

RESEARCH ARTICLE

OPEN ACCESS

NEW APPROACH FOR ACCELERATING DISEASE DETECTION USING A PARALLELISM-NASED CNN MODEL

Mohammed Amine Mami^{*1}, Nadir Berrouane² and Imene Baba Ahmed³

^{1,3}University of Oran1-Ahmed Ben Bella, Computer Science Department, Oran, Algeria

²University of Relizane, Computer Science Department, Relizane, Algeria

¹<http://orcid.org/0000-0002-6327-3650>, ²<http://orcid.org/0009-0003-3462-8373>, ³<http://orcid.org/0009-0007-8273-6490>

E-mail: aminehome@yahoo.fr, nadirberrouane@yahoo.com, babaahmedimene@gmail.com

ARTICLE INFO

Article History

Received on: March 07, 2026

Reviewed on: April 07, 2026

Accepted on: July 01, 2026

Published on: July 31, 2026

Keywords:

Deep Learning,
Machine Learning,
parallel programming,
Pyramid Histogram of Oriented
Gradient (PHOG),
Fourier Transform,
Discrete Cosine Transforms
(DCT).

ABSTRACT

In light of the recent pandemics like COVID phase in 2020 that the world is going through, with all the gaps and incompleteness in diagnostics, as well as the slowness of decision-making for patients, the interest of AI to solve in the first place the various problems linked to diagnosis and secondly having a rapid response to be able to intervene in time to save lives, becomes a priority of priorities.

This work follows work which aimed to reduce the training data necessary in the deep learning process to increase the possibility of its use in cases where time is essential and the contribution was on a basis of extracting features manually with extraction methods and techniques from machine learning combine with deep learning. In our approach, we keep the same logic and we apply or integrate the principle of parallel programming into the process in order to further reduce processing and decision-making time. This will give us a very important advantage in making decisions and helping doctors with diagnosis which will give an advantage over diseases.



Copyright ©2026 by authors and Galileo Institute of Technology and Education of the Amazon (ITEGAM). This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).

I. INTRODUCTION

In the present day, AI is evolving so rapidly, its applications are being applied at all levels, enabling significant advances in all domains: industry, daily life, speech and image processing, medical and military domains, among others, all of which have an impact on our lives. The AI encompasses several techniques and methods enabling machines (computers in most cases), among other things, to learn patterns from data without being explicitly programmed. Machine learning in general, and deep learning in particular, is proving particularly effective in medical imaging, which is our research domain. The use of deep learning offers a number of advantages, and its automated operation eliminates human intervention and is less influenced by human subjective factors, leading to objective results. However, there are still a number of challenges to be met, despite the extent of the advances and impacts in the medical world. With the recent pandemic, and the need to treat too many patients in record time, learning with a small amount of data in a short space of time has led us to review certain techniques and try to combine them with others in order to achieve an effective solution while also minimizing dependence on expensive hardware resources. This leads us to find innovative solutions to reduce learning time and adapt Artificial Intelligence algorithms to maximize results while at the same time minimizing data and time constraints. The most frequently tested methods in the medical imaging domain are, depending on their degree of integration alone or in combination with other methods, Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), Generative Antagonistic Networks (GAN) and auto-encoders. Medical imagery, in all its forms - X-rays, MRIs, scans, etc. - contains relevant information to enable doctors to diagnose, monitor and treat patients. The analysis of these images using AI gives doctors and researchers a considerable advantage, enabling them to act faster and in real time to save lives and improve patient follow-up. In our approach, the proposal of solutions aimed at reducing execution time by integrating the notion of parallelism in the learning phase to save time, in addition to combining knowledge extraction methods to be able to apply deep learning even with a limited knowledge base.

II. MATERIALS AND METHODS

In the medical imaging domain, a great many works in recent years have centered on several methods, but more especially the method that attracts the most attention in this field is deep learning, which gives good results for improving diagnosis, early detection of disease, pathology monitoring and treatment prescribing. For improved diagnosis, [1] investigates the effectiveness of image feature extraction and deep learning (CNN), in improving the diagnosis of COVID-19 from chest X-ray images. Based on a two-stage process, the first step is to extract image features, using algorithms such as PHOG, Fourier, GABOR and DCT, followed by image classification using CNN in the second step. The results generated by the simulations demonstrate an accuracy advantage over traditional machine learning techniques such as SVM, KNN and Naïve Bayes. For therapy prescription, [2] introduces a novel approach to early detection of Alzheimer's disease to predict disease progression and subsequently personalize treatment, using deep learning with the EfficientNet-B5 architecture to analyze MRI images of the brain. In terms of monitoring patients with various diseases, [3] illustrates the contribution of deep learning in analyzing data by means of the EchoNet model to identify features, cardiac in this study, by estimating cardiac function such as pacemaker lead detection, left atrium volume and left ventricular hypertrophy with an accuracy superior to human interpretation. The success in also predicting systemic phenomena such as age and gender demonstrates the potential of deep learning to improve cardiac (and not only) diagnostics and cardiovascular risk prediction. Still for early detection, [4] showing in the domain, for example, of breast cancer screening via the interpretation of mammographic images with precision reducing the rate of false positives and negatives using deep learning.

