Neural Computing and Applications
https://doi.org/10.1007/s00521-021-06322-x

S. I : IWANN 2019

A novel keyframe extraction method for video classification using deep neural networks

Rukiye Savran Kızıltepe[1] · John Q. Gan[1] · Juan José Escobar[2]

Received: 18 January 2020 / Accepted: 8 July 2021

Abstract

Combining convolutional neural networks (CNNs) and recurrent neural networks (RNNs) produces a powerful architecture for video classification problems as spatial–temporal information can be processed simultaneously and effectively. Using transfer learning, this paper presents a comparative study to investigate how temporal information can be utilized to improve the performance of video classification when CNNs and RNNs are combined in various architectures. To enhance the performance of the identified architecture for effective combination of CNN and RNN, a novel action template-based keyframe extraction method is proposed by identifying the informative region of each frame and selecting keyframes based on the similarity between those regions. Extensive experiments on KTH and UCF-101 datasets with ConvLSTM-based video classifiers have been conducted. Experimental results are evaluated using one-way analysis of variance, which reveals the effectiveness of the proposed keyframe extraction method in the sense that it can significantly improve video classification accuracy.

Keywords Deep learning · Convolutional neural networks · Recurrent neural networks · Keyframe extraction · Video classification

1 Introduction

Video has become more popular in many applications in recent years due to increased storage capacity, more advanced network architectures, as well as easy access to digital cameras, especially in mobile phones. According to recent statistics, more than 500 h of video is uploaded onto the Internet every minute and sharp rise in the number of videos is expected to continue in the coming decades due to the increase in demand for video content [1]. Therefore, this increase is a remarkable issue and brings serious challenges for video indexing, archiving, and retrieval systems. The main subject of videos on social networking Web sites is human actions. Automatic classification of their semantic content is essential for appropriate use and management of these videos. However, the classification of video content remains a challenging task owing to the complexity of video data.

Action recognition problems have been addressed using deep learning approaches in both image and video domains. Convolutional neural networks (CNNs) have achieved state-of-the-art results in the recent decade. CNN applications to video-based tasks are not so successful as those in image domains, e.g., object detection [2], segmentation [3], pattern recognition [4], and classification [5, 6]. Therefore, the power of recurrent neural networks (RNNs) in sequence learning has been employed to gather temporal information for improving video classification performance. Although combining CNNs and RNNs has achieved good results [7, 8], the representation of temporal information is still a demanding problem due to complex variations in actions and dynamic background in videos.

✉ Rukiye Savran Kızıltepe
rs16419@essex.ac.uk

John Q. Gan
jqgan@essex.ac.uk

Juan José Escobar
jjescobar@ugr.es

[1] School of Computer Science and Electronic Engineering, University of Essex, Colchester CO4 3SQ, UK

[2] Department of Computer Architecture and Technology, CITIC, University of Granada, Granada, Spain

Published online: 02 August 2021

The performance of action recognition has been improved remarkably by transfer learning and use of extra training data. Extensive video datasets such as HMDB51 [9], UCF-101 [10], Sports-1M [11], and Youtube-8M [12] have been published, and the state-of-the-art results have recently been reported on these benchmarking datasets [13–17].

The majority of the current video classification methods classify videos by assigning a label to each frame. Nonetheless, considering all frames equally weakens the classification performance as some frames have more distinctive information than others. We argue that it is essential to select keyframes for better classification performance. Thus, this paper proposes a novel keyframe extraction method by identifying an action template to preserve the succinct content, in which the entire video is represented in a set of keyframes. The main novelty of this work is the proposed keyframe extraction algorithm that employs an action template for each video to extract and select the most distinctive frames with both static and dynamic backgrounds, without the need of using complex procedures.

The main contributions of this work are the identification of the best architecture for combining CNNs and RNNs for video classification and the proposal of the action template-based keyframe extraction, which aims to extract more informative frames by calculating the similarity only between action regions, rather than whole frames. The former was partly presented in [18], which has been substantially extended and serves as a baseline to test the newly proposed method. In this paper, extensive experiments have shown that the action template-based keyframe extraction method significantly outperformed the frame selection methods used in our experiments for comparison purposes.

The rest of the paper is organized as follows: The related work is reviewed in Sect. 2 and the proposed keyframe extraction method is described in Sect. 3. Experiments conducted are detailed in Sect. 4, while the results are analyzed and summarized in Sect. 5. Conclusions are given in Sect. 6.

2 Related work

Keyframe extraction approaches can be generally categorized into six groups: uniform sampling-based, shot boundary-based, shot activity-based, visual content-based, motion analysis-based, and clustering-based. Although uniform sampling-based methods are easy and computationally efficient, these methods may fail to represent the video in two possible scenarios: no enough keyframes for a short semantically important video and too many keyframes with similar content for a long static segment [19].

Early works on keyframe extraction focused on shot boundary-based techniques [20, 21]. Basically, this technique employs the first or middle frame of each shot as the keyframe after shot boundary detection [22]. Video shot boundary detection methods were reviewed by Dey et al. [22]. Although shot boundary-based methods are easy to use and generalize, the extracted keyframe cannot represent the visual content entirely and it is not stable.

Shot activity-based approach is used to select keyframes considering the frame with least difference from other frames in terms of a given similarity measure. Based on this concept, Lagendijk et al. proposed a keyframe selection method with the assumption that ‘every keyframe represents a contiguous interval in a shot’ [23]. In this work, the limits of intervals and the location of keyframe within each interval are optimized. Similarly, the Lloyd–Max algorithm is used in the design of a scalar quantizer in [24].

Visual content-based approach has been explored for visual content-based information retrieval and keyframe-based video summarization. In this approach, visual features of video clips are extracted to analyze keyframes in movie segments. Zhong and Smoliar proposed an integrated system solution using video content information obtained from a parsing process [25]. Human attention mechanism has been simulated to produce semantic video summary based on keyframe extraction. Visual attention of each frame is quantified using a descriptor named *attention quantifier*, which indicates color conspicuousness and the motion with more attention involved [26]. There have been many attempts to analyze visual content of video for keyframe extraction for video partitioning and summarization [27, 28].

