3D Convolutional Neural Networks for Human Action Recognition

Shuiwang Ji SHUIWANG.JI@ASU.EDU
Arizona State University, Tempe, AZ 85287, USA

Wei Xu XW@SV.NEC-LABS.COM
Ming Yang MYANG@SV.NEC-LABS.COM
Kai Yu KYU@SV.NEC-LABS.COM
NEC Laboratories America, Inc., Cupertino, CA 95014, USA

Abstract

We consider the fully automated recognition of actions in uncontrolled environment. Most existing work relies on domain knowledge to construct complex handcrafted features from inputs. In addition, the environments are usually assumed to be controlled. Convolutional neural networks (CNNs) are a type of deep models that can act directly on the raw inputs, thus automating the process of feature construction. However, such models are currently limited to handle 2D inputs. In this paper, we develop a novel 3D CNN model for action recognition. This model extracts features from both spatial and temporal dimensions by performing 3D convolutions, thereby capturing the motion information encoded in multiple adjacent frames. The developed model generates multiple channels of information from the input frames, and the final feature representation is obtained by combining information from all channels. We apply the developed model to recognize human actions in real-world environment, and it achieves superior performance without relying on handcrafted features.

1. Introduction

Recognizing human actions in real-world environment finds applications in a variety of domains including intelligent video surveillance, customer attributes, and shopping behavior analysis. However, accurate recognition of actions is a highly challenging task due to cluttered backgrounds, occlusions, and viewpoint variations, etc. Therefore, most of the existing approaches (Efros et al., 2003; Schüldt et al., 2004; Dollár et al., 2005; Laptev & Pérez, 2007; Jhuang et al., 2007) make certain assumptions (e.g., small scale and viewpoint changes) about the circumstances under which the video was taken. However, such assumptions seldom hold in real-world environment. In addition, most of these approaches follow the conventional paradigm of pattern recognition, which consists of two steps in which the first step computes complex handcrafted features from raw video frames and the second step learns classifiers based on the obtained features. In real-world scenarios, it is rarely known which features are important for the task at hand, since the choice of feature is highly problem-dependent. Especially for human action recognition, different action classes may appear dramatically different in terms of their appearances and motion patterns.

Deep learning models (Fukushima, 1980; LeCun et al., 1998; Hinton & Salakhutdinov, 2006; Hinton et al., 2006; Bengio, 2009) are a class of machines that can learn a hierarchy of features by building high-level features from low-level ones, thereby automating the process of feature construction. Such learning machines can be trained using either supervised or unsupervised approaches, and the resulting systems have been shown to yield competitive performance in visual object recognition (LeCun et al., 1998; Hinton et al., 2006; Ranzato et al., 2007; Lee et al., 2009a), natural language processing (Collobert & Weston, 2008), and audio classification (Lee et al., 2009b) tasks. The convolutional neural networks (CNNs) (LeCun et al., 1998) are a type of deep models in which trainable filters and local neighborhood pooling operations are applied alternatingly on the raw input images, resulting in a hierarchy of increasingly complex features. It has been shown that, when trained with appropri-

Appearing in *Proceedings of the 27th International Conference on Machine Learning*, Haifa, Israel, 2010. Copyright 2010 by the author(s)/owner(s).

ate regularization (Ahmed et al., 2008; Yu et al., 2008; Mobahi et al., 2009), CNNs can achieve superior performance on visual object recognition tasks without relying on handcrafted features. In addition, CNNs have been shown to be relatively insensitive to certain variations on the inputs (LeCun et al., 2004).

As a class of attractive deep models for automated feature construction, CNNs have been primarily applied on 2D images. In this paper, we consider the use of CNNs for human action recognition in videos. A simple approach in this direction is to treat video frames as still images and apply CNNs to recognize actions at the individual frame level. Indeed, this approach has been used to analyze the videos of developing embryos (Ning et al., 2005). However, such approach does not consider the motion information encoded in multiple contiguous frames. To effectively incorporate the motion information in video analysis, we propose to perform 3D convolution in the convolutional layers of CNNs so that discriminative features along both spatial and temporal dimensions are captured. We show that by applying multiple distinct convolutional operations at the same location on the input, multiple types of features can be extracted. Based on the proposed 3D convolution, a variety of 3D CNN architectures can be devised to analyze video data. We develop a 3D CNN architecture that generates multiple channels of information from adjacent video frames and performs convolution and subsampling separately in each channel. The final feature representation is obtained by combining information from all channels. An additional advantage of the CNN-based models is that the recognition phase is very efficient due to their feed-forward nature.

