INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL
Online ISSN 1841-9844, ISSN-L 1841-9836, Volume: 21, Issue: 5, Month: October, Year: 2026
Article Number: 7566, https://doi.org/10.15837/ijccc.2026.5.7566

Scalable Distributed Deep Learning for Lung Disease Diagnosis: A Spark–Elephas Cross-Modality Framework with GPU Cluster Scalability Analysis

Zeineb BEN MESSAOUD, Maher BACCAR, Heni BOUHAMED, Monia HAMDI

Zeineb BEN MESSAOUD
Digital Research Center of Sfax, Sfax, Tunisia
Laboratory of Signals, SysteMs, Artificial Intelligence and Networks (SM@RTS)
zeineb.benmessaoud@crns.rnrt.tn

Maher BACCAR
Higher Institute of Computer Science and Multimedia Gabes, Gabes, Tunisia
maher.baccar@isimg.tn

Heni BOUHAMED
Advanced Technologies for Image and Signal Processing Unit (ATISP), Sfax University, Sfax, Tunisia
Research and Development Department, Zetta-Spark, Sfax, Tunisia
heni.bouhamed@fsegs.usf.tn

Monia HAMDI*
Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia
mshamdi@pnu.edu.sa
*Corresponding author

Abstract

The rapid growth of large-scale medical imaging datasets has exposed critical limitations of single-node deep learning workflows for clinical decision support in lung disease diagnosis. We present a unified, Spark-based distributed framework for scalable multi-class lung disease classification from both chest X-ray (CXR) and computed tomography (CT) images. Our approach integrates Apache Spark with Elephas to enable synchronous and asynchronous data-parallel training of Keras models via a parameter-server architecture on a GPU-enabled cluster, supporting binary and multi-class classification across multiple pulmonary pathologies including COVID-19 pneumonia, viral pneumonia, lung opacity, and normal cases. We implement transfer learning with DenseNet169, ResNet50, and MobileNetV2 and evaluate three classification tasks: binary lung disease detection and three-class diagnosis on CXR images, and three-class diagnosis on CT scans. Experiments on large public datasets demonstrate that the proposed system achieves high diagnostic performance, reaching 97.63% accuracy for three-class CXR and 95.49% for binary CT classification, while providing substantial runtime gains over single-node baselines. On a multi-node Spark cluster, synchronous training attains near-linear to super-linear scaling with speedups up to 6.26× and parallel efficiency exceeding 156%, attributed to improved cache utilization and reduced

I/O contention across distributed workers. Comparative analysis consistently demonstrates that synchronous parameter updates outperform asynchronous training in both convergence stability and final diagnostic accuracy across all tasks and modalities. This work delivers an end-to-end, cross-modality, distributed deep learning pipeline with explicit scalability characterization. The proposed framework is generalizable beyond pulmonary imaging and demonstrates how cluster computing infrastructures can be leveraged to build scalable, high-throughput experimental pipelines for multi-class lung disease classification in research settings.

Keywords: Distributed transfer learning, Multi-class lung disease diagnosis, Apache Spark–Elephas, Cross-modality medical imaging, Parameter-server architecture, Synchronous vs. asynchronous training, Scalability characterization.

1 Introduction

Lung disease represents one of the leading causes of morbidity and mortality worldwide, encompassing a broad spectrum of pulmonary pathologies including COVID-19 pneumonia, viral pneumonia, lung opacity, and other thoracic abnormalities that place sustained and growing demands on clinical diagnostic infrastructures [21, 28]. Accurate and timely differentiation among these conditions is clinically critical, as their radiographic presentations frequently overlap, making automated multi-class classification from medical images both a challenging and high-impact research problem. Chest X-rays (CXR) and Computed Tomography (CT) are the primary imaging modalities employed in this context, offering complementary diagnostic perspectives that are widely available across healthcare settings and exhibit characteristic radiographic patterns associated with distinct pulmonary conditions [21, 28]. Leveraging these modalities effectively at scale requires more than accurate models; it requires computational infrastructures capable of training and serving deep learning workloads over large, heterogeneous datasets.

Deep Convolutional Neural Networks (CNNs) have demonstrated strong and consistent performance for automated lung disease detection and classification from both CXR and CT images. However, training such models on large-scale, multi-modal datasets covering multiple pathological categories is computationally expensive and routinely exceeds the capabilities of single-node systems. These bottlenecks are further amplified when tackling complex multi-class tasks, benchmarking multiple CNN architectures simultaneously, or conducting cross-modality evaluations at clinical scale. The challenge is therefore not merely one of model design, but equally one of computational infrastructure: scalable, fault-tolerant, and resource-efficient training pipelines are a prerequisite for deploying high-throughput diagnostic systems in real-world healthcare environments.

Cluster computing frameworks such as Apache Spark [29, 30] offer a natural and well-established solution by providing distributed data processing, fault tolerance through Resilient Distributed Datasets (RDDs), and dynamic resource elasticity. Nevertheless, integrating GPU-accelerated deep learning workflows with Spark in a manner that preserves both scalability and diagnostic model performance remains non-trivial. Existing approaches have largely addressed this challenge in isolation—either focusing on single imaging modalities, restricting evaluation to binary classification, operating on small datasets that limit generalizability, or providing only limited scalability characterization on real multi-node hardware [2, 3, 4, 15, 16, 19, 20]. To the best of our knowledge, very limited work has delivered a unified, end-to-end distributed pipeline that simultaneously addresses multi-class lung disease classification across both CXR and CT modalities, benchmarks established pre-trained CNN architectures under distributed execution, and rigorously characterizes synchronous versus asynchronous training dynamics on a real GPU cluster.

To address these gaps, we present a unified Spark–Elephas distributed deep learning framework for scalable multi-class lung disease diagnosis from both CXR and CT images. Our system couples Apache Spark with Elephas to implement data-parallel training of Keras models via a parameter-server architecture on a GPU-enabled multi-node cluster. Within this framework, we perform transfer learning with three widely used CNN backbones DenseNet169, ResNet50, and MobileNetV2 across three classification tasks: binary and three-class classification on CXR, and three-class classification on CT, spanning pathological categories including COVID-19 pneumonia, viral pneumonia, lung opacity, and normal cases. The resulting pipeline is end-to-end, covering data ingestion, preprocessing, distributed

training, and evaluation, and is explicitly architected for generalizability beyond the pulmonary domain.

From a distributed systems perspective, a central objective of this work is not only to obtain high diagnostic accuracy, but to rigorously characterize the scalability, efficiency, and training dynamics of synchronous and asynchronous distributed schemes on a real multi-node, multi-GPU Spark cluster. We report detailed measurements of runtime, speedup, and parallel efficiency under varying cluster configurations and training strategies, and analyze the observed super-linear scaling effects in the context of cache behavior and I/O contention reduction.

This work makes the following specific contributions:

1. **Unified cross-modality distributed framework:** An end-to-end Spark–Elephas pipeline for data-parallel training of Keras CNNs on large-scale lung disease imaging data, supporting both CXR and CT modalities and binary and multi-class classification across four pulmonary pathology categories.

2. **Synchronous vs. asynchronous training on a real GPU cluster:** Synchronous and asynchronous parameter-server-based strategies are rigorously evaluated on a multi-node, multi-GPU Spark cluster, with synchronous updates shown to yield more stable convergence and consistently higher diagnostic accuracy.

3. **Distributed benchmarking of established CNN backbones:** DenseNet169, ResNet50, and MobileNetV2 are benchmarked under distributed execution; diagnostic metrics (accuracy, sensitivity, specificity, F1, AUC) and systems metrics (runtime, speedup, parallel efficiency) are jointly analyzed.

4. **Scalability and efficiency characterization:** A detailed scaling study demonstrates near-linear and super-linear speedups with parallel efficiencies exceeding 100%, related to cache behavior and reduced I/O contention.

