# Human Action Recognition Using A Multi-Modal Hybrid Deep Learning Model

Hany A. El-Ghaish[1]
hany.el-ghaish@ejust.edu.eg
Mohamed E. Hussein[1,2]
mohamed.e.hussein@ejust.edu.eg
Amin Shoukry[1,2]
amin.shoukry@ejust.edu.eg

[1] Computer Science and Engineering Department, Egypt-Japan University of Science and Technology,
New Borg Al-Arab, Alexandria, Egypt

[2] Computer and Systems Engineering Department, Alexandria University, Alexandria, Alexandria, Egypt

**Abstract**

Human action recognition is a challenging problem, especially in the presence of multiple actors and/or multiple scene views. In this paper, multi-modal integration and a hybrid deep learning architecture are deployed in a unified action recognition model. The model incorporates two main types of modalities: 3D skeletons and images, which together capture the two main aspects of an action, which are the body motion and part shape. Instead of a mere fusion of the two types of modalities, the proposed model integrates them by focusing on specific parts of the body, whose locations are known from the 3D skeleton data. The proposed model combines both Convolutional Neural Networks (CNN) and Long Short Term Memory (LSTM) deep learning architectures into a hybrid one. The model is called **MCL**, for (**M**)ulti-Modal (**C**)NN + (**L**)STM. MCL consists of two sub-models: **CL1D** and **CL2D** that simultaneously extract the spatial and temporal patterns for the two sought input modality types. Their decisions are combined to achieve better accuracy. In order to show the efficiency of the MCL model, its performance is evaluated on the large NTU-RGB+D dataset in two different evaluation scenarios: cross-subject and cross-view. The obtained recognition rates, **74.2 %** in cross-subject and **81.4%** in cross-view, are superior to the current state of the art results.

## 1 Introduction

Human action recognition is a vital branch of computer vision research because of its many application domains, such as computer gaming, surveillance, and robotic vision. The difficulty of the problem can be reduced if the body pose is available [27]. The human body pose can be captured using Motion Capture (MoCap) systems, which could be either camera-based or inertial sensor-based. Such systems are accurate but expensive. Nowadays, there is a great focus on body pose data collected using inexpensive depth sensors, such as the Kinect sensor, which provide both RGB and depth images in addition to the 3D body skeleton [9].

There are lots of techniques used for human action recognition from 3D skeleton data that are based on hand-engineered features. In [11], the covariance of 3D joint positions over the action sequence is used as a descriptor, and the Support Vector Machine (SVM) is used as a classifier. In [25], a view invariant action classification based on histogram of 3D skeleton joint locations (HOJ3D) is proposed. HOJ3Ds are projected using linear discriminant analysis (LDA) and clustered to visual words, which are used to train a Hidden Markov Model (HMM) to extract the temporal dependencies. The skeleton configuration and actions can be mapped to points and curves, respectively, in a lie group and then SVM can be deployed for classification, as in [20]. It is also common to combine multiple sources of features, such as in [6], where skeleton joint angles, joint angular velocities, and joint velocities, are combined into a 1D feature vector for each frame. Feature vectors for consecutive frames are then concatenated and fed to a Random Forests classifier. The drawback of all the aforementioned methods so far is that they need hand-engineering of features.

Recently, deep-learning techniques have become widely used in many application domains. Convolution Neural Networks (CNN) [13] is a neural network architecture that is typically used in deep networks for automatic feature extraction from raw data. The CNN1D, CNN2D, and CNN3D can be applied to extract features from 1D signals, 2D images, and 3D volumetric or video data, respectively. The extracted features are then fed to a classifier, which is typically a fully connected (dense) neural network layer(s). In [22], CNN3D is proposed for action recognition from the raw RGB videos to extract the spatio-temporal patterns of the performed actions. In [23], three CNN2D channels are initially trained on the ImageNet dataset, then fine tuned to be used in action recognition. The three CNN2D channels receive raw depth data for action classification to extract features from the weighted hierarchical depth motion maps sequence of an action. In [2], the motion represented by the optical flow and different poses from the tracks of human body parts over the action are aggregated to form the inputs to two parallel CNN2D. The output of each CNN2D is used to construct a frame descriptor (min and max values), and hand-crafted P-CNN features are formed from the concatenation of both the frame descriptors and video descriptor (differences between frame descriptors). P-CNN features are used to train a SVM classifier.

Another group of techniques relies on Recurrent Neural Networks (RNNs) or Long-Short Term Memory (LSTM) for capturing the temporal dependencies in the skeleton joint sequences for action recognition, such as [4] and [29]. The gesture recognition method in [14] uses a bidirectional LSTM with one hidden layer for backward and forward LSTMs.

