CHAPTER IV

BODY JOINTS AND TRAJECTORY GUIDED 3D DEEP CONVOLUTIONAL DESCRIPTORS FOR HUMAN ACTIVITY IDENTIFICATION

Among the many exciting areas of study in computer vision, action recognition in video is at the forefront. Videos, in contrast to still photos, which only include spatial information, are spatio-temporal streams in three dimensions (3D). Much research has focussed on how to take into consideration both appearance and motion data for video-based action recognition. Cao et al. (2016) used a 3D deep Convolutional Neural Network (3D-CNN) to aggregate convolutional activations into discriminative descriptors based on the joint positions, demonstrating HAR with JDD. Using this procedure, the video was cut into equal-length segments. In order to further analyze the video, a three-dimensional convolutional feature map was generated. After that, they combined the activations at each matching spot and the combined activations into a single clip. The l_2 approach was then used to pool the attributes of all the clips and normalize them. To complete the classification process, a linear SVM was used. Further, the positions of the body's joints were identified using either manual annotation or a commercially available skeleton estimate method (Cao et al., 2017). Motivated by the need to concurrently learn direction from the body's joints and collect spatiotemporal data, researchers developed a proposed model. The body-joint guided feature pooling was achieved using a sampling strategy after modeling the pooling process as a bilinear product operation. It was however extremely expensive to calculate the positions of body joints from a large dataset using the skeleton estimation approach, and it took a significant amount of time to perform. Furthermore, it was imperative that recognition accuracy be dramatically improved.

To improve recognition accuracy, this procedure combines an optical flow extraction with a bilinear model with two streams. By employing this technique, optical flows can be extracted automatically. Optical flows are locations along trajectories connecting body joint in 2 videos. The bilinear product of two C3D streams allows for the simultaneous capture of spatiotemporal information, feature extraction, and the

generation of pooling descriptors for video sequences. At last, a linear support vector machine is used to classify the human-trained video descriptors.

4.1 PROPOSED METHODOLOGY

By monitoring body joints with optical flow, or trajectory points in the video sequence, a two-stream bilinear C3D network model is used to automatically predict the spatiotemporal critical places in 3D convolutional feature maps. Next, a general bilinear formulation is employed to do forward and inverse computations for this model.

4.1.1 Joints and Trajectory-Pooled 3D Deep Convolutional Descriptors

The Fig. 4.3 architecture of the C3D network is used in this process.

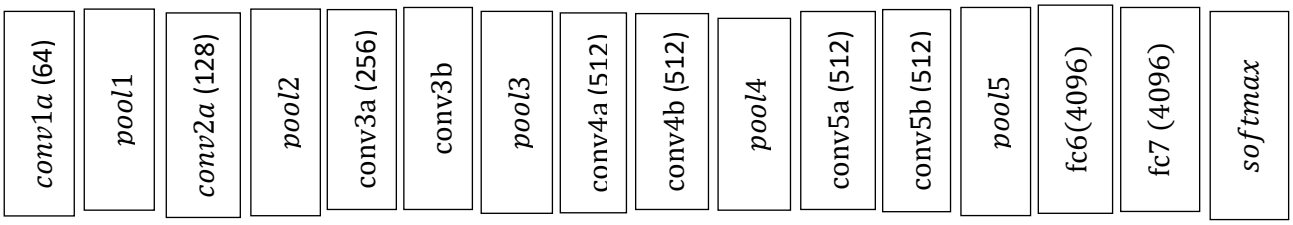


Figure 4.1. Architecture of C3D Network

Parenthesis denote the total number of convolutional filters. After each convolutional $(conv)$ layer, a ReLu layer is added. With stride and padding of 1 in both dimensions, the 3D convolution kernel is $d \times k \times k(3 \times 3 \times 3)$, where d is the temporal depth and k is the spatial size. Except for pool1, which uses a $1 \times 2 \times 2$ pooling kernel, all other kernels are $2 \times 2 \times 2$. C3D accepts a 20-frame clip as input and scales each frame down to 256×256 pixels (width×height).

