Automatic Fall Detection for the Care of Older Adults in Smart Cities

Sara Judith Ríos Dueñas, Jose Mejia, Alberto Ochoa, Jose Díaz, Lidia Rascon, Nelly Gordillo, and Eddy Sánchez-DelaCruz

Abstract As the number of elderly people increases, it is a necessity for smart cities to take care of elder special needs. As people age, the likelihood of accidents increases because of their motor skills decrease over time, this risk is not only latent when they live alone but also exists within the nursing homes. Because of this, constant care for older adults is a necessity for smart cities, this may not always be possible due to the lack of family members who can care for the elder or in the case of the nursing homes, the staff may not be enough to care for all adults. This leaves the need for systems that can constantly monitor older adults and respond and alert automatically in the event of accidents. In order to find a means to improve the quality of life of older adults who may suffer accidents, in this chapter it is presented an algorithm based on neural networks for the automatic fall detection of older adults in nursing homes. The implemented model was trained and tested on a database of video images containing fall situations.

Keywords CNNs architecture · Fall detection · Smart city

1 Introduction

One of the smart cities aims, is to provide more flexible, efficient, and sustainable services to its citizens. These services include (Mohanti et al. 2016) [24] smart infrastructure, smart transportation, smart energy, and smart healthcare. In this chapter we will focus on this last aspect of smart cities.

As life expectancy has increased the number of elderly people has also increased, this creates a necessity for smart cities to take care of elder special needs [8]. Older people are more prone to suffer accidents or put themselves in risk situations because

S. J. R. Dueñas · J. Mejia (✉) · A. Ochoa · J. Díaz · L. Rascon · N. Gordillo
Universidad Autónoma de Ciudad Juárez, Av. Hermanos Escobar, Ciudad Juárez, Mexico
e-mail: jose.mejia@uacj.mx

E. Sánchez-DelaCruz
Instituto Tecnológico Superior de Misantla, Veracruz, Mexico

A. Ochoa-Zezzatti et al. (eds.), *Technological and Industrial Applications Associated With Industry 4.0*, Studies in Systems, Decision and Control 347,
https://doi.org/10.1007/978-3-030-68663-5_4

their motor skills decrease over time, thus they needed a constant attention [5, 25, 26]. The risk is not only latent when they live alone but also exists within nursing homes. Older adults who are residents of the nursing homes may also suffer falls, among other accidents, or put themselves in risk situations, because they cannot be constantly taken care of. The absence of staff could be because they can be found attending to someone else patient, thus neglecting the other residents, or because insufficient personnel, therefore, it is important that those in charge of these residences improve their accident prevention and care system [7, 27].

These problems prompted for the incorporation of algorithms that can carry out automatic fall detection in older adults in real time, by means of portable devices. In recent times several proposals have opted for the use of neural networks to track people [1]. For example, structures were proposed to achieve the detection of falls using an Elman's neural network, that monitor older adults using portable devices [2]. Similarly, in [6] a system was designed and implemented for the fall detection in indoor spaces using WiFi devices. The main contribution is that it has greater comfort for users since they do not have to be charged with a device, which allows them to perform their daily activities with greater comfort and safety, in addition to that it was implemented in the design a base signal with greater sensitivity for the detection of activity, which allows to have a greater segmentation for the fall detection or activities similar to these.

Here, the design of an architecture, based on convolutional neural networks, is proposed for the detection of falls in older adults.

2 Theory

In this section is carry out a revision of the subjects of machine learning and specifically the topic of deep learning architectures. Additionally, a revision of the basic concepts of video processing is done.

2.1 Machine Learning

In the area of computer science, various methods have been created for the purpose of facilitate complex processes carried out by human beings. This area is called machine learning (ML), which is a set of techniques for training machines that are capable of learn to solve several problems. In Fig. 1, it is shown a graphic representation of the basic components of ML [3, 9].

Deep learning is a branch of the ML that has been developed for the design of algorithms based on artificial neural networks (ANN), in this branch different architectures have been de-signed that resemble biological ANN, thus creating a computational counterpart focused on the resolution of all kinds of problems, besides that this allows massive handling of information.