We evaluated the developed 3D CNN model on the TREC Video Retrieval Evaluation (TRECVID) data[1], which consist of surveillance video data recorded in London Gatwick Airport. We constructed a multi-module event detection system, which includes 3D CNN as a module, and participated in three tasks of the TRECVID 2009 Evaluation for Surveillance Event Detection. Our system achieved the best performance on all three participated tasks. To provide independent evaluation of the 3D CNN model, we report its performance on the TRECVID 2008 development set in this paper. We also present results on the KTH data as published performance for this data is available. Our experiments show that the developed 3D CNN model outperforms other baseline methods on the TRECVID data, and it achieves competitive performance on the KTH data without depending on handcrafted features, demonstrating that the 3D CNN model is more effective for real-world environments such as those captured in TRECVID data. The experiments also show that the 3D CNN model significantly outperforms the frame-based 2D CNN for most tasks. We also observe that the performance differences between 3D CNN and other methods tend to be larger when the number of positive training samples is small.

[1] `http://www-nlpir.nist.gov/projects/trecvid/`

2. 3D Convolutional Neural Networks

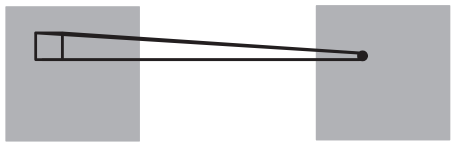
In 2D CNNs, 2D convolution is performed at the convolutional layers to extract features from local neighborhood on feature maps in the previous layer. Then an additive bias is applied and the result is passed through a sigmoid function. Formally, the value of unit at position (x, y) in the jth feature map in the ith layer, denoted as v_{ij}^{xy}, is given by

$$v_{ij}^{xy} = \tanh\left(b_{ij} + \sum_m \sum_{p=0}^{P_i-1} \sum_{q=0}^{Q_i-1} w_{ijm}^{pq} v_{(i-1)m}^{(x+p)(y+q)}\right), \tag{1}$$

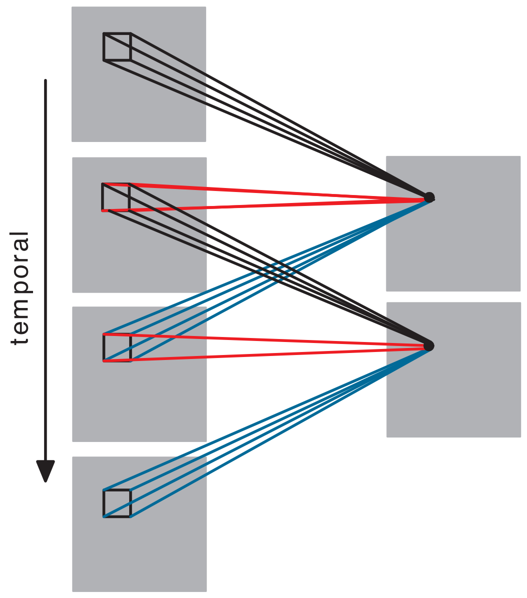
where $\tanh(\cdot)$ is the hyperbolic tangent function, b_{ij} is the bias for this feature map, m indexes over the set of feature maps in the $(i-1)$th layer connected to the current feature map, w_{ijk}^{pq} is the value at the position (p, q) of the kernel connected to the kth feature map, and P_i and Q_i are the height and width of the kernel, respectively. In the subsampling layers, the resolution of the feature maps is reduced by pooling over local neighborhood on the feature maps in the previous layer, thereby increasing invariance to distortions on the inputs. A CNN architecture can be constructed by stacking multiple layers of convolution and subsampling in an alternating fashion. The parameters of CNN, such as the bias b_{ij} and the kernel weight w_{ijk}^{pq}, are usually trained using either supervised or unsupervised approaches (LeCun et al., 1998; Ranzato et al., 2007).