CNN can be combined with LSTM to perform classification, bringing the best of the two worlds together: automatic features extraction by the CNN, and the capturing of the temporal dependencies in sequences of these features by the LSTM. Towards this end, we propose a (**M**)ulti-modal (**C**)NN + (simultaneously)STM (**MCL**) action recognition model. The model benefits from the presence of 3D skeleton data and image data to capture the two main aspects of an action: the motion and part shape. Moreover, it integrates the modalities by focusing only on the relevant body parts in the images instead of letting the network discover already known information on its own, which has the effect of reducing the training time and reducing the possibility of over-fitting. In order to evaluate the effectiveness of the proposed recognition model, we tested it on the NTU RGB+D [19] human action recognition dataset, which is a large dataset, suitable for the training needs of deep learning models.

The rest of the paper is organized as follows: the most recent works related to the proposed model are discussed in Section 2. Section 3 introduces the architecture of the proposed model (MCL). Experiments are presented and results are analyzed in Section 4. The conclusion of the paper and the future work are summarized in Section 5.

## 2 Related Work

The problem of recognizing human actions from 3D skeleton and/or RGB images is not new. However, the techniques for solving the problem are continuously evolving and improving. In the following, we summarize the most recent research related to the proposed model, which applies deep learning techniques, and show how our proposed approach is different.

In [1], CNN3D plus LSTM are used as a two-step action classification model. CNN3D receives the action video then extracts the spatio-temporal features by applying a set of 3D filters. The derived features are passed to a LSTM, which works as a classifier for the whole feature sequence. Unlike our proposed method, this method uses only a single modality, which is the video for the whole body. Our method fuses and integrates multiple modalities.

In [4], a Hierarchical Bidirectional RNN (HBRNN) is introduced. The whole body skeleton is divided into five parts: left arm, right arm, spine, left leg, and right leg. Parts are input to parallel branches of the model at the lowest level of the hierarchy. The outputs from pairs of branches are merged at the next level to represent bigger body parts, and so on until the whole body is represented. Finally, the output is classified using a fully-connected softmax layer. Similar to this approach, our proposed model divides the body into parts. However, our approach also fuses multiple modalities, not just 3D skeleton data.

In [29], an action recognition model based on a deep LSTM with fully connected layers is suggested. A regularization term is added to automatically learn the co-occurrences among body joints and a dropout technique is proposed to the LSTM gates to prevent over-fitting. In our method, deep LSTM is also used, but preceded with CNN1D and CNN2D to extract features from skeleton as well as part image data.

In [8], the ordered sequence of optical flow of action frames is computed and then fed to a LSTM in temporal order for robust classification of actions. This approach captures the body motion via optical flow while in our approach, we capture it via the skeleton data, which is known to be useful for action recognition [27]. Moreover, this approach does not use the shape information in the images as we do.

Part-aware LSTM [19] is proposed for action recognition, which separates the memory cell of the LSTM into sub-cells, one for each human body part. Finally, the outputs of all sub-cells are concatenated in one vector. The aim of this design was to force the system to learn how to classify the action based on the long-term temporal representation of each part alone before concatenating their outputs to make the final decision. Different from our approach, this approach only uses 3D skeleton data and does note use CNN for feature extraction.

Spatio-Temporal LSTM (ST-LSTM) is proposed in [13], in which a tree-traversal method is used for the spatial representation of 3D skeleton. Also, a trust gate is added to LSTM, for temporal representation, ST-LSTM(TT+TG) , to check the reliability of the inputs. This model is currently considered the state of the art in human action recognition from skeleton data. However, it relies only on skeleton data without fusing multiple modalities, as we do.

In [24] a multi-modal gesture recognition system is proposed which uses Deep Belief Networks (DBN) and CNN3D to manage the skeleton dynamics and RGB/depth images, respectively. Then, their decisions are fused to learn the emission probabilities of a Hidden Markov Model (HMM). The ModDrop model [16] fuses multiple modalities (articulated pose, depth maps, audio stream, and RGB images) in a deep convolutional gesture recognition system with multiple branches to deal with all modalities. Some modalities are dropped during training to overcome the presence of noisy data. Similar to [16, 24], we use multiple modalities. However, we focus on integrating the modalities and using a hybrid end-to-end system that includes both spatial feature extraction and temporal modeling.