The benefits of using deep learning to achieve higher performance are not without their challenges, and specific solutions are needed to overcome difficulties such as resolution problems, real-time constraints, and the management of distributed systems, with the introduction of complexity in asynchronous communications and fault management. Several innovative techniques can considerably reduce the time needed for deep learning and inference, with better computing capacities (GPU, TPU [5], and distributed systems with data parallelism [6] or model parallelism [7]. Many recent works use distributed deep learning for the optimal execution of different processes. [8] explores methods for parallel and distributed training of deep neural networks (DNNs) on the prototype Tianhe-3 system, aiming to accelerate DNN training by exploiting the massive power available. [9] suggests optimizations of Cyclic Weight Transfer (CWT) to improve distributed learning in medical imaging in the face of data variability and by optimizing the number of iterations according to sample size, the cyclic learning rate to promote convergence, adjusting sampling and introducing a cyclic weight in the loss function. Some of the methods used, applied in addition to deep learning, in the medical imaging domain, without of course listing them all, the most used being classical neural networks (ANN), feature-based methods (SVM, K-NN), transfer learning, image segmentation, generative adversarial networks (GAN), Region Proposal Network (RPN), Visual Saliency Maps, etc. Representing just a small glimpse, these studies illustrate the powerful potential impact of deep learning in helping to transform medical imaging, offering unprecedented opportunities to improve the quality and efficiency of healthcare, according to our domain, and in other domains without exception.

III. PARALLEL AND DISTRIBUTED PROGRAMMING

Parallel and distributed programming is an essential approach for fully exploiting the power of modern computer systems. It enables complex problems to be solved by dividing tasks and executing them simultaneously on multiple processors or computers.

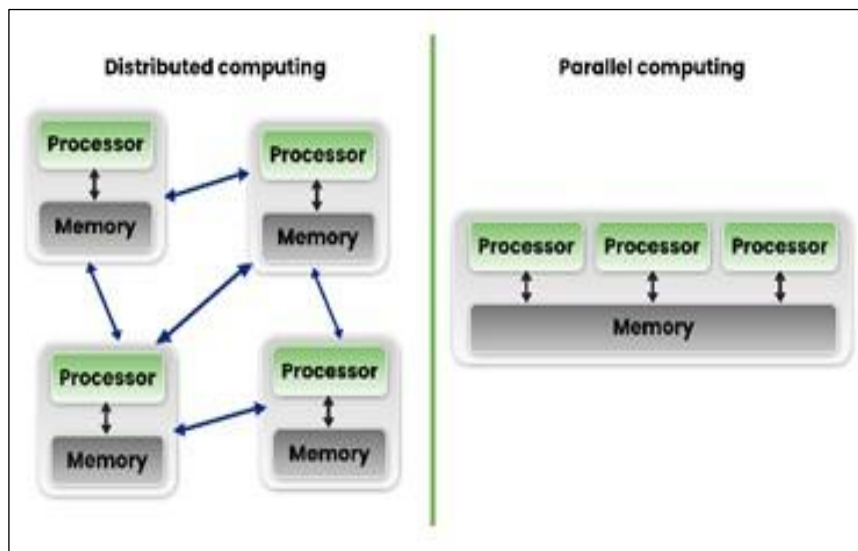


Figure 1: Parallel and Distributed Programming.
Source: [10].

III.1 DISTRIBUTED DEEP LEARNING

This section provides an overview of existing approaches for distributing deep neural network training across multiple machines or GPUs. Training models on massive volumes of data presents considerable challenges when limited to a single machine. To overcome these obstacles, these techniques aim to accelerate the training process, optimise the use of available resources, and reduce the limitations associated with the memory and computing capacity of a single system. However, they do not always take into account all the constraints specific to distributed systems, such as variable communication delays, fault tolerance, or large-scale management. Several approaches have been proposed in the literature to solve these problems and improve the scalability of deep learning [7].

