Postprint: A Study on Human Fall Behavior Using a CNN-LSTM Hybrid Model

Authors: She Xiangyang, Su Xuewei

Date: 2018-10-11T00:00:00+00:00

Abstract

Human fall behavior recognition in video surveillance is of great significance for improving the quality of elderly care and reducing the social burden of elderly support. Traditional pattern recognition methods rely on manually selected features, have low intelligence levels, and exhibit insufficient recognition accuracy. Deep learning models possess strong generalization capabilities and feature extraction is performed automatically. However, current deep learning models cannot effectively combine the spatial and temporal features of fall behaviors in surveillance videos. To this end, we propose a human fall behavior recognition method based on a hybrid CNN (convolutional neural network) and LSTM (long-short term memory) model. This model adopts a two-layer structure, where videos are input into the network in groups of 5 frames each; CNN extracts spatial features from the video sequences, LSTM extracts features along the temporal dimension of the videos, and finally a softmax classifier is used for recognition. Experiments demonstrate that this method can effectively improve the accuracy of fall recognition.

Full Text

Preamble

Research on Human Fall Behavior Using CNN and LSTM-Based Hybrid Model

She Xiangyang, Su Xuewei
(College of Computer Science & Technology, Xi' an University of Science and Technology, Xi' an 710054, China)

Abstract: The detection of human fall behavior in video surveillance is of great significance for improving the quality of care for the elderly and reducing the burden of social pension systems. Traditional pattern recognition methods rely

on manually selected features, suffer from low intelligence, and achieve insufficient recognition accuracy. Deep learning models offer strong generalization capabilities and automatic feature extraction. However, existing deep learning models cannot effectively combine spatial and temporal features of fall behavior in surveillance videos. To address this limitation, we propose a hybrid model that combines CNN (convolutional neural network) and LSTM (long short-term memory) for human fall behavior recognition. This two-layer model processes video input in groups of 5 frames: the CNN extracts spatial features from video sequences, the LSTM extracts temporal features along the time dimension, and a softmax classifier performs final recognition. Experimental results demonstrate that this approach effectively improves fall detection accuracy.

Keywords: falling behavior recognition; convolutional neural network; long short-term memory; time dimension

0 Introduction

With the aging of society, injuries and fatalities caused by falls among elderly individuals occur frequently. Accurate and efficient recognition of fall behavior in surveillance videos holds important practical significance for the safety protection of older adults [1]. Numerous researchers have conducted extensive studies on fall behavior recognition in videos, proposing various approaches that fall into two main categories: shallow traditional pattern recognition methods and deep learning-based methods.

In the shallow traditional pattern recognition domain, literature [2] manually extracted multiple features including the aspect ratio of human contour bounding boxes, Hu moment features, eccentricity of human contours, and human body axis angles, fusing these features for fall detection using SVM. Literature [3] extracted time-domain and frequency-domain features sensitive to fall behavior and employed singular value decomposition for dimensionality reduction before reconstructing fall features, using an SVM classifier for detection. The recognition rates of these methods depend on manually extracted features; once the extracted features are suboptimal, fall detection performance suffers significantly.

In deep learning models, multi-layer modeling of data obtains feature representations from video data, avoiding tedious manual feature extraction while demonstrating strong generalization capabilities. Literature [4] proposed a deep learning-based human activity recognition method using CNN, where convolutional neural networks analyze local features to obtain output features for classification. However, this method only captures local spatial features while losing temporal characteristics. Literature [5] presented an LSTM-based deep learning approach for human activity recognition, modeling temporal sequences to train and recognize human behaviors. The limitation lies in its extraction of only temporal features while losing local spatial information.

To better obtain both spatial and temporal features from video data, some researchers have combined CNN and LSTM, successfully applying them to video classification and description tasks. Ng et al. [6] processed image data and optical flow data through separate CNNs to obtain spatial information from video frame sequences, then fed the CNN outputs into LSTM to mine temporal relationships, finally predicting video categories via softmax. Venugopalan et al. [7] sampled short videos into 16-frame image sequences to represent entire videos, extracted features using CNN, applied mean pooling to these 16-frame features to obtain video encoding features, and then used LSTM decoding to generate video descriptions.

