ASTES

*Advances in Science, Technology and Engineering Systems Journal*
*Vol. 5, No. 6, 762-767 (2020)*
www.astesj.com
*Special Issue on Multidisciplinary Innovation in Engineering Science & Technology*

**ASTES Journal**
**ISSN: 2415-6698**

# Optimization of Multi-user Face Identification Systems in Big Data Environments

Majdouline Meddad[*,1], Chouaib Moujahdi[2], Mounia Mikram[1,3], Mohammed Rziza[1]

[1]*LRIT Laboratory, Associated Unit to CNRST (URAC 29) Rabat IT Center, Faculty of Sciences, Mohammed V University in Rabat, 10010, Morocco*

[2]*Scientific Institute of Rabat, Mohammed V University in Rabat, 10010, Morocco*

[3]*School of Information Sciences in Rabat, 10010, Morocco*

ARTICLE INFO

*Article history:*
*Received: 29 August, 2020*
*Accepted: 14 November, 2020*
*Online: 24 November, 2020*

*Keywords:*
*Big Data*
*Biometric Distribution*
*Face Identification*
*Convolutional neural network*

ABSTRACT

*Computer vision offers several strategies that permit computers to comprehend the substance of inputted data to extract the relevant highlights features. That gives the possibility to develop several successful recognition systems like face identification. One of the enormous difficulties these days is the way to have a prompt identification face in a multi-client identification system. We propose in this paper, an optimization of some face identification systems in big data environments that manages the cost of an appropriate identification time while holding a good performance of the system. The used systems are based on: CNN with Inception V3, PCA, LDA and LBPH. Our experimental results indicate that the performance can be preserved while reducing considerably the running time. LBPH with a Radius equal to 2 and 8 neighbors gives the best accuracy of that is equal to 95,71% with an identification time of 4.44 seconds.*

## 1 Introduction

Computer vision offers several techniques for machines to understand the content of images like human vision. To understand the content of images, machines need some image processing steps to be able to extract features from the images for several applications such as face identification, object recognition and object tracking.

The substantial idea of a biometric system is to utilize acquisitions of behavioral and/or physiological characteristics of human like face, fingerprint, signature and voice for recognition/verification. Currently, several face identification systems use large-scale databases that contain a big number of subjects such as celabA Dataset that contains 10177 identities and 202599 images. Large-scale databases present a big open challenge to create an operational biometric identification that offers an acceptable identification time. The Scalability [1] of such tremendous systems makes the challenge harder to be solved. This work is concerned with the creation and optimization of such biometric identification systems.

This paper is an extension of a previous work presented during 2019 in the 1st International Conference on Smart Systems and Data Science (ICSSD) [2], where we have tested and built several identification systems (i.e., LBPH [3],[4], PCA [5], LDA [5] and deep learning with inception V3 [6]) that use a big database with some hard conditions.

The remainder of the paper is structured as follows. The related works are presented in Section 2. Section 3 presents the different used face identification systems with all optimization steps. Our experimental results are introduced in Section 4 Our conclusion and perspectives are given in Section 5.

## 2 Related Work

Identification time, efficiency and security are three crucial issues from which large-scale identification systems suffer. Due to these problems. Several published kinds of research deal with such problems. On one hand, some published approaches tackle security issues whom we can cite [7],[8] that implemented Biometric anti-spoofing technique by using the trait randomization technique.

On the other hand, published approaches can be part of two significant classes: learning distribution approach ( first class) and Processing distribution approaches (second class). The principal idea of the second class is sharing tasks to several workers in the cluster to handle the issue of treatment large amounts of data in a few

[*]Corresponding Author: Majdouline Meddad, Email: majdouline_meddad@um5.ac.ma

times by employing the big data technologies in the cloud environment for parallel-distribution treatment. Well known, deep learning models are hefty in computing when it needs more iterations to get the best performant model which needs to use an approach of the first class with big data technologies to distribute the learning process.

