ISSN 1870-4069

Convolutional Neural Network in a Pseudo-Distributed Environment for Classification of Chest X-Ray Images of Patients with Pneumonia

Alexandra K. Medrano Roldán, Julia P. Sánchez Solís, Vicente García Jiménez, Rogelio Florencia Juárez, Gilberto Rivera Zárate

Universidad Autónoma de Ciudad Juárez,
Mexico

al137706@alumnos.uacj.mx,{julia.sanchez, vicente.jimenez, rogelio.florencia, gilberto.rivera}@uacj.mx

Abstract. In recent years, there has been an increase in the volume of medical data, generating hundreds of terabytes (TB)/petabytes (PB) of data from different sources. This has led to the emergence of innovative technologies such as Apache Spark, which is a framework that allows the analysis of data in memory based on distributed processing. However, since it is a relatively new technology, both Spark and the other tools that have been developed as a complement, do not have orderly and updated documentation. In this project, a convolutional neural network was implemented in a pseudo-distributed environment for the automatic classification of chest X-Ray images of patients with pneumonia using the Dist-Keras library. Thus, it was possible to explore how the convolutional neural network behaves in Spark as the size of the database increases. While the time was showing an increase as the database grew, the accuracy, precision and sensitivity metrics showed a non-stable behavior.

Keywords: Spark, convolutional neural networks, Dist-Keras.

1 Introduction

This work addresses the behavior of a Convolutional Neural Network (CNN) in a pseudo-distributed environment as the size of the database analyzed increases. The pseudo-distributed environment was configured with Spark, while the database contains chest X-Ray images of patients with and without pneumonia.

In order to analyze the behavior of the CNN in the pseudo-distributed environment, ten data sets were created, from the original database, of different sizes starting with 10% and increasing from ten in ten to 100%. Likewise, to avoid the imbalance of the classes during the analysis, the number of images was reduced from 5,856 to 3,166, where in each data set, there were half of the images belonging to the class with pneumonia and the other half without pneumonia.

Afterwards, the images were normalized and transformed to be displayed in a csv file. The CNN was trained and evaluated with each data set for analysis, so the results correspond to the average of ten executions. Because of the necessary equipment to simulate a distributed environment was not available, the CNN was implemented in a single computer, that is, in a pseudo-distributed environment.

This paper is organized as follows. Section 2 presents a brief summary of some related works. Section 3 shows a description of CNNs. Section 4 presents the evaluation metrics used in this work. Sections 5 and 6 give a short description of Apache Spark and Dist-Keras, respectively. Section 7 shows the implemented method. Section 8 presents the results and discussions. Finally, Section 9 concludes this paper.

2 State of the Art

This section describes some works that deal with the analysis of data, particularly images in a distributed environment.

In [3], a set of medical applications within a Grid Infrastructure is described as well as the Medical Applications on a Grid Infrastructure Connection (MAGIC-5) project. In this work, the authors propose to improve the performance of disease detection programs, such as those in MAGIC-5, by allowing remote access to hundreds of data generated in different hospitals. The MAGIC-5 tool contains algorithms that allow the analysis of: mammograms, for the detection of breast cancer; Computed Tomography (CT) Scans, for the detection of lung cancer; and positron emission tomography, for the early diagnosis of Alzheimer.

In [14], an architecture, which encompasses the parallel image processing in the cloud, as well as a mass data processing engine in Hadoop, are presented. The algorithms that were used for the experiments were: Discrete Fourier Transform (DFT), face detection and template comparison. The results of the experiments showed that the design of this architecture can handle a huge number of large images and videos. However, there were problems regarding the distribution of information and response time.

In [15], a toolkit, called Kira, was built for the processing of astronomical images using Apache Spark and running on Amazon Elastic Compute Cloud (Amazon EC2). The authors reported that the experience with Kira showed that applications in the Apache Spark area are an alternative for multitasking scientific applications and that, specifically, Apache Spark can be easily integrated with existing libraries.

There are three main differences between the works mentioned above and the present project. First, in this work the analysis was applied to a database with images of patients with and without pneumonia. Second, the image classification method selected was a CNN. Finally, the third difference is that the algorithm was implemented in a pseudo-distributed environment, using Apache Spark.