Building upon previous research, we propose a hybrid CNN-LSTM model for fall behavior recognition. After simple preprocessing of video datasets, the hybrid model employs CNN' s sliding window and weight sharing [8] to obtain local spatial features from video sequences as input to the next layer, while utilizing LSTM' s temporal characteristics to capture time features from video data. This combination leverages the respective advantages of both architectures. Additionally, since deep learning automatically extracts behavioral features, the cumbersome manual feature extraction process is avoided, and fall behavior recognition accuracy improves significantly.

1.1 Convolutional Neural Networks

CNN is a deep learning network first proposed by Fukushima [9] in 1980, typically consisting of an input layer, convolutional layers, pooling layers, fully connected layers, and an output layer. The basic structure of convolutional neural networks is shown in [Figure 1: see original paper].

a) Input Layer. The input layer marks the beginning of the entire network. In image processing, the input to a convolutional neural network is typically a pixel matrix of an image X.

b) Convolutional Layer. The convolutional layer is the most important component of CNN. Based on the concept of local receptive fields from biological visual cells, each node in the convolutional layer receives input from only a small region of the previous neural network layer. The convolutional layer performs deeper analysis on each small region to obtain more abstract features. There are three convolution types: full, same, and valid. Let H_i represent the feature map of layer i (assuming layer i is a convolutional layer). The specific generation process of H_i is [10]:

$$H_i = f(H_{i-1} \otimes W_i + b_i)$$

where W_i denotes the weight vector of the convolutional kernel at layer i; represents the convolution operation between the kernel and the image or feature map at layer i-1; f(•) is the activation function, taking the algebraic sum of the convolution output and the bias b_i of layer i as its argument to produce feature

map H_i of layer i. Common activation functions include relu, sigmoid, and tanh.

Figure 2 illustrates the convolutional layer calculation process using same convolution as an example, where the red box contains the original matrix. The convolution operation computes the sum of element-wise products. The calculation for the gray portion is as follows:

$$(1\times0)+(0\times0)+(-1\times0)+(1\times0)+(0\times1)+(-1\times0)+(1\times0)+(0\times0)+(-1\times1)+1=-1<0$$

This example uses relu as the activation function, with the relu formula:

$$g(x)=\max(0,x)$$

According to formula (2), the final value is 0.

c) Pooling Layer. A pooling layer is often added between convolutional layers. Pooling layers effectively reduce matrix dimensions, thereby decreasing the number of nodes in the final fully connected layer and reducing overall network parameters. Common pooling methods include max-pooling and average-pooling. Figure 3 shows the pooling results R for the convolution results from Figure 2.

d) Fully Connected Layer. Convolutional neural networks typically include 1-2 fully connected layers at the end to produce final classification results. After several convolutional and pooling layers, image information has been abstracted into higher-level features. We can view convolutional and pooling layers as automatic feature extraction processes; after feature extraction, fully connected layers are still needed to complete classification tasks.

e) Output Layer. The commonly used output layer is the softmax layer, primarily for classification problems. Given an input x, softmax provides the probability distribution across different categories. For a raw measurement h_j that input x belongs to class j, the softmax formula is:

$$P(y=j|x)=\frac{e^{h_j}}{\sum_k e^{h_k}}$$

where h_j represents the probability that given input x belongs to class j.

1.2 Long Short-Term Memory Networks

Long Short-Term Memory (LSTM) is a special type of recurrent neural network (RNN) designed to overcome RNN' s inability to handle long-distance dependencies. In RNN, hidden layer nodes within the same layer have certain connections —when sequence images are input sequentially into the network, hidden layer

node calculations depend not only on current input layer inputs but also on activation values from hidden layer nodes at the previous time step. For input sequence x = (x_1, x_2, ···, x_t), the RNN network layer produces hidden layer sequence h = (h_1, h_2, ···, h_t) and output sequence y = (y_1, y_2, ···, y_t) as follows [6, 11, 12]:

$$h_t = f(W_{xh}x_t + W_{hh}h_{t-1} + b_h)$$
$$y_t = W_{ho}h_t + b_o$$

where f(•) represents the activation function used in the hidden layer; W_{xh} denotes the weight matrix from input layer to hidden layer; W_{hh} represents the weight matrix from hidden layer to hidden layer; W_{ho} is the weight matrix from hidden layer to output layer; b_h and b_o are bias vectors for the hidden layer and output layer, respectively.