For the first class, the crucial idea presented in [9] is merging the utility of the neural network algorithm with the MapReduce technique to face the extensive data problem in handwritten characters. The authors in [10] proposed for the handwriting recognition digit by firstly applying an elastic distortion to the entered data then Implemented a computing method with distribution to gain the time cost by mapping the dataset into several machines of the cluster. In [11], the author proposed some parallelization methods which are MRBPNN 1, 2 and 3) based on a MapReduce technique to face exhaustive scenarios in the conditions of the size of data and neurons.

For the second class, [12] displays a MapReduce technique of the process of converting frames of video to grayscale images as a preprocessing technique to extract features from converted images. In [13], the author apply a distributed medical application in two different techniques of distribution such as MapReduce and Sun Grid Engine regarding running time. In [14], the author introduced an approach for storing and processing satellite images by a remote sensing processing tools OTB with the MapReduce technique of Big Data technologies. In [15], the author proposed to use the MapReduce for parallel model execution to extract the similarity of the tumor image. We trust that the principal drawback of this class is hard to manage the cluster to ensure continuity of processing in a distributed platform and Hadoop technology requires a significant memory.

## 3 Face Identification In Big Data Environment

The conception of a biometric system carries four principal parts as shown in FIG.1.

- The sensor part (Grey Box) that obtains the biometric information to remove a computerized portrayal during enlistment, identification or verification.
- The characteristic extraction part (Green Box) that empowers decreasing the dimensionality of the representation, emitted by the sensor part, by extricating the significant capabilities. This part can possess a sub-part to test the norm of gained biometric information.
- The database part (Red Box) that comprises the biometric formats of the enlisted subjects inside the framework.
- The decision part (Orange Box) that contrasts the test format and the enlisted formats of the database part requires a decision with regards to the sort of the framework.

The correspondence between these parts becomes tedious inside of having huge database for identification systems. To fix this difficulty, we will utilize some technologies for parallelizing these correspondence tasks inside various slaves. Indeed, current datasets are quickly expanding in terms of size and multifaceted nature, also, there is an importunate need to create solutions to extract pertinent data by utilizing methods based on statistics.

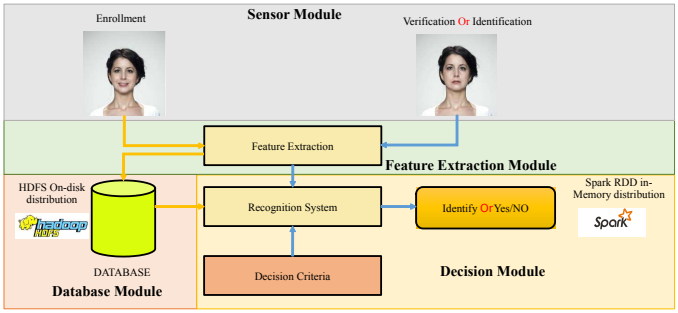


Figure 1: The general conception of a biometric system [16]

An effective biometric framework might be a framework that primarily controls the capacity of the database to be changed in terms of size or scale. they are numerous big data frameworks that are utilized for overseeing a huge data, as Apache Spark framework [17] and Hadoop [18] that run dataflow diagrams over the slaves of a cluster.

In the data-parallel computing process, a client program is arranged into an execution plan graph (EPG), This EPG is the central information structure utilized for distributing tasks, control distributed jobs in the slaves and adapting to non-critical failure. Once a job is running, EPG is unaltered at runtime aside from some restricted adjustments. This makes it hard to utilize dynamic optimization procedures that could significantly improve the disseminated execution dependent on runtime data. Optimus [19] is a framework for progressively rewriting an EPG at runtime.

MillWheel [20] is a system for constructing a data processing with a high size of data in a few time where the user describes the code and the directed computation graph for specific nodes then the framework oversees determined state and recording continuously the flow with non-critical failure in stream processing.

For supporting embedded devices, Sonora [21] is a framework that enhance the stream abstraction for the accompanying reasons. To begin with, streams effectively bind together numerous data activities in cloud and Mobile.

