Enhancing Machine Learning Through Advanced Optimized Parallelized Model Aggregation: a Novel Theory of Optimized Parallelized Ensemble Learning (OPEL)

Jephter Pelekamoyo, Libati Hastings M. and Derrick Ntalasha

November 4, 2024

Enhancing Machine Learning through Advanced Optimized Parallelized Model Aggregation:

A novel theory of Optimized Parallelized Ensemble Learning (OPEL)

Jephter Kapika Pelekamoyo
Department of Information and Communication Technology, The Copperbelt University Kitwe,
jephkapi@outlook.com
Department of Information and Communication Technology, Kapasa Makasa University, Zambia,
Jephter.pelekamoyo@kmu.ac.zm

Hastings M Libati
Department of Information and Communication Technology, The Copperbelt University Kitwe, Zambia,
libati@cbu.ac.zm

Derrick Ntalasha
Department of Information and Communication Technology, The Copperbelt University Kitwe, Zambia,
derick.ntalasha@cbu.ac.zm

***Abstract*—**This study presents a parallelized multi-mode ensemble learning framework to optimize computational efficiency, speed and model accuracy, which is a novel framework for optimizing machine learning ensemble multi-model selection using parallelized, execution and voting mechanisms, using a proposed theory, 'Optimized Parallelized Ensemble Learning' (OPEL), for optimized voting. By formulating theoretical mathematical models to guide model selection, weighting, and parallel execution strategies and utilizing performance metrics like the Matthews correlation coefficient to select top-performing models, with parallel processing incorporated to enhance efficiency, experimental simulations were conducted on real-world datasets using high-performance computing platform. Coupled with comparative analysis with traditional methods, reveals improved computation speed and accuracy under varying conditions. This paper henceforth introduced key innovations, which include the Parallelized Model Execution (PME) approach, Consensus-Based Model Selection (CMS), and Optimized Parallel Voting Mechanism (OPVM), each contributing to reduced computational time and improved model performance. The study demonstrates significant gains in computational speed and accuracy through parallelization and advanced voting techniques, with a time complexity reduction as defined by Amdahl's Law. The proposed ensemble learning framework is validated as both computationally efficient and robust in diverse, large-scale AI applications.

Keywords*— *Ensemble; Parallelization, Optimization; Efficiency; Voting; Accuracy

I. Introduction

Despite the traditional ensemble methods being powerful, they often suffer with scalability and efficiency, especially in large-scale, heterogeneous settings and when coupled with fast-changing data, which is usually small data. The proposed framework in this study addresses these challenges by introducing a multi-model selection mechanism that identifies the best-performing models within a given set of classical machine learning models and ensemble learning models alike, coupled with parallel processing techniques to expedite the computation processes. An optimized weighting algorithm is employed to ensure that models contributing most effectively to the task at hand are prioritized, thereby improving the robustness of the final decision. Experimental evaluations demonstrate that the proposed method outperforms conventional ensemble approaches in terms of both speed and accuracy, particularly in distributed computing environments. This work contributes to the growing body of research on scalable machine learning and offers a practical solution for real-world applications where computational resources are limited for big data. This theory advances the field of machine learning and artificial intelligence by providing a scalable, efficient, and robust alternative to traditional ensemble methods, making it particularly well-suited for modern distributed computing challenges.

Currently, it's difficult to efficiently combine multiple machine learning models to improve decision-making accuracy and performance in scenarios where data is distributed or computational resources are limited, in a timely efficient manner. Traditional ensemble methods like Random Forests or Ada Boosting rely on training and aggregating multiple models, which can be computationally expensive and difficult to scale in distributed systems or with large datasets. Furthermore, existing methods often lack robustness in selecting the best-performing models in heterogeneous environments, where different models may excel in different aspects of the task.

This paper proposes a theory, to develop a novel ensemble framework that leverages model selection and parallel processing techniques to optimize decision-making in distributed and resource-constrained environments. This framework seeks to minimize computational overhead while maximizing the accuracy and robustness of the ensemble model, by formulating an efficient method for selecting the best-performing models from a pool of candidate models based on their performance metrics. Followed by integrating parallel computing strategies to distribute the computational load across multiple processors, thereby reducing the time required to train and evaluate the ensembles. All that will be done, following an algorithm that assigns appropriate weights to models in the ensemble based on their performance, ensuring that the most reliable models have a greater influence on the final decision. The study will also conduct a comprehensive evaluation by comparing the proposed framework with traditional methods to demonstrate its advantages in terms of scalability, efficiency, and accuracy. Finally, testing the proposed ensemble framework by measuring its performance in terms of accuracy, speed, and market user satisfaction.

II. Hypotheses

- Null hypothesis 1: The time (T) complexity T(n, P) for model execution increases with an increase in the number of processing units P when the problem size is n as when using an optimized parallelled voting mechanism (OPVM), compared to serial voting mechanisms [2].
- Null hypothesis 2: Models selected using an optimized parallel voting mechanism (OPVM) will not have a statistically significant higher Matthews correlation coefficient (MCC) compared to traditional static models preselected voting mechanisms [3] given varying sample sizes.

III. RELATED WORKS

Amdahl's work is critical for this current work, particularly Amdahl's Law (1967), which provides a fundamental understanding of the limitations of parallel processing by quantifying the maximum speedup achievable when only part of a task is parallelized while considering that some parts of the task must remain serial. This law underscores the inherent limitations in achieving significant performance gains through parallelization, particularly when a substantial portion of a task cannot be parallelized has to parallel model aggregation in machine learning apply these scalability principles to enhance model performance, particularly in handling large datasets and complex models [4][5][6]. Amdahl's law is critical in understanding the limits of parallelization although it does not extend to machine learning, it still provides critical insights into the limits of parallel computation. This study extends those principles by applying Amdahl's law to the specific context of machine learning model selection and weighted voting mechanisms.