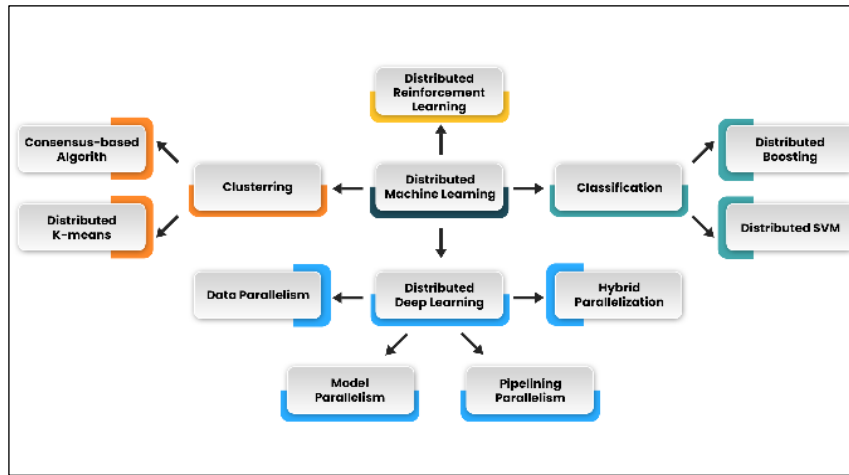


Figure 2: Distributed Machine Learning and Deep Learning Algorithms.
Source: [11].

There are two main approaches to distributed deep learning: model parallelization and data parallelization. Both strategies optimize neural network training by distributing tasks across multiple resources. Data parallelism figure Fig. 3. is considered one of the simplest forms of parallelism. It involves replicating the model, optimizer parameters and training code across all available devices (machines), it's speeds up the processing of the overall dataset by utilizing multiple devices simultaneously. It also allows for larger overall batch sizes without requiring a large number of gradient accumulations. However, in order to function correctly, each replica must be able to load the entire model with at least one micro-batch instance [12]. Model parallelism figure Fig. 4. consists of distributing the computational load of inference and backpropagation in a deep neural network across multiple machines or processors. This distribution can be done at different levels of granularity; assigning different layers of the network to different nodes, divide the neurons of the same layer between several devices or like some approaches distribute each matrix operation independently in order to optimise the load between agents as much as possible [7].

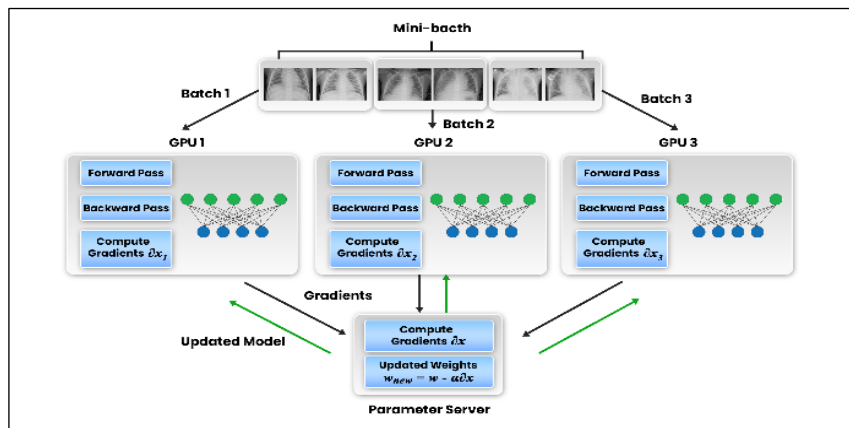


Figure 3: Data Parallelizing.
Source: [13].

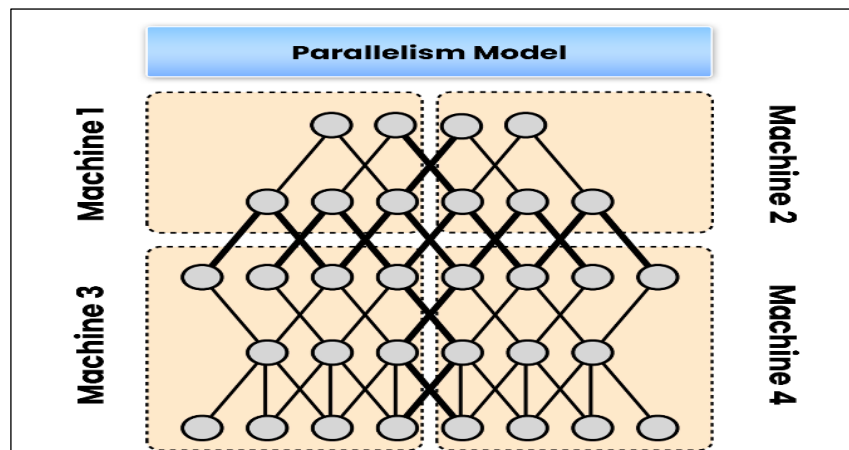


Figure 4: Model parallelism.
Source: [14].

IV. PROPOSED APPROACH

Based on what we have mentioned in the previous section, we have decided to use CNN reason of its effective results on a dataset containing images specially on classification problems. The approach is divided into six phases.

- Network and architecture configuration phase.
- Model Preprocessing phase.
- Parallelization of convolution processing.
- Model training parallelization phase.
- Model testing phase.
- Model performance evaluation phase.