5. **High-throughput multi-class diagnostic performance:** Accuracy up to 97.63% for CXR three-class and 95.49% for CT binary classification, achieved with training times substantially reduced compared to single-node baselines.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 describes the datasets and computing infrastructure. Section 4 details the distributed training pipeline. Section 5 presents experimental results and scalability analysis. Section 6 concludes the paper.

2 Related Work

The convergence of deep learning with distributed computing has created significant opportunities for scalable medical image analysis. We review prior work through two complementary lenses—medical classification contributions and distributed infrastructure contributions—before synthesizing remaining gaps.

Medical Classification Contributions: From a medical imaging standpoint, prior work has predominantly focused on binary COVID-19 or pneumonia detection using CXR and CT. Awan et al. [2] achieved 100% binary and 98.55% three-class accuracy using InceptionV3, ResNet50, and VGG19 on small CXR datasets (708–1,063 images). Benbrahim et al. [3] reported 99.01% and 98.03% accuracy on 320 CXR images, while their subsequent CT study [4] achieved lower performance (82.87%–85.64% using VGG16, VGG19, and Xception), highlighting modality-specific challenges. Mezzoudj et al. [15] attained 91.88% accuracy on 13,808 CXR images using custom DCNNs, later improving to 95.53% on 18,000 augmented images [16]. Ravikumar and Sriraman [19] reported 98.72% accuracy on 5,863 CXR images with custom architectures. Roldán et al. [20] achieved 44.9%–55.0% accuracy with notable performance instability. Despite these results, medical limitations persist: most studies are binary, multi-class differentiation remains underexplored, cross-modality evaluation is absent, and dataset sizes are modest.

Distributed Infrastructure Contributions: From a distributed computing perspective, Apache Spark is the predominant framework. Awan et al. [2] and Benbrahim et al. [3, 4] established Spark-based feasibility on Databricks but provided limited scalability characterization. Mezzoudj et al. [15, 16] reported training times (5h 28min, 18,909s) using barrier execution and Spark–TensorFlow Distributor on single-node setups. Ravikumar and Sriraman [19] compared synchronous (1.5× speedup) and asynchronous (0.6× speedup) training on GPU using Elephas, though software configurations were under-specified. Roldán et al. [20] highlighted training stability challenges in pseudo-distributed environments. Infrastructure gaps include: limited scalability characterization on multi-node hardware, under-specified software environments hindering reproducibility, scarce comparative analysis of training strategies under identical conditions, and predominantly single-node evaluations.

Gap Synthesis and Our Contribution: Synthesizing both perspectives, prior work establishes important foundations but reveals three fundamental gaps: (1) modality-specific implementations lacking validation across both CXR and CT; (2) restriction to binary classification, limiting applicability to multi-class clinical scenarios; and (3) insufficient comparative analysis of synchronous versus asynchronous training with established pre-trained architectures on real multi-node distributed hardware. Our work addresses all three gaps simultaneously through an end-to-end Spark–Elephas pipeline supporting multi-class classification across both CXR and CT, benchmarked on large public datasets with explicit scalability characterization and comparative training strategy analysis, as summarized in Table 1.

Table 1: Comparison of Distributed Deep Learning Approaches for Pulmonary Disease Imaging

Reference	Architecture	Framework	Task	Infrastructure	Training
Awan et al. (2021) [2]	InceptionV3, ResNet50, VGG19	Spark + DL Pipelines	Binary & 3-class	Databricks	Synchronous
Benbrahim et al. (2020) [3]	InceptionV3, ResNet50	Spark + DL Pipelines	Binary	Databricks	Synchronous
Benbrahim et al. (2021) [4]	VGG16, VGG19, Xception	Spark + Keras TF	Binary	Databricks	Synchronous
Mezzoudj et al. (2023) [15]	Custom DCNN	Spark + TF Distributor	Binary	Single-node	Synchronous
Ravikumar et al. (2023) [19]	Custom CNN	Spark + Elephas	Binary	Google Colab	Both
Roldán et al. (2020) [20]	CNN	Spark + Dist-Keras	Binary	Single-node	Asynchronous
Mezzoudj et al. (2025) [16]	Custom DCNN	Spark + TF Distributor	Binary	Single-node	Synchronous
Our Work	DenseNet169, ResNet50, MobileNetV2	Spark + Elephas	Binary & Multi-class	Multi-node GPU	**Both**

3 Materials and Experimental Setup

3.1 Image Datasets

We utilize two publicly available medical imaging collections for cross-modality lung disease evaluation, yielding three classification configurations.

Chest X-ray Dataset. We use the COVID-19 Radiography Database [18], which contains carefully curated CXR images with characteristic radiographic patterns including bilateral peripheral ground-glass opacities, consolidations, and vascular enlargement within affected regions, predominantly affecting the lower lung zones with multifocal distribution [28]. This database supports two classification configurations: a three-class task (COVID-19, Normal, Viral Pneumonia) and a four-class task (COVID-19, Lung Opacity, Normal, Viral Pneumonia).

CT Dataset. We employ the SARS-CoV-2 CT-Scan Dataset [25], comprising CT images showing

typical pulmonary patterns including bilateral ground-glass opacities (GGOs), consolidations, crazy-paving pattern, and reverse halo sign. These findings evolve temporally from early GGOs to progressive consolidations and eventual resolution [24].

Table 2: Summary of Medical Imaging Datasets

Modality	Task	Classes	Total Images
CXR	3-class	COVID-19, Normal, Viral Pneumonia	15,153
CXR	4-class	COVID-19, Lung Opacity, Normal, Viral Pneumonia	21,165
CT	2-class	COVID-19, Non-COVID	2,481

Figure 1: Representative images: CT 2-class (left) and CXR 3-class (right)

3.2 Dataset Split Protocol

To ensure rigorous and unbiased evaluation, all datasets were partitioned using a stratified 80/10/10 split applied prior to any augmentation, with stratified sampling preserving class distributions across subsets—particularly critical given the class imbalance in the CXR 4-class configuration. Critically, splits were performed at the patient level (or study level for CT volumes), not at the image or file level: patient identifiers ensured that all images from the same patient were assigned exclusively to a single subset, and each CT volume was assigned in its entirety to one split. Near-duplicate images were also removed prior to splitting using perceptual hashing. This patient-level protocol prevents data leakage, ensuring that test performance reflects generalization to unseen patients rather than memorization of patient-specific features or dataset artifacts. The 10% validation set was used solely for model selection and early stopping, while the 10% test set served as an unbiased benchmark for final evaluation. All reported evaluation metrics values were computed exclusively on this held-out test set.

3.3 Distributed Computing Infrastructure

3.3.1 Cluster Architecture and Hardware

We deploy a multi-master cluster with high availability configuration optimized for distributed deep learning workloads:

- Cluster Topology: 3 Master Nodes (fault-tolerant cluster management), 4 Worker Nodes (distributed computations), and 1 Edge Node (job submission access point).
- Hardware Resources: Each node is equipped with 12 CPU cores, 32 GB RAM, and NVIDIA RTX 4060 GPUs (8 GB GDDR6) across worker nodes.
- Network Infrastructure: High-speed interconnects for efficient gradient synchronization with minimal communication overhead.

3.3.2 Distributed Training Framework

We operate a comprehensive big data ecosystem powered by ZettaSpark, with the following components:

- **Distributed Processing:** Apache Spark 3.5.0 [29, 30] for distributed data processing and model training via resilient distributed datasets (RDDs). Configuration: executor memory 16 GB, 4 executor cores, driver memory 8 GB, and 4 partitions per worker.
- **Storage Management:** Hadoop HDFS 3.3.6 [23] with replication factor 3 for data reliability and fault tolerance across multiple modalities. Rack-aware data placement was enabled to minimize network transfer during training.
- **Distributed Learning:** Elephas 6.2.1 [7] for synchronous and asynchronous distributed training integration, extending TensorFlow 2.13.0 and Keras 2.13.0 to enable distributed deep learning on Spark clusters. Configuration: parameter server with gossip-based synchronization, top-k gradient compression ($k = 10\%$), and prefetched batch buffering (queue size 4).
- **Resource Management:** Apache YARN 3.3.6 [27] for dynamic resource allocation with capacity scheduler and dedicated queues for deep learning workloads, ensuring resource isolation and optimized utilization across parallel training tasks.
- **GPU Acceleration:** CUDA 11.8 with cuDNN 8.6.0 and NVIDIA Driver 525.60.13, enabling mixed precision training (FP16 computation with FP32 master weights) across multiple NVIDIA RTX 4060 GPUs.
- **Deep Learning Framework:** TensorFlow 2.13.0 with Keras 2.13.0 as the primary deep learning backend, providing the core neural network implementation and training loop functionality.

