ACTION RECOGNITION IN VIDEO SEQUENCES USING DEEP BI-DIRECTIONAL LSTM WITH CNN FEATURES

Seminar Report

Submitted in partial fulfillment of the requirements for the award of degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

of

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

Submitted By

AFNA SALAM

Department of Computer Science & Engineering
Mar Athanasius College Of Engineering Kothamangalam

ACTION RECOGNITION IN VIDEO SEQUENCES USING DEEP BI-DIRECTIONAL LSTM WITH CNN FEATURES

Seminar Report

Submitted in partial fulfillment of the requirements for the award of degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

of

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

Submitted By

AFNA SALAM

Department of Computer Science & Engineering
Mar Athanasius College Of Engineering Kothamangalam

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
MAR ATHANASIUS COLLEGE OF ENGINEERING
KOTHAMANGALAM

CERTIFICATE

This is to certify that the report entitled **Action Recognition in Video Sequences using Deep Bi-directional LSTM with CNN features** *submitted by* ***Mr. AFNA SALAM****, Reg. No.* ***MAC15CS017*** *towards partial fulfillment of the requirement for the award of Degree of Bachelor of Technology in Computer science and Engineering from APJ Abdul Kalam Technological University for December 2018 is a bonafide record of the seminar carried out by her under our supervision and guidance.*

.....	.....	.....
Prof. Joby George	**Prof. Neethu Subash**	**Dr. Surekha Mariam Varghese**
Faculty Guide	***Faculty Guide***	***Head of the Department***

Date: Dept. Seal

ACKNOWLEDGEMENT

First and foremost, I sincerely thank the God Almighty for his grace for the successful and timely completion of the seminar.

I express my sincere gratitude and thanks to Dr. Solly George, Principal and Dr. Surekha Mariam Varghese, Head Of the Department for providing the necessary facilities and their encouragement and support.

I owe special thanks to the staff-in-charge Prof. Joby george, Prof. Neethu Subash and Prof. Joby Anu Mathew for their corrections, suggestions and sincere efforts to co-ordinate the seminar under a tight schedule.

I express my sincere thanks to staff members in the Department of Computer Science and Engineering who have taken sincere efforts in helping me to conduct this seminar.

Finally, I would like to acknowledge the heartfelt efforts, comments, criticisms, co-operation and tremendous support given to me by my dear friends during the preparation of the seminar and also during the presentation without whose support this work would have been all the more difficult to accomplish.

ABSTRACT

Action recognition in video sequences is a challenging problem of computer vision due to the similarity of visual contents. Recurrent neural network (RNN) and long short-term memory (LSTM) have achieved great success in processing sequential multimedia data and can be used to yield the state-of-the-art results in action recognition in video sequences. First, deep features are extracted using CNN from every sixth frame of the videos, which helps reduce the redundancy and complexity. Next, the sequential information among frame features is learnt using DB-LSTM network. It is capable of learning long term sequences and can process lengthy videos by analyzing features for a certain time interval. Experimental results show significant improvements in action recognition using the proposed method on three benchmark data sets including UCF-101, YouTube 11 Actions, and HMDB51 compared with the state-of-the-art action recognition methods.

Contents

List of Figures

List of Abbreviation

LSTM	Long Short Term Memory
RNN	Recurrent Neural Network
CNN	Concurrent Neural Network
DB-LSTM	Deep Bidirectional Long Short Term Memory
kNN	Kth Nearest Neighbour
ML-LSTM	Multi Layer- Long Short Term Memory
STF	Space Time Features
SVM	Support Vector Machine
STV	Space Time Volume
MHI	Motion History Image
HMDB	Human Metabolome Database

Introduction

Action recognition in video sequences is a challenging problem of computer vision due to the similarity of visual contents [1], changes in the viewpoint for the same actions, camera motion with action performer, scale and pose of an actor, and different illumination conditions [2]. Human actions range from simple activity through arm or leg to complex integrated activity of combined arms,legs,and body. For example,the legs motion for kicking a football is a simple action, while jumping for a head-shoot is a collective motion of legs, arms, head, and whole body [3]. Generally, human action is a motion of body parts by interacting with objects in the environment. In the context of videos, an action is represented using a sequence of frames, which humans can easily understand by analyzing contents of multiple frames in sequence.In this paper,we recognize human actions in a way similar to our observation of actions in real life.We use LSTM to consider the information of previous frames in automatic understanding of actions in videos.

One of the key motivations, which attracts researchers to work in action recognition, is the vast domain of its applications in surveillance videos, robotics, human-computer interaction, sports analysis, video games for player characters, and management of web videos. Action recognition using video analysis is computationally expensive as processing a short video may take a long time due to its high frame rate. As each frame plays an important role in a video story, keeping information of sequential frames for long time,makes the system more efficient. Researchers have presented many solutions for this problem such as motion, space-time features, and trajectories. The proposed method uses recurrent neural network LSTM to analyze frame to frame change of action videos. RNNs are building blocks of a connected neuron with input units, internal (or hidden) units, and output units, having an activation at time t, which can selectively process data in sequence. As it processes one element at a time, it can model outputs, consisting of sequence of elements that are not independent [4].

The RNN architecture provides strength to processing and nding hidden patterns in time-space data such as audio, video, and text. RNN processes data in sequential way such that at

each time t, it gets input from the previous hidden state and new data. The data is also multiplied with weights, biases are added, and is fed to activation functions. Due to the large number of calculations, the effect of the initial inputs becomes negligible for the upcoming sequence of data after few layers, resulting in vanishing gradient problem.The solution to this problem is LSTM. The main idea of LSTM architecture is its memory cell,input gate,output gate, and forget gate, which can maintain its state over time T_N, and non-linear gating units which regulate the information ow into/out of the cell. Researchers have presented different variations of LSTM such as multi-layer LSTM and bidirectional LSTM for processing sequential data. The proposed method analyzes the complex pattern in the visual data of each frame, which cannot be efficiently identified using simple LSTM and multi-layer LSTM.

