Leveraging Spark-TensorFlow Distributor for Distributed Deep Convolutional Neural Networks: Accelerating large-scale COVID-19 Detection

Saliha Mezzoudj[1], Meriem Khelifa[2], Yasmina Saadna[3]
[1]Department of Computer Science, University of Algiers, Algiers, Algeria
[2]Artificial Intelligence of Information Technologies, Department of Computer Science and Information Technologies, University of Kasdi Merbah Ouargla, Algeria
[3]Labstic laboratory, Batna 2 University, Batna, Algeria
E-mail: s.mezzoudj@univ-alger.dz, khelifa.meriem@univ-ouargla.dz, y.saadna@univ-batna2.dz

Keywords: Chest X-ray, DCNN, spark-tensorflow-distributor, distributed training

Received: November 5, 2024

The COVID-19 epidemic has been a critical global challenge due to its high mortality rate and rapid spread. Initial diagnostic methods, such as chest X-rays and reverse transcriptase polymerase chain reaction (RT-PCR), can be time-consuming and require enhanced efficiency, especially in scenarios where large-scale testing is necessary. Combining deep learning (DL) and big data analytics tools is crucial for developing accurate and fast systems to detect COVID-19 infections and prevent further transmission. In this study, we introduce a distributed Deep Convolutional Neural Network (DCNN) framework designed for the detection of COVID-19, utilizing a comprehensive dataset of chest X-ray images. Our approach leverages the spark-tensorflow-distributor, which integrates TensorFlow 2.x with Apache Spark 3.x, enabling distributed training within the Spark framework. The system utilizes Spark's barrier mode to execute distributed training tasks, significantly enhancing the speed of both training and testing phases. We classify chest X-ray images into two categories: COVID-19 and non-COVID-19. The experimental setup involves processing the large-scale dataset of chest X-ray images in a parallelized manner using the Spark framework. Our results demonstrate that the proposed DCNN model achieves a classification accuracy of 95.53%, with a testing time of 0.06 seconds and a training time of 18,909 seconds. Furthermore, when compared to other state-of-the-art methods for COVID-19 classification, our approach proves to be more efficient and effective. This study highlights the potential of combining distributed computing frameworks with deep learning techniques to address largescale medical image analysis challenges, particularly in the context of COVID-19 detection.

Povzetek: Predlagan je pristop za hitro odkrivanje COVID-19 iz velikih zbirk rentgenskih slik prsnega koša z uporabo distribuiranega globokega konvolucijskega nevronskega omrežja (DCNN).

1 Introduction

In December 2019, COVID-19 (coronavirus) was identified in a group of pneumonia patients [13], [14], leading to its declaration as a global health emergency by the World Health Organization (WHO) [15]. The WHO emergency panel underscored the significance of rapid detection, isolation, early treatment, and the implementation of a robust contact tracing strategy to control the spread of COVID-19 [16]. One of the primary diagnostic methods for identifying the coronavirus is the examination of sputum samples through polymerase chain reaction (RT-PCR) [10]. However, RT-PCR is a time-consuming process, resulting in delayed diagnoses for patients [17]. When there is a large number of patients, the scarcity of RT-PCR tests and specialized laboratory equipment for COVID-19 detection becomes a significant issue [20]. Another diagnostic approach involves manually analyzing X-ray images by human specialists or radiologists [20].

Manual diagnosis of COVID-19 through X-ray images has limitations, such as the need for trained radiologists, potential human error, longer diagnostic times, and a higher likelihood of false-negative results. As a result, there is an urgent need for tools that can optimize available resources, improve diagnostic accuracy, and reduce processing time [10], [3]. Deep learning (DL) and Big Data Analytics (BDA) are such tools [20], [21], [4]. They can expedite decision-making processes and predict the progression of COVID-19 cases [22], [4]. Many countries have successfully managed the COVID-19 pandemic by leveraging Big Data and Machine Learning methods, marking a significant milestone in epidemic control [3]. These countries have implemented effective safety measures, incorporating big-data analytics and Artificial Intelligence to address emergencies [3], [39]. They are gathering vast amounts of data and using it creatively, integrating Deep Learning and Big Data analytics to confront ongoing crises [3].

X-ray imaging has emerged as a primary method for diagnosing COVID-19 cases [18], [19]. This study focuses on utilizing X-ray images for COVID-19 detection. In this paper, we propose a rapid and accurate approach to detect COVID-19 from large-scale X-ray databases by classifying images as either COVID-19 or non-COVID-19 (see Fig. 1) using a distributed Deep Convolutional Neural Network (DCNN) model on the Spark-TensorFlow-Distributor framework [1]. This framework, built on top of TensorFlow's 'tensorflow.distribute.Strategy', enables distributed training on the Spark platform.

The Spark framework is optimized for parallel processing of large datasets [23], [24]. Furthermore, the Hadoop Distributed File System (HDFS) is utilized for efficient storage and management of big data, which is especially critical for COVID-19 data due to its rapid and large-scale generation [23], [40]. Our key contributions are as follows:

1. Enabling parallel processing of large-scale chest X-ray COVID-19 images using the Spark-TensorFlow-Distributor framework.

2. Combining deep learning with the Spark-TensorFlow-Distributor framework for the rapid and efficient classification of large volumes of chest X-ray images. Our approach achieves outstanding training and testing times through the barrier mode of Spark, optimizing distributed training tasks.

3. Presenting the findings from the proposed fine-tuning approach for the DCNN classifier.

4. Comparing the performance of our classifier with other models, such as ANN and AlexNet, in terms of accuracy and processing time.

5. Evaluating our approach against recent state-of-the-art methods regarding classification accuracy and execution time.

The remainder of this paper is organized as follows: Section 2 reviews related works in COVID-19 detection and classification. Section 3 provides the background and tools used in this study. Section 4 details the proposed method. Section 5 presents the classification performance. Experimental results in Section 6. Finally, the conclusions are drawn in Section 7.

2 Related works

Recently, the application of Deep Learning techniques has become widespread in predicting and identifying COVID-19 due to its high accuracy [1]. Hemdan et al. [32] introduced a system for the recognition of COVID-19 based on chest X-ray images. This system incorporates seven image classifiers, collectively known as COVIDX-Net, achieving notable performance, with DenseNet201 and VGG19 classifiers achieving an accuracy of 90%. Basu et al. [31] proposed an advanced method for COVID-19 detection that employs transfer learning with a pre-trained deep convolutional neural network (DCNN) using chest X-ray images. The authors achieved an overall accuracy of 90.1%. However, this system requires a limited number of chest X-ray images for COVID-19 identification. Similarly, for COVID-19 detection from chest X-ray images, the authors in [25] proposed an advanced DCNN model utilizing pre-trained transfer architectures such as ResNet50, InceptionV3, and Inception-ResNetV2. Their approach demonstrated exceptional predictive performance in terms of time and accuracy, tested on a limited set of X-ray images. To address limited training data and processing time, they employed transfer learning using the ImageNet dataset [25]. The results validated the superior accuracy of the ResNet50 model across both training and testing evaluations.

In another study [12], the authors proposed a model using Explainable Artificial Intelligence (XAI) techniques to detect COVID-19 from chest X-ray (CXR) images. They employed transfer learning, data augmentation, and lung segmentation for faster model training and improved performance. By using the ResNet model, they achieved the highest classification performance (F1-Score: 98%). The authors in [11] developed an efficient method for classifying images as either COVID-19 or non-COVID-19 using a CNN for feature extraction and SVM for classification. By incorporating image augmentation, segmentation, and cropping techniques, they achieved optimal results, with a training accuracy of 99.8% and a testing accuracy of 99.1% on a dataset of 3000 images.

The rapid spread of COVID-19 has generated vast amounts of data, which continues to grow exponentially [8]. This data can be effectively utilized by applying big data analytics techniques for COVID-19 detection. Several studies have used big data analytics and artificial intelligence (AI) methods to enhance COVID-19 detection performance [8]. For instance, the authors in [30] explores the integration of AI, big data tools, and nature-inspired computing (NIC) to improve the performance of COVID-19 detection. Moreover, the authors of reference [6] utilized a Hive-based distributed system operating on a Hadoop cluster to query and analyze large COVID-19 datasets, aiming to extract valuable insights. Hadoop serves as a parallel computing system for processing vast volumes of data, while Hive functions as a data warehouse for the Hadoop cluster. Their proposed approach facilitates queries and predictive analytics on massive COVID-19 datasets.