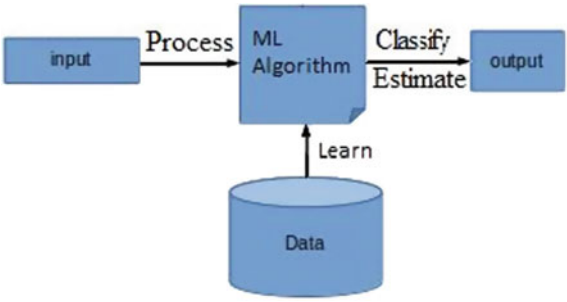

Fig. 1 Graphic representation of the basic components of ML

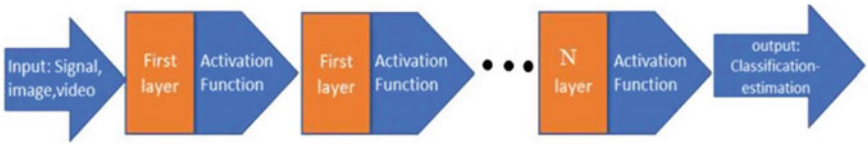

Fig. 2 Basic components of deep learning architectures, a "N" layers network

In deep learning, the word architecture refers to structures that have been designed in which neurons are connected within it, these structures also determine the behaviour of the network for carry out different tasks [9]. The general structures of the deep learning architectures are the layers, activations functions, and training parameters (Fig. 2).

2.2 Layers

The grouping of neurons is referred to as a layer and the number of layers used will determine the degree of an architecture [9, 10].

In [10] it is mentioned that the basic neural network structure must have three basic elements based on layers, which are: input layer, in which all the information from the external environment is received; the hidden layer, in which the information is processed and has no contact with the outside environment; and the output layer, which delivers a response from the neural network. The connections weighs of neurons within a network are changed during the learning stage to adapt to the problem at hand [10].

In [9] it is mentioned that the design of different neural networks architectures is important since the complexity of certain problems can vary. Neural networks are compared to Lego blocks since with these you can build almost any structure you want, like-wise with neural networks. The way in which a neural network is designed will depend on the skill and mastery of the designer.

There are several libraries focused on the management and design of neural networks such as *tensorflow* and *keras* [9].

There are multiple types of neural networks, among which the most essential are the multilayer perceptron neural network (MLP) that was one of the first models of neural network; the recurrent neural network (RNN), used mainly for language modelling due to their gradient re-duction characteristics; and the convolutional neural network (CNN), which is used, among other applications, for object tracking in videos [9, 11]. The CNN will be the most relevant for the porposes of the present chapter.

The CCNs are used for the processing of topology information in grid form, examples of these are time series data such as 1D cells, image data, such as 2D pixel cells, image classification, tracking and recognition of objects, the use of convolutional neural networks have had great success in practical applications [12, 13]. The term convolutional refers to a mathematical operation called convolution that is a type of linear operation [12].

3 Proposed Methodology

An architecture based on a convolutional neural network, consists of two main stages, the convolution stage and a second stage called pooling or reduction, which is used in most convolutional neural networks [12, 13].

The convolution function is an operation of the arguments of two functions, in the case of neural networks, the first argument would be the input layer (input) and the second argument a filter (kernel) between which they are applied. In the convolution operation, one of the advantages of this method is that the filter is used to obtain the same characteristics throughout the entire layer and this is of great help since it allows the reduction of training parameters and neural connections [12–14].

In [12] it is mentioned that convolution can help improve a neural network architecture by reducing interactions or reducing weights since the kernel is smaller than the input.

The pooling function consists of a reduction in size of the input matrix, this is done by means of a statistical extraction like the maximum of the blocks of the input matrix, or the average, or the minimum, among others. It is important mention that just as this technique has its advantages in reduction it also contains disadvantages such as loss of precision [14].

A typical convolutional layer is made up of several parallel convolutions that produce linear outputs, these are fed to a nonlinear input function, that optionally is followed by the application of the pooling function to modify the output in a layer [12].

3.1 Activation Functions

In [15] it is mentioned that the biological counterpart of artificial neural networks has states, when they are active or vice versa, likewise artificial ones have this characteristic. In the case of artificial neural networks those states are generated by "activation functions", these functions express when the state in which the architecture neuron is working is active or inactive. These functions can be linear or nonlinear such as the functions sigmoid or hyperbolic tangent. Linear: in the linear activation function, the output is a linear function of the input, while in the case of nonlinear functions, several types are used for example: the Sigmoid activation function where the output is mapped within the range of 0 to 1, so this affects the activation function on its slope [15]; The Hyperbolic tangent activation function where the output is within a range from −1 to 1 [15].

3.2 Training Parameters of a Neural Network

Neural networks have a great capacity for learning and to achieve their proper functioning must go through a stage called training patterns. The objective of the training is the adjustment of the network with the data and the proper development of the network to make it capable of performing a task [16, 17].

Training can be classified into two types, supervised training, unsupervised training.

- Unsupervised learning: In this type of training the weights of the network must be adjusted according to the correlation that exists in the input data, in this type of network training data is entered at the input, but it is not known what the output will be [17].
- Supervised learning: in this type of training the neural network is presented with certain data at the entrance, for training. This mode of training the network has the purpose of adjusting weights so that the output generated by the neural network is similar enough to the input. Supervised training in a few words performs the function of network supervision [16].