4.1.1.1 Body Joints and Optical Flow Mapping Schemes*:*

Methods for JTDD that use 3D convolutional feature maps to map body joints and optical flow to points are compared and contrasted here. Using a ratio scaling technique, in which the output of the network is multiplied by the input in both space and time, the original video frame's body joint and optical flow coordinates can be converted into feature maps.

$$\left(x_c^i, y_c^i, t_c^i\right) = \left(\overline{(r_x^i \cdot x_v)}, \overline{\left(r_y^i \cdot y_v\right)}, \overline{(r_t^i \cdot t_v)}\right) \quad (4.1)$$

$$\left(l_c^i, m_c^i, n_c^i\right) = \left(\overline{(r_l^i \cdot l_v)}, \overline{(r_m^i \cdot m_v)}, \overline{(r_n^i \cdot n_v)}\right) \quad (4.2)$$

Point coordinates in the i^{th} 3D convolutional feature maps are represented by $\left(x_c^i, y_c^i, t_c^i\right)$, while the rounding operator in (4.1) and (4.2) is represented by $\overline{(\cdot)}$. where (x_v, y_v, t_v) are the body joint coordinates from the original video sequence and $\left(r_x^i, r_y^i, r_t^i\right)$ are the size ratios of the i^{th} 3D convolutional feature maps to the video clip. Similarly, (l_v, m_v, n_v) are the original video sequence's trajectory point coordinates, while $\left(r_l^i, r_m^i, r_n^i\right)$ are the size ratios of the i^{th} 3D convolutional feature maps, respectively.

The exact location of the point on the convolutional feature map that corresponds to the body joint and trajectory point is determined by considering the kernel size, stride, and padding of each layer. This method is known as coordinate mapping. To calculate the mapping connection layer by layer between points. Suppose p_i is a point on the i^{th} layer, and its coordinates are (x_i, y_i, t_i) and (l_i, m_i, n_i), respectively. Point p_{i+1} is found by projecting p_i onto the next layer deeper. Following is the notation for the coordinate mapping from the i^{th} convolutional layer to the $(i+1)^{th}$ pooling layer:

$$x_{i+1} = \frac{1}{s_i^x}\left(x_i + padding_i^x - \frac{k_i^x - 1}{2}\right) \quad (4.3)$$

$$y_{i+1} = \frac{1}{s_i^y}\left(y_i + padding_i^y - \frac{k_i^y - 1}{2}\right) \quad (4.4)$$

$$z_{i+1} = \frac{1}{s_i^z}\left(z_i + padding_i^z - \frac{k_i^z - 1}{2}\right) \quad (4.5)$$

$$l_{i+1} = \frac{1}{s_i^l}\left(l_i + padding_i^l - \frac{k_i^l - 1}{2}\right) \quad (4.6)$$

$$m_{i+1} = \frac{1}{s_i^m}\left(m_i + padding_i^m - \frac{k_i^m - 1}{2}\right) \quad (4.7)$$

$$n_{i+1} = \frac{1}{s_i^n}\left(n_i + padding_i^n - \frac{k_i^n - 1}{2}\right) \quad (4.8)$$

All of these formulas rely on the x −coordinates of the i^{th} layer's stride (s_i^x), kernel size (k_i^x), and padding ($padding_i^x$). Similar considerations are given to the y, z, l, m and n, dimensions.

For ReLU layers, the coordinate mapping relationship is maintained as the size of the feature maps is not changed during the operations. The formula for this mapping between coordinate systems is as follows:

$$(x_{i+1}, y_{i+1}, z_{i+1}) = (x_i, y_i, t_i) \quad (4.9)$$

$$(l_{i+1}, m_{i+1}, n_{i+1}) = (l_i, m_i, n_i) \quad (4.10)$$

The coordinate mapping connection between feature maps and the video sequence cannot be determined without considering the layers that came before it. Following the application of the values of C3D kernel sizes, strides, and paddings into (4.3)-(4.5) and (4.9), the following relationship is created between point coordinates in the i^{th} convolutional feature maps and the locations of body joints in the input video sequence:

$$\left(x_c^i, y_c^i\right) = \frac{1}{2^{i-1}} \cdot \left(x_v - \frac{2^{i-1}-1}{2}, y_v - \frac{2^{i-1}-1}{2}\right) \quad (4.11)$$

$$t_c^i = \frac{1}{2^{i-2}} \cdot \left(t_v - \frac{2^{i-2}-1}{2}\right) \quad (4.12)$$

The following description of the relationship between the coordinates of points in the i^{th} convolutional feature map and the points of trajectories in the input video sequence can be tweaked by repeatedly substituting different values for the kernel size, the stride length, and the padding length into equations (4.6)-(4.8), and (4.10).

$$\left(l_c^i, m_c^i\right) = \frac{1}{2^{i-1}} \cdot \left(l_v - \frac{2^{i-1}-1}{2}, l_v - \frac{2^{i-1}-1}{2}\right) \quad (4.13)$$

$$n_c^i = \frac{1}{2^{i-2}} \cdot \left(n_v - \frac{2^{i-2}-1}{2}\right) \quad (4.14)$$

