Journal of AI and Data Mining
Vol 8, No 2, 2020, 177-188.
DOI: 10.22044/JADM.2019.7903.1929

Hand Gesture Recognition from RGB-D Data using 2D and 3D Convolutional Neural Networks: a comparative study

M. Kurmanji and F. Ghaderi*

Human Computer Interaction Lab., Electrical and Computer Engineering Department, Tarbiat Modares University, Tehran, Iran.

Received 26 December 2018; Revised 09 April 2019; Accepted 16 June 2019
**Corresponding author: fghaderi@modares.ac.ir (F. Ghaderi).*

Abstract
Despite considerable enhances in recognizing hand gestures from still images, there are still many challenges in the classification of hand gestures in videos. The latter comes with more challenges including higher computational complexity and arduous task of representing temporal features. Hand movement dynamics, represented by temporal features, have to be extracted by analyzing the total frames of a video. So far, both the 2D and 3D convolutional neural networks (CNNs) have been used to manipulate the temporal dynamics of the video frames. 3D CNNs can extract the changes in the consecutive frames and tend to be more suitable for the video classification task; however, they usually need more time. On the other hand, using techniques like tiling, it is possible to aggregate all the frames in a single matrix and preserve the temporal and spatial features. This way, using 2D CNNs, which are inherently simpler than 3D CNNs, can be used to classify the video instances. In this paper, we compare the application of 2D and 3D CNNs for representing temporal features and classifying hand gesture sequences. Additionally, providing a two-stage two-stream architecture, we efficiently combined color and depth modalities and 2D and 3D CNN predictions. The effects of different types of augmentation techniques are also investigated. The results obtained confirm that an appropriate usage of 2D CNNs outperforms a 3D CNN implementation in this task.

Keywords: *Convolutional Neural Networks, Deep Learning, Hand Gesture Recognition, Video Classification.*

1. Introduction

As a new approach to design human computer interfaces, hand gesture recognition (HGR) has attracted the attention of many researchers in the recent years. Some potential applications include touch-less control for TVs and computers, robot control, augmented and virtual reality systems, and sign language translation [1]. In general, hand gestures are categorized into static and dynamic poses of human hand. Dynamic gestures are those that run continuously over time [2]. Recognition of hand gestures, especially dynamic gestures, can be a challenging task due to the followings reasons: cluttered background, tiny finger movements, fast hand movements, different hand shapes and skin colors, different illumination settings, and high number of gestures [3-5]. The last challenge would dramatically increase the training time and decrease the recognition accuracy in a learning system [6,7].

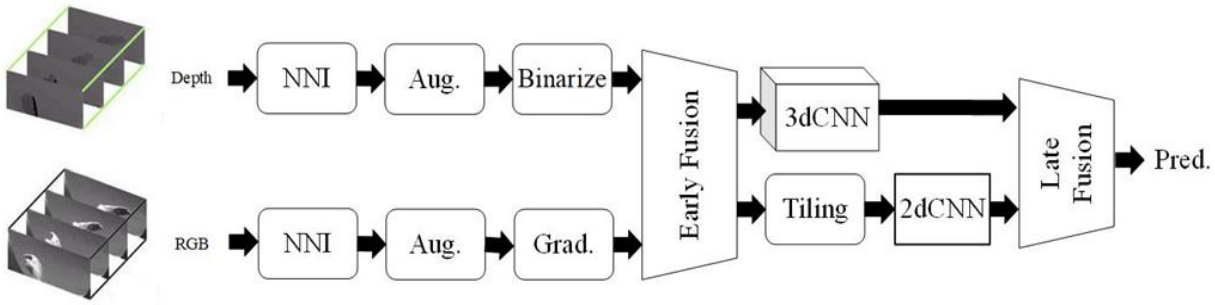


Figure 1. Block diagram of the proposed method. Input is the RGB and depth modalities of hand gesture videos, and the output is the predicted label. The following modules are used in the framework. NNI: Nearest Neighbor Interpolation, to fix video lengths; Aug: Data Augmentation. Binarize layer generates a binary image from depth data. Grad. Layer calculates Gradient from intensity channel of color data using Canny operator, Tiling, generates a 2D pattern from sequence of video

Many efforts have been made in the last two decades to overcome these challenges and improve HGR systems. These efforts can be divided into two general categories. Methods in the first group use wearable devices to track hand parts, while the methods in the second group only use the computer vision techniques [7]. Wearable devices can improve segmentation of hand in pictures and also help to better recognize dynamics of hands. This will lead to an absolute and fast feature extraction. However, due to the lack of convenience, these class of methods are not popular [8].
In the recent years, many vision-based methods have been proposed. In [9], a number of descriptors for joints are defined that represent hand state, and the aim is to learn these descriptors. Different feature vectors are used in these methods. In [10], a specific gesture is defined through a set of special-temporal descriptors. Elmezian et al. [11] extracted a feature vector, which was composed of geometric and physical characteristics of hand motions. Hosseini and Hassanian in [12] used a motion detection approach together with a mean shift method to track hand and recognize gestures in sign language. The SIFT and SUFT algorithms are also widely used for extracting the key points of frames. Dandu et al. [13] have used Local Histogram Features Descriptor, being also popular in other fields of image processing. To learn these feature vectors, many machine learning algorithms have been used, from which HMM [14], artificial neural networks [15], and SVM [16] are the most popular ones. Nachiket et al. [17] explored Hidden Markov models as an approach for modeling and recognizing dynamic hand gestures. They used two types of features, HOG and CNN, to train Markov models و and resulted that CNN features better represented hand shape features than HOG features.
In the last few years, deep learning algorithms, and in particular, Convolutional Neural Networks (CNNs), have revolutionized the signal processing field. Their most significant usage was in image processing problems. CNNs extract valuable features through consecutive convolution and max-pooling layers. These features can better represent images compared to the hand-crafted features in traditional methods, and are very useful in various tasks. Mahmoudi et al. [18] perfectly used CNN-based features in a multi-target tracking problem for pedestrian tracking. A successful example is GoogleNet, a very deep multi-column architecture [19], which yielded unprecedented results in ImageNet challenge.
In order to reduce the training time, the researchers suggested shallow CNNs, e.g. AlexNet [20]. Despite the achieved saving in time, this shallow networks yield reasonable results in classifications tasks. Convolutional neural networks are also useful for video classification tasks and the most popular architecture for this application is 3D CNN [21]. Zhi et al. investigated 3D sequences to learn spatio-temporal features of human actions from depth sequences [22]. They used features from joint descriptions together with features from 3D CNN to do a classification of human actions using a SVM classifier. Necati et al. [23] used a multi-stream 3D CNN fused with a majority voting approach to predict human gestures from raw video data.
On the other hand, using depth modality together with color data, one can improve the performance of the HGR systems. Many efforts to fuse these modalities and take advantage of the information contents of both data channels have been reported [24] [25] [26]. Moeini et al. [27], after a process of extracting features, fused texture and depth feature vectors to train an SVM. Saba et al. [28] provided rich features by segmenting hand shape in images to learning algorithms. The authors in [29] have effectively joined depth and color data through a deep convolutional neural network to detect

