A novel distributed deep learning approach for large-scale chest X-ray covid-19 images detection

Saliha MEZZOUDJ (saliha.mezzoudj@yahoo.com)
University of Algiers Benyoucef Benkhedda

Imane BELKESSA
University of Algiers Benyoucef Benkhedda

Feriel BOURAS
University of Algiers Benyoucef Benkhedda

Meriem KHELIFA
University of Ouargla

Research Article

Keywords: covid-19 image classification, chest X-ray covid-19 images, COVDCNN, DCNN, Spark TensorFlow Distributor, barrier mode

Posted Date: February 7th, 2023

DOI: https://doi.org/10.21203/rs.3.rs-2534755/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License. Read Full License

Additional Declarations: No competing interests reported.
A novel distributed deep learning approach for large-scale chest X-ray covid-19 images detection

Mezzoudj Saliha · Belkessa Imane · Bouras Feriel · Khelifa Meriem

Abstract Recently, the high mortality rates caused by the covid-19 pandemic have formed a crucial problem to support its enormous and rapid spread. Due to the need for accurate and rapid diagnosis of patients, it is necessary to develop a self-operating recognition model as a fast recognition system to detect covid-19 infection and prevent the rapid spread among people. Chest X-ray images have been demonstrated to be useful in detecting the covid-19 cases. Furthermore, covid-19 being a new disease, the annotated large-scale CXR dataset (big data) for this particular disease is virtually nonexistent. We propose exploiting a large-scale chest X-ray dataset and training a distributed deep neural network to address the limited data issue. In this paper, we propose a new distributed deep learning-based approach (COV-DCNN) for detecting large-scale chest X-ray covid-19 images using ImageDataGenerator to do dynamic data preprocessing and splitting and spark TensorFlow distributor framework to execute distributed train-
ing tasks using a Spark job in barrier mode. Furthermore, we adopted a distributed Deep Convolutional Neural Network (DCNN), ANN, AlexNet on Spark TensorFlow Distributor to classify large-scale chest X-ray covid-19 images. The highest accuracy was obtained by the DCNN model, which is 91.88% during 0.006s for each case, 84.62% for the ANN model, and 73.8% for the AlexNet model. The obtained results achieve high classification accuracy at a faster speed than other state-of-the-art classification methods.

**Keywords** covid-19 image classification · chest X-ray covid-19 images · COV-DCNN · DCNN · Spark TensorFlow Distributor · barrier mode

1 Introduction

Recently, several diseases have affected the whole world. Therefore, research in the field of pulmonology has great importance in public health studies and recently mainly focuses on covid-19 [Holanda et al., 2010]. The early detection of this latter and the accurate separation of non-covid-19 cases at least cost and in the early stages of the disease are among the main challenges of the current Coovid-19 pandemic. Regarding the novelty of the disease, diagnostic methods based on radiological images need more precision despite their numerous applications in diagnostic centers [Ghaderzadeh and Asadi, 2021].

A huge amount of data must be processed in all health applications. Data type, data size, security, and other characteristics are more important in the data processing. The term "big data" refers to data with certain characteristics: volume, speed, value, integrity, and variability. This big data needs to be stored, processed, and analyzed for the desired results. Medical data is more complex when it comes to predicting outputs, which will have more significance for patient treatment. Due to their importance, there is a need to develop efficient and better algorithms, techniques, and tools to analyze big medical data [Shyni and Chitra, 2022]. Generally, a typical covid-19 image classification system starts with a pre-processing step to normalize the input image. Then, an indexing phase is applied to extract discriminant features from covid-19 images. Finally, a decision mechanism is performed by a robust classifier to classify the covid-19 image of the given person. The output result is presented as covid-19 labels, such as covid/non-covid. For effective covid-19 cases detection and classification, machine learning methods are widely used in the literature. Zhang et al. in [Yao et al., 2020] applied Support Vector Machine (SVM) model. The clinical information and blood/urine test data were used to validate the method’s performance. The experimental results achieved 81.4%. However, the deep learning methods are considered as the most used methods to detect covid-19 [Panahi et al., 2021]. Furthermore, the conventional covid-19 classification systems have focused on small databases of covid-19 images. Therefore, it is important to generalize and train these systems on large-scale databases [Shyni and Chitra, 2022], [Benbrahim et al., 2020]. The key challenge is to design a computationally efficient and accurate covid-19 classification system on large-scale covid-19 databases [Shyni and Chitra, 2022], [Benbrahim et al., 2020]. Accordingly, this paper presents the problem of covid-19 classification on large-scale covid-19 databases by proposing a new covid-19 classification approach (COV-DCNN) on the Spark TensorFlow Distributor framework. Our main contributions
are:
(1) The parallel processing of large-scale chest X-ray covid-19 images using a new technology called Spark TensorFlow Distributor framework.

