3D Convolutional Neural Networks for Human Action Recognition

Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu

Abstract—We consider the automated recognition of human actions in surveillance videos. Most current methods build classifiers based on complex handcrafted features computed from the raw inputs. Convolutional neural networks (CNNs) are a type of deep models that can act directly on the raw inputs. 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 the spatial and the 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 combines information from all channels. To further boost the performance of 3D CNN models, we propose to regularize the models with high-level features and combine the outputs of a variety of different models. We apply the developed models to recognize human actions in the real-world environment of airport surveillance videos, and it achieves superior performance in comparison to baseline methods.

Index Terms—deep learning, convolutional neural networks, 3D convolution, model combination, action recognition

1 Introduction

Recognizing human actions in the 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. [1]–[11]. Most of the current approaches [12]–[16] 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 the real-world environment. In addition, most of the methods follow a two-step approach in which the first step computes features from raw video frames and the second step learns classifiers based on the obtained features. In real-world scenarios, it is rarely known what features are important for the task at hand, since the choice of features 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 [17]–[21] are a class of machines that can learn a hierarchy of features by building high-level features from low-level ones. 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 [17], [19], [22]–[24], human action recognition [25]–[27], natural language processing [28], audio classification [29], brain-computer interaction [30], human tracking [31], image restoration [32], denoising [33], and segmentation tasks [34]. The convolutional neural networks (CNNs) [17] 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 appropriate regularization [35]–[37], CNNs can achieve superior performance on visual object recognition tasks. In addition, CNNs have been shown to be invariant to certain variations such as pose, lighting, and surrounding clutter [38].

As a class of deep models for feature construction, CNNs have been primarily applied on 2D images. In this paper, we explore the use of CNNs

- *S. Ji is with the Department of Computer Science, Old Dominion University, Norfolk, VA 23529.*
  E-mail: sji@cs.odu.edu
- *W. Xu is with Facebook, Inc., Palo Alto, CA 94304.*
  E-mail: emailweixu@gmail.com
- *M. Yang and K. Yu are with NEC Laboratories America, Inc., Cupertino, CA 95014.*
  E-mails: {myang,kyu}@sv.nec-labs.com

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 [39]. 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 the spatial and the 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. To further boost the performance of 3D CNN models, we propose to augment the models with auxiliary outputs computed as high-level motion features and integrate the outputs of a variety of different architectures in making predictions.

We evaluated the developed 3D CNN model on the TREC Video Retrieval Evaluation (TRECVID) data, which consist of surveillance video data recorded at London Gatwick Airport. We constructed a multi-module event detection system, which includes the 3D CNN as a major module, and participated in three tasks of the TRECVID 2009 Evaluation for Surveillance Event Detection [25]. Our system achieved the best performance on all three participated action categories (i.e., CellToEar, ObjectPut, and Pointing). To provide an 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, demonstrating that the 3D CNN model is more effective for real-world environments such as those captured in TRECVID data. The experiments also validate that the 3D CNN model significantly outperforms the frame-based 2D CNN for most tasks.

The key contributions of this work can be summarized as follows:

- We propose to apply the 3D convolution operation to extract spatial and temporal features from video data for action recognition. These 3D feature extractors operate in both the spatial and the temporal dimensions, thus capturing motion information in video streams.
- We develop a 3D convolutional neural network (CNN) architecture based on the 3D convolution feature extractors. This CNN architecture 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.
- We propose to regularize the 3D CNN models by augmenting the models with auxiliary outputs computed as high-level motion features. We further propose to boost the performance of 3D CNN models by combining the outputs of a variety of different architectures.
- We evaluate the 3D CNN models on the TRECVID 2008 development set in comparison with baseline methods and alternative architectures. Experimental results show that the proposed models significantly outperforms 2D CNN architecture and other baseline methods.

The rest of this paper is organized as follows: We describe the 3D convolution operation and the 3D CNN architecture employed in our TRECVID action recognition system in Section 2. Some related work for action recognition is discussed in Section 3. The experimental results on TRECVID and KTH data are reported in Section 4. We conclude in Section 5 with discussions.

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 an unit at position (x, y) in the jth feature map in the ith layer, denoted as v_{ij}^{xy}, is given by

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

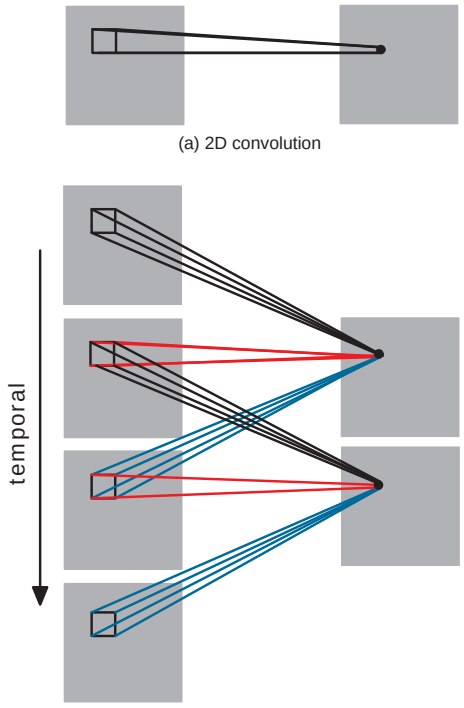


Fig. 1. Comparison of 2D (a) and 3D (b) convolutions. In (b) the size of the convolution kernel in the temporal dimension is 3, and the sets of connections are color-coded so that the shared weights are in the same color. In 3D convolution, the same 3D kernel is applied to overlapping 3D cubes in the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

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 enhancing the 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 learned using either supervised or unsupervised approaches [17], [22].