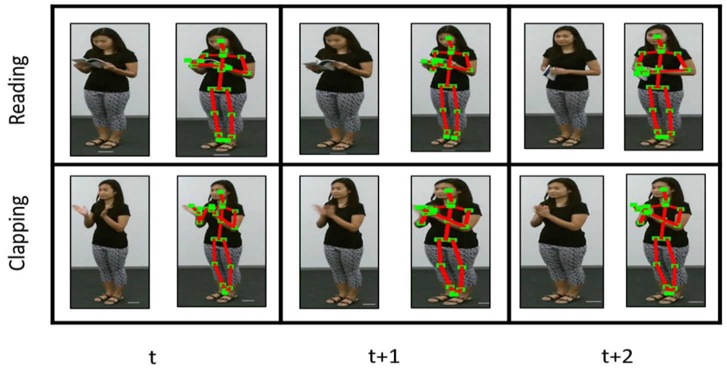

Figure 1: One running example of two actions, Reading and Clapping, from the NTU dataset [19], where the skeleton poses are relatively similar at times $t$, $t+1$, and $t+2$ while the body shapes are different. Upon integrating pose with part shapes, our model can discriminate between the two actions.

## 3 Network Model for Representing Multiple Modalities

Luckily, commonly used depth sensors readily provide data in multiple modalities, such as 3D skeleton, RGB images, depth maps, and even more. These modalities are complementary to one another as far as action recognition is concerned. For example, the 3D skeleton data capture the pose and motion of the action performer. However, they are often very noisy. Moreover, they do not capture other important information about the action, such as the hand shape and the manipulated object. This extra information can possibly be extracted from RGB images and/or depth maps. Therefore, we believe that fusing multiple modalities is unavoidable for robust action recognition.

Multiple modalities are not only complementary, they are also integrable. For example, 3D skeleton data give us information about the location of the body and its parts in the images and depth maps. Hence, if we are training a network on an image, we do not need to waste space and processing time in areas of the image outside the body, nor we need to waste the network's training epochs in trying to automatically recognize where the important parts in the image are. Therefore, we believe that integrating multiple modalities can be very helpful for efficient and effective action recognition.

In our proposed model, we make use of the two notions above. First, our model uses both 3D skeleton data and image data. Second, we benefit from our knowledge of the 3D skeleton data to focus only on the body parts such as face, left hand, right hand, left foot, and right foot, in the images because these parts typically include most of the information needed to recognize the performed action that is not captured in the skeleton data. Figure 1 shows samples of two different actions, in which the poses are similar at times $t$, $t+1$, and $t+2$ while the body part shapes (the hands in this case) can differentiate between them.

The proposed model consists of two sub-models: CL1D (for CNN+LSTM for 1D data) to receive the 3D skeleton stream, and CL2D (for CNN+LSTM for 2D data) to receive the image stream for different body parts. The two sub-models benefit from the power of the CNN to extract the spatial dependencies of the input stream, which is adjacent joints in the case of the skeleton, and adjacent pixels in the case of the images. Then, the extracted

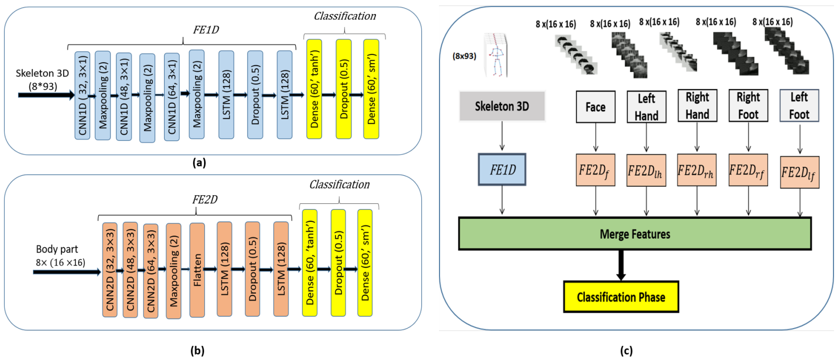

Figure 2: The architectures of the two sub-models, CL1D and CL2D, and the unified model, MCL: (a) CL1D's architecture to deal with the skeleton data; (b) CL2D's architecture to deal with images for one body part; (c) MCL's architecture, which combines models for skeleton data and images for multiple body parts (face, left hand, right hand, left foot, and right foot).

features are fed to a LSTM to model the temporal pattern in the extracted features. The outputs of the two sub-models are combined to make the final classification decision. In the next subsections, the architectures of **CL1D**, **CL2D**, and **MCL** are presented.