4.1.1.2 Aggregation of Body Joint Points and Optical Flow

The video descriptor is utilized for categorization after the obtained features of frames throughout time have been aggregated. specified that C3D takes frames from the specified video as input, this makes sense. Points in 3D feature maps can be localized using body joints and trajectory points in video frames, hence revealing the

best spots to pool. Each body joint and trajectory point in a video frame is represented as a C -dimensional feature vector, where C is the number of feature map channels. C-dimensional feature vector $f_k^{i,t}$ for the k^{th} frame of the k^{th} clip is the outcome of a pooling operation directed by the i-th body joint. In a similar vein, $g_k^{i,t}$ represents the C-dimensional feature vector gathered under the guidance of the i^{th} trajectory point at the t^{th} frame of the k^{th} clip.

These methods are applied to the combined feature vectors from each frame of a video in order to provide a descriptor for the video. To create a representation of a frame, one joins together the feature vectors from that frame's pool. This characteristic of the $C \times N \times O \times L$ dimensions is expressed as:

$$f_k g_k = \begin{bmatrix} f_k^{1,1} g_k^{1,1}, f_k^{2,1} g_k^{2,1}, \dots, f_k^{N,1} g_k^{O,1}, f_k^{1,2} g_k^{1,2}, \\ f_k^{2,2} g_k^{2,2}, \dots, f_k^{N,2} g_k^{O,2}, \dots, f_k^{N,L} g_k^{O,L} \end{bmatrix} \quad (4.15)$$

In the equation (4.15), N is the total number of body joints, O is the total number of trajectory points in each frame, and L is the total duration of the video series. Then, a video descriptor is constructed $L2$ normalization by averaging the frame representations $\{f_1 g_1, f_2 g_2, \dots, f_k g_k\}$, where k is the total amount of video clips that have been counted. This method of aggregating JTDD yields the $C \times N \times O \times L$ dimension.

Another method of data aggregation involves transforming the feature vectors of each body joint and trajectory point in a single frame into a $C \times N \times O$ dimensional representation.

$$f_k^t g_k^t = \left[f_k^{1,t} g_k^{1,t}, f_k^{2,t} g_k^{2,t}, \dots, f_k^{N,t} g_k^{O,t} \right] \quad (4.16)$$

After that, L representations $\{f_k^1 g_k^1, f_k^2 g_k^2, \dots, f_k^L g_k^L\}$ within the same frame define that frame. It uses max + min pooling, where max + min pooling is defined as choosing the maximum value and the minimum value of each feature dimension, to merge these representations into a frame descriptor. There are a total of k frames. Finally, a video descriptor is constructed from a max + min pooled and $L2$ normalized set of k frame descriptors.

4.1.2 Two-Stream Bilinear C3D Model using Body Joints and Optical Flow

Selecting activations on convolutional feature maps at the relevant sites of body joints and trajectory points is equivalent to assigning hard weights to the activations, and this is how JTDD's novel body joint and optical flow driven feature pooling is implemented. An individual convolutional feature map is created for each body joint and trajectory point in a given video sequence, and then all of that information is integrated into a single heat map with the same spatiotemporal dimensions as the feature maps. The dimensions are written as $l \times h \times w$, where l represents length, h represents height, and w represents weight. Each point is represented by a 1 in the body joint location and trajectory, whereas all other positions are represented by 0. Pooling on a single feature map guided by the heat map of a single body joint and trajectory point can be described as a pixel-wise product between the 3D feature map and the 3D heat map, followed by a sum over all pixels.

JTDD makes use of M $(M = N \times O \times L)$ heat maps. The original $M \times l \times h \times w$ heat maps are then transformed into a 2D matrix $\mathcal{A}$ with M rows and $l \times h \times w$ columns. C3D data is converted into a 2D matrix $\mathcal{B}$ with C rows and $l \times h \times w$ columns using the $C \times l \times h \times w$ feature mapping. The resulting expression for the bilinear product is as follows:

$$\mathcal{P} = \mathcal{A}\mathcal{B}^T \tag{4.17}$$

In equation (4.17), $\mathcal{P}$ is an $M \times C$ matrix and $\mathcal{B}^T$ is the transposition of $\mathcal{B}$. All the values in the product matrix $\mathcal{P}$ are strung together as a long vector to achieve functionality similar to JTDD for video sequence. Similarly to JTDD, the feature can be expressed as L representations in $N \times C$ dimensions.