Apache Spark, Naiad [22] and Hadoop run in less calculation time when there is enough memory inside the group of machines to convey the working arrangement of the calculation. Spark keeps up MapReduce's linear adaptability and adaptation to non-critical failure, it is a lot quicker, a lot simpler to program, the distributed data of the scope of computationally heavy tasks, including graphic process, interactive queries, flows and Machine Learning (ML).

In this paper, the jobs of a huge face identification system are parallelized by utilizing Hadoop[23] with Apache Spark introduced by Directed Acyclic Graph (DAG) in figure 3. Every Spark application includes a driver program that completes the user's function and carries out distributes operations on a group of the machine (cluster). The main reflection that Spark furnishes could be resilient distributes dataset (RDD), which could be an assortment of parceled items on a cluster that will run in equal. Users can even request that

Spark remain RDD in memory, permitting it to proficiently reutilize operation in parallel. when a node is failed RDD can automatically be recovered.

### 3.1 Distributed tasks of LBPH, PCA and LDA in Apache Spark

The utilized identification system is predicated on [3]. However, we have made some changes, to support it inside the Apache Spark, following the stages beneath (see additionally figure 2).

We firstly make first RDD (RDD1) by loading all test images such as Binary Files from HDFS then we split it into sixty parts. Secondly, we apply a transformation of the RDD1 to make RDD2 by transforming the color image to grayscale. Thirdly, we create RDD3 by applying transforms of loaded images in RDD 1 to extract real labels. Fourthly, we made RDD 4 by extracting the content of previous RDDs (2 and 3) by applying a collective action. Finally, the identity of each image is parallelized and compared with the right label extracted in RDD3.

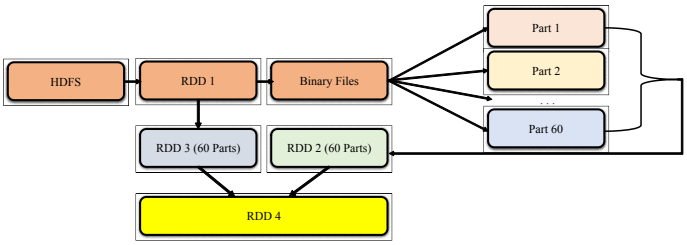


Figure 2: The process of parallelization in Apache Spark

To diminish the running time within the identification stage using the normal PCA, LBPH and LDA identification systems. Initially, the first Apache Spark RDD was created by recovering the test set in 60 parts as binary files from the Hadoop HDFS, and afterward, every thread of our computer changes every picture of every part to grayscale to create the second RDD of grayscale images. Then a third RDD is created by extracting the right label of each test image to collect their identity.

Secondly, we extract the PCA, LDA and LBPH characteristic vectors and compare the vector of the test with those of the learning set.

Thirdly, we predict the identity of the image of the test with the minimal distance overall comparison of the images from the learning set. We used HDFS in the database module to store images and Apache Spark for characteristic extraction and decision parts.

### 3.2 Pre-trained CNN model, LBPH, PCA and LDA for Multi-user identification system

In this sub-section, we have utilized the Transfer learning procedure where a model produced for tackling an issue is reutilized for a subsequent issue in a timesaving way by starting from patterns that have been learned in the first task, in this manner you raise antecedent learning and keep away from starting from scratch.

We have utilized the Inception V3 as an extractor of pertinent characteristics. It is utilized besides to dispose of the last completely connected layer, at that point, we train the rest of the model as a learning set element extractor for the new dataset. Inception V3 extract 2048 values for every picture. We call these highlights CNN codes. Whenever we have removed the codes for whole pictures. We have prepared a totally connected layer with a Softmax classifier for our system.

A while later, to claim less period, due to the mechanism of spark's build-in broadcast the model is firstly distributed to the cores of our machine in [24]. At that point, this model is used on each core and utilized to identify the identity of an entered image. For this work, the models of a huge identification face system are distributed by utilizing just Apache Spark technology [23].

In our work, every worker takes an image to identify by turning the facial identification system based on PCA, LDA or LBPH to predict the identity of a client by calculating the similarity value with all over the training set as shown in figure 3(a).