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 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

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

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 denoted by 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

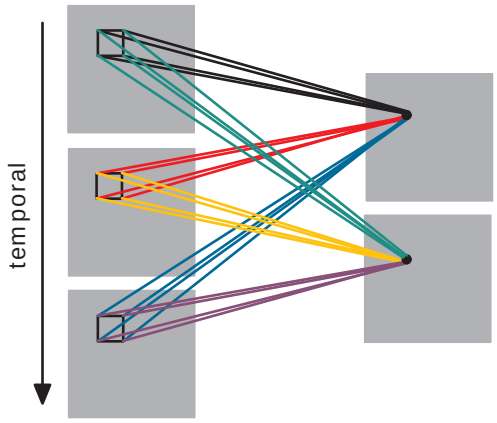


Fig. 2. Extraction of multiple features from contiguous frames. Multiple 3D convolutions can be applied to contiguous frames to extract multiple features. As in Figure 1, the sets of connections are color-coded so that the shared weights are in the same color. Note that all the 6 sets of connections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

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 employed to encode our prior knowledge on features, and this scheme usually leads to better performance as compared to the random initialization.

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. 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 a reduced spatial resolution. 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. 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 a reduced spatial resolution. 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.

After 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. All the trainable parameters in this model are initialized randomly and trained by the online error back-propagation algorithm as described in [17]. 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.

2.3 Model Regularization

The inputs to 3D CNN models are limited to a small number of contiguous video frames due to the increased number of trainable parameters as the size of input window increases. On the other hand, many human actions span a number of frames. Hence, it is desirable to encode high-level motion information into the 3D CNN models. To this end, we propose to compute motion features from a large number of frames and regularize the 3D CNN models by using these motion features as auxiliary outputs (Figure 4). Similar ideas have been used in image classification tasks [35]–[37], but its performance in action recognition is not clear. In particular, for each training action we generate a feature vector encoding the long-term action information beyond the information contained in the input frame cube to the CNN. We then encourage the CNN to learn a feature vector close to this feature. This is achieved by connecting a number of auxiliary output units to the last hidden layer of CNN and clamping the computed feature vectors on the auxiliary units during training. This

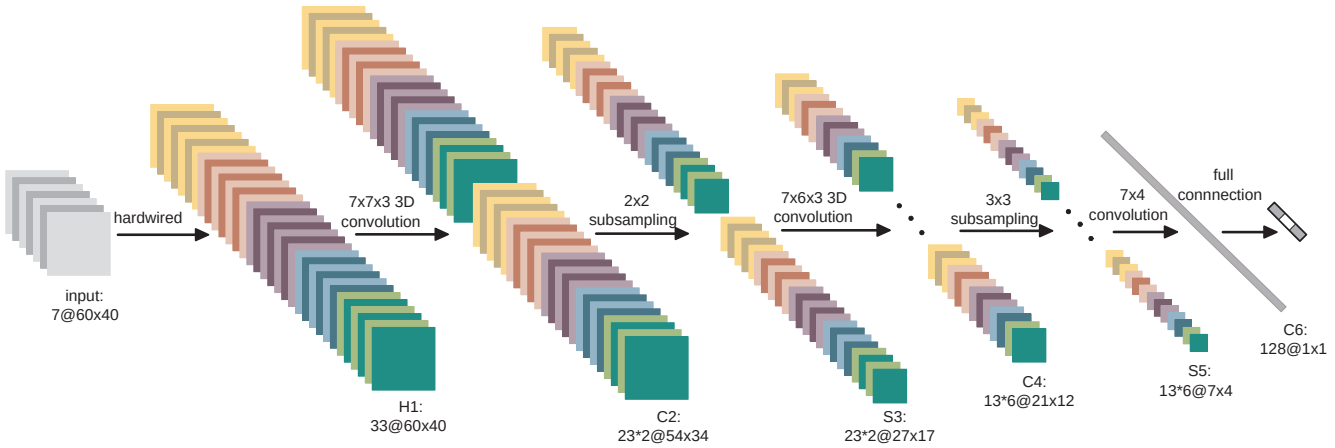


Fig. 3. A 3D CNN architecture for human action recognition. This architecture consists of 1 hardwired layer, 3 convolution layers, 2 subsampling layers, and 1 full connection layer. Detailed descriptions are given in the text.

will encourage the hidden layer information to be close to the high-level motion feature. More details on this scheme can be found in [35]–[37]. In the experiments, we use the bag-of-words features constructed from dense SIFT descriptors [40] computed on raw gray images and motion edge history images (MEHI) [41] as auxiliary features. Results show that such regularization scheme leads to consistent performance improvements.

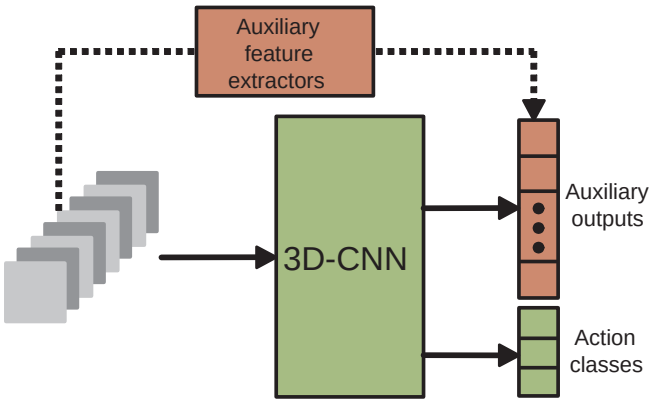


Fig. 4. The regularized 3D CNN architecture.

2.4 Model Combination

Based on the 3D convolution operations, a variety of 3D CNN architectures can be designed. Among the architectures considered in this paper, the one introduced in Section 2.2 yields the best performance on the TRECVID data. However, this may not be the case for other data sets. The selection of optimal architecture for a problem is challenging, since this depends on the specific applications. An alternative approach is to construct multiple models and combine the outputs of these models for making predictions [42]–[44]. This scheme has also been used in combining traditional neural networks [45]. However, the effect of model combination in the context of convolutional neural networks for action recognition has not been investigated. In this paper, we propose to construct multiple 3D CNN models with different architectures, hence capturing potentially complementary information from the inputs. In the prediction phase, the input is given to each model and the outputs of these models are then combined. Experimental results demonstrate that this model combination scheme is very effective in boosting the performance of 3D CNN models on action recognition tasks.