In [30], the authors proposed a model to distinguish COVID-19 from four other viral chest diseases by utilizing various body sensors, which gather data on parameters such as temperature, blood pressure, heart rate, respiratory patterns, glucose levels, and more. The data is stored in a cloud database and analyzed by AI-enabled expert systems to identify infected or suspected COVID-19 patients. Similarly, Gupta et al. [28] identified COVID-19 cases in

[1]spark-tensorflow-distributor: https://docs.databricks.com/en/machine-learning/train-model/distributed-training/spark-tf-distributor.html

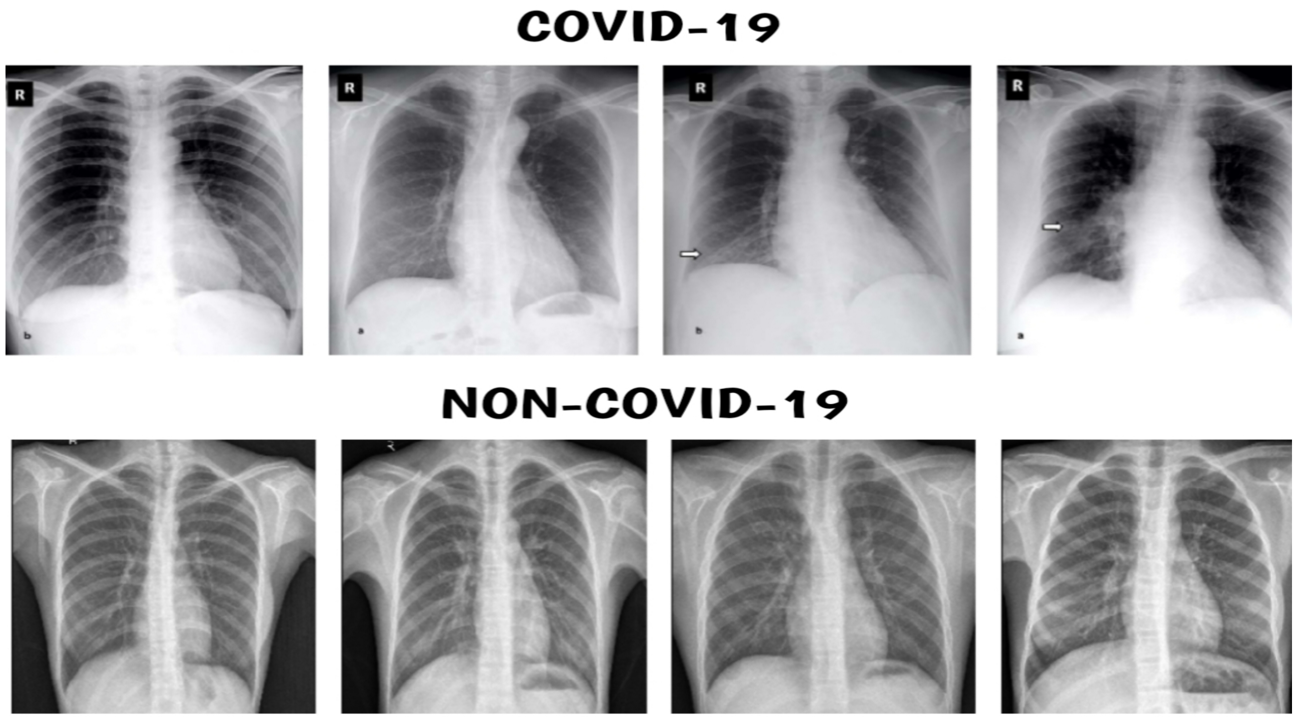

Figure 1: Sample X-ray images of normal (non-covid-19 cases) and (covid-19 positive cases)

India using models such as random forests, linear models, support vector machines, decision trees, and neural networks. The random forest model was selected for its superior performance, and K-fold cross-validation was employed to assess model consistency. They also introduced a novel federated learning (FL) approach to improve the global GAN model's ability to generate realistic COVID-19 images without sharing sensitive data.

The authors of reference [27] developed a system to detect COVID-19 using chest X-ray images sourced from Kaggle repositories. Their system is based on pre-trained Apache Spark big data framework models using Deep Transfer Learning (DTL) via Convolutional Neural Networks (CNNs), specifically InceptionV3 and ResNet50. Their study showed that both models achieved impressive results, with InceptionV3 surpassing 99% accuracy and ResNet50 reaching up to 98%. The proposed model accelerates COVID-19 detection and reduces associated costs. As part of future work, the authors plan to enhance their system by integrating TensorFlow with Apache Spark to develop a new model for detecting COVID-19 in chest CT images. This integration will enable the creation of a model that can operate on large computing clusters.

The authors in [38] introduced a method called DisCOV, which involves training a distributed COVID-19 detection model using a large-scale chest X-ray dataset, in collaboration with edge and cloud computing. To enhance training efficiency and ensure model accuracy, they implemented a resource allocation algorithm to minimize time, cost, and energy consumption during training. Their method achieved a maximum accuracy of 94%. In another study, the authors in [29] used the COVID-Net CNN model to detect COVID-19 from a large dataset of 13,975 CXR images. They applied various data augmentation techniques such as translation, rotation, horizontal flipping, zooming, and intensity shifting, achieving a recognition accuracy of 93.38%.

3 Materials and methods

A widely adopted approach for COVID-19 detection involves the use of chest X-rays, which offer a cost-effective method. However, this method requires further enhancements to improve its efficiency. Most existing studies have relied on chest X-ray images for COVID-19 detection using deep learning models, though typically on a limited set of images [3], [27]. To address these limitations and facilitate the development of models capable of running in a distributed manner across clusters, this study proposes a novel approach that combines big data analytics with Deep Convolutional Neural Networks (DCNNs) for detecting COVID-19 from a large volume of chest X-ray images. The primary objective of our research is to enhance COVID-19 detection performance, particularly with respect to training and execution times.

In our approach, we use TensorFlow version 2 on Apache Spark version 3. [2], [3] enabling deep learning models to ex-

[2]Blog, M. (2020). Distributed tensorflow on apache spark 3.0. https://blog.madhukaraphatak.com/tensorflow-on-spark-3.0

[3]pypi (2021). Spark tensorflow distributor. https://pypi.org/project/spark- tensorflow-distributor/

Table 1: Comparison of COVID-19 detection approaches in related works

Study	Method	Dataset Size	Accuracy	Key Features
[32]	COVIDX-Net	Not specified	90%	DenseNet201, VGG19 classifiers
[31]	Transfer learning with DCNN	Not specified	90.1%	Pre-trained DCNN, transfer learning
[25]	ResNet50, InceptionV3, Inception-ResNetV2	Limited dataset	High accuracy	Transfer learning, limited images
[12]	XAI, Transfer Learning, Data Augmentation	Not specified	F1-Score: 98%	Explainable AI, lung segmentation
[11]	CNN + SVM	3000 images	99.8% (train), 99.1% (test)	Image augmentation, segmentation
[6]	Hadoop, Big Data Analytics	Large dataset	N/A	Hive-based system, big data analytics
[28]	Random Forests, SVM, Neural Networks	Not specified	High performance	Federated learning, multi-model approach
[27]	Apache Spark, DTL with CNNs (InceptionV3, ResNet50)	Kaggle dataset	InceptionV3: 99%, ResNet50: 98%	Deep Transfer Learning, big data
[38]	DisCOV (Distributed model)	Large dataset	94%	Distributed training, edge-cloud collaboration
[29]	COVID-Net CNN	13,975 images	93.38%	Data augmentation, large dataset

ecute within the Apache Spark framework in a distributed fashion. A detailed description of the proposed methodology is provided in the following sections.

3.1 Big data

Big Data refers to the vast amounts of data generated in various forms (such as text, data, sound, and video) across unstructured, structured, and semi-structured formats. This overwhelming volume of data surpasses the capabilities of traditional data processing methods. It involves the collection, storage, and analysis of large datasets to uncover patterns, trends, and valuable insights [24], [41].

3.2 Apache spark

Apache Spark is an open-source, distributed computing system designed to process large volumes of data. It provides a fast, general-purpose cluster computing framework and supports several programming languages, including Scala, Java, Python, and R. Spark offers high-level APIs for distributed data processing, including batch processing, streaming, machine learning, and graph processing. Known for its in-memory processing capabilities, Spark outperforms traditional MapReduce-based frameworks in terms of speed [37], [24], [56].

3.3 Spark-TensorFlow-distributor

The ‘spark-tensorflow-distributor‘ is a Python package that enables distributed training with TensorFlow on Spark clusters. It builds upon the ‘tensorflow.distribute.Strategy‘ API, a key feature of TensorFlow version 2.x that allows for distributing training across multiple GPUs, machines, or TPUs [4], [5]

Furthermore, the ‘spark-tensorflow-distributor‘ package implements the barrier execution mode in Spark to facilitate distributed TensorFlow training on Apache Spark version 3.x clusters. In contrast to Spark version 2, which was limited to distributing training using the traditional MapReduce execution model (where workers operate independently without communication), Spark version 3.0 introduces a barrier execution mode. This new mode enables seamless communication between workers during the training process, making it more suitable for deep learning applications. In our study, we leverage ‘spark-tensorflow-distributor‘ to implement distributed TensorFlow training with version 2.x on Spark version 3.x, utilizing a large dataset of chest X-ray (CXR) images for COVID-19 detection. This approach enhances training times and improves the overall speed of the system.