20 gestures of Italian sign language. In [30], the authors detected body gestures by learning joint representations through a multi-stream multi-scale CNN. Malchanov et al. [26] ranked first in the VIVA hand gesture recognition challenge using a two-stream 3D CNN. Each stream had a different resolution. Multi-stream architectures have shown promising outcomes, especially in video classifications. This class of architectures helps to find more features in videos and create more stable models. In [31], different types of feature fusion in multi-stream architectures have been investigated. The most used fusion types are late fusion, early fusion, and slow fusion.

Dynamic hand gesture recognition is a special case of video classification. An important issue to be considered when analyzing dynamic gestures is the inclusion of temporal evolution of the gestures. Temporal features represent hand dynamics, and are substantial to recognize hand movements in video. It seems that 3D convolutional filters are suitable for extracting these features [26] by doing convolution along time axis. However, these filters may not perform well on videos with tiny movements of objects, e.g. hand gesture videos. Furthermore, 3D convolutions are time-consuming. Given a 2D filter of size $m \times n$, the complexity of applying this filter to an image of size $W \times H$ is of order $O(W \times H \times m \times n)$. For a 3D input image of size $m \times n \times t$ and 3D filter of size $W \times H \times L$, convolution operation is of order $O(W \times H \times L \times m \times n \times t)$.Researchers also exploited Deep Recurrent Neural Networks to deal with temporal features in hand gesture recognition tasks [32] [33]. Ce Li et al. [34] integrated a bidirectional LSTM network and a bidirectional Gated Recurrent network with Fisher criterion to create discriminative models to detect hand gestures. Vijay et al. [35] used a decovolutional neural network to select representative frames from multi-frames of a hand gesture video and then trained a long-term recurrent network using these representative frames. Malchanov et al. [36] created a recurrent neural network with 3D CNNs to classify hand gestures. They used a connectionist temporal classification to train their model and fuse multiple RRNs.

In this paper, we compared the effectiveness of learning temporal representation of hand gestures using 2D and 3D CNNs. To this end, a pipeline for processing depth and color data was processed. This processing includes several steps to overcome overfitting and illumination condition. The color and depth stream are then combined into a new data frame carrying information from both modalities. 2D CNNs that performed well in image classification problem are also used in our proposed method for classification task. The results obtained show that this method improves recognition accuracy and also decreases train and recognition times compared to the 3D models. To keep advantages of both sides, we included another step of fusion by which we combined results of 2D and 3D models in parallel to provide final predictions.

The proposed pipeline includes preprocessing steps such as large scale data augmentation. Moreover, Canny operator and normalization techniques are applied to enhance the prediction quality.

The contributions of this work are as follow:

a. We showed that 2D CNNs could perform better than 3D CNNs for video classification of hand gestures in terms of classification performance and time complexity.
b. We introduced a new video representation by fusing color and depth modalities and making it suitable for training regular 2D CNNs.

The structure of this paper is as what follows. In Section 0 we describe the classification framework and the details of the proposed framework for video classification. In Sections 2 and 3 the proposed structure is validated and the results are reported. The concluding discussions are presented in Section 0.

2. Proposed method

The block diagram of Figure 1 illustrates the process flow of the proposed framework. The inputs are color and depth modalities of hand movement videos, and the objective is to assign a label to each input sample specifying the hand gesture class. The learning framework has two fusion stages, i.e. early fusion and late fusion. The details of the stages are as what follows:

A. Early fusion stage:

The purpose of the early fusion stage is to combine depth and color modalities. This stage has two separate flows for each data modality. The input to the upper flow in Figure 1 is *n* frames of depth data relating to video sample s_i. These frames could be gray scale images, as in the case of VIVA dataset or depth maps. The input to the lower flow is *n* frames of color data relating to sample s_i.

For the sake of increasing performance, some pre-processing steps were applied to the images before merging them. The pre-processing steps includes **a**: normalization of the video length, **b**: data augmentation, **c**: calculating gradient of color images, and **d**: binarizing depth images. For datasets that lack

the depth modality, we simply ignore depth processing flow and early fusion. The details of the steps are as follows:

a: Since we used well-known CNNs with pre-defined structures, the lengths of all videos were fixed to 32 frames for VIVA dataset and 16 frames for Cambridge HGD. We fixed video lengths using the Nearest Neighbors Interpolation.

b: In this research work, several data augmentation techniques are used, i.e. 1) reversing video frames, 2) translation, 3) rotation, 4) random crop, 5) fixed-pattern drop [26], 6) random-pattern drop [26]. For translation augmentation, we translated all frames of video ±8 pixels horizontally and ±6 pixels vertically. For rotation augmentation, we rotated videos by ±10 degrees around the central point. For random crop, we used two random windows of different sizes to crop videos. The drop augmentation (5 and 6 above) is such that $p = 50\%$ of pixels of each frame is randomly dropped to zero. In fixed-pattern drop, the pixels are randomly selected first and their values are dropped to zero for all frames of the video, while in random-pattern drop augmentation, the dropped pixels are randomly selected for each individual frame. Figure 2 illustrates augmentation of one single frame in a video. For the sake of more convenient evaluation and considering the effects of different augmentation methods on the results, we divided the augmentation methods into two groups, namely 1 to 3 (of the above augmentation methods) into one group and the rest into the second group. We call the first group Aug1 and the second one Aug2.

c: we used gradient of image sequences. We computed contours of hand shape from intensity channel using Canny operator of size 3×3 pixel with $\sigma = 0.33$. Afterwards, for each frame *i,* we subtracted *i*th *frame* from frame, where $(i + d) < L$, and L is the length of the video. These two will make the model more robust against different illumination conditions and also cluttered background. It is worth mentioning that we did not perform any normalization on image channels.

d: In the case of depth modality presence, we mapped the depth information to a binary image by applying a threshold to the depth images.