Agarwal and Chowdary (2021), proposed an ensemble learning-based adaptive model for automatic hate speech detection that aims to improve cross-dataset generalization and their expert model addressed the strong user bias present in their annotated datasets. The experiments they conducted demonstrated the effectiveness of the usage of their proposed model on recent topics such as COVID-19 and the US presidential elections. Their model used ensemble-based adaptive classifier, A-Stacking, utilizes multiple base classifiers in combination with a meta-classifier, employing Support Vector Machine Classifier (SVM), Gradient Boosting Decision Trees (GBDT), Multi-Layer Perceptron Classifier (MLP), kNeighbors Classifier, ELM classifier15, along with Logistic Regression for the meta-classifier and to perform clustering, they utilized the SimpleKMeans clustering algorithm with varying values [7].

Agarwal et al. (2023), accelerate the automatic detection of hate speech on social media platforms (SMPs), by implementing parallelizing bagging, A-stacking, and random sub-space algorithms. They evaluated the serial and parallel versions of the machine learning models on standard high-dimensional hate speech datasets and the parallel models demonstrated a substantial increase in speed with remarkable efficiency, affirming that the proposed models are well-suited for this particular application. They observed that parallelizing the algorithms does not compromise the accuracy compared to running machine learning ensemble algorithms sequentially on a single machine [8]

Aldjanabi et al. (2021), covered the development of a classification system that identified offensive and hate speech using a multi-task learning (MTL) model built on a pre-trained Arabic language model. Through training the MTL model on the same task using different cross-corpora representing variations in offensive and hate context. The results indicated that the developed MTL model exhibited significant performance improvements compared to existing models in the literature, outperforming them on three out of four evaluated datasets for Arabic offensive and hate speech detection tasks. The findings demonstrate the superior classification performance of the developed MTL model in comparison to previously proposed models [9].

Kapil and Ekbal (2020), introduced a deep multi-task learning (MTL) framework, which aimed at enhancing the performance of individual classification tasks by leveraging valuable information from multiple related tasks. The proposed MTL model adopted a shared-private scheme, where shared and private layers were assigned to capture shared features and task-specific features from five classification tasks. Through experiments conducted on five datasets, the Shared-Private Multi-Task Learning (SP-MTL) framework leveraged the benefits of multiple related tasks and demonstrated promising results in terms of macro-F1 and weighted-F1 performance metrics [10].

Dietterich and Thomas (2000) provide an overview of ensemble learning and bagging predictors methods in the paper titled 'Ensemble Methods in Machine Learning'. They emphasized how combining multiple models can improve overall prediction accuracy. The paper discusses various ensemble techniques, including bagging, boosting, and stacking [3]. Similar principles were proposed by Breiman (1996), where the author introduced the concept of bagging (Bootstrap Aggregating), where multiple versions of a predictor are trained on different subsets of the data, and their predictions are averaged to improve robustness [11]. Dietterich (2000), describes the Bagging (Bootstrap Aggregating) method, where multiple versions of a predictor are trained on different samples of the training set and combined by averaging their predictions [3].

While Hansen and Salamon (1990), proposed creating ensembles of neural networks to improve generalization by averaging predictions from multiple independently trained networks [12]. Neural network ensembles are well known for significantly improving model accuracy and reducing overfitting, particularly in complex tasks like image recognition. However, as the proposed method involves training multiple neural networks independently, this increases computational costs and may require substantial computational resources, particularly for deep networks.

AdaBoost Algorithm is among the other models used among the multiple models, which Freund and Schapire (1997), studied. In their work, they introduced the AdaBoost algorithm, which improves weak learners by focusing on the instances that previous models struggled to classify. The emphasis was on iteratively adjusting weights to improve overall accuracy [13] AdaBoost is an ensemble technique that combines weak classifiers to create a strong classifier by iteratively adjusting the weights of incorrectly classified examples, thereby reducing bias and variance, and significantly improving the performance of weak classifiers [13].

Breiman (2001), introduced Random Forests, an ensemble learning method that builds multiple decision trees and merges them to get a more accurate and stable prediction (Breiman 2001). His work is similar to the one proposed in this paper, as it merges multiple decision trees for more stable predictions. Unlike Breiman's Random Forest algorithm, which involves creating an ensemble of decision trees, each trained on a random subset of the data, with the final prediction based on the majority vote of the trees [14] it does not incorporate parallel computation efficiency nor incorporate a weighted voting system that is optimized for parallel computation.

Teh et al., (2006), introduce hierarchical models that allow for sharing statistical strength across different groups of data. The authors leverage Bayesian nonparametrics to build a flexible model that can be parallelized across clusters [15]. The method allowed for a more nuanced model that could capture complex dependencies within the data, and parallelization improves scalability.

Cortes and Vapnik (1995), developed Support Vector Machines (SVMs) as a method for finding the optimal hyperplane that separates data into different classes, maximizing the margin between classes [16]. Zanghirati and Zanni (2003), explore the parallelization of SVM training using quadratic programming, significantly reducing the computational time for large datasets[17] [18]. The study used a parallel decomposition technique to solve the quadratic programming problem in SVM training, distributing the workload across multiple processors[17]. Their technique significantly reduced training time for large datasets by parallelizing the optimization process. Their working principle is similar to the one proposed in this paper. But instead of parallelizing SVMs alone, the current method integrates a voting mechanism and equally focuses on a more generalized framework applicable across different models.

Dean et al. (2012) present a method for distributed training of deep neural networks through model parallelism, where different segments of a neural network are distributed across multiple machines. This approach enables the handling of extremely large datasets and models, facilitating the training of deep networks with billions of parameters. Their study demonstrated the scalability of deep learning systems and laid the groundwork for practica[19].