3.4 Deep learning

Deep learning, a subset of machine learning, involves using neural networks with multiple layers (deep neural networks) to model and solve complex problems. By emulating the human brain, these networks can learn hierarchical representations of data, enabling the model to perform tasks such as pattern recognition and decision-making.

3.5 Chest X-ray imaging

Chest X-ray imaging is one of the most commonly used medical imaging techniques for diagnosing and assessing the extent of infections, offering several advantages [35]. These advantages include its ease of processing, which reduces imaging time and minimizes the risk of spreading COVID-19 cases [36]. Additionally, chest X-rays are more economical compared to other imaging modalities and involve lower radiation doses compared to CT scans.

However, despite these benefits, chest X-ray imaging has limitations. It is less sensitive than other techniques, which may lead to false predictions, particularly in the early stages of the disease [32].

4 Methodology

4.1 System architecture

We propose a novel system for large-scale COVID-19 image classification that leverages the Spark-TensorFlow-Distributor, enabling distributed training across multiple GPUs or machines (see Fig. 2). The proposed system utilizes Spark-TensorFlow-Distributor to process the deep

[4]pypi (2021). Spark tensorflow distributor. https://pypi.org/project/spark- tensorflow-distributor/

[5]databricks (2020). Distributed training with tensorflow 2. https://docs.databricks.com/machine-learning/train-model/distributed-training/spark-tf-distributor.html

learning model in a parallel and distributed manner, handling large datasets on the Spark platform. The system performs in-memory computations and adopts a master/slave architecture, consisting of two primary nodes: the master node (pilot process) and the slave node (worker process), as illustrated in Fig. 2. The system's input consists of a large volume of COVID-19 images, and the output is classified as either COVID-19 or non-COVID-19.

Our approach employs a distributed Deep Convolutional Neural Network (DCNN)-based deep learning model for large-scale COVID-19 image detection. This model operates with Spark-TensorFlow-Distributor, which is deployed on top of an Apache Spark cluster (version 3) integrated with TensorFlow (version 2). The YARN cluster manager is responsible for scheduling and resource allocation across applications, ensuring efficient parallel data training by balancing the workload and controlling training costs. A key feature of the proposed system is the use of Spark-TensorFlow-Distributor, an open-source package that facilitates distributed training with TensorFlow on Spark clusters. This package is built on top of 'tensorflow.distribute.Strategy', a core feature of TensorFlow 2. The system requires Spark 3.0.1, Python 3.6+, and TensorFlow 2.1.0+ to function effectively. After training our DCNN model, we utilize Spark-TensorFlow-Distributor to perform distributed training on our Spark clusters.

4.1.1 Advantages of Spark 3.x over Spark 2.x

Spark 2.x relies on the traditional Map/Reduce execution model, where a Spark program typically consists of a sequence of map and reduce stages. This model, inspired by Hadoop, works well for many big data tasks such as ETL (Extract, Transform, Load), SQL operations, and basic machine learning tasks. However, Spark 2.x's scheduling approach is less efficient when implementing deep learning frameworks.

To address this limitation, Spark 3.x introduces a new execution mode known as the "Mirrored-Strategy Runner" in barrier execution mode, which is specifically designed for distributed deep learning tasks. Unlike the traditional Map/Reduce model, this execution mode operates differently. In the Map/Reduce model, tasks within a stage are independent and do not communicate with each other; consequently, if a task fails, only that specific task is retried. In contrast, the "Mirrored-Strategy Runner" mode executes all tasks within a stage simultaneously, and if any task fails, the entire stage is retried. Moreover, this mode allows for communication among tasks within the same stage. Although this scheduling model improves efficiency for deep learning frameworks, it is still less optimal than the barrier execution mode. Therefore, Spark 3.x introduces barrier execution mode with the Mirrored-Strategy Runner to optimize distributed deep learning, offering an advantage over the standard Map/Reduce model [8], [9].

The primary steps of the proposed system architecture, as shown in Fig. 2, include image preparation, preprocessing, and large-scale COVID-19 image classification.

4.2 Step 1: Dataset preparation and preprocessing

The first step involves preparing and processing the dataset for training. To train the Deep Convolutional Neural Network (DCNN) model, a set of labeled images is required.

4.2.1 Dataset preparation

The process begins by loading a large collection of COVID-19 chest X-ray images from the Hadoop Distributed File System (HDFS) [42], [24], which stores a massive number of images in a distributed computing architecture. These images are subsequently divided into multiple equal-sized chunks. Each chunk is duplicated in an HDFS data node within our master-slave architecture. Once the data is loaded and partitioned, a set of Spark workers processes the individual images from each chunk.

4.2.2 Dataset preprocessing

Each Spark worker performs parallel preprocessing on its assigned chunk of COVID-19 images using the Keras `ImageDataGenerator`. This method enables real-time image preprocessing by loading the image dataset into memory and generating batches of preprocessed data. By utilizing this approach, image preprocessing occurs dynamically during the training process, which allows for better memory management. This dynamic generation of preprocessed images helps in handling large datasets efficiently.

Several preprocessing techniques are employed in medical imaging applications to enhance the quality of images. For COVID-19 image analysis, essential preprocessing methods include image resizing, image enhancement, and image segmentation [35]. In this study, we primarily use image resizing to standardize the COVID-19 chest X-ray images. Image resizing is necessary to reduce computational costs and processing time, as large pixel sizes can significantly increase the time required for training [11]. During preprocessing, images are resized to 128x128 pixels and transformed into a NumPy array compatible with Keras, ensuring better recognition performance in the DCNN model.

4.3 Step 2: COVID-19 image classification

In this phase, deep learning methods are applied using Apache Spark to accelerate the training and classification processes on a large-scale COVID-19 image dataset. The Deep Convolutional Neural Network (DCNN) architecture is employed to train and classify chest X-ray images, categorizing them into COVID-19 and non-COVID-19 groups. The DCNN model is constructed and compiled using the Spark-TensorFlow-Distributor framework. The DCNN has achieved significant advancements in various pattern recognition tasks, including image classification [33]. The modern DCNN architecture consists of al-

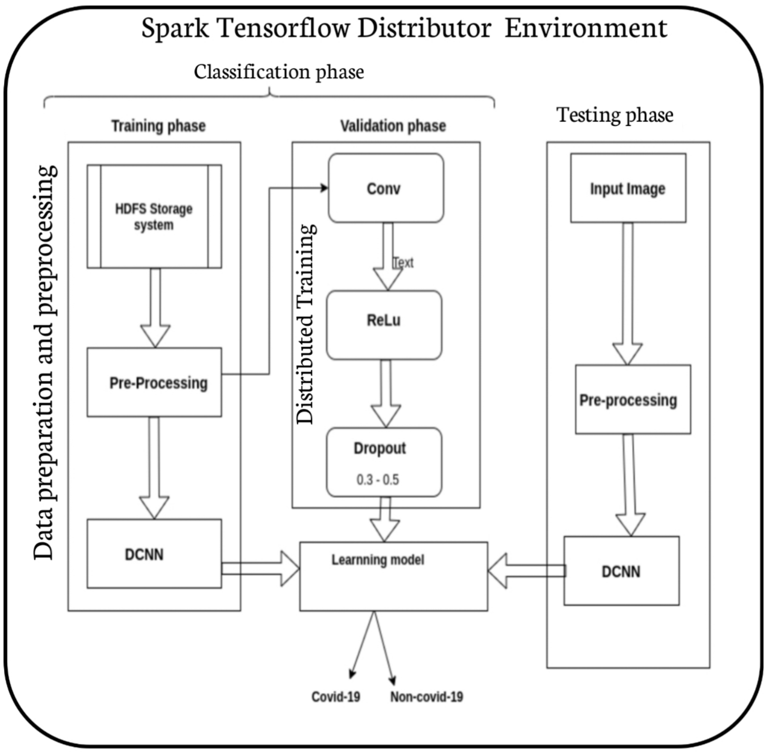

Figure 2: Architecture of our proposed system

ternating convolutional layers (CL) and max-pooling (MP) layers, followed by fully connected layers (FC).