We use the RGB data format to merge the depth and color images into a single structure by placing intensity channel of the color image and the binarized image of depth in the first and second channels, respectively. The third channel is filled with the indices of the frames in the frame sequence. As far as we know, it is the first time that depth and color modalities are merged in this way.

If we define an input as $W \times H \times n \times C$, where *W*, *H*, *n* and *C* are width, height, number of frames, and number of channels respectively, in the case of VIVA dataset, the input to the depth and color flows are of size $57\times125\times n\times1$, where n varies for different video samples, and the inputs to early fusion layer are of size $32\times32\times32\times1$ and the output of this layer is of size $32\times32\times32\times3$. In the case of Cambridge dataset, the input to color flow is of size $20\times20\times20\times3$ and the output after preprocessing is of size $20\times20\times16\times3$. As the CNN models we used in these paper expect 3 channels for input images, for Cambridge dataset that lacks depth information, we used the gradient value of each pixel of the frame in the 2nd channel of the RGB data structure.

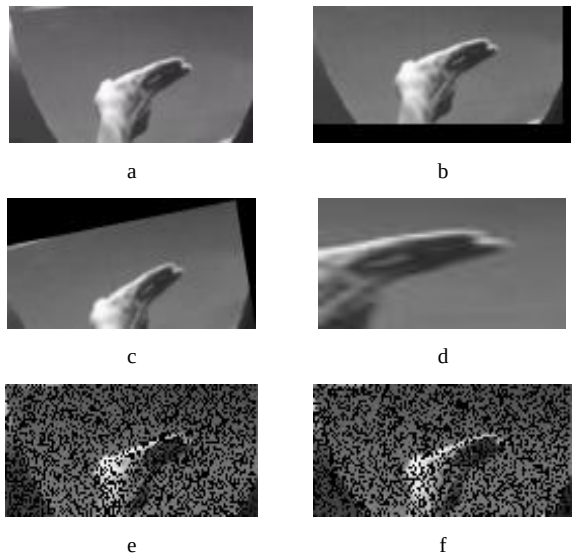
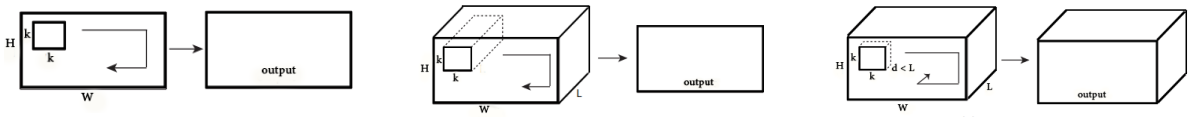


Figure 2. Data augmentation, a) original image, b) translation, c) rotation, d) random crop, e) fixed-pattern drop, and f) random-pattern drop.

B. Late fusion stage:

The purpose of the late fusion stage is to exploit the extracted features from both the 2D and 3D CNNs. In the case of dynamic hand gestures, which sometimes include small finger movements, retaining the precise ordering of these movements is necessary to achieve a correct prediction. 3D networks learn temporal features by performing convolution and max-pooling operations in the direction of the time axis. Figure 3 shows the difference between 2D and 3D convolutional filters. As it can be seen in this figure, applying a 2D filter to a single frame (Figure 3.a) or a 3D filter with the depth equal to the number of stacked frames on all the frames (Figure 3.b.), would result in a 2D output.



a b c

Figure 3. Graphical illustration of the effects of 2D and 3D convolutions on 2D (images) and 3D (videos) input data. a) 2D input and 2D convolution b) 3D input and 2D *convolution* c) 3D input and 3D convolution [21]. W, H, and L are width, height and length (number of frames) of the input data, respectively. Size of the 2D Convolutional filter is k×k and the 3D filter is of size k×k×d, where d corresponds to the depth of the filter.

That means 2D filters discard the temporal information at the first layer [21]. In this type of networks, the output value corresponding to the point (x, y) in the *j*th feature map at the *i*th layer of network can be calculated as follows [37]:

$$v_{(i,j)}^{(x,y)} = tanh(b_{ij} + \sum_{m} \sum_{p=0}^{P_i-1} \sum_{q=0}^{Q_j-1} w_{ijm}^{pq} v_{(i-1)m}^{(x+p)(y+q)}) \quad (1)$$

where, $tanh(\cdot)$ is the hyperbolic tangent used as the activation function and b_{ij} is the bias corresponding to the *j*th feature map at *i*th layer. *m* indicates the feature map in the previous layer ($(i-1)th$ layer), which is connected to the current (*j*th) feature map. Convolution Filter at the *i*th layer is of size $P_i \times Q_i$, and W_{ijm}^{pq} is the weight of the filter maps at layer i connected to the point (p, q). In 3D networks, however, given an input of shape $H \times W \times C \times L$, where *W*, *H*, *C* and *L* are width, height, number of channels, and length of the video (in terms of frames), respectively, by applying a 3D convolution filter with depth of *l*, where $l < L$, the output will be a 3D volume conveying temporal features on 3rd dimension, as shown in Figure 3.c. Therefore, applying 3D max-pooling and 3D convolutional filters will retain temporal information at the subsequent layers. In this case, the output volume at *i*th layer and *j*th feature map corresponding to the point (x, y, z) will be as follows [37]:

$$v_{(i.j)}^{(x,y,z)} = tanh(b_{ij} + \sum_{m} \sum_{p=1}^{P_i-1} \sum_{q=0}^{Q_i-1} \sum_{r=0}^{R_i-1} w_{ijm}^{pqr} v_{(i-1)m}^{(x+p)(y+q)(z+r)}) \quad (2)$$

where, R_i is the depth of convolution filter at the *i*th layer. However, temporal features obtained by 3D convolutional filters may not be appropriate for representing a displacement. Displacement in video appears as special differences in consecutive frames. In videos with shorter lengths, just like hand gestures, there are very tiny displacements in the subsequent frames, and this makes it difficult to detect displacements for temporal filters.

Considering the pros and cons of 2D and 3D convolution networks for our application, two different approaches were used in the final classification stage.

3D CNN: as discussed and can be seen in Figure 3.c, 3D networks can extract temporal features by sliding convolution filter, across the time axis. In the case of hand gesture videos, this feature would be useful especially for those gestures with simple hand movement.