In the proposed method, features of video frames are analyzed for action recognition. Deep features from every sixth frame of a video are extracted using pre-trained AlexNet. Next, an architecture of DB-LSTM is developed with two layers at each forward and backward pass for learning sequence information in the features of video frames. The proposed method is capable of recognizing actions in long videos because the video is processed in N time steps. Our system has less computational complexity as it only processes ve frames per second. The implementation of DB-LSTM has a high capacity of learning sequences and frame to frame change in features due to small change in visual data of videos. These properties make the proposed method more suitable for action recognition in videos.

Related Works

Over the last decade, researchers have presented many hand-crafted and deep-nets based approaches for action recognition.

2.1 Hand-crafted approaches

The earlier work was based on hand-crafted features for non-realistic actions, where an actor used to perform some actions in a scene with simple background. Such systems extract low level features from the video data and then feed them to a classifier such as support vector machine (SVM), decision tree, and KNN for action recognition. n pattern recognition, the k-nearest neighbors algorithm (k-NN) is a non-parametric method used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression For instance, the geometrical properties of space time volume (STV) called action sketch, were analyzed by Yilmaz and Shah. A set of action descriptors is called an action sketch. The first step in is to generate STV by solving the point correspondence problem between consecutive frames. The correspondences are determined using a two-step graph theoretical approach. After the STV is generated, actions descriptors are computed by analyzing the differential geometric properties of STV. Finally, using these descriptors, we perform action recognition, which is also formulated as graph theoretical problem. They stacked body contours in time axis by capturing direction, speed, and shape of STV for action recognition. Gorelick et al. presented human action as three-dimensional shapes made from the silhouettes in the STV. They used the poisson equation method to analyze 2D shapes of actions and extracted space time features(STF) containing local space-time saliency, action dynamics, shape structure, and orientation. Their method used a non-realistic dataset and, in certain cases, two different actions resulted the same 2D shapes in STV, making the representation of different actions difficult.

Hu et al. used two types of features: motion history image (MHI) and histogram of

oriented gradients feature (HOG). The MHI is a static image template helps in understanding the motion location and path as it progresses. In MHI, the temporal motion information is collapsed into a single image template where intensity is a function of recency of motion. Thus, the MHI pixel intensity is a function of the motion history at that location, where brighter values correspond to a more recent motion. Using MHI, moving parts of a video sequence can be engraved with a single image, from where one can predict the motion flow as well as the moving parts of the video action. The former is the foreground image subtracted from the background scenario whereas the later one is magnitudes and directions of edges. These features were then fused and classied through a simulated annealing multiple instance learning SVM. In machine learning, support vector machines (SVMs) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting). An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall. Liuetal. extracted motion and static features for realistic videos. They pruned the noisy motion feature by applying motion statistics to acquire stable features. In addition, they also used ”PageRank” to mine the most informative static features and construct discriminative visual vocabularies. However, these hand-crafted features based methods have certain limitations. For instance, STVs based methods are not effective for recognizing multiple person actions in a scene. STF and MHI based techniques are more suitable for simple datasets. To process complex datasets, we need hybrid approaches which can combine different features and preprocessing such as motion detection, background segmentation,HOG,SIFT,and SURF. But such hybrid methods increase the computational complexity of the target system. These limitations can cause difficulty for lengthy videos and real-time applications with continuous video streaming.

2.2 Deep-net based approaches

Besides hand-crafted features based approaches for action recognition, several deep learning based methods were also proposed in recent years. Deep learning has shown significant improvement in many areas such as image classification, person re-identification, object detection,speech recognition and bioinformatics. For instance, a straight forward implementation of action recognition using deep networks is developed through 3D convolutional networks by Ji et al [5]. They applied 3D convolutional kernels on video frames in a time axis to capture both spatial and temporal information. They also claimed that their approach can capture motion and optical ow information because frames are connected by fully connected layers at the end. A multi-resolution CNN framework for connectivity of features in time domain is proposed to capture local spatio-temporal information. This method is experimentally evaluated on a new "YouTube 1 million videos dataset" of 487 classes. The authors claimed to have speed up the training complexity by foveated architecture of CNN. They improved the recognition rate for large dataset up to 63.9% but their recognition rate on UCF101 is 63.3%,which is still too low for such important task of action recognition.

A two-stream CNN architecture is proposed in which first stream captures spatial and temporal information between frames and second one demonstrates the dense optical flow of multiple frames. They have increased the amount of data for the training CNN model by combining two datasets. In a paper, authors used two CNN models for processing each individual frame of the input video for action recognition. The output of intermediate layers of both architectures is processed by special 1x1 kernels in fully connected layers. The method finally used 30 frames unrolled LSTM cell connected with the output of CNN in training.

The feature maps of pre-trained model are analyzed by Bilen et al. for video representation named as dynamic image. They added rank pooling operator and approximate rank pooling layer in fine tuning phase,which combine maps of all frames to a dynamic image as one representation of the video. Deep learning based approaches have the ability to accurately identify hidden patterns in visual data because of its huge feature representation pipeline. On the other hand, it requires huge amount of data for training and high computational power

for its processing. In this work, we have balanced the complexity of the system and action recognition accuracy. Our method is computationally efficient as it analyzes only each sixth frame of the video, which is an optimal value for frame jump verified through different experiments. For better action recognition, we have intelligently combined CNN and LSTM due to its state-of-the-art results on visual and sequential data.