2.1. 3D Convolution

In 2D CNNs, convolutions are applied on the 2D feature maps to compute features from the spatial dimensions only. When applied to video analysis problems, it is desirable to capture the motion information encoded in multiple contiguous frames. To this end, we propose to perform 3D convolutions in the convolution stages of CNNs to compute features from both spatial and temporal dimensions. The 3D convolution is achieved by convolving a 3D kernel to the cube formed by stacking multiple contiguous frames together. By this construction, the feature maps in the convolution layer is connected to multiple contiguous frames in the



(a) 2D convolution



(b) 3D convolution

Figure 1. Comparison of 2D (a) and 3D (b) convolutions. In (b) the size of the convolution kernel in the temporal dimension is 3, and the sets of connections are color-coded so that the shared weights are in the same color. In 3D convolution, the same 3D kernel is applied to overlapping 3D cubes in the input video to extract motion features.

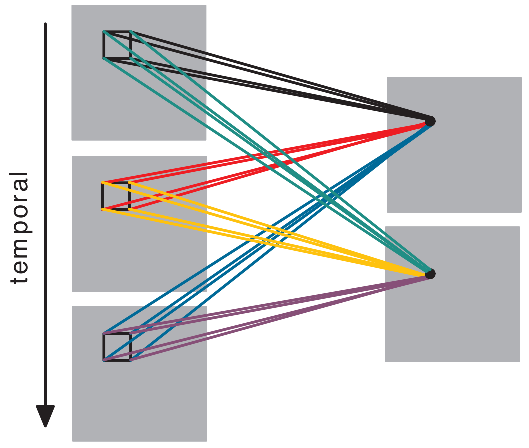


Figure 2. Extraction of multiple features from contiguous frames. Multiple 3D convolutions can be applied to contiguous frames to extract multiple features. As in Figure 1, the sets of connections are color-coded so that the shared weights are in the same color. Note that all the 6 sets of connections do not share weights, resulting in two different feature maps on the right.

previous layer, thereby capturing motion information. Formally, the value at position (x, y, z) on the jth feature map in the ith layer is given by

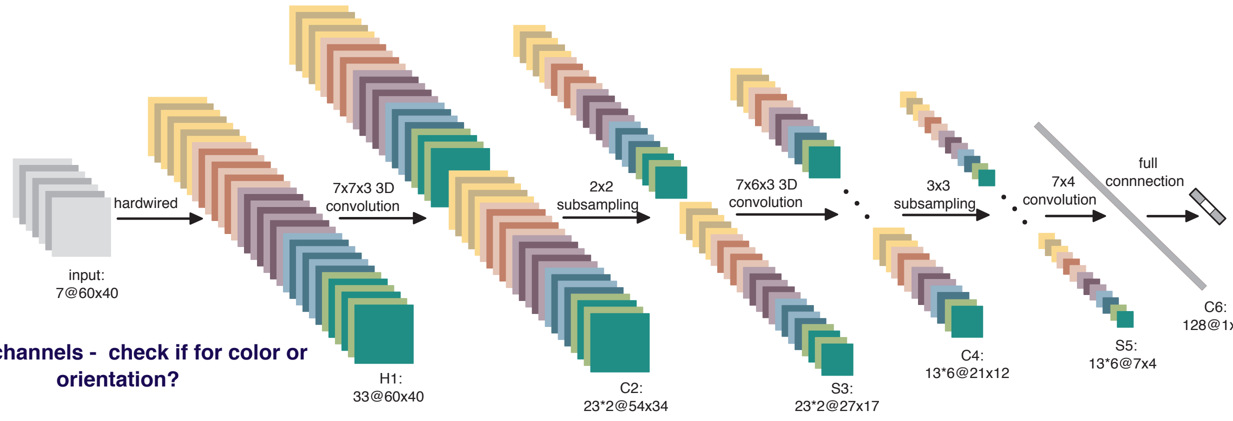
$$v_{ij}^{xyz} = \tanh\left(b_{ij} + \sum_m \sum_{p=0}^{P_i-1} \sum_{q=0}^{Q_i-1} \sum_{r=0}^{R_i-1} w_{ijm}^{pqr} v_{(i-1)m}^{(x+p)(y+q)(z+r)}\right), \tag{2}$$

where R_i is the size of the 3D kernel along the temporal dimension, w_{ijm}^{pqr} is the (p, q, r)th value of the kernel connected to the mth feature map in the previous layer. A comparison of 2D and 3D convolutions is given in Figure 1.