2D CNN: Considering the high computational cost of 3D CNN from one hand and the promising performance of the 2D CNN architectures, we also used the latter in our framework. In order to use these networks, we had to change the structure of the input data and tile the frames to make a single 2D frame. In the next sub-section, we describe the details of the proposed method to use 2D CNNs for classification of hand gesture videos.

Video classification using 2D CNNs:

In our proposed method, representation of videos that include temporal data is learned using famous 2D CNNs like GoogleNet [19] and AlexNet [20]. We use a 2D pattern to represent a video. This pattern is so that all frames of video keep together in a tiled, ordered, and non-overlapping pattern to form a single image. To create such a tiled pattern, for every video of size *L* frames with each frame of size $m \times n$, frames are arranged in a row-major manner to form a matrix of size $w \times m \times \frac{L}{w} \times n$, where, every $w < L$ frames make one row. The mentioned parameters should be set in a way that satisfies restrictions of the used 2D CNN. In the constructed matrix, the element (frame) at location (i, j) is the $[(i-1) \times w \times m + j]th$ frame in the video. An important issue to be considered when concatenating the frames is the effect of discontinuity of boundary pixels in the adjacent frames on the convolved images. This phenomenon causes two

problems. firstly, areas near the common border of any two adjacent frames in the tiled pattern may have different colors or intensities. These differences are mis-leading for convolutional filters and make them learn the differences as a new edge or pattern while these regions are not logically relevant. Moreover, in some cases, two adjacent frames are not consecutive in time (for example, the *j*th frame in the *i*th row and the *j*th frame in the $(i+1)th$ row). To avoid these two problems, we simply added a padding area around any frame before putting it in the tiled pattern. The size of the padding must be greater or equal to the half size of the largest filter in the first layer of CNN. Figure 4 shows a tiled pattern for a video of hand gesture. The constructing frames of this image have been generated by the Binarized depth frames.

Figure 4. A sample tiled pattern. Each row presents a sub-gesture. There is a zero padding between each two adjacent rows.

In 3D cases, convolution filters first move in the time axis direction and then move horizontally or vertically. Thus a filter looks at a 2D sub-space of video and traces its changes in time and repeats this to cover the whole video, while in the 2D cases, convolution filters move on the images in a row-wise manner. On the other hand, when we do tile mapping on video frames, every gesture is turned to L/w sub-gestures (every w gesture in one row). Thus the convolutional filters see sub-gestures that are temporally ordered. In this method, temporal data appears as ordered patterns in frames. In fact, time component is mapped into a 2D space.

C. Classification

We used the predictions of both 3D and 2D networks to improve the accuracy. For this, we trained two CNNs in parallel and separately. We optimized a separate loss function for each one of these networks, and while predicting labels of one video, we used element-wise product of the predictions as the final prediction. Given $\vec{P}_{2d}(C \mid x)$ and $\vec{P}_{3d}(C \mid x)$ as output predictions of 2D CNN and 3D CNN, respectively. Equations (3) to (6) show how we fused them. This fusion happens in the Late Fusion Layer as in Figure 1. This is the last layer of model that outputs predictions. $\vec{P}_{new}(C \mid x)$ is the new prediction vector:

$$P_{2d}(C=i \mid \mathbf{x},\mathbf{W},\mathbf{b}) = softmax(\mathbf{W}.\mathbf{x}+\mathbf{b}) = \frac{e^{(\mathbf{w}_i\mathbf{x}+\mathbf{b}_i)}}{\sum_j e^{(\mathbf{w}_j\mathbf{x}+\mathbf{b}_j)}} \quad for\ i=1,\mathrm{K}\ ,k \tag{3}$$

$$p_{3d}(C=i \mid \mathbf{x},\mathbf{W},\mathbf{b}) = softmax(\mathbf{W}.\mathbf{x}+\mathbf{b}) = \frac{e^{(\mathbf{w}_i\mathbf{x}+\mathbf{b}_i)}}{\sum_j e^{(\mathbf{w}_j\mathbf{x}+\mathbf{b}_j)}} \quad for\ i=1,\mathrm{K}\ ,k \tag{4}$$

$$p_{new}(C=i \mid \mathbf{x},\mathbf{W},\mathbf{b}) = p_{3d}(C=i \mid \mathbf{x},\mathbf{W},\mathbf{b}) \otimes p_{2d}(C=i \mid \mathbf{x},\mathbf{W},\mathbf{b}) \quad for\ i=1,\mathrm{K}\ ,k \tag{5}$$

$$C_{predict} = \mathrm{argmax}_i P(C=i \mid \mathbf{x},\mathbf{w},\mathbf{b}) \quad for\ i=1,\mathrm{K}\ ,k \tag{6}$$

where, $\otimes$ is the element-wise product, *k* is the number of labels. $\boldsymbol{x}$ is the input video, and *C* represents different possible *k* classes. $\boldsymbol{W}$ and $\boldsymbol{b}$ are weights and biases of the last layer, respectively. The probability that the input video is classified as class *i* is returned by $p[i]$.
In order to use the full advantages of 2D networks, we used two well-known CNNs, AlexNet and GoogleNet. These two networks are trained with the same configuration, summarized in Table 1. It is worth mentioning that here, the 2nd version of GoogleNet, i.e. inception_v2, is used. The optimization procedure consists of updating network weights $\boldsymbol{W}$ through minimizing a cost function over a dataset D. To this end, CrossEntropy cost function is used.

$$E = \frac{-1}{|D|}\sum_{n=1}^{|D|}\log(p_n \mid l_n, \mathbf{w}) \tag{7}$$

where, l_n is the true label of sample *n*. Optimization is done through stochastic gradient descent on mini-batches of size 30. Here, we used Nesterov accelerated gradients (NAGs) to update network weights $\omega \in W$ at each iteration using (8).

$$\nabla\omega_i \left\langle \frac{\delta L}{\delta w_i - 1} \right\rangle_{batch}, v_i = \mu v_{i-1} - \lambda w_i, w_i = w_{i-1} - \lambda \nabla w_i \tag{8}$$

where, λ is the learning rate, μ is the momentum, ∇w_i is the average gradient of cost function over all samples in a batch, and v_i is the new weight. All the convolutional layers of CNNs are initialized with *Xavier* with a variance of 0.001. For each layer, *Xavier*

selects values from a zero-mean Gaussian distribution with a fixed variance. For 3D models, C3D and Low Resolution Network (LRN) [26] models were used. Again, these two networks are trained under the same configuration, as in Table 2. In order to avoid overfitting, we exploited two techniques in all models: drop-out for the last fully connected layers of CNNs and data augmentation.