IV.1 NETWORK AND ARCHITECTURE CONFIGURATION

This section describes the network configuration adopted and the master-slave architecture implemented as part of our experiments. The aim is to provide a clear overview of the distributed environment used, which is essential for implementing parallel processing and evaluating system performance. In order to ensure fast, stable and efficient communication, we interconnected three machines within a local area network (LAN). The configuration of this network and the steps involved in setting it up are presented below:

- Connecting the machines to the same local area network.
- Assigning IP addresses.
- Configuring a hosts file.
- Configuring SSH access between machines.

IV.2 MODEL PREPROCESSING PHASE

This phase consists of designing and implementing the CNN model for the training model with the Train folder of the dataset. Our architecture is inspired by high-performance CNN models while being adapted to the specific constraints of our problem. The proposed architecture, illustrated in Figure Fig. 5, is defined to many layers as below: consists of

- Four convolution blocks
- Fully connected layers.
- Each convolution block includes a convolution layer using ReLU activation and a maximum pooling layer for dimensionality reduction.
- The number of filters increases gradually (16, 32, 64, 128) to allow the network to extract increasingly complex and abstract features from the images.
- The extracted features are then flattened via a Flatten layer and passed to two dense layers.
- The first dense layer with 512 units and ReLU activation learns the relationships between features,
- The output layer uses sigmoid activation to produce a binary classification probability.

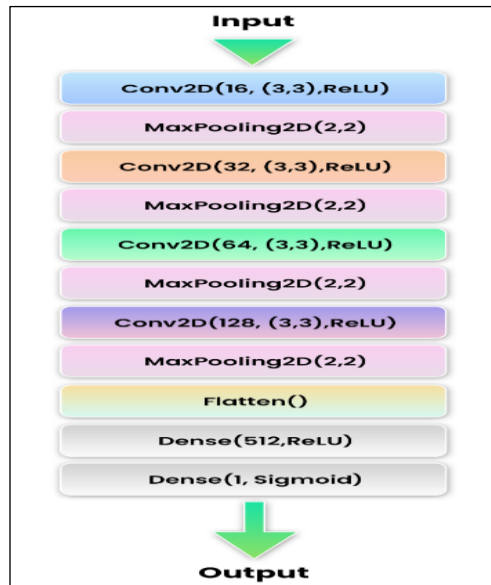


Figure 5: CNN Model Architecture.
Source: Authors, (2026).

IV.3 PARALLELISATION OF CONVOLUTION PROCESSING

The parallel convolution function was designed to distribute convolution filters and input data among the various nodes in the system. It enables convolution operations to be performed locally on each node, followed by a results aggregation phase. This function takes the following parameters:

- Layer: the convolution layer to be applied.
- Data: the input data on which the convolution will be performed.
- Rank: the rank of the process in the MPI hierarchy (rank 0 corresponds to the master node, the others to the slave nodes).
- Size: the total number of MPI processes currently running.
- Comm: The MPI communicator used for communication between processes [15].

IV.4 MODEL TRAINING PARALLELIZATION PHASE

Our parallelization approach for training convolutional neural networks is based on the principle of data parallelism. This technique allows the computational load to be distributed efficiently across multiple processing nodes, significantly reducing training time while potentially improving the performance of the final model. The parallelization strategy consists of dividing image batches between the different MPI processes available. To do this, we calculate the number of batches to be processed per process and use a different seed for each process when initializing the data generators. This approach ensures that each process works on a different subset of the training data. Each MPI process independently and in parallel trains a complete copy of the CNN model on its own subset of the data.

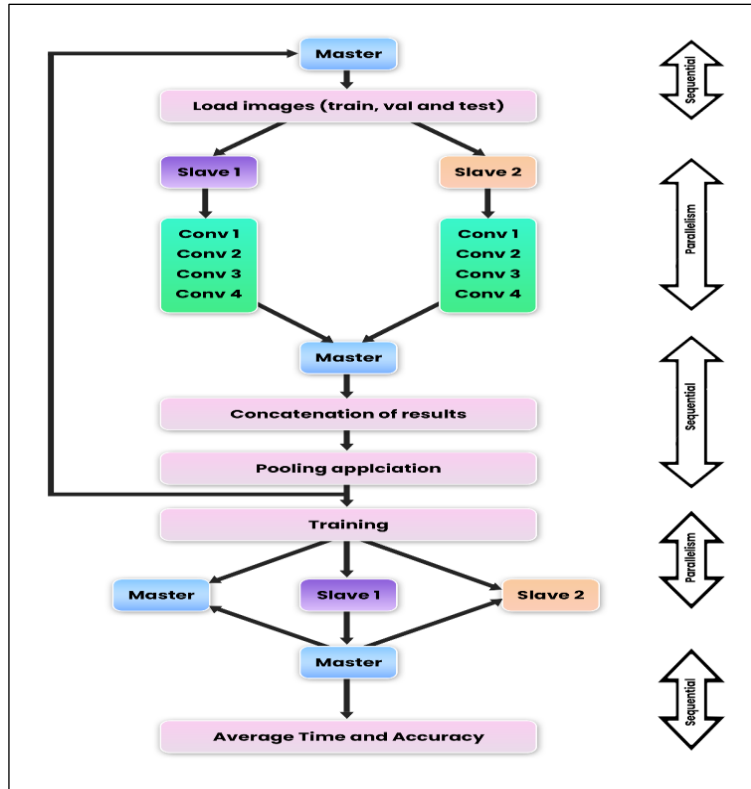


Figure 6: Diagram illustrating the proposed solution.
Source: Authors, (2026).