In the first convolutional layer, an activation map of size $128\times128\times32$ is generated after applying 32 filters. The output of this layer is passed to the max-pooling layer, where 2×2 filters are used to perform down-sampling. The second convolutional layer takes the output from the first layer, which has a size of $64\times64\times32$, and applies 64 filters, resulting in an activation map of size $64\times64\times64$. Another max-pooling layer with a 2×2 window is applied, reducing the size to $32\times32\times64$. Subsequently, a third convolutional layer with 128 filters produces an activation map of size $16\times16\times128$, followed by another max-pooling layer that reduces the size to $8\times8\times128$. The final convolutional layer applies 256 filters to this output, producing a $4\times4\times256$ volume. A fully connected layer with 1200 neurons is then applied, followed by the output layer with a single neuron.

For activation functions, we use the Rectified Linear Unit (ReLU) for all layers except the final one. The ReLU function is commonly used in classification tasks, as it activates the convolutional layers by outputting the input directly if it is positive, and zero otherwise. Mathematically, ReLU is defined as:

$$f(x) = \max(0, x) \tag{1}$$

ReLU introduces non-linearity into the model, allowing it to learn complex patterns, while being computationally efficient compared to other activation functions. For the final output layer, we use the sigmoid function, which transforms real-valued inputs to a range between 0 (non-COVID-19) and 1 (COVID-19), making it suitable for binary classification tasks. The sigmoid function is mathematically defined as:

$$f(x) = \frac{1}{1+e^{-x}} \tag{2}$$

4.3.1 Pooling layer

The pooling layer serves as a down-sampling technique used to reduce the spatial dimensions of each feature map while retaining essential information. Max-pooling is the most commonly used type of pooling, where the maximum value from a group of values is selected. In our model, each feature map undergoes processing with a 2×2 max-pooling layer, which reduces the dimensionality and highlights the most important features. This operation helps to decrease computational complexity and focuses on the relevant features within the image.

4.3.2 Flattening layer

The flattening layer converts the multi-dimensional output from the preceding convolutional or pooling layers into a one-dimensional array, which is then fed into the fully connected layers. This transformation allows the network to make predictions based on the features learned in previous layers. In our DCNN architecture, we use a fully connected layer with 1300 neurons, followed by an output layer with a single neuron.

4.3.3 Dropout layer

To mitigate overfitting during training, we use a dropout layer in the DCNN architecture. In each training iteration, a random selection of neurons is temporarily omitted with a specified probability. This regularization technique helps improve the model's ability to generalize to unseen data by reducing reliance on specific neurons. In our model, we applied a dropout rate of 0.3 in the convolutional layers and 0.5 in the fully connected layers.

4.3.4 Optimizer

For optimizing the model, we employed the Adaptive Moment Estimation (Adam) optimizer. Adam dynamically adjusts the network's weights during training based on the provided data. It is particularly advantageous for networks that handle large datasets and parameters, offering efficiency in both computation and memory usage.

4.3.5 Model training

We carefully optimized the hyperparameters for our learning approach by evaluating other complex architectures, such as Artificial Neural Networks (ANN), AlexNet, and DCNN. Hyperparameter tuning was conducted for each model by adjusting key parameters such as learning rate, batch size, and weight decay to achieve optimal performance. For the DCNN model, this process was repeated extensively to refine the hyperparameters, ensuring the best possible performance in terms of classification accuracy.

5 Classification performance

To evaluate the performance of our Deep Convolutional Neural Network (DCNN) model, we employed various metrics. A confusion matrix was utilized to assess the model's performance, that includes the following four key parameters:

- **True Positive (TP)**: The number of correctly predicted positive samples.
- **True Negative (TN)**: The number of correctly predicted negative samples.
- **False Positive (FP)**: The number of incorrectly predicted positive samples.
- **False Negative (FN)**: The number of incorrectly predicted negative samples.

5.1 Accuracy

Accuracy is a commonly used metric to evaluate the model's overall performance, representing the model's ability to correctly differentiate between classes in the test set. The accuracy is defined as:

$$\text{ACC} = \frac{TP + TN}{TP + FP + FN + TN} \tag{3}$$

As shown in Equation (3), accuracy is the proportion of correctly predicted labels over the total number of samples in the dataset.

5.2 Precision

Precision, as defined in Equation (4), represents the proportion of accurately predicted positive labels out of all predicted positive labels. It is a measure of the model's ability to avoid false positives:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4}$$

5.3 Recall

Recall, as defined in Equation (5), represents the ratio of correctly predicted positive labels to the total number of actual positive labels. It measures the model's ability to correctly identify positive instances:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{5}$$

5.4 Specificity

Specificity, as shown in Equation (6), is used to measure the model's ability to accurately recognize negative instances, i.e., the proportion of correctly identified negative samples:

$$\text{Specificity} = \frac{TN}{FP + TN} \tag{6}$$

5.5 F1-score

The F1-score, as defined in Equation (7), is the harmonic mean of Precision and Recall. It provides a balance between the two metrics, particularly useful when the data is imbalanced:

$$\text{F1-score} = \frac{2 \times (\text{Recall} \times \text{Precision})}{\text{Recall} + \text{Precision}} \tag{7}$$

The F1-score is particularly useful for evaluating the overall performance when there is an uneven class distribution between the positive and negative classes.

5.6 Area under the curve (AUC)

The Area Under the Curve (AUC) is a performance metric that evaluates the ability of the model to distinguish between positive and negative classes. Specifically, the AUC measures the area under the Receiver Operating Characteristic (ROC) curve. AUC values range from 0 to 1, with higher values indicating better model performance. An AUC value of 0.5 indicates a model with no discrimination ability (random guessing), while a value of 1.0 represents perfect classification performance.

$$\text{AUC} = \int_0^1 \text{ROC Curve} \tag{8}$$

5.7 Matthews correlation coefficient (MCC)

The Matthews Correlation Coefficient (MCC) is a metric used to evaluate the quality of binary classifications. It considers all four components of the confusion matrix (TP, TN, FP, FN) and is particularly useful when the classes are imbalanced. The MCC is defined as:

$$\text{MCC} = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{9}$$

MCC values range from -1 to +1. A value of +1 indicates perfect classification, while -1 indicates total disagreement between the predicted and actual labels. A value of 0 indicates no better performance than random classification.

5.8 Cohen's kappa

Cohen's Kappa is a statistic that measures inter-rater agreement for categorical items. It is used to assess the agreement between two raters or classifiers and adjusts for the possibility of random chance. Cohen's Kappa is defined as:

$$\kappa = \frac{P_o - P_e}{1 - P_e} \tag{10}$$

Where: - P_o is the observed agreement (proportion of times the model's predictions match the true labels). - P_e is the expected agreement (proportion of times the labels would match due to chance).

Kappa values range from -1 to 1, where 1 indicates perfect agreement, 0 indicates agreement no better than chance, and negative values indicate less agreement than would be expected by chance.

6 Experimental results

6.1 Environment description

To evaluate the performance of the proposed method, we utilized a chest X-ray (CXR) image dataset sourced from Kaggle [6]. his dataset consists of 13,808 CXR images, which were used for both training and testing our model. The images were categorized into two classes: COVID-19 and non-COVID-19. The implementation was carried out using the Apache Spark framework on a single-node cluster (standalone setup). The specifications of the cluster are as follows:

– **Software Environment**: Ubuntu 18.04 LTS, Spark 3.0.1, Keras, Python 7.13, and TensorFlow 2.7.

– **Hardware Environment**: Intel Core i5-4210U CPU with a 1.70 GHz clock speed (4 cores) and 8 GB of RAM.

Presentation of the dataset

Access to large-scale COVID-19 datasets for non-commercial research is often limited due to privacy concerns. As a result, many studies in this area rely on smaller, publicly available datasets. For this study, we employed a comprehensive dataset available on Kaggle, which includes chest X-ray (CXR) images of COVID-19-positive and pneumonia cases. This dataset, curated by a collaborative team from the University of Qatar, Bangladesh, Pakistan, and Malaysia, in partnership with medical experts, contains 13,808 PNG-format CXR images, each with a resolution of 299 pixels.

For the analysis, we focused on a subset of 12,000 X-ray images, consisting of 3,000 COVID-19 cases and 9,000 non-COVID-19 cases. Due to the inherent class imbalance in the dataset, with fewer COVID-19 samples, an image augmentation technique was applied to increase the number of COVID-19 images. The augmentation methods, which included image shifts, flips, and rotations, resulted in a total of 18,000 images (9,000 COVID-19 and 9,000 non-COVID-19). The dataset was then partitioned into 80% for training and 20% for testing, as suggested in [10]. The training set contained 14,400 images, equally distributed between the two classes (7,200 COVID-19 and 7,200 non-COVID-19), while the testing set included 3,600 images (1,800 from each class). This balanced approach ensured robust training and evaluation of the model.