2.Experiments

D. Datasets

For evaluating the proposed method, we used two different datasets: VIVA HGD [6] and Cambridge HGD [38]. These datasets have two challenging attributes in common. First, gestures are performed by different people. Secondly, the sample videos are recorded under different illumination conditions. This enables us to evaluate the robustness of the trained method against subject and illumination variations.

Table 1. Configuration of 2D CNN's. Moment: momentum, acc: accumulation, lr: learning rate. O: Optimizer, Es: Epoc step, Mo: Moment, Bs: Batch size, Bc: Batch Acc, Nes: Nesterov

Max epochs	O	Es.	Mo.	Bc.	Bs.	Lr.
200	Nes.	10	0.9	2	25	10^{-2}

Table 2. Configuration of 3D networks.

Max epochs	O	Es.	M	Bc.	Bs.	Lr
200	SGD	10	0.9	2	25	10^{-3}

Table 3. Mean and standard deviation of accuracy and recall. T1 and T2 represent the training and testing times, respectively. std: standard deviation.

CNN type	VIVA Dataset					
	Accuracy	std	Recall	std	T1 (min)	T2 (sec
GoogleNet	72.5	6.1	66.5	6.5	480	5
AlexNet	70.5	7	63.5	7	155	3.5
	Cambridge Dataset					
GoogleNet	77	2.3	69	2.4	365	5
AlexNet	75	2.5	65	2.5	102	3.5

VIVA HGD is a video dataset of hand gestures. This dataset is designed and captured in order to facilitate research work for automatic interpretation of manual control commands issued by people inside Vehicles. This dataset consists of 32 different gestures, among which 19 gestures are considered for this challenge. Gestures are performed by 8 different subjects while sitting on the driver or the front passenger seats. Each one of the subjects performed their gestures under different illumination conditions. Videos of hand gestures are grayscale, and there is a depth video corresponding to each gesture. This dataset consists of 885 samples of hand gestures including color and depth measurements for each.

Cambridge HGD consists of 900 videos of 9 different gestures performed by 2 different subjects. These gestures are a combination of 3 primitive hand movements and 3 basic postures. Movements are labeled rightward, leftward, and contract and postures are labeled flat, spread, and V-shape. Unlike VIVA HGD, this dataset does not contain depth information. Gestures are performed under 5 different illumination conditions, and images are in the RGB format.

E. Experimental setup

We used CAFFE framework for our modeling. For 2D Networks, we used the main version of CAFFE [39], and for 3D Networks, we used VIDEO_CAFFE [40] and C3D CAFFE [21]. All libraries were used on a machine with an Intel Core i7 CPU, 16 GB RAM and Ubuntu 14.04 OS installed. We trained our networks on a NVIDIA GTX 1070 GPU.

3. Results

For evaluation, we used leave-one-subject-out-cross-validation. This way, video samples of one subject would be kept for test, while the other subject's samples are used to train models. This has been repeated for all subjects and mean, and standard deviation of accuracies are reported.

At first, we evaluated the 2D structure. For this purpose, the 3D and the late fusion modules are discarded from the framework. Considering the fact that the Cambridge HGD dataset lacks the depth information, we just used color data, and hence, the depth stream of Figure 1 is not included. Table 3 shows the results for 2D Networks. For both datasets, GoogleNet performed better than AlexNet by 2% in accuracy. On the other hand, AlexNet requires less train and test time due to its shallower architecture. The AlexNet and GoogleNet convergence times for the VIVA dataset were 155 and 480 minutes, respectively.

Table 4 shows the effect of the data augmentation on the performance of the framework. The results obtained are reported for both 2D CNNs. The purpose is to see the effect of increasing number of data samples by augmentation on the model accuracy.

Table 4. . Mean and standard deviation of accuracy and recall. T1 and T2 represent the training and testing times, respectively. std: standard deviation.

CNN type	Accuracy	std	Recall	std	T1 (min)	T2 (sec)
			VIVA Dataset			
GoogleNet	72.5	6.1	66.5	6.5	480	5
AlexNet	70.5	7	63.5	7	155	3.5
			Cambridge Dataset			
GoogleNet	77	2.3	69	2.4	365	5
AlexNet	75	2.5	65	2.5	102	3.5

Table 5. Effect of data augmentation on classification accuracy. '-' Means that none of the data augmentation functions are applied. Aug1 includes frame reversing, translation, and rotation. Aug2 includes random crop, fixed-pattern drop, and random-pattern drop. '+' means that both augmentations are applied simultaneously.

CNN type	No augmentation	Aug1	Aug2	Aug1+Aug2
		VIVA Dataset		
GoogleNet	54.4	63.1	68.8	72.5
AlexNet	54	60.3	67.9	70.5
		Cambridge Dataset		
GoogleNet	63.6	69.9	73.5	77
AlexNet	62.8	67.6	71.4	75

Table 6. Effect of gradient and differentiation layer on VIVA dataset.

Network	Measure	-	Grad	Diff	Grad+Diff
GoogleNet	Accuracy	63.5	70	66	72.5
AlexNet		60.6	69	64.1	70.5
GoogleNet	Recall	50	64.2	53	66.5
AlexNet		49	59.8	53	63.5

Table 7. Comparison between 2D model and 3D model results. t1 and t2 represent the training and testing times

CNN type	Accuracy	Recall	T1(min)	T2(sec)
		VIVA Dataset		
GoogleNet	72.5	66.5	480	5
AlexNet	70.5	63.5	155	3.5
C3D	68	63	560	5.5
LRN	69	64	490	4
		Cambridge Dataset		
GoogleNet	77	69	365	5
AlexNet	75	65	102	3.5
C3D	72	66	569	5.5
LRN	70	65.6	404	4

Moreover, with grouping of the augmentation methods, we can compare the methods that alter pixel values (Aug2) with those that just shift pixels without alteration (Aug1). As it can be seen in this Table 5, the outcome of GoogleNet is more affected by data augmentation compared with that of AlexNet. By applying augmentation, GoogleNet and AlexNet accuracies, respectively, improved 18.1% and 16.5% on the VIVA dataset. In addition, the results show that in both cases, Aug2 methods have a major effect on accuracy compared with the Aug1 methods.

The next evaluation was on the effect of the gradient and differentiation layer. As we mentioned earlier, this layer is added to make model more robust against illumination conditions and also for background removal.