4 End-to-End Distributed Training Pipeline

4.1 Pipeline Initialization

The pipeline begins with an initialization phase establishing the distributed computing environment and loading the medical imaging datasets. This phase involves: Spark context creation with optimized configurations for distributed deep learning; data loading from HDFS with proper stratified partitioning; cluster configuration with YARN resource management; and parameter server initialization for gradient aggregation and weight synchronization.

Our framework implements a master-worker topology with parameter server architecture specifically engineered for large-scale medical image analysis across multiple imaging modalities. This design integrates distributed data processing with parallelized deep learning training across a cluster of GPU-equipped nodes. The parameter server manages global model parameters W_t, coordinates weight synchronization, and handles gradient aggregation from all worker nodes. Each worker node contains a model replica, a data shard with partitioned medical images, and performs local gradient computation $\nabla f(x_i)$. Synchronous weight updates follow:

$$W_{t+1} = \text{UPDATE}\left(W_t,\, \lambda,\, \frac{1}{N}\sum_{i=1}^{N} \nabla f(x_i)\right) \tag{1}$$

where W_t represents weights at iteration t, λ is the learning rate, and $\nabla f(x_i)$ are gradients from worker i.

4.2 Data Preprocessing

We implement a comprehensive preprocessing pipeline for both CXR and CT modalities to ensure data quality and enhance model generalization:

Normalization: All images are resized to 224×224 pixels and normalized to the $[0, 1]$ range:

$$I_{\text{normalized}} = \frac{I_{\text{original}}}{255.0} \tag{2}$$

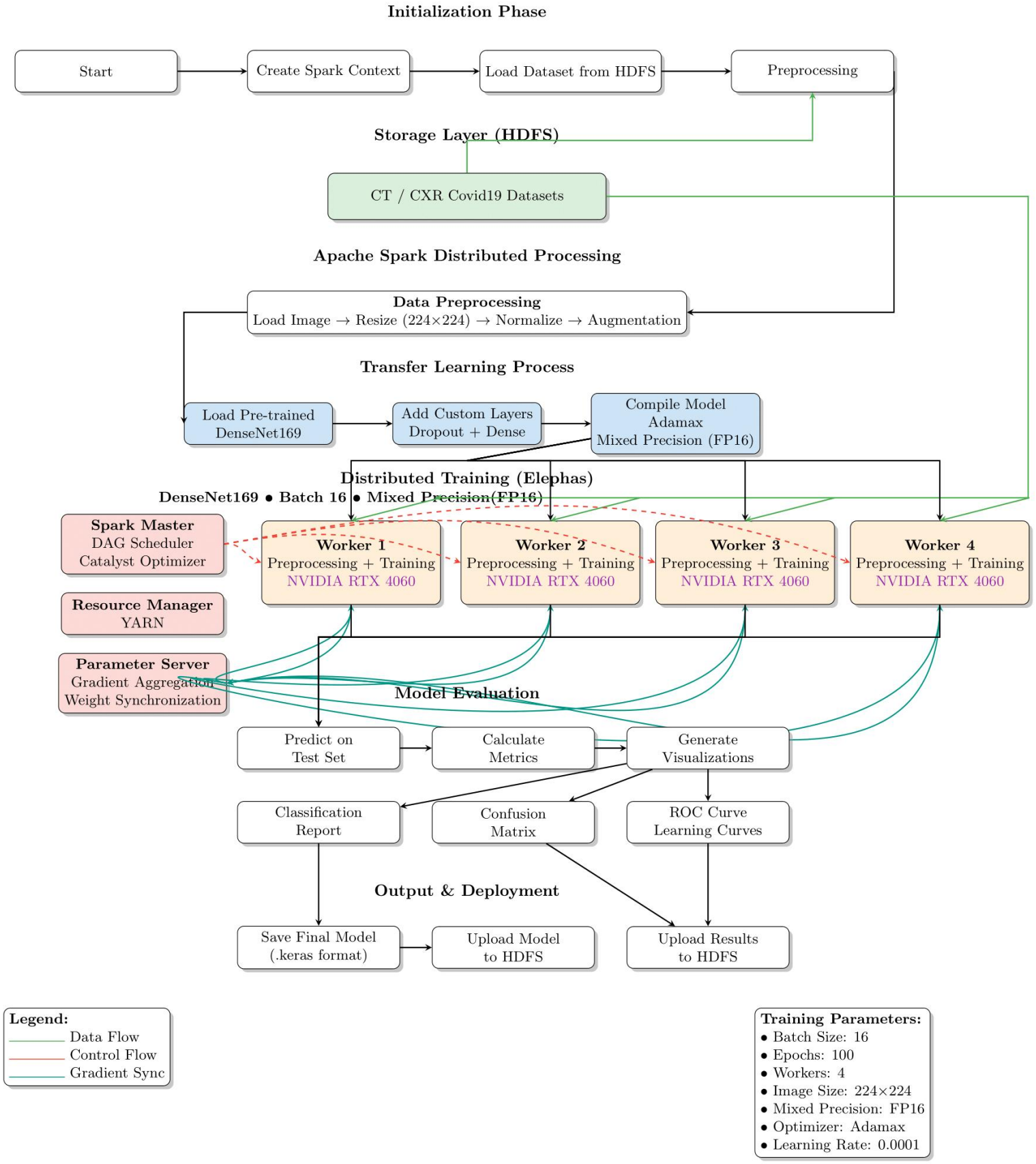


Figure 2: Spark-based distributed training pipeline for multi-class lung disease classification (CXR/CT)

Data Augmentation: To enhance model generalization while preserving anatomical validity, we apply a conservative augmentation strategy on-the-fly during distributed training, consisting primarily of random horizontal flipping ($p = 0.5$):

$$I_{\text{augmented}} = T_{\text{horizontal-flip}}(I_{\text{original}}) \tag{3}$$

Horizontal flipping is clinically justified for pulmonary classification, as the pathologies considered—opacities, consolidations, and infiltrates—are bilaterally independent and not contingent on left-right orientation, unlike cardiac or localized nodule tasks. The pretrained backbones were also trained with this augmentation on ImageNet, and CT axial slices are routinely flipped during standard preprocessing. Augmentations are applied exclusively to the training set.

Imbalanced Data Handling: We address class imbalance through weighted loss functions, computing class weights w_c inversely proportional to class frequencies [5]:

$$w_c = \frac{N_{\text{total}}}{C \times N_c} \tag{4}$$

where N_{total} is total samples, C is number of classes, and N_c is samples in class c.

4.3 Deep Learning Architectures

4.3.1 Transfer Learning Models

We employ three pre-trained CNN architectures—DenseNet169, ResNet50, and MobileNetV2—using ImageNet pre-trained weights to leverage learned feature representations from natural images. This transfer learning approach benefits medical imaging tasks where dataset sizes are often limited [9, 10, 22].

DenseNet169 Architecture We select DenseNet169 for its dense connectivity patterns where each layer receives feature maps from all preceding layers, promoting feature reuse and mitigating vanishing gradient problems[10]. With 169 layers and approximately 14.3 million parameters, DenseNet169 achieves efficient feature extraction while maintaining computational efficiency through dense connections that reduce parameter redundancy.

ResNet50 Architecture We utilize ResNet50 for its residual learning with skip connections that address the degradation problem in deep networks[9]. The 50-layer architecture with 25.6 million parameters enables training of very deep networks through identity mapping, which preserves gradient flow during backpropagation and prevents performance saturation.

