ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal
Regular Issue, Vol. 8 N. 2 (2019), 5-18
eISSN: 2255-2863
DOI: http://dx.doi.org/10.14201/ADCAIJ201982518

Learning Representations from Spatio-Temporal Distance Maps for 3D Action Recognition with Convolutional Neural Networks

M. Naveenkumar and S. Domnic

Department of Computer Applications, National Institute of Technology, Tiruchirappalli, Tamilnadu, India
mnaveenmtech@gmail.com, domnic@nitt.edu

KEYWORD

Spatio-Temporal Distance Maps; CNN; Skeleton maps

ABSTRACT

This paper addresses the action recognition problem using skeleton data. In this work, a novel method is proposed, which employs five Distance Maps (DM), named as Spatio-Temporal Distance Maps (ST-DMs), to capture the spatio-temporal information from skeleton data for 3D action recognition. Among five DMs, four DMs capture the pose dynamics within a frame in the spatial domain and one DM captures the variations between consecutive frames along the action sequence in the temporal domain. All DMs are encoded into texture images, and Convolutional Neural Network is employed to learn informative features from these texture images for action classification task. Also, a statistical based normalization method is introduced in this proposed method to deal with variable heights of subjects. The efficacy of the proposed method is evaluated on two datasets: UTD MHAD and NTU RGB+D, by achieving recognition accuracies 91.63% and 80.36% respectively.

1. Introduction

Action recognition is increasingly becoming an active research field because it has many real-time applications such as content-based video search, automatic video indexing, assisted living, robotics, human-computer interaction and smart video surveillance. The traditional research examined action recognition from RGB videos. Unfortunately, RGB data is sensitive to many factors: (i) illumination changes (ii) background clutter (iii) occlusions etc. (Poppe, 2010). Besides, RGB sensors cannot capture the motion in 3D. The depth sensor, which captures 3D information, overcomes the above factors. In the seminal work, Shatton *et al,.* (Shotton *et al.*, 2011) proposed a methodology to extract the skeleton joints from depth data in real time. It has generated a renewed interest in the research community to use of skeleton data for action recognition. Moreover, Skeleton map is invariant to viewpoints or appearances compared with depth map, thus suffering less intra-class variance (Zhang *et al.*, 2017).

Traditional methods for skeleton based action recognition have been focused on designing handcrafted features (Xia *et al.*, 2012) (Wang *et al.*, 2014) (Vemulapalli *et al.*, 2014). These methods can be classified into three categories: (i) joint based, (ii) mined joint based and (iii) dynamics based methods. The joint based methods

capture the correlation between the joints for action recognition task where as the mined joint based methods identify which body parts are discriminative for action recognition. In dynamics based methods, the temporal dynamics are captured from the 3D trajectories of the skeleton action sequence for action classification task. The recent survey for skeleton based action recognition can be found in paper (Presti and La Cascia, 2016).

In recent years, deep learning methods have achieved superior results than the traditional methods in the field of computer vision. Two types of deep learning models received much attention for action recognition task. They are (i) Recurrent Neural Networks (RNNs) and (ii) Convolutional Neural Networks (CNNs). Different RNN based structures such as hierarchical RNN (Du *et al.*, 2015b), spatio-temporal long short-term memory (LSTM) (Liu *et al.*, 2016), part-aware LSTM (Shahroudy *et al.*, 2016), spatio-temporal attention based RNN (Song *et al.*, 2017) and two stream RNN (Wang and Wang, 2017) have been proposed to learn discriminative features from skeleton data for action recognition. The above methods concatenate the coordinates of joints at each time step before applying RNN based methods. Thus, spatial geometrical relations among different joints are lost in this pre-processing stage. In contrast, CNNs directly extract information from texture images which are encoded from skeleton sequences. Different CNN based methods are proposed for skeleton based action recognition. In (Du *et al.*, 2015a), the texture images are constructed by mapping 3D (x,y,z) coordinates of joints to red, blue and green values respectively. In the work (Wang *et al.*, 2016), the joint trajectories are extracted and encoded into color images by using hue, saturation, and values. In (Li *et al.*, 2017), four Joint Distance Maps (JDMs) are constructed from the pairwise distances of skeleton joints. Then, JDMs are encoded into texture images. In the above works (Du *et al.*, 2015a) (Wang *et al.*, 2016) (Li *et al.*, 2017), the action sequences are encoded into texture images. Hence, the action recognition problem is converted into image classification problem. CNN is employed for action classification task. To the best of our knowledge, there is no deep learning method which can capture effective spatio-temporal information from skeleton data.

In this paper, we address the action recognition task using skeleton data. The contributions of this paper are summarized as follows.

1. Since an action is a spatio-temporal event, both spatial and temporal features are needed for action recognition task. In this context, we employ five Distance Maps (DMs) in spatial domain and temporal domain to capture the pose and temporal variations.
2. A statistical based normalization method is introduced to deal with variable heights of subjects.
3. Five CNNs are trained on the proposed ST-DMs and multiplication fusion is employed on the resultant score vectors to assign the action label for an unknown action sequence.
4. Our method dramatically outperforms the recent state of the art methods. The experiments have been conducted in two settings: single view and cross view.

The rest of the paper is organized as follows. Section 2 gives a brief overview of related works in the literature and the proposed method is presented in Section 3. The experimental results are described in Section 4. The conclusions are drawn in the final section.

2. Related Work

In this section, we briefly review the literature related to our work i.e. skeleton based approaches for action recognition. Existing literature about the skeleton based action recognition can be classified into two types: Handcrafed feature based methods (Traditional methods), Deep learning based methods.