The proposed framework

In this section, the proposed framework and its main components are discussed in detail including the recognition of an action A_I from the sequence of frames in video V_I using DB-LSTM and features extraction through CNN for F_N frames. The procedure for action recognition is divided into two parts: First, we extract CNN features from the frames of video V_I with jump J_F in sequence of frames such that the jump J_F does not affect the sequence of the action A_I in the video. Second, the features representing the sequence of action A_I for time interval T_S (such as T_S = 1 sec) are fed to the proposed DB-LSTM in C_N chunks, where each C_N chunk is the features representation of the video frame and input to one RNN step. At the end, the final state of each time internal T_S is analyzed for final recognition of an action in a video. The proposed framework is shown in Figure. 3.1. Each step of the proposed method is discussed in separate section. The input and output parameters of the proposed method are given in Figure. 3.2.

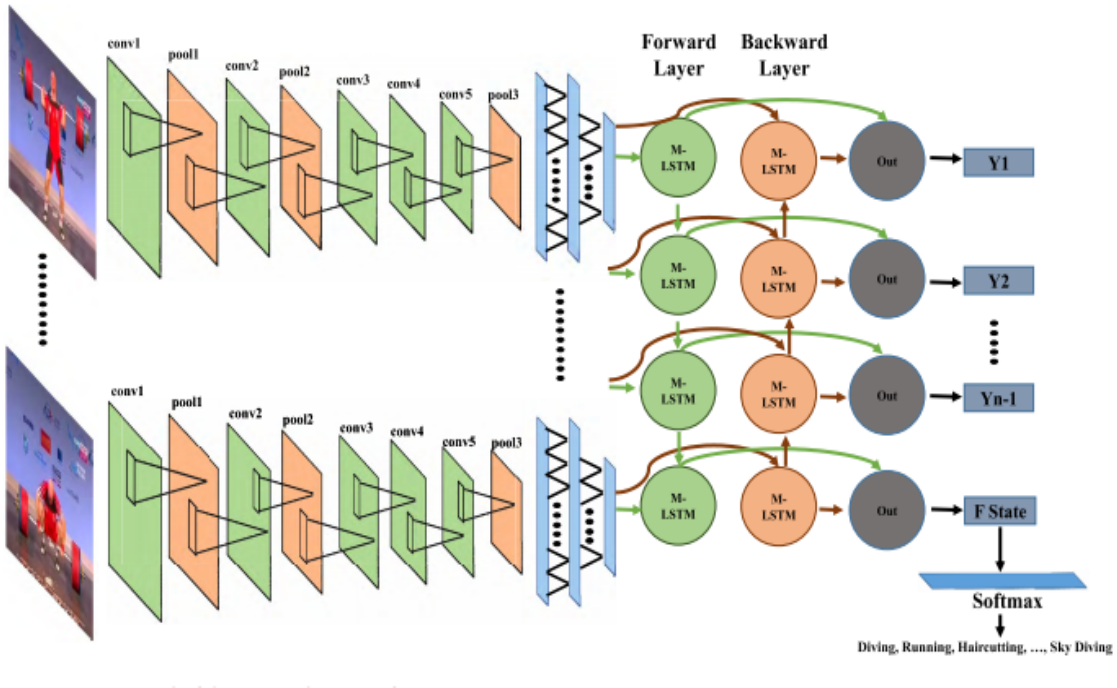

Fig. 3.1: Framework of the proposed DB-LSTM for action recognition.

V_I	Action video.
A_I	Action in video V_I.
F_N	Number of frames in video V_I
J_F	Jump between frames during extracting features
s_t	Output of current state of RNN
$b^{i,f,o}$	Biases of input, output, and forget gates of LSTM cell
T_S	Time interval of action feed to DB-LSTM.
C_N	Number of chunks in T_S.
DB-LSTM	Deep bidirectional LSTM.
FC8	Fully connected layer of CNN.
x_t	Input to RNN at time t.
$W^{i,f,o}$	Weights of input, output, and forget gates of LSTM cell

Fig. 3.2: Description of input and output parameters used in the proposed DB-LSTM for action recognition.

3.1 Preparation and feature extraction

CNN is a dominant source for the representation and classification of images. In the case of video data, each individual frame is represented by CNN features, followed by finding the sequential information between them using DB-LSTM. A video is a combination of frames moving at 30 to N frames per second. Thirty to fifty frames in a unit time have many redundant frames, whose processing is a computationally expensive process. Considering this processing complexity, we jump six frames when processing a video for action recognition. It is evident

from the experiments that a six frame jump does not affect the sequence of the action. As CNN finds hidden patterns in images, it captures all the tiny changes in each frame. These changes in sequential form are learnt through RNN for action recognition in a video.

Training a deep learning model for image representation requires thousands of images and also requires high processing power such as GPU for the weight adjustment of the CNN model. Getting the required model using this strategy is an expensive process, which is solved using transform learning[6] where a trained model can be used for other purposes. In the proposed method, we used parameters of the pre-trained CNN model, called AlexNet for feature extraction, which is trained on large scale ImageNet dataset of more than 15 million images.

AlexNet is one of the deep ConvNets designed to deal with complex scene classification task on Imagenet data. The task is to classify the given input into one of the 1000 classes. AlexNet has five convolution layers,three pooling layers,and three fully connected layers. Each layer is followed by a norm and ReLU non linear activation function. The extracted features vector from FC8 layer is one thousand dimensional. The features of each frame are considered as one chunk for one input step of RNN. C_N chunks for T_S time interval are feed to RNN. Thus for one second with six frame jump in video, we process six frames out of thirty frames. When we feed features of six frames, RNN processes it in six chunks. The final state of the RNN is counted for each T_S for final recognition.