MobileNetV2 Architecture We include MobileNetV2 for its efficient depthwise separable convolutions and inverted residuals. The architecture uses bottleneck layers and employs two types of blocks. With only 3.5 million parameters, MobileNetV2 provides a lightweight alternative suitable for resource-constrained environments while maintaining competitive performance through efficient depthwise separable convolutions [22].

4.3.2 Unified Implementation Strategy

A consistent implementation strategy is applied across all architectures:

- **ImageNet Pre-trained Weights and Fine-tuning Strategy:** All models are initialized with ImageNet weights. A progressive fine-tuning strategy is employed: early convolutional layers are initially frozen to preserve general feature detectors, with layers progressively unfrozen from top to bottom during training (unfreezing every 20 epochs) to adapt domain-specific representations.
- **Custom Classification Head:** Each architecture is augmented with Global Average Pooling, Dropout, Batch Normalization [11], a Dense layer with ReLU activation, and a final softmax output layer. Dropout rates are model-specific: 0.4 for DenseNet169, 0.5 for ResNet50, and 0.3 for MobileNetV2.

- **Loss Function:** Class-weighted categorical cross-entropy [5]:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} w_c \cdot y_{i,c} \cdot \log(\hat{y}_{i,c}) \tag{5}$$

  where w_c are class weights computed via inverse frequency weighting for each task, $y_{i,c}$ is the ground truth, and $\hat{y}_{i,c}$ is the predicted probability.

- **Training Configuration:** Adamax optimizer [12] with adaptive learning rate (10^{-4} to 5×10^{-5}), L2 weight decay ($\lambda = 10^{-4}$ to 10^{-5}), and label smoothing regularization. A ReduceLROnPlateau scheduler reduces the learning rate by a factor of 0.5 when validation loss plateaus for 3 epochs, with a minimum learning rate of 5×10^{-5}. Training proceeds for a maximum of 100 epochs with early stopping (patience=5) based on validation loss.

- **Batch Size:** Per-worker batch size is 16, yielding a global batch size of $16 \times N_{\text{workers}}$ (i.e., 64 for 4 workers).

- **Mixed Precision Training:** FP16 computation with FP32 master weights [17] reduces memory requirements by approximately 40% while maintaining numerical stability, particularly crucial for distributed training across multiple GPU workers.

4.4 Distributed Training Framework

4.4.1 Distributed Training Algorithm

Algorithm 1 formalizes our synchronous data-parallel training with a parameter-server architecture implemented via Elephas on Spark.

The gradient aggregation at the parameter server follows:

$$\nabla F = \frac{1}{N}\sum_{i=1}^{N} \nabla f_i \tag{6}$$

where N is the number of workers. The asynchronous variant allows immediate parameter updates without synchronization barriers, trading potential training instability for reduced communication overhead.

4.4.2 Data Parallelism Strategy

We implement data parallelism by replicating model architecture across all workers, each training on a distinct stratified data partition. The synchronous update rule [6, 13] is:

$$\theta_{t+1} = \theta_t - \eta \cdot \frac{1}{N}\sum_{i=1}^{N} \nabla L(\theta_t; \mathcal{B}_i) \tag{7}$$

where θ_t are model parameters at iteration t, η is the learning rate, and $\nabla L(\theta_t; \mathcal{B}_i)$ denotes gradients on mini-batch $\mathcal{B}_i$ at worker i. Key advantages of the synchronous approach include deterministic updates, training stability through consistent gradient directions, and reproducibility with fixed random seeds.

4.4.3 Memory and Computational Optimizations

To maximize resource utilization, we implement: Mixed Precision Training (FP16/FP32) [17] reducing memory usage by approximately 40%; stratified Smart Data Partitioning maintaining class balance across workers; Gradient Compression via selective gradient quantization [1, 14]; and Memory-Aware Batch Loading with dynamic batch sizing based on available GPU memory.

Algorithm 1 Distributed Training of Proposed Framework

At Master Node:
Create Spark session and initialize Parameter Server
Load datasets (CXR + CT) from HDFS
Distribute stratified data partitions to worker nodes
Initialize global model W_0 and broadcast to workers
Main Training Loop:
for each epoch $t = 1$ to T **do**
    Broadcast current parameters W_t to all workers
    for each worker $i = 1$ to N **in parallel do**
        Load data partition D_i
        Compute gradients $\nabla f_i = \nabla L(W_t; D_i)$
        Send ∇f_i to parameter server
    end for
    Aggregate: $\nabla F = \frac{1}{N} \sum_{i=1}^{N} \nabla f_i$
    Update: $W_{t+1} = W_t - \lambda \nabla F$
    Synchronize all workers with W_{t+1}
end for
Save final model and evaluate on held-out test set
At Worker Nodes:
Receive initial parameters W_0 from master
Load distributed data partition
for each training iteration **do**
    Receive current parameters W_t
    Compute forward and backward pass
    Send local gradients ∇f_i to parameter server
    Receive updated parameters W_{t+1}
end for

4.4.4 Fault Tolerance and Recovery

Robust fault tolerance is incorporated through: automatic model checkpointing at optimal validation performance; automatic task reallocation for failed worker nodes without complete job restart; HDFS-based data storage [23] with replication factor 3; and real-time monitoring of training metrics with centralized logging across all worker nodes.

5 Experimental Results

5.1 Evaluation Metrics

The proposed framework was evaluated using both diagnostic and computational performance metrics. Diagnostic performance was assessed using accuracy, sensitivity (recall), specificity, precision, F1-score, and the area under the ROC curve (AUC-ROC), following the standard definitions in [8, 26]. For the multi-class CXR tasks, weighted averages were computed to account for class imbalance. Computational scalability was evaluated using Speedup, Efficiency, and Scaling Efficiency, defined as

$$\text{Speedup} = \frac{\text{Single-node Training Time}}{\text{Distributed Training Time}}, \tag{8}$$

$$\text{Efficiency} = \frac{\text{Speedup}}{\text{Number of Workers}} \times 100\%, \tag{9}$$

$$\text{Scaling Efficiency} = \frac{\text{Observed Speedup}}{\text{Theoretical Speedup}} \times 100\%. \tag{10}$$

5.2 Diagnostic Performance

5.2.1 Architectural Comparison

Table 3 presents the overall accuracy of the three CNN backbones across all classification tasks. DenseNet169 consistently achieves the highest accuracy across all datasets and tasks. On the CXR 3-class task it reaches 97.63%, outperforming ResNet50 and MobileNetV2 by substantial margins. This advantage persists in the more complex 4-class configuration and on the CT dataset, underscoring its robust feature extraction capability for pulmonary image analysis.

Table 3: Model Performance Across CXR and CT Classification Tasks (*DenseNet169, Synchronous Training*)

Architecture	Parameters	CXR 3-class	CXR 4-class	CT 2-class
DenseNet169	14.3M	**0.9763**	**0.9315**	**0.9549**
ResNet50	25.6M	0.9434	0.8885	0.9215
MobileNetV2	3.5M	0.9287	0.8642	0.9012

5.2.2 Synchronous vs. Asynchronous Training

Table 4 presents the accuracy training-time trade-off between synchronous and asynchronous training strategies across all datasets. Synchronous training consistently delivers higher accuracy across all datasets, with a 1.7–2.7 percentage point improvement over asynchronous training, albeit at slightly longer training times. This systematic advantage supports the choice of synchronous updates in safety-critical medical applications requiring diagnostic precision.

Table 4: Synchronous vs. Asynchronous Training Performance (*DenseNet169*)

Dataset	Strategy	Training Time	Accuracy	Conv. Epochs
CXR 3-class	Synchronous	8.7 min	**0.9763**	18
	Asynchronous	6.9 min	0.9511	22
CXR 4-class	Synchronous	9.34 min	**0.9315**	20
	Asynchronous	7.8 min	0.9050	25
CT 2-class	Synchronous	5.1 min	**0.9549**	16
	Asynchronous	4.1 min	0.9380	19

5.2.3 Per-Class Diagnostic Metrics

To provide a complete clinical evaluation, Tables 5, 6, and 7 report per-class sensitivity, specificity, precision, F1-score, and AUC for each classification task, computed on the held-out test set using the best-performing DenseNet169 model with synchronous training.