(2) The combination of the two technologies, deep learning and Spark TensorFlow Distributor framework, allows data analysts to classify images with high efficiency and permit them to create models that can run on clusters quickly.

(3) The proposed COV-DCNN approach is intended to classify large volumes of chest X-ray covid-19 images speedily and efficiently. COV-DCNN achieves excellent training time without compromising classification accuracy.

The remainder of this paper is organized as follows: Section 2 describes the related works in covid-19 detection and classification, then, in Section 3 we present the background and tools used in this work. Section 4 describes our proposed method, section 5 discusses the experimental study we have performed. Finally, conclusion is given in Section 6.

Fig. 1 Sample X-ray images of Normal cases and COVID-19 positive cases

2 Related works

Recently, research on covid-19 recognition has widely emerged. Accordingly, covid-19 recognition is processed in three ways [Alaif et al., 2021]: (i) The first one employs machine learning (ML) methods for covid-19 recognition. (ii) The second focuses on deep learning (DL) methods. (iii) The third way is covid-19 recognition based on combining machine and deep learning methods.

This paper considers the deep learning methods for covid-19 recognition, which have made disease detection and classification in medical imaging more powerful and automated [Ahemad et al., 2022]. Many covid-19 recognition works have been
proposed in the literature: In this section, we briefly highlight some research related to covid-19 detection using DL and X-ray images. We start with a capsule network-based framework called COVID-CAPS for diagnosing covid-19 from CXR images [Afshar et al., 2020]. It consists of several capsule and convolutional layers. Results based on CXR image datasets show that COVID-CAPS has an advantage over CNN-based models with an accuracy of 95.7.

To improve the detection performance of covid-19 cases, the authors [Tang et al., 2021] have proposed a model that uses CXR images as a learning set for the detection of covid-19 cases; this model is generated by combining multiple snapshot models from COVID-Net. The outcomes confirmed the superiority accuracy in pairs of the training and testing grade of the EDL-COVID model compared with COVID-Net.

In [Ahsan et al., 2021], the authors proposed and tested six modified deep learning models: VGG16, InceptionResNetV2, ResNet50, MobileNetV2, ResNet101, and VGG19 to detect covid-19 cases from CXR images. They achieved as a result that the modified VGG16 and MobileNetV2 models can distinguish patients from covid-19 symptoms with near 99% accuracy. In another study [Sethi et al., 2020], the authors proposed and tested six modified deep learning models: VGG16, InceptionResNetV2, ResNet50, MobileNetV2, ResNet101, and VGG19 to detect covid-19 cases from CXR images. They achieved that the modified VGG16 and MobileNetV2 models can distinguish patients from covid-19 symptoms with nearly 99.

Another work is based on a large dataset that is proposed by the authors in [Wang et al., 2020]; they used COVID-Net as a CNN model to detect covid-19 cases drawn from 13,975 CXR images to help clinicians improve screening. The authors augmented the data using the following methods: translation, rotation, horizontal flip, zoom, and intensity shift. The recognition accuracy achieved was 93.38.

