty of Engineering, The University of The West Indies, 2020, pp. 512–518. https://doi: 10.47412/dhhv5862

[15] A. Boukhalfa, N. Hmina, and H. Chaoui, "Parallel processing using big data and machine learning techniques for intrusion detection," IAES International Journal of Artificial Intelligence, vol. 9, no. 3, 2020. https://doi: 10.11591/ijai.v9.i3.pp553-560

[16] Z. Haider, J. Ge, K. Willis, Y. Zhao, and K. Wang, "A Novel Method of Transit Detection Using Parallel Processing and Machine Learning." www.JSR.org

[17] S. Al Bayyat, A. Alomran, M. Alshatti, A. Almousa, R. Almousa, and Y. Alguwaifli, "Parallel Inference for Real-Time Machine Learning Applications," Journal of Computer and Communications, vol. 12, no. 01, pp. 139–146, 2024. https://doi: 10.4236/jcc.2024.121010

[18] A. Ghimire and F. Amsaad, "A Parallel Approach to Enhance the Performance of Supervised Machine Learning Realized in a Multicore Environment," Mach Learn Knowl Extr, vol. 6, no. 3, pp. 1840–1856, Aug. 2024, https://doi: 10.3390/make6030090

[19] V. Ashqi Saeed and S. R. M. Zeebaree, "Enhancing AdaBoost Performance: Comparative Analysis of CPU Parallel Processing on Breast Cancer Classification," The Indonesian Journal of Computer Science, vol. 13, no. 2, Apr. 2024. https://doi: 10.33022/ijcs.v13i2.3793

[20] Suprapto, Wahyono, N. Rokhman, and F. D. Adhinata, "A Parallel Approach of Cascade Modelling Using MPI4Py on Imbalanced Dataset," International Journal of Computing and Digital Systems, vol. 15, no. 1, pp. 1289–1302, Mar. 2024. https://doi: 10.12785/ ijcds /150191

[21] B. Haval and S. Zeebaree, "Parallel Processing Impact on Random Forest Classifier Performance: A CIFAR-10 Dataset Study," Apr. 2024. https://www.researchgate.net/publication/380576940

[22] N. M. Salih and S. R. Zeebaree, "Performance Evaluation of Extra Trees Classifier by using CPU Parallel and Non-Parallel Processing," Indonesian Journal of Computer Science Attribution, vol. 13, no. 2, pp. 2024–1859, Apr. 2024.

[23] S. Muhammed Sulaiman and S. R. M Zeebaree, "Impact of Parallel Processing on LightGBM Implementation a Comparative Analysis CPU on Iris Plants Dataset," Indonesian Journal of Computer Science Attribution, vol. 13, no. 2, p. 2270, 2024.

[24] O. M. Ahmed, S. R. Zeebaree, and S. Askar, "Comparative Analysis of XGBoost Performance for Text Classification with CPU Parallel and Non-Parallel Processing," Indonesian Journal of Computer Science Attribution, vol. 13, no. 2, pp. 2024–1781, 2024.

[25] S. Aleissa, M. Alakkas, Z. Albugeaey, H. Alshelaly, S. Alotaibi, and T. Alzubaidi, "Leveraging Parallel Computing for Enhanced Stock Movement Forecasting Using Machine Learning," in Proceedings - 2024 7th International Women in Data Science Conference at Prince Sultan University, WiDS-PSU 2024, Institute of Electrical and Electronics Engineers Inc., 2024. pp. 67–72. https://doi: 10.1109/WiDS-PSU61003.2024.00028

[26] S. A. Ludwig, J. Al-Sawwa, and A. M. Misquith, "Parallelization of the Bison Algorithm Applied to Data Classification," Algorithms, vol. 17, no. 11, Nov. 2024. https://doi: 10.3390/a17110501

[27] "Parfor."Accessed: Dec. 19, 2024. https://www.mathworks.com/help/parallel-computing/parfor.html

[28] "Nested Parfor and for-Loops and Other Parfor Requirements." Accessed: Dec. 19, 2024. https://www.mathworks.com/help/parallel-computing/nested-parfor-loops-and-for-loops.html

[29] "Decide When to Use Parfor." Accessed: Dec. 19, 2024. https://www.mathworks.com/help/parallel-computing/decide-when-to-use-parfor.html

[30] E. B. Hamdi, J. A. Sunaryo, and S. Y. Prasetyo, "Fusion of pre-trained CNN models for cat breed classification: A comparative study," in E3S Web of Conferences, EDP Sciences, Sep. 2023. https://doi: 10.1051/e3sconf/202342601014

[31] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar, "Cats and dogs," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 3498–3505, 2012. https://doi: 10.1109/CVPR.2012.6248092

[32] R. Hamdy and M. Younis, "Performance Evaluation of Artificial Neural Network Methods Based on Block Machine Learning Classification," *AL-Rafidain Journal of Computer Sciences and Mathematics*, vol. 17, no. 2, pp. 111–123, Dec. 2023. https://doi: 10.33899/csmj.2023.142250.1079