2.1. Handcrafed feature based methods

These methods can be classified into three categories: (i) joint based, (ii) mined joint based and (iii) dynamics based methods. The joint based methods capture the correlation between the joints for action recognition task. For example, Müller *et al.* (Müller *et al.*, 2005) introduce a class of boolean features expressing geometric relations between body points of a pose. Kerola *et al.* (Kerola *et al.*, 2014) has employed a graph based approach for

action recognition. In this work, an action sequence is represented as spatio-temporal graph and edge weights are calculated based on the pair wise distances. Hussein *et al.* (Hussein *et al.*, 2013) has used the covariance matrix of skeleton sequence as feature descriptor for action recognition and the multiple covariance matrices are generated to capture the relation between joint movement and time. Mined joint based methods try to learn which body parts are discriminative for action recognition. In paper (Climent-Pérez *et al.*, 2012), a genetic algorithm is proposed to identify informative joints for a specific class. Then, K-means algorithm is employed for action recognition task. In work (Wang *et al.*, 2013), skeleton joints are grouped into five body parts. Then, data-mining techniques are applied to obtain distinctive poses in spatial domain and temporal domain. Support Vector Machine is employed for action classification. In dynamics based methods, the temporal dynamics are captured from the 3D trajectories of the skeleton action sequence for action classification task. Dynamic based approaches uses linear dynamical systems (LDS) (Chaudhry *et al.*, 2013) or hidden Markov models (HMM) or mixed approaches (Presti *et al.*, 2014) for modeling of actions.

2.2. Deep learning based methods

There are two types of deep learning models received much attention for action recognition task. They are (i) Recurrent Neural Networks (RNNs) and (ii) Convolutional Neural Networks (CNNs).

Different RNN based structures such as hierarchical RNN (Du *et al.*, 2015b), spatio-temporal long short-term memory (LSTM) (Liu *et al.*, 2016), part-aware LSTM (Shahroudy *et al.*, 2016), spatio-temporal attention based RNN (Song *et al.*, 2017) and two stream RNN (Wang and Wang, 2017) have been proposed to learn discriminative features from skeleton data for action recognition. The above methods concatenate the coordinates of joints at each time step before applying RNN based methods. Thus, spatial geometrical relations among different joints are lost in this pre-processing stage.

In contrast, CNNs directly extract information from texture images which are encoded from skeleton sequences. Different CNN based methods are proposed for skeleton based action recognition. In (Du *et al.*, 2015a), the 3D (x,y,z) coordinates of joints in skeleton action sequence are mapped into red, blue and green values respectively. Hence, the skeleton action sequence is converted into the color image, and the action recognition problem is converted to image classification problem, and then the powerful image classifiers such as Convolution Neural Networks (CNN) are employed for action recognition. Due to the small size of the converted color images, it is difficult to fine-tune the existing CNN. In the work (Wang *et al.*, 2016), the joint trajectories are extracted and encoded into color images by using hue, saturation, and values. The encoded trajectories are used in CNN as inputs for action classification. The joint trajectories capture temporal variations, but fail to extract structural dynamics within a frame in the action sequence. In this context, Li *et al.* (Li *et al.*, 2017) proposed four Joint Distance Maps based on the pairwise distances of skeleton joints within the frame and the CNN was adopted for action recognition. From the above works, we have observed that the success of an action recognition task is dependent on how effectively model captures the spatial and temporal dynamics from action sequences. This paper employs five Distance Maps (DMs), referred to as Spatio Temporal Distance Maps (ST-DMs), to deal with spatial and temporal variations for action recognition task whereas the paper (Li *et al.*, 2017) uses spatial features and it does not consider temporal variations for action recognition.

3. Proposed Method

In this section, we first introduce the some necessary backgrounds. Then, we present two phases, Feature Extraction and Action Representation, of the proposed method. Finally, the action classification phase is discussed.

3.1. Preliminaries

3.1.1. *Basic CNN components*

The CNN is a popular deep learning architecture. LeCun *et al.* (LeCun *et al.*, 1990) introduced the modern framework of the CNN, named as LeNet-5, in their seminal work. LeNet-5 has been trained with backpropagation

algorithm for Handwritten digit recognition problem. Due to the lack of the large training data and computing resources at that time, Lenet-5 doesn't give good performance on complex problems such as image and video classification. Numerous variants of the CNN architectures have been proposed in the literature since 2006. The basic components of all these variants are very similar. In general, the basic components of CNN are: Convolution layer, Activation function, Pooling layer, Loss Function.

The *convolution layer* is composed of several kernels. The weights in kernels are learned from the training data. The output of a convolution layer is a feature map and the number of feature maps generated are dependent on the number of kernels used in convolution operation. Let F is a convolved feature map and feature value at location (p, q) in *ith* feature map of *jth* layer is calculated as in equation (1). Typically, several convolution layers are stacked in CNN. In general, first convolution layer is designed to learn low level features and last convolution layer is for high level abstract features.

$$F^j_{p,q,i} = W^{j^T}_i X^j_{p,q} + b^j_i \tag{1}$$

where W^j_i, b^j_i are the wight vector and bias terms of *ith* filter of *jth* layer. $X^j_{p,q}$ is the input patch of *jth* layer.

Typical *activation functions* are sigmoid, tanh and Rectified Layer (ReLu). Among these, Relu is frequently used to reduce the training time of CNN. It makes the negative values to zero in feature map F. Let A is activation map and the value at location (p, q) in *ith* activation map of *jth* layer is calculated as in equation (2). The function *max*() returns the maximum value in the given list of parameters. (Note that both two parameters are scalars in function *max*()).

$$A^j_{p,q,i} = max(0, F^j_{p,q,i}) \tag{2}$$

Pooling is important concept in CNN. The *pooling layer* gives shift-invariance and lowers the computational burden by reducing the resolution of the feature maps. It is usually placed between two convolution layers. Typically, there are two types of pooling operations: (i) average pooling; (ii) max pooling. Let pool(.) is the pooling function, it is denoted mathematically as in equation 3.

$$A^j_{p,q,i} = pool(A^j_{r,s,i}), \forall (r,s) \in N_{p,q} \tag{3}$$

where *Np*,*q* is neighbourhood region of location (p, q). After stacking of convolution and pooling layers, one or more fully-connected (FC) layers are placed in CNN. In FC layer, all neurons in previous layer are connected to each neuron in the current layer. Hence, it helps for high level reasoning.