2.5 Model Implementation

The 3D CNN models are implemented in C++ as part of NEC's human action recognition system [25]. The implementation details are based on those of the original CNN as described in [17], [46]. All the subsampling layers apply max sampling as described in [47]. The overall loss function used to train the regularized models is a weighted summation of the loss functions induced by the true action classes and the auxiliary outputs. The weight for the true action classes is set to 1 and that for the auxiliary outputs is set to 0.005 empirically. All the model parameters are randomly initialized as in [17], [46] and are trained using the stochastic diagonal Levenberg-Marquardt method [17], [46].

TABLE 1
The number of samples on the five dates extracted from the TRECVID 2008 development data set.

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

In this method, a learning rate is computed for each parameter using the diagonal terms of an estimate of the Gauss-Newton approximation to the Hessian matrix on 1000 randomly sampled training instances.

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 [48], 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 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 [16] 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 [49], 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 hard-coded parameters.

In speech and handwriting recognition, time-delay neural networks have been developed to extract temporal features [50]. In [51], 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 [32].

4 EXPERIMENTS

We focus on the TRECVID 2008 data to evaluate the developed 3D CNN models for action recognition in surveillance videos. Meanwhile, we also perform experiments on the KTH data [13] to compare with previous methods.

4.1 Action Recognition on TRECVID Data

The TRECVID 2008 development data set consists of 49-hour videos captured at 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 the current 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. Multiple 3D CNN models are evaluated in this experiment, including the one described in Figure 3.

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. The detailed procedure for tracking is described in [52], and some sample results are shown in Figure 5. Based on the detection and tracking results, a bounding box for each human that performs action was

Fig. 5. Sample human detection and tracking results from camera numbers 1, 2, 3, and 5 (left to right).

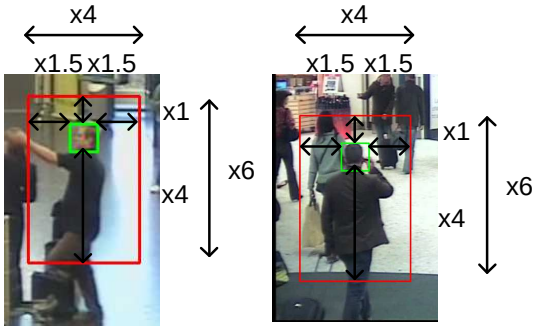


Fig. 6. Illustration of the procedure to crop the bounding box from the head tracking results.

computed. The procedure to crop the bounding box from the head tracking results is illustrated in Figure 6. 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 [53]. 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.

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 four other methods, which build state-of-the-art spatial pyramid matching (SPM) features from local features computed on dense grid or spatio-temporal interest points (STIPs). For these methods, we construct SPM features based on local invariant features computed from each image cube as used in 3D CNN. Then a one-against-all linear SVM is learned for each action class. For dense features, we extract SIFT descriptors [40] from raw gray images or motion edge history images (MEHI) [41]. Local features on raw gray images preserve the appearance information, while MEHI concerns with the shape and motion patterns. The dense SIFT descriptors are calculated every 6 pixels from 7×7 and 16×16 local image patches. For features based on STIPs, we employ the temporally integrated spatial response (TISR) method [54], which has shown promising performance on action recognition. The local features are softly quantized (each local feature can be assigned to multiple codebook words) using a 512-word codebook. To exploit the spatial layout information, we employ the spatial pyramid matching (SPM) method [55] to partition the candidate region into 2×2 and 3×4 cells and concatenate their 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 $\text{SPM}^{\text{cube}}_{\text{gray}}$, the one based on MEHI as $\text{SPM}^{\text{cube}}_{\text{MEHI}}$, and the one based on TISR as $\text{SPM}^{\text{cube}}_{\text{TISR}}$. We also concatenate $\text{SPM}^{\text{cube}}_{\text{gray}}$ and $\text{SPM}^{\text{cube}}_{\text{MEHI}}$ feature vectors into a single vector, leading to the 16384-dimensional feature representation denoted as $\text{SPM}^{\text{cube}}_{\text{gray+MEHI}}$.

In the first set of experiments, we report the performance of the 3D CNN architecture described in Figure 3 as this model achieved the best performance. This architecture is denoted as $\text{3D-CNN}^{\text{s}}_{332}$, since the five channels are convolved separately (the superscript s), and the first two convolutional layers use 3D convolution and the last convolutional layer use 2D convolution (the subscript 332). We also report the performance of the regularized 3D CNN model based on $\text{3D-CNN}^{\text{s}}_{332}$. In this model, denoted as $\text{3D-RCNN}^{\text{s}}_{332}$, the auxiliary outputs are obtained by applying PCA to reduce the dimensionality of 8192-dimensional $\text{SPM}^{\text{cube}}_{\text{gray}}$ and $\text{SPM}^{\text{cube}}_{\text{MEHI}}$ features to 150 dimensions and then concatenating them into a 300-dimensional feature vector.

TABLE 2
Performance of the seven methods under multiple false positive rates (FPR). The AUC scores are multiplied by 10^3 for ease of presentation. The highest performance in each case is shown in bold face.