It is possible that it will automatically zero in on discriminative regions of feature maps using a 3D convolutional neural network (CNN) that has been pre-trained utilizing the guidance of body joint locations and trajectory data. After convolutional layer 5b ($conv5b$) and fully connected (fc) layers, a C3D without ReLU is used to regress the corresponding points' heat maps. Training sigmoid cross entropy loss, on a pixel-by-pixel basis, is defined as

$$e = \frac{1}{M}\sum_{m=1}^{M}\sum_{k=1}^{l}\sum_{j=1}^{h}\sum_{i=1}^{w}\left(p_{ijk}^{m}\log\hat{p}_{ijk}^{m} + \left(1 - p_{ijk}^{m}\right) \times \log\left(1 - \hat{p}_{ijk}^{m}\right)\right) \tag{4.18}$$

In equation (4.18), $\hat{p}_{ijk}^{m}$ is the ground-truth heat map value of the mth channel at location (i, j, k) and i, j, k are the indices in width, height, and length, and p_{ijk}^{m} is the sigmoid value of conv5b's output at location (i, j, k) of the m^{th} channel. For every body joint and trajectory node, the 3D attention model creates heat maps of soft weight. In order to generate video sequence descriptors, the convolutional feature maps from the original C3D are combined with the bilinear product using heat maps.

After that, the body's joint positions and trajectories are used to train a two-stream bilinear C3D model that automatically captures the spatiotemporal features and provides guidance. Both the attention stream (which employs the parameters from the pre-trained 3D attention model) and the feature stream (which employs the $conv$ layer from the original C3D) are combined using the bilinear product. The whole network is trained using the class label as input. Forward and inverse calculations using a general bilinear product used in this model are presented. The following is an explicit formula for computing a general form of the bilinear product:

$$\mathcal{P} = \mathcal{A}\mathcal{W}\mathcal{B}^{T} \tag{4.19}$$

The network is given a matrix of parameters, denoted by $\mathcal{W}$, of the form $\mathcal{P} \in \mathbb{R}^{M\times\mathcal{C}}, \mathcal{A} \in \mathbb{R}^{M\times K_1}, \mathcal{W}\mathbb{R}^{K_1\times K_2}, \mathcal{B} \in \mathbb{R}^{\mathcal{C}\times K_2}$. In contrast to (4.17), it is not required that $\mathcal{A}$ and $\mathcal{B}$ have the same number of columns here. In addition, $\mathcal{W}$ is used to learn additional statistical properties shared by $\mathcal{A}$ and $\mathcal{B}$. With its two independent streams, attention, feature extraction, keypoint pooling, and classification are all seamlessly integrated into this bilinear C3D network model. Using softmax loss and the provided class label, the complete network may be trained from the ground up. Using back propagation, it can calculate the bilinear layer's gradients as follows:

$$\frac{\partial e}{\partial \mathcal{A}} = \frac{\partial e}{\partial \mathcal{P}}\mathcal{B}\mathcal{W}^{T} \tag{4.20}$$

$$\frac{\partial e}{\partial \mathcal{B}} = \left(\frac{\partial e}{\partial \mathcal{P}}\right)^{T}\mathcal{A}\mathcal{W} \tag{4.21}$$

$$\frac{\partial e}{\partial \mathcal{W}} = \mathcal{A}^{T}\frac{\partial e}{\partial \mathcal{P}}\mathcal{B} \tag{4.22}$$

As the loss function e is back-propagated, its gradient with respect to the activation function $\mathcal{A}$ is indicated by $\frac{\partial e}{\partial \mathcal{A}}$. With the two-stream bilinear model, video

descriptors are generated by combining trajectory points and body joint locations. Finally, video descriptors are classified using linear support vector machines (Fan et al., 2008) to facilitate accurate action detection from video sequences. The SVM is often built as a hyperplane in an infinite-dimensional metric space. The hyperplane with the biggest functional margin (distance to nearest training sequences of any class) yields the most accurate HAR. Where x_i represents the video descriptors (instances) and y_i represents the labels (-1,+1), the training dataset is represented as a collection of instance-label pairs $(x_i, y_i), i = 1, \dots, n, x_i \in \mathbb{R}^n, y_i \in \{-1, +1\}$. For each class, the following unconstrained optimization problem must be solved in order to determine the hyperplane with the highest margin.

$$\min_w \frac{1}{2} w^T w + C \sum_{i=1}^{n} \xi(w; x_i, y_i) \quad (4.23)$$

The weight of the training sequences x_i is denoted by w in (4.23), while the penalty parameter is denoted by $C > 0$. Human actions can be identified once this optimization challenge is solved.