Table 5: Per-Class Performance Metrics for CXR 2-Class Classification

Class	Sensitivity	Specificity	Precision	F1-Score	AUC
Non-COVID	0.9836	0.9760	0.9756	0.9796	0.98
COVID-19	0.9760	0.9836	0.9839	0.9799	0.98
Weighted Avg.	**0.9797**	**0.9798**	**0.9798**	**0.9797**	**0.98**

Table 6: Per-Class Performance Metrics for CXR 3-Class Classification

Class	Sensitivity	Specificity	Precision	F1-Score	AUC
COVID-19	0.9641	0.9861	0.9562	0.9601	0.985
Normal	0.9804	0.9677	0.9843	0.9823	0.995
Viral Pneumonia	0.9776	0.9971	0.9704	0.9740	0.988
Weighted Avg.	**0.9764**	**0.9765**	**0.9763**	**0.9763**	**0.989**

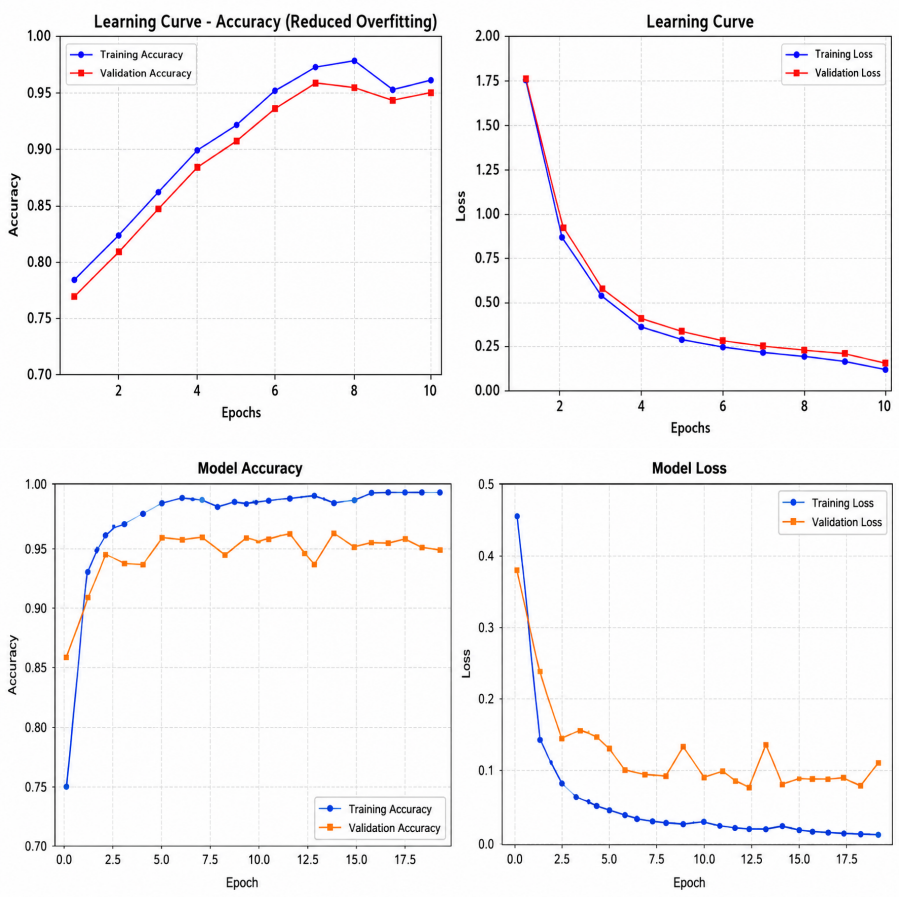
Table 7: Per-Class Performance Metrics for CXR 4-Class Classification

Class	Sensitivity	Specificity	Precision	F1-Score	AUC
COVID-19	0.9420	0.9863	0.9342	0.9381	1.000
Lung Opacity	0.8987	0.9716	0.9264	0.9123	0.980
Normal	0.9421	0.9390	0.9348	0.9384	0.980
Viral Pneumonia	0.9701	0.9945	0.9220	0.9455	1.000
Weighted Avg.	**0.9316**	**0.9666**	**0.9307**	**0.9308**	**0.990**

The CT 2-class task achieves near-perfect and balanced performance for both classes, with sensitivity and specificity exceeding 0.97 for all categories. In the CXR 3-class task, all three classes achieve F1-scores above 0.96, with Viral Pneumonia attaining the highest specificity (0.9971) and Normal the highest AUC (0.995). In the CXR 4-class task, Lung Opacity presents the greatest challenge, with sensitivity of 0.8987 and F1 of 0.9123, reflecting the high visual similarity between Lung Opacity and Normal cases. COVID-19 and Viral Pneumonia achieve AUC = 1.000, indicating perfect class separability for these categories.

5.2.4 Convergence and Discriminative power

Learning curves (Figure 3) show stable convergence with minimal overfitting and close alignment between training and validation metrics across all tasks, indicating excellent generalization on unseen data. ROC curves (Figure 4) confirm strong discriminative capability, with AUC values of 0.985–0.995 for the CXR 3-class task and near-perfect discrimination for the remaining configurations. Per-class metrics in Tables 5–7 further confirm well-balanced performance, particularly the 96.4% sensitivity for COVID-19 detection in the CXR 3-class task.



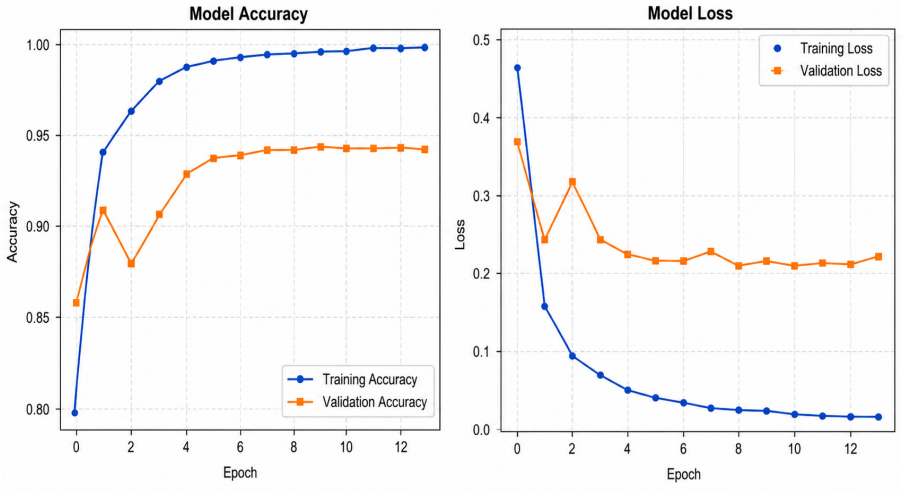


Figure 3: Learning curves: CT 2-class (top), CXR 3-class (center), and CXR 4-class (bottom), showing stable convergence with minimal overfitting across all tasks