As for motion analysis-based approach, a novel algorithm was proposed for selecting keyframes within shots from video by employing optical flow computations to detect local minima of action in a single shot [29]. This work measures the motion in a shot by utilizing optical flow analysis, where key frames are selected at the local minima of the action. Mizher et al. have also proposed an action keyframe extraction method based on L1-norm and accumulated optical flows [30]. Similar approach has been observed for salient region-based keyframe extraction by using optical flow and calculating mutual information entropy [31].

Clustering-based methods have been used to extract key frames. The idea is that frames are grouped based on their low-level features by using a clustering method like K-means and the most similar frames with the groups’ centers are selected [23]. Dynamic Delaunay graph clustering through iterative edge pruning technique has also been used to extract keyframes [32]. Tan et al. demonstrated KGAF-means method by adopting K-means and the artificial fish swarm algorithms to extract keyframe sequences [33].

The proposed method in this paper aims to tackle some important limitations of the aforementioned approaches. Although extracting keyframes using shot-based approaches is easy to use, early approaches are unable to capture the temporal information. As for clustering-based approaches, they are sensitive to the type of adopted kernel and the number of clusters, and high in time complexity [34]. Furthermore, video is a special kind of media content that includes temporal information and complex background. Another limitation of the mentioned methods is handling entire frame differences rather than a specific region of interest. This paper proposes a novel approach based on the similarity between regions of interest in consecutive frames to address these limitations. Different from the previous works, we employ an action template to find the region of interest for each video.

It is noteworthy that some related work on deep neural networks for video classification has been presented in our previous work [18].

3 The proposed method

Keyframe extraction is a principal pre-processing step in video analysis. The purpose of extracting keyframes is to get more discriminate information from the video in an effective manner. Each video has its own unique characteristics such as saturation, brightness, contrast, camera angle, vibration, blur, location of the action, number of actors, type of action, length, and background. Considering a large number of variables in each video and treating all videos equally bring about a major weakness in keyframe extraction. Thus, it is necessary to recognize the region of action in continuous action video. Considering the variations in the complex video data, it is a challenging task to find the location of action, based on which this paper proposes a new method for keyframe extraction.

3.1 The proposed keyframe extraction approach

In general, the location of action in a movie is related to the point on the screen and camera, to which reviewer's attention is paid. It is observed that attention is paid to the central area mostly while recording and watching. Therefore, the outside regions of video frames are usually cropped off before identifying the region of interest. Then, the area of action is formulated as a region in the center of video frames, which produces either the biggest difference or lowest similarity between consecutive frames, leading to a template for the video to track the action area. Calculating only the difference in regions of action between frames throughout the video helps to extract keyframes more accurately and effectively by reducing the influence of possible dynamically changing backgrounds..

The proposed keyframe extraction method consists of four steps: (1) identify an action template; (2) specify the location of an action; (3) calculate action similarities to find distinctive frames; and (4) select a preset number of keyframes in chronological order. The four steps of the proposed keyframe extraction method can be summarized as follows:

(1) Action template identification:

- Frame decomposition.
- Frame cropping.
- Define three possible regions for action template.
- Calculate mean squared error (MSE) for each possible region using the first two frames.
- Choose the region that produces the largest MSE as an action template.

(2) Action location specification:

- Find the region of interest on each frame by matching the action template against overlapped regions in each frame using the correlation coefficient defined in Eq. (3).

(3) Keyframe extraction:

- Calculate the structural similarity measure (S_i) between regions of interest on consecutive frames (f_i, f_{i-1}).
- Compare the similarity score with thresholds $T_1 = [0.65, 0.90]$ and $T_2 = [0.65, 0.95]$ (these threshold values were chosen by analysis of significance in action changes in our experiments):

  $$0.65 < S_i < 0.90 \rightarrow \text{add } f_i \text{ to primary list}(p_f)$$
  $$0.65 < S_i < 0.95 \rightarrow \text{add } f_i \text{ to alternative list}(a_f)$$

- Repeat the above till end of the video, with N_{p_f} frames extracted into p_f and N_{a_f} frames extracted in to a_f.

(4) Keyframe selection:

- Set the number of keyframes (N_{k_f}).
- Find keyframe ratio (k):

$$k = \begin{cases} \left\lfloor \frac{N_{p_f}}{N_{k_f}} \right\rfloor, & \text{if } N_{p_f} \geq N_{k_f} \\ \left\lfloor \frac{N_{a_f}}{N_{k_f}} \right\rfloor, & \text{otherwise} \end{cases}$$

- Return the indexes of keyframes by choosing a frame from every k frames from keyframe list p_f if $N_{p_f} \geq N_{k_f}$, or from keyframe list a_f otherwise.

In the first step, each frame is cropped by taking an appropriate number of pixels out (depending on frame resolution) from each side of a frame to create a general action area. As depicted in Fig. 1, the inner area is then divided into three different candidate templates. It has been observed that background changes between consecutive frames result in small structural similarity (SSIM) and action differences lead to large MSE. The candidate area having the largest MSE between consecutive frames is assigned as an action template. The mean squared error between two regions of frames X and Y is computed as follows:

$$\mathrm{MSE}(X,Y) = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}[Y(i,j) - X(i,j)]^2 \tag{1}$$

where m and n are the number of rows and columns in the region of interest, respectively. The structural similarity formulated by Wang et al. [35] is adopted in this paper and defined as follows:

$$\mathrm{SSIM}(X,Y) = \frac{(2\mu_X\mu_Y + C_1)(2\sigma_{XY} + C_2)}{(\mu_X^2 + \mu_Y^2 + C_1)(\sigma_X^2 + \sigma_Y^2 + C_2)} \tag{2}$$

where μ_X and μ_Y denote the average of pixel values in X and Y, respectively, σ_X and σ_Y are the variance of pixel values in X and Y, respectively, and σ_{XY} is the co-variance of pixel values in X and Y. $C_1 = (k_1L)^2$ and $C_2 = (k_2L)^2$ are small constants where L denotes the dynamic range of pixel values. MSE and SSIM are calculated for each frame region and used to select frames well representing the action (Fig. 2).