6.2 Results and discussion

Table 2 presents the results of training and testing the Artificial Neural Network (ANN) model with a dropout rate of 0.25 on the 18000 chest X-ray (CXR) dataset. The metrics included are the training and testing accuracy, test epoch time, and overall training time for different batch sizes. It is observed that the highest testing accuracy (84.62%) is achieved with a batch size of 16, which required a training time of 49 minutes. As the batch size increased to 32, the test accuracy declined significantly to 50%, despite a slight reduction in training time (48 minutes 23 seconds). Furthermore, with a batch size of 100, the test accuracy decreased to 64.89%, and the training time was further reduced to 47

[6]COVID-19 Radiography Database. COVID-19 Chest X-ray images and Lung masks Database. https://www.kaggle.com/datasets/tawsifurrahman/covid19-radiography-database/data

Table 2: The effect of batch size on the ANN model with 13,808 CXR dataset

Batch Size	Accuracy of Training (%)	Test Accuracy (%)	Test Epoch Time	Training Time
16	99.79	84.62	53s	49 min
32	98.80	50.00	42s	48 min 23s
100	99.84	64.89	35s	47 min 25s

minutes 25 seconds. These results indicate that increasing the batch size from 16 to 100 improved training efficiency by reducing training time but negatively impacted model accuracy.

Table 3 compares the training and testing times, along with the accuracy of the ANN model, for various numbers of epochs with a dropout rate of 0.25. It is evident that the number of epochs has no significant effect on the training accuracy of the ANN model, which reaches 100% after eight epochs. However, the testing accuracy fluctuates across different epoch values, peaking at 84.62% after six epochs. The primary impact of increasing the number of epochs is on training time, where longer training durations are observed as the epoch count increases, highlighting a trade-off between computational time and performance.

Table 4 shows the impact of different epoch counts on the performance of the DCNN model, without dropout, using a batch size of 32. The results reveal that while training accuracy remains high at 100% from 18 epochs onwards, testing accuracy improves with an increasing number of epochs and stabilizes at 91.88% after 100 epochs. Notably, training time increases with the number of epochs, with the longest duration recorded at 5 hours, 28 minutes, and 50 seconds for 100 epochs. The testing accuracy shows a clear trend: it is lowest at 60.43% after just 2 epochs, increases gradually to 70.73% at 25 epochs, and reaches the optimal value of 91.88% after 100 epochs. This indicates that while longer training times are necessary for optimal classification performance, the model benefits from additional epochs, especially when the number of epochs exceeds 18.

Table 5 illustrates the effect of dropout rates on the accuracy of the DCNN model using the same dataset and batch size of 32. The results demonstrate that increasing dropout rates in both the convolutional layer (CL) and fully connected layer (FC) improves the model's classification accuracy. The highest accuracy of 95.53% was achieved with a dropout rate of 0.3 in the CL and 0.5 in the FC layer. For lower dropout rates (0.0 to 0.2 in both layers), the accuracy is relatively lower, fluctuating between 90.53% and 93.50%. However, introducing dropout rates of 0.3 or 0.4 in the CL and FC layers significantly boosts the model's performance, with the optimal result observed at a dropout rate of 0.3 in the CL and 0.5 in the FC layer. The findings indicate that regularization using dropout not only prevents overfitting but also contributes significantly to improved generalization, thus enhancing the model's overall classification capability. The results suggest that a balanced dropout strategy (e.g., 0.3 in CL and 0.5 in FC) yields the best performance, while further increasing the dropout rate in the FC layer beyond 0.5 may lead to diminishing returns in accuracy.

Table 7 details the optimized architecture of the deep convolutional neural network (DCNN), achieving a peak accuracy of 95.53% through systematic hyperparameter tuning and refinement. The learning rate was set to **0.001**, striking a balance between convergence speed and stability. The Adam optimizer was employed, leveraging the combined benefits of AdaGrad and RMSProp to ensure efficient and adaptive gradient updates. To enhance generalization, **dropout rates** were configured at **0.3** for convolutional layers and **0.5** for fully connected layers, preventing overfitting. A **batch size of 32** was chosen, providing a balance between training speed and model performance. Additionally, **weight decay (0.0001)** was incorporated to further regularize the model. The DCNN architecture consisted of multiple convolutional layers, progressively increasing the number of filters from **32** in the first layer to **256** in the final convolutional layer. Each convolutional layer was followed by a **2×2 max-pooling layer**, aiding in feature extraction and dimensionality reduction. The fully connected layers consisted of **1200 neurons**, with a **sigmoid activation function in the output layer**, suitable for binary classification. Regularization techniques, including dropout, weight decay, and max-pooling, were employed to enhance robustness. The model was trained using the **spark-tensorflow-distributor framework**, leveraging distributed computing for efficient training on large datasets. Furthermore, as illustrated in Table 6, our proposed system, which employs a DCNN model with the fine-tuning configuration detailed in Table 7, achieved the following results:

To provide a more comprehensive evaluation, we included the following metrics:

- **Area Under the Curve (AUC)**: The ROC curve was plotted, and the AUC was calculated to be **0.98**, indicating excellent discriminative ability.
- **Matthews Correlation Coefficient (MCC)**: The MCC value of **0.93** highlights the model's balanced performance, especially for imbalanced datasets.
- **Cohen's Kappa**: A value of **0.92** was obtained, indicating strong agreement between predicted and actual labels, adjusted for chance.

Table 3: Study of the effect of epochs on the ANN model

Epochs	Training Accuracy (%)	Test Accuracy (%)	Time of Epoch Test	Training Time
2	96.89	67.45	55 sec	16 min 12 sec
4	98.59	84.46	54 sec	32 min 5 sec
6	99.79	84.62	53 sec	48 min
8	100	75.87	59 sec	1h 3 min 52 sec
10	100	72.60	56 sec	1h 19 min 39 sec
14	100	70.13	55 sec	1h 51 min 33 sec
18	100	72.29	54 sec	2h 28 min 5 sec
20	100	70.65	55 sec	2h 45 min 5 sec

Table 4: Study of the effect of epochs on the DCNN model with 13,808 CXR dataset

Number of Epochs	Training Accuracy (%)	Training Time	Testing Accuracy (%)	Testing Time
2	88.79	4 min 4 sec	60.43	51 sec
4	95.89	8 min 15 sec	61.88	49 sec
8	70.10	16 min 33 sec	70.41	50 sec
18	100	43 min 48 sec	69.73	51 sec
25	100	58 min 50 sec	70.73	52 sec
100	100	5 h 28 min 50 sec	91.88	50 sec

Table 5: Study of the dropout effect on the DCNN model

Dropout in CL	Dropout in FC	Accuracy
0.0	0.0	90.53
0.0	0.1	90.69
0.0	0.2	90.88
0.0	0.5	90.73
0.1	0.0	92.40
0.1	0.1	92.55
0.1	0.2	92.16
0.2	0.0	92.66
0.2	0.1	92.88
0.2	0.2	92.95
0.2	0.5	93.50
0.3	0.0	93.96
0.3	0.1	94.60
0.3	0.2	94.99
0.3	0.5	95.53
0.4	0.0	95.06
0.4	0.1	93.89
0.4	0.2	93.54
0.4	0.5	92.33

Table 6: Performance metrics of the proposed model

Metric	Value
Sensitivity (Recall)	0.94
Specificity	0.96
Precision	0.95
Accuracy	0.9553
F1-Score	0.95
Training Time (seconds)	18909
Testing Time per Image (seconds)	0.006
Number of Images	18000

The tables 8 and 9 present the effects of batch size and dropout rates at the fully connected (FC) layer on the AlexNet model using CXR images. The results indicate that neither batch size nor dropout significantly impacts accuracy or training time in this architecture. The highest classification accuracy, 73.8%, was consistently achieved across various configurations, with minimal variation in training time. For example, a batch size of 16 achieved 73.8% accuracy with a training time of 1 hour, 16 minutes, and 45 seconds, while other batch sizes produced similar results.