3 Convolutional Neural Network

A CNN is a deep and feed-forward artificial neural network, where a hierarchical structure is maintained through the learning of representation of the internal characteristics. In addition to generalizing them in image problems, CNNs have also achieved results in problems related to natural language processing and speech recognition [10].

The main idea of CNNs is to devise a solution to reduce the number of parameters allowing a network to be deeper with fewer parameters [2]. The three most common types of layers in a CNN are: convolution layer, pooling and Rectified Linear Units (ReLU), as well as the fully connected layer [1].

In this project, the data was divided into two data sets, one for training and the other for testing. The CNN learned from with the training data set, and then its performance was evaluated with the testing data set.

4 Evaluation Metrics

The evaluation of the model is the most important step in the development of any machine learning solution since it determines if it is necessary to review the previous steps before continuing [12]. The metrics used in this project are described below:

- **Accuracy:** The accuracy of a classifier is the percentage of a set of test examples that were correctly classified [4]. It is defined as:

$$Accuracy = \frac{TP + TN}{P + N},$$

  where TP is the number of positive cases that are truly positive, and TN is the number of cases that the test marks as negative and they are truly negative. Meanwhile, P and N are the total positive and negative examples, respectively.

- **Sensitivity:** The sensitivity or true positive rate (TPR) is the probability of positive examples correctly classified [4]. It is defined as:

$$Sensitivity = \frac{TP}{TP + FN},$$

  where FN is the number of positive cases that were classified as negative.

- **Specificity:** The specificity or rate of true negatives (TNR) is the probability of correctly classified negative examples [4]. It is defined as:

$$Specificity = \frac{TN}{TN + FP},$$

where FP is the number of negative cases that the algorithm classified as positive.

- **Precision:** The precision is the result of dividing the true positive values (TP), among all the positive classifications (P') [8]. The formula is as follows:

$$Precision = \frac{TP}{TP + FP}.$$

5 Apache Spark

Apache Spark is a data analysis system in memory based on distributed processing. It is built on Hadoop and, therefore, uses Hadoop Distributed File System (HDFS) as a file system for data storage. However, there are differences between Hadoop and Spark. One of them is that Hadoop stores the data on disk to run the analysis, while Spark uses a cache mechanism in memory to store the data and process it [13].

6 Dist-Keras

Distributed Keras [7] is a distributed deep learning framework. It was developed based on Apache Spark and Keras [5], with the aim of significantly reduce training with distributed machine learning algorithms.

The distributed machine learning approach that follows is the parallel data paradigm, in which copies of the model are held in different trainer distributed on different nodes, although they may be on the same machine. In addition to that, the data set is divided in such a way that each replicated model can be trained in a different partition of the complete data set [6].

7 Method

In this section, the method used to carry out the project is presented.

7.1 Experimental Approach

In this project, the database with chest X-Ray images that were collected and labeled in [9] was used. This database has 5,856 images of chest X-Ray of children, where 4,273 were from patients with pneumonia, while 1,583 were from patients with a healthy diagnosis. Of the chest X-Ray images with pneumonia, 2,780 were due to bacteria and 1,493 were due to virus.

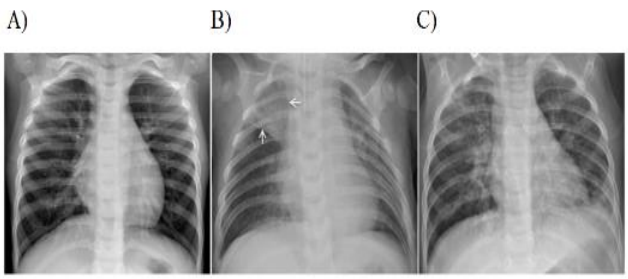

Fig. 1. A) Normal, B) Bacterial pneumonia, C) Viral pneumonia [9].

The chest X-Ray images were taken as part of the usual clinical care of the patients and reviewed to eliminate low quality or illegible scans, then they were qualified by experts. In Figure 1 there are three images of different patients belonging to the three categories of pneumonia: without, bacterial and viral.

7.2 Data Preparation

For the preparation of the data, first, because the images had different resolutions, it was decided to scale all the images to 100x100.

On the other hand, of the 5,856 images, 238 were in color, that is, they had the three RGB channels, while the rest had only one-color channel, in other words, the rest of the images were gray-scale. To normalize the images, those 238 that were in color were changed to gray-scale.