IV.5 MODEL TESTING PHASE

This phase consists of testing the trained model from phase C utilizing image from the test folder of our dataset.

IV.6 MODEL TESTING PHASE

This phase is about evaluating the model using classification metrics such as Confusion Matrix, Accuracy, Precision, Sensitivity, and Specificity. These metrics help us to be able to compare the results of different execution methods of the CNN model (Our approach; parallel convolution and training, Parallelized convolution only, sequential execution). We will show in section 6 that our approach gives the best results.

V. EXPERIMENTAL RESULTS

In this section, we present the experimental results obtained by the proposed approach and compare them to classic approach.

V.1 DATASET COLLECTION

The used dataset in this experience is collected by Daniel Kermany Kang Zhang Michael Goldbaum and it is available on Mendeley data website. It's also published by Paul Mooney in Kaggle website. The data is composed of 5856 images of chest X-rays and it is divided into two categories: 1583 images of healthy people and the rest are those who have Pneumonia [16]. The images used in this work are illustrated in Fig. 7. The figure shows the images used for experimentation. The first row is a normal image and the second row shows a Pneumonia image. In figure Fig. 8. We illustrate the structure of this dataset.

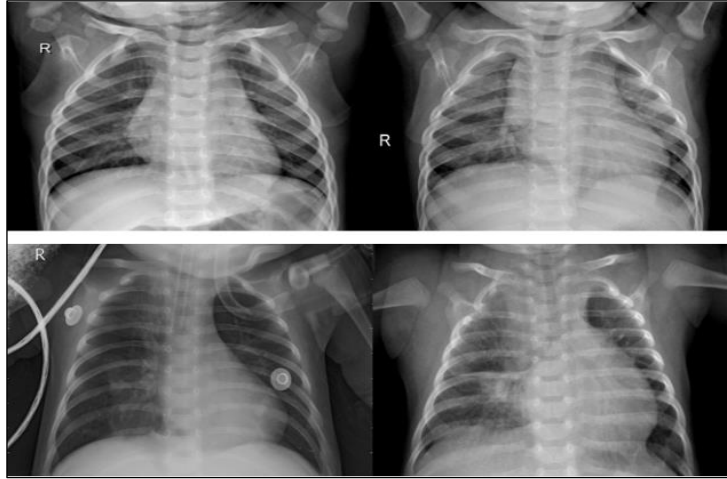


Figure 7: The images used in this work.
Source: Authors, (2026).

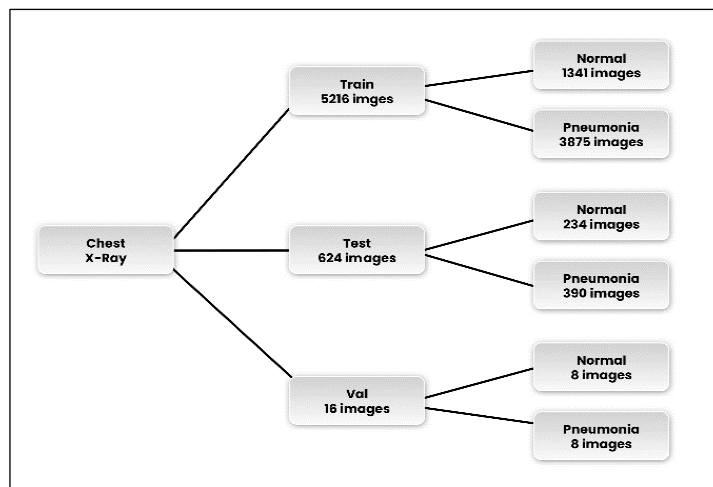


Figure 8: The structure of the Dataset.
Source: Authors, (2026).