Note that a 3D convolutional kernel can only extract one type of features from the frame cube, since the kernel weights are replicated across the entire cube. A general design principle of CNNs is that the number of feature maps should be increased in late layers by generating multiple types of features from the same set of lower-level feature maps. Similar to the case of 2D convolution, this can be achieved by applying multiple 3D convolutions with distinct kernels to the same location in the previous layer (Figure 2).

2.2. A 3D CNN Architecture

Based on the 3D convolution described above, a variety of CNN architectures can be devised. In the following, we describe a 3D CNN architecture that we have developed for human action recognition on the TRECVID data set. In this architecture shown in Figure 3, we consider 7 frames of size 60×40 centered on the current frame as inputs to the 3D CNN model. We first apply a set of hardwired kernels to generate multiple channels of information from the input frames. This results in 33 feature maps in the second layer in 5 different channels known as gray, gradient-x, gradient-y, optflow-x, and optflow-y. The gray channel contains the gray pixel values of the 7 input frames. The feature maps in the gradient-x and gradient-y channels are obtained by computing gradients along the horizontal and vertical directions, respectively, on each of the 7 input frames, and the optflow-x and optflow-y channels contain the optical flow fields, along the horizontal and vertical directions, respectively, computed from adjacent input frames. This hardwired layer is used to encode our prior knowledge on features, and this scheme usually leads to better performance as compared to random initialization.



5 diff. channels - check if for color or orientation?

Figure 3. A 3D CNN architecture for human action recognition. This architecture consists of 1 hardwired layer, 3 convolution layers, 2 subsampling layers, and 1 full connection layer. Detailed descriptions are given in the text.

We then apply 3D convolutions with a kernel size of $7 \times 7 \times 3$ (7×7 in the spatial dimension and 3 in the temporal dimension) on each of the 5 channels separately. To increase the number of feature maps, two sets of different convolutions are applied at each location, resulting in 2 sets of feature maps in the C2 layer each consisting of 23 feature maps. This layer contains 1,480 trainable parameters. In the subsequent subsampling layer S3, we apply 2×2 subsampling on each of the feature maps in the C2 layer, which leads to the same number of feature maps with reduced spatial resolution. The number of trainable parameters in this layer is 92. The next convolution layer C4 is obtained by applying 3D convolution with a kernel size of $7 \times 6 \times 3$ on each of the 5 channels in the two sets of feature maps separately. To increase the number of feature maps, we apply 3 convolutions with different kernels at each location, leading to 6 distinct sets of feature maps in the C4 layer each containing 13 feature maps. This layer contains 3,810 trainable parameters. The next layer S5 is obtained by applying 3×3 subsampling on each feature maps in the C4 layer, which leads to the same number of feature maps with reduced spatial resolution. The number of trainable parameters in this layer is 156. At this stage, the size of the temporal dimension is already relatively small (3 for gray, gradient-x, gradient-y and 2 for optflow-x and optflow-y), so we perform convolution only in the spatial dimension at this layer. The size of the convolution kernel used is 7×4 so that the sizes of the output feature maps are reduced to 1×1. The C6 layer consists of 128 feature maps of size 1×1, and each of them is connected to all the 78 feature maps in the S5 layer, leading to 289,536 trainable parameters.

By the multiple layers of convolution and subsampling, the 7 input frames have been converted into a 128D feature vector capturing the motion information in the input frames. The output layer consists of the same number of units as the number of actions, and each unit is fully connected to each of the 128 units in the C6 layer. In this design we essentially apply a linear classifier on the 128D feature vector for action classification. For an action recognition problem with 3 classes, the number of trainable parameters at the output layer is 384. The total number of trainable parameters in this 3D CNN model is 295,458, and all of them are initialized randomly and trained by online error back-propagation algorithm as described in (LeCun et al., 1998). We have designed and evaluated other 3D CNN architectures that combine multiple channels of information at different stages, and our results show that this architecture gives the best performance.