It is essential to choose appropriate *loss function* to train CNN for a specific task. Typical loss functions for CNN are: (i) hinge loss; (ii) softmax loss. Hinge loss function is used to train a SVM classifier. The hinge loss for multi-class svm is calculated as stated in equation 4. Let **W** is the weight vector of a classifier and N is the number of instances in the training set. If training set contains M classes, the hinge loss *Lh* is defined as stated in equation (4).

$$L_h = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{M} [max(0, 1 - \delta(y^i, j) W^T A_i] \tag{4}$$

where A_i is the input instance and y^i is the corresponding label for X_i. $\delta(y^i, j) = 1$ if $y^i = j$, other wise $\delta(y^i, j) = -1$. Softmax loss is frequently used to train a CNN. Let L_s denotes the softmax loss of a classifier, it is defined as the combination of logistic and softmax functions as stated in equation (5).

$$L_s = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{M} [y^i = j] log c^i_j \tag{5}$$

where $c_j^i = \frac{e^{z_j^i}}{\sum_{k=1}^{M} e^{z_k^i}}$ and z_j^i are the activations of a densely connected layer. Hence $z_j^i = W_j^T A^i + b_j$

3.1.2. *Regularization*

Training a CNN is a optimization problem. Overfitting is the main problem in training a CNN. It can be solved by introducing the regularization parameters. Regularization adds the additional terms to the objective function.

Mainly, two typical techniques are used in CNN: (i) *lc* norm regularization (ii) dropout. If the soft max loss function of a classifier is $L_s(\theta, x, y)$, the *lc* norm regularization loss $E(\theta, x, y)$ is defined as given in equation (6).

$$E(\theta, c, y) = L_s(\theta, x, y) + \lambda \sum \|\theta_j\|_c^c \tag{6}$$

if $c \geq 1$, the *lc* norm regularization is convex and the optimization problem is easier. Otherwise, the *lc* norm is non convex. Dropout is the another regularization method, introduced by Hinton *et al.* (Hinton *et al.*, 2012). In work (Hinton *et al.*, 2012), dropout is applied to fully connected layers to make learning more accurate even absence of some information.

3.1.3. *Optimization*

To optimize the performance for CNN, the key techniques are data augmentation and weight initialization. CNN is a deep neural network and millions of parameters have to be learned from the training data. Hence, a large quantity of training data is needed to set optimal values for these learn-able parameters. For some specific problems, acquisition of data is very difficult. In this context, data augmentation is the solution for optimizing performance of a CNN. The well known augmentation methods are mirroring (Yang and Patras, 2015), shifting (Salamon and Bello, 2017), sampling (Russakovsky *et al.*, 2015a) and rotating (Xie and Tu, 2015). In works (Xie *et al.*, 2015) (Xu *et al.*, 2015), additional augmentation methods are introduced to optimize the learning in fine-grained recognition tasks. The loss function of CNN is non-convex and very difficult to train (Choromanska *et al.*, 2015). To achieve fast convergence in training, weight initialization plays a vital role. In work (Russakovsky *et al.*, 2015b), the weights are initialized with zero mean gaussian distribution and standard deviation is 0.01.

3.1.4. *Stochastic gradient descent*

In general, the backpropagation algorithm is used to train a CNN. It uses gradient descent to update the parameters. Many variants of gradient descent methods are available in the literature (Qian, 1999). The parameters are updated using standard gradient descent algorithm as $\theta t+1 = \theta t - \eta \nabla \theta E[L(\theta t)]$. $L(\theta t)$ is the objective function and $E[L(\theta t)]$ is the expectation of $L(\theta t)$ over the full training set. η is the learning rate. In stochastic gradient descent, $E[L(\theta t)]$ is calculated on each randomly picked input instance (Xt, yt) as stated in equation (7)

$$\theta_{t+1} = \theta_t - \eta_t \nabla_\theta L(\theta_t; X^t, y^t) \tag{7}$$

In some contexts, the $E[L(\theta t)]$ is calculated over mini batch for stable convergence of CNN. To choose optimal value for learning rate η, Schaul *et al.* (Schaul *et al.*, 2013) proposed the learning rate schedules to adjust the η during training. In paper (Qian, 1999), momentum is introduced to update the parameters using velocity vector as stated in equations (8) and (9)

$$v_{t+1} = \gamma v_t - \eta_t \nabla_\theta L(\theta_t; x^t, y^t) \tag{8}$$

Table 1: Layers Description of AlexNet

<table>
<tr><th>S.No</th><th>Name of Layer</th><th>N.º of Layers</th><th>Description</th></tr>
<tr><td>1</td><td>Input</td><td>1</td><td>It takes the image as input of size 227 × 227 × 3</td></tr>
<tr><td>2</td><td>Convolutional</td><td>5</td><td>The output of this layer is called feature map. These layers play a vital role in feature engineering. The details of the five convolutional layers are as follows.
<table>
<tr><th>layer</th><th>kernel size</th><th>n.º of kernels</th><th>stride</th><th>padding</th></tr>
<tr><td>conv1</td><td>11 × 11</td><td>96</td><td>4</td><td>0</td></tr>
<tr><td>conv2</td><td>5 × 5</td><td>256</td><td>1</td><td>2</td></tr>
<tr><td>conv3</td><td>3 × 3</td><td>384</td><td>1</td><td>1</td></tr>
<tr><td>conv4</td><td>3 × 3</td><td>384</td><td>1</td><td>1</td></tr>
<tr><td>conv5</td><td>3 × 3</td><td>256</td><td>1</td><td>1</td></tr>
</table></td></tr>
<tr><td>3</td><td>ReLu</td><td>7</td><td>Rectified Layer unit (ReLu) is used to make negative values to zero. This layer helps to reduce the training time of CNN</td></tr>
<tr><td>4</td><td>Normalization</td><td>2</td><td>Local Response Normalization helps generalization and also to reduce the error rates.</td></tr>
<tr><td>5</td><td>Max Pooling</td><td>3</td><td>This layer reduces the number of parameters by subsampling the output of Convolutional layer. The features produced by max pooling layer are invariant to scale and orientation.</td></tr>
<tr><td>6</td><td>Fully Connected</td><td>3</td><td>It has connections to all activations in the previous layer as in regular neural networks.</td></tr>
<tr><td>7</td><td>Drop out</td><td>2</td><td>This layer helps to reduce the overfitting problem in training a CNN</td></tr>
<tr><td>8</td><td>Soft max</td><td>1</td><td>This is generalization of Logistic regression. This is used for multi-class classification</td></tr>
<tr><td>9</td><td>Output</td><td>1</td><td>The output layer contains labels of the classes</td></tr>
<tr><td>Total layers</td><td></td><td></td><td>25</td></tr>
</table>