3.3 Tools Used for the Development of ANN Architectures

There exist several tools facilitating the programming of architectures (see Table 1), in the next section we describe some of the most common.

Pyhton Programming Language

In [21] it is mentioned that Python is a program for the design of algorithms and is a "high-level, interpreted and multipurpose programming language". This program has

Table 1 Available software to implement ANN

Software	Open source	Link
Computational	Yes	cntk.ai
Network toolkit		
Deeplearning4j	Yes	deeplearning4j.org
Caffe	Yes	caffe.berkeleyvision.org
Theano	Yes	deeplearning.net/software/theano
Torch	Yes	torch.ch
Python	Yes	python.org
TensoFlow/Keras	Yes	tensorflow.org
WEKA	Yes	cs.waikato.ac.nz/ml/weka/

great versatility because it can be operated on various operating systems or platforms. Python has several libraries, for the development of different types of algorithms, so it is important to have them to start a project as this will facilitate the design. This program has special libraries for the design of neural networks, for example: tensor flow; pytorch; and keras, which allow the management of highly complex architectures. In the following subtopics, some of the libraries used for the design of neural networks will be discussed.

Tensor Flow
It is a platform designed to work and design ML architectures, since it is an ecosystem with varied libraries and tools that facilitate the design of this type of algorithms. This platform is focused on solving real-world problems by applying deep learning [22].

Keras
It is a Python library designed for the management of neural networks, in addition that its main focus is to speed up the process of developing an algorithm by reducing the time to design and allows working on *tensorflow*. It is a highly friendly library for the design of neural networks and allows the management and creation of CNN and RNN type networks, including the combination of both. *Keras* has a special library for the design of convolutional neural networks called "keras api functional" [23].

3.4 Video Processing

There are several video formats and each one has different characteristics; it is important to know this information since if you want to work with video in neural networks you must have a library that has the ability to read the format that is being used. Below is a list of some common video formats:

- AVI: Audio Video Interleave, one of the advantages of this format is that it can be read by almost all decoders, however, the weight of the videos is high, occupying more memory space compared to other formats.
- WMV: this type of digital video format was created by the Microsoft company.
- FLV: this format was developed by the adobe flash player company and is of high quality, one of the most important advantages is that it is accepted by almost all operating systems.
- MOV: format developed by the Apple company, audio and video; this format is exclusive for devices belonging to the company

3.5 Action Recognition Through Video Sequences

The line of investigation of the recognition of actions in humans has focused on the processing of signals and images, in recent times the analysis of video images has been incorporated for the recognition of actions for the development and design of strategies to analyse images of video, in order to be able to detect threats or risk situations through video surveillance using cameras [18].

Strategies have been investigated that allow the automatic recognition of activities and patterns in which there is movement in order to track objects or people, for the detection or analysis of events [18]. The techniques frequently used for recognition of actions are, Fisher vectors, Hof system, neural networks and dense trajectories, among others [19].

The recognition of actions can be carried out in different ways, either by robot vision or a human computer interaction. Most of the studies that recognize actions, use video recordings in 2D cameras, even though, human actions are generally performed in a 3D space, this is mainly due because 2D video recording is pervasive, in this chapter we focus on this type of input (2D), since is the most commonly used in nursing homes.

For the analysis of actions, the human body must be recognized, the human body could be considered as a system in which limb and trunk movements are combined. Some methods that perform the recognition used the joints and trunk of the skeleton, such as the work of [6] which use Temporary Pyramids to organize and analyse joint data.

The recognition of actions has become a highly active, and most of the new developments are done using ANN, however, these developments have not had the same progress in recognition as the methods based on more classic methods. Part of the use of neural networks not having the same progress is because in some cases the databases used are too small for network train-ing or have noise [6].

The use of neural networks for the recognition of actions has had a greater challenge compared to the simple classification of images due to the variations of movement in the video, in addition to the need for a greater number of examples in the training of the network to extract as much information as possible and classify the actions. On most architectures, this has been solved by applying an architecture of

two channels, one temporary and one spatial to separately train the networks in which the actions and appearances are separated, that is, the area where the action is taking place. In the temporal channel the network is classifying everything that entails the action, while in the spatial channel, the space where the action is performed is observed [19].

4 Methods

In this section, the scheme of the procedure for the development of a convolutional architecture for the aim of detect falls will be presented. The algorithm was trained with video images for the detection of falls of older adults.

The database was obtained from *Lei2- Laboratoire Electronique*, Informatique et image UMR CNRS 6306 Fall detection Dataset [20, 28]. This database shows different situations of daily life where accidents happen. It was designed for the prevention of reports of injuries, falls of older adults which are of the main causes that considerably reduce mobility and independence. The video is recorded at 25 frames/s and each with the resolution of 320 × 240 pixels, with different lengths of each video. The database consists of 191 videos in *Audio Video Interleave* (.avi) format. Also, note that for the making of this dataset, "videos are anonymous, and all those visible in the videos signed an authorization of diffusion and use of these videos for research purpose only" [20].