3 Materials and methods

Recently, the combination of deep learning and Big Data technologies can confront many challenges and achieve very high performances. In general, Using deep learning with Apache Spark technologies can overcome many challenges in image classification and help to create models running in a distributed way on clusters [Benbrahim et al., 2020]. This section introduces some methods and tools used in our proposed approach.

3.1 Big data

Big Data is a very extreme amount of data in different forms (text, data, sound, video, etc.) and formats, which are unstructured, structured, and semi-structured data. This extreme amount of data led to traditional processes and storage failure to confront it, for thus, big data introduces new methods for its process and storage. [Mezzoudj, 2020]
3.2 Apache Spark

Apache Spark is an open-source framework focused on the MapReduce programming model to hold a large amount of data (big data) by processing them in a distributed way on clusters [Mezzoudj, 2020].

3.3 Spark TensorFlow Distributor:

In our work, we use Spark TensorFlow Distributor, which is a distributed TensorFlow2.x on Apache Spark 3.x [Blog, 2020], [pypi, 2021]. This latter was carried out to process a large amount of covid-19 images (Big data) using massively parallel architectures, improving the classification performance of CXR covid-19 images, and ensuring the rapidity of our proposed system. Indeed, Spark TensorFlow Distributor is a python library that implements the barrier execution mode of spark to implement distributed TensorFlow training on top of the spark 3.0 cluster. The first spark versions were limited to distributed training because of the traditional execution mode using Map/Reduce; this latter runs jobs in each worker without communication between workers; thus, Spark 3.0 uses a new execution mode named barrier execution, which allows us to easily train deep learning models in a distributed way, that allows communication between workers during the execution. Spark TensorFlow distributor an open-source native package that distributes training steps with TensorFlow on their Spark clusters [pypi, 2021], [databricks, 2020].

3.4 Deep learning (DL)

It is a specific type of machine learning, mainly based on using specific types of artificial neural network methods with many layers and nodes. [Beyer et al., 2000].

3.5 chest X-ray:

Chest X-ray is one common medical imaging modality used to diagnose and analyze the severity of an infection; it has many advantages. These are as follows:
- X-ray imaging is the most often utilized medical imaging tool for the diagnosis of covid-19 due to its large availability [Duran-Lopez et al., 2020].
- It can be processed by easy methods and thus reduces the imaging time which decreases the probability of spreading the covid-19 cases [López-Cabrera et al., 2021].
- It is economical when compared with other medical imaging modalities.
- It generates a low radiation dose when compared to a CT scan.

Although the advantages are cited above, X-rays are less sensitive, which may lead to a false prediction of the disease with early symptoms. [Hemdan et al., 2020].

4 COV-DCNN: distributed DCNN for covid-19 image recognition

We propose a new system for large-scale covid-19 images classification (Fig. 2). The proposed system uses the Spark TensorFlow Distributor to process the deep
learning method in distributed and parallel way using big dataset, which performs in-memory computations, which uses a master/slave architecture that includes two basic nodes, that are master node (pilot process) and slave node (slave process). The input of the proposed system is a massive volume of covid-19 images, and the output is covid-19 or no covid-19 images.

Indeed, we propose a distributed DCNN-based deep learning method for big covid-19 image recognition. The distributed DCNN model is deployed on the top of the Apache Spark cluster. The yaran cluster manager handles scheduling and resource allocation across applications. The cluster manager controls the cost of parallel data training times to reach workload balancing. Our proposed approach is divided into two main steps, which are mentioned in Fig. 3

**Step 1: Data preparation and preprocessing for COV-DCNN:** which concerns the preparation and processing of big data, in this step we split data across the HDFS [Mezzoudj, 2020] into many parts and process the large chest X-ray images. Furthermore, we use the dynamic data allocation using dataimagegenerator with batch size and shuffle equal to true as parameters, this produce preprocessing images dynamically during the training of our model, it does
**Fig. 3** Proposed approach on Spark framework.
all this with better memory management so that we can train a huge dataset [machine learning, 2020].