Chu et al. (2006), introduced the MapReduce framework, using parameter server architecture to efficiently scale distributed machine learning models across multiple servers, optimizing both storage and computation, allowing for large-scale machine learning tasks to be handled more effectively in a distributed environment. Their framework utilized data distribution and parallel computation, making it a foundational method for processing vast datasets in a distributed manner [20]. Similarly, Li et al (2014), used parameter server architecture to efficiently scale distributed machine learning models across multiple servers, optimizing both storage and computation. This facilitated the parallel training of machine learning models [1]. This approach significantly improves the scalability of machine learning training by efficiently handling parameter updates across distributed systems but introduces latency and synchronization issues, particularly in highly distributed systems with non-uniform communication speeds.

Cole and Vishkin (1986), proposed a 'Theoretical Parallel Model', the development of deterministic algorithms for parallel computation, including techniques for reducing contention and improving efficiency [21]. Cole and Vishkin (1986) developed deterministic algorithms for parallel computation, emphasizing techniques to reduce contention among processors and enhance overall computational efficiency. Their work is instrumental in the creation of parallel algorithms that operate under strict deterministic conditions, ensuring consistent and predictable performance across different computational tasks [21]. While Cole and Vishkin (1986), provided essential insights into the development of deterministic parallel algorithms, it does not extend these principles to machine learning or model aggregation.

Graham (1966), worked on load-balancing issues in parallel computation, addressing the inefficiencies that arise when tasks are not evenly distributed across processors. The primary focus is on ensuring that each processor in a parallel computing environment is utilized effectively to avoid bottlenecks that can occur when tasks are not evenly distributed [22]. His work was further amplified by Brent (1974), who offered a fundamental analysis of the efficiency of parallel algorithms, concentrating on minimizing communication overhead and ensuring effective load balancing across processors, and established key principles for optimizing parallel computation, particularly by reducing the time complexity of parallel algorithms and ensuring that tasks are distributed in a manner that maximizes processor utilization [23]. Karp and Ramachandran (1990), further comprehensively examined parallel algorithms, particularly within the context of shared-memory architecture[24].

Shalev-Shwartz et al. (2011), introduced the Pegasos algorithm, a stochastic sub-gradient descent method for efficiently training support vector machines (SVMs). The algorithm was particularly notable for its scalability, making it well-suited for handling large datasets. The Pegasos algorithm significantly reduces the computational complexity of SVM training, providing a more practical solution for real-world, large-scale machine-learning tasks [5]

Zhang et al. (2013), proposed a divide-and-conquer approach for scaling kernel ridge regression on large datasets by splitting data into smaller subsets and processing subsets independently in parallel before combining the results solving the problem on each subset, and then combining the results [25]. According to Zhang et al. (2013), for finite-rank kernels and Gaussian kernels, their theory ensured that the number of processors, denoted as m, can increase almost linearly, for Sobolev spaces, the number of processors can grow polynomially with N. The partitioning led to a substantial reduction in computation time and cost [25].

Elkan (1997), study titled "Boosting and Naive Bayesian Learning" challenges the assumption that boosting, a technique primarily known for improving decision tree models, can indeed enhance the performance of Naive Bayes by focusing on difficult-to-classify instances, leading to improved overall accuracy. Elkan (1997), argues that boosting applied to naive Bayesian classifiers yields combination classifiers that are representationally equivalent to standard feedforward multilayer perceptrons. However, this study did not explore boosting in a distributed or parallel computing context, focusing instead on the theoretical and practical implications within a single-machine environment [26].

Kumar and Gupta's (1994) study provides a comprehensive analysis of the scalability of parallel algorithms across various computing architectures, focusing on shared memory, distributed memory, and hybrid systems. Their work emphasizes the importance of load balancing and minimizing communication overhead to optimize scalability, offering a strong theoretical foundation for parallel computation. However, the study lacks a focus on machine learning-specific applications, such as model selection and ensemble voting, and some of the discussed architectures are now outdated. In contrast, modern approaches to parallel model aggregation in machine learning apply these scalability principles to enhance model performance, particularly in handling large datasets and complex models [4]. While Kumar and Gupta's work is foundational, contemporary methods extend these concepts to address the unique challenges of machine learning in distributed environments.

The proposed theory diverges from existing works in its approach to various model integration, optimization, and parallelization. Traditional ensemble methods use fixed voting schemes, while the proposed framework introduces a dynamic weighted voting mechanism based on real-time model performance metrics. This allows for adaptation to changing data distributions and resource availability, improving robustness and performance. The framework also leverages parallel and distributed computing to optimize the integration and combination of multiple models, minimizing communication overhead and ensuring load balancing. Most related works focus on either ensemble learning or parallel computing separately, while the proposed framework uniquely integrates a weighted voting mechanism into a parallel computing context. It offers a generalized framework applicable to many machine learning models, utilizing both parallel processing and ensemble techniques. The papers also draw on established theories like Amdahl's Law and Brent's theorem to provide new insights into the trade-offs between processor count, overhead, and model accuracy in parallel environments.

IV. METHODOLOGY

The study employs a combination of theoretical modelling, experimental simulations, and comparative analysis to develop and validate the proposed parallelized multi-mode ensemble learning framework.

A. *Theoretical Modeling*

The initial phase of the study involves developing the theoretical underpinnings of the parallelized ensemble framework. This includes formulating mathematical models to describe the selection and weighting of models within the ensemble, as well as the parallel processing strategies.

B. *Algorithm Development*

Based on the theoretical models, algorithms are developed for model selection, weighting, and parallel processing. The model selection algorithm identifies the top-performing models from a pool of candidates using metrics such as accuracy, precision, recall, the Matthews correlation coefficient also known as the phi coefficient and the confusion matrix. The weighting algorithm then assigns a rank to each selected model based on its relative performance. Parallel processing techniques are incorporated to optimize the computational efficiency of the framework at every relevant stage and process.