Then, in order to adapt the database to the image classification algorithm, each image was represented as a single vector within a csv file. In this file each row represents a different image. The first column of each row is the class to which the image belongs, while the rest of the columns correspond to the pixels. In the case of images with normal diagnosis, the label was 0, while, for the chest X-Ray images of patients with pneumonia, it was assigned the label 1. Because the final set of images had a resolution of 100x100 pixels, each image was transformed to a vector with 10,001 columns: one for the label and the remaining 10,000 for each of the pixels in the image.

For the classification, it was decided to consider only two classes: with normal and pneumonia diagnosis, grouping patients with pneumonia due to bacteria and viruses into a single class: pneumonia.

For the final preparation of the data set, the images in the folder val were integrated into the ones of test and train. Later, in order to avoid the imbalance of classes for each data set, the number of images per folder was reduced. The reduction was done randomly, where the number of images that would make up 100%, was defined according to the number of images in the class with the lowest presence in the database: without pneumonia or normal. Once the 100% data sets were defined, the other data set were obtained by percentages, as it is shown in the Table 1. In order to be able to use the images with the classifier, the code fragment for image preprocessing found in [6]

Table 1. Database division.

Percentage	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Test	48	96	144	192	241	289	337	385	433	484
Train	268	536	804	1072	1340	1608	1876	2144	2412	2682

Table 2. Confusion matrix for the classes with and without pneumonia.

		Predicted class		
		With pneumonia	Without pneumonia	Total
Actual Class	With pneumonia	TP	FN	P
	Without pneumonia	FP	TN	N
	Total	P’	N’	P + N

was used. In this code section, pixel values were normalized. Instead of having values between 0 and 255, they were normalized to range between 0 and 1.

7.3 Modeling

The image classification method selected for this work was a CNN, implemented in a pseudo-distributed environment using Apache Spark 2.4 together with Hadoop 2.7. The computer used has as operating system Ubuntu 18.04.2 LTS, has 12 GB of RAM and an Intel Core i7-2600K processor, with 3.40 GHz x 8. It was necessary to create a virtual environment in Anaconda installing Python 3.5.5.

The CNN was implemented based on the image processing example of the Modified National Institute of Standards and Technology (MNIST) database in [6], and in order to implement this network in a pseudo-distributed environment, the Dist-Keras library was used. Specifically, the distributed optimizer Asynchronous Elastic Averaging Stochastic Gradient Descent (AEASGD).

7.4 Evaluation

The metrics evaluated were time, accuracy, precision, sensitivity and specificity. Because the Dist-Keras library can only evaluate accuracy and time, the missing metrics: precision, sensitivity, and specificity were implemented in the library code. For the evaluation of the metrics the class with pneumonia was defined as positive, while the negative class was that of patients without pneumonia. With the above, the confusion matrix would be as shown in Table 2.

8 Results and Discussions

In this section, are shown the values of the metrics as well as their analysis. The metrics includes: time, accuracy, sensitivity and specificity, precision and, finally, the ROC Curve.

8.1 Time

When evaluating the training time of the CNN in the pseudo-distributed system, the overall results obtained from the ten executions per data set were averaged and are shown in Table 4, with their respective standard deviation.

When analyzing the results, in a matter of time, the CNN in Spark behaves as expected: the greater the size of data, the greater the training time. Table 3 shows the increase in time as the size of the database increases. The times obtained as the number

Table 3. Increase in time.

Percentage	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**100%**
Test	53.1	85.8	112.2	170.1	184.3	243.0	340.6	345.0	393.9	374.5
Train	-	536	804	1072	1340	1608	1876	2144	2412	2682

of images was increased, can be modeled with the function 1:

$$t(n) = 0.13n + 4.11\,, \tag{1}$$

where t is the time in seconds and n the number of images. The function was obtained through the method of least squares. The function indicates that time grows linearly as the number of images increases, meaning that a larger number of images would not imply large or exponential growth.

8.2 Accuracy

When evaluating the accuracy of the CNN after the training, the overall results obtained from the ten executions per data set were averaged and are shown in Table 4, with their respective standard deviation.