Step 2: Big covid-19 image classification: In this step, we have employed deep Learning on Apache Spark to permit fast deep learning training and classification tasks, and also we have used the DCNN, ANN, and AlexNet architectures to classify large chest X-ray covid-19 images, in two kinds of images (covid-19 and non-covid-19).

4.1 Step 1: Dataset preparation and preprocessing for COV-DCNN

The goal of this step is to prepare and process the training dataset for the COV-DCNN module. To train the DL model, We have to prepare a set of labeled images.

4.1.1 Data preparation

The process starts by reading big covid-19 images. After loading the heterogeneous covid-19 images from HDFS [Mezzoudj, 2020]. These loaded images are divided into many equal chunks of images. A set of spark workers processes individual images. Then every chunk will be duplicated in an HDFS data node of our master–slave architecture. We store the enormous data in a distributed computer architecture, which allows to duplicate all data on all workers.

4.1.2 Dataset preprocessing

Each spark worker apply a parallel preprocessing on its covid-19 images chunk using keras ImageDataGenerator, this latter it able to produce real-time image preprocessing by loading the image dataset in memory and generates batches of preprocessing data, which means it can generate preprocessing images dynamically during the training of our model, it does all this with better memory management so that we can train a huge dataset. Indeed, numerous pre-processing methods are used in medical imaging applications to improve the quality of medical images. The most important methods which are used to enhance X-rays and CT scans of COVID-19 images are image resizing, image enhancement, and image segmentation [Shyni and Chitra, 2022]. To process our x-rays covid-19 images, we use image resizing, which is necessary to standardize our dataset. In a parallel way, we transform all the images in the dataset to a fixed dimension (224, 224) using the ImageDataGenerator technique for better recognition performance of the DCNN model. Furthermore, we use batch size=35 using the ImageDataGenerator technique, which generates tensor image data batches.

4.2 Step 2: Big covid-19 image classification

In this step we built and compile our DCNN model on spark-tensorflow-distributor.

In fact, Deep Convolutional Neural Networks (DCNN) have achieved outstanding performance breakthroughs in many pattern recognition tasks such as image classification (M. Liu et al., 2016). The modern DCNN is built using alternating
convolution layers (CL) and max-pooling (MP) layers followed by a number of fully connected layers (FC) [Krizhevsky et al., 2017]. During training the input to our DCNN is a 128 × 128 images passed through a 32-character CL and a filter of size (3*3) and a MP of size (2*2). For the second layer, a CL of 32 characters and a size filter (3*3), and a size MP (2*2). Finally, for the fully connected layer (FC), the first layer with 100 character output and the second FC has a single output. We use a RELU activation function for all layers except the last, the function is sigmoid. We use a rectified linear units activation function (ReLUs) for all layers [Krizhevsky et al., 2017], a linear activation function for the MP layers, and a softmax activation function for the last layer (output layer). Fig. 4 depicts the CNN Architecture for covid-19 Image classification. The detailed architecture of the used DCNN is summarized in Fig. 3. The main key of our proposed system is the using the spark-tensorflow-distributor, this package requires spark3.0.1, Python 3.6+, and tensorflow2.1.0+ to run. After building and compiling our DCNN, we use Spark-tensorflow-distributor, that is an open-source native package in TensorFlow that helps users do distributed training with TensorFlow on their Spark clusters. It is built on top of tensorflow.distribute.Strategy, which is one of the major features in TensorFlow 2.

Spark2.x. vs Spark3.x (which used in our approach):
Spark 2.x feature’s:
- It execute only Map/Reduce based job.
- A spark program is a set of map and reduce stages.
- Map/Reduce in spark inspired by Hadoop Map/Reduce.
- This works great for many big data workloads like ETL, SQL, Normal Machine Learning, etc.