3.2 Convolutional Neural Networks

Convolutional Neural Networks (ConvNets or CNNs) are a category of Neural Networks that have proven very effective in areas such as image recognition and classification. ConvNets have been successful in identifying faces, objects and traffic signs apart from powering vision in robots and self driving cars. CNNs, like neural networks, are made up of neurons with learnable weights and biases. Each neuron receives several inputs, takes a weighted sum over them, pass it through an activation function and responds with an output. Unlike neural networks, where the input is a vector, here the input is a multi-channeled image. Channel is a conventional term used to refer to a certain component of an image. An image from a standard digital camera will have three channels red, green and blue, each having pixel values in the

range 0 to 255. A grayscale image, on the other hand, has just one channel. The convolution layer is the main building block of a convolutional neural network. The primary purpose of Convolution in case of a ConvNet is to extract features from the input image. Convolution preserves the spatial relationship between pixels by learning image features using small squares of input data. The convolution layer comprises of a set of independent filters. Each filter is independently convolved with the image. All these filters are initialized randomly and become our parameters which will be learned by the network subsequently. A filter slides over the input image (convolution operation) to produce a feature map. The convolution of another filter over the same image gives a different feature map. A pooling layer is another building block of a CNN. Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network. Spatial Pooling (also called subsampling or downsampling) reduces the dimensionality of each feature map but retains the most important information. Spatial Pooling can be of different types: Max, Average, Sum etc. Pooling layer operates on each feature map independently. ReLU stands for Rectified Linear Unit and is a non-linear operation. RELU is just a non linearity which is applied similar to neural networks. It is an element wise operation (applied per pixel) and replaces all negative pixel values in the feature map by zero. The FC is the fully connected layer of neurons at the end of CNN. Neurons in a fully connected layer have full connections to all activations in the previous layer. The Fully Connected layer is a traditional Multi Layer Perceptron that uses a softmax activation function in the output layer (other classifiers like SVM can also be used, but will stick to softmax in this post). The term Fully Connected implies that every neuron in the previous layer is connected to every neuron on the next layer.The sum of output probabilities from the Fully Connected Layer is 1. This is ensured by using the Softmax as the activation function in the output layer of the Fully Connected Layer. The Softmax function takes a vector of arbitrary real-valued scores and squashes it to a vector of values between zero and one that sum to one.

3.2.1 3D Convolutional Neural Networks architecture

Most current methods build classifiers based on complex handcrafted features computed from the raw inputs. Convolutional neural networks (CNNs) are a type of deep model that can act directly on the raw inputs. However, such models are currently limited to handling 2D inputs. Video can naturally be decomposed into spatial and temporal components. The spatial part, in the form of individual frame appearance, carries information about scenes and objects depicted in the video. The temporal part, in the form of motion across the frames, conveys the movement of the observer (the camera) and the objects.The paper develop a novel 3D CNN model for action recognition. This model extracts features from both the spatial and the temporal dimensions by performing 3D convolutions, thereby capturing the motion information encoded in multiple adjacent frames. The developed model generates multiple channels of information from the input frames, and the final feature representation combines information from all channels. To further boost the performance, we propose regularizing the outputs with high-level features and combining the predictions of a variety of different models.

When applied to video analysis problems, it is desirable to capture the motion information encoded in multiple contiguous frames. To this end, we propose to perform 3D convolutions in the convolution stages of CNNs to compute features from both spatial and temporal dimensions. The 3D convolution is achieved by convolving a 3D kernel to the cube formed by stacking multiple contiguous frames together. By this construction, the feature maps in the convolution layer are connected to multiple contiguous frames in the previous layer, thereby capturing motion information.

A general design principle of CNNs is that the number of feature maps should be increased in late layers by generating multiple types of features from the same set of lower level feature maps. Similarly to the case of 2D convolution, this can be achieved by applying multiple 3D convolutions with distinct kernels to the same location in the previous layer.The inputs to 3D CNN models are limited to a small number of contiguous video frames due to the increased number of trainable parameters as the size of input window increases. On the other hand, many human actions span a number of frames. Hence, it is desirable to encode high-level motion information into the 3D CNN models.

3.3 Recurrent Neural Network

RNNs are introduced for analyzing hidden sequential patterns in both temporal sequential and spatial sequential data. A recurrent neural network (RNN) is a class of artificial neural network where connections between nodes form a directed graph along a sequence. This allows it to exhibit temporal dynamic behavior for a time sequence. Unlike feed-forward neural networks, RNNs can use their internal state (memory) to process sequences of inputs. Video is also sequential data in which movements in visual contents are represented in many frames such that sequence of frames help in understanding the context of an action. RNNs can interpret such sequences but forget the earlier inputs of the sequence in case of long term sequences. This problem is known as the vanishing gradient problem, which can be solved through a special type of RNN called LSTM. Long Short-Term Memory (LSTM) is a specific recurrent neural network (RNN) architecture that was designed to model temporal sequences and their long-range dependencies more accurately than conventional RNNs. It is capable of learning long term dependencies. Its special structure with input,output,and forget gates controls the long term sequence pattern identification. The gates are adjusted by a sigmoid unit that learns during training where it is to open and close. Eq. 1 to Eq. 7 [7] explain the operations performed in LSTM unit, where x_t is the input at time t (in our case it is chunk C). f_t is the forget gate at time t, which clears information from the memory cell when needed and keeps are cord of the previous frame whose information needs to be cleared from the memory. The output gate o_t keeps information about the upcoming step, where g is the recurrent unit, having activation function $tanh$ and is computed from the input of the current frame and state of the previous frame s_{t-1}. The hidden state of an RNN step is calculated through tanh activation and memory cell c_t. As the action recognition does not need the intermediate output of the LSTM, we made nal decision by applying softmax classifier on the final state of the RNN network.