C. *Experimental Simulations*

The The algorithms developed were implemented in a simulation environment to evaluate their performance. The simulations were run on real-world datasets. Metrics such as computation time and accuracy wre recorded. The experiments wre conducted using a high-performance computing platform with a memory of 32 gigabytes and an Intel Core i9-10980HK processor, leveraging parallel functions from the 'System.Threading.Tasks' library. Datasets used for simulations included real-world datasets collected from the market used in [27], [28], and also [28].

D. *Comparative Analysis*

The results from the experimental simulations are compared with the performance of traditional methods and traditional ensembles using iterations of from 100 and tolerances of 1e-4, and with varying training sample sizes. The models used include the probabilistic coordinate descent, sequential minimal optimization with polynomial kernel, iterative reweighted least squares with logistic regression method FanChenLin support vector regression with Gaussian kernel, linear regression newton method, AdaBoost with decision stump and threshold learning method, AdaBoost with logistic regression methods and iterative reweighted least squares with logistic regression method, and AdaBoost with Decision Tree with C45 learning. Key performance indicators (KPIs) like accuracy and processing time are compared across different methods.

E. Statistical Validation

In this study, a dataset which was collected through a survey to evaluate the performance of a developed tool using the proposed framework was used, to assess the effectiveness of the application. The dataset included historical weather data, encompassing low and high temperatures, alongside local market inventory levels, supply records, and sales records [27], [28] for supply chain resilience binary decision making, together with the 'hotel booking cancellation prediction dataset [18], consisted of a total of 35000 samples. With the statistical t-tests, the researchers could determine the significance of key performance indicators associated with the proposed framework. The tests provided statistical evidence to support or refute the impact of the framework on supply chain resilience. The weather and supply chain dataset consisted of 293 stakeholders.

V. METHODOLOGY

A. Parallelized Model Execution (PME)

This PME is a computational approach where multiple machine learning models are trained and evaluated concurrently on separate processing units on the same dataset or input to obtain results in parallel rather than sequentially. This parallelisation reduces the overall computational time while maintaining or improving model performance. Parallel execution concepts are rooted in the broader field of parallel computing [2], [29].

Given M models $\{M_1, M_2, \dots, M_n\}$ and P processing units $\{P_1, P_2, \dots, P_n\}$, PME distributes the computation of each model across the processors. The time complexity $T(n, P)$ for training and evaluation is reduced from $T(n)$ (sequential execution) to:

$$T(n,P) = \frac{T(n)}{P} + O(\frac{n}{P}.\log P)$$

where $O(\frac{n}{P}.\log P)$ represents the overhead of parallelisation, including communication and synchronization costs [29], [30]. The results are given as:

$$\{R_1, R_2, \dots, R_n\} \\ = Parallel.Invoke(f_1(t), f_2(t), \dots, f_n(t))$$

Where R_i represents the result of the models f_i applied to input t.

The equation herein

$$T(n,P) = \frac{T(n)}{P} + O(\frac{n}{P}.\log P)$$

is used to describe the parallel running time of an algorithm when executed on P processors [31].

Components of the Equation:

i. $T(n, P)$, represents the total time required to run an algorithm on P processors when the problem size is n.

ii. $\frac{T(n)}{P}$: $T(n)$ is the time it takes to run the algorithm sequentially (on a single processor) for a problem size n. Dividing $T(n)$ by P suggests that the algorithm can be broken down into P parallel tasks, each of which takes the same ratio of the divided amount of time as compared to the sequential algorithm. However, this assumes ideal conditions, such as perfect parallelism without any overhead.

iii. $O(\frac{n}{P}.\log P)$: represents the overhead associated with parallelism. It accounts for factors like communication between processors, synchronization, and load balancing.

iv. $\frac{n}{P}$: indicates that the problem is being divided across P processors, and each processor handles a portion $\frac{n}{P}$ of the workload [31].

$\log P$: comes from the communication cost, as in many parallel algorithms, communication overhead increases logarithmically with the number of processors.

B. Consensus-Based Model Selection (CMS)

After executing models in parallel, the Theory of Parallelized Model Voting and Selection proposes selecting the top-performing models based on a voting mechanism where the results are evaluated for consistency and accuracy. CMS is an ensemble learning technique that selects the best-top-performing models given by the formula below, based on a voting mechanism.

$$M_{best} = Mode(R_1, R_2, \dots, R_n)$$

The selection process considers not only the individual performance metrics but also the agreement among models. Where, M_{best} represents the most frequently best-performing models, as determined by a voting mechanism across all parallel executions.[3]

Let M_i be the $i-th$ model with a performance metric θ_i. The final decision D is made by considering the consensus among the models:

$$D = \text{argmax}_{\text{i}\in\{1,\dots,\text{n}\}} \sum_{j=1}^{m} \left(w_j.\,\delta(M_i, M_j)\right)$$

where w_j is the weight of the $j-th$ model, and $\delta(M_i, M_j)$is a similarity function between models M_i and M_j [3], [32].

Argmax [33]: This function returns the index i of the model M_i that maximizes the expression that follows it. In other words, it finds the model M_i for which the sum of $\sum_{j=1}^{m}\left(w_j.\,\delta(M_i, M_j)\right)$ is the largest and $\boldsymbol{i} \in \{\boldsymbol{1}, \dots, \boldsymbol{n}\}$. The model selection is done from a set of n models, where i ranges from 1 to n.

i. $\sum_{j=1}^{m}$: The summation is over m models that are considered for consensus. The summation aggregates the weighted similarity between the model M_i and each other model M_j.

ii. W_j: represents the weight assigned to the j-th model. This weight could be based on the model's performance, reliability, or another criterion.

iii. $\delta(M_i, M_j)$: is a similarity function that measures how similar the models M_i and M_j are. It could be based on performance metrics, predictions, or any other feature that can quantify similarity.