VI. RESULTS

VI.1 RESULTS OF EXECUTION FOR SEQUENTIAL AND CONVOLUTION PARALLEL CNN MODEL

The results obtained from 15 independent runs are summarized in Table 1 and 2, which presents descriptive statistics for training time and model accuracy. Analysis of the results obtained with the sequential model (Table 1) shows that the CNN model executed sequentially has an average training time of 2777.74 seconds, with a standard deviation of 83.59 seconds, reflecting relative temporal stability between different executions. The execution time varies from 2,667.06 s to 2,897.92 s, a maximum deviation of approximately 230 seconds. In terms of model performance, the average accuracy reaches 90.59%, with a standard deviation of 0.0228, indicating good robustness and consistency of predictions. The minimum accuracy observed is 83.49%, while the maximum value reaches 92.95%, highlighting moderate but acceptable variability in the context of learning without optimization. For the CNN model with convolution parallelization (Table 2), the results indicate an average training time of 3023.68 seconds, with a standard deviation of 158.57 seconds, reflecting moderate variability between different runs. The shortest time observed was 2758.57 seconds, while the longest was 3229.63 seconds. In terms of performance, the average accuracy reached 91.08%, which is slightly higher than the result obtained in the sequential version. The standard deviation remains reasonable (2.05%), indicating overall consistency of the model, despite some variations between runs.

Table 1: Performance statistics (time and accuracy) of sequential training of the CNN model.

	Time(S)	Precision
Average	2777,7407	0,9059 (90,59%)
Standard deviation	83,5931	0,0228 (2,28 %)
Minimum	2667,0600	0,8349 (83,49 %)
Maximum	2897,9200	0,9295 (92,95 %)

Source: Authors, (2026).

Table 2: Performance statistics (time and accuracy) of sequential training of the CNN model.

	Time(S)	Precision
Average	3023,6794	0,9108 (91,08 %)
Standard deviation	158,5696	0,0205 (2,05 %)
Minimum	2758,5724	0,8766 (87,66 %)
Maximum	3229,6260	0,9439 (94,39 %)

Source: Authors, (2026).

The graph (Fig. 9) below illustrates the variation in training time (in seconds) and accuracy obtained during the 15 simulations performed to assess the stability of the model's performance.

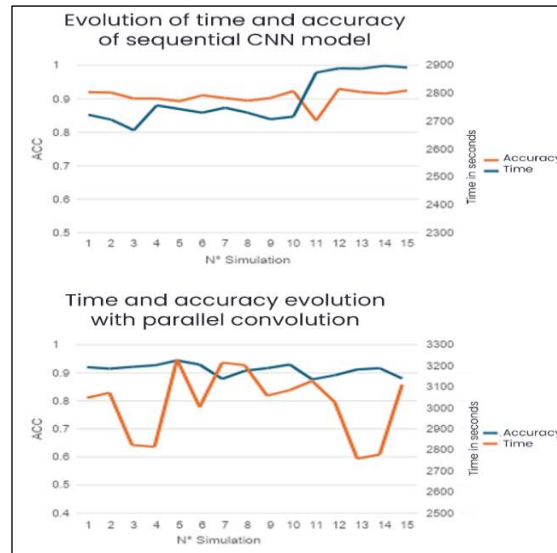


Figure 9: Time evolution and accuracy of sequential and convolution parallel CNN model.

Source: Authors, (2026).

Fig.10 illustrates the accuracy curve obtained directly when executing the sequential CNN code over 40 epochs. It shows how accuracy improves over the course of training.

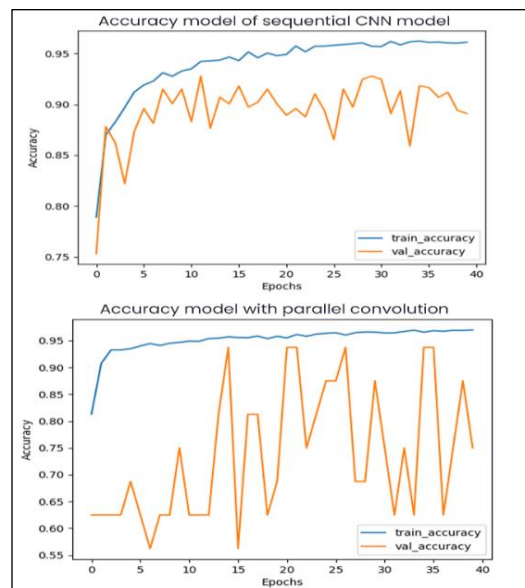


Figure 10: Evolution of model accuracy of sequential and convolution parallel CNN model.

Source: Authors, (2026).

VI.1 RESULTS OF EXECUTION FOR MODEL WITH PARALLEL CONVOLUTION AND TRAINING

The results obtained from 15 independent runs are summarised in Table 3, which presents descriptive statistics for training time and model accuracy.

Table 3: Performance statistics (time and accuracy) of training of the CNN model with parallelisation of convolution and training.

	Time(S)	Precision
Average	965,8547	0,9055 (90,54 %)
Standard deviation	4,5443	0,0101 (1,01 %)
Minimum	957,1146	0,892 (89,20 %)
Maximum	971,7502	0,9246 (92,46 %)

Source: Authors, (2026).