## 3.1 CL1D: The Pose and Motion Model

The 3D skeleton is an excellent source of information about the body pose and motion over time. The CL1D sub-model focuses on the 3D skeleton modality to capture such information. The 3D skeleton is typically provided as the 3D coordinates for a number of body joints. In the following, we will first explain the pre-processing performed on those raw joint coordinates. Then, we will explain the architecture of the CL1D sub-model.

### 3.1.1 3D Skeleton Preprocessing

There are three main preprocessing steps: (1) Joint rearrangement, (2) Coordinate normalization, and (3) Frames sampling.

Joint rearrangement aims to put the skeleton coordinates in the form that enables the CNN1D to detect the spatial pattern in the body part joints. We followed the joint arrangement scheme proposed in [15], which selects only 16 out of the 25 skeleton joints provided by the Kinect v2 sensor. The joints are arranged in a tree-traversal order while allowing repetition of some joints such that the total number of joint instances becomes 31. Joint coordinates are stacked so that joints of the same traversal path are put next to one another and ordered according to their position in the traversal tree. In this way, neighboring coordinates are more correlated, which makes it easier for the network to learn patterns of correlation between the adjacent joint coordinates.

The second processing stage is the joint coordinate normalization. The objective of this step is to make the joint coordinates invariant to the subject's location and orientation with respect to the camera and invariant to the subject's skeleton size. It is performed over two phases. The first phase transforms the skeleton joint coordinates from the camera coordinates to subject coordinate system, whose origin and main axes are defined with respect to the subject's body. The second phase maps each dimension of the joint coordinates over the entire sequence onto the interval $[0,1]$. In other words, this phase treats all the joints coordinates in the whole sequence as one group of points and linearly maps this group into the unit 3D box such that each of the box's boundaries includes at least one point.

The last pre-processing step is frame sampling, which has two objectives: First, to fix the length of the input sequence to make it suitable for training the network. Second, to make the model more robust by introducing some randomness. Particularly, we follow the frame sampling scheme of [19], which divides the input sequence into $N$ equally long segments, and then randomly chooses one frame from each segment. Similar to [19], we chose $N = 8$.

### 3.1.2 CL1D's Architecture

The first sub-model receives randomly chosen frames from $N$ segments, as explained above. For the skeleton produced by the Kinect v2, which has 25 joints, only 16 joints are selected, some of which are repeated to reach 31 joints as in [15]. Therefore, the skeleton is represented by 93 values (31 joints $\times$ 3 dimensions). With $N = 8$, the input layer to CL1D has ($8 \times 93$) nodes. The overall architecture of CL1D is shown in Figure 2(a). The architecture consists of two stages: 1D Feature Extraction ($FE1D$ ) and Classification.

The feature extraction stage, $FE1D$, uses three consecutive convolutional layers with numbers of filters 32, 48, and 64, respectively, all of size $3 \times 1$, and a rectified linear (relu) activation function to simplify and improve the speed of training [3]. This three-layer design allows for three levels of feature extraction, which would hopefully match the intuition of detecting low level (edge-like) patterns at the lowest level, body part motion at the second level, and whole body motion at the top level (similar to the three-level manually-designed hierarchy in [1]).

The output of each convolutional layer is reduced in size using a max-pooling layer that halves the number of features. After that, the reduced features are sent one sample at a time to the LSTM to learn the temporal pattern present in the sequence of features over time. The number of LSTM neurons was chosen to be 128, which was found to give the best validation accuracy over the trials of 16, 32, 64, 128, and 256 neurons. We use two cascaded LSTMs of the same architecture except that the first returns the whole generated output sequence ($8 \times 128$) after each time step while the second returns the last output (128). The two LSTMs follow the formulation in [7] to update their input ($i^t$), forget ($f^t$), output ($o^t$) gates, in addition to cell state ($C^t$) as well as its output vector ($h^t$), at each time step. The input vector ($U^t$) to the first LSTM layer is the extracted feature maps of the previous layer while the output vector of the first LSTM layer is the input to the second LSTM layer. For completeness, we include below the LSTM update equations:

$$i^t = sigmoid(W_{ui}U^t + W_{hi}h^{t-1} + W_{ci}C^{t-1} + b_i) \tag{1}$$

$$f^t = sigmoid(W_{uf}U^t + W_{hf}h^{t-1} + W_{cf}C^{t-1} + b_f) \tag{2}$$

$$o^t = sigmoid(W_{uo}U^t + W_{ho}h^{t-1} + W_{co}C^{t-1} + b_o) \tag{3}$$

$$C^t = f^t c^{t-1} + i^t tanh(W_{uc}U^t + W_{hc}h^{t-1} + b_c) \tag{4}$$

$$h^t = o^t tanh(C^t) \tag{5}$$

Where $i^t$,$f^t$, and $o^t$, are the input, forget, and output gates, respectively; $U^t$ and $C^t$ are the the input vector and the cell state at time $t$; $h^t$ and $h^{t-1}$ are the output vectors (hidden states) at time $t$ and $t-1$, respectively; $b_c$ is the bias for cell state; also, $b_i$, $b_f$, and $b_o$ are the biases of the input, forget, and output gates, respectively; while $W_{ui}$, $W_{hi}$, $W_{ci}$, $W_{uf}$, $W_{hf}$, $W_{cf}$, $W_{uo}$, $W_{ho}$, $W_{co}$, $W_{uc}$, $W_{hc}$, and $W_{co}$ are weight matrices.

Finally, the classification stage of CL1D uses two Fully Connected Layers (FCL), with a number of hidden neurons equal to the number of classes in the dataset. The first FCL uses tanh as an activation function and the second uses the softmax activation function to determine the probability of assigning an input sequence to one of the action classes. We also use a dropout layer [3] of probability 0.5 after the first LSTM layer to avoid over-fitting.

## 3.2 CL2D: The Part-Shape Model

Despite the utility of 3D skeleton data in identifying an action, they are lacking vital elements, such as both hands and feet shapes and the manipulated objects. In the second sub-model of our proposed model, CL2D, we use the images as a complementary modality to 3D skeleton data. To make processing and learning faster, we integrate our knowledge about the skeleton data by cropping only specific body part regions from the images. In the following, we will first explain the pre-processing steps, then, the architecture of CL2D and how it is used with the body parts is explained.

### 3.2.1 Image Preprocessing

The preprocessing of RGB images involves three stages. First, we convert every RGB image to grayscale in order to save in time and space complexities. Then, the target body parts, which are the face, left hand, right hand, left foot, and right foot, in this paper, are cropped from the image. This is done by finding the smallest bounding box in the image that includes all the part's joints. Then, a margin of 20% of each dimension of the bounding box is added to the two sides of that dimension to make sure the entire part is visible in the cropped patch. Finally, all cropped patches of the body parts are resized to $16 \times 16$. In the second step, the intensity is normalized by subtracting the mean pixel values over the training samples and dividing by the range, which is 255. Finally, in the third step, to fix the number of frames, the same frame sampling procedure used in the CL1D sub-model is applied (Section 3.1).

### 3.2.2 CL2D's Architecture

Figure 2(b) illustrates the architecture of the CL2D sub-model, which is used with each body part to represent its shape. CL2D, similar to CL1D, consists of two stages: 2D Part Feature Extraction ($FE2D_{part}$) and Classification. As shown in Figure 2(c), five instances of $FE2D_{part}$, which are $FE2D_f$, $FE2D_{lh}$, $FE2D_{rh}$, $FE2D_{lf}$, and $FE2D_{rf}$, are simultaneously deployed in our model to extract face, left hand, right hand, left foot, and right foot part 2D features, which are not captured in the 3D skeleton data. Similar to CL1D's architecture, CL2D deploys both convolutional and LSTM layers. However, different from CL1D's architecture, CL2D uses only one max-pooling layer after the third convolutional layer because the input dimension is relatively small, $16 \times 16$. Similar to CL1D, the CL2D deploys two cascaded LSTM layers, of 128 neurons each, after the three CNN2D, max-pooling, and flatten layers. The LSTM layers follow the update equations 1- 5. A dropout layer of probability 0.5 is added in between the two LSTM layers to avoid over-fitting. As illustrated in Figure 2(b), 32, 48, and 64 filters are used for the first, second, and third CNN2D layers of CL2D, respectively, all of size $3 \times 3$. The first CNN2D layer receives 8 images one at a time of size $16 \times 16$.

The output of each Flatten layer of the feature extraction phase of all parts are concatenated together to generate one feature vector that represents the body shape in each action. The classification stage of the CL2D sub-model is identical to the CL1D sub-model's classification stage.