As shown in Table 6, this layer has a considerable effect on the model in terms of accuracy and recall. The illumination condition is more realistic and challenging in the VIVA dataset, and the background of the videos is noisier. Moreover, it contains more distinct illumination types than the Cambridge dataset. So VIVA dataset is sufficient to evaluate the performance of this layer. Table 7 compares the results of the absence and presence of this layer.

One adjustable parameter of this layer is the distance between the frames that are differentiated. Figure 5 illustrates the diagram of the distance (d) versus mean accuracy. The best results are achieved with $d = 1$ and $d = 2$. As space increases, the logical connection between the frames decreases.

In the first fusion stage (i.e. early fusion), we merged the RGB-D data into *a* new data frame that carry information from both modalities as well as temporal order of frame sequences.

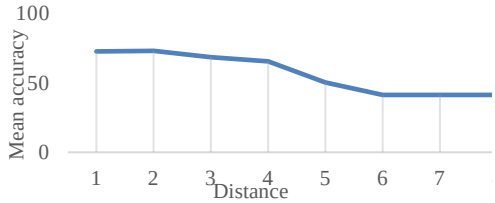


Figure 5. Mean accuracy versus *the* distance between the frames in differentiation layer.

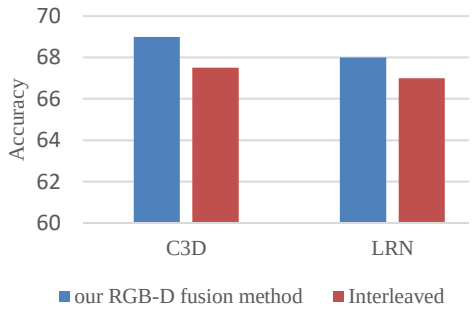


Figure 6 comparing our *early* fusion method vs. interleaved RGB-D fusion introduced in [20].

The early and late Fusion types have been well-studied and compared in the literature. Here, we compared the results of our method with another early fusion scheme used in [20]. They stacked depth frames among color frames after normalization and before feeding to network. Figure 6 shows the results on two 3D CNNs. As the results declare, in both cases, our merging method performs better.

The proposed method enables us to exploit high-performance 2D CNNs for video classification. These networks are well-designed and evaluated on different datasets, and the results are reported here.

For a fair comparison between the 2D and 3D CNNs, we repeated all pre-processing steps, as explained earlier, when evaluating the 3D networks. A comparison of the results of the 2D and 3D networks is reported in Table 7. According to these results, 2D networks outperform 3D models in both accuracy and time.

The proposed method uses both the 2D and 3D networks to obtain better predictions. As illustrated in Figure 1, final model is a two-stream architecture which fuses predictions of 2D and 3D networks. The data entered into these networks has passed through the same preprocessing modules. Table 8 shows the results of the model on both datasets.

Table 8. Results for the final two-stream method.

Network type	Accuracy	Recall
	VIVA Dataset	
GooleNet C3D	74.5	68
GoogleNet LRN	75	68.5
AlexNet C3D	71.7	64.9
AlexNet LRN	72.2	65
	Cambridge Dataset	
GoogleNet C3D	78.5	71.5
GoogleNet LRN	78	72
AlexNet C3D	75.7	67
AlexNet LRN	75.8	67.2

Table 9. Results of different methods on VIVA Hand Gesture Dataset

Method	Accuracy (inter-subject)
LRN [26]	**67**
HRN [26]	**68**
LRN + HRN [26]	**70**
GoogleNet + C3D	**74.5**

4. Comparison with other research works

We have two considerations to make a fair comparison between our results and the other works. Firstly, we compare our results only with those of the other CNN-based methods. There are some research works that combine different techniques, such as RNNs, in order to extract temporal features. However, the focus of our work is on CNNs, and therefore, such works are excluded from our comparison. Secondly, the evaluations should be inter-subject, as we have described earlier in this section. One of the most cited papers with these criteria is [26]. We re-implemented the full model of this paper with all pre-processing and post-processing. Table 9 shows the comparing of results.

5. Concluding discussions

In this work, we investigated the effects of 2D and 3D convolutional neural networks on the dynamic hand gesture recognition task. We found out that 2D CNN could outperform 3D CNNs for learning temporal representations, specifically for videos with tiny movements like hand gesture dynamics. In addition, 2D CNNs have lower time complexity compared with 3D convolutional operations.

We mapped hand gesture videos to a 2D tiled pattern of temporally-ordered non-overlapping frames. This data was used to train well-known and efficient CNNs like GoogleNet and AlexNet. We also proposed two types of fusion in this paper, i.e. an early fusion for combining depth and color modalities and a late fusion to merge predictions of 2D and 3D CNNs. We also used gradient of images and differentiated consecutive frames to make the model robust against cluttered background and illumination conditions.

VIVA and Cambridge hand gesture datasets were used to evaluate our method. Since hand gesture datasets are small with a limited number of data samples, we applied two groups of data augmentation methods on them. The augmentation methods in the first group just shift or rotate images in different directions, and the ones in the second group modify some pixels.

As it can be observed in Table 3, standard deviation of accuracies for different subjects is low for both datasets, and it shows that the model is robust against subject variation. Standard deviation for VIVA is greater than that of Cambridge dataset. This is because of two reasons: **first**, the number of subjects is much more than that of the Cambridge dataset, and **secondly**, as it has been mentioned before, in the VIVA dataset, every subject has performed gestures in a different illumination condition. Thus it is probable that the

model does not have enough samples of specific illumination criteria during training, and hence, would not learn them properly. This is not the case for the Cambridge dataset, in which videos with different illumination conditions are included for all subjects.

Moreover, the results obtained confirm that the proposed method to use 2D CNNs, outperforms the case of using 3D CNNs like C3D. According to Table 7, using the proposed framework, GoogleNet achieved an accuracy of 72.5% against C3D model with 68% and LRN with 69% accuracy on VIVA dataset. The 2D method, mapped videos into a 2D space so that temporal deviations are depicted as objects in an image. Furthermore, deviations of temporal order are considered. The results obtained show that while 3D CNNs extract temporal features through convolution over time axis, 2D method perform better for cases like dynamic hand gestures with tiny movements.