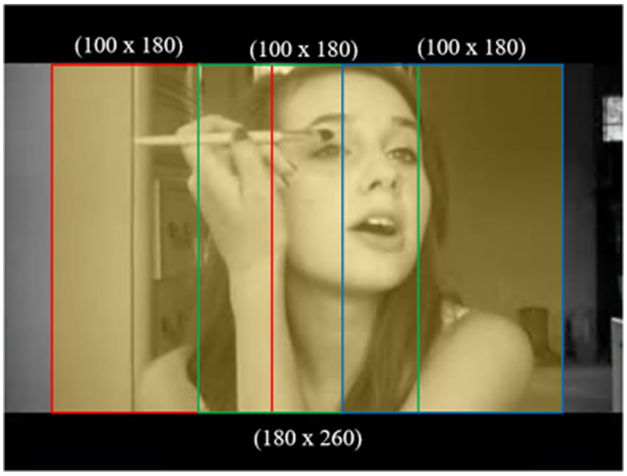

Fig. 1 Defining the possible locations of an action template. Red, green, and blue boxes represent the borders of three possible templates (Color figure online)

In the second step, after having determined the action template for each video, template-based correlation coefficient matching method is used to find at what position the template most closely matches the data in a region of each frame. This operation slides throughout each frame and compares the overlapped patterns of size $w \times h$ to the template, where w and h are width and height of the template, respectively. Then, the best matches are found as global maximums. Regarding color channels, template summation in the function is done over all channels and different mean values are used for each channel. The formula for the template-based matching method is:

$$R(x,y) = \sum_{x',y'}(T'(x',y') \cdot I'(x+x',y+y')) \tag{3}$$

where $R(x,y)$ is the correlation coefficient score for a single overlapped position of (x,y) representing the coordinates of each pixel in the frame. $T'(x',y')$ is the average of pixel values of the template T, where (x',y') represents the coordinates of each pixel in the template, given as:

$$T'(x',y') = T(x',y') - \frac{1}{(w \cdot h)} \cdot \sum_{x'',y''} T(x'',y''). \tag{4}$$

On the other hand, $I'(x+x',y+y')$ is the average of pixel values of a given frame I in the region overlapped with the template T, given as:

$$I'(x+x',y+y') = I(x+x',y+y') - \frac{1}{(w \cdot h)} \cdot \sum_{x'',y''} I(x+x'',y+y'') \tag{5}$$

where $x'' = 0,\ldots,w-1$ and $y'' = 0,\ldots,h-1$ which represent the new coordinates of (x,y) in the template after moving the center of the template over the frame. $T(x',y')$ is the pixel values for a pixel (x,y) in the template, while $I(x+x',y+y')$ is the pixel value for the corresponding pixel position in the frame. After performing the template matching procedure, the region of interest on each frame is localized where the highest matching probability takes place.

In step 3, it is very challenging to distinguish between action changes and background changes in consecutive frames. Through an extensive investigation, it has been discovered that the structural similarity between two regions of interest in two consecutive frames is more sensitive to action changes than background changes. After analyzing both background and action changes in these areas, two rules are proposed in this paper to specify upper and lower bounds of similarity range. Action changes are found mostly important in the interval [0.65, 0.90], and the lower similarity is mainly due to the dramatic change in dynamic background. Rarely, the difference between regions in consecutive frames is not in this range. However, if there are not enough keyframes extracted using the above interval, more frames are extracted by extending the upper bound of the interval up to 0.95.

Finally, keyframes are selected by using a keyframe ratio in chronological order. The pseudocode of the proposed algorithm is demonstrated in Algorithm 1.

3.2 Deep neural network architectures based on VGG-16 for video classification

In 2014, Simonyan and Zisserman [6] introduced a VGG-16 network architecture trained on 1000 image categories using image data for the ImageNet Large Scale Visual Recognition Competition (ILSVRC). VGG-16 consists of 16 convolutional layers with relatively small convolution filters (3x3). We used the pre-trained neural network VGG-16 to generalize the pre-learnt feature representations using transfer learning. In our previous work [18], ConvLSTM and LSTM with local features extracted using VGG-16 outperformed those using global features. Thus, this paper

Algorithm 1: Action template-based keyframe extraction

```
Function Keyframe_Extraction($V, N_{k_f}$)
    Input  : The input video stream, $V$
    Input  : The number of keyframes, $N_{k_f}$
    Output: The list of keyframes, $K$
    while input video is available do
        // Action template identification
        Get next video and crop outline areas;
        Frame decomposition $F = F_1, F_2, ..., F_N$;
        Select three candidate action regions $R1_{f1}, R2_{f1}, R3_{f1}$ on the first frame
          $F_1$ and $R1_{f2}, R2_{f2}, R3_{f2}$ the second frame $F_2$;
        Calculate MSE scores between candidate action regions:
          $MS_1 = MSE(R1_{f1}, R1_{f2}), MS_2 = MSE(R2_{f1}, R2_{f2}), MS_3 =$
          $MSE(R3_{f1}, R3_{f2})$;
        Choose action template considering the largest MSE score gained from two
          consecutive action regions;
        // Action location specification
        Extract region of interest on each frame by matching the action template
          against overlapped regions $r = r_1, r_2, ..., r_n$
        // Keyframe extraction
        foreach consecutive frame pairs $f_i$ and $f_{i+1}$ do
            Calculate the structural similarity measure $S_i$ between regions of
              interest ($r_i$ and $r_{i+1}$)
            if $0.65 < S_i < 0.90$ then // $T_1 = [0.65, 0.90]$
                Add $F_i$ to the primary keyframe list ($p_f$)
            end
            if $0.65 < S_i < 0.95$ then // $T_2 = [0.65, 0.95]$
                Add $F_i$ to the alternative keyframe list ($a_f$)
            end
            // Keyframe selection. Find keyframe ratio, $k$:
                $$k = \begin{cases} \left\lfloor \frac{N_{p_f}}{N_{k_f}} \right\rfloor, \text{ if } N_{p_f} \geq N_{k_f} \\ \left\lfloor \frac{N_{a_f}}{N_{k_f}} \right\rfloor, \quad \text{otherwise} \end{cases}$$
            if $N_{p_f} \geq N_{k_f}$ then
                Add every $k$ frames of the primary list $p_f$ to $K$
            else
                Add every $k$ frames of the alternative list $a_f$ to $K$
            end
        end
    end
    return the list of keyframes, $K$
End
```