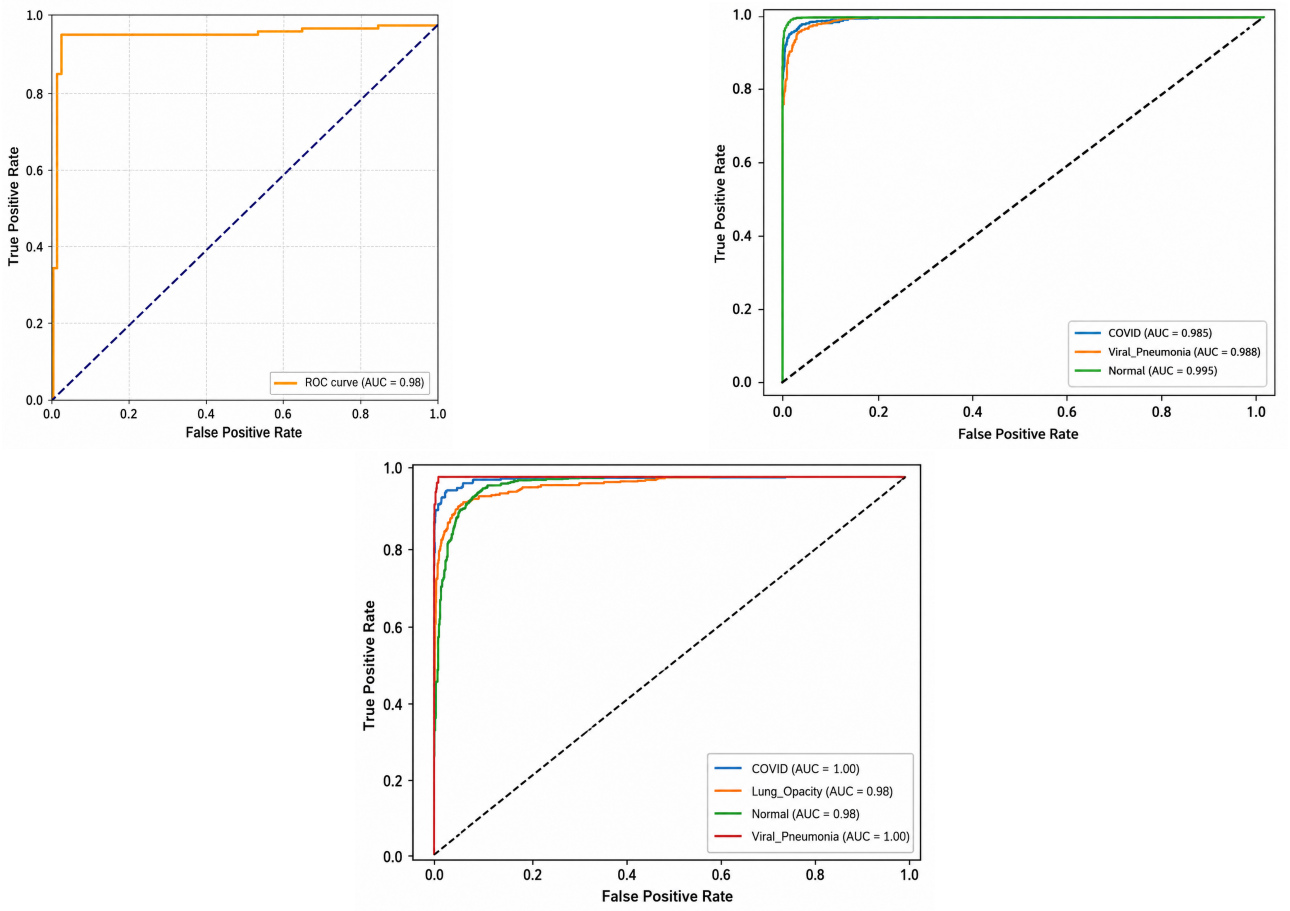


Figure 4: ROC curves: CT 2-class (AUC = 0.98, left); CXR 3-class (COVID: 0.985, Viral Pneumonia: 0.988, Normal: 0.995, center); CXR 4-class (weighted AUC = 0.998, right)

5.3 Computational Efficiency and Scalability

Distributed Speedup and Efficiency :Table 8 quantifies the core benefit of distributed execution across all dataset configurations. The framework achieves significant speedups across all configurations, most notably a 6.26× acceleration on the CXR 4-class dataset, reducing training time from 58.48 to 9.34 minutes and demonstrating a super-linear scaling efficiency of 156.5%.

Table 8: Distributed Training Efficiency Across Cluster Configurations

Dataset	Configuration	Training Time	Speedup	Efficiency
CXR 3-class	Single Node	38.40 min	1.0×	100%
	4 Workers	8.7 min	4.41×	110.3%
CXR 4-class	Single Node	58.48 min	1.0×	100%
	4 Workers	9.34 min	6.26×	156.5%
CT 2-class	Single Node	22.40 min	1.0×	100%
	4 Workers	5.1 min	4.39×	109.8%

Scaling Characteristics: Figure 5 illustrates the near-linear to super-linear speedup achieved by our framework as nodes are added. The super-linear scaling (efficiency $> 100\%$) arises from optimized resource utilization: larger datasets benefit from increased CPU cache residency and reduced I/O contention as the cluster grows. The framework maintains high efficiency through: optimized data partitioning minimizing communication overhead; efficient gradient aggregation at the parameter server; overlapping computation and communication phases; and memory-aware data loading strategies.

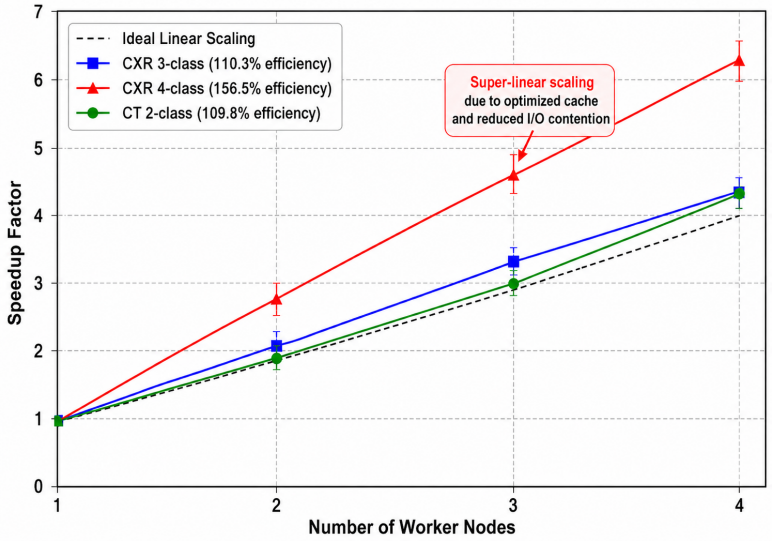


Figure 5: Scaling efficiency across different worker configurations showing near-linear speedup for medical imaging workloads

Comparative Analysis: Compared with previous distributed implementations, our framework achieves substantially higher computational scalability, reaching up to 6.26× speedup compared with the 1.5× synchronous speedup reported by Ravikumar et al. [19]. In addition, class weighting improves minority-class recognition, increasing the F1-score of the *Lung Opacity* class by 4.33–5.23%.

As summarized in Table 9, existing studies differ considerably in datasets, class definitions, preprocessing, and evaluation protocols; therefore, the reported accuracies are presented for qualitative context rather than direct comparison. Our primary contribution lies in demonstrating a scalable distributed framework for multi-class CXR (3- and 4-class) and CT classification on the largest datasets considered, highlighting its practical applicability for large-scale medical image analysis.

Table 9: Performance Comparison with Existing Distributed Approaches

Study	Architecture	Mod.	Cls.	Images	Accuracy
Benbrahim (2020) [3]	InceptionV3, ResNet50	CXR	2	320	99.01%
Awan (2021) [2]	InceptionV3, ResNet50, VGG19	CXR	2	708	100%
Awan (2021) [2]	InceptionV3, ResNet50, VGG19	CXR	3	1,063	97–98.55%
Benbrahim (2021) [4]	VGG16, VGG19, Xception	CT	2	746	82.87–85.64%
Mezzoudj (2023) [15]	Custom DCNN	CXR	2	13,808	91.88%
Mezzoudj (2025) [16]	Custom DCNN	CXR	2	18,000	95.53%
Roldán (2020) [20]	CNN (Dist-Keras)	CXR	2	3,166	44.9–55.0%
Ravikumar (2023) [19]	Custom CNN	CXR	2	5,863	98.72%
Our Work	DenseNet169 DenseNet169 DenseNet169	CXR CXR CT	3 4 2	**15,153** **21,165** **2,481**	**97.63%** **93.15%** **95.49%**

5.4 Discussion

Performance Analysis :DenseNet169 delivers strong and consistent performance across all datasets: 97.63% for CXR 3-class, 93.15% for CXR 4-class, and 95.49% for CT binary classification. Compared with existing distributed methods, our framework scales to substantially larger datasets (15,153–21,165 images) while maintaining competitive or superior accuracy. Although Awan et al. [2] report up to 100% and 98.55% accuracy, their experiments are restricted to much smaller datasets (708 and 1,063 images), underscoring the stronger scalability and generalization of our approach. Relative to Benbrahim et al. [4], who achieved at most 85.64% accuracy on CT classification, our results highlight the benefits of the optimized DenseNet169 architecture and our synchronous training strategy. Our results also substantially exceed those of Roldán et al. [20], whose accuracies ranged from 44.9% to 55.0% with unstable distributed behavior.