The equation is used to select the best model M_i from a set of n models by evaluating which models has the highest total weighted similarity with the other models in the set. Essentially, it finds the models that are most in agreement with the others (according to the similarity function δ), weighted by the importance of each model. And **D** is the decision, the selected model indexes. The models with the highest cumulative weighted similarity across all other models is chosen as the best or most representative model.

With several machine learning models predicting the same outcome. Each has a different performance, even though they may produce similar results. The equation helps determine which models are the most "trusted" based on how its predictions align with the other models, considering the reliability (weights) of each model's performance, to be selected as the final model. This is particularly useful in ensemble learning, where combining the outputs of multiple models often leads to better performance than using a single model.

C. *Optimized Parallel Voting Mechanism (OPVM)*

OPVM is an enhancement of traditional voting mechanisms where the weight of each model's vote is adjusted dynamically based on its performance and the confidence level of its predictions. This method of aggregating the outputs of parallel models to determine the most reliable prediction is based on majority voting, weighted voting, or other aggregation techniques.

$$P_{optimal} = Max(\sum_{i=1}^{n} w_i . R_i)$$

Where w_i are weights assigned to each model's result based on prior performance, and $P_{optimal}$ is the optimized prediction derived from the weighted sum of the models' outputs.

For a set of models M_i and their predictions y_j, the weighted vote V is computed as:

$$V = \sum_{i=1}^{n} (\alpha_i . y_i)$$

where α_i is the confidence of model M_i [12], [34]

D. *Time Complexity Reduction via Parallel Execution (TCRPE)*

The theory predicts that the overall time complexity of model selection can be reduced by executing multiple models in parallel, as opposed to sequentially, thus achieving faster convergence to the best model. TCRPE refers to the reduction in computational time achieved by leveraging parallel processing in training and evaluating machine learning models. The theory quantifies the trade-off between the number of processing units and the speedup in execution, from the principle of

$$T_{parallel} = \max(T_{f_1}, T_{f_2}, \dots, T_{f_n}),$$

where $T_{parallel}$ the time taken in parallel execution, compared to

$$T_{\mathbf{sequential}} = \sum_{i=1}^{n} T_{f_i}$$

for sequential execution.

The speedup S achieved by parallel execution is defined as,

$$S = \frac{T(n)}{T(n,P)},$$

where $T(n)$ is the time taken in a sequential process and $T(n, P)$ is the time taken using P processing units. Ideally, S approaches P, but in practice, it is limited by overheads and the non-parallelizable fraction of the task, as described by Amdahl's Law:

$$S = \frac{1}{f + \frac{1-f}{P}}$$

where f is the fraction of the task that is inherently serial [2], [35].

E. *Time Complexity Reduction via Parallel Execution (TCRPE)*

This theory posits that by combining parallelized model execution with optimized voting mechanisms, it is possible to achieve superior model selection and prediction accuracy in ensemble learning. The theory establishes that:

1. The consensus-based selection ensures that the chosen models are robust and reliable, potentially improving the overall accuracy of AI systems.

2. By formalizing parallel execution and voting, the theory leads to significant gains in computational efficiency, particularly in large-scale AI applications.

3. Effective parallelization of model training and evaluation significantly reduces computation time, allowing for the exploration of more complex models within a feasible timeframe [29], [30].

4. It extends ensemble learning by integrating parallelism directly into the all phases, allowing for more efficient and accurate model selection. By dynamically adjusting the voting mechanism based on model performance and confidence, the theory ensures that the ensemble's decision-making process is not only faster but also more reliable [12], [34].

5. This theory provides a framework for selecting the best machine learning models in scenarios where multiple models need to be evaluated rapidly. A consensus-based approach, refined by the OPVM, leads to the selection of the most robust models, thereby improving the overall accuracy of predictions [3], [32].

6. The theory is applicable in environments where computational resources allow for parallel execution, such as in distributed computing or cloud-based AI systems.

The theory predicts that model selection through OPEL is faster, and have higher Matthews Correlation Coefficient (MCC) and lower error rates as compared to those selection by traditional ensemble methods.

F. Algorithm Development

1. Initialize Data:
   - Prepare input data containing independent variables
   - Prepare output data containing dependent variables
   - Prepare a test set for prediction.
2. Set Parameters:
   - Set random generator seed for reproducibility.
   - Define convergence parameters like iterations and tolerance.
3. Model Training:
   - Initialize multiple machine learning models with different learning algorithms
   - Train each model using the input data and output data.
4. Model Evaluation:
   - Use each trained model to compute predictions for the test set.
   - Calculate evaluation metrics for each model using confusion matrices and Matthews Correlation Coefficients.
5. Determine Top-best Models:
   - Identify the top-best models based on the Matthews Correlation Coefficient.
   - Evaluate the performance of the best models by comparing them against the test set.
6. Output Results:
   - Display the results of each model, including the prediction status, error rate, and correlation coefficient.
   - Identify and display the indices of the best-performing models.

VI. RESULTS

The training time for the following machine learning models, Probabilistic Coordinate Descent (PCD), Iterative Reweighted Least Squares (IRLS), Sequential Minimal Optimization with Polynomial kennel (SMOP), Threshold Learning (THL), AdaBoost with Decision Stump (AdBDS), AdaBoost with Logistic Regression (AdBRL) and AdaBoost with Decision Tree (AdBDT), and the total training time for the serial processing given by the formula, $T_{\mathbf{sequential}} = \sum_{i=1}^{n} T_{f_i}$ and the parallel processing, given by the formula $T_{parallel} = \max(T_{f_1}, T_{f_2}, \dots, T_{f_n})$, on samples of size 35000 and 32500 is given in Table 1.

