INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL
Online ISSN 1841-9844, ISSN-L 1841-9836, Volume: 19, Issue: 3, Month: June, Year: 2024
Article Number: 6428, https://doi.org/10.15837/ijccc.2024.3.6428

CCC Publications

AGORA UNIVERSITY PRESS

Deep recurrent neural networks distributed on a Hadoop/Spark cluster for fall detection

M. Hamdi, H. Bouhamed, F. Badreddine, R. Alkanhel

Monia Hamdi
Information Technology, College of Computer and Information Sciences
Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia
mshamdi@pnu.edu.sa

Heni Bouhamed*
Advanced Technologies for Image and Signal Processing Unit (ATISP)
Sfax University, Sfax, Tunisia
* Corresponding author: heni.bouhamed@fsegs.usf.tn

Fady Badreddine
Advanced Technologies for Image and Signal Processing Unit (ATISP)
Sfax University, Sfax, Tunisia
fadybadreddine@gmail.com

Reem Ibrahim Alkanhel
Department of Information Technology, College of Computer and Information Sciences,
Princess Nourah bint Abdulrahman University,
P.O. Box 84428, Riyadh 11671, Saudi Arabia
rialkanhal@pnu.edu.sa

Abstract

Falls detection approaches struggle with both Big Data scalability and upholding individual privacy, this research work proposed a novel approach for posture recognition followed by fall detection, taking advantage of the synergy between Random Forests and Uniform Local Binary Patterns (uLBP) histograms for an accurate and fast posture identification while respecting privacy. Additionally, it referred to deep recurrent neural networks distributed on a Hadoop and Spark platform for time series analysis in fall detection. This combination of methods allowed us to achieve acceptable real-time monitoring precision. This study, therefore addressed two objectives simultaneously: efficiency and scalability in posture recognition using Random Forests and uLBP, and fall detection relying on the recurrent neural network (RNN) for time series processing. The suggested solution is designed for home telemonitoring, where scalability and effective data management are supported through Hadoop/Spark. The integration of these technologies promotes reliable detection without any privacy violation, paving the way for a wider adoption of home monitoring systems for an increasing population of dependent individuals.

Keywords: Posture recognition, Fall detection, Random Forest, uLBP histograms, Deep recurrent neural networks, GRU, Hadoop, Spark, Time series, Big Data.

1 Introduction

Healthcare is one of the sectors where the impact of Big Data could be most revolutionary. Faced with today's technological innovations, solutions for disease diagnosis, medical monitoring, and patient care are emerging. For example, falls, considered a major threat to seniors, have significant physical, financial, and psychological consequences. They pose a substantial financial burden, accounting for between 0.85% and 1.5% of total healthcare expenditures in such countries as the United States, the United Kingdom, and Australia [1]. One of the major challenges lies in monitoring dependent individuals at home. For these individuals, a simple fall can trigger tragic consequences, both physically and in terms of subsequent complications, potentially compromising their autonomy and drastically affecting their quality of life. Detecting these falls goes beyond the notion of mere convenience; it is a medical emergency. The interesting issue to address is on the way to ensure both prompt and effective fall detection, especially when it occurs at home, while preserving the privacy and independence of the concerned individual.

To efficiently handle Big datasets, technological platforms such as Hadoop and Spark prove relevant in providing a promising response to this issue. The use of sophisticated deep learning algorithms has recently gained popularity in this field [3, 4, 5, 6]. Several studies have shown that Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks are the most popular in recognizing human actions where temporal and spatial information is crucial [7]. Moreover, the Recurrent Neural Networks (RNNs) are proven to be excellent for temporal data processing while CNNs are proficient in spatial analysis. This has made their combined use optimal for video analysis. It has become possible to provide a real-time analysis of data collected from sensors located in the homes of dependent individuals. This analysis could also allow for an instant detection of a fall as well as an identification of behaviors or patterns indicating an increased risk for the monitored individual.

The objective of this research was to merge the urgent clinical need of early detection with technological advances in the field of Big Data. Based on a methodology applied on a Hadoop/Spark platform, we aimed to create a system that is both accurate and responsive; i.e. a system that could provide enhanced protection for dependent individuals while respecting their privacy. In fact, privacy and discretion are an absolute priority in the suggested approach. To this end, the proposed method opted for the uLBP histograms as the only information to store via video sensors. These histograms allow optimizing data processing by reducing personal information while ensuring precise detection, initially distinguishing five distinct daily activities: sitting, standing, lying, crawling, and bending [2]. Subsequently, the postures time series were used for fall detection through an appropriate deep learning model deployed on a scalable Big Data platform. Our work stands out in the field of fall detection research by developing the first scalable Big Data distributed deep learning model, specifically designed to operate in parallel on a Hadoop/Spark cluster through an efficient exploitation of distributed data and respecting people's privacy even in intimate settings, as we only store the uLBP histograms, making it impossible to reconstruct the original images.

The remaining of the paper is structured as follows: Section II contextualized the research study, emphasizing the treated issue and examining the current state of the art. Section III introduced the theoretical and technological framework of the study. Section IV introduced the adopted approach for posture identification using uLBP histograms and Random Forests. It also detailed the use of LSTM/GRU networks for fall detection. The achieved experimental results were revealed and discussed in section V before drawing the final conclusions and suggesting some future prospects in the final section.

2 State of the Art: Study Framework and Existing Approaches

2.1 State of the Art

The majority of fall detection methods involve the recognition of postures. Consequently, this research work first focused on introducing the existing approaches related to both posture recognition and fall detection.

2.1.1 Posture Recognition

Posture Recognition is a multidimensional field that uses a variety of techniques to achieve precise results. In fact, several methods and techniques were proposed relying mathematical functions, 3D sensors, silhouette graphics, or 3D human models. Jeng et al., [8], for instance, introduced a function for posture analysis based on such parameters as the body vertical dimension, its planar shape, variations in its center of gravity, and the time during which these parameters are observed. Height and area are key indicators that vary distinctly according to posture - standing, lying, or in intermediate positions like sitting or leaning. Together with these measurements, significant changes in the center of gravity play a crucial role in identifying dynamic movements such as walking. On the other hand, Diraco, Leone [9], and Siciliano explored posture recognition using a TOF (Time of Flight) 3D sensor. Their research highlights the advantages of TOF sensors, involving privacy preservation and obtaining real 3D information. They developed a specific computational framework for posture classification using volumetric and topological descriptors. In a different context, Modarres and Soryani [10] propose a new representation based on a body silhouette graph called BPG. This descriptor relies on approximating the silhouette using elliptical basis functions and structuring these features into a graph. The BPG is designed to be close to a real human skeleton. This descriptor has several advantages, such as extracting precise landmark points from the silhouette and capturing the structure and relationships among these points. Boulay, Brémond, and Thonnat [11] introduce a method that combines 2D and 3D techniques with a 3D human model for posture recognition. Their method distinguishes various general and detailed postures independently of viewing angles. They also consider integrating temporal information to ensure a more accurate recognition. Li, H., and Sun, Q., [12] used three separate neural networks simultaneously: the first analyzed the outer contour of the body and its general shape, the second focused on the internal structure, especially the joints and the third studied the geometry. The features remain stable regardless of rotation, movement, or image size. After individual training on body representations, the conclusions of these networks are fused using the "D-S evidence theory" to produce a consistent and accurate final evaluation. Finally, a study by AlFayez, F and Bouhamed, H [13] proved the effectiveness of a posture recognition method using uniform Local Binary Pattern (uLBP) histograms and Random Forests, offering high performance, reducing computational complexity and preserving privacy.