Method	FPR	Measure	CellToEar	ObjectPut	Pointing	Average
3D-RCNN$^{s}_{332}$	0.1%	Precision	0.5717	**0.7348**	0.8380	**0.7148**
		Recall	0.0211	**0.0348**	0.0169	**0.0243**
		AUC($\times 10^3$)	0.0114	**0.0209**	0.0096	**0.0139**
	1%	Precision	0.3917	**0.5384**	0.7450	**0.5584**
		Recall	0.1019	**0.1466**	0.0956	**0.1147**
		AUC($\times 10^3$)	0.6272	**0.9044**	0.5665	**0.6993**
3D-CNN$^{s}_{332}$	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
SPM$^{cube}_{gray}$	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
SPM$^{cube}_{MEHI}$	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
SPM$^{cube}_{gray+MEHI}$	0.1%	Precision	0.5279	0.6422	**0.8657**	0.6786
		Recall	0.0177	0.0226	**0.0211**	0.0205
		AUC($\times 10^3$)	0.0098	0.0129	**0.0122**	0.0116
	1%	Precision	0.3753	0.4668	**0.7671**	0.5364
		Recall	0.0951	0.1102	**0.1078**	0.1043
		AUC($\times 10^3$)	0.5822	0.6431	**0.6478**	0.6244
TISR	0.1%	Precision	0.3391	0.6201	0.8553	0.6048
		Recall	0.0081	0.0205	0.0193	0.0160
		AUC($\times 10^3$)	0.0041	0.0121	0.0112	0.0091
	1%	Precision	0.2831	0.4628	0.7502	0.4987
		Recall	0.0625	0.1083	0.0983	0.0879
		AUC($\times 10^3$)	0.3717	0.6134	0.5897	0.5249

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 seven methods is summarized in Table 2, and the average performance over all action classes is plotted in Figure 7. We can observe from these results that the 3D CNN models outperform the frame-based 2D CNN model, SPM$^{cube}_{gray}$, and SPM$^{cube}_{MEHI}$ 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. Concatenation of SPM$^{cube}_{gray}$ and SPM$^{cube}_{MEHI}$ features yield improved performance over individual features, but the performance is still lower than that of the 3D CNN models. We can also observe that our models also outperform the method based on the spatio-temporal feature TISR. Overall, the 3D CNN models outperform other methods consistently as can be seen from the average performance in Figure 7. In addition, the regularized model yields higher performance than the one without regularization

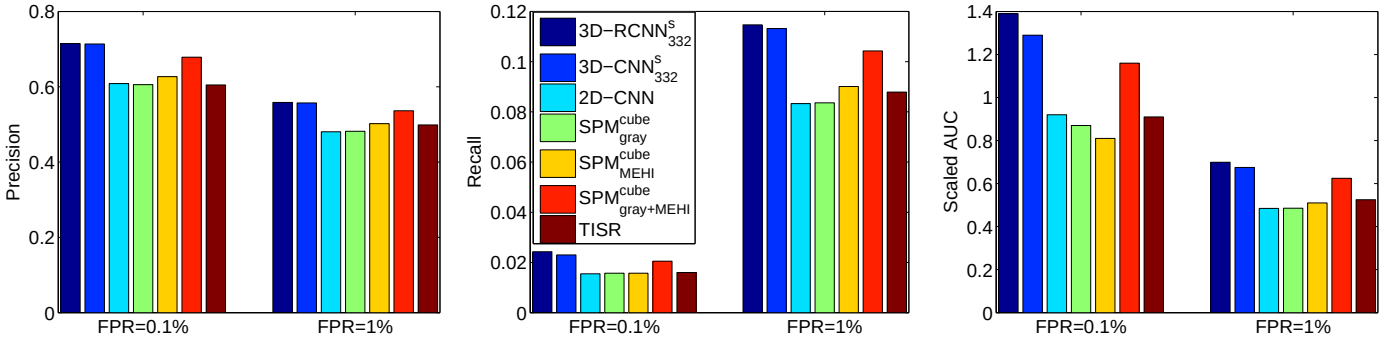


Fig. 7. Average performance comparison of the seven methods under different false positive rates (FPR). The AUC scores at FPR=0.1% and 1% are multiplied by 10^5 and 10^3, respectively, for better visualization.

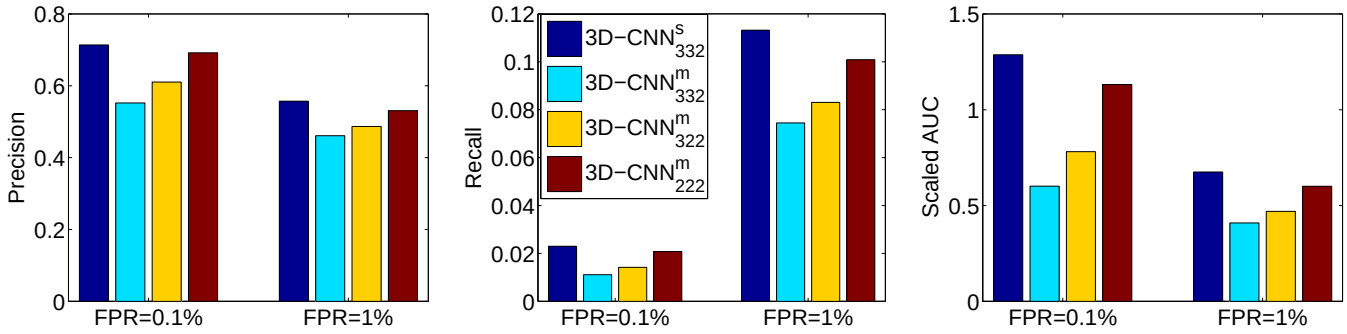


Fig. 8. Average performance comparison of the four different 3D CNN architectures under different false positive rates (FPR). The AUC scores at FPR=0.1% and 1% are multiplied by 10^5 and 10^3, respectively, for better visualization.

in all cases. Although the improvement by the regularized model is not significant, the following experiments show that significant performance improvements can be obtained by combining the two models.

To evaluate the effectiveness of model combination in the context of CNN for action recognition, we develop three alternative 3D CNN architectures described below.