However, this kind of scheduling is inefficient for implementing deep learning frameworks. In order to run a distributed deep learning in Spark, spark3.x implements a new execution mode called barrier execution mode ”MirroredStrategyRunner” which is different than standard Map/Reduce model.

Indeed, Map/Reduce, all tasks in a stage are independent of each other and they

1. begin
2. from spark_tensorflow_distributor import MirroredStrategyRunner
3. import os
4. from keras.preprocessing.image import ImageDataGenerator
5. launch spark session
6. os.environ["SPARK_HOME"] = "/opt/spark"
7. from pyspark import SparkConf, SparkContext
8. conf = SparkConf().setMaster("local").setAppName("pyspark")
9. Generate allocation dynamic
10. from pyspark import SparkConf
11. from pyspark import SparkContext
12. conf = SparkConf().setAppName("Spark dynamic allocation").
13. set("spark.shuffle.service.enabled", "true").
14. set("spark.dynamicAllocation.initialExecutors", "10").
15. set("spark.dynamicAllocation.executorIdleTimeout", "30s").
16. set("spark.executor.cores", "4").
17. set("spark.executor.memory", "1000m")
18. sc = SparkContext.getOrCreate(conf)
19. spark = SparkSession.builder.config(conf=conf)
20. /*Pre-processing step*/
21. create a new generator
22. imagegen = ImageDataGenerator()
23. read image with generator
24. var testing = sc.textFile("hdfs://localhost:9000/project/testimages")
25. test = imagegen.flow_from_directory("testing", class_mode = "binary", shuffle = False, batch_size = 35, target_size = (224, 224))
26. var training = sc.textFile("hdfs://localhost:9000/project/trainimages")
27. train = imagegen.flow_from_directory("training", class_mode = "binary", shuffle = False, batch_size = 35, target_size = (224, 224))
28. Building the DCNN model
29. cnn.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
30. model.fit(train, epochs=50, batch_size=16) runner = MirroredStrategyRunner(num_slots=1, local_mode=True, use_gpu=USE_GPU)
31. runner.run(train)
32. End

don’t communicate to each other. If one of the task fails, only that task will be retried. But in Barrier execution mode “MirroredStrategyRunner” all tasks in a stage will be started together and if one of the task fails whole stage will be retried again. All those tasks can communicate to each other [pypi, 2021], [databricks, 2020]. Algorithm 1 shows the pseudo-code of COV-DCNN: distributed DCNN for covid-19 image recognition. In the following algorithm 1, we describe the most significant instructions, enumerated from (1 to 36). In the bellow code, we import MirroredStrategyRunner from spark tensorflow distributor library, which implements barrier execution mode.
5 Experimental results

5.1 Environment description

To evaluate the performance of our method COV-DCNN on a large-scale image dataset, we use a CXR-type image base [Kaggle, ]. The dataset used contains 13808 CXR images to train and test our approach. Then the images were grouped into sick (covid-19) and not sick groups. In this environment, we use the Apache Spark framework. Here we adopt a single node cluster (standalone), the specifications of this cluster are as follows:

- **Software Environment**: Ubuntu 18.04 LTS, Spark 3.0.1, Spark MLlib, keras, python 7.13, and tensorflowOnSpark 2.7.
- **Environmental hardware**: Intel Core i5-42104 CPU speed 1.70 GHZ*4, 8 GB of RAM.

5.2 Presentation of the databases

The dataset [Kaggle, ] used in our work concerns CXR images of positive COVID-19 cases as well as images of normal pneumonia, the image base collection and classification are made by a team of researchers from the University of Qatar, Bangladesh, Pakistan and Malaysia in collaboration with doctors, researchers have done several updates (increase: add numbers of CXR images) to the base [Kaggle, ], This latter contains 3,616 positive cases of Covid-19, as well as 10,192 normal cases. Therefore, the database contains 13,808 CXR images, the latter are of the PNG type, having 299 pixels in width and 299 pixels in height. We divide the data into percentage of 80% for training and 20% for test. and also we performed AlexNet, DCNN, ANN algorithms.