4.1 Fall and Non-fall Labelling

In this stage, the classification of the frames of each video was done manually. Each video was observed frame by frame until it is found the moment of the fall, then the video was di-vided into two parts (fall/no fall).

It is important to mention that in the videos provided by the database, after the fall, the actors who starred in the fall do not rise again, therefore, it is considered to be within the fall stage In Fig. 3, it is show the two parts (fall/no fall) of the same video.

4.2 Pre-processing

A maximum of 50 frames were set per video, this to reduce the computational cost and reduce training times, there were 206 videos of which only those that were more than 50 frames in length were selected Fig. 4.

Videos that had more than 50 frames were cut to have only 50 frames long. The next pre-processing step consisted in a resize of each frame. A 70 × 70 size change

Fig. 3 Video classification into fall and not fall, the left image is a video segment with not fall, and the image on the right is a fall, images taken from videos in the dataset [20, 28]

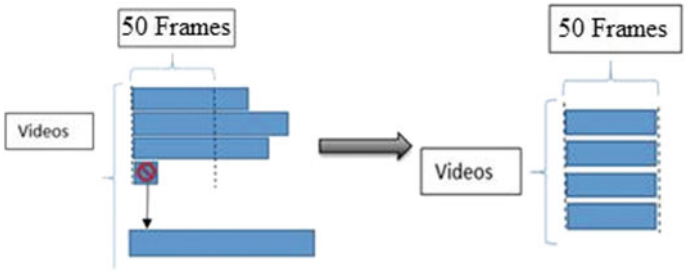

Fig. 4 From the database all videos were cut to 50 frames and videos with less than 50 frames were discarded

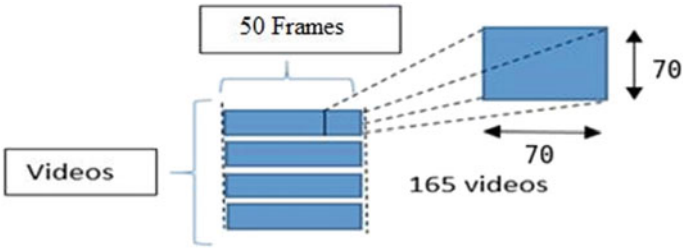

Fig. 5 Each video frame was reduced to a size of 70 × 70 pixels

was made to each frame of the videos (Fig. 5) this also was done to reduce the computational cost.

4.3 Architecture

A model was designed for the detection of falls, it was trained with video images from the database. The model has several layers which are composed of 3D convolutions, pooling stages, and dense layers. The use of 3D convolution instead of the more

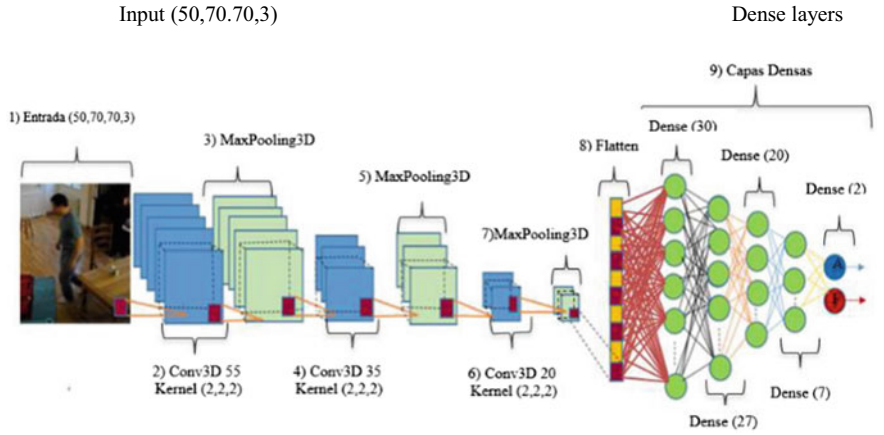

Fig. 6 Artificial neural network proposed architecture

common 2D convolution, is because of the video add another dimension (time) to the input image, thus, this extra dimension is used to obtain additional features from the video. It should be noted that in the 3D convolution, the kernels are three-dimensional structures. Each layer of the proposed architecture performs an important function for proper functioning of the model. The convolutional layers in the architecture, extract features in the video that help to classify the video, max-pooling layers are used to decrease the size of the data and computational cost, and dense layers classify the video type (fall/no fall).

The model was coded in Keras. Figure 7, provides an outline of the input dimensions and number of parameters for each layer.

Next, each layer of the model is described

Stage 1: Input

- The input has a dimension of [50 × 70 × 70 × 3] which means that the input consists of 50 frames of a size of [70 × 70] pixels × 3 channels corresponding to the RBG colors.