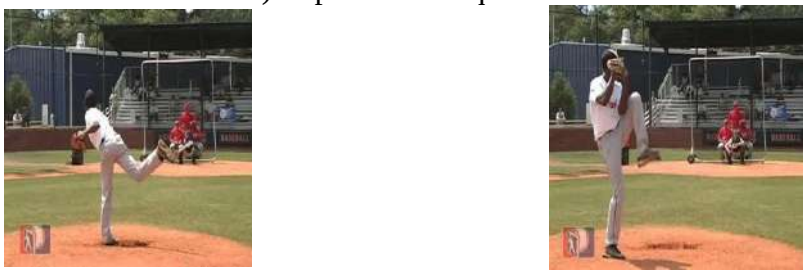
4.2 RESULTS AND DISCUSSIONS

Matlab2017b was used to evaluate the proposed JTDD model's recognition accuracy in contrast to that of the baseline JDD model. In order to measure effectiveness, the Penn Action dataset's 2326 video sequences from 15 action classes are used. The average number of annotated body joints each frame is 13, but there are videos in which not all of the marked body joints are in the frame or visible. The suggested model is trained on one half of the dataset, while the other half is used for testing. In order to train an attention model, the dataset is split into a training set of 84 and a testing set of 16 occurrences. The videos are compiled from a wide range of online sources. There are typically between 50 and 100 frames in a video. C3D features, points on a trajectory, and body joint coordinates are employed as starting points. As a result, they assess JTDD with these enhancements and compare it to other pooling schemes. The recognition accuracy is defined as the percentage of accurate identifications (TPs) in comparison to the total number of cases. Calculated as:

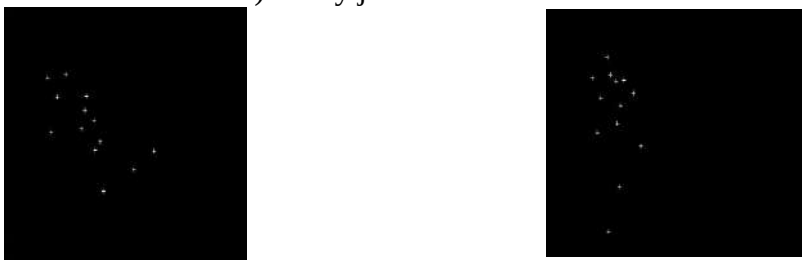
$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (4.24)$$

To clarify, FP refers to a False Positive result and FN to a False Negative one. Joint and trajectory point extraction is shown in Fig. 4.4.

a). Input video sequence



b). Body joints extraction



c). Trajectory points extraction

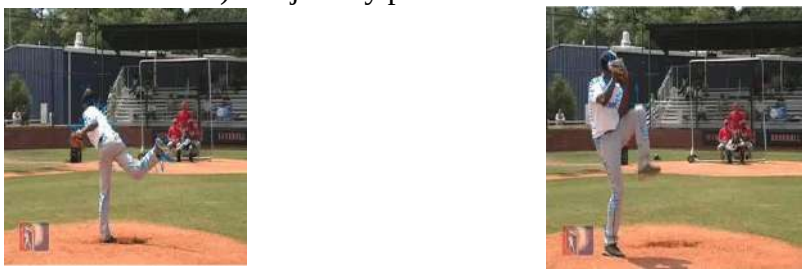


Figure 4.2. Results for Extraction of Body Joints and Trajectory Points

Table 4.1 displays the outcomes on the Penn Action dataset.

Table 4.1. Recognition Accuracy of Baselines and JTDD with Different Configurations on Penn Action Dataset

	Concatenate all the activations	JTDD Ratio Scaling (1×1×1)	JTDD Coordinate Mapping (1×1×1)	JTDD Ratio Scaling (3×3×3)	JTDD Coordinate Mapping (3×3×3)
Joint coordinates + trajectory coordinates	0.6120	-	-	-	-
$fc7$	0.7211	-	-	-	-
$fc6$	0.7368	-	-	-	-
$conv5b$	0.7052	0.8014	0.8599	0.8086	0.8367
$conv5a$	0.6305	0.7583	0.7834	0.7533	0.7628
$conv4b$	0.5324	0.7697	0.7601	0.7847	0.7993
$conv3b$	0.4297	0.7136	0.6845	0.7021	0.7014

In Table 4.1, the first column displays the recognition accuracies achieved by employing body joint coordinates with trajectory coordinates as a feature, as well as C3D features. Using only the locations of body joints and optical flow (in the form of trajectory points) is shown to be wasteful in this research. There is an improvement in discriminative power when using C3D features, which concatenate all activations in a given layer into a single long vector. $fc7$ has somewhat lower recognition accuracy than $fc6$. Since the original C3D on Penn Action dataset does not fine-tune, the second fc layer is probably more suitable for the classification of the pre-trained dataset. Pooling experiments at different 3D $conv$ layers for JTDD are presented, as are body joint and trajectory point mapping strategies.