### 3.3 Combining Pose-Motion and Part-Shape Models in MCL

The main contribution of our work is to combine and integrate both the pose and motion, which are captured by the CL1D sub-model, and body parts shapes, which are captured by the the CL2D sub-model. Both of the mentioned sub-models, CL1D and CL2D, are first trained separately. Afterwards, their flattened output features ($FE1D$, $FE2D_f$ , $FE2D_{lh}$, $FE2D_{rh}$, $FE2D_{rf}$, and $FE2D_{lf}$) are concatenated to generate one feature vector, $F$, which represents both motion and shape of the body as shown in equation (6). The classification phase of MCL receives the $F$ and trains two consecutive FCLs with a number of neurons equal to the dataset classes in each. The first FCL uses tanh activation function while the second uses a softmax activation function. The overall architecture of the proposed combined model (**MCL**) is shown in Figure 2(c).

$$F = [FE1D \| FE2D_f \| FE2D_{lh} \| FE2D_{rh} \| FE2D_{rf} \| FE2D_{lf}] \tag{6}$$

## 4 Experiments

In this section, we begin with a brief description of the dataset, then, we explain the implementation details, and finally, we present our experimental results and comparisons.

### 4.1 Dataset Description

We performed our experiments on one of the challenging datasets, which is the NTU-RGB+D [19]. This dataset consists of 56880 samples[1] of 60 action classes. This large size fits well with the data-hungry nature of deep learning. The actions are performed by 40 subjects of different ages and body lengths. Also, the dataset is collected simultaneously from three Kinect cameras with different viewing angles and distances from the subject, making an overall of 17 different setups. We followed the evaluation protocol suggested by the authors of the dataset [19], which consists of two scenarios: cross-view and cross-subject. While many other action recognition datasets have been publicly released [21, 25, 28], none of them suits our method, which requires the availability of both skeleton and image sequences, along with the mapping between joints and their locations in the images. Moreover, none of them is large enough for reliably training deep learning models.

### 4.2 Implementation Details

Experiments are conducted on a workstation with an Intel (R) Xeon(R) CPU E5-2699 v3 @ 2.30 GHZ, 256 GB RAM, and a GPU of type NVIDIA Quadro K4200, running a Windows 10 64-bit operating system. We used the latest version of Theano and Keras packages [12] for the implementation of the sub-models (CL1D and CL2D) and the combined model (MCL).

[1]The authors reported that 302 samples are missing skeletons and should be removed.

The two sub-models, CL1D and CL2D, are trained using the root mean square propagation optimizer[2], which is a good choice when LSTM is involved in the model.

We followed the same technique described in [19] to filter out erroneous skeleton models in the data.

The proposed combined model has about 6 million parameters, which means that the model needs a huge amount of time to be trained. To overcome this problem, we deployed fine-tuning by training each sub-model separately. The first sub-model is trained as an independent model over the 3D skeleton data until no improvement in validation accuracy is observed. Then, we saved the model weights that achieved the highest validation accuracy. At the same time we trained the CL2D sub-model which receives the body parts. However, because this sub-model is considerably slower to train (one day to be trained for only 15 epochs on our machine), we saved only the weights of this model that achieved 35% as a validation accuracy. To train the combined model, we loaded the saved weights of the two sub-models, froze the first sub-model from further training, and left the second sub-model opened to continue training with the merging part[3]. In this way, we reduced the number of learning parameters and increased the speed of training the combined model.

## 4.3 Experimental Results and Discussions

As explained earlier, we performed two types of experiments as in [19]: cross-view (CV) and cross-subject (CS). Table 1 summarizes the obtained results for our experiments compared to the state of the art method, which is the ST-LSTM (TT+TG) model [15] and other methods.

Table 1 shows the results of nineteen methods/models that were tested in both the CS and CV scenarios. Some of them used hand-crafted features, and others used deep learning methods to automatically extract features. The results of the first 13 methods were obtained from [19]. The methods from 1 to 6 in the table used hand-crafted features based on the depth and/or 3D skeleton data. HOG$^2$ [17], Super Normal Vector [26], and HON4D [18] achieved their highest score (32.24%, 32.82%, and 30.56 % respectively) in the CS scenario because these representations are not view-point invariant. On the other hand, Lie Groups [20], Skeleton Quads [6], and FTP Dynamic Skeletons [10] achieved better scores (52.76 %, 41.56%, and 65.22 % ) in the CV scenario because these representations are view-point invariant and hence perform better in the CV scenario because in this scenario the same subject may appear in training and testing, which make the problem easier. Deep learning techniques are used from method 7 to the end of the table. The best scores that were achieved in both CV and CS scenarios, which are 62.99 % and 70.27 %, respectively, are obtained by using 2-Layer P-LSTM [19]. Methods 14, 15 and 16 are the work of [15], which is considered the current state of the art with scores outperforming prior methods.

The last three rows in the table contain our results. When CL1D is used to classify actions based on the body motion alone, the recognition rate for both CS and CV experiments are alert(64.2% and 72.11%), respectively. However, when MCL is used to capture both the body motion and part shape, the results go up to **74.2 %** and **81.4%**, for the CS and CV scenarios, respectively, which are superior to the current state of the art.