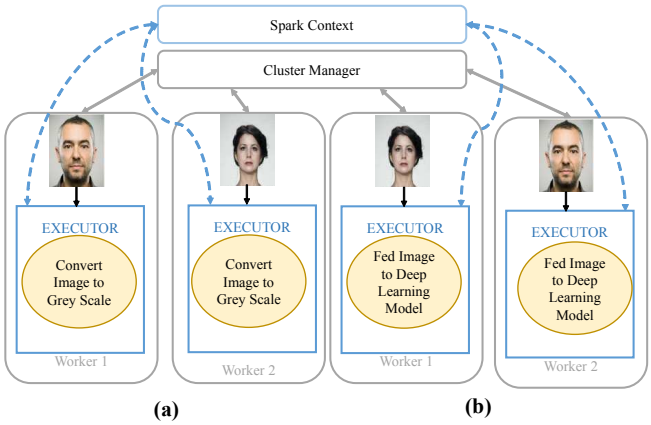


Figure 3: Multi-user identification system: (a) for PCA, LDA and LBPH, (b) for Deep Learning Model

Firstly, the image from RGB (color) to grayscale is converted, afterwards, Every worker compute the eigenvectors, LBP histogram and fisherFace vectors then every worker of the cluster identifies the identity of the tested image from saved PCA, LDA or LBPH models which contain all computed values as shown in figure 3(a) and shared to all cores of the machine.

We employed in this paper a pre-trained CNN Model (Inception V3). Firstly, we have only trained the classification part of the Inception V3 model due to the computational cost of learning to create a model that can identify the person from his/her face image which we need to reutilize the model trained from the Imagenet database to our chosen dataset, we have trained the classification part to induce the correct weights and bias of the fully connected layer (FC) for identification face system, thereafter, we fed the test image into the model to urge the identity, this is often the task of every worker within the cluster as illustrated in figure 3(b) where every worker takes a test image from Hdfs then fed it into the model to extract characteristics that contain 2048 values of every vector then pass it through classification part (Fully connected layer trained to create the facial identification system) then introduce the identity as the last step.