5.3 Results of the proposed approach

<table>
<thead>
<tr>
<th>Batch size</th>
<th>Accuracy of drive (%)</th>
<th>Test accuracy (%)</th>
<th>Test epoch time</th>
<th>Training time</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>99.79</td>
<td><strong>84.62</strong></td>
<td><strong>53s</strong></td>
<td><strong>49 min</strong></td>
</tr>
<tr>
<td>32</td>
<td>98.80</td>
<td>50</td>
<td>42s</td>
<td>48 min 23s</td>
</tr>
<tr>
<td>100</td>
<td>99.84</td>
<td>64.89</td>
<td>35s</td>
<td>47 min 25s</td>
</tr>
</tbody>
</table>

Table 1 The effect of batch on the ANN model with 13808 CXR dataset.

The table 1 summarizes the accuracy of the ANN model test and training time on our database with a drop of 0.25. We notice that the best classification is obtained with a batch size of 16 (**84.62%**); the training lasted 49 min. With an increase in batch size (32), the model accuracy decreased, with a training time of 48min23s. Using a batch size of 100, the model reached an accuracy of 64.89%, and the ANN took less time to train (47 min 25 sec). The increase of the batch size from 16 to 100 led to an acceleration of the training, this saving learning time.
Table 2 Study of the effect of epochs on the ANN model with 13808 CXR.

<table>
<thead>
<tr>
<th>Epochs</th>
<th>Training accuracy (%)</th>
<th>Test accuracy (%)</th>
<th>Time of epoch test (s)</th>
<th>Training time</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>96.89</td>
<td>67.45</td>
<td>55sec</td>
<td>16min12sec</td>
</tr>
<tr>
<td>4</td>
<td>98.59</td>
<td><strong>84.46</strong></td>
<td>54sec</td>
<td>32min5sec</td>
</tr>
<tr>
<td>6</td>
<td>99.79</td>
<td><strong>84.62</strong></td>
<td><strong>53sec</strong></td>
<td>48min</td>
</tr>
<tr>
<td>8</td>
<td>100</td>
<td>75.87</td>
<td>59sec</td>
<td>1h3min52sec</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>72.60</td>
<td>56sec</td>
<td>1h19min39sec</td>
</tr>
<tr>
<td>14</td>
<td>100</td>
<td>70.13</td>
<td>55sec</td>
<td>1h51min33sec</td>
</tr>
<tr>
<td>18</td>
<td>100</td>
<td>72.29</td>
<td>54sec</td>
<td>2h28min5sec</td>
</tr>
<tr>
<td>20</td>
<td>100</td>
<td>70.65</td>
<td>55sec</td>
<td>2h45min5sec</td>
</tr>
</tbody>
</table>

The table 2 shows the comparisons of training and test times, and the accuracy of the ANN model with different numbers of epochs and a dropout rate of 0.25. We notice that the number of epochs has no impact on the accuracy rate of the ANN, and has an impact on the training time of this latter, where the increase of the number of epochs increases the time of our DL method.

Table 3 and table 4 show performance results (accuracy and time), where table 3 is built using the DCNN model with a varied dropout using 32 patch, and for the table 4 is made with a DCNN without dropout by changing the number of epochs with 32 patch. We studied the effect of dropout rates on the performance of the model. The table 3 summarizes the precision performance results. We notice that the best classification is obtained by using a dropout with rates of 0.2 and 0.5 in the convolutional layer and the fully connected layer, respectively.

Table 3 Study of the dropout effect on the DCNN model with 13808 CXR.