- 3D-CNN$^{m}_{332}$ denotes the architecture in which the different channels are "mixed", and the first two convolutional layers use 3D convolution, and the last layer use 2D convolution. "mixed" means that the channels of the same type (i.e., gradient-x and gradient-y, optflow-x and optflow-y) are convolved separately, but they are connected to the same set of feature planes in the first convolutional layer. In the second convolutional layer, all five channels are connected to the same set of feature planes. In contrast, for models with superscript s, all five channels are connected to separate feature planes in all layers.
- 3D-CNN$^{m}_{322}$ denotes a model similar to 3D-CNN$^{m}_{332}$, but only the first convolutional layer use 3D convolution, and the other two layers use 2D convolution.
- 3D-CNN$^{m}_{222}$ denotes a model similar to 3D-CNN$^{m}_{332}$, but all three convolutional layers use 2D convolution.

The average performance of these three models along with that of 3D-CNN$^{s}_{332}$ is plotted in Figure 8. We can observe that the performance of these three alternative architecture is lower than that of 3D-CNN$^{s}_{332}$. However, we show in the following that combination of these models can lead to significant performance improvement.

To evaluate the effectiveness of model combination, we tuned each of the five models (3D-RCNN$^{s}_{332}$, 3D-CNN$^{s}_{332}$, 3D-CNN$^{m}_{222}$, 3D-CNN$^{m}_{322}$, and 3D-CNN$^{m}_{332}$) individually and then combine their outputs to make prediction. We combine models incrementally in order of decreasing individual performance. That is, the models are sorted in a decreasing order of individual performance and they are combined incrementally from the first to the last. The reason for doing this is that, we expect individual

TABLE 3
Performance of different combinations of the 3D CNN models. In this table, numbers 1 through 5 represent the models 3D-RCNN$^{s}_{332}$, 3D-CNN$^{s}_{332}$, 3D-CNN$^{m}_{222}$, 3D-CNN$^{m}_{322}$, and 3D-CNN$^{m}_{332}$, respectively. The highest performance in each case is shown in bold face.

FPR	Measure	Method	CellToEar	ObjectPut	Pointing	Average
0.1%	Precision	1	0.5717	0.7348	0.8380	0.7148
		1+2	0.6547	0.7477	0.8710	0.7578
		1+2+3	0.6716	**0.7736**	0.8810	0.7754
		1+2+3+4	**0.7015**	0.7588	**0.8869**	**0.7824**
		1+2+3+4+5	0.6948	0.7477	0.8853	0.7759
	Recall	1	0.0211	0.0348	0.0169	0.0243
		1+2	0.0299	0.0372	0.0220	0.0297
		1+2+3	0.0323	**0.0429**	0.0242	0.0331
		1+2+3+4	**0.0369**	0.0395	**0.0256**	**0.0340**
		1+2+3+4+5	0.0364	0.0372	0.0252	0.0329
	AUC($\times 10^3$)	1	0.0114	0.0209	0.0096	0.0139
		1+2	0.0194	0.0213	0.0109	0.0172
		1+2+3	0.0206	0.0240	0.0117	0.0187
		1+2+3+4	0.0227	**0.0244**	**0.0132**	**0.0201**
		1+2+3+4+5	**0.0230**	0.0237	0.0123	0.0197
1%	Precision	1	0.3917	0.5384	0.7450	0.5584
		1+2	0.4301	0.5573	0.7795	0.5890
		1+2+3	0.4623	**0.5700**	0.7902	0.6075
		1+2+3+4	0.4737	0.5621	0.7942	0.6100
		1+2+3+4+5	**0.4816**	0.5657	**0.7973**	**0.6149**
	Recall	1	0.1019	0.1466	0.0956	0.1147
		1+2	0.1193	0.1583	0.1156	0.1311
		1+2+3	0.1357	**0.1666**	0.1232	0.1418
		1+2+3+4	0.1424	0.1612	**0.1261**	**0.1433**
		1+2+3+4+5	**0.1428**	0.1590	0.1256	0.1425
	AUC($\times 10^3$)	1	0.6272	0.9044	0.5665	0.6993
		1+2	0.7564	0.9758	0.6762	0.8028
		1+2+3	0.8428	**1.0240**	0.7490	0.8720
		1+2+3+4	**0.8990**	0.9708	**0.7862**	**0.8853**
		1+2+3+4+5	0.8966	0.9539	0.7522	0.8675

models with high performance will lead to more significant improvements when they are combined. We report the combined performance for each combinations in Table 3 and plot the average performance in Figure 9. We can observe that combination of models leads to significant performance improvements in almost all cases except the combination of 3D-CNN$^{m}_{332}$, which leads to slight performance degradation. This shows that the different architectures encode complementary information, and combination of these models effectively integrates these information though the performance of some of the individual models is low. Figures 10-12 show some sample actions in each of the three classes that are classified correctly and incorrectly by the combined model. It can be observed that most of the misclassified actions are hard to recognize even by human.

To highlight the performance improvements over our previous result in [26], we report the best performance achieved by the methods in [26] and that of the new methods proposed in this paper in Table 4. We can observe that our new methods in this paper improve over the previous results significantly in all cases.

4.2 Action Recognition on the KTH Data

We evaluate the 3D CNN model on the KTH data [13], 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 [16]. 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 [16]. 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

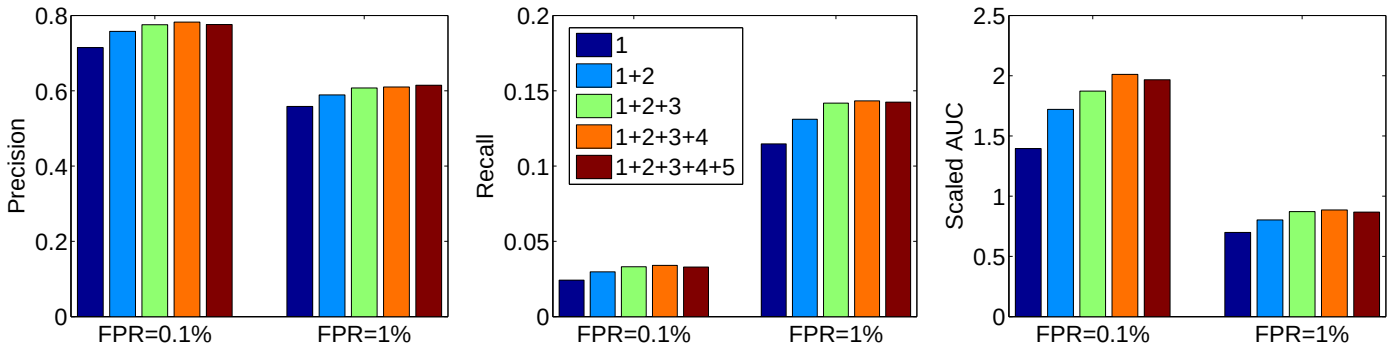


Fig. 9. Performance of different combinations of the 3D CNN architectures. The AUC scores at FPR=0.1% and 1% are multiplied by 10^5 and 10^3, respectively, for better visualization. See the caption of Table 3 and texts for detailed explanations.