Per-class metrics reveal that COVID-19 is correctly identified with 96.4% sensitivity and 98.6% specificity in the CXR 3-class task, demonstrating strong clinical relevance. Viral Pneumonia achieves the highest specificity (99.7%), virtually eliminating false positives. In the CT binary task, both classes exceed 97.6% sensitivity and specificity, confirming reliable cross-modality performance. Across all datasets, synchronous training consistently outperforms asynchronous training by 1.7–2.7 percentage points in accuracy, supporting the choice of synchronous updates in safety-critical applications.

A performance limitation was observed in the CXR 4-class task, where accuracy decreased from 97.63% (3-class) to 93.15% due to increased inter-class similarity. The confusion matrix indicates that *Lung Opacity* is mainly misclassified as *Normal* (49 cases), with substantially fewer errors toward *COVID-19* (12 cases) and none toward *Viral Pneumonia*. These results suggest that the model effectively captures COVID-19-specific radiographic features, while the remaining errors reflect the inherent difficulty of distinguishing subtle pulmonary opacities from normal chest radiographs.

Scalability and Efficiency Analysis: The proposed distributed framework achieved super-linear scaling, reaching an efficiency of 156.5% on the CXR 4-class dataset. To explain this behaviour, we analysed the underlying system telemetry together with the memory hierarchy and I/O characteristics of both the single-node and distributed executions.

During single-node training, the working set of DenseNet169, including model parameters, intermediate feature maps, and high-resolution image tensors, exceeded the effective capacities of the CPU cache hierarchy (L2/L3) and GPU memory caches. Consequently, frequent cache misses forced re-

peated accesses to main memory, increasing memory latency and reducing computational throughput. In the distributed Spark configuration, the dataset was partitioned into smaller subsets ($\mathcal{D}_i$), substantially reducing the working set processed by each worker. The reduced memory footprint fitted more efficiently within the 8 GB GDDR6 memory of each NVIDIA RTX 4060 GPU and the processor cache hierarchy, thereby improving memory locality, reducing cache misses, and shifting execution from a memory-bound to a compute-bound regime.

System telemetry further revealed that the baseline single-node execution was strongly I/O-bound. Loading the complete dataset of 21,165 chest X-ray images on demand saturated the local storage bandwidth (approximately 110 MB/s), limiting GPU utilization to only 38% because the accelerator frequently stalled while waiting for data. In contrast, under the four-worker Spark configuration, each worker processed only its local data partition, which remained resident in the operating system page cache after the first training epoch. Consequently, subsequent epochs accessed images directly from memory rather than disk, effectively eliminating most disk I/O latency. This cache-resident data pipeline, combined with parallel data streaming through Hadoop HDFS, increased average GPU utilization to approximately 88%, allowing computation and data delivery to proceed concurrently with minimal idle time.

These complementary effects—improved memory locality, elimination of disk I/O bottlenecks, increased GPU utilization, and parallel data ingestion—collectively explain the observed 6.26× reduction in training time and the measured super-linear scaling efficiency of 156.5%. Rather than resulting solely from parallel computation, the performance gain arises from simultaneously alleviating memory and storage bottlenecks that constrained the single-node implementation.

Clinical Implications : Beyond diagnostic accuracy, the architecture of the proposed distributed framework directly addresses the operational bottlenecks of deploying deep learning within complex hospital environments. Traditional single-node models struggle to handle the continuous, high-volume influx of clinical imaging data generated across multiple departments. By leveraging Apache Spark's streaming capabilities and resilient distributed datasets (RDDs), this framework can be seamlessly integrated into existing hospital infrastructures, such as Picture Archiving and Communication Systems (PACS). Incoming chest X-rays and CT scans can be parallelized on-the-fly via localized GPU-enabled nodes, minimizing diagnostic latency without disrupting daily clinical workflows. Furthermore, the framework's near-linear scalability makes it an ideal backbone for institutional or regional Clinical Decision Support Systems (CDSS). By handling multi-modal, large-scale imaging streams concurrently, the system can support real-time triaging of acute pulmonary conditions, enabling healthcare facilities to dynamically scale their computational resources as clinical demand fluctuates

Limitations and Future Work: Despite the promising results reported, several limitations warrant acknowledgment. The proposed system is designed as a decision-support tool rather than a replacement for clinical diagnosis; prospective clinical trials would be necessary before any real-world deployment. Interpretability also remains an open challenge, as the current framework does not incorporate explainability methods such as Grad-CAM or Score-CAM. This is particularly relevant given the high AUC values observed and the well-documented risk of shortcut learning on public medical imaging datasets, where models may exploit spurious correlations—including acquisition artifacts, text labels, or image borders—rather than learning clinically meaningful radiographic features.

The evaluation presented herein relies exclusively on public datasets, with performance reported on held-out subsets of the same sources. While this constitutes an acceptable experimental first step, it does not adequately support claims of clinical generalizability. Domain shift arising from hospital-specific acquisition protocols, scanner variability, demographic differences, and image preprocessing pipelines may substantially affect real-world performance. External validation on independent, multi-center clinical cohorts therefore remains essential to establish robustness. Furthermore, the study is limited to three conventional CNN architectures; more recent paradigms such as Vision Transformers or foundation models have not been explored. The framework has also been demonstrated on CXR and CT only, leaving its applicability to other modalities—such as MRI or ultrasound—unverified.

Future work will focus on integrating explainability tools to facilitate clinical interpretability and model auditing, and on enhancing discrimination between visually similar classes through contrastive learning and attention-based mechanisms. Extension of the framework to transformer-based architectures, multi-label classification, and 3D CT volume processing is also planned. Rigorous external

validation on multi-center, prospectively collected clinical datasets will be pursued to assess generalizability across diverse populations, scanners, and acquisition protocols. Finally, applicability to broader imaging modalities and disease domains remains an important direction for future investigation. Collectively, these efforts aim to strengthen the robustness, interpretability, and clinical translatability of distributed deep learning frameworks for medical imaging.

6 Conclusion

This work presented a unified Spark–Elephas distributed framework for scalable multi-class lung disease diagnosis from large-scale CXR and CT imaging datasets, spanning four pulmonary pathology categories and multiple classification granularities. By integrating Keras CNN models with a parameter-server architecture on a GPU-enabled Spark cluster, we demonstrated that high diagnostic performance can be achieved without sacrificing scalability or throughput. DenseNet169 delivered the strongest results across all tasks—up to 97.63% accuracy for CXR three-class and 95.49% for CT binary classification—with per-class AUC values from 0.98 to 1.00. Synchronous parameter updates consistently outperformed asynchronous training by 1.7–2.7 percentage points across all datasets, with more stable convergence. The framework achieved near-linear and super-linear speedups (up to 6.26×, parallel efficiency 156.5%), attributable to improved cache utilization and reduced I/O contention. These results confirm that cluster computing platforms can support high-throughput lung disease diagnosis at scales beyond prior distributed medical imaging studies. Future work will address the Normal/Lung Opacity separability challenge through advanced representation learning (contrastive learning, attention mechanisms), explore federated or hybrid cloud–edge deployment, and extend the framework to broader thoracic and multi-organ imaging tasks.