The reason RNN cannot discover relationships between frames with long time intervals is that RNN lacks memory units to store and output information. Unlike standard RNN, the LSTM architecture [13] uses memory cells to store and output information, thereby gaining memory capability for inputs from longer time periods.

LSTM includes new input x_t, output h_t, input gate i_t, forget gate f_t, and output gate o_t. The input gate i_t determines which parts enter the current state c_t for updating based on x_t. The forget gate f_t decides which information to discard. Through the forget gate and input gate, the LSTM structure can more effectively determine which information should be forgotten and which should be retained. The specific structure is shown in [Figure 4: see original paper].

In Figure 4, the symbol represents element-wise vector multiplication; represents vector concatenation; + represents vector sum. The LSTM components are updated as follows [14, 15]:

$$i_t = \sigma(W_{xi}x_t + U_{hi}h_{t-1} + b_i)$$
$$f_t = \sigma(W_{xf}x_t + U_{hf}h_{t-1} + b_f)$$
$$c_t = \tanh(W_{xc}x_t + U_{hc}h_{t-1} + b_c)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ c_t$$
$$o_t = \sigma(W_{xo}x_t + U_{ho}h_{t-1} + b_o)$$
$$h_t = o_t \circ \tanh(c_t)$$

where represents the sigmoid activation function; denotes element-wise vector multiplication; W_{xi}, W_{xf}, W_{xc}, W_{xo} represent weight matrices from input layer to input gate, forget gate, memory cell, and output gate, respectively; U_{hi}, U_{hf}, U_{hc}, U_{ho} represent weight matrices from

hidden layer to input gate, forget gate, memory cell, and output gate, respectively; b_i, b_f, b_c, b_o are bias values for input gate, forget gate, memory cell, and output gate, respectively.

2 Fall Behavior Recognition Using Hybrid Deep Neural Network Model

2.2.1 Convolutional Neural Network Processing Layer

The model uses convolutional neural networks to extract spatial information from video frames, generating behavioral representation features. Let N represent the number of frames in the input image sequence. For a single image with pixel matrix size P×Q, using same convolution with kernel size k×k, zero-padding of length k/2 is added to the original image matrix to create a pixel matrix of size (P+2 k/2)×(Q+2 k/2), where · denotes the floor function. Let v_{ij} represent the pixel value at position (i,j) in the expanded pixel matrix. Then all pixel values falling within the k-th sliding window can be represented as a window matrix:

$$X_{ij} = \begin{pmatrix} v_{i-\lfloor k/2 \rfloor, j-\lfloor k/2 \rfloor} & \cdots & v_{i-\lfloor k/2 \rfloor, j+\lfloor k/2 \rfloor} \\ \vdots & \ddots & \vdots \\ v_{i+\lfloor k/2 \rfloor, j-\lfloor k/2 \rfloor} & \cdots & v_{i+\lfloor k/2 \rfloor, j+\lfloor k/2 \rfloor} \end{pmatrix}$$

For each window matrix, convolution operations are performed using formula (1) to obtain current window features:

$$Y_{ij} = f(x \otimes W + b)$$

Given relu' s fast convergence properties, the activation function f adopts the following relu function:

$$g(x) = \max(0, x)$$

After completing convolution, pooling operations are performed. Max pooling is selected for processing to obtain the maximum feature value from each window matrix:

$$R_{ij}^{n} = \mathrm{Max}(Y_{ij})$$

where R_{ij}^n represents the feature matrix after convolution and pooling operations for the n-th image in the sequence.

Applying the above operations to sequence images, the feature matrices for each frame in the sequence can be represented as R = (R_1, R_2, ···, R_n), where n N.

2.2.2 Long Short-Term Memory Model Processing Layer

One output R from the CNN layer corresponds to one LSTM input at time t. At time t, based on formulas (6)-(11), the LSTM unit components are updated as follows:

$$
\begin{aligned}
i_t &= \sigma(W_{ri}R_t + U_{hi}h_{t-1} + b_i) \\
f_t &= \sigma(W_{rf}R_t + U_{hf}h_{t-1} + b_f) \\
c_t &= \tanh(W_{rc}R_t + U_{hc}h_{t-1} + b_c) \\
c_t &= f_t \circ c_{t-1} + i_t \circ c_t \\
o_t &= \sigma(W_{ro}R_t + U_{ho}h_{t-1} + b_o) \\
h_t &= o_t \circ \tanh(c_t)
\end{aligned}
$$

where represents the sigmoid activation function; R_t denotes the feature matrix input at time t; W_{ri}, W_{rf}, W_{rc}, W_{ro} represent weight matrices from input layer to input gate, forget gate, memory cell, and output gate, respectively; U_{hi}, U_{hf}, U_{hc}, U_{ho} represent weight matrices from hidden layer to input gate, forget gate, memory cell, and output gate, respectively; b_i, b_f, b_c, b_o are bias values for input gate, forget gate, memory cell, and output gate, respectively.