3. Related Work

CNNs belong to the class of biologically inspired models for visual recognition, and some other variants have also been developed within this family. Motivated by the organization of visual cortex, a similar model, called HMAX (Serre et al., 2005), has been developed for visual object recognition. In the HMAX model, a hierarchy of increasingly complex features are constructed by the alternating applications of template matching and max pooling. In particular, at the S1 layer a still input image is first analyzed by an array of Gabor filters at multiple orientations and scales. The C1 layer is then obtained by pooling local neighborhoods on the S1 maps, leading to increased invariance to distortions on the input. The S2 maps are obtained

Table 1. The number of samples in each class on each of the five dates extracted from the TRECVID 2008 development data set. The total number of samples on each date and in each class are also shown.

Date\Class	CellToEar	ObjectPut	Pointing	Negative	Total
20071101	2692	1349	7845	20056	31942
20071106	1820	3075	8533	22095	35523
20071107	465	3621	8708	19604	32398
20071108	4162	3582	11561	35898	55203
20071112	4859	5728	18480	51428	80495
Total	13998	17355	55127	149081	235561

by comparing C1 maps with an array of templates, which were generated randomly from C1 maps in the training phase. The final feature representation in C2 is obtained by performing global max pooling over each of the S2 maps.

The original HMAX model is designed to analyze 2D images. In (Jhuang et al., 2007) this model has been extended to recognize actions in video data. In particular, the Gabor filters in S1 layer of the HMAX model have been replaced with some gradient and space-time modules to capture motion information. In addition, some modifications to HMAX, proposed in (Mutch & Lowe, 2008), have been incorporated into the model. A major difference between CNN- and HMAX-based models is that CNNs are fully trainable systems in which all the parameters are adjusted based on training data, while all modules in HMAX consist of handcrafted connections and parameters.

In speech and handwriting recognition, time-delay neural networks have been developed to extract temporal features (Bromley et al., 1993). In (Kim et al., 2007), a modified CNN architecture has been developed to extract features from video data. In addition to recognition tasks, CNNs have also been used in 3D image restoration problems (Jain et al., 2007).

4. Experiments

We perform experiments on the TRECVID 2008 data and the KTH data (Schüldt et al., 2004) to evaluate the developed 3D CNN model for action recognition.

4.1. Action Recognition on TRECVID Data

The TRECVID 2008 development data set consists of 49-hour videos captured at the London Gatwick Airport using 5 different cameras with a resolution of 720 × 576 at 25 fps. The videos recorded by camera number 4 are excluded as few events occurred in this scene. In this experiments, we focus on the recognition of 3 action classes (*CellToEar*, *ObjectPut*, and *Pointing*). Each action is classified in the one-against-rest manner, and a large number of negative samples were generated from actions that are not in these 3 classes. This data set was captured on five days (20071101, 20071106, 20071107, 20071108, and 20071112), and the statistics of the data used in our experiments are summarized in Table 1. The 3D CNN model used in this experiment is as described in Section 2 and Figure 3, and the number of training iterations are tuned on a separate validation set.

As the videos were recorded in real-world environments, and each frame contains multiple humans, we apply a human detector and a detection-driven tracker to locate human heads. Some sample human detection and tracking results are shown in Figure 4. Based on the detection and tracking results, a bounding box for each human that performs action was computed. The multiple frames required by 3D CNN model are obtained by extracting bounding boxes at the same position from consecutive frames before and after the current frame, leading to a cube containing the action. The temporal dimension of the cube is set to 7 in our experiments as it has been shown that 5-7 frames are enough to achieve a performance similar to the one obtainable with the entire video sequence (Schindler & Van Gool, 2008). The frames were extracted with a step size of 2. That is, suppose the current frame is numbered 0, we extract a bounding box at the same position from frames numbered -6, -4, -2, 0, 2, 4, and 6. The patch inside the bounding box on each frame is scaled to 60 × 40 pixels.

This is good for us b/c for a speeding car will only get around ~.2 seconds of video. If get 24 fps, means have ~ 5 frames to detect texting

To evaluate the effectiveness of the 3D CNN model, we report the results of the frame-based 2D CNN model. In addition, we compare the 3D CNN model with two other baseline methods, which follow the state-of-the-art bag-of-words (BoW) paradigm in which complex handcrafted features are computed. For each image cube as used in 3D CNN, we construct a BoW feature based on dense local invariant features. Then a one-