Stage 2: First convolutional layer (Conv3D_1)

- It is composed of 55 3D filters/kernels with each kernel of size [2 × 2 × 2].
- The output feature maps have dimension of [49 × 69 × 69, 55] obtained, which means that 55 layers of 49 frames with a size of [69 × 69] were obtained at the output, the input dimension was decreased because of the convolution operation. A linear activation was used.

Stage 3: first pooling layer (Maxpooling3D_1)

- The maxpooling layer, as shown in Fig. 6, decrease the size of the data by applying the maximum to blocks of 2 × 2 × 2 pixels, resulting in an output of [23, 34, 34,

```
[ ]
Layer (type)                 Output Shape              Param #
=================================================================
conv3d_1 (Conv3D)            (None, 49, 69, 69, 55)    1375
_________________________________________________________________
max_pooling3d_1 (MaxPooling3 (None, 24, 34, 34, 55)    0
_________________________________________________________________
conv3d_2 (Conv3D)            (None, 23, 33, 33, 35)    15435
_________________________________________________________________
max_pooling3d_2 (MaxPooling3 (None, 11, 16, 16, 35)    0
_________________________________________________________________
conv3d_3 (Conv3D)            (None, 10, 15, 15, 20)    5620
_________________________________________________________________
max_pooling3d_3 (MaxPooling3 (None, 5, 7, 7, 20)       0
_________________________________________________________________
flatten_1 (Flatten)          (None, 4900)              0
_________________________________________________________________
dense_1 (Dense)              (None, 30)                147030
_________________________________________________________________
dense_2 (Dense)              (None, 27)                837
_________________________________________________________________
dense_3 (Dense)              (None, 20)                560
_________________________________________________________________
dense_4 (Dense)              (None, 7)                 147
_________________________________________________________________
dense_5 (Dense)              (None, 2)                 16
=================================================================
Total params: 171,020
Trainable params: 171,020
```

Fig. 7 Keras listing of the parameter for the proposed mode

55] which are 55 layers with 24 frames of a size of [34 × 34], which are half the size of each frame of the previous layer

Stage 4: second convolutional layer (Conv3D_2)

- As can be seen from Fig. 6, the second convolution layers has 35 filters of size [2 × 2 × 2], the output dimension is now of 23 frames with a size of [33 × 33].
- This layer has the ReLu activation function, since it has a good performance with convolutional networks. This is a nonlinear function that make zero all negative values [29].

Stage 5: Max pooling (Maxpooling_2)

- The pooling layer as an output of [11, 16, 16, 35], which are 35 layers with 11 frames of a size of [16 × 16] which are half the size of each frame of the previous layer.

Stage 6: third convolutional layer (Conv3D_3)

- In Fig. 6, it is shown that in the third convolution has an output of 20 layers each with 10 frames with a size of [15 × 15].

Stage 7: Max pooling (Maxpooling_2)

- In this pooling layer the is deceased to a size of [11, 16, 16, 35], which are 35 layers with 11 frames of a size of [16 × 16] which are half the size of each frame of the previous layer.

Stage 8: Flatten

- In this stage a grouping of the characteristics of the last layer is made as a single column vector, this is needed in *keras* to be able to make connections with the dense layer.

Stage 9: Dense layers: These are the classifying layers:

- Dense layer (30): 30 neurons, with *ReLu* activation function.
- Dense layer (27): 27 neurons with *ReLu* activation function.
- Dense layer (20): 20 neurons with *ReLu* activation function.
- Dense layer (7): 7 neurons with *ReLu* activation function.
- Dense layer (2): there are 2 neurons, in this layer it is classified whether an input is "fall" or "not fall", the *Softmax* activation function was used, which give a probabilistic output [29]. The training of the model was carried out within 10 epochs. The training was carried out using the *google Colab* platform and using a Tensor Processing Unit.

4.4 Evaluation Metrics

The Binary crossentropy metric was used to assess the model during training (loss function), this function gives the value of loss at each epoch, this to know the probability of the predictions of the model, that is to say how good or bad is the model. The predictions when having high loss values will mean that the prediction is bad otherwise if the values are low it will mean that the predictions are good [23, 30].

The Accuracy metric was also used as evaluation of the performance of a model to know the accuracy of a model [23].

Additionally, the training was carried out using the optimization method of adaptive moments, "Adam", which is a stochastic optimizer, that is, it optimizes randomly the parameters [23, 31].

To assess the model with the test data, the confusion matrix and the area under the curve (AUC) of the receiver operation characteristic (ROC) curve were used.

5 Results

This chapter will discuss the results obtained from the neural network model for the detection of falls. The first section presents the results of the training, accuracy

obtained in the last peri-od of training, and the second section describes the results of the evaluation with the test set using the ROC curve and the confusion matrix [33, 34].