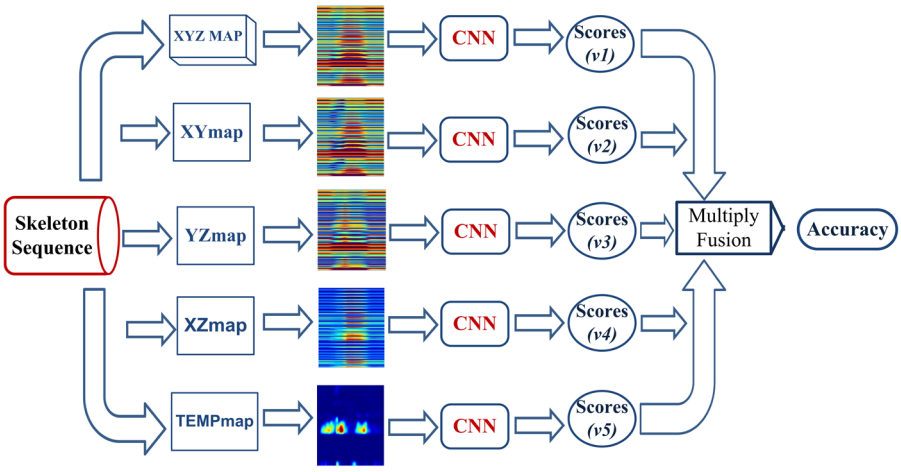

Figure 1: Frame work of the proposed method

$$\theta_{t+1} = \theta_t + v_t + 1 \tag{9}$$

where γ is momentum term, $vt+1$ is the current velocity vector. In general, the training process using SGD don't converge. The training process should stop at the point where the performance stops increasing. The two popular techniques to stop training process are (i) early stopping criteria (ii) choosing fixed number of epochs.

3.2. Feature Extraction

The proposed method contains four main parts as illustrated in Fig. 1. They are the construction of ST-DMs from input skeleton data, encoding of ST-DMs to texture color images, training the CNN model and the fusion of scores. The four Distance Maps in the spatial domain are referred to *XY Zmap*, *XY map*, *Y Zmap* and *XZmap*. To deal with actions that contains human to human interaction, every action is assumed to be performed by two subjects (main subject and auxiliary subject). If an action sequence contains only one subject, a shadow subject is copied from main subject (Ding *et al.*, 2017). Suppose an action sequence *A* contains *M* skeleton frames and each skeleton frame contains *2N* joints, where *N* joints are related to main subject and other *N* joints are for auxiliary subject. A = *{F*1, ……*FM}*, where *F* represents a frame and $F_i = \{J_1^i, ...J_{2N}^i\}$. The J_j^i represents the 3D coordinates (*x*, *y*, *z*) of *jth* joint of *ith* frame. The 2D orthogonal projections of 3D Joint *J* are referred as *Jxy*, *Jyz* and *Jxz*. The four spatial DMs are constructed as stated in equations (10) to (13). The fifth feature map, named as *TEMP map*, is calculated in the time domain as given in equation (14).

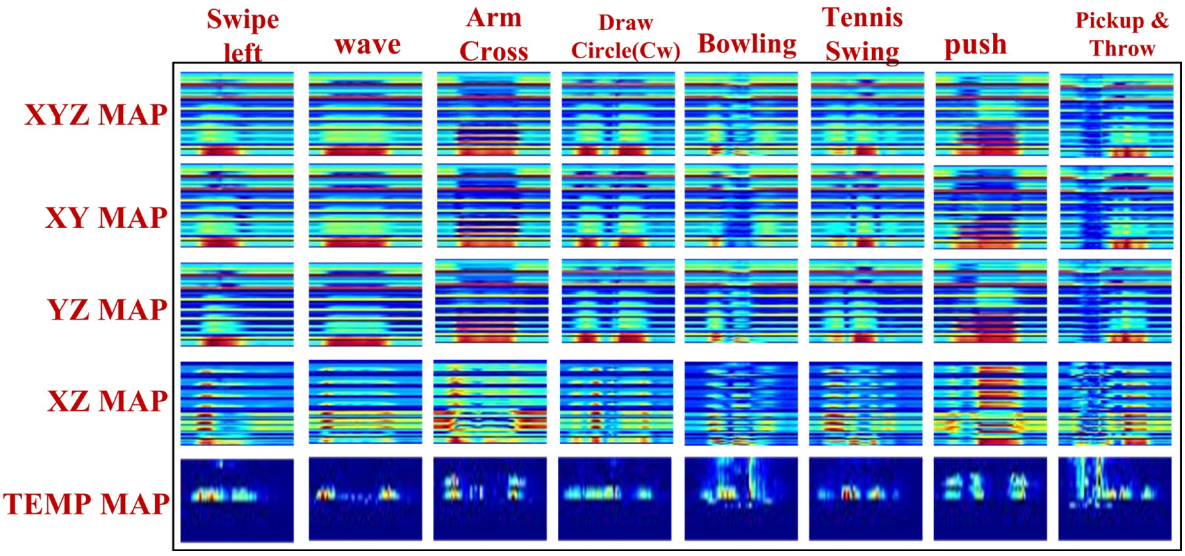

Figure 2: Sample color images of Spatio-Temporal Maps generated by the proposed method on UTD MHAD dataset.