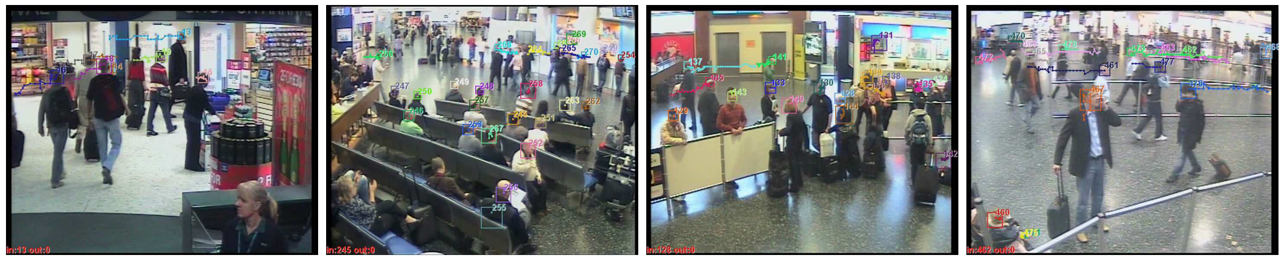


Figure 4. Sample human detection and tracking results from camera numbers 1, 2, 3, and 5, respectively from left to right.

against-all linear SVM is learned for each action class. Specifically, we extract dense SIFT descriptors (Lowe, 2004) from raw gray images or motion edge history images (MEHI) (Yang et al., 2009). Local features on raw gray images preserve the appearance information, while MEHI concerns with the shape and motion patterns. These SIFT descriptors are calculated every 6 pixels from 7×7 and 16×16 local image patches in the same cubes as in the 3D CNN model. Then they are softly quantized using a 512-word codebook to build the BoW features. To exploit the spatial layout information, we employ similar approach as the spatial pyramid matching (SPM) (Lazebnik et al., 2006) to partition the candidate region into 2×2 and 3×4 cells and concatenate their BoW features. The dimensionality of the entire feature vector is $512 \times (2 \times 2 + 3 \times 4) = 8192$. We denote the method based on gray images as $\text{SPM}^{\text{cube}}_{\text{gray}}$ and the one based on MEHI as $\text{SPM}^{\text{cube}}_{\text{MEHI}}$.

We report the 5-fold cross-validation results in which the data for a single day are used as a fold. The performance measures we used are precision, recall, and area under the ROC curve (ACU) at multiple values of false positive rates (FPR). The performance of the four methods is summarized in Table 2. We can observe from Table 2 that the 3D CNN model outperforms the frame-based 2D CNN model, $\text{SPM}^{\text{cube}}_{\text{gray}}$, and $\text{SPM}^{\text{cube}}_{\text{MEHI}}$ significantly on the action classes *CellToEar* and *ObjectPut* in all cases. For the action class *Pointing*, 3D CNN model achieves slightly worse performance than the other three methods. From Table 1 we can see that the number of positive samples in the *Pointing* class is significantly larger than those of the other two classes. Hence, we can conclude that the 3D CNN model is more effective when the number of positive samples is small. Overall, the 3D CNN model outperforms other three methods consistently as can be seen from the average performance in Table 2.

4.2. Action Recognition on KTH Data

We evaluate the 3D CNN model on the KTH data (Schüldt et al., 2004), which consist of 6 action classes performed by 25 subjects. To follow the setup in the HMAX model, we use a 9-frame cube as input and extract foreground as in (Jhuang et al., 2007). To reduce the memory requirement, the resolutions of the input frames are reduced to 80×60 in our experiments as compared to 160×120 used in (Jhuang et al., 2007). We use a similar 3D CNN architecture as in Figure 3 with the sizes of kernels and the number of feature maps in each layer modified to consider the $80 \times 60 \times 9$ inputs. In particular, the three convolutional layers use kernels of sizes 9×7, 7×7, and 6×4, respectively, and the two subsampling layers use kernels of size 3×3. By using this setting, the $80 \times 60 \times 9$ inputs are converted into 128D feature vectors. The final layer consists of 6 units corresponding to the 6 classes.

As in (Jhuang et al., 2007), we use the data for 16 randomly selected subjects for training, and the data for the other 9 subjects for testing. The recognition performance averaged across 5 random trials is reported in Table 3 along with published results in the literature. The 3D CNN model achieves an overall accuracy of 90.2% as compared with 91.7% achieved by the HMAX model. Note that the HMAX model use handcrafted features computed from raw images with 4-fold higher resolution.