The total average training time observed is approximately 965.8547 seconds, or about 16.1 minutes, which corresponds to an average duration of 24 seconds per epoch. These results highlight the stability of execution time in a parallelized context. The table below presents the main statistics recorded during the various simulations. The accuracy of the parallelized CNN model was measured at the end of each training epoch. It was monitored throughout the process, revealing a gradual and steady improvement in performance. The final results demonstrate the stability and effectiveness of the parallelized version of the model, with an average accuracy of 90.54% and low dispersion between the different simulations. The following graph (Fig. 11) illustrates the evolution of training time and accuracy measured for each of the 15 simulations performed, in order to visualize the stability and performance of the parallelized model.

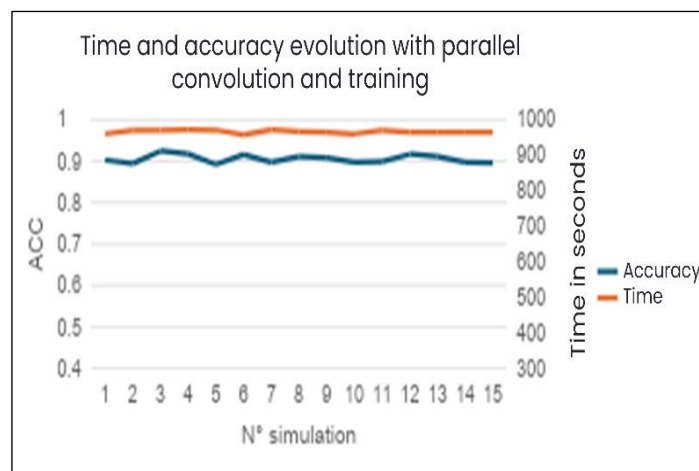


Figure 11: Evolution of time and accuracy with parallel convolution and training.

Source: Authors, (2026).

The following figure (Fig. 12) also shows the accuracy curve obtained during training of the parallelized CNN model, as displayed directly by the program. It reflects the gradual evolution of the model's performance over the course of 40 epochs.

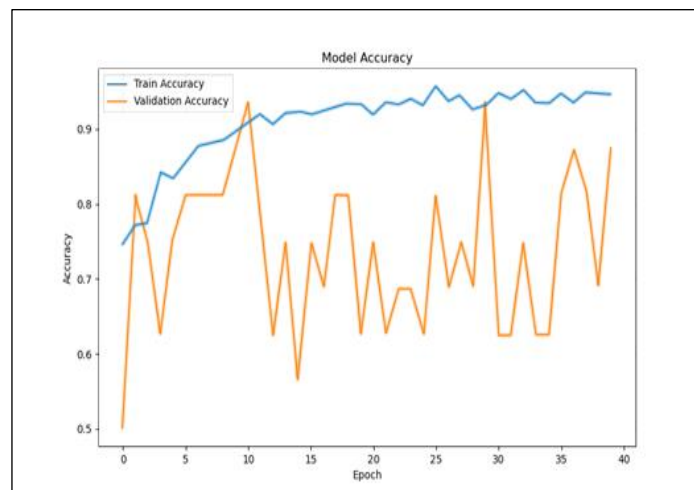


Figure 12: Evolution of model accuracy with parallel convolution and training.

Source: Authors, (2026).

VII. DISCUSSIONS

The table below summarises the average performance observed on the three versions of the CNN model tested: sequential execution, the version with parallelisation of the convolution layer only, and the version with parallelisation of both the convolution and the training process.

Table 4: Comparative results.

Model version	Parallelised convolution only	Parallelised Convolution and Training
Configuration	3 VMs (8 CPU, 16 Go RAM)	3 VMs (8 CPU, 16 Go RAM)
Average time	3023,6794	965,8547
Average accuracy	91,0800	90,5400

Source: Authors, (2026).

VII.1 COMPARISON OF EXECUTION TIMES

Analysis of training times reveals a significant reduction in average time when the model is parallelized at both the convolution and training levels. With an average time of 965.85 seconds, this version is approximately three times faster than the other two, which have durations of 2777.74 seconds (sequential) and 3023.68 seconds (convolution only). On the other hand, parallelizing only the convolution layer results in a slight increase in execution time, probably due to the overhead of communication and thread management in a distributed environment.

VII.2 COMPARISON OF ACCURACIES

In terms of accuracy, the three versions of the model achieve broadly comparable results, with a slight advantage for the version with parallelized convolution alone (91.08%), compared to 90.59% for the sequential version and 90.54% for the fully parallelized version. These differences remain small (less than 1%) and can be attributed to natural variations during training or effects related to parallelization, but no significant loss of performance is observed despite the significant reduction in time.