$$XYZmap = \{dist_{3D}(J_i^f, J_j^f)|i,j = 1..,2N; i \neq j; f = 1..,M\} \quad (10)$$

$$XYmap = \{dist_{2D}(Jxy_i^f, Jxy_j^f)|i,j = 1..,2N; i \neq j; f = 1..,M\} \quad (11)$$

$$YZmap = \{dist_{2D}(Jyz_i^f, Jyz_j^f)|i,j = 1..,2N; i \neq j; f = 1..,M\} \quad (12)$$

$$XZmap = \{dist_{2D}(Jxz_i^f, Jxz_j^f)|i,j = 1..,2N; i \neq j; f = 1..,M\} \quad (13)$$

$$TEMPmap = \{dist_{3D}(J_i^f, J_i^{f+1})|i = 1..,2N; f = 1..,M-1\} \quad (14)$$

Where *f* represents the frame number, *J* refers (x,y,z) coordinates of joint, *Jxy* refers (x,y) coordinates of the joint and so on. *dist*3*D* () is the Euclidean distance of two points in Euclidean 3-space where as *dist*2*D* () is the Euclidean distance of two points in Euclidean 2-space. suppose, $r = (r1, r2, \ldots, rn)$ and $s = (s1, s2\ldots, sn)$ are two points in Eucledean n-space, the *distnD* () is calculated as:

$$dist_{nD}(r,s) = \sqrt{(s_1 - r_1)^2 + (s_2 - r_2)^2 + ...(s_n - r_n)^2} \quad (15)$$

Table 2: Recognition accuracy of Five DMs and their Multiplication Fusion on UTD MHAD and NTU RGB+D datasets

Feature	UTD MHAD CS (%)	NTU RGB+D	
		CS (%)	CV (%)
XYZmap	81.63	73.26	81.36
XYmap	80.93	69.04	73.63
YZmap	77.67	69.50	69.23
XZmap	71.86	65.19	72.10
TEMPmap	63.95	61.87	65.90
Mul(ALL)	91.16	80.36	86.94

For an Action *A*, when the distances calculated for a single frame are arranged in a single column, four matrices are generated for four spatial features respectively. Each matrix of size $(2N^2 - N) \times M$. In the context of the *TEMP map* feature, the distances calculated for two consecutive frames are arranged in a single column. As a result, a matrix is formed of size $2N \times (M-1)$. Since *M* is vary from one action sequence to another, feature matrices do not contain fixed number of columns for all the sequences in the training set. To avoid this problem and produce matrices with fixed number of columns *M1*, bi-linear interpolation is used to resize the spatial map from $(2N^2 - N) \times M$ to $(2N_2 - N) \times M1$ and temporal map from $2N \times (M-1)$ to $2N \times M1$. Since subjects can have different scale in their height and width in the skeleton data, the feature values extracted from skeleton data need to be normalized to fit the desired range [0-1]. Hence, a normalization equation is proposed in this work as given in equation (16). To keep the values in [0-255] range, the normalized matrix is multiplied by 255 as in equation (17). The resultant matrices looks like gray images with 256 intensity levels. In order to adopt the AlexNet (Krizhevsky *et al.*, 2012) for classification, the gray images are encoded into color images by adopting jet color bar (Li *et al.*, 2017). The color images of ST-DMs for different actions in the UTD MHAD dataset is

shown in Fig. 2. As a result, the action recognition problem is converted into an image classification problem and CNN is fine-tuned for action classification in this paper.

$$NormalizedM = \frac{M - min(M)}{max(M) - min(M)} \tag{16}$$

$$GrayImage = NormalizedM * 255 \tag{17}$$

Where M is a feature matrix to be normalized, min(M) is minimum value in M and max(M) is the maximum value in M.

3.3. Action Classification

The popular CNN architecture, AlexNet (Krizhevsky *et al.*, 2012), is fine-tuned for action classification in this paper. Since the input layer of AlexNet takes 227 × 227 × 3 images, all encoded color images are resized using bicubic interpolation (Keys, 1981). The layers used in Alexnet are Convolution, Relu, Max pooling, Fully connected, drop out and soft max, which are discussed in Section 3.1. The details of different layers of AlexNet used for experiments are described in Table 1. Five CNNs are trained using color images related to five features as illustrated in Fig. 1. This letter adopts multiplication fusion (Li *et al.*, 2017) to produce the final test result. For an action sequence A in testing set, if $v1$, $v2$, $v3$, $v4$, and $v5$ are the vectors related to scores of five CNNs, the label is assigned for action A as given in equation (18). AlexNet is trained using the solver "Stochastic Gradient Decent with Momentum" with momentum value 0.9. In general, the initial learning rates are small values for fine tuning the network. The learning rate is initially set to 0.0001 for fine tuning and the mini-batch size is set to 128 for all the experiments in this paper.

Table 3: Recognition accuracy results on UTD MHAD Dataset

	Accuracy(%)
ElC-KSVD, 2014 (Zhou *et al.*, 2014)	76.19
Kinect and Inertial, 2015 (Chen *et al.*, 2015)	79.10
Joint trajectory maps, 2016 (Wang *et al.*, 2016)	85.81
Joint Distance Maps, 2017 (Li *et al.*, 2017)	88.10
Our method	**91.16**

$$Label\ of\ test\ instance\ A = Find\_Max\_index(v1 \lozenge v2 \lozenge v3 \lozenge v4 \lozenge v5) \tag{18}$$

where ◊ represents the element wise multiplication and *Find_Max_index*(.) is the function to find the index (class label) of maximum value.