2.1.2 Fall Detection

The identification of falls is of paramount importance, especially when it comes to ensuring the protection of the elderly or those in dependent situations. A fall can result in consequences ranging from simple bruises to severe fractures, or even fatal outcomes. Fall detection, a subject of crucial importance, has garnered increasing academic interest since the 1990s. In a pioneering study, Lord and Colvin (1991) [14] introduced the use of a miniature accelerometer combined with a microcomputer chip for fall detection. Similarly, Williams et al., (1998) [15] explored the application of a piezoelectric shock sensor coupled with a mercury tilt switch to monitor body orientation. Along with these initial advancements, the issue of optimizing the sensors placement arose swiftly. Bourke et al., (2007) [16] conducted a comparative study and concluded that the trunk was preferable to the thigh for sensor positioning. This research was fundamental, and the article has been widely cited in this field. Despite the initial success of accelerometers, technological innovation led to the exploration of new approaches. Bourke and Lyons (2008) [17] proposed the use of a single gyroscope, measuring the rotation angle, angular velocity, and angular acceleration simultaneously. Nevertheless, it was claimed that exclusive reliance on accelerometers or gyroscopes could limit the detectors robustness (Tsinganos and Skodras, 2018) [18]. It is in this context that Li et al., (2009) [19] highlighted the added value of fusing the data gathered from the accelerometer and gyroscope, which provided a better discrimination between falls and non-falls occurrences. The emergence of visual sensors introduced a new dimension to this research field, as well. Rougier et al., (2011) [20] implemented sophisticated techniques to track and analyze human silhouettes through video sequences. In a similar vein, Diraco et al., (2010) [21] investigated the use of the ToF camera for fall detection. However, and despite its advantages, the ToF camera has limitations in terms of cost and resolution. Finally, the advent of the Kinect depth camera by Microsoft stimulated a deep shift in fall detection studies. Its combination with accelerometers became a major research topic, as evidenced by the work of Li et al., (2018) [22], revealing that this combination significantly improves detection compared to the individual use of each sensor. The rapid evolution of technologies and algorithms in artificial intelligence has led to significant advancements in the field of fall detection. We are now witnessing a notable transition from traditional methods to more advanced approaches, such as deep learning. Historically, machine learning required precise mapping functions between manually extracted features from raw training data and corresponding output labels, as seen in the works of Saleh and Jeannès (2019) [23] and Wu et al., [24], relying on wearable or ambient sensors like accelerometers. Handcrafted features heavily depend on the domain experts' knowledge. With the shift towards visual sensors, Deep Learning, particularly Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks, has become more prevalent, as illustrated by recent research [26**?**]. Unlike machine learning, deep learning not only determines a mapping function but also learns relevant features hierarchically. In the current landscape of computer vision, deep learning has gained the status of a golden standard, logically leading to the evolution from traditional machine learning methods to more advanced deep learning strategies for recent applications. This evolution is obviously illustrated by the adoption of the LSTM algorithm, a specific variant of recurrent neural networks designed to efficiently handle long sequence learning tasks. It has demonstrated impressive performance with an average recall rate of 95% in current fall detection systems.

2.2 Study context

This research work focused on the detection of falls among dependent individuals at home using a Big Data platform. It comes within a context where technology and artificial intelligence have become essential tools to address significant societal challenges. With the gradual but steady growth in the elderly and dependent population, the need to ensure their safety at home has become a major concern. Statistics show a significant rise in the number of domestic accidents, particularly falls, resulting in serious consequences on their well-being and life quality. The rapid and accurate detection of falls through the recognition of posture sequences is of particular importance. Not only does it allow for prompt intervention in case of an accident, it also helps prevent hazardous situations through monitoring unusual behaviors. However, the implementation of such technological solutions also raises ethical and practical questions, especially regarding the privacy of individuals and the system's effectiveness in real-world conditions. The present research work took into account these facts and developed an intelligent system, based on Machine Learning and/or Deep Learning models, that can accurately recognize different postures and subsequently detect falls. This can be performed while operating in a distributed environment at the processing and storage levels, providing scalability to adapt to the acquisition of large amounts of input data or the increased complexity of required Deep Learning models. It is, therefore, crucial to identify the various questions and challenges our study sought to address. Indeed, posture recognition and fall detection pose multiple challenges, related not only to the technological aspects but also the ethical ones. The key issues that would be addressed and answered in our study are such hot topics as detection reliability, privacy preservation, and scalability. These would be achieved while considering massive data management and seamless technological integration with other devices or platforms to form a comprehensive and connected surveillance solution.

2.3 Criticism to the existing approaches

Based on the literature review, aside from the approach proposed in [13], which is limited to posture recognition, no other method or approach has ventured into scalable Big Data platforms. Indeed, the used Deep Learning techniques have been deployed in a conventional way, relying on the computational capacity of CPU/GPU cores at the level of a single execution server. Furthermore, the data that can be processed is also contingent on the storage space of the server or, alternatively, on cloud storage space, requiring substantial data transfer to traverse the entire dataset and use it in all learning processes. As a result, the objective of our work was to use the approach with the Random Forest and uLBP described in [13] to detect the postures' time series. This aimed to subsequently detect falls through a deep recurrent neural network model, all while deploying both stages on a Hadoop/Spark Big Data platform.

3 Technical and Technological backgrounds

In order to effectively process even large quantities of data rapidly, we have set up and simulated a multi-node Hadoop cluster coupled with Spark, which would serve as the foundation for various experiments ranging from posture recognition with the Random Forest to fall detection with deep recurrent neural networks (LSTM and GRU). All of these experiments would leverage the distributed capabilities of Keras/TensorFlow libraries on the Hadoop/Spark cluster facilitated by the Elephas library. We clarified all of these concepts in the following subsections.

3.1 Hadoop

Hadoop is an open-source framework designed to process large datasets. Developed in 2006 by Doug Cutting and Michael J. Cafarella, it is currently managed by the Apache Software Foundation. Inspired by Google's architecture, particularly the Google File System and MapReduce, Hadoop operates in a distributed computing environment. The Hadoop ecosystem, widely recognized in the world of Big Data, includes key components such as the Hadoop core, MapReduce, and HDFS, along with other complementary components like Apache Hive, HBase, and Zookeeper. This ecosystem enables organizations and researchers to analyze huge amounts of data in an optimized way. Hadoop's distinctive feature lies in its operation in clusters, allowing the mobilization of multiple servers, referred to as "nodes," to handle substantial data volumes. It facilitates the utilization of storage and computing resources across servers grouped in clusters, enabling a distributed processing on large datasets. This provides the basic elements necessary for the operation of applications and services. The ecosystem around Hadoop has evolved over time, and now incorporates countless tools and applications specialized in the field of Big Data. Among the offered tools, Apache Spark has always been referred to for distributed processing systems.