The video data, after preprocessing, is randomly divided into training and test datasets. The training data is used for model construction and parameter adjustment, while the test data evaluates model performance. The CNN network extracts spatial features from each frame, then the CNN network outputs are resized and sequentially input into the LSTM network to obtain sequential temporal features. The average of LSTM outputs across all time steps is computed to predict final classification results. The basic structure of the hybrid model is shown in [Figure 5: see original paper].

3 Experiments

3.1 Dataset and Test Environment

This paper uses the CASIA dataset [16] as test data, provided by the Institute of Automation, Chinese Academy of Sciences. All videos were simultaneously captured by three uncalibrated stationary cameras positioned at horizontal, oblique, and top-down angles, with a frame rate of 25fps, huffyuv compression encoding, and resolution of 320×240. [Figure 6: see original paper] shows selected original video frames.

Experiments were conducted using the Python-based deep learning library Keras in a GPU-accelerated environment. Specific experimental environment configurations are shown in .

Table 1: Experimental Environment Configuration

Component	Specification
GPU	NVIDIA Titan XP
RAM/Storage	64GB/4.2TB
OS	Ubuntu 14.04
Python	Python 3.6
Framework	Keras

3.2 Experimental Protocol and Parameter Settings

We selected bending-walking, squatting, fainting, jumping, running, and walking behaviors from the top-down view of single-person actions in the CASIA dataset as experimental data. Falls constitute the abnormal dataset, while bending-walking, squatting, jumping, running, and walking form the normal dataset. After preprocessing the video data, we obtained 955 fall sequence images and 840 non-fall sequence images, randomly selecting 80% as the training dataset and 20% as the test dataset.

Since the original data consists of color image sequences with unstable color channel characteristics, we preprocess the raw images by converting them to single-channel images and scaling pixel values to the [0,1] interval through simple normalization for use as final experimental data.

To determine the optimal number of input sequence frames and validate the effectiveness of the hybrid model, we conducted two sets of comparative experiments under identical environmental conditions, using the same dataset and data volume:

a) Input sequences with frame numbers of 3, 4, 5, 6, and 7 into the hybrid model for comparison.

b) Comparative experiments for fall detection using SVM, CNN [4], LSTM [5], and the hybrid model proposed in this paper.

Deep learning models involve parameters including sliding window size, number of convolution kernels, activation functions, LSTM node count, optimization methods, and learning rates. We selected optimization methods (Adam, SGD, RMSprop), learning rates (0.0001, 0.001, 0.01, 0.1), LSTM node counts (32, 64, 128), and activation functions (relu, tanh) for experimental optimization, with remaining parameters set to defaults. The loss function uses categorical cross-entropy. Through experimental comparison, optimal parameter settings are shown in .

Table 2: Parameter Settings

Parameter	Value
Loss Function	categorical_crossentropy
LSTM Nodes	128
Activation Function	relu
Learning Rate	0.001
Optimizer	Adam

3.3 Experimental Results and Analysis

Comparative experiments with different sequence frame numbers yielded results shown in [Figure 7: see original paper]. The figure indicates that experimental performance reaches optimum when the sequence number is 5. This may be because too few sequence frames lose partial fall behavior information and cannot adequately represent fall behavior, while too many sequence frames reduce the overall training sample size and cannot properly train the model. Therefore, experiments use sequences of 5 frames each, with accuracy used to evaluate the model. Accuracy results are shown in .