5.1 Training Results

- It was obtained as a result of 10 training periods with an accuracy of 0.971 in the last period of training.
- Training time was approximately 3 min per epoch.
- The training set was 178 videos among which there were 85 fall and 93 non-fall videos.
- Figure 8 presents the confusion matrix which shows the performance of the model using the 178 videos of the training set.

As a result, it was obtained that of the 178 videos evaluated:

True Positive (VP):

83 VP

False Negative (FN):

2 FN

False positive (FP):

3 FP

True Negative (VN):

90 VN

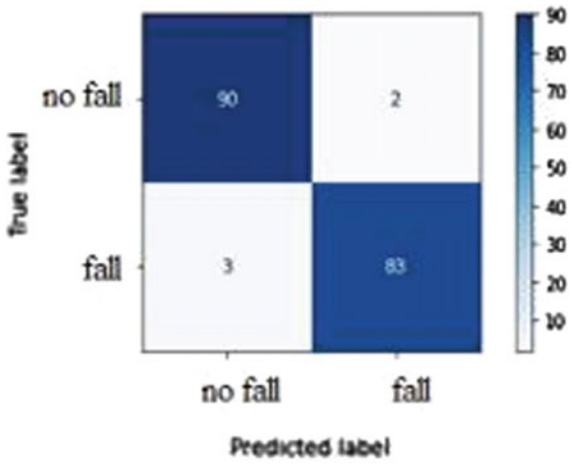

Fig. 8 confusion matrix of the training stage

From the confusion matrix it is possible to calculate the following metrics:

$$\text{Sensitivity} = \text{VP}/(\text{Total Positives}) = 83/86 = 0.965$$
$$\text{Specificity} = \text{VN}/(\text{Total Negatives}) = 90/92 = 0.978$$
$$\text{Accuracy} = (\text{VN} + \text{VP})/\text{Total} = (90 + 83)/178 = 0.971$$

As training results, an accuracy of 0.971, a sensitivity of 0.965 and a specificity of 0.978 were obtained, as well as successfully achieving 173 videos of 178 and only err in 5, these results are indicative of how functional it can be the model created.

It is mentioned in [4] that the larger the training set, the greater the possibility of a neural network to learn, when a large amount of training data is not used, it may be that the model is over-adjusted or over-trained, that is, it will learn characteristics which are not the ones that the model wants to learn and when new scenarios are presented it does not achieve the detection objective.

In this model, an activation function called ReLu was used for training, which as mentioned earlier, this function helped to reduce training times [29], since preliminary tests were carried out with different models in this project without using ReLu activation function and training times ranged from 8 to 12 min.

The ReLu activation function helped the model to train since without this activation function the model stagnated at an accuracy of 0.55.

The results obtained from the confusion matrix of the training set are not the validation results of the model, this test was performed to know how the model behaved with the data used in the survey.

Of the 178 videos used in the training set, the model was successful in 175 videos and only failed in 5, this could be because the detection of people or actions implies a great challenge due to the variability in the funds of the scenarios, In addition to the lighting changes or if there are partial or total occlusions within the video images, all these may be factors that may affect the detection, causing the detection results to be incorrect [4].

5.2 Test Set Evaluation

Two methods were used to evaluate the proposed model with the test set. The first method used is the ROC curves and as a second method a confusion matrix was used [32]. To perform these tests a set of 40 videos was separated from the dataset.

ROC curve

This validation method was used as it tests the model by means of the true positive rate and false negative rate. As a training set, 40 videos were separated from the database, these videos were used only for the evaluation of the model's performance. Figure 9 shows the evaluation of the performance of the model evaluated with the ROC curve, the highest point of sensitivity is between 0.3 and 1, so it is an indicator

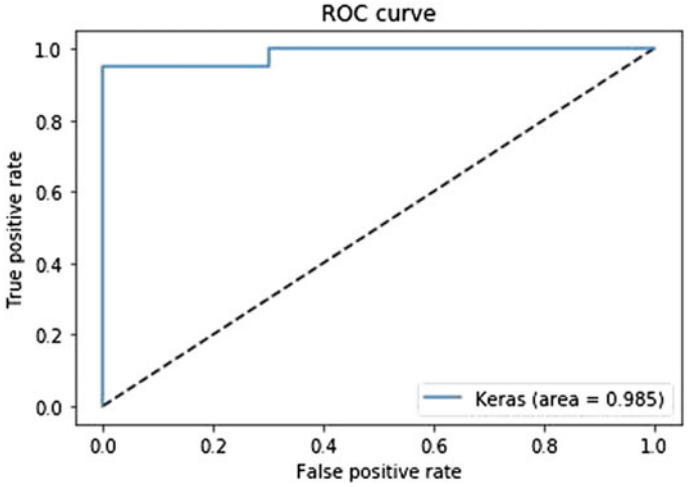

Fig. 9 ROC curve of the proposed model

that the model has good performance for the Fall detection also shows that this result shows that the model was correct in most of the videos in the test set.