TABLE 4
Comparison of the best performance achieved by our previous methods in [26] (ICML models) with the new methods proposed in this paper (New models).

	Precision		Recall		AUC	
FPR	0.1%	1%	0.1%	1%	0.1%	1%
New models	0.7824	0.6149	0.0340	0.1433	0.0201	0.8853
ICML models	0.7137	0.5572	0.0230	0.1132	0.0129	0.6752

converted into 128D feature vectors. The final layer consists of 6 units corresponding to the 6 classes.

As in [16], we use the data for 16 randomly selected subjects for training, and the data for the other 9 subjects for testing. Majority voting is used to produce labels for a video sequence based on the predictions for individual frames. The recognition performance averaged across 5 random trials is reported in Table 5 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. Also, some of the methods in Table 5 used different training/test splits of the data.

5 Conclusions and Discussions

We developed 3D CNN models for action recognition in this paper. These models 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 obtained by combining information from all channels. We developed model regularization and combination schemes to further boost the model performance. We evaluated the 3D CNN models on 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.

In this work, we considered the CNN model for action recognition. There are also other deep architectures, such as the deep belief networks [19], [23], 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 [22]. We will explore the unsupervised training of 3D CNN models in the future.

References

[1] I. Laptev and T. Lindeberg, "Space-time interest points," in *Proceedings of the IEEE International Conference on Computer Vision*, 2003, pp. 432–439.

[2] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld, "Learning realistic human actions from movies," in *Proceedings of the 2008 IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

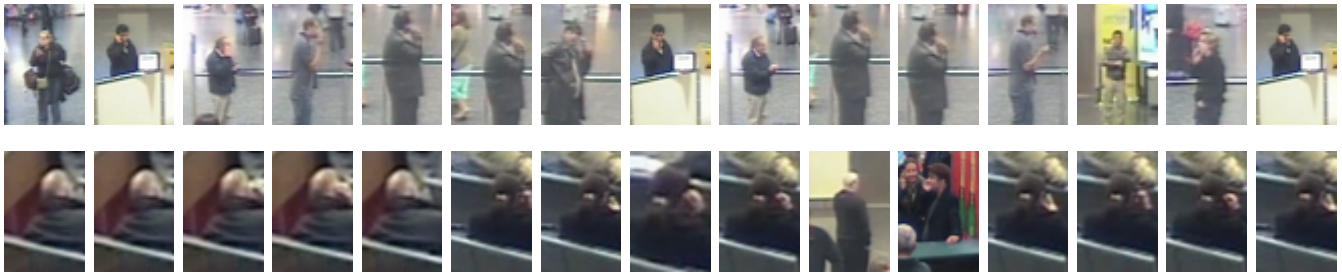


Fig. 10. Sample actions in the CellToEar class. The top row shows actions that are correctly recognized by the combined 3D CNN model while the bottom row shows those that are misclassified by the model.

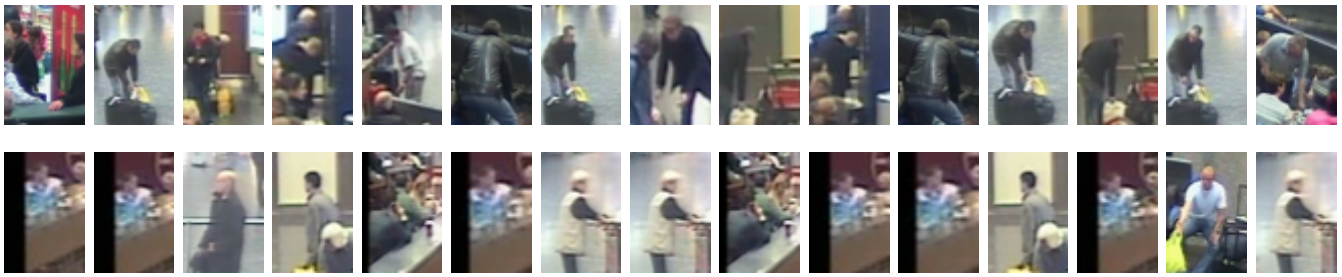


Fig. 11. Sample actions in the ObjectPut class. The top row shows actions that are correctly recognized by the combined 3D CNN model while the bottom row shows those that are misclassified by the model.

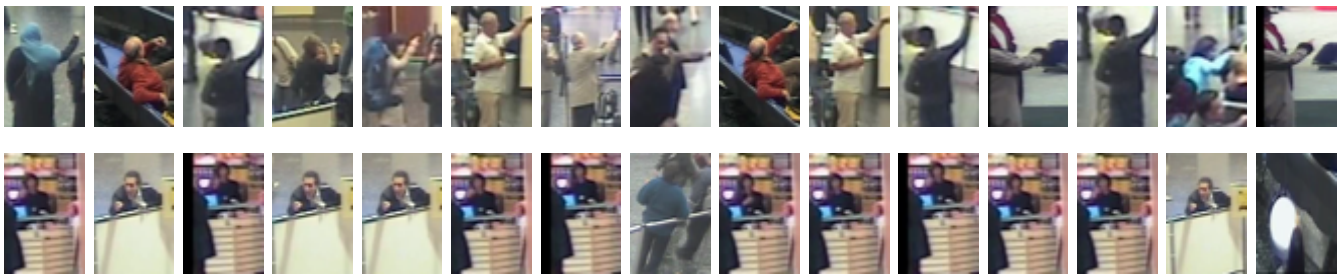


Fig. 12. Sample actions in the Pointing class. The top row shows actions that are correctly recognized by the combined 3D CNN model while the bottom row shows those that are misclassified by the model.