The table contains an extra row for the results of MCL trained on the upper body part as whole after being cropped and reduced to the size of $128 \times 128$. This model was trained for the same number of epochs as the MCL (body parts) model. However, 20 frames are

[2]The optimizer parameters are selected to be as follows: learning rate ($L_r = 0.001$), gradient moving average decay factor ($\rho = 0.9$), fuzzy factor ($\varepsilon = 10^{-8}$), and learning rate decay over each update ($decay = 0.0$).

[3]The MCL is trained for 200 epochs after fine tuning.

| | Method | CS | CV |
|---|---|---|---|
| 1 | HOG$^2$ [17] | 32.24% | 22.27% |
| 2 | Super Normal Vector [26] | 31.82% | 13.61% |
| 3 | HON4D [18] | 30.56% | 7.26% |
| 4 | Lie Group [20] | 50.08% | 52.76% |
| 5 | Skeletal Quads [5] | 38.62% | 41.36% |
| 6 | FTP Dynamic Skeletons [10] | 60.23% | 65.22% |
| 7 | HBRNN-L [4] | 59.07% | 63.97% |
| 8 | 1 Layer RNN [19] | 56.02% | 60.24% |
| 9 | 2 Layer RNN [19] | 56.29% | 64.09% |
| 10 | 1 Layer LSTM [19] | 59.14% | 66.81% |
| 11 | 2 Layer LSTM [19] | 60.69% | 67.29% |
| 12 | 1 Layer P-LSTM [19] | 62.05% | 69.40% |
| 13 | 2 Layer P-LSTM [19] | 62.93% | 70.27% |
| 14 | ST-LSTM (Joint Chain) [15] | 61.7 % | 75.5% |
| 15 | ST-LSTM (Tree Traversal) [15] | 65.2% | 76.1% |
| 16 | ST-LSTM (TT +TG) [15] | 69.2% | 77.7% |
| 17 | **CL1D** | **64.2%** | **72.11%** |
| 18 | **MCL (upper body)** | **70.03%** | **78.01%** |
| 19 | **MCL (body parts)** | **74.2%** | **81.4%** |

Table 1: Cross subjects and Cross views accuracies in NTU RGB+D dataset

sampled per sequence in this model because training with less samples caused under-fitting. Despite the extra information provided to this model, it took much longer training time to exceed the state of the art results, and yet fell clearly behind the MCL model with body parts, especially in the CS scenario. This verifies our initial hypothesis that combining and integrating modalities, as well as leveraging the powers of CNN and LSTM are effective mechanisms in action recognition.

We also notice that the enhancement obtained by our proposed method in the CV scenario is less than the enhancement obtained in the CS scenario. This is primarily because part images are not view-invariant. Addressing this issue is part of our future work.

## 5 Conclusion and Future Work

The **MCL** model is proposed in this paper to combine both CNN and LSTM deep learning techniques and also incorporate multiple modalities for action recognition. The CNN layers are used to extract features from an input modality. The type of the deployed CNN depends on the type of the modality, e.g. CNN1D is used for skeleton data and CNN2D is used for images. A prototype for MCL is developed that combined the two modalities of 3D skeleton and images. Not only the two modalities are combined, but, they are also integrated by using knowledge of body location in the image from the skeleton data to crop the body parts. Experiments on the large NTU-RGB+D dataset in two experimental scenarios (CS and CV) showed the superiority of the proposed model over prior work. In the future, we will extend this work by deploying and integrating more modalities.

## References

[1] Moez Baccouche, Franck Mamalet, Christian Wolf, Christophe Garcia, and Atilla Baskurt. Sequential deep learning for human action recognition. In *International Workshop on Human Behavior Understanding*, pages 29–39. Springer, 2011.

[2] Guilhem Chéron, Ivan Laptev, and Cordelia Schmid. P-cnn: Pose-based cnn features for action recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3218–3226, 2015.

[3] George E Dahl, Tara N Sainath, and Geoffrey E Hinton. Improving deep neural networks for lvcsr using rectified linear units and dropout. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8609–8613. IEEE, 2013.

[4] Yong Du, Wei Wang, and Liang Wang. Hierarchical recurrent neural network for skeleton based action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1110–1118, 2015.

[5] Georgios Evangelidis, Gurkirt Singh, and Radu Horaud. Skeletal quads: Human action recognition using joint quadruples. In *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pages 4513–4518. IEEE, 2014.

[6] Simon Fothergill, Helena Mentis, Pushmeet Kohli, and Sebastian Nowozin. Instructing people for training gestural interactive systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1737–1746. ACM, 2012.