3.2 Spark

The Apache Spark is an in-memory distributed data analysis system. Created by Matei Zaharia at the AMPLab of the University of California, Berkeley, it was later offered to the Apache Software Foundation, becoming a top-level project on February 24, 2014. It offers high-level APIs in Scala, Python, Java, and R, accompanied by powerful libraries such as ML for machine learning, Spark SQL for SQL support, Spark Streaming for real-time streaming, and GraphX for graph processing. Spark was developed to overcome the limitations of Hadoop's MapReduce. Thanks to its fast in-memory processing capabilities and advanced Directed Acyclic Graph (DAG) execution engine, Spark tasks can run several times faster than equivalent MapReduce tasks on a Hadoop cluster. At a high level, Spark distributes the execution of Spark applications across the nodes of the cluster. Each Spark application has a SparkSession object in its main program, representing a connection to the cluster manager. The cluster manager provides computing resources to Spark applications. Once connected to the cluster, Spark acquires executors on worker nodes and then sends them to the application code. An application typically performs multiple tasks in response to a Spark action. Each task is then divided into a Directed Acyclic Graph (DAG) of smaller phases or tasks, which are distributed and sent to the executors to be performed. Spark provides a rich library for machine learning called Spark ML. It stands out as a scalable machine learning library, involving various learning algorithms and useful tools for tasks such as classification, regression, clustering, collaborative filtering, and dimensionality reduction. It also seamlessly integrates with other Spark components like Spark SQL and Spark Streaming. The library is compatible with Java, Scala, and Python, allowing for its integration into complete workflows within Spark applications.

3.3 Random Forests

Random Forest [30] is a machine learning algorithm that was proposed by Leo Breiman and Adele Cutler in 2001. It uses multiple decision trees for classification and regression. It also uses "bagging" (bootstrap aggregation) to reduce variance while maintaining low bias. This process forms individual trees from subsets of training data. Random Forest also applies a method called "feature bagging," which selects a subset of features to reduce correlation between decision trees. This diversity in feature selection makes the trees less similar. For classification, the final prediction is decided by a majority vote of the trees. As for regression, it is referred to as the average of the results. The Random Forest is endowed with some Key hyperparameters such as its ability to adjust to refine the model's accuracy, the number of trees in the forest, the maximum depth of each tree, and the number of features evaluated at each node split. A higher number of trees can enhance complexity and potentially improve the model's accuracy. Deepening the trees can also contribute to a better accuracy by allowing more detailed data separation. Additionally, choosing more features at each split increases the randomness of the method, which, under certain conditions, can improve the prediction accuracy.

3.4 Recurrent Neural Networks

Recurrent Neural Networks (RNN) [32] belong to a specific class of neural networks frequently used in Deep Learning [33, 34]. What distinguishes RNNs is their ability to use previous results as input for subsequent operations. This feature makes them well-suited for handling sequential data. To address the challenge of vanishing gradients in RNNs, a first solution was introduced with the Long Short-Term Memory (LSTM) by S. Hochreiter et al [35], later refined by F. Gers and J. Schmidhuber [36]. The LSTM unit is the fundamental element of an LSTM structure. It consists of a combination of gates and cells working together to produce an output. The second solution is more simple and consists in introducing the Gated Recurrent Unit (GRU), which is a variant of RNNs introduced by Cho et al [38] in 2014. This GRU has two regulation gates, namely the update gate and the reset gate. These gates allow the unit to make a decision about the amount of information to pass to the next time step. Owing to this structure, GRUs can capture dependencies over long distances; they are even simpler in terms of parameterization compared to LSTMs.

3.5 Deep Learning on Spark

3.5.1 TensorFlow and Keras

TensorFlow [44], designed by Google as a successor to Theano, has quickly become the preferred deep learning framework for many professionals. Equipped with a flexible Python API, it operates with a powerful engine developed in C/C++. Keras [45], which was initially developed by François Chollet, a member of the Google team, is a neural network API written in Python and stands out for its open-source nature. Designed to run in conjunction with frameworks like Theano or TensorFlow, Keras simplifies modeling and experimentation in deep learning. It is a high-level API that does not handle low-level operations such as matrix multiplication or tensor convolutions. It uses TensorFlow as the backend engine for such calculations, confirming the integrated and complementary nature of the relationship between the two tools. Leveraging the advanced capabilities of TensorFlow, Keras allows for simplified and faster construction of deep learning models. Together, these synergies enable less code and complexity when designing and experimenting with complex neural architectures. Additionally, users benefit from the robustness, efficiency, and scalability of TensorFlow on a central processing unit (CPU) or multi-core graphics processing unit (GPU), on the one hand, and can opt to conduct distributed training on multiple CPUs or GPUs across a machine cluster through platforms like Spark using extensions such as DistKeras and Elephas, on the other.

3.5.2 Elephas

To harness the distributed computing capabilities provided by Spark, opted to rely on Elephas [46] in this study. Elephas is able to take advantage of the computational power of Spark and the flexibility of Keras/Tensorflow neural networks. Besides, it simplifies and accelerates the parallelization of model training, making the processing of voluminous or big data more feasible and efficient. Further, it is an open-source library for scaling up Keras/TensorFlow learning codes across multiple CPUs or GPUs in a machine cluster.

Distributed Training Process with Elephas The Elephas distributed training process is split into several steps:
(a) Master Model Initialization: Elephas starts by creating a Spark session and initializes a parameter server to manage the master model.
(b) Data Distribution: The master model evenly distributes the data among the different working nodes, also known as workers or slaves (in a Hadoop/Yarn cluster).
(c) Model Replication: A copy of the master model is then created on each working node. This prepares each node to independently perform the training.
(d) Parameter Broadcasting: The master model transmits its parameters to the model copies on the working nodes to ensure consistency before starting the training.
(e) Training Commencement: Each working node starts training with its portion of data and its local copy of the model.
(f) Parameter Update: After training, the working nodes send parameter updates to the master model. The master model uses these updates to adjust its own parameters, incorporating the learning achieved by each node.
(g) Training Completion: Once all working nodes have finished training and sent their updates, the master model consolidates this information to finalize the trained model.

Elephas Estimator Configuration The Keras/Tensorflow model (such as LSTM or GRU) is enclosed in an Elephas estimator, configured to specify the chosen distributed training method. The optimization parameters and critical hyperparameters, such as the number of epochs and batch size, are also determined. Elephas offers two synchronization modes:
(a) Asynchronous Mode: Each node (or worker) independently updates the model parameters as soon as it finishes processing its data batch, without waiting for other nodes. This accelerates the training since nodes do not need to wait for each other. However, updates occur independently, which can sometimes lead to conflicting occurrences, making the model convergence less stable.
(b) Synchronous Mode: All nodes (or workers) perform their computations on their respective data batches but wait for each node to finish before updating the model parameters globally. This means that the model updates are based on information from all the nodes, generally making the model convergence more stable compared to the asynchronous mode. However, the synchronous mode can be slower as it requires all nodes to wait for the outcomes from the other nodes before proceeding with the update.

4 Our Approach

Our approach focuses on two main steps: the recognition of the monitored individual's posture sequence followed by fall detection, all deployed on a Hadoop/Spark platform. Our first step is applied to a series of images, beginning with the extraction of the individual's silhouette through pixel differentiation with a reference image. Since our approach places privacy and discretion at the forefront, we opted for uLBP histograms as the sole information to be stored via video sensors for posture recognition using the Random Forest model [2]. Our second step then uses the temporal series of the postures for fall detection through a deep recurrent neural network model. Figure 1 illustrates the above-introduced steps of our approach.