VIII. CONCLUSIONS

The work carried out is in the field of Deep Learning and distributed systems. We are particularly interested in the introduction of parallelism in medical image processing. Our project focuses on the application of parallelism techniques to improve efficiency and reduce costs in medical image processing. Our experiment consisted of testing three scenarios: parallelism in a distributed system for convolution only, parallelism in a distributed system for Convolution and Training, and sequential execution. It proved that it is possible to leverage the potential of deep learning even with limited machines. The higher the number of nodes, the lower the learning time can be. However, it is crucial to effectively manage synchronization and bottlenecks to avoid inefficiencies. Using libraries like mpi4py is key to handling these challenges. This study demonstrated that, despite hardware constraints, Deep Learning can be optimized for resource-constrained environments, paving the way for broader and more accessible applications in the medical field. To go even further, the use of Horovod, a parallelization library for Deep Learning, could offer better management of gradient communication and synchronization, reducing overhead and improving overall performance. In addition, the adoption of advanced models such as EfficientNet, which have demonstrated better performance in medical image processing, could also contribute to the improvement of these systems. The addition of a GUI interface would facilitate the use of this system by non-experts, thus making the technology more accessible. Finally, the integration of enhanced security measures is crucial to protect sensitive medical data and ensure the confidentiality and integrity of the processed information. The identified improvement prospects offer opportunities to continue improving the performance and efficiency of this system in the future.

IX. AUTHOR'S CONTRIBUTION

Conceptualization: Mami Mohammed Amine, Nadir Berrouane and Imene Baba Ahmed.

Methodology: Mami Mohammed Amine and Nadir Berrouane.

Investigation: Mami Mohammed Amine and Nadir Berrouane.

Discussion of results: Mami Mohammed Amine, Nadir Berrouane and Imene Baba Ahmed.

Writing – Original Draft: Mami Mohammed Amine and Nadir Berrouane and Imene Baba Ahmed.

Writing – Review and Editing: Mami Mohammed Amine and Nadir Berrouane.

Resources: Nadir Berrouane.

Supervision: Mami Mohammed Amine and Nadir Berrouane.

Approval of the final text: Mami Mohammed Amine, Nadir Berrouane and Imene Baba Ahmed.

X. REFERENCES

- [1] N, Berrouane. M, Benyettou. I, Benchennane. (2022). Deep Learning and Feature Extraction for Covid 19 Diagnosis.
- [2] Singh, C. Prabha, Hari Mohan Dixit, Shalini Kumari. (2023). Alzheimer Disease Detection using Deep Learning. Retinder deep.
- [3] Amirata Ghorbani, David Ouyang, Abubakar Abid, Bryan He, Jonathan H Chen, Robert A Harrington, David H Liang, Euan A Ashley, James Y Zou. (2020). Deep learning interpretation of echocardiograms.
- [4] Improving breast cancer screening with artificial intelligence. (2022). AI in Mammography.
- [5] Sumit Saha. (2018). A comprehensive guide to convolutional neural networks—the eli5 way. Towards Data Science, 15.
- [6] Ming Chen. (2022). Analysis of Data Parallelism Methods with Deep Neural Network.

- [7] Corentin Hardy. (2019). Contribution au développement de l'apprentissage profond dans les systèmes distribués. PhD thesis, Rennes 1.
- [8] Jia Wei, Xingjun Zhang, Zeyu Ji, Jingbo Li & ZhengWei. (2021). Deploying and scaling distributed parallel deep neural networks on the Tianhe-3 prototype system.
- [9] N. Balachandar, Ken Chang, Jayashree Kalpathy-Cramer, D. Rubin. (2020). Accounting for data variability in multi-institutional distributed deep learning for medical imaging.
- [10] K. Josue, (2025). Programmation parallèle et distribuée, Institut Supérieur d'Informatique et de Gestion.
- [11] Z. Y. Mohammad Dehghani, (2023). From distributed machine to distributed deep learning: a comprehensive survey, Journal of Big Data, vol. 10, n° 1158.
- [12] A. McKinney, (2023). A Brief Overview of Parallelism Strategies in Deep Learning, 26 february 2023. [Online] Available: <https://afmck.in/posts/2023-02-26-parallelism/>.
- [13] M. Bekuzarov, (2023). An Introduction to Parallel and Distributed Training in Deep Learning, 2 Jul 2023. [Online]. Available: <https://medium.com/better-programming/parallel-and-distributed-training-in-deep-learning-for-beginners-part-1-introduction-612a4534a117>.
- [14] Tamanna, (2023). Model Parallelism Techniques and Optimizations for Deep Learning Models, 3 Jul 2023. [Online]. Available: <https://medium.com/@tam.tamanna18/model-parallelism-techniques-and-optimizations-for-deep-learning-models-3c9f1e81fb2a>.
- [15] A. Y. Zohra, (2024). Introduction du parallélisme dans le traitement d'image médicale par le deep learning, université Oran1 - Ahmed Ben Bella, Oran.
- [16] K,Daniel, Z,Kang, G,Michael (2018). "Labeled Optical Coherence Tomography (OCT) and Chest X-Ray Images for Classification", Mendeley Data, V2, doi: 10.17632/rscbjbr9sj.2.