$$i_t=\sigma((\mathrm{x}_t+s_{t-1})W^i + b_i) \qquad \text{(Equ: 1)}$$

$$f_t=\sigma((\mathrm{x}_t+s_{t-1})W^f + b_f) \qquad \text{(Equ: 2)}$$

$$o_t=\sigma((\mathrm{x}_t+s_{t-1})W^o + b_o) \qquad \text{(Equ: 3)}$$

g=tanh($(x_t$+$s_{t-1})W^g$ + b_g) (Equ: 4)

c_t=$c_{t-1}.f_t$+$g.i_t$ (Equ: 5)

s_t=tanh(c_{t-1}).o_t (Equ: 6)

final_state= soft max(V_{st}) (Equ: 7)

Training large data with complex sequence patterns (such as video data) are not identified by the single LSTM cell. Therefore, in the proposed approach, we use ML-LSTM by stacking multiple LSTM cells to learn long term dependencies in video data.

3.4 Conventional Long Short Term Memory

RNN with LSTM have emerged as an effective and scalable model for several learning problems related to sequential data. Earlier methods for attacking these problems have either been tailored toward a specific problem or did not scale to long time dependencies. LSTMs on the other hand are both general and effective at capturing long-term temporal dependencies. They do not suffer from the optimization hurdles that plague simple recurrent networks (SRNs) and have been used to advance the state of the art for many difficult problems. This includes handwriting recognition and generation, language modeling and translation, acoustic modeling of speech, speech synthesis, protein secondary structure prediction , analysis of audio, and video data among others.

The central idea behind the LSTM architecture is a memory cell, which can maintain its state over time, and nonlinear gating units, which regulate the information flow into and out of the cell. Most modern studies incorporate many improvements that have been made to the LSTM architecture since its original formulation. However, LSTMs are now applied to many learning problems, which differ significantly in scale and nature from the problems that these improvements were initially tested on.

The LSTM contains special units called memory blocks in the recurrent hidden layer. The memory blocks contain memory cells with self-connections storing the temporal state of

the network in addition to special multiplicative units called gates to control the flow of information. Each memory block in the original architecture contained an input gate and an output gate as shown in Figure 3.3. The input gate controls the flow of input activations into the memory cell.

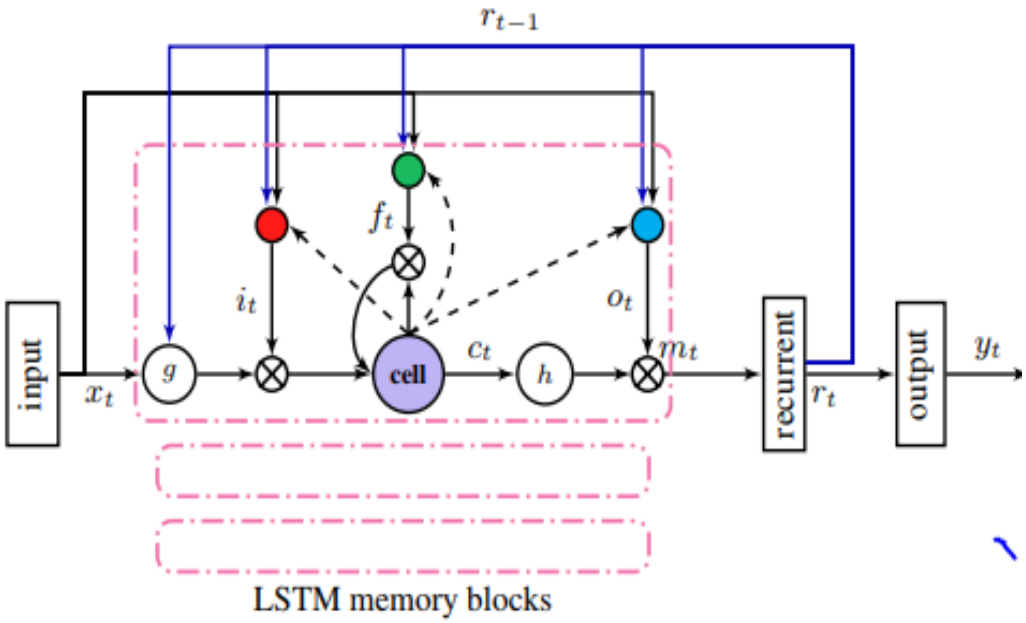

Fig. 3.3: LSTMP RNN architecture. A single memory block is shown for clarity.

The output gate controls the output flow of cell activations into the rest of the network. Later, the forget gate was added to the memory block. This addressed a weakness of LSTM models preventing them from processing continuous input streams that are not segmented into subsequences. The forget gate scales the internal state of the cell before adding it as input to the cell through the self-recurrent connection of the cell, therefore adaptively forgetting or resetting the cells memory. In addition to Figure 3.4, the modern LSTM architecture contains peephole connections from its internal cells to the gates in the same cell to learn precise timing of the outputs.