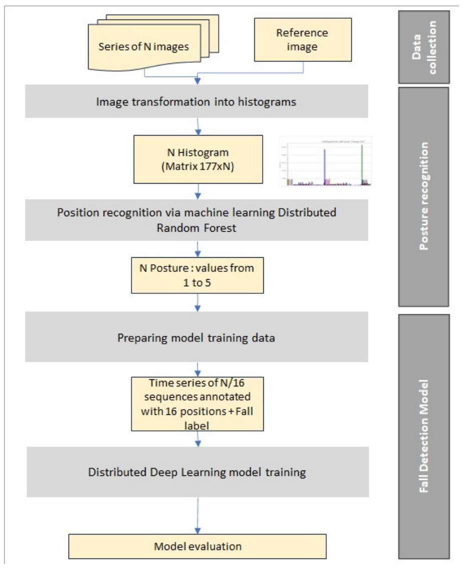

Series of N images
Reference image
Data collection
Image transformation into histograms
N Histogram (Matrix 177xN)
Position recognition via machine learning Distributed Random Forest
N Posture : values from 1 to 5
Posture recognition
Preparing model training data
Time series of N/16 sequences annotated with 16 positions + Fall label
Distributed Deep Learning model training
Model evaluation
Fall Detection Model

Figure 1: Block diagram of proposed model.

4.1 Posture recognition

In our posture recognition process (see Figure 2), a series of images, including a reference image, is first converted to grayscale. Then, and after subtracting the reference image to isolate the silhouette, features are extracted using uLBP patterns at three different scales (r=1,2, and 3). These data are afterwards processed by a distributed Random Forest model via the Apache Spark, capable of distinguishing five distinct postures, providing a structured method for human posture recognition trained on the same data in Alfayez et al., 2023 [13].

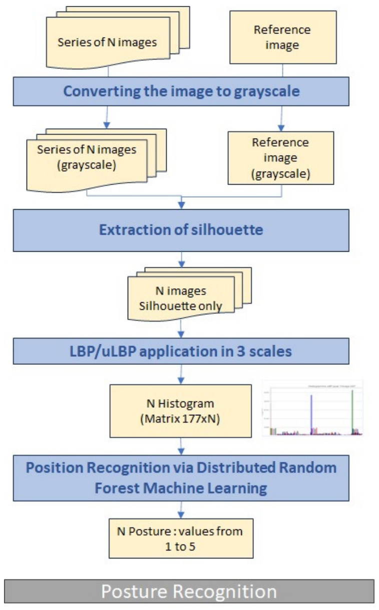

Series of N images
Reference image
Converting the image to grayscale
Series of N images (grayscale)
Reference image (grayscale)
Extraction of silhouette
N images Silhouette only
LBP/uLBP application in 3 scales
N Histogram (Matrix 177xN)
Position Recognition via Distributed Random Forest Machine Learning
N Posture : values from 1 to 5
Posture Recognition

Figure 2: Posture Recognition process.

4.1.1 Image Grayscale Conversion

Before initiating the recognition process, a collection of photos is gathered, with each photo representing a specific posture. Color can sometimes introduce additional variables that complicate the analysis. Indeed, nuances and color variations in images can play a significant role in data interpretation. This is where the crucial step of converting color images to grayscale comes into play (see Figure 3). This process involves removing color information from the image but retaining brightness details, simplifying the analysis while preserving essential elements of the image. This entails transforming the values of the three-color channels (Red, Green, Blue) into a single value representing light intensity. OpenCV was used to perform this conversion.

Figure 3: Grayscale image.

4.1.2 Silhouette Extraction

Silhouette extraction [13] is performed by subtracting the current image from the reference image. In other words, the brightness value for each pixel in the current image is subtracted from the corresponding brightness value of a pixel in the reference image.

4.1.3 Conversion of Images into LBP/uLBP Histograms

The LBP (Local Binary Pattern) is a form of grayscale used to measure texture. It was initially introduced by Ojala et al., [29] as a way to assess the local contrast of an image. In its original definition, the LBP is designed around a neighborhood of eight pixels, using the grayscale value of the central pixel as a threshold. If a neighbor has a value greater than or equal to that of the central pixel, it is assigned a value of 1; otherwise, it receives a value of 0. This sequence of values, after thresholding (either 0 or 1), is then converted into a decimal number representing the LBP value. The principle of calculating the original LBP is illustrated in Figure 4. As for the arrangement of neighbors around the central pixel, it can vary. They can be positioned at different distances from the central pixel, defined by a radius, which can be one, two, or three units, corresponding to radii r=1, r=2, and r=3, respectively. It is also worth noting that the number of neighbors can vary; it may reach 8 for r=1 and further increase for larger radii. These binary codes are then interpreted as decimal numbers, creating a range of values from 0 to 255 for each radius. These values are used to form a histogram that constitutes the LBP feature of the image.

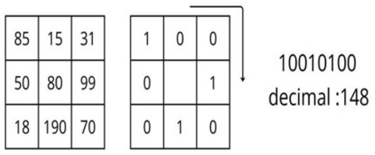
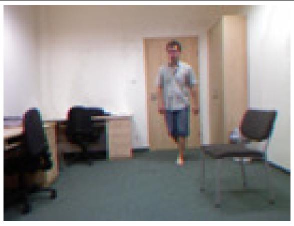
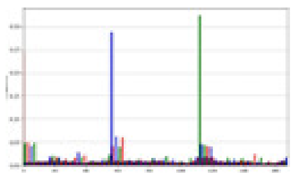
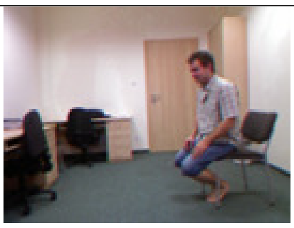
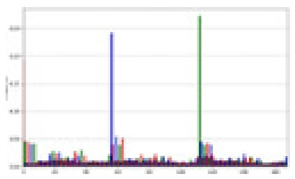
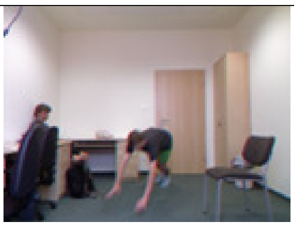
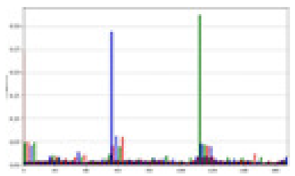
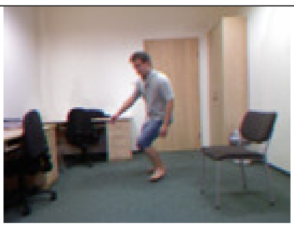
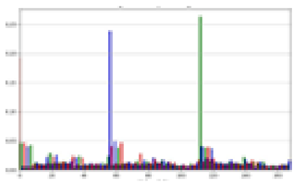
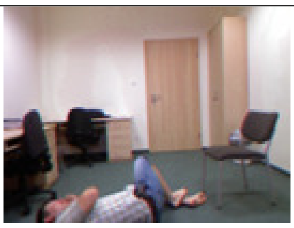
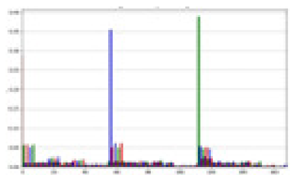


Figure 4: LBP matrix with R=1 and neighborhood = 8.