Ethical and Regulatory Compliance

All datasets are publicly available and were used in compliance with their licenses (e.g., CC BY 4.0). They were fully de-identified by their providers in accordance with HIPAA and GDPR. As this study involves secondary analysis of anonymized public data, IRB approval was not required.

Code availability

Source code and implementation of the distributed deep learning framework are available from the author upon request, including Spark configurations, model architectures, and training scripts for reproducibility.

Acknowledgements

The authors gratefully acknowledge the technical collaboration with ZettaSpark, computational resources from the Higher Institute of Computer Science and Multimedia of Gabès, and the open-source community for their tools and datasets.

Funding

This research was funded by Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2026R125), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia.

Conflict of Interest

The authors declare no known competing financial interests or personal relationships that could have influenced the work reported in this paper.

Author Contributions

Zeineb BEN MESSAOUD: Methodology, Validation, Investigation, Data Curation,Writing, Review & Editing.
Maher Baccar: Conceptualization, Methodology, Software, Validation, Formal Analysis, Investigation, Data Curation, Writing, Visualization.
Heni Bouhamed: Formal Analysis, Supervision, Validation, Review.
Monia Hamdi: Supervision, Project Administration, Resources, Review.

References

[1] Alistarh, D.; Grubic, D.; Li, J.; Tomioka, R.; Vojnovic, M. (2017). QSGD: Communication-efficient SGD via gradient quantization and encoding, *Advances in Neural Information Processing Systems*, 30, 1709–1720.

[2] Awan, M.J.; Bilal, M.H.; Yasin, A.; Nobanee, H.; Khan, N.S.; Zain, A.M. (2021). Detection of COVID-19 in chest X-ray images: A big data enabled deep learning approach, *International Journal of Environmental Research and Public Health*, 18(19), 10147.

[3] Benbrahim, H.; Hachimi, H.; Amine, A. (2020). Deep transfer learning with Apache Spark to detect COVID-19 in chest X-ray images, *Romanian Journal of Information Science and Technology*, 23, 117–129.

[4] Benbrahim, H.; Hachimi, H.; Amine, A. (2021). Deep transfer learning pipelines with Apache Spark and Keras TensorFlow combined with logistic regression to detect COVID-19 in chest CT images, *Walailak Journal of Science and Technology*, 18(11), 13109.

[5] Buda, M.; Maki, A.; Mazurowski, M.A. (2018). A systematic study of the class imbalance problem in convolutional neural networks, *Neural Networks*, 106, 249–259.

[6] Dean, J.; Corrado, G.; Monga, R.; et al. (2012). Large scale distributed deep networks, *Advances in Neural Information Processing Systems*, 25, 1223–1231.

[7] Elephas: Distributed Deep Learning with Keras & Spark. https://github.com/maxpumperla/elephas. Accessed: 2025-03-20.

[8] Fawcett, T. (2006). An introduction to ROC analysis, *Pattern Recognition Letters*, 27(8), 861–874.

[9] He, K.; Zhang, X.; Ren, S.; Sun, J. (2016). Deep residual learning for image recognition, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778.

[10] Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. (2017). Densely connected convolutional networks, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 4700–4708.

[11] Ioffe, S.; Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift, *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 448–456.

[12] Kingma, D.P.; Ba, J. (2014). Adam: A method for stochastic optimization, *arXiv preprint arXiv:1412.6980*.

[13] Li, M.; Andersen, D.G.; Park, J.W.; Smola, A.J.; et al. (2014). Scaling distributed machine learning with the parameter server, *Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation (OSDI)*, 583–598.

[14] Lin, Y.; Han, S.; Mao, H.; Wang, Y.; Dally, W.J. (2017). Deep gradient compression: Reducing the communication bandwidth for distributed training, *arXiv preprint arXiv:1712.01887*.

[15] Mezzoudj, S.; Belkessa, I.; Bouras, F.; Khelifa, M. (2023). A novel distributed deep learning approach for large-scale chest X-ray COVID-19 images detection, *Research Square*. https://doi.org/10.21203/rs.3.rs-2534755/v1

[16] Mezzoudj, S.; Khelifa, M.; Saadna, Y. (2025). Leveraging Spark–TensorFlow Distributor for distributed deep convolutional neural networks: Accelerating large-scale COVID-19 detection, *Informatica*, 49, 173–188.

[17] Micikevicius, P.; Narang, S.; Alben, J.; et al. (2017). Mixed precision training, *arXiv preprint arXiv:1710.03740*.

[18] Rahman, T.; et al. (2021). COVID-19 radiography database, *Kaggle*. https://www.kaggle.com/datasets/tawsifurrahman/covid19-radiography-database

[19] Ravikumar, A.; Sriraman, H. (2023). Real-time pneumonia prediction using pipelined Spark and high-performance computing, *PeerJ Computer Science*, 9, e1258.

[20] Roldán, A.; Sánchez-Solís, J.; Jiménez, V.; Juárez, R.; Zárate, G. (2020). Convolutional neural network in a pseudo-distributed environment for classification of chest X-ray images of patients with pneumonia, *Research in Computing Science*, 149, 101–110.

[21] Rubin, G.D.; Ryerson, C.J.; Haramati, L.B.; et al. (2020). The role of chest imaging in patient management during the COVID-19 pandemic: A multinational consensus statement from the Fleischner Society, *Radiology*, 296(1), 172–180.

[22] Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. (2018). MobileNetV2: Inverted residuals and linear bottlenecks, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 4510–4520.

[23] Shvachko, K.; Kuang, H.; Radia, S.; Chansler, R. (2010). The Hadoop distributed file system, *Proceedings of the IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, 1–10.

[24] Simpson, S.; et al. (2020). Radiological Society of North America expert consensus statement on reporting chest CT findings related to COVID-19, *Radiology: Cardiothoracic Imaging*, 2(2), e200152.

[25] Soares, E.; Angelov, P.; Biaso, S.; Froes, M.H.; Abe, D.K. (2020). SARS-CoV-2 CT-scan dataset: A large dataset of real patients CT scans for SARS-CoV-2 identification, *medRxiv*.https://www.kaggle.com/datasets/plameneduardo/sarscov2-ctscan-dataset

[26] Sokolova, M.; Japkowicz, N.; Szpakowicz, S. (2009). A systematic analysis of performance measures for classification tasks, *Information Processing & Management*, 45(4), 427–437.

[27] Vavilapalli, V.K.; Murthy, A.C.; Douglas, C.; et al. (2013). Apache Hadoop YARN: Yet another resource negotiator, *Proceedings of the 4th Annual Symposium on Cloud Computing (SOCC)*, 1–16.

[28] Wong, H.Y.F.; Lam, H.Y.S.; Fong, A.H.T.; et al. (2020). Frequency and distribution of chest radiographic findings in patients positive for COVID-19, *Radiology*, 296(2), E72–E78.

[29] Zaharia, M.; Chowdhury, M.; Franklin, M.J.; Shenker, S.; Stoica, I. (2010). Spark: Cluster computing with working sets, *Proceedings of the 2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud'10)*.

[30] Zaharia, M.; Chowdhury, M.; Das, T.; Dave, A.; Ma, J.; McCauly, M.; Franklin, M.J.; Shenker, S.; Stoica, I. (2012). Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing, *Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI'12)*.

Journal's webpage: http://univagora.ro/jour/index.php/ijccc/

This journal is a member of, and subscribes to the principles of,
the Committee on Publication Ethics (COPE).
https://publicationethics.org/members/
international-journal-computers-communications-and-control

Cite this paper as:

Ben Messaoud, Z.;Baccar, M.; Bouhamed, H.; Hamdi, M. (2026). Scalable Distributed Deep Learning for Lung Disease Diagnosis: A Spark–Elephas Cross-Modality Framework with GPU Cluster Scalability Analysis, *International Journal of Computers Communications & Control*, 21(5), 7566, 2026.

https://doi.org/10.15837/ijccc.2026.5.7566