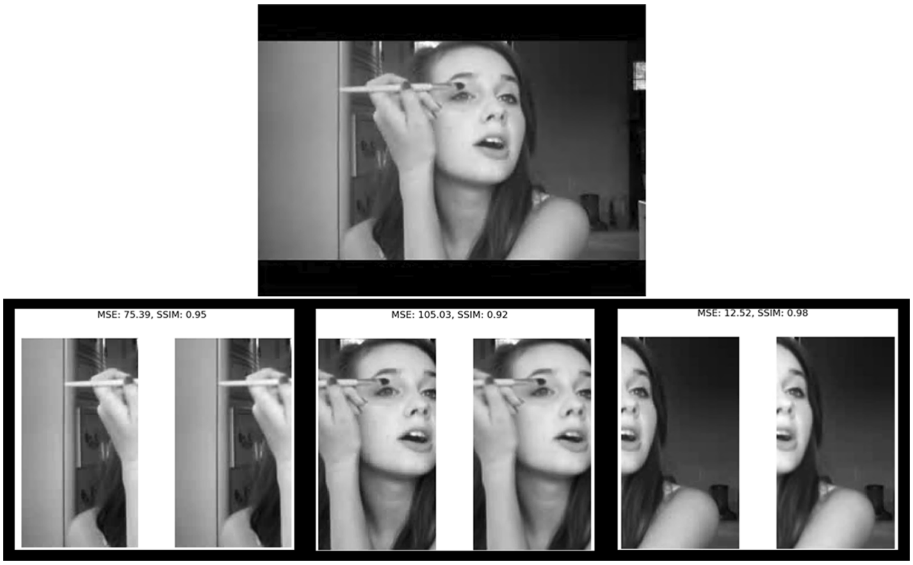

Fig. 2 An example of action template identification

uses the ConvLSTM(1) and LSTM(1) architectures used in [18]. Apart from the newly proposed keyframe extraction method, we also conducted the experiments using not only 20 but also 101 categories of the UCF-101 dataset and evaluating the proposed methods on the KTH action recognition dataset as well. Moreover, we optimized the parameters of the networks using hold-out validation method on the validation split of the training dataset. The two video classification architectures to classify 101 categories are shown in Fig. 3.

LSTM is one of the most common approaches for sequence modeling. Previous studies [36–38] have demonstrated that LSTM is a robust method to represent long-range dependencies. Its main advantage is that its memory cell c_t accumulates the state information. The cell is modified by controlling the input gate i_t and forget gate f_t at timestamp t. Once the cell is fed with a new input x_t, it accumulates input information, provided that the input gate is on. If the forget gate is activated, the previous cell information c_{t-1} could be forgotten. The output gate o_t checks the current cell output c_t to decide whether it is propagated to the final state h_t or not. In this study, we follow the hidden layer function of LSTM described in [39]:

$$\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i)\\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f)\\
c_t &= f_t \cdot c_{t-1} + i_t \cdot \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c)\\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co} \cdot c_t + b_o)\\
h_t &= o_t \cdot \tanh(c_t)
\end{aligned} \tag{6}$$

where '$\cdot$' denotes the Hadamard product, σ represents the sigmoid function, and tanh denotes the hyperbolic tangent function. In Eq. (6), W_{pq} and b_q are weight matrix and bias for the respective gates, where the subscript p can be either the input x, the cell output c , or the hidden state h and the subscript q can be either the input gate i, the forget gate f, the memory cell c , or the output gate o.

In VGG-16-LSTM, the local features extracted by VGG-16 from video frames are fed into LSTM to access spatiotemporal information. The number of units in the output space was set to 1024, and ReLU was used as the activation function.

An end-to-end trainable ConvLSTM was proposed by extending the fully connected LSTM to have convolutional structures in both the input-to-state and state-to-state transitions for precipitation nowcasting [40]. The purpose of precipitation nowcasting is to predict future precipitation intensity over a relatively short period of time in a local area, and it can be seen as a video prediction problem with a fixed camera with the weather radar [41]. It was shown that ConvLSTM extracts better spatiotemporal correlations than the fully connected LSTM for precipitation nowcasting [40]. We follow the formulation of ConvLSTM defined by [40], where '$\circledast$' and '$\cdot$' denote the convolution operator and Hadamard product, respectively:

$$\begin{aligned}
i_t &= \sigma(W_{xi} \circledast x_t + W_{hi} \circledast h_{t-1} + W_{ci} \cdot c_{t-1} + b_i)\\
f_t &= \sigma(W_{xf} \circledast x_t + W_{hf} \circledast h_{t-1} + W_{cf} \cdot c_{t-1} + b_f)\\
c_t &= f_t \cdot c_{t-1} + i_t \cdot \tanh(W_{xc} \circledast x_t + W_{hc} \circledast h_{t-1} + b_c)\\
o_t &= \sigma(W_{xo} \circledast x_t + W_{ho} \circledast h_{t-1} + W_{co} \cdot c_t + b_o)\\
h_t &= o_t \cdot \tanh(c_t).
\end{aligned} \tag{7}$$

Inspired by the mentioned study, ConvLSTM was used to build a new architecture for video classification, which takes the advantage of its capacity in capturing

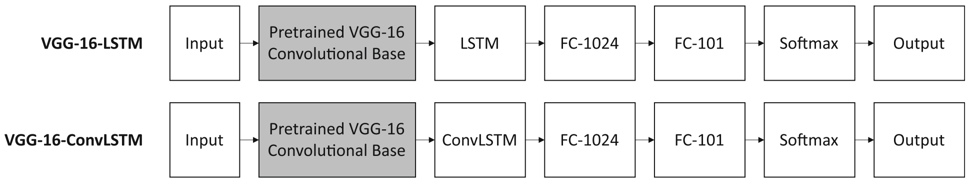

Fig. 3 The architectures of the networks used in VGG-16-LSTM and VGG-16-ConvLSTM experiments

spatiotemporal information throughout time series. We add one ConvLSTM layer on top of the spatial feature maps extracted by VGG-16 and use the hidden states for video classification. This layer contains 64 hidden states, 7 × 7 kernels, and the stride of convolution is set to 1 to perform the experiments described in Sect. 4.