4. Experiments

The efficacy of the proposed method is evaluated on two benchmark datasets for action recognition. All the experiments are carried out using a workstation with 16 GB RAM, NVIDIA Quadro K2200 graphics card. The number of epochs are set to 100 for all the experiments.

4.1. Datasets

4.1.1. *UTD MHAD dataset*

UTD MHAD dataset (Chen *et al.*, 2015) uses a kinect camera and a wearable inertial sensor to capture depth, skeleton joints, RGB data and inertial sensor data. The skeleton is represented by using 20 joints. It contains 27 actions, performed by 8 subjects with each one performs an action four times. As a result, 864 (27x8x4 = 864) data sequences are generated. After removing 3 corrupted sequences, the total data sequences are 861. For fair comparison, we follow the cross subject protocol proposed in the paper (Chen *et al.*, 2015). For cross subject evaluation, the odd subjects (1,3,5,7) are used for training and even subjects (2,4,6,8) are used for testing. As a result, there are 431 action sequences in training set and 430 in testing set.

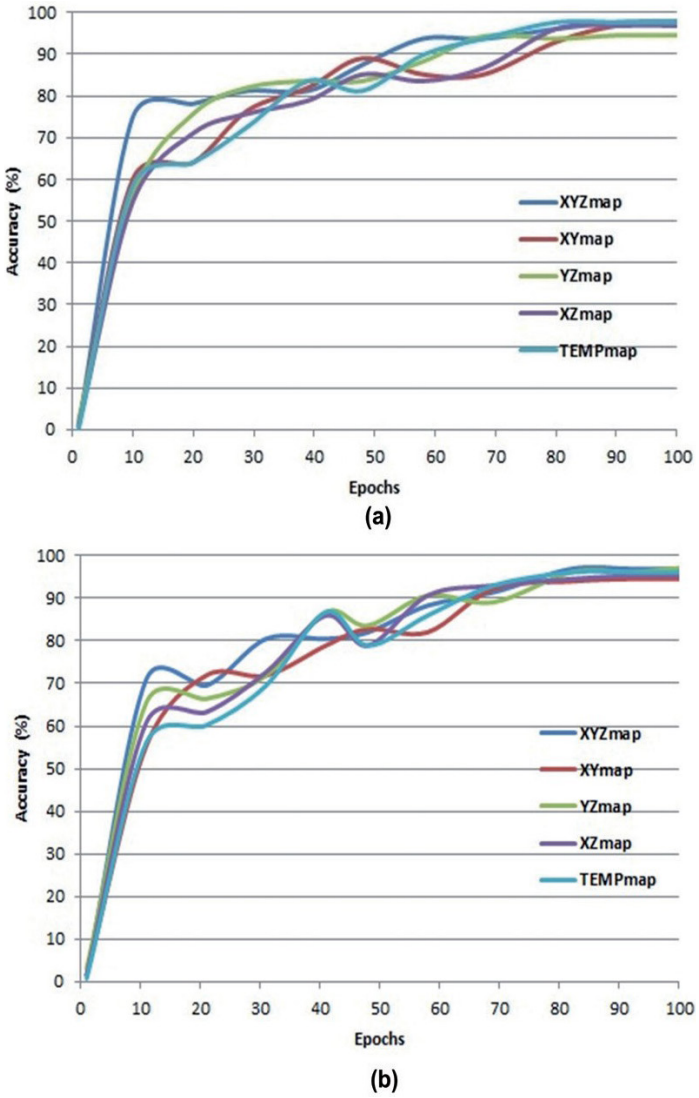

Figure 3: Accuracies on Training set with number of epochs on the NTU RGB+D dataset for both cross subject and cross view settings. (a) Cross Subject (b) Cross View

Table 4: Recognition accuracy results on NTU RGB+D Dataset

	cross-subject(%)	cross-view (%)
Dynamic Skeletons, 2013 (Ohn-Bar and Trivedi, 2013)	60.20	65.20
Lie Group, 2014 (Vemulapalli *et al.*, 2014)	50.10	52.80
Skeletal Quads (Evangelidis *et al.*, 2014)	38.60	41.40
LieNet, 2017 (Huang *et al.*, 2017)	61.30	67.00
HBRNN, 2015 (Du *et al.*, 2015b)	59.10	64.00
Deep LSTM, 2016 (Shahroudy *et al.*, 2016)	60.70	67.30
Part-aware LSTM, 2016 (Shahroudy *et al.*, 2016)	62.90	70.30
ST-LSTM, 2016 (Liu *et al.*, 2016)	65.20	76.10
ST-LSTM + Trust Gate, 2016 (Liu *et al.*, 2016)	69.20	77.70
Ensemble LSTM, 2017 (Lee *et al.*, 2017)	74.60	81.25
STA-LSTM,2017 (Song *et al.*, 2017)	73.40	81.20
GCA-LSTM, 2017 (Liu *et al.*, 2017)	74.40	82.80
Two-stream RNN, 2017 (Wang and Wang, 2017)	71.30	79.50
Multi-task RNN, 2018 (Wang and Wang, 2018)	-	82.60
JTM, 2016 (Wang *et al.*, 2016)	73.40	75.20
JDM, 2017 (Li *et al.*, 2017)	76.20	82.30
CNN+LSTM, 2018 (Núñez *et al.*, 2018)	67.50	76.21
Our method	**80.36**	**86.94**

4.1.2. *NTU RGB+D dataset*

To the best of our knowledge, NTU RGB+D dataset (Shahroudy *et al.*, 2016) is largest action recognition dataset and it uses three kinect v2 sensors to capture the depth and skeleton information. The skeleton is represented using 25 joints. It contains 60 action classes performed by 40 subjects. There are 56,880 action sequences and more than 4 million frames in this dataset. After removing missing skeletons, the dataset contains 56,578 action sequences. This dataset is challenging in two aspects: (i) large intra class variations; (ii) view point variations. Due to large scale of this dataset, it is highly suitable for deep learning. We follow the two experimental protocols, namely cross subject and cross view protocols, proposed in paper (Shahroudy *et al.*, 2016). In cross subject test, the actions pertaining to the subjects 1, 2, 4, 5, 8, 9, 13, 14, 15, 16, 17, 18, 19, 25, 27, 28, 31, 34, 35, 38 are considered for training and remaining are for testing. As a result, the training set contains 40,091 samples where as testing set consisting of 16,487 action sequences. In cross view evaluation, the samples captured using camera 2 and 3 for training and camera 1 samples for testing.