In the case of accuracy, not much variability is shown, in relation to the other metrics as the size of the database increases. The range is maintained between 0.45 and 0.55, or what would be between an interval of 45 to 55% accuracy on average. Indicating that, for the most part, it only classifies well 50% of the cases of the total. It is important to mention here that the highest values were obtained with the 10% data set, that is, the data set with the smallest size.

8.3 Sensitivity and Specificity

The average of values, as well as their standard deviation, obtained when evaluating the Sensitivity and Specificity of the CNN is shown in the Table 4.

For the sensitivity, values were obtained between 0.27 and 0.65, or what would be between an interval of 27 to 65% of sensitivity. There is not such a clear trend depending on the size of the database as there was with the metric accuracy. Specificity, a metric that has the same relationship as sensitivity, but with respect to the negative class, or in this case, with respect to the class to which patients without pneumonia belong, obtained values ranged from 0.31 to 0.67, or from 31 to 67%.

Table 4. Results.

Percentage	Time		Accuracy		Sensitivity		Specificity		Precision	
	M	SD	M	SD	M	SD	M	SD	M	SD
10%	53.05	0.774	0.550	0.151	0.554	0.370	0.546	0.450	0.709	0.266
20%	85.83	8.653	0.466	0.062	0.460	0.393	0.471	0.374	0.422	0.117
30%	112.2	1.904	0.510	0.090	0.400	0.391	0.619	0.437	0.627	0.239
40%	170.1	58.79	0.449	0.053	0.266	0.333	0.632	0.369	0.337	0.259
50%	184.3	15.67	0.477	0.052	0.451	0.443	0.503	0.436	0.470	0.249
60%	243.0	67.42	0.476	0.056	0.641	0.409	0.310	0.388	0.522	0.184
70%	340.6	59.71	0.516	0.068	0.647	0.441	0.383	0.422	0.429	0.188
80%	345.0	54.96	0.520	0.048	0.640	0.445	0.399	0.420	0.474	0.273
90%	393.9	53.44	0.505	0.048	0.455	0.419	0.555	0.404	0.522	0.086
100%	374.5	20.61	0.498	0.014	0.322	0.468	0.574	0.464	0.396	0.306

Considering both sensitivity and specificity, we can see that, in general, the CNN was able to better classify images of patients without pneumonia than of patients with pneumonia. This could be because, among the images with pneumonia, there were those of patients infected by viruses and bacteria, which were grouped into a single class: pneumonia.

8.4 Precision

The mean values, as well as the standard deviation, are shown in Table 4 and were obtained when evaluating the precision of the ten executions of the CNN.

In the case of the precision metric, the values ranged from 0.34 to 0.71, or what would be between 34 and 71%. The average values were higher, although these will have to be taken with reservations, this due to the behavior of the metric to produce high values in situations such as, for example, suppose you have 30 examples of patients with pneumonia and only one of them is classified correctly and with no *PF* (Positive false) values, then the precision value would be 1 or 100%, this despite there are 29 images of patients with pneumonia that were classified incorrectly.

8.5 ROC Curve

In the ROC curve, the sensitivity values are on the y-axis, while the values of 1-specificity are on the x-axis. The graph is shown in Figure 2.

In Figure 2, it can be seen that there are some data sets in the desirable area, such as the 10%, 30%, 70%, 80% and 90% data sets. On the other hand, the rest of the data sets tended to classify the images in a more random way.

The random behavior in the metrics could be since only one epoch was defined in the training for the ten data sets, for memory reasons, despite the increase in the size of

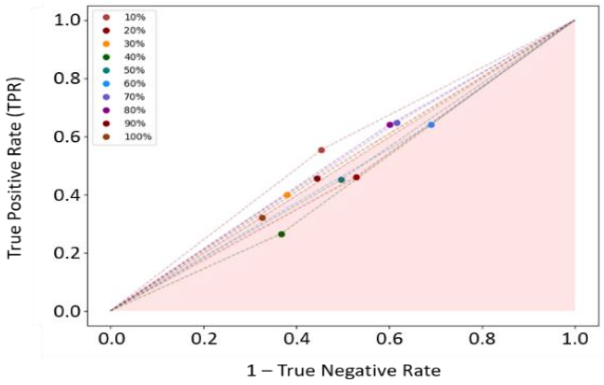

Fig. 2. ROC Curve.

the training data sets. This behavior of the algorithm is logical, since the larger the database, the more complex it becomes. Therefore, it is necessary to increase the number of epochs as the size of the data set increases. In addition, there is the possibility of the noisy factor that can be present in the database from the moment the images were captured. Regarding the noisy factor, although it is a latent problem, it was left like that in order to focus on the behavior of the CNN in Spark.