4 Experiments

In this section, the datasets used, the experimental setup, and the evaluation method are described.

4.1 Datasets

In this study, the UCF-101 and the KTH datasets are used to evaluate the neural network architectures with the proposed keyframe extraction method and two more keyframe extraction methods for classifying human actions from video clips. The UCF-101 dataset includes 13,320 clips from 101 non-overlapping classes, with a resolution of 240 × 360 pixels. All clips in the UCF-101 have a fixed frame rate of 25 frames per second (FPS). The minimum and maximum lengths of the clips are 1.06 s and 71.04 s, respectively. The KTH action recognition dataset consists of six types of human actions with over 2300 video sequences. Clips in this dataset have a fixed frame rate of 25 FPS and resolution of 160 × 120 pixels.

The UCF-101 dataset has defined three training–testing splits, aiming to facilitate benchmarking algorithms. In our previous experiment [18], only the first 20 categories of the dataset were used due to limited time and computing facility, and the first training–testing split was adopted to generate training and testing data. However, we conducted the experiments in this paper using all categories with the three training–testing splits of the UCF-101 and the KTH datasets.

In the experiments, hold-out method was used to split training data into two subsets: 70% for training and the remaining 30% for validation. Testing dataset was never used during training and validation, but only used for producing the testing accuracy of each tested method.

4.2 Experiment design

During the training process, parameter tuning is carried out with the hold-out validation technique. The best parameters are identified based on the validation scores. After that, the model with the best parameters is evaluated on testing data by predicting unseen test videos' classes.

The proposed network architectures are implemented by using TensorFlow-gpu v1.12 on an GPU NVIDIA TITAN X using the CUDA v9.0 toolkit. The batch size is set to 128, and the cost is minimized by using the stochastic gradient descent (SGD) optimizer. The number of epochs is determined using early stopping by observing the change in validation loss. Dropout is used as a regularization method, disabling some neurons within ConvLSTM and fully connected layers with a probability of 0.5.

4.3 Evaluation method

In the experiments, confusion matrices are produced for performance analysis and accuracy is used for the comparison of the performances achieved by different architectures. 180 training, validation, and testing accuracy scores have been collected with the three training–testing splits released by the UCF-101 organization (10 times per split and 30 times per classifier). Similarly, 90 accuracy scores have been collected with the official training, validation, and test splits of the KTH dataset (15 times per classifier).

The Kolmogorov–Smirnov test is a normality test which compares the observed cumulative distribution with the cumulative distribution that would occur if the data were normally distributed [42], and it has been used for calculating numerical means for assessing normality. As for the test of homogeneity of variances, Levene statistic has been applied to the dependent variable and shows variances of groups are homogenous based on mean and median. Levene's test is simply a one-way analysis of variance on the absolute values of the differences between each observation and the mean of its group and is appropriate for testing the null hypothesis [43]. Furthermore, ANOVA test has been conducted to compare the variance differences to figure out whether the results are significant. Afterward, Tukey's honest significant difference (HSD) test has been run to determine whether the specific groups' means are different. The results are presented in Sect. 5.

Table 1 Average accuracy scores achieved by different architectures on the UCF-101 dataset with 20 categories [18]

Model	Training accuracy (%)	Validation accuracy (%)	Testing accuracy (%)
VGG-16-ConvLSTM	99.24	97.45	**82.04**
VGG-16-LSTM(a)	98.86	97.91	81.27
VGG-16-BLSTM(a)	97.76	95.19	76.20
VGG-16-VOTE(a)	78.71	76.55	73.20
VGG-16-LSTM(b)	78.41	79.47	67.62
VGG-16-VOTE(b)	73.70	67.05	64.04
VGG-16-BLSTM(b)	89.17	75.37	61.18

The highest accuracy score is highlighted in bold

Table 2 Average accuracy scores achieved by LSTM and ConvLSTM network architectures based on the UCF-101 (20 categories) using two keyframe extraction methods where (1) and (2) indicate one frame per second and the proposed method, respectively

Architecture	Training accuracy (%)	Validation accuracy (%)	Testing accuracy (%)
ConvLSTM(2)	98.95	98.37	**88.15**
LSTM(2)	96.48	95.53	83.10
ConvLSTM(1)	99.24	97.45	82.04
LSTM(1)	98.86	97.91	81.27

The highest accuracy score is highlighted in bold

5 Results and discussion

The VGG-16-ConvLSTM and VGG-16-LSTM architectures presented in our previous work [18] for video classification are used as baseline methods to evaluate the proposed method in this paper. One of the findings of the previous work [18] is that using global features can help achieve better classification performance over local features. It can be highlighted that the fundamental difference between local and global features is the way of representing input frames in terms of the whole frame or frame patches, which provide different information on the input to the video classifier. Seven different classification networks using either local or global features extracted by the pre-trained VGG-16 were compared in the previous work [18]. The extracted features were fed into a newly added fully connected layer in baseline VGG-16-VOTE (a), and the fully connected layer of VGG-16 was included in VGG-16-VOTE (b). Similar to the baseline methods, LSTM was employed to access spatiotemporal information over the features in VGG-16-LSTM (a) and VGG-16-LSTM (b). To test the effect of directional connections in LSTM structure for action recognition, VGG-16-BLSTM (a) and VGG-16-BLSTM (b) were implemented by using Bidirectional LSTM. The VGG-16-ConvLSTM architecture was proposed with convolutional structures in state transitions. Table 1 shows the results obtained from the earlier study [18] in which VGG-16-ConvLSTM (82.04%) significantly outperformed the other networks followed by VGG-16-LSTM (81.27%) with local features at 0.05

Table 3 Average accuracy scores achieved by LSTM and ConvLSTM network architectures on datasets KTH and UCF-101 (101 categories) using three keyframe extraction methods where (1), (2), and (3) indicate one frame per second method, the proposed action template-based method, and optical flow-based keyframe extraction method, respectively