[7] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[8] Alexander Grushin, Derek D Monner, James A Reggia, and Ajay Mishra. Robust human action recognition via long short-term memory. In *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pages 1–8. IEEE, 2013.

[9] Fei Han, Brian Reily, William Hoff, and Hao Zhang. Space-time representation of people based on 3d skeletal data: A review. *arXiv preprint arXiv:1601.01006*, 2016.

[10] Jian-Fang Hu, Wei-Shi Zheng, Jianhuang Lai, and Jianguo Zhang. Jointly learning heterogeneous features for rgb-d activity recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5344–5352, 2015.

[11] Mohamed E Hussein, Marwan Torki, Mohammad Abdelaziz Gowayyed, and Motaz El-Saban. Human action recognition using a temporal hierarchy of covariance descriptors on 3d joint locations. In *IJCAI*, volume 13, pages 2466–2472, 2013.

[12] Vassili Kovalev, Alexander Kalinovsky, and Sergey Kovalev. Deep learning with theano, torch, caffe, tensorflow, and deeplearning4j: Which one is the best in speed and accuracy? 2016.

[13] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[14] Grégoire Lefebvre, Samuel Berlemont, Franck Mamalet, and Christophe Garcia. Blstm-rnn based 3d gesture classification. In *International Conference on Artificial Neural Networks*, pages 381–388. Springer, 2013.

[15] Jun Liu, Amir Shahroudy, Dong Xu, and Gang Wang. Spatio-temporal lstm with trust gates for 3d human action recognition. In *European Conference on Computer Vision*, pages 816–833. Springer, 2016.

[16] Natalia Neverova, Christian Wolf, Graham Taylor, and Florian Nebout. Moddrop: adaptive multi-modal gesture recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(8):1692–1706, 2016.

[17] Eshed Ohn-Bar and Mohan Trivedi. Joint angles similarities and hog2 for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 465–470, 2013.

[18] Omar Oreifej and Zicheng Liu. Hon4d: Histogram of oriented 4d normals for activity recognition from depth sequences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 716–723, 2013.

[19] Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and Gang Wang. Ntu rgb+ d: A large scale dataset for 3d human activity analysis. *arXiv preprint arXiv:1604.02808*, 2016.

[20] Raviteja Vemulapalli, Felipe Arrate, and Rama Chellappa. 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*, pages 588–595, 2014.

[21] Jiang Wang, Zicheng Liu, Ying Wu, and Junsong Yuan. Mining actionlet ensemble for action recognition with depth cameras. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1290–1297. IEEE, 2012.

[22] Keze Wang, Xiaolong Wang, Liang Lin, Meng Wang, and Wangmeng Zuo. 3d human activity recognition with reconfigurable convolutional neural networks. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 97–106. ACM, 2014.

[23] Pichao Wang, Wanqing Li, Zhimin Gao, Jing Zhang, Chang Tang, and Philip O Ogunbona. Action recognition from depth maps using deep convolutional neural networks. 2015.

[24] Di Wu, Lionel Pigou, Pieter-Jan Kindermans, Nam Do-Hoang Le, Ling Shao, Joni Dambre, and Jean-Marc Odobez. Deep dynamic neural networks for multimodal gesture segmentation and recognition. *IEEE transactions on pattern analysis and machine intelligence*, 38(8):1583–1597, 2016.

[25] Lu Xia, Chia-Chih Chen, and JK Aggarwal. View invariant human action recognition using histograms of 3d joints. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, pages 20–27. IEEE, 2012.

[26] Xiaodong Yang and YingLi Tian. Super normal vector for activity recognition using depth sequences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 804–811, 2014.

[27] Angela Yao, Juergen Gall, Gabriele Fanelli, and Luc Van Gool. Does human action recognition benefit from pose estimation? In *Proceedings of the British Machine Vision Conference*, pages 67.1–67.11. BMVA Press, 2011. ISBN 1-901725-43-X. http://dx.doi.org/10.5244/C.25.67.

[28] Kiwon Yun, Jean Honorio, Debaleena Chattopadhyay, Tamara L Berg, and Dimitris Samaras. Two-person interaction detection using body-pose features and multiple instance learning. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, pages 28–35. IEEE, 2012.

[29] Wentao Zhu, Cuiling Lan, Junliang Xing, Wenjun Zeng, Yanghao Li, Li Shen, and Xiaohui Xie. Co-occurrence feature learning for skeleton based action recognition using regularized deep lstm networks. *arXiv preprint arXiv:1603.07772*, 2016.