5. Conclusions and Discussions

We developed a 3D CNN model for action recognition in this paper. This model construct features from both spatial and temporal dimensions by performing 3D convolutions. The developed deep architecture generates multiple channels of information from adjacent input frames and perform convolution and subsampling separately in each channel. The final feature representation is computed by combining information from all channels. We evaluated the 3D CNN model using the TRECVID and the KTH data sets. Results show that the 3D CNN model outperforms compared methods on the TRECVID data, while it achieves competitive performance on the KTH data, demonstrating its superior performance in real-world environments.

Table 2. Performance of the four methods under multiple false positive rates (FPR). The performance is measured in terms of precision, recall, and AUC. The AUC scores are multiplied by 10^3 for the ease of presentation. The highest value in each case is highlighted.

Method	FPR	Measure	CellToEar	ObjectPut	Pointing	Average
3D CNN	0.1%	Precision	**0.6433**	**0.6748**	0.8230	**0.7137**
		Recall	**0.0282**	**0.0256**	0.0152	**0.0230**
		AUC($\times 10^3$)	**0.0173**	**0.0139**	0.0075	**0.0129**
	1%	Precision	**0.4091**	**0.5154**	0.7470	**0.5572**
		Recall	**0.1109**	**0.1356**	0.0931	**0.1132**
		AUC($\times 10^3$)	**0.6759**	**0.7916**	0.5581	**0.6752**
2D CNN	0.1%	Precision	0.3842	0.5865	**0.8547**	0.6085
		Recall	0.0097	0.0176	**0.0192**	0.0155
		AUC($\times 10^3$)	0.0057	0.0109	**0.0110**	0.0092
	1%	Precision	0.3032	0.3937	0.7446	0.4805
		Recall	0.0505	0.0974	**0.1020**	0.0833
		AUC($\times 10^3$)	0.2725	0.5589	**0.6218**	0.4844
$\text{SPM}^{\text{cube}}_{\text{gray}}$	0.1%	Precision	0.3576	0.6051	0.8541	0.6056
		Recall	0.0088	0.0192	0.0191	0.0157
		AUC($\times 10^3$)	0.0044	0.0108	**0.0110**	0.0087
	1%	Precision	0.2607	0.4332	0.7511	0.4817
		Recall	0.0558	0.0961	0.0988	0.0836
		AUC($\times 10^3$)	0.3127	0.5523	0.5915	0.4855
$\text{SPM}^{\text{cube}}_{\text{MEHI}}$	0.1%	Precision	0.4848	0.5692	0.8268	0.6269
		Recall	0.0149	0.0166	0.0156	0.0157
		AUC($\times 10^3$)	0.0071	0.0087	0.0084	0.0081
	1%	Precision	0.3552	0.3961	**0.7546**	0.5020
		Recall	0.0872	0.0825	0.1006	0.0901
		AUC($\times 10^3$)	0.4955	0.4629	0.5712	0.5099

In this work, we considered the CNN model for action recognition. There are also other deep architectures, such as the deep belief networks (Hinton et al., 2006; Lee et al., 2009a), which achieve promising performance on object recognition tasks. It would be interesting to extend such models for action recognition. The developed 3D CNN model was trained using supervised algorithm in this work, and it requires a large number of labeled samples. Prior studies show that the number of labeled samples can be significantly reduced when such model is pre-trained using unsupervised algorithms (Ranzato et al., 2007). We will explore the unsupervised training of 3D CNN models in the future.

Acknowledgments

The main part of this work was done during the internship of the first author at NEC Laboratories America, Inc., Cupertino, CA.

References

Ahmed, A., Yu, K., Xu, W., Gong, Y., and Xing, E. Training hierarchical feed-forward visual recognition models using transfer learning from pseudo-tasks. In *ECCV*, pp. 69–82, 2008.

Bengio, Y. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.

Bromley, J., Guyon, I., LeCun, Y., Sackinger, E., and Shah, R. Signature verification using a siamese time delay neural network. In *NIPS*. 1993.

Collobert, R. and Weston, J. A unified architecture for natural language processing: deep neural networks with multitask learning. In *ICML*, pp. 160–167, 2008.