Architecture	Dataset	Training accuracy (%)	Validation accuracy (%)	Testing accuracy (%)
ConvLSTM(2)	KTH	87.41	67.34	**71.13**
LSTM(2)	KTH	65.06	85.50	68.66
ConvLSTM(1)	KTH	84.16	69.52	68.27
ConvLSTM(3)	KTH	82.02	54.92	66.17
LSTM(1)	KTH	58.43	82.26	64.58
LSTM(3)	KTH	76.85	45.12	62.83
ConvLSTM(2)	UCF-101	94.27	90.02	**67.39**
ConvLSTM(1)	UCF-101	97.19	92.19	65.44
LSTM(2)	UCF-101	93.79	86.76	64.27
LSTM(1)	UCF-101	94.09	89.64	63.86
ConvLSTM(3)	UCF-101	87.71	82.75	60.04
LSTM(3)	UCF-101	92.96	85.46	49.62

The highest accuracy scores are highlighted in bold

Table 4 One-way ANOVA of performance achieved by different network architectures where d*f*, SS, MS, and *F* refer to degrees of freedom, sum of squares, mean sum of squares, and *F* score, respectively

	df	SS	MS	*F*	*p* value
Between groups	11	0.847	0.077	304.725	.000*
Within groups	258	0.065	0.000		
Total	269	0.913			

The values are significant at the 0.05 level

significance level ($p = .046$). In our previous study, one frame per second was extracted to reduce the amount of frames in classification.

In this paper, experiments with the proposed keyframe extraction method were conducted using the first 20 categories of the UCF-101 dataset in the first place to investigate how the proposed keyframe extraction method can improve the video classification performance of the LSTM- and ConvLSTM-based network architectures in comparison with our previous work [18].

Table 2 shows the results obtained on the first 20 categories of UCF-101 in which ConvLSTM(1) and LSTM(1) refer to the previous method selecting one frame per second, whereas ConvLSTM(2) and LSTM(2) indicate the proposed method. It can be seen that the architectures using keyframes extracted by the proposed method, ConvLSTM(2) and LSTM(2), outperformed the previous

Table 5 Post hoc comparisons using Tukey's HSD on KTH dataset

Method		Mean difference ($I - J$)	SE	*p* value	95% confidence interval	
(*I*)	(*J*)				Lower bound	Upper bound
LSTM(1)	ConvLSTM(1)	– 0.016*	0.003	.000	– 0.023	– 0.008
	LSTM(2)	– 0.004	0.003	.612	– 0.012	0.003
	ConvLSTM(2)	– 0.035*	0.003	.000	– 0.043	– 0.028
	LSTM(3)	0.142*	0.003	.000	0.135	0.150
	ConvLSTM(3)	0.038*	0.003	.000	0.031	0.046
ConvLSTM(1)	LSTM(1)	0.016*	0.003	.000	0.008	0.023
	LSTM(2)	0.012*	0.003	.000	0.004	0.019
	ConvLSTM(2)	– 0.020*	0.003	.000	– 0.027	– 0.012
	LSTM(3)	0.158*	0.003	.000	0.151	0.166
	ConvLSTM(3)	0.054*	0.003	.000	0.046	0.062
LSTM(2)	LSTM(1)	0.004	0.003	.612	– 0.003	0.012
	ConvLSTM(1)	– 0.012*	0.003	.000	– 0.019	– 0.004
	ConvLSTM(2)	– 0.031*	0.003	.000	– 0.039	– 0.024
	LSTM(3)	0.147*	0.003	.000	0.139	0.154
	ConvLSTM(3)	0.042*	0.003	.000	0.035	0.050
ConvLSTM(2)	LSTM(1)	0.035*	0.003	.000	0.028	0.043
	ConvLSTM(1)	0.020*	0.003	.000	0.012	0.027
	LSTM(2)	0.031*	0.003	.000	0.024	0.039
	LSTM(3)	0.178*	0.003	.000	0.170	0.185
	ConvLSTM(3)	0.074*	0.003	.000	0.066	0.081
LSTM(3)	LSTM(1)	– 0.142*	0.003	.000	– 0.150	– 0.135
	ConvLSTM(1)	– 0.158*	0.003	.000	– 0.166	– 0.151
	LSTM(2)	– 0.147*	0.003	.000	– 0.154	– 0.139
	ConvLSTM(2)	– 0.178*	0.003	.000	– 0.185	– 0.170
	ConvLSTM(3)	– 0.104*	0.003	.000	– 0.112	– 0.097
ConvLSTM(3)	LSTM(1)	– 0.038*	0.003	.000	– 0.046	– 0.031
	ConvLSTM(1)	– 0.054*	0.003	.000	– 0.062	– 0.046
	LSTM(2)	– 0.042*	0.003	.000	– 0.050	– 0.035
	ConvLSTM(2)	– 0.074*	0.003	.000	– 0.081	– 0.066
	LSTM(3)	0.104*	0.003	.000	0.097	0.112