<table>
<thead>
<tr>
<th>Dropout in CL</th>
<th>Drop out in FC</th>
<th>Test Accuracy (%)</th>
<th>test time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.2</td>
<td><strong>81.30</strong></td>
<td>58</td>
</tr>
<tr>
<td>0</td>
<td>0.5</td>
<td>73.23</td>
<td>54</td>
</tr>
<tr>
<td>0.2</td>
<td>0.1</td>
<td>77.43</td>
<td>56</td>
</tr>
<tr>
<td>0.2</td>
<td>0.5</td>
<td><strong>91.31</strong></td>
<td><strong>52</strong></td>
</tr>
<tr>
<td>0.5</td>
<td>0.2</td>
<td>60.17</td>
<td>50</td>
</tr>
<tr>
<td>0.5</td>
<td>0.4</td>
<td>54.43</td>
<td>56</td>
</tr>
</tbody>
</table>

Table 4 Study of the effect of epochs on the DCNN model with 13808 CXR.
Table 4 Study of the effect of epochs on the DCNN model with 13808 CXR.

<table>
<thead>
<tr>
<th>Number of epochs</th>
<th>Training accuracy (%)</th>
<th>Training time</th>
<th>Testing accuracy (%)</th>
<th>Testing time</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>88.79</td>
<td>4 min 4 sec</td>
<td>60.43</td>
<td>51 sec</td>
</tr>
<tr>
<td>4</td>
<td>95.89</td>
<td>8 min 15 sec</td>
<td>61.88</td>
<td>49 sec</td>
</tr>
<tr>
<td>8</td>
<td>70.10</td>
<td>16 min 33 sec</td>
<td>70.41</td>
<td>50 sec</td>
</tr>
<tr>
<td>18</td>
<td>100</td>
<td>43 min 48 sec</td>
<td>69.73</td>
<td>51 sec</td>
</tr>
<tr>
<td>25</td>
<td>100</td>
<td>58 min 56 sec</td>
<td>70.73</td>
<td>52 sec</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>5h 28 min 50 sec</td>
<td>91.88</td>
<td>550 sec</td>
</tr>
</tbody>
</table>

The table 4 presents the influence of increasing the number of epochs on the training of a DCNN. We notice that the number of epochs has an impact on the precision and the training time. The more the number of epochs is increased, the precision increases with an improvement in training time. We notice that the best classification is obtained by using 100 epochs; the DCNN reached an accuracy of 91.88% in 5h 28min 50s.

Table 5 Étude de l’effet de batch avec un drop out de 0.4 sur le modèle AlexNet avec 13808 CXR.

<table>
<thead>
<tr>
<th>Batch</th>
<th>Training Accuracy (%)</th>
<th>Training time</th>
<th>Testing Accuracy (%)</th>
<th>Epoch time</th>
<th>loss function</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>73.82</td>
<td>1h 16min 45 sec</td>
<td>73.8</td>
<td>52 sec</td>
<td>0.575</td>
</tr>
<tr>
<td>60</td>
<td>73.82</td>
<td>1h 16min 50 sec</td>
<td>73.8</td>
<td>59 sec</td>
<td>0.57</td>
</tr>
<tr>
<td>128</td>
<td>68.78</td>
<td>1h 20min 45 sec</td>
<td>73.8</td>
<td>55 sec</td>
<td>0.60</td>
</tr>
<tr>
<td>300</td>
<td>73.50</td>
<td>1h 15min 56 sec</td>
<td>73.8</td>
<td>53 sec</td>
<td>0.60</td>
</tr>
<tr>
<td>500</td>
<td>73.79</td>
<td>1h 16min 50 sec</td>
<td>73.8</td>
<td>52 sec</td>
<td>0.60</td>
</tr>
</tbody>
</table>

The tables 5 and 6, show that the batch size and the dropout at the level of the FC layer did not influence either the precision or the training time with the AlexNet architecture.