Figures 3 and 4 present a comparison of training durations for the DCNN and ANN models. The DCNN achieved optimal performance with a training time of 18,909 seconds (approximately 5 hours and 15 minutes) over 100 epochs, attaining a recognition accuracy of 91.88%. In contrast, the ANN model required significantly

Table 7: Summary of hyperparameters and fine-tuning strategies for deep convolutional neural network (DCNN) model

Parameter	**Value/Description**
Learning Rate	0.001
Dropout Rates	- Convolutional Layers (CL): 0.3 - Fully Connected Layers (FC): 0.5
Optimizer	Adam (combines AdaGrad and RMSProp)
Batch Size	32
Weight Decay	0.0001
Convolutional Layers	- 1st CL: 32 filters, output: 128x128x32 - 2nd CL: 64 filters, output: 64x64x64 - Final CL: 256 filters, output: 4x4x256
Max-Pooling Layers	2x2 window size
Fully Connected Layers	- FC layers: 1200 neurons - Output layer: 1 neuron
Activation Functions	- ReLU for all layers except output - Sigmoid for output layer
Training Framework	spark-tensorflow-distributor
Regularization Techniques	- Dropout layers - Weight decay - Max-pooling for dimensionality reduction

Table 8: Study of the batch effect with a dropout of 0.4 on the AlexNet model

Batch Size	**Training Accuracy (%)**	**Training Time**	**Testing Accuracy (%)**	**Epoch Time**	**Loss Function**
16	73.82	1h 16m 45s	73.8	52s	0.575
60	73.82	1h 16m 50s	73.8	59s	0.57
128	68.78	1h 20m 45s	73.8	55s	0.60
300	73.50	1h 15m 56s	73.8	53s	0.60
500	73.79	1h 16m 50s	73.8	52s	0.60

less training time of 1,960 seconds (approximately 33 minutes) over 6 epochs—to achieve an accuracy of 84.62%, both utilizing a batch size of 32. Additionally, these experiments examined the effect of memory allocation per executor. When memory allocation exceeded 2,048 MB, training times for both the DCNN and ANN decreased substantially. Conversely, when memory allocation was below this threshold, training times increased notably.

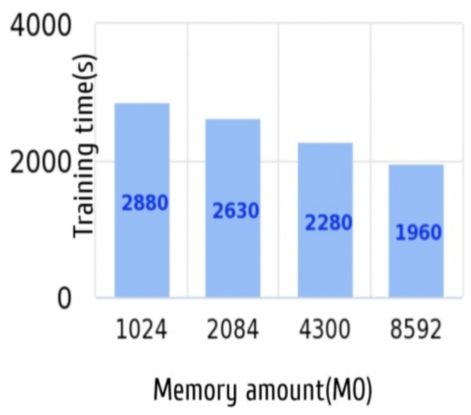

Figure 3: Impact of amount memory Spark's on the training time of ANN method (where the best training time is 1960s for 6 epochs) using memory size of 8592MO

6.3 Performance comparison with state-of-the-art models

Table 10 compares the performance of various COVID-19 detection models using chest X-ray images, highlighting key metrics such as sensitivity, specificity, precision, accuracy, F1-score, and computational efficiency. Among the 2020 models, COVID-Net and COVID-CAPS demonstrated moderate accuracy (85% and 95%, respectively), while Shashank's method (2020) achieved high accuracy (96%) and F1-score (97%). VGG19 and DenseNet201 (2020) showed perfect specificity and precision but lower sensitivity (83%), whereas ResNetV2 variants and MobileNetV2 (2020) struggled with specificity-

Table 9: Study of the dropout effect with 5 layers and 16 batches on the AlexNet model

Dropout in FC	Training Accuracy (%)	Testing Accuracy (%)	Epoch's Test Time	Training Time
(0.2, 0.3, 0.4)	73.82	73.8	52 sec	1h 18m 30s
(0.4, 0.4, 0.2)	73.06	73.8	52 sec	1h 16m 41s
(0.4, 0.4, 0.4)	73.82	73.8	53 sec	1h 19m 30s

Table 10: Comparison between our proposed approach with 13808 CXR and related works

Method	Year	Sensitivity	Specificity	Precision	Accuracy	F1-Score	Training time	Testing time	Images
COVID-Net [43]	2020	0.90	0.80	-	0.85	0.22	-	-	447
COVID-CAPS [44]	2020	0.90	0.95	-	0.95	-	-	-	-
Shashank [45]	2020	0.98	0.91	0.96	0.96	0.97	-	-	364
VGG19 [32]	2020	0.83	1.00	1.00	0.90	0.81	2641	4.0	50
CovidGAN [46]	2020	0.95	0.94	0.90	0.94	0.92	-	-	192
ResNet50 [25]	2020	-	1.00	1.00	0.98	0.98	-	-	2800
DenseNet201 [32]	2020	0.83	1.00	1.00	0.90	0.81	2122	6.00	50
ResNetV2 [32]	2020	1.00	0.62	0.40	0.70	0.57	1086	2.00	50
-ResNetV2 [32]	2020	1.00	0.71	0.60	0.80	0.75	1988	6.00	50
Xception [32]	2020	1.00	0.71	0.60	0.80	0.75	2035	3.00	50
MobileNetV2 [32]	2020	1.00	0.55	0.20	0.60	0.33	389	1.00	50
Inception [10]	2021	0.93	0.97	0.97	0.96	0.96	1800	0.014	3760
VGG16 (Fine Tuning) [47]	2020	-	-	-	0.8526	-	-	-	380
RESNET18 (Fine Tuning) [47]	2020	-	-	-	0.8842	-	-	-	380
RESNET101 (Fine Tuning) [47]	2020	-	-	-	0.8737	-	-	-	380
RESNET50 (Fine Tuning) [47]	2020	-	-	-	0.9263	-	-	-	380
VGG19 (Fine Tuning) [47]	2020	-	-	-	0.8947	-	-	-	380
AlexNet [48]	2021	0.9388	-	0.9518	0.9479	0.9021	-	-	100
GoogleNet [48]	2021	0.9338	-	0.9512	0.8889	0.9485	-	-	100
DenseNet201 [48]	2021	0.9285	-	0.9785	0.9056	0.8608	-	-	100
ResNet18 [48]	2021	0.8913	-	0.9591	0.8728	0.9407	-	-	100
ResNet50 [48]	2021	0.8819	-	0.9828	0.9455	0.9069	-	-	100
Inceptionv3 [48]	2021	0.8925	-	0.8928	0.9640	0.9411	-	-	100
VGG16 [48]	2021	0.8905	-	0.9480	0.9651	0.9585	-	-	100
XceptionNet [48]	2021	0.9411	-	0.8919	0.8874	0.8919	-	-	100
VGG19 [48]	2021	0.8863	-	0.9658	0.8886	0.9104	-	-	100
Inceptionresnetv2 [48]	2021	0.9122	-	0.9375	0.9681	0.9213	-	-	100
Inceptionresnetv3 [25]	2021	-	0.9740	0.8240	0.9770	0.9030	16027	-	3141
ResNet101 [25]	2021	-	0.9990	0.9890	0.9470	0.6860	17841	-	3141
ResNet152 [25]	2021	-	0.98	0.7570	0.9280	0.6090	18802	-	3141
Inception-ResNetV2 [25]	2021	-	0.9830	0.84	0.9530	0.7680	23078	-	3141
VGG 16 [53]	2023	0.96	-	0.96	0.96	0.96	-	-	6000
Inceptionresnetv3 [53]	2023	0.91	-	0.91	0.91	0.90	-	-	6000
ResNet [53]	2023	0.98	-	0.98	0.98	0.98	-	-	6000
VGG 19 [53]	2023	0.89	-	0.96	0.96	0.92	-	-	6000
Ensemble learning on Spark [54]	2023	-	-	-	0.9431	-	-	-	6500
Proposed Model	2025	0.94	0.96	0.95	0.9553	0.95	18909	0.006	18000

precision trade-offs despite high sensitivity. Models from 2021, such as Inception [10] and InceptionResNetV2 [25], delivered strong accuracy (96–97.7%) but required extensive training times (e.g., 16,027–23,078 seconds). In 2023, ResNet [53] achieved near-perfect accuracy (98%) and F1-score (98%), while the proposed 2024 model stands out with balanced performance: 95.53% accuracy, 0.94 sensitivity, 0.96 specificity, and a remarkably low testing time (0.006 seconds), despite a longer training time (18,909 seconds) likely due to its large dataset (18,000 images). Computational efficiency is a key differentiator, with the proposed model's sub-second testing time outperforming older architectures like VGG19 (4 seconds) and ResNet50 (unreported but likely higher). While some models (e.g., ResNet50 [25] in 2020) achieved higher specificity (100%), the proposed model's balanced metrics, scalability, and real-time inference capabilities position it as a robust solution for clinical deployment, particularly when compared to fine-tuned models like RESNETFine50 (92.63% accuracy) or ensemble methods (94.31% accuracy). Its combination of high accuracy, competitive F1-score (95%), and minimal latency underscores its suitability for rapid, large-scale diagnostic applications.