Table 1: Different Model Runtime

Sample (n)	32500		35000	
	Serial	Parallel	Serial	Parallel
PCD	142	387	178	268
IRLS	197	495	182	291
SMOP	84759	84638	48374	49070
THL	15	79	17	37
AdBDS	937	1779	411	893
AdBRL	607	1264	627	1151
AdBDT	3176	3867	3572	4009
Total	**89833**	**84638**	**53361**	**49070**

Achieving the following cumulative total serial and parallel processing time in milliseconds, taken to run samples of varying sizes (n) using serial computation and parallel computation runtime, as shown in Table 2, with their total speedups.

Table 2: Serial and Parallel Computation Runtime

	Runtime(ms)		
Sample (n)	Serial	Paralleled	SpeedUp
35000	60834	55226	1.10155
32500	100880	95243	1.05919
30000	89143	84894	1.05005
27500	56838	55008	1.03327
25000	51494	49402	1.04235
22500	33307	31150	1.06925
20000	32455	29777	1.08994
17500	33776	31685	1.06599
15000	31638	30387	1.04117
12500	14694	13368	1.09919
10000	11410	10345	1.10295

7500	6627	6264	1.05795
5000	4089	3593	1.13805
2500	1678	1212	1.38449
50	341	261	1.30651

Shown graphically in figure 1, with their trend line.

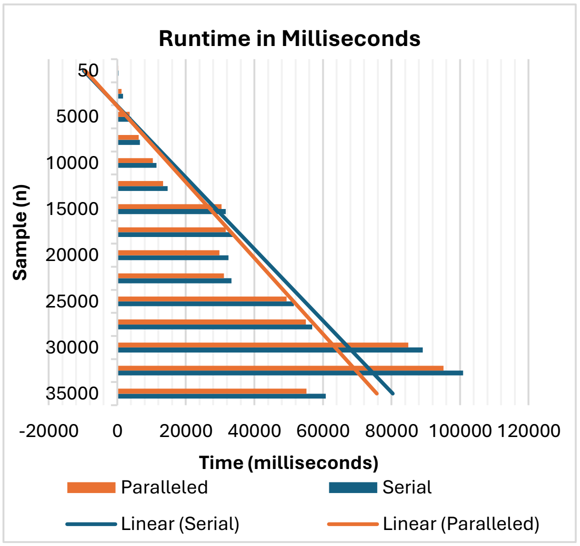

Figure 1: Runtime in milliseconds

Following that, was a voting mechanism that selects the top best-performing models dynamically, given by the formula, $M_{best} = Mode(R_1, R_2, \ldots, R_n)$, the final decision D, made by considering the consensus among the most performing models, given by

$$D = \operatorname{argmax}_{i\in\{1,\ldots,n\}} \sum_{j=1}^{m} \left(w_j . \delta(M_i, M_j)\right),$$

for the selected model indexes, using W_j, which is the weight assigned to the j-th model based on the model's performance, using the performance metrics for getting the similarity $\delta(M_i, M_j)$ of models M_i and M_j. The results as shown in Table 3, of the models dynamic selected using the index W_j based on the Matthews Correlation Coefficient (MCC) and lower error rates, for each varying sample sizes (n).

Table 3: Model Indeces and Performances

Samples	Index	Model	Error	Coefficient
	7	AdBDT	0.2354	0.42935
35000	0	PCD	0.23828	0.42246
	7	AdBDT	0.23529	0.43115
32500	5	AdBDS	0.23942	0.42392
	7	AdBDT	0.23517	0.43093
30000	0	PCD	0.23723	0.42608
	7	AdBDT	0.23585	0.42975
27500	2	SMOP	0.23484	0.42973
	7	AdBDT	0.23516	0.43008
25000	5	AdBDS	0.23784	0.42616
	2	SMOP	0.2324	0.43266
22500	7	AdBDT	0.23378	0.43165
	7	AdBDT	0.23385	0.43201
20000	5	AdBDS	0.2359	0.42794
	7	AdBDT	0.23331	0.43394
17500	5	AdBDS	0.23697	0.42889
	7	AdBDT	0.23287	0.43282
15000	2	SMOP	0.23373	0.42867
	7	AdBDT	0.23128	0.43657
12500	5	AdBDS	0.23792	0.42459
	7	AdBDT	0.2315	0.43644
10000	2	SMOP	0.2314	0.43438
	7	AdBDT	0.23093	0.43648
7500	2	SMOP	0.23093	0.43446
	7	AdBDT	0.2328	0.42127
5000	0	PCD	0.2354	0.41792
	2	SMOP	0.2348	0.42186
2500	7	AdBDT	0.25	0.39789
	5	AdBDS	0.06	0.85538
50	7	AdBDT	0.18	0.54554

In Table 3, the acronyms, PCD is Probabilistic Coordinate Descent, IRLS is Iterative Reweighted Least Squares, SMOP is Sequential Minimal Optimization with Polynomial kennel, THL is Threshold Learning, AdBDS is AdaBoost with Decision Stump, AdBRL is AdaBoost with Logistic Regression and AdBDT is AdaBoost with Decision Tree, in this context.

For statistical validation of the performance between serial and parallelised computation runtimes, it was done using a paired sample t-test, with the results shown in Table 4 and Table 5, using the software Minitab-21.4 for the Serial and Paralleled Computational runtime with data from Table 2.

Table 4: Descriptive Statistics for the Population of Serial and Parallel Computation

Descriptive Statistics				
Sample	N	Mean	StDev	SE Mean
Serial	15	35280	31339	8092
Paralleled	15	33188	29757	7683

Table 5:µ_difference: population mean of (Serial - Paralleled)

Estimation for Paired Difference				Test	
Mean	StDev	SE Mean	95% CI for µ_difference	T-Value	P-Value
2093	1784	461	(1104, 3081)	4.54	0.000

VII. DISCUSSION

From Table 3, the top-two-performing models selected using $M_{best} = Mode(R_1, R_2, \dots, R_n)$, varied with varying sample sizes (n), as the model's Matthews Correlation Coefficient (MCC) varied as well, which was used for selecting the model indexes. This proposed model selection mechanism effectively identified the top-performing models from a diverse set of candidates, leading to improved accuracy and robustness in the ensemble's predictions. The weighting algorithm ensured that models with higher reliability had a greater influence on the final decision, further enhancing the overall performance.