The training time of the DCNN and ANN is depicted in Fig. 5 and Fig 6. In this experiment, a sets of tests are conducted to compare the training time of DCNN (Where the first training time is 5h 28min 50s=2880s for 100 epochs, recognition accuracy = 91.88%), and ANN (Where the first training time is 48min=2880s, recognition accuracy = 84.62% for 6 epochs) with batch size of 16. These experiments analyze the impact of varying the memory size for each executor (between 1024MO and 8592MO). For example, for an image with an amount of memory equal to or greater than 2048 MO, the training time of ANN and DCNN is decreased significantly, whereas, for an amount of memory less than 2048 MO, the training time is more important. Therefore, when the amount of memory allocated
The table 7 presents the effect of scaling on the performance of ANN and DCNN under Spark, using four different resolutions, which are 128x128, 224x224, and 256x256. The best accuracy obtained for image classification is 91.88.

5.4 Performance comparison of the proposed approach with some state-of-the-art models

Table 8 shows the recognition accuracy comparisons using some recent approaches with different sizes of training datasets. On the same dataset, we can notice that our proposed method achieves the highest recognition accuracy compared with [Sarker et al., 2020] method. This enhancement in the recognition accuracy is due
Fig. 6 Impact of amount memory Spark’s on the Training Time of ANN method (Where the first training time is 48min=2880s, recognition accuracy = 84.62% for 6 epochs) using memory size of 1024MO.

<table>
<thead>
<tr>
<th>Author</th>
<th>Approach</th>
<th>Dataset</th>
<th>Number of images</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Akter et al., 2021]</td>
<td>ResNet50</td>
<td>A chest X-ray database [Thibault, ]</td>
<td>52000</td>
<td>84</td>
</tr>
<tr>
<td>[Akter et al., 2021]</td>
<td>VGG16</td>
<td>A chest X-ray database [Thibault, ]</td>
<td>52000</td>
<td>91</td>
</tr>
<tr>
<td>[Akter et al., 2021]</td>
<td>AlexNet</td>
<td>A chest X-ray database [Thibault, ]</td>
<td>52000</td>
<td>91</td>
</tr>
<tr>
<td>[Sarker et al., 2020]</td>
<td>InceptionV3</td>
<td>A chest X-ray database [Thibault, ]</td>
<td>52000</td>
<td>94</td>
</tr>
<tr>
<td>Our approach</td>
<td>DCNN</td>
<td>A COVIDx dataset (the same database used in our approach)</td>
<td>13800</td>
<td>87</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[Kaggle, ]</td>
<td>13808</td>
<td>91.88</td>
</tr>
</tbody>
</table>

Table 8 Comparison between our proposed approach with 13808 CXR and related works.

to our DCNN model architecture, which improves the results of large-scale covid-19 classification.

6 Conclusion

We presented a new method for large-scale covid-19 image classification (COV-DCNN) using Spark TensorFlow Distributor. The proposed approach uses the DCNN model. Our approach has been evaluated on two covid-19 and non-covid-19 classes; it is tested on the X-ray covid-19 radiography dataset to classify and detect covid-19 positive cases. The experimental tests appear that our proposed system COV-DCNN, which uses the DCNN method, achieves the highest accuracy (91.88%) in 0.006s compared to the ALexNET and ANN and AlexNet methods. Another advantage of this proposed approach is its robustness against image scale changes. Additionally, our method using Spark TensorFlow Distributor on large-scale X-ray covid-19 images showed significant improvement in training and recognition time.

The combination of deep learning and Spark TensorFlow Distributor makes our approach environment more interesting, and we can build a model running on large computing clusters. The proposed system can be adopted as a supportive
decision-making system to help radiologists in clinics and hospitals to screen patients immediately.

7 Declarations:

-The authors have no relevant financial or non-financial interests to disclose. The authors have no competing interests to declare that are relevant to the content of this article.
-All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript.
-The authors have no financial or proprietary interests in any material discussed in this article.
-The authors declare that the data supporting the findings of this study are available within the article and its supplementary information files.

References