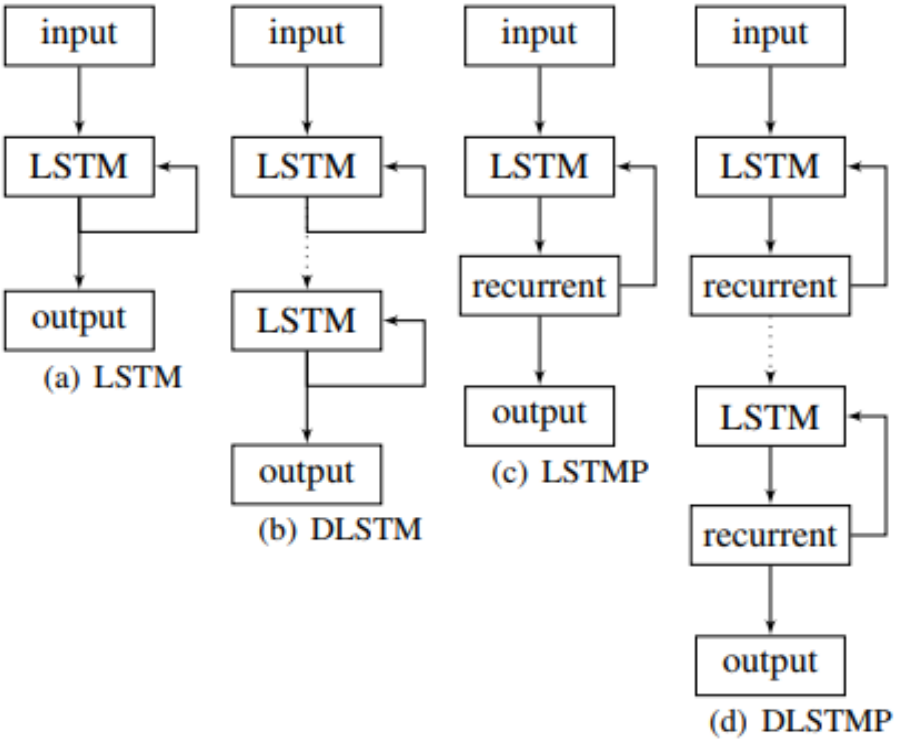

Fig. 3.4: LSTM RNN architectures.

3.4.1 LSTM training

We followed the same procedure as training max-pooled network with two modifications: First, the videos label was backpropagated at each frame rather than once per clip. Second, a gain g was applied to the gradients backpropagated at each frame. g was linearly interpolated from 0...1 over frames t = 0...T. g had the desired effect of emphasizing the importance of correct prediction at later frames in which the LSTMs internal state captured more information. Compared empirically against setting g = 1 over all time steps or setting g = 1 only at the last time step T (g = 0 elsewhere), linearly interpolating g resulted in faster learning and higher accuracy. For the final results, during training the gradients are backpropagated through the convolutional layers for fine tuning.

3.4.2 Long Short Term Memory inference

In order to combine LSTM frame level predictions into a single video-level prediction, we tried several approaches: 1) returning the prediction at the last time step T, 2) max-pooling the predictions over time, 3) summing the predictions over time and return the max 4) linearly weighting the predictions over time by g then sum and return the max. The accuracy for all four approaches was less than 1% different, but weighted predictions usually resulted in the best performance, supporting the idea that the LSTMs hidden state becomes progressively more informed as a function of the number of frames it has seen

3.5 Multi layers Long Short Term Memory

The performance of the deep neural network has been boosted by increasing the number of layers in the neural network models. The same strategy is followed here for RNN by stacking two LSTM layers to our network. By adding this new layer, RNN captures higher level of sequence information. In standard RNN, data is fed to single layer for activation and processing before output, but in time sequence problems, we need to process data on several layers. By stacking LSTM layers, each layer in the RNN is a hierarchy that receives the hidden state of the previous layer as input. Figure. 3.5 shows a multi-layer LSTM.

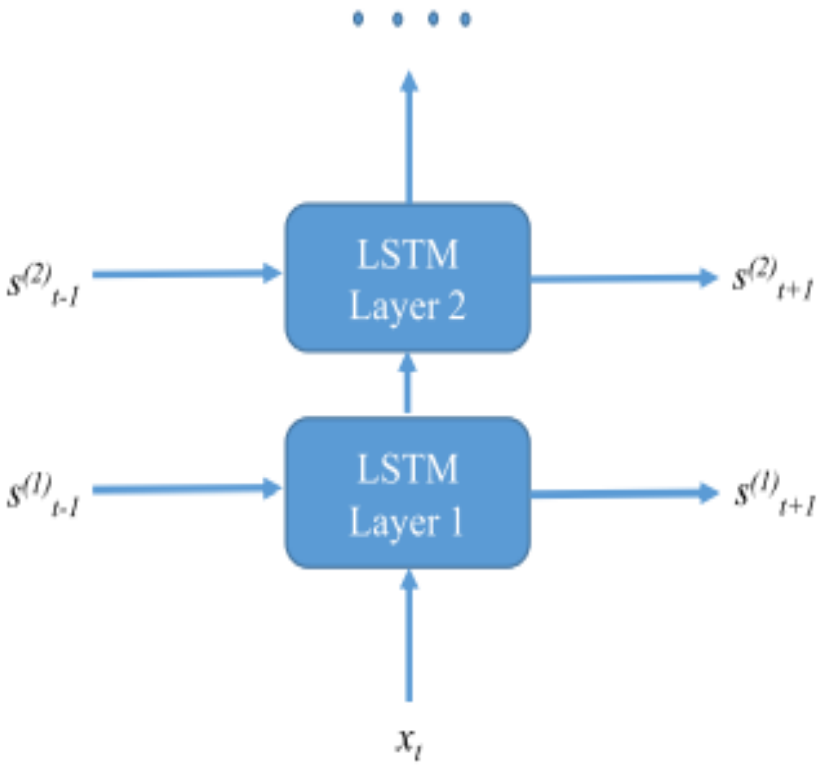

Fig. 3.5: Two layer LSTM network.