The integration of parallel processing techniques reduced the computation time compared to traditional ensemble methods. The framework demonstrated marginal performance, handling larger datasets and more complex models faster, showing the formula $T_{parallel} = \max(T_{f_1}, T_{f_2}, \dots, T_{f_n})$, computes faster than using $T_{\mathbf{sequential}} = \sum_{i=1}^{n} T_{f_i}$. This resulted in speedups, $S = \frac{T(n)}{T(n,P)}$, achieved by parallel execution from serial execution, especially for smaller samples.

The proposed framework consistently outperformed traditional ensemble methods in terms of accuracy and computational efficiency. The statistical validation in table 5, confirmed that these improvements were significant, from the paired sample t-test, where the t-value was 4.5 and the p-value was 0.00 indicate that there is a statistically significant difference between the paired samples being tested. A t-value of 4.5 is relatively high, suggesting that the difference between the means is much larger than what would be expected due to random variation alone, implying that it is highly unlikely that this difference occurred by chance. Therefore, there is strong evidence to suggest that the treatment or condition under comparison had a meaningful effect.

VIII. CONCLUSION

The Theory of Optimized Parallelized Ensemble Learning (OPEL) introduces a structured framework to enhance the efficiency and accuracy of ensemble learning by leveraging parallelization and optimized dynamic voting. This comprehensive approach, which integrates theoretical development with empirical validation, ensures the framework is both scientifically rigorous and practically relevant, addressing key challenges in contemporary ensemble learning. The study demonstrates that the proposed framework—incorporating dynamic model selection, optimized weighting, and parallel processing—offers substantial advantages over traditional methods, particularly in distributed and resource-constrained environments. The approach improves decision-making accuracy and enhances computational efficiency, making it a valuable tool for large-scale machine-learning applications. It aligns with established principles in parallel computing and ensemble methods while offering a novel platform for future research and practical application in machine learning. Building upon existing work in ensemble learning, parallel processing, and distributed systems, OPEL introduces significant innovations in dynamic weighted voting and real-time performance optimization. These advancements enable the framework to achieve superior scalability, flexibility, and robustness compared to traditional approaches.

IX. CONTRIBUTIONS TO THE BODY OF KNOWLEDGE

While the proposed theory shares some foundational ideas with the reviewed works, it diverges significantly in its approach to model integration, optimization, and parallelization. Unlike traditional methods such as Bayesian Model Averaging (BMA) or ensemble techniques like AdaBoost and Random Forests, which typically rely on a single type of base model (e.g., decision trees) and employ static or probabilistic voting schemes (e.g., majority voting or fixed-weighted voting), the proposed framework introduces a dynamic weighted voting mechanism. This mechanism adjusts weights in real time based on performance metrics such as accuracy and precision, enabling the system to adapt to evolving data distributions and resource availability, thereby enhancing overall robustness and performance.

Additionally, the proposed theory leverages parallel and distributed computing not only to scale individual models but also to enhance model training, combination, and voting mechanisms. It optimizes the integration of multiple models by minimising communication overhead, dynamically allocating resources, and ensuring load balancing across different computational nodes. Unlike most traditional parallel processing methods, which focus primarily on scaling individual models (e.g., parallel neural networks or distributed XGBoost), this framework addresses scalability at a broader level.

Designed to be highly scalable, the framework can handle large datasets and complex models across distributed environments. It is also flexible, allowing for the dynamic addition or removal of models based on performance metrics and available computational resources. Unlike existing approaches in federated learning or distributed deep learning, which often concentrate on specific scalability challenges related to the training of individual models, the proposed framework addresses scalability in both model integration and optimization. This ensures that the system can efficiently scale across both data and computational resources.

X. REFERENCES

[1] R. Hoyle and C. St, *Handbook of Structural Equation Modeling*. 2012.

[2] D. Bertsekas and J. Tsitsiklis, "Parallel and distributed computation : numerical methods / Dimitri P.

Bertsekas, John N. Tsitsiklis," *SERBIULA (sistema Librum 2.0)*, Jan. 1989.

[3] T. G. Dietterich, "Ensemble Methods in Machine Learning," in *International Workshop on Multiple Classifier Systems*, Springer, 2000, pp. 1–15. doi: https://doi.org/10.1007/3-540-45014-9_1.

[4] V. Kumar and A. Gupta, "Analyzing Scalability of Parallel Algorithms and Architectures," *J Parallel Distrib Comput*, vol. 22, no. 3, pp. 379–391, Sep. 1994, doi: 10.1006/JPDC.1994.1099.

[5] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter, "Pegasos: primal estimated sub-gradient solver for SVM," *Math Program*, vol. 127, no. 1, pp. 3–30, Mar. 2011, doi: 10.1007/s10107-010-0420-4.

[6] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," in *Proceedings of the April 18-20, 1967, spring joint computer conference on - AFIPS '67 (Spring)*, New York, New York, USA: ACM Press, 1967, p. 483. doi: 10.1145/1465482.1465560.

[7] S. Agarwal and C. R. Chowdary, "Combating hate speech using an adaptive ensemble learning model with a case study on COVID-19," *Expert Syst Appl*, vol. 185, p. 115632, Dec. 2021, doi: 10.1016/J.ESWA.2021.115632.

[8] S. Agarwal, A. Sonawane, and C. R. Chowdary, "Accelerating automatic hate speech detection using parallelized ensemble learning models," *Expert Syst Appl*, vol. 230, p. 120564, Nov. 2023, doi: 10.1016/j.eswa.2023.120564.