The Uniform LBP, designated as uLBP, is an LBP variant aimed at reducing the dimensionality of patterns. Unlike the traditional LBP, which can describe up to 256 different patterns for a set of 8 neighbors, the uLBP focuses on so-called "uniform" patterns, retaining only 59 of them. A pattern

Table 1: The five different postures taken into consideration

Posture	Real image	Silhouette	Histogram
1. Standing			
2. Sitting			
3. Crawling			
4. Bending			
5. Lying			

is considered uniform if it exhibits a maximum of two transitions (0-1 or 1-0) when traversing the pattern in a loop. The main advantage of the uLBP lies in its ability to focus on the most relevant and common texture patterns in images while reducing noise from less frequent patterns. In the context of our study, the use of the uLBP allows for a more compact and efficient representation of texture information while preserving essential features for posture recognition. Following the application of the uLBP at three different scales, for a standard neighborhood of 8 pixels, the resulting number of uniform patterns amounts to 59. Thanks to the three mentioned scales or radii ($r = 1, 2, and 3$), three uLBP histograms are generated for each image, each of which has a dimension of 59, corresponding to the uniform patterns. By merging the histograms from the three scales, a combined dimension of $59 \times 3 = 177$ is achieved. These 177 values make up the inputs, or features, for the recognition model (see 1).

4.1.4 Posture Recognition with the Random Forest

The choice of the Random Forest model for our study is justified by several crucial aspects related to efficiency and algorithmic complexity. According to Alfayez et al., 2023 [13], the uLBP features processing offers constant complexity, promoting a seamless integration into a distributed environment where the processing time is a determining factor. Random Forests stand out for their less complex structure and fast execution compared to the Deep Learning architectures such as CNNs or DFFNNs, whose complexity increases exponentially with the number of layers and neurons. The complexity of a Random Forest model is related to that of decision trees, which is of the order of O(n*log(n)*d), making it particularly suitable for fast processing of large data volumes. Despite the good performance achieved by theDFFNN and CNN models, we still opted for the Random Forests owing to their simplicity. However, simpler models such as logistic regression and decision trees yielded significantly lower results, which reinforced our reliance on the Random Forests for our application [13].

4.2 Fall Detection Model

The fall detection process (see 5) begins by organizing the postures into temporal sequences, each annotated with a label indicating whether a fall occurred or not. These temporal series are then integrated into a distributed Spark DataFrame, which is subsequently used for training and testing distributed LSTM/GRU models. After training, the model is evaluated using some appropriate metrics.

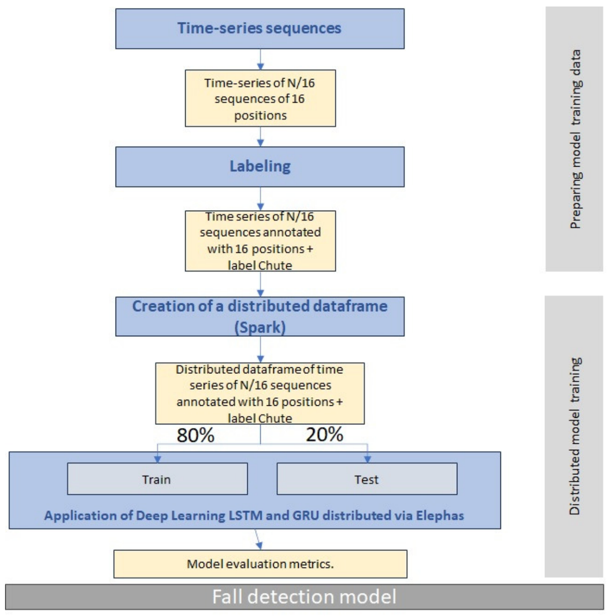

Time-series sequences
Time-series of N/16 sequences of 16 positions
Labeling
Time series of N/16 sequences annotated with 16 positions + label Chute
Creation of a distributed dataframe (Spark)
Distributed dataframe of time series of N/16 sequences annotated with 16 positions + label Chute
80%
20%
Train
Test
Application of Deep Learning LSTM and GRU distributed via Elephas
Model evaluation metrics.
Fall detection model
Preparing model training data
Distributed model training

Figure 5: Fall detection model process.

The main objective of the proposed approach was to detect falls within a predefined period using sequences of postures. Choosing appropriate algorithms for a given problem is not a trivial decision. There is certainly no ideal algorithm that works for all problems; yet, some algorithms are recognized to offer better performance than others on specific problems. In our case, algorithms such as the LSTM and GRU are most suitable, given their acknowledged ability to address time series problems, essential for analyzing the movements and postures of the monitored person over time. These models are particularly well-suited for understanding temporal dependencies and patterns in the data, which is crucial for accurately predicting fall events.

5 Experimental Study

The experimental study of our project was conducted on a Hadoop cluster (version 3.3.2). The configuration is based on virtual machines running Ubuntu 18, using Oracle VirtualBox 6.1.28. Our cluster is structured around two master nodes, each equipped with a RAM of 5 GB and two processor cores, ensuring efficient resource management with high availability. Our cluster also includes two slave nodes, each of which is fitted with a RAM of 8 GB and four processor cores, providing acceptable computing capacity for a distributed task processing. The slave nodes also have 60 GB of disk space, ensuring sufficient storage for prototyping processed data. Finally, our cluster also has an edge node representing a gateway for cluster usage without directly accessing the master. We integrated Apache

Table 2: Test and Validation databases summary.

	Test Data [27]	Validation Data [28]
Number of Video sequences	70	24
Number of images	2704	992
Camera Type	Microsoft kinect	Economic IP
Camera number	2	8

Spark 2.2.2 as the distributed engine for our Hadoop/Yarn cluster. The entire setup was installed on a PC with an Octa-core processor (providing 16 execution threads) and 40 gigabytes of RAM.

5.1 Data Collection

For the development and validation of our model, we gathered two distinct databases (2). The first was dedicated to training and testing our algorithms, while the second was used to validate the performance of our model.

5.1.1 Test Database

Our test database [27] consists of video sequences captured by state-of-the-art motion capture and computer vision devices, including Microsoft Kinect cameras and sensors such as PS Move and x-IMU. With a detailed series of 70 video sequences, including 30 devoted to simulating falls and 40 recording daily activities. This database provides the necessary data to train our model to differentiate between these two types of events (see 6).

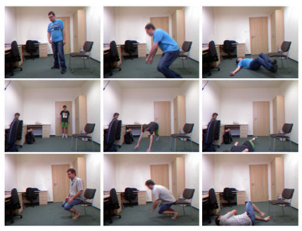


Figure 6: Example of image sequence for the test database.

5.1.2 Validation Database

The second image database used for validation [28] relies on a multi-camera system with eight economical IP cameras, arranged to fully cover a room. The recorded video sequences contained several common challenges such as high video compression, shadows, reflections, a complex background, lighting variations, worn objects, and occlusions, which may lead to segmentation errors. The dataset includes a variety of daily activities and simulated falls (see Figure 7), performed by a subject and captured from different angles by all the cameras. Falls were performed with precaution, using a mattress to protect the subject. This dataset represents a valuable tool for research and applications aimed at improving the segmentation and detection techniques for human activities in complex environments.

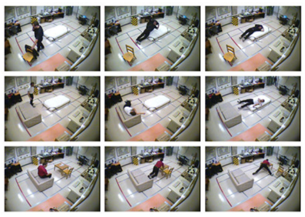


Figure 7: Image sequence for the validation data base.

5.1.3 Reference Image

A reference image is a photograph captured in a specific environment without the presence of mobile elements or subjects, such as people (see Figure 8). This photo serves as the basis for the silhouette extraction process.

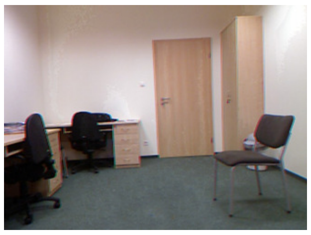


Figure 8: Example of image reference for the test database.