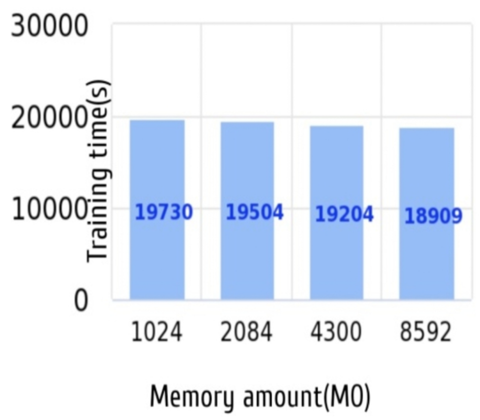

Figure 4: Impact of amount memory Spark's on the training time of DCNN method (where the best training time is 18909s for 100 epochs) using memory size of 8592MO

7 Conclusion

In this paper, we have proposed an advanced method for COVID-19 detection using deep learning and big data analytics on large-scale chest X-ray images. By leveraging the power of distributed training with Spark-TensorFlow-Distributor, we have successfully accelerated the classification process, achieving high accuracy (95.53%) in minimal time (0.06s). Our approach significantly outperforms traditional models, such as ANN and AlexNet, in terms of both classification accuracy and processing time. The integration of the distributed DCNN model with Spark's in-memory computation capabilities enables efficient handling of large-scale data, facilitating rapid and scalable training of deep learning models.

We have demonstrated the effectiveness of our system by comparing it with other state-of-the-art methods, highlighting its superior performance in terms of training and testing time. This work emphasizes the potential of combining deep learning and big data frameworks for real-time, large-scale COVID-19 detection, contributing to the timely identification of infected individuals and supporting public health efforts.

7.1 Limitations

Our approach benefits from distributed computing but is limited by infrastructure availability, requiring significant resources for large-scale deployment. The model's generalizability is constrained by its training on a specific chest X-ray dataset, and performance on diverse medical imaging data requires further evaluation. Training deep neural networks demands high-performance GPUs or TPUs, and real-time applications may require dedicated computing environments.

7.2 Future Scope

In our future research, we intend to explore further hyperparameters, such as the quantity of CL and filter dimensions, to improve the recognition capabilities of our DCNN-based method on our database. Furthermore, we intend to evaluate our model on a more extensive dataset and deploy it on large computing clusters to increase the validity of our results while optimizing our proposal through various optimization techniques referenced in [49], [50], [51], and [52].

References

[1] Lingguo Zou, Meihua Zhang (2023) Variational Autoencoder Model Combining Deep Learning and Statistical Methods for COVID-19 Detection, *Informatica*, pp. nn–mm. https://doi.org/10.31449/inf.v48i22.6921

[2] Riswantini, D., Nugraheni, E., Arisal, A., Khotimah, P. H., Munandar, D., and Suwarningsih, W. (2021). Big data research in fighting COVID-19: Contributions and techniques. *Big Data and Cognitive Computing*, 5(3), 30–89. https://doi.org/10.3390/bdcc5030030.

[3] Kaleem, S., Sohail, A., Tariq, M. U., Babar, M., and Qureshi, B. (2023). Ensemble learning for multi-class COVID-19 detection from big data. *PLOS ONE*, 18(10), e0292587. https://doi.org/10.1371/journal.pone.0292587.

[4] G. Nagarajan and D. B. L. D. (2019) Predictive analytics on big data-an overview, *Informatica*, 43(4), 425-459. https://doi.org/10.31449/inf.v43i4.2577

[5] Benbrahim, H., Hachimi, H., and 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(S, SI), S117–S129. https://www.romjist.ro/volumes/2020/23S/23S117.pdf.

[6] J. Sheng, J. Amankwah□Amoah, Z. Khan, and X. Wang (2021) COVID□19 pandemic in the new era of big data analytics: Methodological innovations and future research directions, *British Journal of Management*, 32(4), 1164-1183. https://doi.org/10.1111/1467-8551.12441

[7] Gaur, L., Bhatia, U., Jhanjhi, N. Z., Muhammad, G., and Masud, M. (2023). Medical image-based detection of COVID-19 using deep convolution neural networks. *Multimedia Systems*, 29(3), 1729–1738. https://doi.org/10.1007/s00542-022-06756-0.

[8] Alsunaidi, S. J., Almuhaideb, A. M., Ibrahim, N. M., Shaikh, F. S., Alqudaihi, K. S., Alhaidari, F. A., and Alshahrani, M. S. (2021). Applications of big data

analytics to control COVID-19 pandemic. *Sensors*, 21(7), 2282. https://doi.org/10.3390/s21072282.

[9] Tahmassebi, A., Ehtemami, A., Mohebali, B., Gandomi, A. H., Pinker, K., and Meyer-Baese, A. (2019). Big data analytics in medical imaging using deep learning. In *Big Data: Learning, Analytics, and Applications* (pp. 86–101). Springer. https://doi.org/10.1007/978-3-030-15756-3$_7$.

[10] Panahi, A. H., Rafiei, A., and Rezaee, A. (2021). FCOD: Fast COVID-19 Detector based on deep learning techniques. *Informatics in Medicine Unlocked*, 22, 100506. https://doi.org/10.1016/j.imu.2021.100506.

[11] Ayalew, A. M., Salau, A. O., Tamyalew, Y., Abeje, B. T., and Woreta, N. (2023). X-ray image-based COVID-19 detection using deep learning. *Multimedia Tools and Applications*, 1–19. https://doi.org/10.1007/s11042-023-15556-2.

[12] Sarp, S., Catak, F. O., Kuzlu, M., Cali, U., Kusetogullari, H., Zhao, Y., and Guler, O. (2023). An XAI approach for COVID-19 detection using transfer learning with X-ray images. *Heliyon*, 9(4), e14547. https://doi.org/10.1016/j.heliyon.2023.e14547.

[13] Sakr, A. S., Pawiak, P., Tadeusiewicz, R., Pawiak, J., Sakr, M., and Hammad, M. (2023). ECG-COVID: An end-to-end deep model based on electrocardiogram for COVID-19 detection. *Information Sciences*, 619, 324–339. https://doi.org/10.1016/j.ins.2022.10.029.

[14] Siddique, A. A., Talha, S. U., Aamir, M., Algarni, A. D., Soliman, N. F., and El-Shafai, W. (2023). COVID-19 classification from X-ray images: An approach to implement federated learning on decentralized dataset. *Computers, Materials Continua*, 3883–3901. https://doi.org/10.32604/cmc.2023.029907.

[15] Heymann, D. L., & Shindo, N. (2020) COVID-19: what is next for public health?, *The Lancet*, 395(10224), 542–545. https://doi.org/10.1016/S0140-6736(20)30374-3.

[16] Sohrabi, C., Alsafi, Z., O'Neill, N., Khan, M., Kerwan, A., Al-Jabir, A., & Agha, R. (2020) World Health Organization declares global emergency: A review of the 2019 novel coronavirus (COVID-19), *International Journal of Surgery*, 76, 71–76. https://doi.org/10.1016/j.ijsu.2020.02.034.

[17] Huang, P., Liu, T., Huang, L., Liu, H., Lei, M., Xu, W., ... & Liu, B. (2020) Use of chest CT in combination with negative RT-PCR assay for the 2019 novel coronavirus but high clinical suspicion, *Radiology*, 295(1), 22–23. https://doi.org/10.1148/radiol.2020200330.

[18] Ng, M. Y., Lee, E. Y., Yang, J., Yang, F., Li, X., Wang, H., ... & Kuo, M. D. (2020) Imaging profile of the COVID-19 infection: radiologic findings and literature review, *Radiology: Cardiothoracic Imaging*, 2(1), e200034. https://doi.org/10.1148/ryct.2020200034.

[19] Tian, X., Wang, J., Du, D., Li, S., Han, C., Zhu, G., ... & Lei, M. (2020) Medical imaging and diagnosis of subpatellar vertebrae based on improved Laplacian image enhancement algorithm, *Computer Methods and Programs in Biomedicine*, 187, 105082. https://doi.org/10.1016/j.cmpb.2019.105082.

[20] Solomon, M. D., McNulty, E. J., Rana, J. S., Leong, T. K., Lee, C., Sung, S. H., ... & Go, A. S. (2020) The Covid-19 pandemic and the incidence of acute myocardial infarction, *New England Journal of Medicine*, 383(7), 691–693. https://doi.org/10.1056/NEJMc2015630.

[21] Daniel, S. J. (2020) Education and the COVID-19 pandemic, *Prospects*, 49(1), 91–96. https://doi.org/10.1007/s11125-020-09464-3.

[22] Spinelli, A., & Pellino, G. (2020) COVID-19 pandemic: perspectives on an unfolding crisis, *Journal of British Surgery*, 107(7), 785–787. https://doi.org/10.1002/bjs.11627.

[23] Saliha, M., Ali, B., & Rachid, S. (2019) Towards large-scale face-based race classification on spark framework, *Multimedia Tools and Applications*, 78(18), 26729–26746. https://doi.org/10.1007/s11042-018-7080-0.

[24] Mezzoudj, S., Behloul, A., Seghir, R., & Saadna, Y. (2021) A parallel content-based image retrieval system using spark and tachyon frameworks, *Journal of King Saud University-Computer and Information Sciences*, 33(2), 141–149. https://doi.org/10.1016/j.jksuci.2018.03.010.