[33] P. Y. Rohan et al., "Real-time numerical prediction of strain localization using dictionary-based ROM-nets for sitting-acquired deep tissue injury prevention," Reduced Order Models for the Biomechanics of Living Organs, pp. 385–402, Jan. 2023. https://doi: 10.1016/B978-0-32-389967-3.00027-5

[34] F. M. Ramo and M. N. Kannah, "Detect Multi Spoken Languages Using Bidirectional Long Short-Term Memory Article information Abstract," *Journal of Computer Sciences and Mathematics (RJCM)*, vol. 17, no. 1, pp. 2023–2024, 2023. www.csmj.mosuljournals.com

[35] A. A. Lawan, N. Cavus, R. Yunusa, U. I. Abdulrazak, and S. Tahir, "Fundamentals of machine-learning modeling for behavioral screening and diagnosis of autism spectrum disorder," Neural Engineering Techniques for Autism Spectrum Disorder: Volume 2: Diagnosis and Clinical Analysis, vol. 2, pp. 253–268, Jan. 2022. https://doi: 10.1016/B978-0-12-824421-0.00020-5

[36] O. J. Awujoola, F. N. Ogwueleka, P. O. Odion, A. E. Awujoola, and O. R. Adelegan, "Genomic data science systems of Prediction and prevention of pneumonia from chest X-ray images using a two-channel dual-stream convolutional neural network," Data Science for Genomics, pp. 217–228, Jan. 2022. https://doi: 10.1016/B978-0-323-98352-5.00013-6

[37] M. S. Sandeep, K. Tiprak, S. Kaewunruen, P. Pheinsusom, and W. Pansuk, "Shear strength prediction of reinforced concrete beams using machine learning," *Structures*, vol. 47, pp. 1196–1211, Jan. 2023. https://doi: 10.1016/j.istruc.2022.11.140

[38] A. Kulkarni, D. Chong, and F. A. Batarseh, "Foundations of data imbalance and solutions for a data democracy," *Data Democracy: At the Nexus of Artificial Intelligence, Software Development, and Knowledge Engineering*, pp. 83–106, Jan. 2020. https://doi: 10.1016/B978-0-12-818366-3.00005-8

[39] P. Singh, N. Singh, K. K. Singh, and A. Singh, "Diagnosing of disease using machine learning," *Machine Learning and the Internet of Medical Things in Healthcare*, pp. 89–111, Jan. 2021. https://doi: 10.1016/B978-0-12-821229-5.00003-3

[40] I. D. Mienye and N. Jere, "A Survey of Decision Trees: Concepts, Algorithms, and Applications," *IEEE Access*, vol. 12, pp. 86716–86727, 2024. https://doi: 10.1109/ACCESS.2024.3416838

[41] A. Abdulwahhab Yehya, F. M. Ramo, and A. A. Yehya, "The Intelligent Recognition of Speech Emotions: Survey Study," *Journal of Computer Sciences and Mathematics (RJCM)*, vol. 17, no. 2, pp. 2023–2024, 2023. www.csmj.mosuljournals.com

[42] C. Y. Mohammed, K. Edward, S. Dragan, and R. Anthony, "Evaluating Classification Algorithms for Improved Wastewater System Calibration," *Computing and Control for the Water Industry*, Sep. 2017, Accessed: Jan. 23, 2025. https://web.archive.org/web/20200217175305/https://s3-eu-west-1.amazonaws.com/pstorage-sheffield-5641355/9218269/F78.pdf

[43] R. Fiagbe, "Classification of Adult Income Using Decision Tree," University of Central Florida. 2023. https://www.researchgate.net/publication/370770672_Classification_of_Adult_Income_Using_Decision_Tree

[44] J. Ali, R. Khan, N. Ahmad, and I. Maqsood, "Random Forests and Decision Trees," 2012. www.IJCSI.org

[45] H. Deng, Y. Diao, W. Wu, J. Zhang, M. Ma, and X. Zhong, "A high-speed D-CART online fault diagnosis algorithm for rotor systems," *Applied Intelligence*, vol. 50, no. 1, pp. 29–41, Jan. 2020. https://doi: 10.1007/S10489-019-01516-2.

[46] L. Fawaz Jarallah, "Medical decision support systems for diagnosing diseases based on ensemble learning algorithms," *Journal of Computer Sciences and Mathematics (RJCM)*, vol. 18, no. 2, pp. 115–120, 2024. www.csmj.mosuljournals.com

[47] J. Allgaier and R. Pryss, "Cross-Validation Visualized: A Narrative Guide to Advanced Methods," *Machine Learning and Knowledge Extraction 2024, Vol. 6, Pages 1378-1388*, vol. 6, no. 2, pp. 1378–1388, Jun. 2024, doi: 10.3390/MAKE6020065.

[48] M. Y. Chachan, E. Keedwell, and D. Savic, "An Investigation of Pixel-Based and Object-Based Image Classification in Remote Sensing," Kurdistan Region, Iraq: IEEE, Oct. 2018.