4.2. Results of Action Recognition

Table 2 reports the results of individual ST-DMs and their multiplication fusion. Here, *Mul*(*ALL*) means the multiplication fusion of all five Distance Maps (DM). When comparing the individual DMs, *XY Zmap* outperforms the other DMs on both the datasets. The multiplication fusion of all DMs significantly achieves good performance than other recent works in the literature. From these results, it is concluded that all five DMs have

their significance to achieve recognition accuracy. Table 3 reports the performance of the proposed method with other state of the art methods on UTD MHAD dataset. It is noted from Table 3 that the works (Wang *et al.*, 2016) (Li *et al.*, 2017) achieved 85.81% and 88.10% recognition accuracies respectively, which is not better than that of the proposed method, which achieves the accuracy of 91.16%. The reason is that the proposed method uses both spatial and temporal dynamics where as the works (Wang *et al.*, 2016) (Li *et al.*, 2017) used either spatial or temporal dynamics for action recognition.

Table 4 shows the results on the NTU RGB+D dataset. For this dataset, we compare our results with traditional methods (Ohn-Bar and Trivedi, 2013) (Vemulapalli *et al.*, 2014) (Evangelidis *et al.*, 2014) (Huang *et al.*, 2017), RNN based methods (Du *et al.*, 2015b) (Shahroudy *et al.*, 2016) (Shahroudy *et al.*, 2016) (Song *et al.*, 2017) (Liu *et al.*, 2017) (Wang and Wang, 2017) and CNN based methods (Wang *et al.*, 2016) (Li *et al.*, 2017) (Núñez *et al.*, 2018). It should be noted that the work (Núñez *et al.*, 2018) uses both CNN and LSTM for action recognition. The empirical results show that our method achieves 80.36% and 86.94% accuracies for cross subject and cross view settings respectively, which are higher than the recent existing works. The reason for the achievement of better results by our method is that it uses both spatial and temporal dynamics for action recognition.

To compare and analyze the convergence rates of five ST-DMs employed in this paper, we plot the accuracies on the training data during training phase. The results of five ST-DMs on NTU RGB+D dataset are shown in Figure 3. From this Figure 3, it is observed that the all curves, related to five ST-DMs, converge their maximum values. For Cross subject settings (see Figure 3(a)), all ST-DMs except *XY map*, the training accuracy rate is gradually increased and stabilized after 80*th* epoch, but, for *XY map*, the training accuracy rate is stabilized after 90*th* epoch. For Cross view settings (see Figure 3(b)), for all ST-DMs, the training accuracy rate is stabilized after 80*th* epoch.

5. Conclusion

This paper presented a methodology for action recognition using skeleton data. It employed two types (spatial, temporal) of skeleton features for action recognition. Four features are extracted in the spatial domain to capture the structure (pose) dynamics within a frame, and one feature is extracted in temporal domain to capture the temporal dynamics between consecutive frames. All extracted features are converted into 2D color images. Hence, the 3D action recognition problem is converted into a 2D image classification problem. This representation allows us to fine tune the Convolutional Neural Network without training from scratch. The proposed method is evaluated on two benchmark datasets: UTD MHAD and NTU RGB+D datasets, by achieving recognition accuracies 91.16% and 80.36% respectively.

6. References

Chaudhry, R., Ofli, F., Kurillo, G., Bajcsy, R., and Vidal, R., 2013. Bio-inspired dynamic 3d discriminative skeletal features for human action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 471-478.

Chen, C., Jafari, R., and Kehtarnavaz, N., 2015. Utd-mhad: A multimodal dataset for human action recognition utilizing a depth camera and a wearable inertial sensor. In *Image Processing (ICIP), 2015 IEEE International Conference on*, pp. 168-172. IEEE.

Choromanska, A., Henaff, M., Mathieu, M., Arous, G. B., and LeCun, Y., 2015. The loss surfaces of multilayer networks. In *Artificial Intelligence and Statistics*, pp. 192-204.

Climent-Pérez, P., Chaaraoui, A. A., Padilla-López, J. R., and Flórez-Revuelta, F., 2012. Optimal joint selection for skeletal data from RGB-D devices using a genetic algorithm. In *Mexican International Conference on Artificial Intelligence*, pp. 163-174. Springer.

Ding, Z., Wang, P., Ogunbona, P. O., and Li, W., 2017. Investigation of different skeleton features for CNN-based 3D action recognition. In *Multimedia & Expo Workshops (ICMEW), 2017 IEEE International Conference on*, pp. 617-622. IEEE.

Du, Y., Fu, Y., and Wang, L., 2015a. Skeleton based action recognition with convolutional neural network. In *Pattern Recognition (ACPR), 2015 3rd IAPR Asian Conference on*, pp. 579-583. IEEE.

Du, Y., Wang, W., and Wang, L., 2015b. Hierarchical recurrent neural network for skeleton based action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1110-1118.

Evangelidis, G., Singh, G., and Horaud, R., 2014. Skeletal quads: Human action recognition using joint quadruples. In *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pp. 4513-4518. IEEE.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R., 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Huang, Z., Wan, C., Probst, T., and Van Gool, L., 2017. Deep learning on lie groups for skeleton-based action recognition. In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1243-1252. IEEE computer Society.