[25] Narin, A., Kaya, C., & Pamuk, Z. (2021) Automatic detection of coronavirus disease (COVID-19) using x-ray images and deep convolutional neural networks, *Pattern Analysis and Applications*, 24(6), 1207–1220. https://doi.org/10.1007/s10044-021-00984-y.

[26] Viceconti, M., Hunter, P., & Hose, R. (2015) Big data, big knowledge: big data for personalized healthcare, *IEEE Journal of Biomedical and Health Informatics*, 19(4), 1209–1215. https://doi.org/10.1109/JBHI.2015.2406883.

[27] 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(SI), S117–S129.

[28] Gupta, V. K., Gupta, A., Kumar, D., & Sardana, A. (2021) Prediction of COVID-19 confirmed, death, and cured cases in India using random forest model, *Big Data Mining and Analytics*, 4(2), 116–123. https://doi.org/10.26599/BDMA.2020.9020012.

[29] Wang, L., Lin, Z. Q., & Wong, A. (2020) COVID-Net: A tailored deep convolutional neural network design for detection of COVID-19 cases from chest x-ray images, *Scientific Reports*, 10(1), 19549. https://doi.org/10.1038/s41598-020-76550-z.

[30] Agbehadji, I. E., Awuzie, B. O., Ngowi, A. B., & Millham, R. C. (2020) Review of big data analytics, artificial intelligence and nature-inspired computing models towards accurate detection of COVID-19 pandemic cases and contact tracing, *International Journal of Environmental Research and Public Health*, 17(15), 5330. https://doi.org/10.3390/ijerph17155330.

[31] Basu, S., Mitra, S., & Saha, N. (2020) Deep learning for screening COVID-19 using chest x-ray images, In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, IEEE, pp. 2521–2527. https://doi.org/10.48550/arXiv.2004.10507

[32] Hemdan, E. E. D., Shouman, M. A., and Karar, M. E. (2020). COVIDX-Net: A Framework of Deep Learning Classifiers to Diagnose COVID-19 in X-Ray Images. *arXiv preprint arXiv:2003.11055*. https://doi.org/10.48550/arXiv.2003.11055

[33] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*, 25, 1097–1105. https://doi.org/10.1145/3065386

[34] Dean, J., and Ghemawat, S. (2008). MapReduce: Simplified Data Processing on Large Clusters. *Communications of the ACM*, 51(1), 107–113. https://doi.org/10.1145/1327452.1327492

[35] Duran-Lopez, L., Dominguez-Morales, J. P., Corral-Jaime, J., Vicente-Diaz, S., and Linares-Barranco, A. (2020). COVID-XNet: A Custom Deep Learning System to Diagnose and Locate COVID-19 in Chest X-Ray Images. *Applied Sciences*, 10(16), 5683. https://doi.org/10.3390/app10165683

[36] Lopez-Cabrera, J. D., Orozco-Morales, R., Portal-Diaz, J. A., et al. (2021). Current Limitations to Identify COVID-19 Using Artificial Intelligence with Chest X-Ray Imaging. *Health and Technology*, 11(2), 411–424. https://doi.org/10.1007/s12553-021-00609-8

[37] Karau, H., Konwinski, A., Wendell, P., and Zaharia, M. (2015). *Learning Spark: Lightning-Fast Big Data Analysis*. O'Reilly Media, Inc.

[38] Xu, X., Tian, H., Zhang, X., Qi, L., He, Q., and Dou, W. (2022). DisCOV: Distributed COVID-19 Detection on X-Ray Images with Edge-Cloud Collaboration. *IEEE Transactions on Services Computing*, 15(3), 1206–1219. https://doi: 10.1109/TSC.2022.3142265

[39] Saadna, Y., Behloul, A., and Mezzoudj, S. (2019). Speed Limit Sign Detection and Recognition System Using SVM and MNIST Datasets. *Neural Computing and Applications*, 31(9), 5005–5015. https://doi.org/10.1007/s00521-018-03994-w

[40] Mezzoudj, S. (2020). Towards Large Scale Image Retrieval System Using Parallel Frameworks. In *Multimedia Information Retrieval*. IntechOpen. https://doi.org/ 10.5772/intechopen.94910

[41] Mezzoudj, S., and Melkemi, K. E. (2021). A Hybrid Approach for Shape Retrieval Using Genetic Algorithms and Approximate Distance. In *Research Anthology on Multi-Industry Uses of Genetic Programming and Algorithms* (pp. 205–222). IGI Global. https://doi.org/10.4018/IJCVIP.2018010105

[42] Borthakur, D. (2008). HDFS Architecture Guide. *Hadoop Apache Project*, 53(1-13), 2.

[43] Tartaglione, E., Barbano, A., Berzovini, C., Calandri, M., and Grangetto, M. (2020). Unveiling COVID-19 from Chest X-Ray with Deep Learning: A Hurdles Race with Small Data. *arXiv preprint arXiv:2004.05405*. https://doi.org/10.3390/ijerph17186933

[44] Afshar, P., Heidarian, S., Naderkhani, F., Oikonomou, A., Plataniotis, K. N., and Mohammadi, A. (2020). COVID-CAPS: A Capsule Network-Based Framework for Identification of COVID-19 Cases from X-Ray Images. http://dx.doi.org/10.1016/j.patrec.2020.09.010

[45] Vaid, S., Kalantar, R., and Bhandari, M. (2020). Deep Learning COVID-19 Detection Bias: Accuracy Through Artificial Intelligence. *International Orthopaedics*. https://doi.org/10.1007/s00264-020-04609-7

[46] Waheed, A., Goyal, M., Gupta, D., et al. (2020). COVIDGAN: Data Augmentation Using Auxiliary Classifier GAN for Improved COVID-19 Detection. *IEEE Access*, 8, 91916–91923. https://doi.org/10.1109/ACCESS.2020.2994762

[47] Ismael, A. M., and Sengur, A.: Deep Learning Approaches for COVID-19 Detection Based on Chest X-ray Images, *Expert Systems with Applications*, 114054 (2020). https://doi.org/10.1016/j.eswa.2020.114054

[48] Dhiman, G., Chang, V., Kant Singh, K., and Shankar, A.: Adopt: Automatic deep learning and optimization-based approach for detection of novel coronavirus COVID-19 disease using X-ray images, *Journal of Biomolecular Structure and Dynamics*, 1–13 (2021). https://doi.org/10.1080/07391102.2021.1875049

[49] Nassima Dif, Zakaria Elberrichi (2023) An Efficient Transferred Cascade System for COVID-19 Detection, *Informatica*, pp. nn–mm. https://doi.org/10.31449/inf.v48i4.5923

[50] Vito Janko, Gašper Slapničar, Erik Dovgan, Nina Reščič, Tine Kolenik, Martin Gjoreski, Maj Smerkol, Matjaž Gams, and Mitja Luštrek (2021) Machine Learning for Analyzing Non-Countermeasure Factors Affecting Early Spread of COVID-19, *International Journal of Environmental Research and Public Health*, MDPI, 18(13), 6750. https://doi.org/10.3390/ijerph18136750

[51] Jasim Mohammed Dahr, Alaa Sahl Gaafar (2023) Performance Evaluation of Convolutional Neural Networks for COVID-19 Detection, *Informatica*, pp. nn–mm. https://doi.org/10.31449/inf.v48i21.6568

[52] Ruaa Sadoon, Adala Chaid (2023) Classification of Pulmonary Diseases Using a Deep Learning Approach, *Informatica*, pp. nn–mm. http://dx.doi.org/10.31449/inf.v48i14.6145

[53] Sarp, S., Catak, F. O., Kuzlu, M., Cali, U., Kusetogullari, H., Zhao, Y., and Guler, O.: An XAI approach for COVID-19 detection using transfer learning with X-ray images, *Heliyon*, 9(4), e14547 (2023). http://dx.doi.org/10.1016/j.heliyon.2023.e15137

[54] Kaleem, S., Sohail, A., Tariq, M. U., Babar, M., and Qureshi, B.: Ensemble learning for multiclass COVID-19 detection from big data, *Computers, Materials Continua*, 18(10), 10507–10520 (2023). http://dx.doi.org/10.1371/journal.pone.0292587

[55] Salau, A. O., and Jain, S.: Adaptive diagnostic machine learning technique for classification of cell decisions for AKT protein, *Informatics in Medicine Unlocked*, 23, 100511 (2021). http://dx.doi.org/10.1016/j.imu.2021.100511

[56] S. Mezzoudj, M. Khelifa, Y. Saadna (2024). A Comparative Study of Parallel Processing, Distributed Storage Techniques, and Technologies: A Survey on Big Data Analytics, *International Journal*, 10(5), pp. 86–99. https://doi.org/10.11648/j.ijdsa.20241005.11