Table 3: Recognition Accuracy of Various Models

Model	Accuracy
SVM	82.17%
CNN [4]	83.84%
LSTM [5]	91.67%
CNN+LSTM	94.44%

As shown in Table 3, the accuracy for fall detection on the CASIA database, ranked from highest to lowest, is: CNN+LSTM, LSTM, CNN, SVM. The CNN+LSTM hybrid model achieves higher accuracy than the other three models, benefiting from its effective utilization of CNN' s ability to extract local spatial features through convolution while combining LSTM' s temporal characteristics to capture time features from video sequences. Additionally, deep learning methods avoid cumbersome manual feature extraction and demonstrate stronger generalization capabilities. However, as shown in [Figure 8: see original paper], in terms of training time, the CNN+LSTM hybrid model consumes more time than CNN and LSTM individually due to its complex network structure, while LSTM consumes more time than CNN because of its memory functionality and more complex network architecture.

Figure 8: Training Time Comparison of Various Models

4 Conclusion

This paper proposes a CNN-LSTM hybrid model for fall behavior detection, achieving 94.44% recognition accuracy on the CASIA dataset. Compared with

shallow traditional pattern recognition methods, this approach avoids manual feature extraction and enhances model generalization capability. For deep CNN and LSTM networks, the model can extract not only spatial features from sequential video frames but also temporal information between frames, demonstrating significantly improved recognition rates and certain reliability. Since the experimental dataset uses fixed, single-background scenes with only single-person behaviors, there remains a gap with real-world scenarios. Future work should conduct in-depth research and analysis on fall behaviors in more realistic settings.

References

[1] Mubashir M, Shao L, Seed L. A survey on fall detection: principles and approaches [J]. Neurocomputing, 2013, 100(2): 144-152.

[2] Wang Dafeng, Liu Yongkui, Liu Shuang, et al. Abnormal behavior recognition of fall in surveillance video [J]. Electronic Design Engineering, 2016, 24(22): 1126-1222.

[3] Bai Yong, Sun Xiaowen, Qin Fang, et al. The falling detection algorithm based on SVD feature dimension reduction and SVM [J]. Computer Applications and Software, 2017, 34(1): 247-251.

[4] Wang Zhongmin, Cao Hongjiang, Fan Lin, et al. Method on human activity recognition based on convolutional neural networks [J]. Computer Science, 2016, 43(s2): 56-58.

[5] Kuang Xiaohua, He Jun, Hu Shaohua, et al. Comparison of deep feature learning methods for human activity recognition [J]. Application Research of Computers, 2018, 35(9): 2815-2817, 2822.

[6] Ng Y H, Hausknecht M, Vijayanasimhan S, et al. Beyond short snippets: deep networks for video classification [C]// Computer Vision and Pattern Recognition. 2015: 4694-4702.

[7] Venugopalan S, Xu H, Donahue J, et al. Translating videos to natural language using deep recurrent neural networks [J]. Computer Science, 2015.

[8] Ma Xuezhe, Hovy E. End-to-end sequence labeling via bi-directional lstm-cnns-crf [EB/OL]. (2016-05-29). https://arxiv.org/abs/1603.01354.

[9] Fukushima K. Neocognitron: a hierarchical neural network capable of visual pattern recognition [J]. Neural Networks, 1988, 1(2): 119-130.

[10] Li Yandong, Hao Zongbo, Lei Hang. Survey of convolutional neural network [J]. Journal of Computer Applications, 2016, 36(9): 2508-2515.

[11] Graves A. Supervised sequence labelling with recurrent neural networks [M]. Springer Berlin Heidelberg, 2012.

[12] Graves A, Mohamed A R, Hinton G. Speech recognition with deep recurrent neural networks [C]// Proc of IEEE International Conference on Acoustics, Speech and Signal Processing. 2013: 6645-6649.

[13] Gers F A, Schraudolph N N. Learning precise timing with lstm recurrent networks [M]. JMLR.org, 2003.

[14] Sundermeyer M, Ney H, Schlyuter R. From feedforward to recurrent LSTM neural networks for language modeling [J]. IEEE/ACM Trans on Audio, Speech & Language Processing, 2015, 23(3): 517-529.

[15] Cai M, Liu J. Maxout neurons for deep convolutional and LSTM neural networks in speech recognition [J]. Speech Communication, 2016, 77(C): 65-76.

[16] Behavioral analysis database of Chinese Academy of Sciences [EB/OL]. http://www.cbsr.ia.ac.cn/china/Action Databases CH.asp.

Note: Figure translations are in progress. See original paper for figures.