[9] W. Aldjanabi, A. Dahou, M. A. A. Al-qaness, M. A. Elaziz, A. M. Helmi, and R. Damaševičius, "Arabic Offensive and Hate Speech Detection Using a Cross-Corpora Multi-Task Learning Model," *Informatics*, vol. 8, no. 4, p. 69, Oct. 2021, doi: 10.3390/informatics8040069.

[10] P. Kapil and A. Ekbal, "A deep neural network based multi-task learning approach to hate speech detection," *Knowl Based Syst*, vol. 210, p. 106458, Dec. 2020, doi: 10.1016/J.KNOSYS.2020.106458.

[11] L. Breiman, "Bagging predictors," *Mach Learn*, vol. 24, no. 2, pp. 123–140, Aug. 1996, doi: 10.1007/BF00058655.

[12] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Trans Pattern Anal Mach Intell*, vol. 12, no. 10, pp. 993–1001, 1990, doi: 10.1109/34.58871.

[13] Y. Freund and R. E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," *J Comput Syst Sci*, vol. 55, no. 1, pp. 119–139, Aug. 1997, doi: 10.1006/jcss.1997.1504.

[14] L. Breiman, "Random Forests," *Mach Learn*, vol. 45, no. 1, pp. 5–32, 2001, doi: 10.1023/A:1010933404324.

[15] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei, "Hierarchical Dirichlet Processes," *J Am Stat Assoc*, vol. 101, no. 476, pp. 1566–1581, Dec. 2006, doi: 10.1198/016214506000000302.

[16] C. Cortes and V. Vapnik, "Support-vector networks," *Mach Learn*, vol. 20, no. 3, pp. 273–297, Sep. 1995, doi: 10.1007/BF00994018.

[17] G. Zanghirati and L. Zanni, "A parallel solver for large quadratic programs in training support vector machines," *Parallel Comput*, vol. 29, no. 4, pp. 535–551, Apr. 2003, doi: 10.1016/S0167-8191(03)00021-8.

[18] Youssef Aboelwafa, "Hotel Booking Cancellation Prediction," Kaggle. Accessed: Aug. 10, 2024. [Online]. Available: https://www.kaggle.com/datasets/youssefaboelwafa/hotel-booking-cancellation-prediction

[19] J. Dean *et al.*, "Large Scale Distributed Deep Networks," *Adv Neural Inf Process Syst*, Oct. 2012.

[20] C.-T. Chu *et al.*, "Map-Reduce for Machine Learning on Multicore.," *Adv Neural Inf Process Syst*, vol. 19, pp. 281–288, Jan. 2006.

[21] R. Cole and U. Vishkin, "Deterministic coin tossing with applications to optimal parallel list ranking," *Information and Control*, vol. 70, no. 1, pp. 32–53, Jul. 1986, doi: 10.1016/S0019-9958(86)80023-7.

[22] R. L. Graham, "Bounds for Certain Multiprocessing Anomalies," *Bell System Technical Journal*, vol. 45, no. 9, pp. 1563–1581, Nov. 1966, doi: 10.1002/j.1538-7305.1966.tb01709.x.

[23] R. P. Brent, "The Parallel Evaluation of General Arithmetic Expressions," *Journal of the ACM*, vol. 21, no. 2, pp. 201–206, Apr. 1974, doi: 10.1145/321812.321815.

[24] R. M. KARP and V. RAMACHANDRAN, "Parallel Algorithms for Shared-Memory Machines," *Algorithms and Complexity*, pp. 869–941, Jan. 1990, doi: 10.1016/B978-0-444-88071-0.50022-9.

[25] Y. Zhang, J. C. Duchi, and M. J. Wainwright, "Divide and Conquer Kernel Ridge Regression: A Distributed Algorithm with Minimax Optimal Rates," May 2013, [Online]. Available: http://arxiv.org/abs/1305.5029[26] C. Elkan, "Boosting And Naive Bayesian Learning," Dec. 1997.

[27] J. K. Pelekamoyo and H. M. Libati, "Considerations of an efficiency-intelligent geo-localised mobile application for personalised SME market predictions," *Measurement and Control*, Jul. 2023, doi: 10.1177/00202940231186675.

[28] Jephter Kapika Pelekamoyo and Hastings M. Libati, “Forecasting Market’s Demand And Supply With Machine Learning And Local Weather,” *International Journal of Scientific & Technology Research*, vol. 11, no. 1, pp. 115–119, Jan. 2022, Accessed: Jun. 23, 2023. [Online]. Available: http://www.ijstr.org/paper-references.php?ref=IJSTR-0421-45165

[29] A. Grama, G. Karypis, V. Kumar, and A. Gupta, *Introduction to Parallel Computing*, Second. Addison-Wesley, 2003.

[30] A. S. Tanenbaum and M. Van Steen, *Distributed Systems: principles and paradigms. Second Edition*. Pearson Education. Inc, 2007.

[31] V. Singh, V. Kumar, G. Agha, and C. Tomlinson, “Efficient algorithms for parallel sorting on mesh multicomputers,” *Int J Parallel Program*, vol. 20, pp. 95–131, Apr. 1991, doi: 10.1007/BF01407839.

[32] L. I. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms, Second Edition*. Wiley, 2004. doi: 10.1002/0471660264.

[33] M. Jordan, J. Kleinberg, and B. Schölkopf, *Pattern Recognition and Machine Learning*. New York, NY: Springer, 2006. Accessed: Aug. 30, 2024. [Online]. Available: https://link.springer.com/book/9780387310732

[34] Z.-H. Zhou, J. Wu, and W. Tang, “Ensembling neural networks: Many could be better than all,” *Artif Intell*, vol. 137, no. 1–2, pp. 239–263, May 2002, doi: 10.1016/S0004-3702(02)00190-X.

[35] J. L. Gustafson, “Reevaluating Amdahl’s law,” *Commun ACM*, vol. 31, no. 5, pp. 532–533, May 1988, doi: 10.1145/42411.42415.