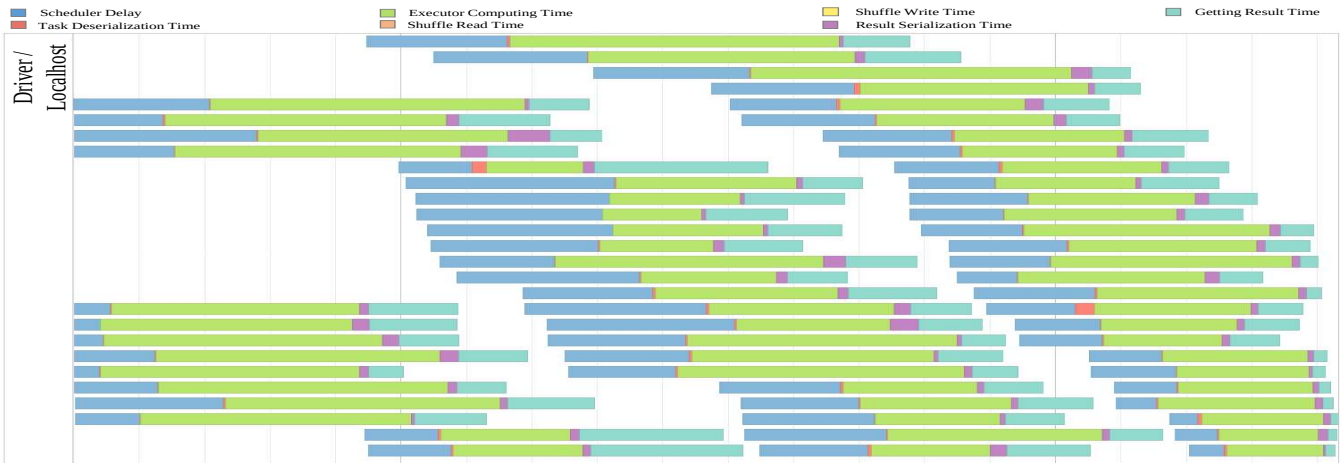


Figure 4: The run time using 6 cores (12 threads) of various tasks

# 4 Experimental Results

In this section, we evaluate the optimization of various identification systems : the FisherFace [5], LBPH [3][4], , FisherFace [5] and Inception V3 [6] that involve 48 layers can distinguish between 1000 items (e.g., Flower, Fish, Bird, etc) in terms of identification time and the precision. For the evaluation, we have got used the Extended Yale Faces B database [25].

## *4.1 Database Description and Evaluation*

We applied 17 changes (e.g., Rotation, Blur, Noise, etc.) for every subject of the database utilized in this paper [26] to make a bigger database and also to make the identification task more harder.
This database involves 28 subjects with 16128 images with 9 poses and in 64 lighting conditions, In our paper, we are just engaged by 26 subjects. Consequently, the database consists of 107730 images in total. We have suggested splitting the database into three sets as described in figure 5.

## *4.2 Results and discussion*

In the aim to reduce the running time, we run our system using Apache Spark technology by making a collection of RDDs. Every thread of the computer core runs a part of partitioned RDD in 60 parts as illustrated in the figure 4. Each job could be a part of the step, as an example, the primary job contains 60 tasks, in as much as our machine involve 12 threads, 12 tasks are going to be run in parallel (see figure 4). figure 6 illustrates a comparison between various face identification systems that utilize EigenFace, LBPH, FisherFace, and CNN model in Apache Spark [24].

The output of the last convolutional layer of the CNN model is called cnn code extracted from the entered images (Trained images) during the primary part of the learning phase, this stage takes 4 hours and 19 minutes by using Apache Spark broadcasting to calculate all codes. Running the identical stage of the learning step it takes 14 hours without utilizing the Apache Spark broadcasting, is an off-the-cuff term of a feature vector that plays a vital part of identification, which is a representation of a picture in less number of values that will be used for classification.

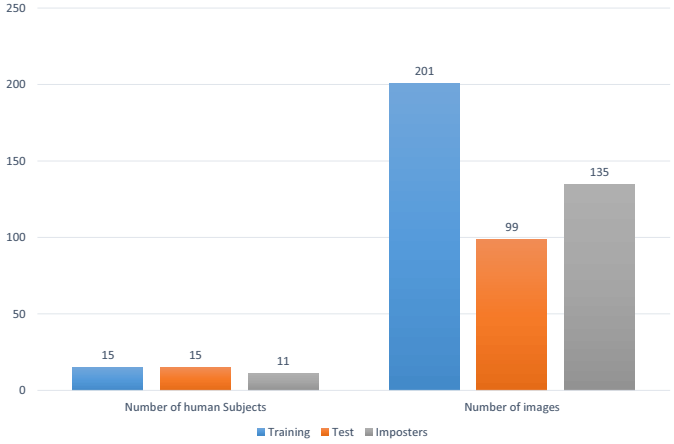


Figure 5: Database Distribution

This final layer was created to give a collection of values that is fitting to the classifier to separate among users. This implies that it needs to be a meaningful and good representation of the images because it should have as many relevant features for the classifier to create an accurate decision by just using a few values.

During the learning phase, every picture is reutilized various times during the learning phase and every code takes an all-inclusive time to be calculated. To quicken the job we conserve these codes on our machine to not recalculate them consistently. On the off chance that we re-execute the framework once more without feed in images to the model.

When the codes are all extracted in the first step, the specific development of the last layer of the neural network model starts. Many series of outputs will be during this step that can show the exactness of the validation and hence the precision of the learning part.