The values are significant at the 0.05 level

Table 6 Post hoc comparisons using Tukey's HSD on UCF-101 dataset

Method		Mean difference $(I - J)$	SE	p value	95% confidence interval	
(I)	(J)				Lower bound	Upper bound
LSTM(1)	ConvLSTM(1)	− 0.037*	0.009	.001	− 0.062	− 0.012
	LSTM(2)	− 0.041*	0.009	.000	− 0.066	− 0.016
	ConvLSTM(2)	− 0.065*	0.009	.000	− 0.091	− 0.040
	LSTM(3)	0.017	0.009	.340	− 0.008	0.043
	ConvLSTM(3)	− 0.016	0.009	.450	− 0.041	0.009
ConvLSTM(1)	LSTM(1)	0.037*	0.009	.001	0.012	0.062
	LSTM(2)	− 0.004	0.009	.998	− 0.029	0.021
	ConvLSTM(2)	− 0.029*	0.009	.017	− 0.054	− 0.003
	LSTM(3)	0.054*	0.009	.000	0.029	0.080
	ConvLSTM(3)	0.021	0.009	.159	− 0.004	0.046
LSTM(2)	LSTM(1)	0.041*	0.009	.000	0.016	0.066
	ConvLSTM(1)	0.004	0.009	.998	− 0.021	0.029
	ConvLSTM(2)	− 0.025	0.009	.059	− 0.050	0.001
	LSTM(3)	0.058*	0.009	.000	0.033	0.083
	ConvLSTM(3)	0.025	0.009	.055	0.000	0.050
ConvLSTM(2)	LSTM(1)	0.065*	0.009	.000	0.040	0.091
	ConvLSTM(1)	0.029*	0.009	.017	0.003	0.054
	LSTM(2)	0.025	0.009	.059	− 0.001	0.050
	LSTM(3)	0.083*	0.009	.000	0.058	0.108
	ConvLSTM(3)	0.050*	0.009	.000	0.024	0.075
LSTM(3)	LSTM(1)	− 0.017	0.009	.340	− 0.043	0.008
	ConvLSTM(1)	− 0.054	0.009	.000	− 0.080	− 0.029
	LSTM(2)	− 0.058*	0.009	.000	− 0.083	− 0.033
	ConvLSTM(2)	− 0.083*	0.009	.000	− 0.108	− 0.058
	ConvLSTM(3)	− 0.033*	0.009	.003	− 0.059	− 0.008
ConvLSTM(3)	LSTM(1)	0.016	0.009	.450	− 0.009	0.041
	ConvLSTM(1)	− 0.021	0.009	.159	− 0.046	0.004
	LSTM(2)	− 0.025	0.009	.055	− 0.050	0.000
	ConvLSTM(2)	− 0.050*	0.009	.000	− 0.075	− 0.024
	LSTM(3)	0.033*	0.009	.003	0.008	0.059

The values are significant at the 0.05 level

method, achieving accuracy scores of 88.15% and 83.10%, respectively.

In order to draw more convincing conclusions, further experiments were conducted with the proposed method (2) using all the 101 categories of the UCF-101 dataset and the KTH dataset in comparison with two commonly used keyframe extraction methods: Method (1) is a baseline method that extracts one frame for each second until the end of the video and method (3) is a motion-based keyframe extraction method that selects keyframes considering the local minima of action between optical flows [29]. The results are summarized in Table 3, in which ConvLSTM(2) achieved the best classification accuracy (71.13%) followed by LSTM(2), ConvLSTM(1), ConvLSTM(3), LSTM(1), and LSTM(3) on the KTH dataset, respectively. Similarly, ConvLSTM(2) outperformed the other methods on the UCF-101 dataset with 67.39% accuracy score. As shown in Table 4, there is a statistically significant difference between classifier groups in terms of one-way ANOVA $(F(11.258) = 304.725, p = .000)$.

The Tukey's HSD post hoc test results on the KTH dataset, as shown in Table 5, show that the classification performance achieved by ConvLSTM(2) is statistically significantly higher than LSTM(1), ConvLSTM(1), LSTM(2), LSTM(3), and ConvLSTM(3) $(p<.05)$ on the KTH dataset. There is a statistically significant difference between all methods except for LSTM(1) and LSTM(2).

The Tukey's HSD post hoc test results on the UCF-101 dataset are presented in Table 6, which demonstrate that ConvLSTM(2) significantly outperformed LSTM(1), ConvLSTM(1), LSTM(3), and ConvLSTM(3) ($p < .05$).

The keyframe extraction method proposed in this paper uses action templates to identify most important regions related to actions in video frames. The experimental results have demonstrated that this action template-based approach to keyframe extraction can extract frames with distinctive actions to significantly improve the performance of deep convolutional neural networks for action recognition from videos. Keyframe extraction methods have been investigated due to their adaptability to video summarization systems and performance improvement in video classification approaches. Keyframe extraction methods enable using more informative input representation while reducing the number of frames. With the advantage of using fewer but more informative frames, the input dimension is reduced and training time is shortened. Moreover, using selected keyframes can effectively improve the accuracy in video classification.

6 Conclusion

In this paper, a template-based keyframe extraction method is proposed which employs action template-based similarity to extract keyframes for video classification tasks. Combining pre-trained CNN with ConvLSTM has achieved the highest classification accuracy among the other architectures. It can be seen that calculating structural similarity between two relevant regions of consecutive frames effectively prevents dynamic background noise from being treated as actions in keyframe selection. The experimental results and the conducted analysis show that the proposed keyframe extraction method can select informative frames reliably and thus significantly improve the performance of deep neural network architectures for video classification. Finally, when finding the relevant area using the extracted action template, the proposed method successfully extracts proper keyframes from human action videos for video classification using deep neural networks. Although the proposed method has outperformed the two commonly used keyframe extraction methods, this study has a few limitations. One of the limitations is that CNN architecture used in the evaluation of the proposed method was not the state-of-the-art architecture. This means that it did not produce the best results. The second limitation of the study is that the technical infrastructure of the experiment was weak with one GPU machine only and could not conduct more comprehensive experiments with larger batches. However, the proposed keyframe extraction method has significantly outperformed the commonly used keyframe extraction methods on two different datasets. Therefore, future work could be focused the on application of the proposed algorithm using more powerful architectures for real-world video classification and video summarizing problems.

Acknowledgements This paper was funded by (1) the Turkish Ministry of National Education, (2) the Spanish Ministry of Science, Innovation, and Universities under Grant PGC2018-098813-B-C31, and (3) ERDF fund.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Clement J (2020) Hours of video uploaded to YouTube every minute 2007–2019. https://www.statista.com/statistics/259477/hours-of-video-uploaded-to-youtube-every-minute. Accessed May 25, 2021
2. Ren S, He K, Girshick R, Sun J (2015) Faster R-NN: towards real-time object detection with region proposal networks. In: Advances in neural information processing systems, pp 91–99
3. Girshick R, Donahue J, Darrell T, Malik J (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 580–587
4. Yin X, Liu X (2018) Multi-task convolutional neural network for pose-invariant face recognition. IEEE Trans Image Process 27(2):964–975
5. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp 1097–1105
6. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556
7. Ballas N, Yao L, Pal C, Courville A (2015) Delving deeper into convolutional networks for learning video representations. arXiv:1511.06432
8. Donahue J, Anne Hendricks L, Guadarrama S, Rohrbach M, Venugopalan S, Saenko K, Darrell T (2015) Long-term recurrent convolutional networks for visual recognition and description. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2625–2634
9. Kuehne H, Jhuang H, Garrote E, Poggio T, Serre T (2011) HMDB: a large video database for human motion recognition. In:

2011 international conference on computer vision. IEEE, pp 2556–2563
10. Soomro K, Zamir AR, Shah M (2012) Ucf101: a dataset of 101 human actions classes from videos in the wild. arXiv preprint arXiv:12120402
11. Karpathy A, Toderici G, Shetty S, Leung T, Sukthankar R, Fei-Fei L (2014) Large-scale video classification with convolutional neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1725–1732
12. Abu-El-Haija S, Kothari N, Lee J, Natsev P, Toderici G, Varadarajan B, Vijayanarasimhan S (2016) Youtube-8m: a large-scale video classification benchmark. arXiv preprint arXiv:160908675
13. Carreira J, Zisserman A (2017) Quo vadis, action recognition? A new model and the kinetics dataset. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4724–4733
14. Duan H, Zhao Y, Xiong Y, Liu W, Lin D (2020) Omni-sourced webly-supervised learning for video recognition. arXiv preprint arXiv:200313042
15. Kalfaoglu M, Kalkan S, Alatan AA (2020) Late temporal modeling in 3D CNN architectures with BERT for action recognition. arXiv preprint arXiv:200801232
16. Mao F, Wu X, Xue H, Zhang R (2018) Hierarchical video frame sequence representation with deep convolutional graph network. In: Proceedings of the European conference on computer vision (ECCV)
17. Qiu Z, Yao T, Ngo CW, Tian X, Mei T (2019) Learning spatio-temporal representation with local and global diffusion. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 12056–12065
18. Kızıltepe RS, Gan JQ, Escobar JJ (2019) Combining very deep convolutional neural networks and recurrent neural networks for video classification. In: International work-conference on artificial neural networks. Springer, pp 811–822
19. Truong BT, Venkatesh S (2007) Video abstraction: a systematic review and classification. ACM Trans Multimed Comput Commun Appl (TOMM) 3(1):3
20. Boreczky JS, Rowe LA (1996) Comparison of video shot boundary detection techniques. J Electron Imaging 5(2):122–129
21. Nagasaka A, Tanaka Y (1992) Automatic video indexing and full-video search for object appearances. In: Proceedings of the IFIP TC2/WG 2.6 second working conference on visual database systems II, pp 113–127
22. Pal G, Rudrapaul D, Acharjee S, Ray R, Chakraborty S, Dey N (2015) Video shot boundary detection: a review. In: Advances in intelligent systems and computing. Springer, pp 119–127
23. Zhuang Y, Rui Y, Huang TS, Mehrotra S (1998) Adaptive key frame extraction using unsupervised clustering. Proc IEEE Int Conf Image Process 1:866–870
24. Gresle P, Huang T (1997) Gisting of video documents: a key frames selection algorithm using relative activity measure. In: The 2nd international conference on visual information systems, pp 279–286
25. Zhang HJ, Wu J, Zhong D, Smoliar SW (1997) An integrated system for content-based video retrieval and browsing. Pattern Recognit 30(4):643–658
26. Geetha P, Pandeeswari ST, Mohanan S (2012) Visual attention based keyframes extraction and video summarization. In: Computer science conference proceedings in computer science & information technology (CS&IT). Citeseer, pp 179–190
27. Barhoumi W, Zagrouba E (2013) On-the-fly extraction of key frames for efficient video summarization. Proc AASRI Conf Intell Syst Control 4:78–84
28. Thakre K, Rajurkar A, Manthalkar R (2016) Video partitioning and secured keyframe extraction of MPEG video. Phys Proc 78:790–798
29. Wolf W (1996) Key frame selection by motion analysis. Proc IEEE Int Conf Acoust Speech Signal Process 2:1228–1231
30. Abdullah SNHS, Ng KW (2017) Action key frames extraction using L1-norm and accumulative optical flow for compact video shot summarisation. In: Advances in visual informatics: 5th international visual informatics conference, IVIC 2017, Bangi, Malaysia, November 28–30, 2017, proceedings, vol 10645. Springer, p 364
31. Bao G, Li D, Mei Y (2020) Key frames extraction based on optical-flow and mutual information entropy. J Phys Conf Ser 1646(1):012–112
32. Kuanar SK, Panda R, Chowdhury AS (2013) Video key frame extraction through dynamic Delaunay clustering with a structural constraint. J Vis Commun Image Represent 24(7):1212–1227
33. Tan L, Song Y, Ma Z, Lv X, Dong X (2020) Deep learning video action recognition method based on key frame algorithm. In: Sun X, Wang J, Bertino E (eds) Artificial intelligence and security. Springer, Cham, pp 62–73
34. Xu D, Tian Y (2015) A comprehensive survey of clustering algorithms. Ann Data Sci 2(2):165–193
35. Wang Z, Bovik AC, Sheikh HR, Simoncelli EP et al (2004) Image quality assessment: from error visibility to structural similarity. IEEE Trans Image Process 13(4):600–612
36. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
37. Ogawa T, Sasaka Y, Maeda K, Haseyama M (2018) Favorite video classification based on multimodal bidirectional LSTM. IEEE Access 6:61401–61409
38. Yue-Hei Ng J, Hausknecht M, Vijayanarasimhan S, Vinyals O, Monga R, Toderici G (2015) Beyond short snippets: deep networks for video classification. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4694–4702
39. Graves A (2013) Generating sequences with recurrent neural networks. arXiv:1308.0850
40. Xingjian S, Chen Z, Wang H, Yeung DY, Wong WK, Woo WC (2015) Convolutional LSTM network: a machine learning approach for precipitation nowcasting. In: Advances in neural information processing systems, pp 802–810
41. Shi X, Gao Z, Lausen L, Wang H, Yeung DY, Wong Wk, Woo Wc (2017) Deep learning for precipitation nowcasting: a benchmark and a new model. In: Advances in neural information processing systems, pp 5617–5627
42. Freund RJ, Mohr D, Wilson WJ (2010) Statistical methods. Academic Press, Cambridge
43. Glass GV (1966) Testing homogeneity of variances. Am Educ Res J 3(3):187–190