The results in Table 7 also show that, 2D CNNs are faster than 3D CNNs in both training and testing. Even GoogleNet that is a very deep network, trained about 80 minutes quicker than C3D. This is because 3D convolution is much more time-consuming than 2D convolution. In fact, as we mentioned earlier, 2D convolution time complexity is $O(W \times H \times m \times n)$, while for 3D convolution, time complexity is $O(W \times H \times L \times m \times n \times t)$.

The gradient and differentiation layer we added to our framework has a great impact on the results. This layer improved accuracy for both GoogleNet and AlexNet by roughly 10%, according to Table 6. Also the augmentation layer tuned CNNs better with more samples. The results in Table 5 showed that, those type of augmentations that alter pixel values performed better than augmentations that just shifted pixels. The outcome of GoogleNet is more affected by data augmentation than that of AlexNet. GoogleNet is a multi-column CNN, which has a much deeper architecture than AlexNet. However, the number of learnable parameters in GoogleNet is much fewer than that of AlexNet (12 times fewer parameter [19]). This means that, while GoogleNet has fewer learnable parameters because of its architecture, it is more effected by augmentation than AlexNet.

Finally, we merged both 2D and 3D networks to reinforce predictions. We got at max 2.5% improvement in accuracy by merging 2D and 3D CNNs predictions compared to their separate predictions. Our final structure that is a two-stage two-stream architecture improved accuracy of recognition on Cambridge HGD by %1.5 and VIVA HGD by 2.5%.

A natural direction for future work is to investigate new 2D CNNs that have been introduced recently, e.g. DenseNet and inception-v4. DenseNet is a compact 2D CNN that can improve train and test times significantly. Moreover, Inception-v4 was more optimized than version 2 of GoogleNet that we used in this work and could improve its accuracy.

References

[1] Yin, X. & Xie, M. (2001). Hand gesture segmentation, recognition and application. Computational Intelligence in IEEE International Symposium on Robotics and Automation pp. 438–443.

[2] Rautaray, S. & Agrawal, A. (2015). Vision based hand gesture recognition for human computer interaction: a survey. Artif. Intell. Rev., vol. 43, no. 1, pp. 1–54.

[3] Ren, Z., Yuan, J., Meng, J. & Zhang, Z. (2013). Robust part-based hand gesture recognition using kinect sensor, IEEE Trans. Multimed., vol. 15, no. 5, pp. 1110–1120.

[4] A. Kulshreshth, C. Zorn, & LaViola. J. (2013), Poster: Real-time markerless kinect based finger tracking and hand gesture recognition for HCI, in IEEE Symposium on 3D User Interfaces (3DUI). pp. 187–188.

[5] Hsiao, Y., Sanchez-Riera, J., Lim, T., Hua, K. & Cheng, W. (2014). LaRED: a large RGB-D extensible hand gesture dataset, in Proceedings of the 5th ACM Multimedia Systems Conference. pp. 53–58.

[6] Ohn-Bar, E. & Trivedi, M. (2014). Hand gesture recognition in real time for automotive interfaces: A multimodal vision-based approach and evaluations. IEEE Trans. Intell. Transp. Syst. vol. 15, no. 6, pp. 2368–2377.

[7] Yamashita, T. & Watasue, T. (2014). Hand posture recognition based on bottom-up structured deep convolutional neural network with curriculum learning. in IEEE International Conference on Image Processing (ICIP)., pp. 853–857.

[8] Yiyi, R., Xie, X., Li, G., & Wang, Z. (2016). Hand Gesture Recognition with Multi-Scale Weighted Histogram of Contour Direction (MSWHCD) Normalization for Wearable Applications. IEEE Trans. Circuits Syst. Video Technol.

[9] LaViola, J. (2014). An introduction to 3D gestural interfaces. ACM SIGGRAPH Courses, pp. 25.

[10] Trindade, P., Lobo, J. & Barreto, J. (2012). Hand gesture recognition using color and depth images enhanced with hand angular pose data. IEEE Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI), pp. 71–76.

[11] Elmezain, M., Al-Hamadi, A. & Michaelis, B. (2009) Hand gesture recognition based on combined features extraction. World Acad. Sci. Eng. Technol., vol. 60, p. 395.

[12] Hosseini, M. & Hassanian, J. (2013). Applying mean shift and motion detection approaches to hand tracking in sign language. Journal of. AI Data Min. JAIDM, vol. 2, no. 1, pp. 15–24.

[13] Reddy, D., Sahoo, J. & Ari, S. (2018) Hand Gesture Recognition Using Local Histogram Feature Descriptor. 2nd International Conference on Trends in Electronics and Informatics (ICOEI). pp. 199-203.

[14] Auephanwiriyakul, S., Phitakwinai, S., Suttapak, W., Chanda, P., & Theera-Umpon, N. (2013). Thai sign language translation using scale invariant feature transform and hidden markov models. Pattern Recognition Letters, 34(11), 1291-1298.

[15] Murthy, G. R. S., & Jadon, R. S. (2010). Hand gesture recognition using neural networks. In 2010 IEEE 2nd International Advance Computing Conference (IACC). pp. 134-138.

[16] Ren, Y., & Zhang, F. (2009). Hand gesture recognition based on MEB-SVM. In 2009 International Conference on Embedded Software and Systems. pp. 344-349.

[17] Deo, N., Rangesh, A., & Trivedi, M. (2016). In-vehicle hand gesture recognition using hidden markov models. In 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC). pp. 2179-2184.

[18] Mahmoudi, N., Ahadi, S. M., & Rahmati, M. (2019). Multi-target tracking using CNN-based features: CNNMTT. Multimedia Tools and Applications, 78(6), 7077-7096

[19] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1-9.

[20] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems. pp. 1097-1105.

[21] Tran, D., Bourdev, L., Fergus, R., Torresani, L., & Paluri, M. (2015). Learning spatiotemporal features with 3d convolutional networks. In Proceedings of the IEEE international conference on computer vision. pp. 4489-4497.

[22] Liu, Z., Zhang, C., & Tian, Y. (2016). 3D-based deep convolutional neural network for action recognition with depth sequences. Image and Vision Computing, 55, 93-100.

[23] Camgoz, N. C., Hadfield, S., Koller, O., & Bowden, R. (2016). Using convolutional 3d neural networks for user-independent continuous gesture recognition. In 2016 23rd International Conference on Pattern Recognition (ICPR). pp. 49-54.

[24] Y. Yao and Y. Fu. (2012). Real-time hand pose estimation from RGB-D sensor. In Multimedia and Expo (ICME), 2012 IEEE International Conference on, 2012, pp. 705–710.