Layer 1 receives input from data x_t while the input of layer 2 is from its previous time step ${s^{(2)}}_{t-1}$, and the output of the current time step of layer one ${s^{(1)}}_t$. The computation of LSTM cell is same as Eq. 1 to Eq. 7 but only the layers information has been added to the superscript of each i_t, f_t, o_t, c_t, and s_t. Eq. 8 shows the procedure of calculating the state of a layer.

$$s_t^1 = \tanh(c_t^1).o_t^1$$

3.6 Deep Long Short Term Memory

As with DNNs with deeper architectures [8], deep LSTM RNNs have been successfully used for speech recognition. Deep LSTM RNNs are built by stacking multiple LSTM layers. Note that LSTM RNNs are already deep architectures in the sense that they can be considered as a feed-forward neural network unrolled in time where each layer shares the same model parameters. One can see that the inputs to the model go through multiple non-linear layers as in DNNs, however the features from a given time instant are only processed by a single nonlinear layer before contributing the output for that time instant. Therefore, the depth in deep LSTM RNNs has an additional meaning. The input to the network at a given time step goes through multiple LSTM layers in addition to propagation through time and LSTM layers. It has been argued that deep layers in RNNs allow the network to learn at different time scales over the input. Deep LSTM RNNs offer another benefit over standard LSTM RNNs: They can make better use of parameters by distributing them over the space through multiple layers. For instance, rather than increasing the memory size of a standard model by a factor of 2, one can have 4 layers with approximately the same number of parameters. This results in inputs going through more non-linear operations per time step.

3.7 Bidirectional Long Short Term Memory

In bidirectional LSTM, the output at time t is not only dependent on the previous frames in the sequence, but also on the upcoming frames. Bidirectional RNNs are quite simple, having two RNNs stacked on top of each other. One RNN goes in the forward direction and another one goes in the backward direction. The combined output is then computed based on the

hidden state of both RNNs. In our work, we are using multiple LSTM layers, so our scheme has two LSTM layers for both forward pass and backward pass.Fig.3.5 shows the overall concept of bidirectional LSTM used in the proposed method.

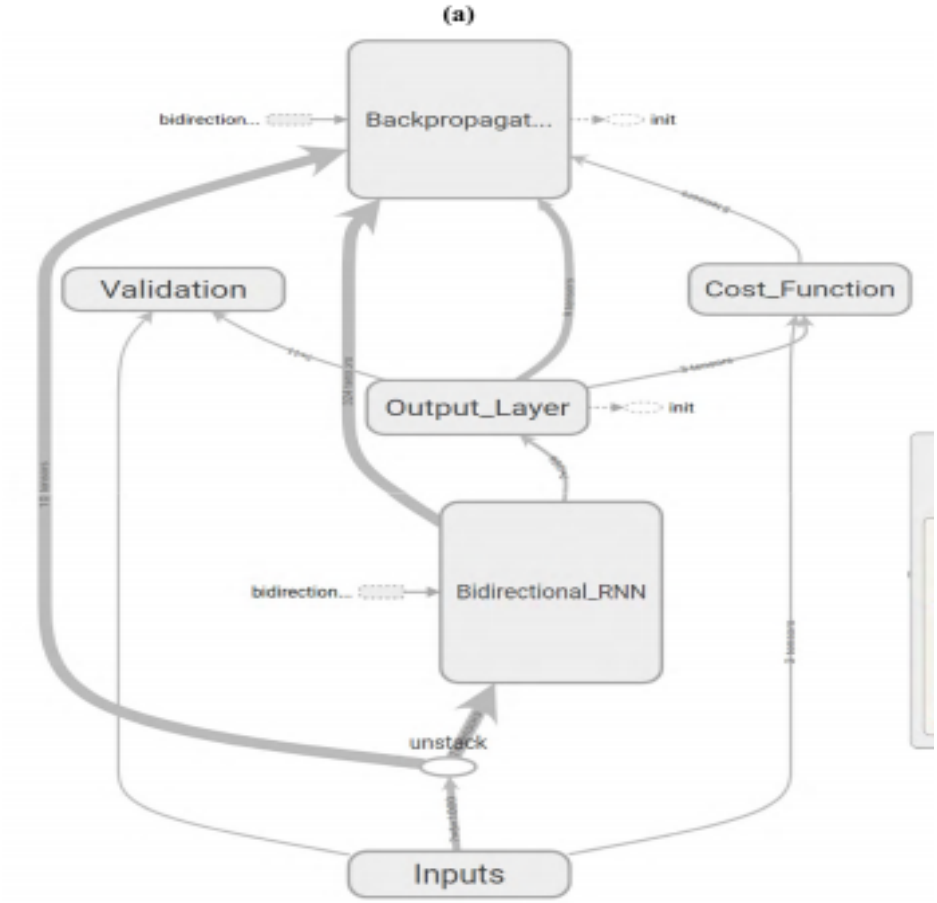

Fig. 3.6: Structure of the proposed DB-LSTM network.

Figure 3.6 shows the structure of the training phase, where the input data is fed to the bidirectional RNN, and the hidden states of forward pass and backward pass are combined in the output layer. The validation and cost is calculated after the output layer and weights and biases are adjusted through back-propagation. For validation, 20% of the data is separated from the dataset and cross entropy is used for error calculation of the validation data. Stochastic optimization with a learning rate of 0.001 isused for cost minimization. It also shows the internal structure of the bidirectional RNN, where ”fw” is forward pass and ”bw” is backward

pass. Both fw and bw consist of two LSTM cells, making our model a deep bidirectional LSTM. The proposed method outperforms other state-of-the-art methods due to its mechanism of computing the output. The output of a frame at time t is calculated from the previous frame at time t1 and the upcoming frame at time t+1 because layers are performing processing in both directions.