5.2 Data Preparation for Fall Detection Models

5.2.1 Temporal Series Sequencing

Once postures are determined via the Random Forest model on Spark, postures are grouped into temporal series. A temporal series, sometimes referred to as a time series, is an ordered collection of data associated with specific moments. Depending on the context, the considered time interval may vary. The dimension of this data, denoted n represents the length of the series. Temporal series aggregate information points collected at consecutive moments. They can be multivariate and often reveal trends, periodicities, and various inherent features. To make the data suitable for our model, posture series from the used images by our model are grouped into a sequence of sequences containing 16 consecutive postures. This choice is not arbitrary; we opted for such value because it corresponds to a duration of approximately 0.8 seconds for a video captured at a frequency of 20 frames per second: The initial formula for free fall is estimated as:

$$h = \frac{1}{2}gt^2 \tag{1}$$

Where:

- h represents the height of the fall in meters
- g represents the acceleration due to gravity, which is approximately 9.81 m/s^2
- t represents the duration of a fall in seconds

The duration of a fall t can be expressed with the following formula:

$$t = \sqrt{\frac{2h}{g}} \tag{2}$$

According to the organization "Our World in Data", the approximate average height of humans during the current century is estimated between 1.6 meter (for women) and 1.7 meter (for men). To measure the height of a fall, we often consider the center of mass of the person located at about 57% of the height (+3%) which represents about one meter from the average height. With a numerical application via formula (2), the theoretical duration of a fall is estimated at 0.45 seconds. Knowing that for the two image databases studied, the cameras take 20 images per second, we chose the size of our frame to be 16 images, allowing us to cover 0.8 seconds, which represents a window long enough to capture the entire fall.
Human movements, especially falls, are often rapid and of short duration. Capturing these movements requires great attention to be able to detect nuances and subtleties that may indicate an impending or ongoing fall. By choosing an 0.8-second window, we can capture this essential information while limiting the impact of superfluous or irrelevant movements.

5.2.2 Data Labeling

Data manual annotation is an important step in preparing datasets for our fall detection model. This task involves a careful examination of each sequence of images and to determine whether it represents a fall or not. Once a sequence is identified as representing a fall, a label of "1" is assigned to that sequence; if not, a label of "0" is assigned. This label is then added to the 17th column of the sequence, acting as a label column for supervised learning.

5.2.3 Model Hyperparameters

Several hyperparameter combinations were tested to find the combination that results in the lowest error rate throughout the test set. In particular, the following hyperparameters were evaluated:

- Batch size: Values between [100, 500] were tested based on the size of the training data.
- Number of epochs: A range of values between [128, 1500] was tested.
- Number of layers: We explored the use of a varied number of LSTM/GRU layers, testing a range from 1 to 8 hidden layers along with a corresponding number of Dropout layers to prevent overfitting.
- Units per layer: For the LSTM, GRU layers, we varied the number of LSTM, GRU units corresponding to the number of neurons in each hidden layer, testing a range from 20 to 300 units.
- Input sequence length: A sequence length of 16 was adopted as an input after careful consideration of the pace of human falls. This length, equivalent to 0.8 seconds of video at a rate of 20 frames per second, is specifically chosen to optimally capture all significant movements of a fall.
- Learning rate: The learning rate was chosen by testing a range of values between 0.1 and 0.0001, following standard practices. This range of values was tested to identify the rate that allows our model to learn optimally without being too aggressive, which could cause jumps over optimal minima, or too slow, leading to a very slow and inefficient convergence.
- Activation function: We selected specific activation functions to take advantage of their distinct benefits: the ReLU function for the LSTM/GRU layers, owing to its ability to effectively model nonlinear behaviors and reduce the risk of gradient vanishing; the Hard Sigmoid function for the recurrent LSTM/GRU activations to speed up computation while managing the information flow through cells; and finally, the Sigmoid function for the output layer since it is highly suitable for binary classification thanks to its normalized output interpretable as a probability.
- Loss function: the binary crossentropy loss function was used as it is particularly suitable for binary classification problems; indeed, it quantifies the model performance by measuring the distance between predictions and actual labels.
- Deactivation dropout: The probability used for Dropout layers is 0.5. This means that during training, each weight update randomly deactivates 50% of the neurons output in the previous layer. This value is often used because it offers a good compromise between regulating efficiency and maintaining network performance. It significantly reduces the risk of overfitting.

5.3 Model Evaluation and Discussion

The hyperparameters that yielded the best results were as follows: an input sequence length of 16, 256 units per layer, 500 epochs, a learning rate of 0.01, a batch size of 16, and a dropout probability of 50%. Our experimental fall detection model was evaluated by calculating a set of standard metrics namely Accuracy, Precision, Recall, F1-score, Specificity and Area Under Curve [48] (Table 3 and 4).

Tables 3 and 4 illustrate the results obtained by applying our approach to the test and validation data. It is evident that our approach with the model using GRU outperforms the one that applies the

Table 3: Comparison of results on the test database.

Method	Accuracy	Precision	Recall	Specificity	F1-score	AUC
Our approach with LSTM	94.7%	90.1%	96%	90%	92,95%	93,56%
Our approach with GRU	97%	100%	93.7%	100%	96,74%	97,70%
Wang et al (2020) [47]	97.33%	97.78%	97.78%	96,67%	97,78%	-
Harrou et al (2019) [48]	96.66%	94%	100%	0,94%	96,60%	97%
Kwolek and kepski (2014) [27]	90%	83.30%	100%	80%	-	-
Dai et al (2018) [49]	94,4%	-	95%	96,7%	-	-
Raza et al (2023) [52]	95,89 %	95,54%	95,38%	-	95,47%	-
Kan et al (2023) [53]	-	90,4%	89,1%	-	-	-
Wang et al (2024) [54]	89,99%	-	90,33%	-	-	-
Yuan et al (2022) [55]	97,43%	-	95,45%	100%	-	-

Table 4: Comparison of results on the validation database.

Method	Accuracy	Precision	Recall	Specificity	F1-score	AUC
Our approach with LSTM	95.7%	88.7%	100%	93.75%	95,89%	93,74%
Our approach with GRU	98.9%	96.8%	100%	98.4%	98,98%	96,78%
Rougier et al (2011) [20]	98%	-	95.4%	95.8%	-	97,8%
Auvinet et al (2011) [28]	-	-	80.6%	100%	-	-
Agrawal al (2023) [50]	-	-	92%	96%	-	-
Mousse et al (2017) [51]	-	-	95.8%	100%	-	-
Yuan et al (2022) [55]	-	88,13%	86,66%	-	87,38%	-
Song et al (2021) [56]	97,23%	-	100%	97,04%	-	-
Chehtiri et al (2021) [57]	92,91%	-	-	-	-	-
Geng et al (2022) [58]	98,55%	-	-	-	-	-

LSTM. Our GRU-based approach is also highly competitive compared to approaches applied to the same test and validation data.