In Table 4.2, the benefits of body joint and trajectory point driven pooling are shown to be superior to those of C3D features. Further, JTDDs across several $conv$ layers are pooled together to test for mutual compensation. Table 4.2 and Fig. 4.5 display the outcomes of several late fusion combinations with SVM scores on the Penn Action dataset.

Table 4.2. Recognition Accuracy of Fusing JTDD from Multiple Layers together on Penn Action Dataset

	Fusion Layers					
	$\boldsymbol{conv5b + fc6}$		$\boldsymbol{conv5b + conv4b}$		$\boldsymbol{conv5b + conv3b}$	
	JDD	**JTDD**	**JDD**	**JTDD**	**JDD**	**JTDD**
Accuracy	0.855	0.867	0.981	0.987	0.860	0.873

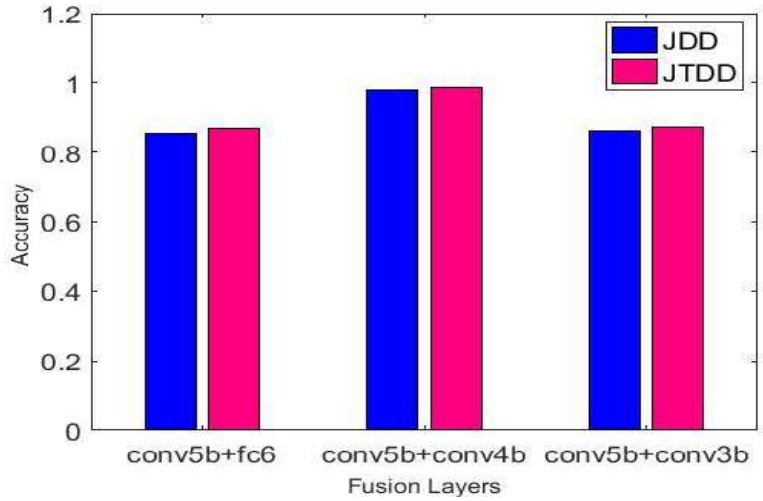


Figure. 4.3 Recognition Accuracy of Fusing JTDD from Multiple Layers together on Penn Action Dataset

Figure 4.3 shows that combining JTDDs from multiple layers almost likely improves recognition results. When JTDDs from conv5b and conv4b are used together, recognition performance is much improved. Accuracy improves as more qualities that complement one another are fused together.

The results of combining JTDD from different layers on the Penn action dataset are shown in Table 4.3 in terms of precision, recall, and f-measure.

Table 4.3. Precision, Recall, and F-measure of Fusing JTDD from Multiple Layers Together on Penn Action Dataset

Performance Metrics	Fusion Layers					
	$conv5b + fc6$		$conv5b + conv4b$		$conv5b + conv3b$	
	JDD	**JTDD**	**JDD**	**JTDD**	**JDD**	**JTDD**
Precision	0.839	0.855	0.968	0.975	0.841	0.856
Recall	0.844	0.861	0.976	0.982	0.848	0.869
F-measure	0.842	0.858	0.972	0.979	0.845	0.863