3.8 Performance evaluation

In this section, the proposed technique is experimentally evaluated and the results are discussed on different benchmark action recognition datasets including UCF101, Action YouTube, and HMDB51. The datasets are divided by following machine learning three splits protocol in training, validation, and testing of 60%, 20%, and 20%, respectively. We have used Caffe toolbox for deep features extraction, tensorow for DB-LSTM, and GeForce-Titan-X GPU for implementation. The training data is fed in mini batches of 512 size with a learning rate of 0.001 for cost minimization and one thousand iteration for learning the sequence patterns in the data. We have compared the proposed technique with recent state-of-the-art methods using the average accuracy score of confusion matrix as the recognition rate on each database. The comparisons with other methods are given in Figure 3.7. The recognition scores are reported from the referenced papers. Some of the cells in Figure.4.1 are blank because those methods have not reported the recognition score on the corresponding dataset.

3.8.1 Visual results and computation

The proposed method is tested on 20% videos of each dataset. The method takes a test video as input and extracts features from its frames with six frame jump. The extracted features are fed to the proposed DB-LSTM in chunks for time interval T. The DB-LSTM returns output for each chunk and finally the video is classified for the highest frequency class in outputs. The incorrect predictions are due to the similarity of visual content, motion of camera, and changes in parts of an actor body in both action categories. We have evaluated the proposed

Method	YouTube	HMDB51	UCF101
Multiresolution CNNs [21]	-	-	65.4%
LSTM with 30 frame unroll [6]	-	-	88.6%
Two-stream CNNs [22]	-	59.4%	88.0%
Multiple dynamic images [23]	-	65.2%	89.1%
RLSTM-g3 [32]	-	55.3%	86.9%
Hierarchical clustering multi-task [33]	89.7%	51.4%	76.3%
VideoDarwin [34]	-	63.7%	-
Discriminative representation [35]	91.6%	28.2%	79.7%
Ordered trajectories [8]	-	47.3%	72.8%
Factorized spatio-temporal CNNs [36]	-	59.1%	88.1%
Temporal pyramid CNNs [37]	-	63.1%	89.1%
Adaptive RNN-CNNs [38]	-	61.1%	-
Improved trajectories [39]	-	57.2%	-
Super-category exploration [40]	-	60.8%	-
Multi-layer fisher vector [41]	-	68.5%	-
Proposed DB-LSTM	**92.84**	**87.64**	**91.21**

Fig. 3.7: Comparison of average recognition score of the proposed DB-LSTM for action recognition with state-of-the-art methods.

method using different experiments with various number of frame jumps for analyzing action in videos.

Experiments	Frame Jump	Average Time Complexity	Average Accuracy
1	4	1.725 sec	92.2%
2	6	1.12 sec	91.5%
3	8	0.90 sec	85.34%

Fig. 3.8: Average time complexity and accuracy on different frame jumps for 30 FPS video clip

Figure 3.8 shows the statistics of the conducted experiments. We have used 6 frame jump in overall experiments because of its optimal results in complexity and accuracy. The proposed method is evaluated on GeForce-Titan-X GPU for feature extraction, training, and testing. The system takes approximately 0.23 sec for feature extraction per frame. Feeding the extracted features to DB-LSTM for classification takes 0.53 sec for 30 frames per second video clip. Overall, the proposed method takes approximately 1.12 seconds for processing of a 1-second

video clip. With these statistics, our method can process 25 frames per second, making it a suitable candidate for action recognition in real-time video processing applications.

Conclusion

The paper proposed an action recognition framework by utilizing frame level deep features of the CNN and processing it through DB-LSTM. First, CNN features are extracted from the video frames, which are fed to DB-LSTM, where two layers are stacked on both forward and backward pass of the LSTM. This helped in recognizing complex frame to frame hidden sequential patterns in the features. We analyzed the video in N chunks, where the number of chunks depend on the time interval ”T” for processing. Due to these properties, the proposed method is capable of learning long term complex sequences in videos. It can also process full length videos by providing prediction for time interval ”T”. The output for small chunks is combined for the final output. The experimental results indicate that the recognition score of the proposed method successfully dominates other recent state-of-the-art action recognition techniques on UCF-101, HMDB51, and YouTube action video datasets. These characteristics make our proposed method more suitable for processing of visual data and can be an integral component of smart systems. The proposed method extracts features from the whole frame of the video. In future, we aim to analyze only the salient regions of the frames for action recognition. Finally, the proposed method can be combined with people counting techniques to intelligently analyze the people crowded behavior and dense situations.

REFERENCES

[1] A. Nanda, D. S. Chauhan, P. K. Sa, and S. Bakshi, "Illumination and scale invariant relevant visual features with hypergraph-based learning for multi-shot person re-identification," Multimedia Tools Appl., pp. 126

[2] K. Soomro, A. R. Zamir, and M. Shah. (2012). UCF101: A dataset of 101 human actions classes from videos in the wild. [Online].

[3] S. Herath, M. Harandi, and F. Porikli, "Going deeper into action recognition: A survey," Image Vis. Comput., vol. 60, pp. 421, Apr. 2017.

[4] Z. C. Lipton, J. Berkowitz, and C. Elkan. (2015). "A critical review of recurrent neural networks for sequence learning."

[5] S. Ji, W. Xu, M. Yang, and K. Yu, "3D convolutional neural networks for human action recognition," IEEE Trans. Pattern Anal. Mach. Intell., vol. 35, no. 1, pp. 221231, Jan. 2013.

[6] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: An astounding baseline for recognition," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops, Jun. 2014, pp. 806813.

[7]] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in Proc. 15th Annu. Conf. Int. Speech Commun. Assoc., 2014, pp. 338342

[8] B. Mahasseni and S. Todorovic, "Regularizing long short term memory with 3D human-skeleton sequences for action recognition," in Proc. IEEE Conf. Comput. Vis. Pattern Recognition., Jun. 2016, pp. 30543062.