Confusion Matrix

The confusion matrix validation method was performed in order to complement the validation of the model. The performance of the model is shown in Fig. 10.

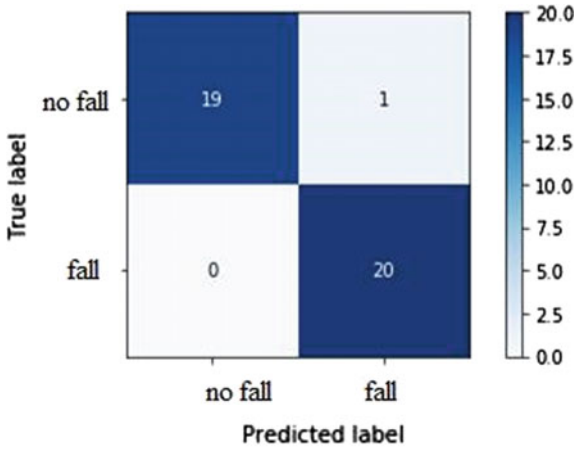

Fig. 10 Confusion matrix over the test set

As a result, it was obtained that of the 40 videos evaluated, and we can see that only one element was confused:

True Positive (VP):

20

VPFalse Negative (FN):

1 FN

False positive (FP):

0 FP

True Negative (VN):

19 VN

The sensitivity, specificity and accuracy of the model were obtained from the results of the confusion matrix.

$$\text{Sensitivity} = \text{VP}/(\text{Total Positives}) = 20/20 = 1$$
$$\text{Specificity} = \text{VN}/(\text{Total Negatives}) = 19/20 = 0.95$$
$$\text{Accuracy} = (\text{VN} + \text{VP})/\text{Total} = (20 + 19)/40 = 0.975$$

It was validated with 20% of the videos used in the training set since it gives us greater confidence in the operation of the model performed. Using a matrix of confusion allows us to see how many videos he evaluated as falling or not falling and in which ones, and in how many he failed, so in this way he gives us quantitative results. From the validation results it can be seen that from 40 videos, the model failed in 1, and correctly classify the others, thus the model has an Accuracy of 0.975, a specificity of 0.95, and a sensitivity of 1.

These results are promising since they indicate that the model works as expected as it manages to discriminate a fall from a non-fall.

6 Conclusions

In this chapter an architecture of a convolutional model for the detection of falls of older adults in smart cities by means of video images is presented. The Architecture was composed of 3D convolutional layers to take advantage of the volumetric structure of the video scenes.

The model function correctly in addition to achieving each of the objectives set at the beginning of the project, the model was trained with video images obtained from a database and an accuracy of 0.97 was obtained which indicates that the model is able to detect a fall, and at the time of validating the model, it was only wrong once,

this is an indicator that the model implemented is highly competitive in relation to [2].

As a future work, the proposed architecture is planned that not only manages to detect the falls, but that it manages to track the test subjects through gait recognition [35–37] as well as distinguish faces, this would be a great contribution to the care of older adults. Also, it is planned to increase the number of videos in the test, to better assess the generalization of the proposed network.