[25] Chen, X., & Koskela, M. (2013). Online RGB-D gesture recognition with extreme learning machines. In Proceedings of the 15th ACM on International conference on multimodal interaction. pp. 467-474.

[26] Molchanov, P., Gupta, S., Kim, K., & Kautz, J. (2015). Hand gesture recognition with 3D convolutional neural networks. In Proceedings of the IEEE conference on computer vision and pattern recognition workshops. pp. 1-7.

[27]. Moeini, A., Faez, K., Sadeghi, H., & Moeini, H. (2016). 2D facial expression recognition via 3D reconstruction and feature fusion. Journal of Visual Communication and Image Representation, 35, 1-14.

[28] Jadooki, S., Mohamad, D., Saba, T., Almazyad, A. S., & Rehman, A. (2017). Fused features mining for depth-based hand gesture recognition to classify blind human communication. Neural Computing and Applications, 28(11), 3285-3294.

[29] Strezoski, G., Stojanovski, D., Dimitrovski, I., & Madjarov, G. (2016). Hand gesture recognition using deep convolutional neural networks. In International Conference on ICT Innovations pp. 49-58.

[30] Neverova, N., Wolf, C., Taylor, G. W., & Nebout, F. (2014). Multi-scale deep learning for gesture detection and localization. In European Conference on Computer Vision pp. 474-490.

[31] Yue-Hei Ng, J., Hausknecht, M., Vijayanarasimhan, S., Vinyals, O., Monga, R., & Toderici, G. (2015). Beyond short snippets: Deep networks for video classification. In Proceedings of the IEEE conference on computer vision and pattern recognition pp. 4694-4702.

[32] Murakami, K., & Taguchi, H. (1991). Gesture recognition using recurrent neural networks. In Proceedings of the SIGCHI conference on Human factors in computing systems (pp. 237-242).

[33] Maraqa, M., & Abu-Zaiter, R. (2008). Recognition of Arabic Sign Language (ArSL) using recurrent neural networks. In 2008 First International Conference on the Applications of Digital Information and Web Technologies (ICADIWT) pp. 478-481.

[34] Li, C., Xie, C., Zhang, B., Chen, C., & Han, J. (2018). Deep Fisher discriminant learning for mobile hand gesture recognition. Pattern Recognition, vol. 77, pp. 276-288.

[35] John, V., Boyali, A., Mita, S., Imanishi, M., & Sanma, N. (2016). Deep learning-based fast hand gesture recognition using representative frames. In 2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA). pp. 1-8.

[36] Molchanov, P., Yang, X., Gupta, S., Kim, K., Tyree, S., & Kautz, J. (2016). Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4207-4215.

[37] Chen, Y., Jiang, H., Li, C., Jia, X., & Ghamisi, P. (2016). Deep feature extraction and classification of hyperspectral images based on convolutional neural networks. IEEE Transactions on Geoscience and Remote Sensing, vol. 54, no. 10, pp. 6232-6251.

[38] Kim, T. K., & Cipolla, R. (2008). Canonical correlation analysis of video volume tensors for action categorization and detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 31, no. 8, pp. 1415-1428.

[39] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., ... & Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. In Proceedings of the 22nd ACM international conference on Multimedia pp. 675-678.

نشریه هوش مصنوعی و داده کاوی

تشخیص ژست دست از داده‌های RGB-D با استفاده از شبکه‌های کانولوشنی دو بعدی و سه بعدی: یک مطالعه مقایسه‌ای

مقداد کرمانجی و فواد قادری*

آزمایشگاه تعامل انسان و کامپیوتر، دانشکده برق و کامپیوتر، دانشگاه تربیت مدرس، تهران، ایران.

ارسال ۲۰۱۸/۱۲/۲۶؛ اصلاح ۲۰۱۹/۰۴/۰۹؛ پذیرش ۲۰۱۹/۰۶/۱۶

چکیده:

با وجود پیشرفت‌های چشم‌گیر در تشخیص ژست دست در تصاویر،کماکان چالش‌های زیادی در زمینه دسته‌بندی ژست‌های دست در ویدیوها وجود دارد. مورد دوم چالش‌های بیشتری مانند پیچیدگی زمانی بالاتر و دشواری نمایش ویژگی‌های زمانی را به همراه دارد. پویایی حرکات دست، که با ویژگی‌های زمانی بیان می‌شود، باید با در نظر گرفتن تمامی فریم‌های یک ویدیو استخراج شود. تا کنون، هر دو نوع شبکه‌های کانولوشنی دو بعدی و سه بعدی برای بازیابی پویایی‌های زمانی در فریم‌های یک ویدیو مورد استفاده قرار گرفته‌اند. شبکه‌های کانولوشنی سه بعدی می‌توانند تغییرات را در فریم‌های متوالی پیدا کنند و به نظر می‌رسد که برای مساله دسته‌بندی ویدیو مناسب‌تر باشند؛ با این وجود، این شبکه‌ها اکثرا به زمان بیشتری برای آموزش و تست نیاز دارند. از طرفی دیگر، با استفاده از روش‌های الحاق، می‌توان تمامی فریم‌های یک ویدیو را در یک ماتریس جمع کرد در حالی که ویژگی‌های زمانی و مکانی آن را حفظ کرد. از این طریق، می‌توان از شبکه‌های کانولوشنی دو بعدی که اساسا از مدل‌های سه بعدی ساده‌تر هستند، برای دسته‌بندی نمونه‌های ویدیویی استفاده کرد. در این مقاله، کاربرد شبکه‌های دو بعدی و سه بعدی برای نمایش ویژگی‌های زمانی و دسته‌بندی دنباله ژست‌های دست در ویدیو مقایسه شده است. علاوه بر این، با معرفی یک معماری دو مرحله‌ای دو جریانی، ماهیت‌های رنگ و عمق و همچنین نتایج پیش‌بینی از شبکه‌های دو بعدی و سه بعدی به شکل کارایی ادغام شده‌اند. همچنین تاثیر استفاده از روش‌های ازدیاد داده بررسی شده‌است. نتایج به دست آمده نشان می‌دهد که به کارگیری مناسب از شبکه‌های کانولوشنی دو بعدی در مقایسه با پیاده‌سازی‌های سه بعدی عملکرد بهتری در این مساله دارند.

کلمات کلیدی: شبکه‌های کانولوشنی، یادگیری عمیق، تشخیص ژست دست، دسته‌بندی ویدیو.