9 Conclusions

In this project, a Convolutional Neural Network (CNN) was implemented in a pseudo-distributed environment in Spark using the Dist-Keras library. The CNN was used for the automatic classification of images from the database of patients with and without pneumonia taken of [11]. The configuration of the CNN was obtained from the example applied to the MNIST database [6].

Database was divided in ten image sets, each one with different size in order to explore the behavior of the CNN as the size changes. The metrics of precision, accuracy, sensitivity and specificity were used to evaluate the CNN. The CNN was run ten times on each image set to perform classification tests.

Although the database increased, the CNN was able to continue the training and classification process, with an increase in time as expected due to the increase in the size of the database. On the other hand, the metrics values became questionable, due to the great variability of them as the size of the database changes.

After the relevant tests, it was confirmed that the training time of the CNN increases as the size of the database is extended. For future work, it is recommended to increase the number of times for the CNN's training, since as the size of the database increases, the analysis process becomes more complex. Similarly, it is recommended to implement the CNN in a cluster, to see what the performance of the CNN in Spark is as the number of slave nodes increases. Finally, images of patients with bacterial and

viral pneumonia could be separated into two distinct classes, with the intention of decreasing the number of false negatives.

Acknowledgment. The authors would like to acknowledge the financial supports from the Mexican PRODEP (Project UACJ-PTC-423).

References

1. Aggarwal-Charu, C.: Neural networks and deep learning: A textbook. Yorktown Heigths, Springer (2018)
2. Aghdam-Hamed, H., Jahani-Heravi, E.: Guide to convolutional neural networks: A practical application to traffic-sign detection and classification. Springer (2017)
3. Belloti, R., Tangaro, S., Cerello, P., Bevilacqua, V.: Distributed medical images analysis on a grid infrastructure. Future Generation Computer Systems, 23, pp. 475−484 (2007)
4. Cali, C., Longobardi, M.: Some mathematical properties of the ROC curve and their applications. Ricerche di Matematica, Springer Nature, 13, pp. 391−402, (2015)
5. Hermans-Joeri, R.: CERN IT-DB Distributed Keras: Distributed deep learning with apache spark and keras. Github Repository (2016)
6. Hermans-Joeri, R.: Distributed Keras: Introduction. https://joerihermans.com/work/distributed-keras/ (2019)
7. Juba, B., Le-Hai, S.: Precision-recall versus accuracy and the role of large data sets. https://pdfs.semanticscholar.org/7abb/63a5cb77a0ada993cfe2328f38689431e2da.pdf. (2019)
8. Keras Documentation: Keras: the Python deep learning library (2019)
9. Kermany, D.S., Goldbaum, M., Cai, W., Valentim, C.C.S.: Identifying medical diagnoses and treatable diseases by image-based deep learning. Cell, 172, pp. 1122−1131 (2018)
10. Kumar- Manaswi, N.: Deep Learning with Applications Using Python: Chatbots and Face, Object, and Speech Recognition with Tensorflow and Keras. Apress (2018)
11. Mooney, P.: X-Ray Images (Pneumonia). https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia/home (2018)
12. Ramasubramanian, K., Singh, A.: Machine learning model evaluation. Machine Learning Using R, Apress (2016)
13. Srinivasa, K.G., Siddesh, G.M., Srinidhi, H.: Apache spark computer communications and networks. Springer International Publishing (2018)
14. Yan Yuzhong, Huang Lei: Large-scale image processing research cloud. In: The Fifth International Conference on Cloud Computing, GRIDs, and Virtualization, pp. 88−93 (2014)
15. Zhang, Z., Barbary, K., Nothaft, F.A., Sparks, E., Zahn, O., Franklin, M.J., Patterson, D.A., Perlmutter, S.: Scientific computing meets big data technology: An astronomy use case. In: IEEE International Conference on Big Data, University of California Press (2015)