Our study stands out in the issue of fall detection research by developing the first scalable Big Data distributed Deep Learning model, specifically designed to operate in parallel on a Hadoop cluster by efficiently exploiting the distributed data. Unlike previous studies, this innovative approach harnesses the combined power of multiple CPU or Graphics Processing Unit (GPU) dealing with units from a set of servers in a Big Data cluster, able to go beyond the limits of the computational capacity of a single computer with a single CPU or GPU. We have proved that our distributed model offers highly competitive performance with an accuracy rate of 94.7% on the test database and 98.9% on the validation one. However, it is worth noting that our work focused on relatively modest-sized datasets. These results are particularly significant compared to those of other studies in the literature. It is true that such studies are while effective; however, they neither rely on distributed architectures nor are they scalable Big Data. The ability to train our model on a distributed cluster opens up promising prospects for real-time and large-scale applications. A distributed training allows us to leverage extended computing resources, potentially capable of handling terabytes of data, which is a significant advancement compared to conventional methods limited by the capacity of a single CPU or GPU. This advancement aligns with a vision where technology adapts to the growing needs of monitoring and prevention in the home healthcare sector, offering solutions that are both smarter and more responsive for the safety of dependent individuals.

In a perspective of continuous innovation, we are planning to integrate various types of sensors, which will enable us to avoid being restricted by single cameras and rely on multi-sensor systems. This diversification will allow more comprehensive and accurate monitoring by providing varied viewpoints and a broader coverage of living spaces. Another perspective is to improve our system so as to detect and track multiple people simultaneously within the same field of view. The integration of facial recognition could target and monitor specific individuals within a group, thus enhancing the personalization of surveillance and the speed of intervention in the event of a fall. Despite the limited size of our databases, we were able to validate the proposed model. Nevertheless, to truly align with Big Data challenges, it will be essential to evaluate and optimize our model on much larger data volumes. This approach is crucial to ensuring the scalability and adaptability of our solution in diverse and large-scale environments. The current discussion around the results highlights the need for such expansion, as well as the possibility of widening the scope to include real and complex life scenarios.

6 Conclusion

By combining the robustness of Hadoop/Spark with advanced image processing and deep learning methods, we have developed a solution that does not only detects falls effectively but also values privacy through the integration of uLBP histograms. Our work stands out in the field of fall detection research by developing the first scalable Big Data distributed deep learning model, specifically designed to operate in parallel on a Hadoop/Spark cluster through an efficient exploitation of distributed data. This innovative approach harnesses the combined power of several CPUs or graphics processing units (GPUs) on a set of servers in a Big Data cluster, surpassing the limits of the computational capacity of a single computer with a single CPU or GPU. The proposed distributed model has proven its competitive performance achieving accuracy rates of 97% and 98.9% on the test database and validation databases, respectively. However, it is worth noting that our work focused on relatively modest-sized datasets. To really address Big Data challenges, it is essential that the suggested model be evaluated and optimized on much larger databases. This approach is crucial to ensuring the scalability and adaptability of our solution in diverse and large-scale environments. Advances such as optimizing existing models, improving the integration of Internet of Things (IoT) devices, and incorporating generative artificial intelligence could further enhance the system accuracy and responsiveness. The development of more intuitive user interfaces will improve the care provided by healthcare professionals, strengthen family support networks, and ensure continuous monitoring and rapid intervention in case of sudden incidents.

Declarations

Competing interests

The authors certify that they have NO affiliations with or involvement in any organization or entity with any financial interest or non-financial interest (such as personal or professional relationships, affiliations, knowledge or beliefs) in the subject matter or materials discussed in this manuscript

Authors' contributions

Heni Bouhamed and Monia Hamdi conceived of the presented idea. Fady Badreddine developed the theory and performed the computations. Reem Alkanhel and Monia Hamdi verified the analytical methods. All authors discussed the results and contributed to the final manuscript.

Funding

The authors extend their appreciation to the Deputyship for Research & Innovation, Ministry of Education in Saudi Arabia for funding this research work through the project number RI-44-0559

Availability of data and materials

Below the link to all codes (python-spark) and data with explanation (readme):
https://github.com/fadybadreddine/Keras-with-Spark-using-Elephas-for-fall-detection-

References

[1] S. Heinrich, K. Rapp, U. Rissmann, C. Becker, and H.-H. König, "Cost of falls in old age : A systematic review," Osteoporosis international : a journal established as result of cooperation

between the European Foun- dation for Osteoporosis and the National Osteoporosis Foundation of the USA, vol. 21, pp. 891–902, 11 2009.

[2] N. Noury, A. Fleury, P. Rumeau, A. Bourke, G. ÓLaighin, V. Rialle, and J.-E. Lundy, “Fall detection – principles and methods,” Conference pro- ceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference, vol. 2007, pp. 1663–6, 02 2007.

[3] S. Herath, M. Harandi, and F. Porikli, “Going deeper into action recogni- tion : A survey,” Image and Vision Computing, vol. 60, 05 2016.

[4] J. Peña Queralta, T. Nguyen gia, H. Tenhunen, and T. Westerlund, “Edge- ai in lora-based health monitoring : Fall detection system with fog compu- ting and lstm recurrent neural networks,” 07 2019.

[5] D. Lie, B. Nukala, N. Shibuya, A. Rodriguez, J. Tsay, T. Nguyen, S. Zu- pancic, and D. Lie, “A wireless gait analysis sensor for real-time robust fall detection using an artificial neural network,” 10 2014.

[6] S. Gharghan, S. Mohammed, and A. A. Al-Naji, “Accurate fall detection and localization for elderly people based on neural network and energy- efficient wireless sensor network,” Energies, vol. 11, p. 2866, 10 2018.

[7] M. Majd and R. Safabakhsh, “A motion-aware convlstm network for ac- tion recognition,” Applied Intelligence, vol. 49, pp. 1–7, 07 2019.

[8] Y. Li, Y. Guanci, Z. Su, S. Li, and Y. Wang, “Human activity recognition based on multi environment sensor data,” Information Fusion, vol. 91, pp. 47–63, 10 2022.

[9] G. Diraco, A. Leone, and P. Siciliano, “Human posture recognition with a time-of-flight 3d sensor for in-home applications,” Expert Systems with Applications, 02 2013.

[10] F. Modarres and M. Soryani, “Body posture graph : A new graph-based posture descriptor for human behavior recognition,” Computer Vision, IET, vol. 7, pp. 488–499, 12 2013.

[11] B. Boulay and M. Thonnat, “Applying 3d human model in a posture re- cognition system,” Pattern Recognition Letters, vol. 27, pp. 1788–1796, 11 2006.

[12] H. Li and Q. Sun, “The recognition of moving human body posture based on combined neural network,” pp. 1–5, 01 2013.

[13] AlFayez, F.; Bouhamed, H. (2023). Machine learning and uLBP histograms for posture recognitionof dependent people via Big Data Hadoop and Spark platform,International Journal of ComputersCommunications & Control, 18(1), 4981, 2023.https://doi.org/10.15837/ijccc.2023.1.4981

[14] D. Lord, C.J. ; Colvin, “Falls in the elderly : Detection and assessment,” IEEE Annual International Conference of the IEEE Engineering in Medi- cine and Biology Society, vol. 13, p. 1938—1939, 02 1991.

[15] G. Williams, K. Doughty, K. Cameron, and D. Bradley, “A smart fall and activity monitor for telecare applications,” vol. 3, pp. 1151 – 1154 vol.3, 01 1998.

[16] A. Bourke, J. O’Brien, and G. ÓLaighin, “Evaluation of a threshold-based tri-axial accelerometer,” J Gait and Posture, vol. 26, pp. 194–199, 01 2006.

[17] A. Bourke and G. ÓLaighin, “A threshold-based fall-detection algorithm using a bi-axial gyroscope sensor,” Medical engineering physics, vol. 30, pp. 84–90, 02 2008.