[3] J. Liu, J. Luo, and M. Shah, "Recognizing realistic actions from videos," in *Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 1996–2003.

[4] Y. Wang and G. Mori, "Max-margin hidden conditional random fields for human action recognition," in *Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 872–879.

[5] O. Duchenne, I. Laptev, J. Sivic, F. Bach, and J. Ponce, "Automatic annotation of human actions in video," in *Proceedings of the 12th International Conference on Computer Vision*, 2009, pp. 1491–1498.

[6] Y. Wang and G. Mori, "Hidden part models for human action recognition: Probabilistic versus max margin," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011.

[7] H. Wang, M. M. Ullah, A. Kläser, I. Laptev, and C. Schmid, "Evaluation of local spatio-temporal features for action recognition," in *British Machine Vision Conference*, 2009, p. 127.

[8] M. Marszalek, I. Laptev, and C. Schmid, "Actions in context," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 2929–2936.

[9] I. Junejo, E. Dexter, I. Laptev, and P. Pérez, "View-independent action recognition from temporal self-similarities," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 1, pp. 172–185, 2011.

[10] V. Delaitre, I. Laptev, and J. Sivic, "Recognizing human actions in still images: a study of bag-of-features and part-based representations," in *Proceedings of the 21st British Machine Vision Conference*, 2010.

[11] Q. Le, W. Zou, S. Yeung, and A. Ng, "Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis," in *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, 2011, pp. 3361–3368.

[12] A. A. Efros, A. C. Berg, G. Mori, and J. Malik, "Recognizing action at a distance," in *Proceedings of the Ninth IEEE International Conference on Computer Vision*, 2003, pp. 726–733.

[13] C. Schüldt, I. Laptev, and B. Caputo, "Recognizing human actions: A local SVM approach," in *Proceedings of the 17th International Conference on Pattern Recognition*, 2004, pp. 32–36.

[14] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie, "Behavior recognition via sparse spatio-temporal features," in

TABLE 5
Action recognition accuracies in percentage on the KTH data. Note that we use the same training/test split as [16], and other methods use different splits.

Method	Boxing	Handclapping	Handwaving	Jogging	Running	Walking	Average
3D CNN	90	94	97	84	79	97	90.2
Schüldt [13]	97.9	59.7	73.6	60.4	54.9	83.8	71.7
Dollár [14]	93	77	85	57	85	90	81.2
Niebles [56]	98	86	93	53	88	82	83.3
Jhuang [16]	92	98	92	85	87	96	91.7
Schindler [53]	–	–	–	–	–	–	92.7

Second Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005, pp. 65–72.

[15] I. Laptev and P. Pérez, "Retrieving actions in movies," in *Proceedings of the 11th International Conference on Computer Vision*, 2007, pp. 1–8.

[16] H. Jhuang, T. Serre, L. Wolf, and T. Poggio, "A biologically inspired system for action recognition," in *Proceedings of the 11th International Conference on Computer Vision*, 2007, pp. 1–8.

[17] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, November 1998.

[18] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, July 2006.

[19] G. E. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, pp. 1527–1554, 2006.

[20] Y. Bengio, "Learning deep architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[21] Y. Bengio and Y. LeCun, "Scaling learning algorithms towards AI," in *Large-Scale Kernel Machines*, L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, Eds. MIT Press, 2007.

[22] M. Ranzato, F. Huang, Y. Boureau, and Y. LeCun, "Unsupervised learning of invariant feature hierarchies with applications to object recognition," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2007.

[23] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009, pp. 609–616.

[24] M. Norouzi, M. Ranjbar, and G. Mori, "Stacks of convolutional restricted Boltzmann machines for shift-invariant feature learning," in *Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[25] M. Yang, S. Ji, W. Xu, J. Wang, F. Lv, K. Yu, Y. Gong, M. Dikmen, D. J. Lin, and T. S. Huang, "Detecting human actions in surveillance videos," in *TREC Video Retrieval Evaluation Workshop*, 2009.

[26] S. Ji, W. Xu, M. Yang, and K. Yu, "3D convolutional neural networks for human action recognition," in *Proceedings of the Twenty-Seventh International Conference on Machine Learning*, 2010, pp. 495–502.

[27] G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler, "Convolutional learning of spatio-temporal features," in *Proceedings of the 11th European conference on Computer vision*, 2010, pp. 140–153.

[28] R. Collobert and J. Weston, "A unified architecture for natural language processing: deep neural networks with multitask learning," pp. 160–167, 2008.

[29] H. Lee, P. Pham, Y. Largman, and A. Ng, "Unsupervised feature learning for audio classification using convolutional deep belief networks," in *Advances in Neural Information Processing Systems 22*, 2009, pp. 1096–1104.

[30] H. Cecotti and A. Graser, "Convolutional neural networks for P300 detection with application to brain-computer interfaces," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, pp. 433–445, 2011.

[31] J. Fan, W. Xu, Y. Wu, and Y. Gong, "Human tracking using convolutional neural networks," *IEEE Transactions on Neural Networks*, vol. 21, pp. 1610–1623, October 2010.

[32] V. Jain, J. F. Murray, F. Roth, S. Turaga, V. Zhigulin, K. L. Briggman, M. N. Helmstaedter, W. Denk, and H. S. Seung, "Supervised learning of image restoration with convolutional networks," in *Proceedings of the 11th International Conference on Computer Vision*, 2007.

[33] V. Jain and S. Seung, "Natural image denoising with convolutional networks," in *Advances in Neural Information Processing Systems 21*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds., 2009, pp. 769–776.