References

1. Chen, Y., Yang, X., Zhong, B., Pan, S., Chen, D., Zhang, H.: CNNTracker: Online discriminative object tracking via deep convolutional neural network. Appl. Soft Comput. J. **38**, 1088–1098 (2016)
2. Chu, C., Chang, C., Chang, T., Liao, J.: Elman neural network identify elders fall signal base on second-order train method. In: 6th International Symposium on Next Generation Electronics (2017)
3. Schuh, A., Campos, M.D.B., Bez, M.: Universal access in human-computer interaction. Interact. Tech. Environ. **9738**(1), 92–101 (2016)
4. Nam, H.: Learning multi-domain convolutional neural networks for visual tracking hyeonseob nam. Comput. Vis. Pattern Recogn. 4293–4302 (2016)
5. Bian, Z., et al.: Fall detection based on body part tracking using a depth camera. IEEE J. Biomed. Heal. Informatics **19**(2), 430–439 (2015)
6. Wang, H., Zhang, D., Wang, Y., Ma, J., Wang, Y., Li, S.: RT-fall: a real-time and contactless fall detection system with commodity wifi devices. IEEE Trans. Mob. Comput. **16**(2), 511–526 (2017)
7. Barillaro, S., et al.: Diseño de sistema IoT de monitoreo y alarma para personas mayo-res Resumen Contexto Introducción Líneas de Investigación, Desarrollo e Innovación, pp. 712–716 (1980)
8. Mohanty, S.P., Choppali, U., Kougianos, E.: Everything you wanted to know about smart cities: The internet of things is the backbone. IEEE Consumer Electron. Mag. **5**(3), 60–70
9. Dipanjan Sarkar, T.G., Bali, R.: Hands-on transfer learning with Python, Primera ed. Pack Publishing Ltd, Birmingham (2018)
10. Martin del Brio, B., Molina, A.S.: Redes Neuronales y Sistemas Difusos, 2a edicion. Alfaomega, Bogota (2001)
11. López, R.F., Fernández, J.M.F.: Las Redes Neuronales Artificiales fundamientos teoricos y aplicaciones prácticas, 1ra edicio. Gesbiblo, S.L, Oleiros (2008)
12. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning, 1ra edicio. MIT, London (2016)
13. Long, J., Shelhamer, E., Darrell, T.: Fully Convolutional Networks for Semantic Segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2015)
14. Babaee, M., Dinh, D.T., Rigoll, G.: A deep convolutional neural network for video sequence background subtraction. Pattern Recogn. **76**, 635–649 (2018)
15. Matich, D.J.: Redes Neuronales: Conceptos Básicos y Aplicaciones. Universidad Tecnológica Nacional, México (2001)
16. Salas, R.: Redes Neuronales Artificiales. Departamento de Computación (2004)
17. Grado, T.D.E., Ingenier, E.N.: Entrenamiento de redes neuronales basado en algoritmos evolutivos (2005)
18. Lanzarini, L., Hasperué, W., Estrebou, C., Ronchetti, F., Monte, A.V.: Aprendizaje Automático aplicado a Reconocimiento de Patrones en Video y Minería de Datos. In: XVIII Workshop de Investigadores en Ciencias de la Computación (2016)

19. Feichtenhofer, C., Pinz, A., Zisserman, A.: Convolutional two-stream network fusion for video action recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2016)
20. Charfi, I., Miteran, J., Dubois, J., Atri, M., Tourki R.: Optimized spatio-temporal descriptors for real-time fall detection: comparison of support vector machine and Adaboost-based classification. J. Electron. Imaging (2013)
21. Montoro, A.F.: Phyton 3 al descubierto, 1ra edicio. RC libros (2012)
22. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Kudlur, M.: Tensor-flow: A system for large-scale machine learning. In: 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16), pp. 265–283 (2016)
23. Gulli, A., Pal, S.: Deep learning with Keras. Packt Publishing Ltd (2017)
24. Silva, B.N., Khan, M., Han, K.: Towards sustainable smart cities: A review of trends, architectures, components, and open challenges in smart cities. Sustain. Cities Soc. **38**, 697–713 (2018)
25. Lemlouma, T., Laborie, S., Roose, P.: Toward a context-aware and automatic evaluation of elderly dependency in smart homes and cities. In: IEEE 14th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM), pp. 1–6. IEEE (2013)
26. Mirri, S., Prandi, C., Salomoni, P., Callegati, F., Campi, A.: On combining crowdsourcing, sensing and open data for an accessible smart city. In: Eighth International Conference on Next Generation Mobile Apps, Services and Technologies, pp. 294–299. IEEE (2014)
27. Xi, H., Ren, X., Zhai, S.G.: Smart pension: the elderly care service innovation with information technology. Sci. Res. Aging **2**(7), 12–20 (2014)
28. Et image, I.: U. C. 6306 Lei2 Laboratoire Electronique, Fall Detection Dataset. Fall Detection Dataset (2013)
29. Samarasinghe, S.: Neural Networks for Applied Sciences and Engineering: From Fundamentals to Complex Pattern Recognition. Auerbach Publications (2016)
30. Godoy, D.: NoComprensión de la entropía cruzada binaria/pérdida de registro: una explicación visual Title (2018). [Online]. Available: https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a. Accessed 28 Oct 2019
31. Kingma, D.P., Jimmy, B.A.: Adam: a method for stochastic optimization. arXiv preprint arXiv: 1412.6980 (2014)
32. Lorca, J.C.: Uso de curvas ROC en investigación clínica. Aspectos teórico-prácticos (2011)
33. Hay, A.M.: The derivation of global estimates from a confusion matrix. Int. J. Remote Sens. **9**(8), 1395–1398 (1988)
34. Zelada, C.: Evaluación de modelos de clasificación (2017)
35. Figueiredo, J., Santos, C.P., Moreno, J.C.: Automatic recognition of gait patterns in human motor disorders using machine learning: a review. Med. Eng. Phys. **53**, 1–12 (2018)
36. Godfrey, A.: Wearables for independent living in older adults: gait and falls. Maturitas **100**, 16–26 (2017)
37. Harris, P., Garcia, M.B.: To fall is human: falls, gait, and balance in older adults. In: New Directions in Geriatric Medicine, pp. 71–90. Springer, Cham (2016)