The precision of the learning shows how the model learned from the pre-owned pictures to distinguish between identities. Validation exactness presents the coordinating rate utilizing an arbitrarily

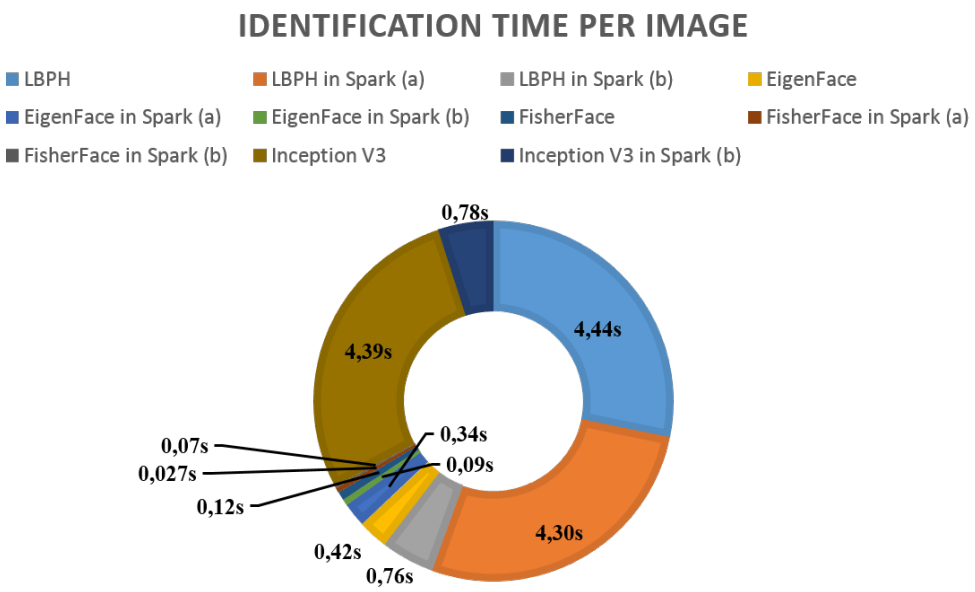


Figure 6: Overall identification time Comparison between The Different Tested Identification Systems with/without distribution

chosen group of pictures from a remarkable learning set. That validation exactness depends on the work of an assortment of data that is not contained inside the learning set if the precision of the learning is high. However, the validation set precision stays low, this suggests that the model is remembering specific highlights inside the learning set pictures. In each validation or learning step, we select 100 arbitrary pictures of the entire validation/learning set, we find its codes inside the cache and brings them into a definitive layer to get identities.

These predictions are contrasted with the specific labels to optimize the classification part and update the bias and weights by utilizing some optimizers like Adam Optimizer and Gradient Descent.

By a learning rate of 0.01 with 250000 iterations using the Gradient Descent Algorithm and Adam Optimizer we respectively get 91, 47% and 91.66%. By a learning rate of 0.1 with 500000 iterations using the Gradient Descent Algorithm and Adam Optimizer we respectively get 91.82% and 92.31%.

By a learning rate of 0.05 with 250000 iterations using the Gradient Descent Algorithm and Adam Optimizer we respectively get 92, 30% and 90.84%. By a learning rate of 0.1 with 500000 iterations using the Gradient Descent Algorithm and Adam Optimizer we respectively get 92.24% and 91.45%.

By a learning rate of 0.001 with 250000 iterations using the Gradient Descent Algorithm and Adam Optimizer we respectively get 87.01% and 91.85%. By a learning rate of 0.1 with 500000 iterations using the Gradient Descent Algorithm and Adam Optimizer we respectively get 88.73% and 92.10%.

We set in the learning phase the proportion of learning set at 80% and we preserved 10% for validation and test. In every iteration we have entered 100 images, after every 10 iterations, we have got evaluated the model using 100 images of the validation set that are chosen arbitrarily. after finishing the learning phase, we fed all the test set through the model to know the precision and how the way the model can classify the inputted faces. After being aware of the good precision of the model in the last part of the learning phase we run our test set for every learning rate and variety of iterations as shown in figure 6. Globally, the precision value is expanded by expanding the number of iterations. If the learning rate is tiny, the precision value is expanded smoothly which makes the learning becomes more reliable, however, the optimization will take longer because steps towards the minimum of the loss function are very small. If the learning rate is high, the precision is expanded rapidly and that we will have the overshoots of the minimum which makes aggravate the loss.