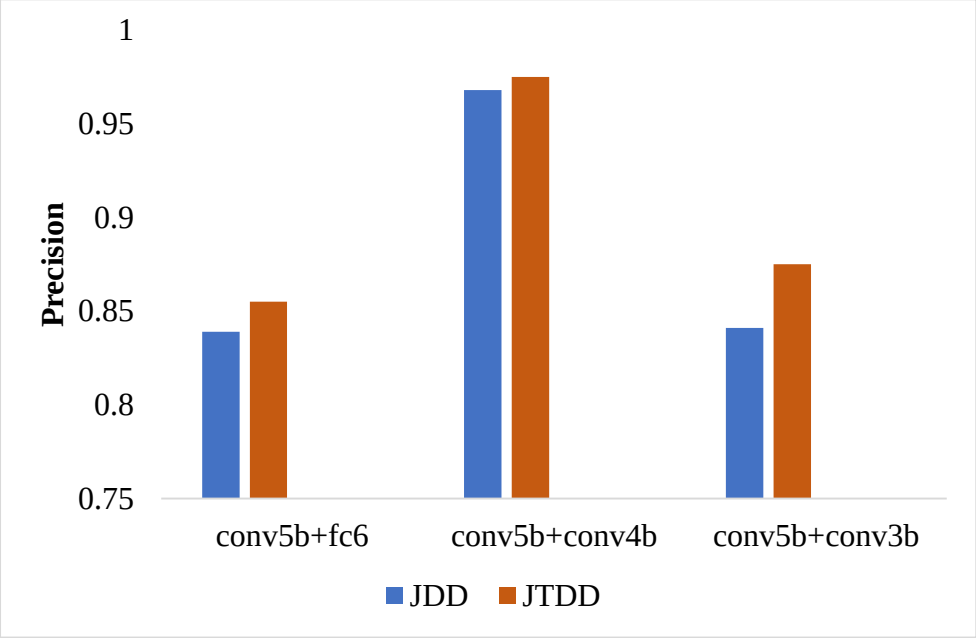


Figure. 4.4 Recognition precision of Fusing JTDD from Multiple Layers together on Penn Action Dataset

Figure 4.4 shows that combining JTDDs from multiple layers almost likely improves recognition results. When JTDDs from conv5b and conv4b are used together, recognition performance is much improved. Precision improves as more qualities that complement one another are fused together.

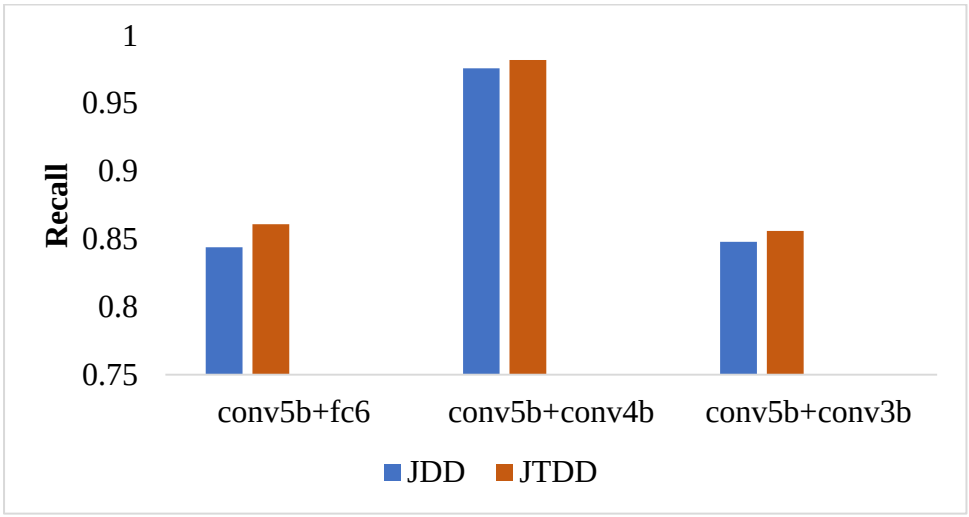


Figure. 4.5 Recognition recall of Fusing JTDD from Multiple Layers together on Penn Action Dataset

Figure 4.5 shows that combining JTDDs from multiple layers almost likely improves recognition results. When JTDDs from conv5b and conv4b are used together, recognition performance is much improved. Recall improves as more qualities that complement one another are fused together.