Hussein, M. E., Torki, M., Gowayyed, M. A., and El-Saban, M., 2013. Human Action Recognition Using a Temporal Hierarchy of Covariance Descriptors on 3D Joint Locations. In *IJCAI*, volume 13, pp. 2466-2472.

Kerola, T., Inoue, N., and Shinoda, K., 2014. Spectral graph skeletons for 3D action recognition. In *Asian Conference on Computer Vision*, pp. 417-432. Springer.

Keys, R., 1981. Cubic convolution interpolation for digital image processing. *IEEE transactions on acoustics, speech, and signal processing*, 29(6):1153-1160.

Krizhevsky, A., Sutskever, I., and Hinton, G. E., 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097-1105.

LeCun, Y., Boser, B. E., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. E., and Jackel, L. D., 1990. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pp. 396-404.

Lee, I., Kim, D., Kang, S., and Lee, S., 2017. Ensemble deep learning for skeleton-based action recognition using temporal sliding lstm networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 1012-1020. IEEE.

Li, C., Hou, Y., Wang, P., and Li, W., 2017. Joint distance maps based action recognition with convolutional neural networks. *IEEE Signal Processing Letters*, 24(5):624-628.

Liu, J., Shahroudy, A., Xu, D., and Wang, G., 2016. Spatio-temporal lstm with trust gates for 3d human action recognition. In *European Conference on Computer Vision*, pp. 816-833. Springer.

Liu, J., Wang, G., Hu, P., Duan, L.-Y., and Kot, A. C., 2017. Global context-aware attention lstm networks for 3d action recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 7, page 43.

Müller, M., Röder, T., and Clausen, M., 2005. Efficient content-based retrieval of motion capture data. In *ACM Transactions on Graphics (ToG)*, volume 24, pp. 677-685. ACM.

Núñez, J. C., Cabido, R., Pantrigo, J. J., Montemayor, A. S., and Vélez, J. F., 2018. Convolutional Neural Networks and Long Short-Term Memory for skeleton-based human activity and hand gesture recognition. *Pattern Recognition*, 76:80-94.

Ohn-Bar, E. and Trivedi, M., 2013. Joint angles similarities and HOG2 for action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 465-470.

Poppe, R., 2010. A survey on vision-based human action recognition. *Image and vision computing*, 28(6):976-990.

Presti, L. L. and La Cascia, M., 2016. 3D skeleton-based human action classification: A survey. *Pattern Recognition*, 53:130-147.

Presti, L. L., La Cascia, M., Sclaroff, S., and Camps, O., 2014. Gesture modeling by hanklet-based hidden markov model. In *Asian Conference on Computer Vision*, pp. 529-546. Springer.

Qian, N., 1999. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145-151. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. et al., 2015a. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211-252.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. et al., 2015b. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211-252.

Salamon, J. and Bello, J. P., 2017. Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal Processing Letters*, 24(3):279-283.

Schaul, T., Zhang, S., and LeCun, Y., 2013. No more pesky learning rates. In *International Conference on Machine Learning*, pp. 343-351.

Shahroudy, A., Liu, J., Ng, T.-T., and Wang, G., 2016. NTU RGB+ D: A large scale dataset for 3D human activity analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1010-1019.

Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., and Blake, A., 2011. Real-time human pose recognition in parts from single depth images. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pp. 1297-1304. Ieee.

Song, S., Lan, C., Xing, J., Zeng, W., and Liu, J., 2017. An End-to-End Spatio-Temporal Attention Model for Human Action Recognition from Skeleton Data. In *AAAI*, volume 1, pp. 4263-4270.

Vemulapalli, R., Arrate, F., and Chellappa, R., 2014. Human action recognition by representing 3d skeletons as points in a lie group. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 588-595.

Wang, C., Wang, Y., and Yuille, A. L., 2013. An approach to pose-based action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 915-922.

Wang, H. and Wang, L., 2017. Modeling temporal dynamics and spatial configurations of actions using two-stream recurrent neural networks. In *e Conference on Computer Vision and Pa ern Recognition (CVPR)*.

Wang, H. and Wang, L., 2018. Learning content and style: Joint action recognition and person identification from human skeletons. *Pattern Recognition*, 81:23-35.

Wang, P., Li, W., Ogunbona, P., Gao, Z., and Zhang, H., 2014. Mining mid-level features for action recognition based on effective skeleton representation. In *Digital lmage Computing: Techniques and Applications (DlCTA), 2014 International Conference on*, pp. 1-8. IEEE.

Wang, P., Li, Z., Hou, Y., and Li, W., 2016. Action recognition based on joint trajectory maps using convolutional neural networks. In *Proceedings of the 2016 ACM on Multimedia Conference*, pp. 102-106. ACM.

Xia, L., Chen, C.-C., and Aggarwal, J., 2012. View invariant human action recognition using histograms of 3d joints. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, pp. 20-27. IEEE.

Xie, S. and Tu, Z., 2015. Holistically-nested edge detection. In *Proceedings of the IEEE international conference on computer vision*, pp. 1395-1403.

Xie, S., Yang, T., Wang, X., and Lin, Y., 2015. Hyper-class augmented and regularized deep learning for fine-grained image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2645-2654.

Xu, Z., Huang, S., Zhang, Y., and Tao, D., 2015. Augmenting strong supervision using web data for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision*, pp. 2524-2532. Yang, H. and Patras, I., 2015. Mirror, mirror on the wall, tell me, is the error small? In *Proceedings of the IEEE*

Conference on Computer Vision and Pattern Recognition, pp. 4685-4693.

Zhang, S., Liu, X., and Xiao, J., 2017. On geometric features for skeleton-based action recognition using multilayer lstm networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 148-157. IEEE.

Zhou, L., Li, W., Zhang, Y., Ogunbona, P., Nguyen, D. T., and Zhang, H., 2014. Discriminative key pose extraction using extended lc-ksvd for action recognition. In *Digital lmage Computing: Techniques and Applications (DlCTA), 2014 International Conference on*, pp. 1-8. IEEE.