Our main aim is to increase the identification time per image of the test while getting the good performance of the models. This purpose is reached by running our system in Apache Spark (see figure 6). As shown in figure 6 the quick system is FisherFace followed by the EigenFace, then the CNN model, and at last the LBPH.

To advance the exactness of systems, we have made some changes to a few parameters of the LBPH algorithm, such as changing the number of neighbors and the Radius value. For Inception V3, we have made changes to a few parameters simply like the number of iterations, the chosen optimizer algorithm, and subsequently the learning rate value.

For the LBPH algorithm with a Radius equal to 1 with 8, 6 and 4 neighbors, we respectively get 94.05%, 93,74% and 91.91% an Accuracy of identification.

For the LBPH algorithm with a Radius equal to 2 and 8,6 and 4 neighbors, we respectively get 95.71%, 95.43% and 94.02% an Accuracy of identification.

All systems are executed in a server that contains 6 cores with 12 threads and 4.70 GHz Turbo frequency with 12Go of the storage memory.

# 5 Conclusion and Perspectives

In this paper, we introduced an optimization of multi-user identification face systems in Big Data environments. Hadoop and Apache Spark are two technologies accustomed to parallel tasks and make multi-user systems. This optimization has been examined and evaluated, as far as accuracy and identification time using various classical identification face systems. Our experimental results show that the performance is preserved while offering an affordable identification time for oversized scale systems. In our future work, we have plan to create a multi-view identification face system using a Siamese Convolutional network to tackle the issue of having a few facial acquisitions.

**Acknowledgment** The financial support of the "Centre National pour la Recherche Scientifique et Technique" CNRST, Morocco, is acknowledged by Majdouline Meddad.

# References

[1] K. Janocha, W. M. Czarnecki, "On loss functions for deep neural networks in classification," Schedae Informaticae, 2016, doi:10.4467/20838476SI.16.004.6185.

[2] M. Meddad, C. Moujahdi, M. Mikram, M. Rziza, "A Multi-User Face Identification system with distributed tasks in a Big Data Environment," in ICSSD 2019 - International Conference on Smart Systems and Data Science, 2019, doi:10.1109/ICSSD47982.2019.9003112.

[3] T. Ahonen, A. Hadid, M. Pietikäinen, "Face description with local binary patterns: Application to face recognition," IEEE Transactions on Pattern Analysis and Machine Intelligence, 2006, doi:10.1109/TPAMI.2006.244.

[4] D. Huang, C. Shan, M. Ardabilian, Y. Wang, L. Chen, "Local binary patterns and its application to facial image analysis: A survey," IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews, 2011, doi:10.1109/TSMCC.2011.2118750.

[5] A. Khan, H. Farooq, "Principal Component Analysis-Linear Discriminant Analysis Feature Extractor for Pattern Recognition," **8**(6), 267–270, 2012.

[6] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016, doi:10.1109/CVPR.2016.308.

[7] K. Okereafor, C. Onime, O. Osuagwu, "Enhancing Biometric Liveness Detection Using Trait Randomization Technique," in 2017 UKSim-AMSS 19th International Conference on Computer Modelling Simulation (UKSim), 28–33, 2017, doi:10.1109/UKSim.2017.44.

[8] K. OKEREAFOR, O. Osuagwu, C. Onime, "Biometric Anti-spoofing Technique Using Randomized 3D Multi-Modal Traits," 2018, doi:10.6084/m9.figshare.12479165.v1.

[9] K. Sun, X. Wei, G. Jia, R. Wang, R. Li, D. C. Oct, "Large-scale Artificial Neural Network : MapReduce-based Deep Learning," .

[10] N. Basit, Y. Zhang, H. Wu, H. Liu, J. Bin, Y. He, A. M. Hendawi, "MapReduce-based deep learning with handwritten digit recognition case study," in Proceedings - 2016 IEEE International Conference on Big Data, Big Data 2016, 2016, doi:10.1109/BigData.2016.7840783.