[18] P. Tsinganos and A. Skodras, “On the comparison of wearable sensor data fusion to a single sensor machine learning technique in fall detection,” Sensors, vol. 18, p. 592, 02 2018.

[19] Q. Li, J. Stankovic, M. Hanson, A. Barth, J. Lach, and G. Zhou, “Accurate, fast fall detection using gyroscopes and accelerometer-derived posture in- formation,” pp. 138–143, 06 2009.

[20] C. Rougier, J. Meunier, A. St-Arnaud, and J. Rousseau, “Robust video surveillance for fall detection based on human shape deformation,” Circuits and Systems for Video Technology, IEEE Transactions on, vol. 21, pp. 611– 622, 06 2011.

[21] G. Diraco, A. Leone, and P. Siciliano, “An active vision system for fall detection and posture recognition in elderly healthcare,” pp. 1536–1541, 03 2010.

[22] L. Xue, L. Nie, H. Xu, and X. Wang, “Collaborative fall detection using smart phone and kinect,” Mobile Networks and Applications, vol. 23, 08 2018.

[23] M. Saleh and R. Le Bouquin Jeannès, “Elderly fall detection using wea- rable sensors : A low cost highly accurate algorithm,” IEEE Sensors Jour- nal, vol. PP, pp. 1–1, 01 2019.

[24] T. Wu, Y. Gu, Y. Chen, Y. Xiao, and J. Wang, “A mobile cloud collabora- tion fall detection system based on ensemble learning,” 07 2019.

[25] Q. Han, H. Zhao, W. Min, H. Cui, X. Zhou, K. Zuo, and R. Liu, “A two- stream approach to fall detection with mobilevgg,” IEEE Access, vol. PP, pp. 1–1, 01 2020.

[26] A. Shojaei, P. Nasiopoulos, J. Little, and M. Pourazad, “"video-based hu- man fall detection in smart homes using deep learning",” pp. 1–5, 05 2018.

[27] B. Kwolek and M. Kepski, “Human fall detection on embedded platform using depth maps and wireless accelerometer,” Computer Methods and Programs in Biomedicine, vol. 117, p. 489–501, 10 2014.

[28] E. Auvinet, C. Rougier, J. Meunier, A. St-Arnaud, and J. Rousseau, “Multiple cameras fall data set,” 01 2011.

[29] T. O. M. P. D. Harwood, “A comparative study of texture measures with classification based on featured distributions,” Pattern Recognition, vol. 29, p. 51—59, 1996.

[30] L. Breiman, “Random forests,” Machine Learning, vol. 45, pp. 5–32, 10 2001.

[31] L. MARKÉTA KRÚPOVÁ, Construction d’un modèle de Machine Lear- ning interprétable pour la tarification en assurance non-vie. PhD thesis, Université de Paris-Dauphine, 12 2022.

[32] R. DE, H. GE, and W. RJ, Learning internal representations by error pro- pagation : Parallel Distributed Processing, Volume 1 : Foundations. 01 1986.

[33] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” Nature, vol. 521, pp. 436–44, 05 2015.

[34] Y. Bengio, P. Simard, and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council, vol. 5, pp. 157–66, 02 1994.

[35] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” Neural Computation, vol. 9, pp. 1735–1780, 11 1997.

[36] F. Gers, J. Schmidhuber, and F. Cummins, “Learning to forget : Continual prediction with lstm,” Neural computation, vol. 12, pp. 2451–71, 10 2000.

[37] c.-m. Own, F. Sha, and W. Tao, “Triplet decoders neural network ensemble system and t-conversion for traffic speed sequence prediction,” IEEE Ac- cess, vol. PP, pp. 1–1, 11 2019.

[38] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” 12 2014.

[39] S. Abdulwahab, M. Jabreel, and D. Moreno, Deep Learning Models for Paraphrases Identification. PhD thesis, 09 2017.

[40] "Hdfs," https ://hadoop.apache.org/docs/stable/hadoop-project- dist/Hadoop hdfs/HdfsDesign.html. Consulté le 12/06/2023.

[41] "Yarn," https ://hadoop.apache.org/docs/current/hadoop-yarn/hadoop- yarn-site/YARN.html. Consulté le 12/06/2023.

[42] "opencv," https ://opencv.org/. Consulté le 12/06/2023.

[43] scikit-image developers, "scikit-image : Image processing in python," 2023.

[44] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., "Tensorflow : A system for large- scale machine learning," in 12th USENIX Symposium on Operating Sys- tems Design and Implementation (OSDI 16), pp. 265–283, 2016.

[45] F. Chollet et al., "Keras," GitHub, 2015.

[46] M. Pumperla, "elephas," GitHub, 2015.

[47] B.-H. Wang, J. Yu, K. Wang, X.-Y. Bao, and K.-M. Mao, "Fall detection based on dual-channel feature integration," IEEE Access, vol. PP, pp. 1–1, 06 2020.

[48] F. Harrou, N. Zerrouki, Y. Sun, and A. Houacine, "An integrated vision- based approach for efficient human fall detection in a home environment," IEEE Access, vol. PP, pp. 1–1, 08 2019.

[49] B. Dai, D. Yang, L. Ai, and P. Zhang, "A novel video-surveillance-based algorithm of fall detection," pp. 1–6, 10 2018.

[50] M. Agrawal and S. Agrawal, "Enhanced deep learning for detecting suspicious fall event in video data," Intelligent Automation Soft Computing, vol. 36, pp. 2653–2667, 01 2023.

[51] M. Mousse, C. Motamed, and E. Ezin, "Percentage of human-occupied areas

[52] A., Raza, M. H., Yousaf, S. A., Velastin, & S. Viriri, (2023). Human Fall Detection from Sequences of Skeleton Features using Vision Transformer. In VISIGRAPP (5: VISAPP) (pp. 591-598) 2023.

[53] X., Kan, S., Zhu, Y., Zhang, & C. Qian, (2023). A lightweight human fall detection network. Sensors, 23(22), 9069.

[54] Y., Wang, & T. Deng, (2024). Enhancing elderly care: Efficient and reliable real-time fall detection algorithm. Digital health, 10, 20552076241233690 2024.

[55] C., Yuan, P., Zhang, Q., Yang, & J. Wang, (2022). Fall detection and direction judgment based on posture estimation. Discrete dynamics in nature and society, 2022.

[56] S., Zou, W., Min, L., Liu, Q., Wang, & X., Zhou, (2021). Movement tube detection network integrating 3d cnn and object detection framework to detect fall. Electronics, 10(8), 898, 2021.

[57] S., Chhetri, A., Alsadoon, T., Al-Dala'in, P. W. C., Prasad, T. A., Rashid, & A. Maag, (2021). Deep learning for vision-based fall detection system: Enhanced optical dynamic flow. Computational Intelligence, 37(1), 578-595.

[58] P., Geng, H., Xie, H., Shi, R., Chen, & Y. Tong, (2022). Pedestrian Fall Event Detection in Complex Scenes Based on Attention-Guided Neural Network. Mathematical Problems in Engineering, 2022.

Cite this paper as:

Hamdi, M.; Bouhamed, H.; Badreddine, F.; Alkanhel, R. (2024). Deep recurrent neural networks distributed on a Hadoop/Spark cluster for fall detection, *International Journal of Computers Communications & Control*, 19(3), 6428, 2024.
https://doi.org/10.15837/ijccc.2024.3.6428