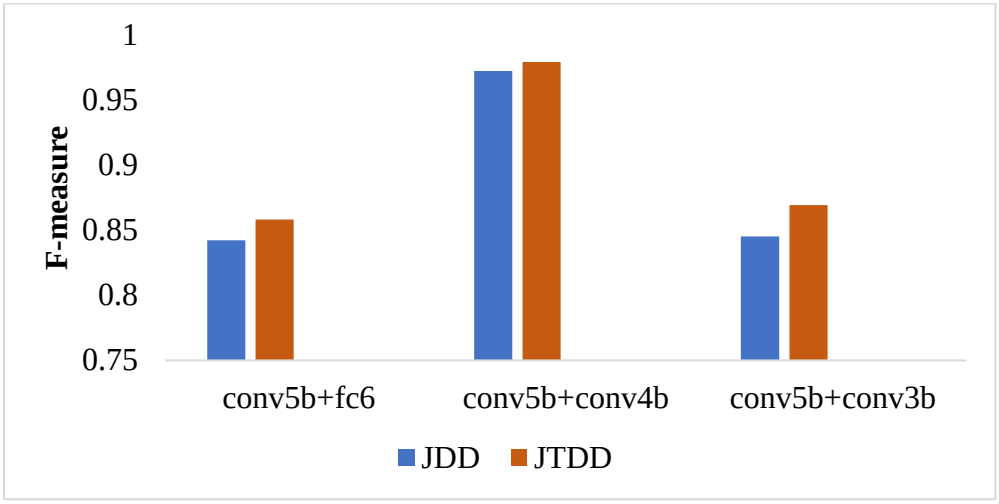


Figure. 4.6 Recognition F-measure of Fusing JTDD from Multiple Layers together on Penn Action Dataset

Figure 4.6 shows that combining JTDDs from multiple layers almost likely improves recognition results. When JTDDs from conv5b and conv4b are used together, recognition performance is much improved. F-measure improves as more qualities that complement one another are fused together.

Table 4.4 and Figure 4.3 show the results of comparing the use of estimated versus ground-truth body joints and trajectory points for evaluating JDD and JTDD on the Penn Action dataset.

TABLE 4.4. Impact of Estimated Body Joints + Trajectories Versus Ground-Truth Body Joints + Trajectories for JDD and JTDD On Penn Action Dataset

Method	GT	Estimated	Difference
JDD $(conv5b)$	0.819	0.777	0.042
JTDD $(conv5b)$	0.835	0.810	0.025

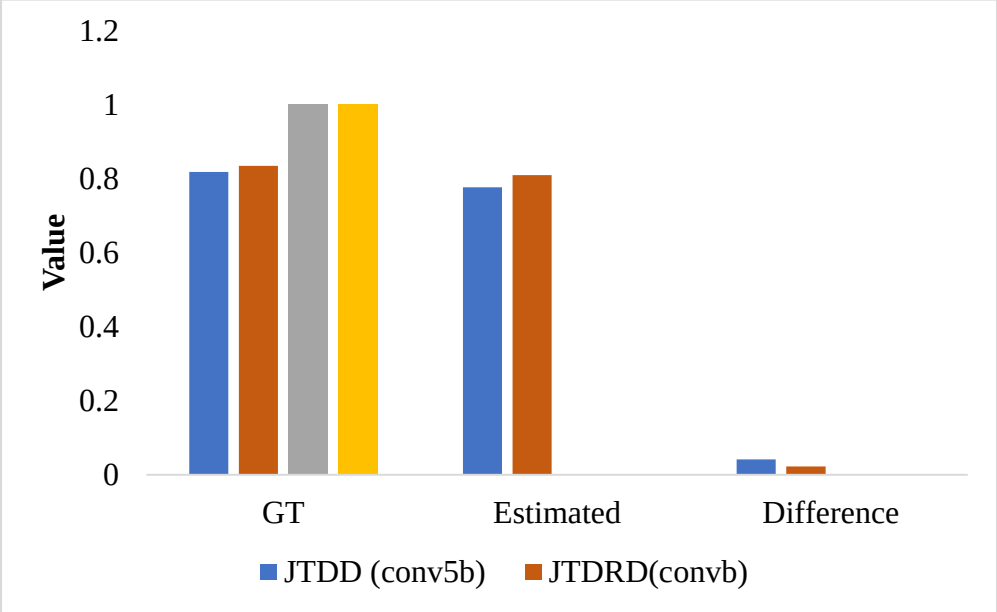


Figure. 4.7 Impact of Estimated Body Joints + Trajectories versus Ground-Truth Body Joints + Trajectories for JDD and JTDD on Penn Action Dataset

Figure 4.7 illustrates JTDD's top performance on the Penn Action Dataset compared to other methods. When using both ground-truth joints and points and anticipated body joints and trajectory points, JTDD improves on JDD by roughly 10%.

4.3 CHAPTER SUMMARY

Using the two-stream bilinear model, i.e. JTDD, this chapter optimizes HAR by extracting both body joints with the optical flow. In this approach, a C3D network is fed two streams of data, and a bilinear product is used to extract a trajectory point for each body joint location. As a result, the spatiotemporal features of video sequences can be created jointly using pooled descriptors. The class label from the bilinear C3D model with two input streams is then combined with the video descriptors in a full-network training approach. In order to classify the video descriptors used in HAR, the linear support vector machine is employed. Experiments demonstrate that the suggested JTDD model's recognition accuracy improves to 0.987 on the Penn Action dataset by merging JTDDs from conv5b and conv4b with GT body joints and trajectory points.