[11] Y. Liu, J. Yang, Y. Huang, L. Xu, S. Li, M. Qi, "MapReduce Based Parallel Neural Networks in Enabling Large Scale Machine Learning," Computational Intelligence and Neuroscience, 2015, doi:10.1155/2015/297672.

[12] M. Yamamoto, K. Kaneko, "Parallel Image Database Processing With Mapreduce and Performance," International Journal of Electronic Commerce Studies, 2012, doi:10.7903/ijecs.1092.

[13] T. Ben-Nun, T. Hoefler, "Demystifying parallel and distributed deep learning: An in-depth concurrency analysis," ACM Computing Surveys, 2019, doi:10.1145/3320060.

[14] I. Chebbi, B. W, F. I. R, "Improvement of Satellite Image Classification : Approach Based on Hadoop / Map Reduce," in 2nd International Conference on Advanced Technologies for Signal and Image Processing - ATSIP'2016 March 21-24, 2016, Monastir, Tunisia, 31–34, 2016.

[15] D. D. Manojbhai, K. K. Pradipkumar, R. Rajamenakshi, "Big image analysis for identifying tumor pattern similarities," in Proceedings of 2016 International Conference on Advanced Communication Control and Computing Technologies, ICACCCT 2016, 2017, doi:10.1109/ICACCCT.2016.7831596.

[16] A. K. Jain, P. Flynn, A. A. Ross, Handbook of Biometrics, Springer Publishing Company, Incorporated, 1st edition, 2010.

[17] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, I. Stoica, "Spark: Cluster computing with working sets," in 2nd USENIX Workshop on Hot Topics in Cloud Computing, HotCloud 2010, 2010.

[18] S. G. Manikandan, S. Ravi, "Big data analysis using apache hadoop," in 2014 International Conference on IT Convergence and Security, ICITCS 2014, 2014, doi:10.1109/ICITCS.2014.7021746.

[19] Q. Ke, M. Isard, Y. Yu, "Optimus: A Dynamic Rewriting Framework for Data-Parallel Execution Plans," in Proceedings of the 8th ACM European Conference on Computer Systems, EuroSys '13, 15–28, Association for Computing Machinery, New York, NY, USA, 2013, doi:10.1145/2465351.2465354.

[20] T. Akidau, A. Balikov, K. Bekiroğlu, S. Chernyak, J. Haberman, R. Lax, S. McVeety, D. Mills, P. Nordstrom, S. Whittle, "MillWheel: Fault-Tolerant Stream Processing at Internet Scale," Proc. VLDB Endow., **6**(11), 1033–1044, 2013, doi:10.14778/2536222.2536229.

[21] X. Chen, I. Beschastnikh, L. Zhuang, F. Yang, Z. Qian, L. Zhou, G. Shen, J. Shen, "Sonora: A Platform for Continuous Mobile-Cloud Computing," Technical report, 2012.

[22] D. G. Murray, F. McSherry, R. Isaacs, M. Isard, P. Barham, M. Abadi, "Naiad: A timely dataflow system," in SOSP 2013 - Proceedings of the 24th ACM Symposium on Operating Systems Principles, 2013, doi:10.1145/2517349.2522738.

[23] A. V. Hazarika, G. Jagadeesh Sai Raghu Ram, E. Jain, "Performance comparision of Hadoop and spark engine," in Proceedings of the International Conference on IoT in Social, Mobile, Analytics and Cloud, I-SMAC 2017, 2017, doi:10.1109/I-SMAC.2017.8058263.

[24] T. Hunter, "Deep Learning with Apache Spark and TensorFlow," 2016.

[25] D. K. K.C. Lee , J. Ho, "Extended Yale Face B," 2016.

[26] A. S. Georghiades, P. N. Belhumeur, D. J. Kriegman, "From few to many: Illumination cone models for face recognition under variable lighting and pose," IEEE Transactions on Pattern Analysis and Machine Intelligence, 2001, doi:10.1109/34.927464.