[34] S. C. Turaga, J. F. Murray, V. Jain, F. Roth, M. Helmstaedter, K. Briggman, W. Denk, and H. S. Seung, "Convolutional networks can learn to generate affinity graphs for image segmentation," *Neural Computation*, vol. 22, no. 2, pp. 511–538, 2010.

[35] A. Ahmed, K. Yu, W. Xu, Y. Gong, and E. Xing, "Training hierarchical feed-forward visual recognition models using transfer learning from pseudo-tasks," in *Proceedings of the 10th European Conference on Computer Vision*, 2008, pp. 69–82.

[36] K. Yu, W. Xu, and Y. Gong, "Deep learning with kernel regularization for visual recognition," in *Advances in Neural Information Processing Systems 21*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds., 2009, pp. 1889–1896.

[37] H. Mobahi, R. Collobert, and J. Weston, "Deep learning from temporal coherence in video," in *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009, pp. 737–744.

[38] Y. LeCun, F. Huang, and L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2004.

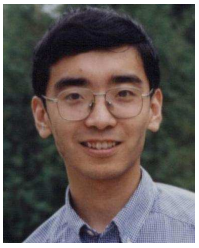
[39] F. Ning, D. Delhomme, Y. LeCun, F. Piano, L. Bottou, and P. Barbano, "Toward automatic phenotyping of developing embryos from videos," *IEEE Transactions on Image Processing*, vol. 14, no. 9, pp. 1360–1371, 2005.

[40] D. G. Lowe, "Distinctive image features from scale invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[41] M. Yang, F. Lv, W. Xu, K. Yu, and Y. Gong, "Human action detection by boosting efficient motion features," in *IEEE Workshop on Video-oriented Object and Event Classification*, 2009.
[42] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, pp. 123–140, 1996.
[43] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proceedings of the Thirteenth International Conference on Machine Learning*, 1996, pp. 148–156.
[44] J. Kittler, M. Hatef, R. P. Duin, and J. Matas, "On combining classifiers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, pp. 226–239, 1998.
[45] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, pp. 993–1001, October 1990.
[46] Y. LeCun, L. Bottou, G. Orr, and K. Muller, "Efficient backprop," in *Neural Networks: Tricks of the trade*, G. Orr and M. K., Eds. Springer, 1998.
[47] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?" in *Proceedings of the 12th International Conference on Computer Vision*. IEEE, 2009.
[48] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio, "Object recognition with cortex-like mechanisms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 3, pp. 411–426, 2007.
[49] J. Mutch and D. G. Lowe, "Object class recognition and localization using sparse features with limited receptive fields," *International Journal of Computer Vision*, vol. 80, no. 1, pp. 45–57, October 2008.
[50] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a Siamese time delay neural network," in *Advances in Neural Information Processing Systems 6*, 1994, pp. 737–744.
[51] H.-J. Kim, J. S. Lee, and H.-S. Yang, "Human action recognition using a modified convolutional neural network," in *Proceedings of the 4th International Symposium on Neural Networks*, 2007, pp. 715–723.
[52] M. Yang, F. Lv, W. Xu, and Y. Gong, "Detection driven adaptive multi-cue integration for multiple human tracking," in *Proceedings of the 12th International Conference on Computer Vision*, 2009, pp. 1554–1561.
[53] K. Schindler and L. Van Gool, "Action snippets: How many frames does human action recognition require?" in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2008.
[54] G. Zhu, M. Yang, K. Yu, W. Xu, and Y. Gong, "Detecting video events based on action recognition in complex scenes using spatio-temporal descriptor," in *Proceedings of the 17th ACM International Conference on Multimedia*, 2009, pp. 165–174.
[55] S. Lazebnik, C. Achmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006, pp. 2169–2178.
[56] J. C. Niebles, H. Wang, and L. Fei-Fei, "Unsupervised learning of human action categories using spatial-temporal words," *International Journal of Computer Vision*, vol. 79, no. 3, pp. 299–318, 2008.

Shuiwang Ji received the Ph.D. degree in computer science from Arizona State University, Tempe, AZ, in 2010. Currently, he is an Assistant Professor in the Department of Computer Science, Old Dominion University, Norfolk, VA. His research interests include machine learning, data mining, and bioinformatics. He received the Outstanding Ph.D. Student Award from Arizona State University in 2010.

Wei Xu received his B.S. degree from Tsinghua University, Beijing, China, in 1998, and M.S. degree from Carnegie Mellon University (CMU), Pittsburgh, USA, in 2000. From 1998 to 2001, he was a research assistant at the Language Technology Institute at CMU. In 2001, he joined NEC Laboratories America working on intelligent video analysis. He has been a research scientist in Facebook since November 2009. His research interests include computer vision, image and video understanding, machine learning and data mining.



Ming Yang received the B.E. and M.E. degrees in electronic engineering from Tsinghua University, Beijing, China, in 2001 and 2004, respectively, and the Ph.D. degree in electrical and computer engineering from Northwestern University, Evanston, Illinois, in June 2008. From 2004 to 2008, he was a research assistant in the computer vision group of Northwestern University. After his graduation, he joined NEC Laboratories America, Cupertino, California, where he is currently a research staff member. His research interests include computer vision, machine learning, video communication, large-scale image retrieval, and intelligent multimedia content analysis. He is a member of the IEEE.

Kai Yu is the head of media analytics department at NEC Laboratories America, where he manages an R&D division, responsible for NEC's technology innovation in image recognition, multimedia search, video surveillance, sensor mining, and human-computer interaction. He has published over 70 papers in top-tier conferences and journals in the area of machine learning, data mining, and computer vision. He has served as area chairs in top-tier machine learning conferences, e.g. ICML and NIPS, and taught an AI class at Stanford University as a visiting faculty. Before joining NEC, he was a senior research scientist at Siemens. He received Ph.D in Computer Science from University of Munich in 2004.