Dollár, P., Rabaud, V., Cottrell, G., and Belongie, S. Behavior recognition via sparse spatio-temporal features. In *ICCV VS-PETS*, pp. 65–72, 2005.

Table 3. Action recognition accuracies in percentage on the KTH data. The methods presented from top to bottom are: 3D CNN, and the methods proposed in (Schüldt et al., 2004), (Dollár et al., 2005), (Niebles et al., 2008), (Jhuang et al., 2007), and (Schindler & Van Gool, 2008). Dashes denote that results for individual classes are not available.

Method	Boxing	Handclapping	Handwaving	Jogging	Running	Walking	Average
3D CNN	90	94	97	84	79	97	90.2
Schüldt	97.9	59.7	73.6	60.4	54.9	83.8	71.7
Dollár	93	77	85	57	85	90	81.2
Niebles	98	86	93	53	88	82	83.3
Jhuang	92	98	92	85	87	96	91.7
Schindler	–	–	–	–	–	–	92.7

Efros, A. A., Berg, A. C., Mori, G., and Malik, J. Recognizing action at a distance. In *ICCV*, pp. 726–733, 2003.

Fukushima, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biol. Cyb.*, 36: 193–202, 1980.

Hinton, G. E. and Salakhutdinov, R. R. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, July 2006.

Hinton, G. E., Osindero, S., and Teh, Y. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

Jain, V., Murray, J. F., Roth, F., Turaga, S., Zhigulin, V., Briggman, K. L., Helmstaedter, M. N., Denk, W., and Seung, H. S. Supervised learning of image restoration with convolutional networks. In *ICCV*, 2007.

Jhuang, H., Serre, T., Wolf, L., and Poggio, T. A biologically inspired system for action recognition. In *ICCV*, pp. 1–8, 2007.

Kim, H.-J., Lee, J. S., and Yang, H.-S. Human action recognition using a modified convolutional neural network. In *Proceedings of the 4th International Symposium on Neural Networks*, pp. 715–723, 2007.

Laptev, I. and Pérez, P. Retrieving actions in movies. In *ICCV*, pp. 1–8, 2007.

Lazebnik, S., Achmid, C., and Ponce, J. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, pp. 2169–2178, 2006.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

LeCun, Y., Huang, F.-J., and Bottou, L. Learning methods for generic object recognition with invariance to pose and lighting. In *CVPR*, 2004.

Lee, H., Grosse, R., Ranganath, R., and Ng, A. Y. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *ICML*, pp. 609–616, 2009a.

Lee, H., Pham, P., Largman, Y., and Ng, A. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *NIPS*, pp. 1096–1104. 2009b.

Lowe, D. G. Distinctive image features from scale invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

Mobahi, H., Collobert, R., and Weston, J. Deep learning from temporal coherence in video. In *ICML*, pp. 737–744, 2009.

Mutch, J. and Lowe, D. G. Object class recognition and localization using sparse features with limited receptive fields. *International Journal of Computer Vision*, 80(1):45–57, October 2008.

Niebles, J. C., Wang, H., and Fei-Fei, L. Unsupervised learning of human action categories using spatial-temporal words. *International Journal of Computer Vision*, 79(3):299–318, 2008.

Ning, F., Delhomme, D., LeCun, Y., Piano, F., Bottou, L., and Barbano, P. Toward automatic phenotyping of developing embryos from videos. *IEEE Trans. on Image Processing*, 14(9):1360–1371, 2005.

Ranzato, M., Huang, F.-J., Boureau, Y., and LeCun, Y. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *CVPR*, 2007.

Schindler, K. and Van Gool, L. Action snippets: How many frames does human action recognition require? In *CVPR*, 2008.

Schüldt, C., Laptev, I., and Caputo, B. Recognizing human actions: A local SVM approach. In *ICPR*, pp. 32–36, 2004.

Serre, T., Wolf, L., and Poggio, T. Object recognition with features inspired by visual cortex. In *CVPR*, pp. 994–1000, 2005.

Yang, M., Lv, F., Xu, W., Yu, K., and Gong, Y. Human action detection by boosting efficient motion features. In *IEEE Workshop on Video-oriented Object and Event Classification*, 2009.

Yu, K., Xu, W., and Gong, Y. Deep learning with kernel regularization for visual recognition. In *NIPS*, pp. 1889–1896, 2008.