IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 13, 2019, accepted March 10, 2019, date of publication March 29, 2019, date of current version April 10, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2906654

A 3D-CNN and LSTM Based Multi-Task Learning Architecture for Action Recognition

XI OUYANG[1], SHUANGJIE XU[1], CHAOYUN ZHANG[2], PAN ZHOU[1], (Member, IEEE), YANG YANG[3], (Member, IEEE), GUANGHUI LIU[4], (Senior Member, IEEE), AND XUELONG LI[5], (Fellow, IEEE)

[1]School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, China
[2]School of Informatics, The University of Edinburgh, Edinburgh EH8 9AB, U.K.
[3]School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610051, China
[4]School of Electronics Engineering, University of Electronic Science and Technology of China, Chengdu 610051, China
[5]Center for Optical Imagery Analysis and Learning (OPTIMAL), School of Computer Science, Northwestern Polytechnical University, Xi'an 710072, China

Corresponding author: Pan Zhou (panzhou@hust.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB1107400, and in part by the National Natural Science Foundation of China under Grant 61401169, Grant 61472147, Grant 61772219, Grant 51678265, and Grant 61871470.

ABSTRACT Multi-task learning (MTL) is a machine learning method to share knowledge for multiple related machine learning tasks via learning those tasks jointly. It has been shown to be capable of effectively improving the generalization capability of each single task (learning just one task at a time). In this paper, we propose a novel MTL architecture that first combines 3D convolutional neural networks (3D CNN) plus the long short-term memory (LSTM) networks together with the MTL mechanism, tailored to information sharing of video inputs. We split each video into several clips and apply the hybrid deep model of 3D CNN and LSTM to extract the sequential features of those video clips. Therefore, our MTL model can share visual knowledge based on those video-clip features among different categories more efficiently. We evaluate our method on three popular public action recognition video datasets. The experimental results show that our novel MTL method can efficiently share detailed information in video clips among multiple action categories and outperforms other multi-task methods.

INDEX TERMS Action recognition, 3D CNN, LSTM, multi-task learning.

I. INTRODUCTION

The standard paradigm in machine learning is to learn one task at a time, however this methodology ignores the rich information contained in the related tasks from the same domain. Multi-task learning (MTL) aims to simultaneously learn multiple related prediction tasks, sharing information across the tasks [1], [4], [6], [74]. It has been shown to improve generalization performance than independently learning each task [2], [3], [5]. Due to the ability of MTL, Kumar and Daume [31] applied this method to recognize handwritten digit number and achieved great performance.

However, those MTL methods follow the same pattern: first, using a feature vector to represent each input example; second, decomposing the overall model parameters into a latent task matrix and combination matrix based on such feature description. The information will be shared in the latent task matrix across different categories. All those MTL methods require using a single feature vector to represent each input example no matter whether inputs are images or videos [32], [33]. This paradigm can be efficient for image inputs, however it has a congenital defect when dealing with videos. Due to that videos consist of rich long-term sequential information, lots of detailed information in videos will be discarded if just using a single feature vector to represent the whole video. It can only share rough general information of each video but not the detailed information within each video. Actually, only some specific video clips are similar between different videos. There are lots of examples when we investigate the task of human action recognition on video data. For example, the motion pattern of straightening knees exists both in the beginning clips of standing up and jumping. However, the following clips of those two actions are very different. Based on this key observation, we redesign a novel

The associate editor coordinating the review of this manuscript and approving it for publication was Zhanyu Ma.

MTL method to share the video clip information and do evaluation on human action recognition datasets.

Human action recognition, as a significant problem in the research community, has attracted a gargantuan amount of attention in computer vision research [7]–[12]. This is due to the wide applications in many fields, such as intelligent public safety surveillance, sociology, human-computer interaction. Meanwhile, it is a challenging task when recognizing human action in natural environment, considering the complex spatiotemporal features, cluttered backgrounds and various angles of view, etc. Feature extraction is one of the key steps in the traditional strategy in computer vision and pattern recognition to solve this problem [13]–[17]. Some feature extraction methods like Histograms of Oriented Gradients (HOG) [18], Histograms of Optical Flow (HOF) [19], and Motion Boundary Histogram (MBH) [20], are applied in this problem to compute complex features artificially. Then, those handcrafted features are used to train classifiers like Logistic Regression, and Support Vector Machines (SVM) [21].

Deep learning techniques have achieved remarkable performance in computer vision, which are focal points of both industrial and academic circles recently [22], [75]–[78]. Deep models are applied widely in many fields of computer vision, especially after that Krizhevsky *et al.* [23] greatly improve the classification accuracy of ImageNet based on the convolutional neural network (CNN). Deep learning also have shown great performance on human action recognition due to its automatic extraction of features [24], [24]–[27].

Given the aforementioned issues, we combine 3D CNNs and a Long-Short Term Memory (LSTM) network to automatically extract a sequential feature representation of each video, and propose an enhanced multi-task framework to share the similar parts of those feature representations across related action categories. In our work, we split each video into several video clips and train a 3D CNN model for those clips. Then these features generated from our 3D CNNs will be fed into LSTM models in order to learn the sequential feature representation of each video. LSTM [29] is one of recurrent neural network models [30] which enables long-range temporal interval learning. We use this network to gain a sequential feature representation of different video clips in each action video, rather than to use just a single feature vector to represent a video [32], [34]. Then, we build up a weight matrix to get the weighted mean of the feature sequence generated from the LSTM model, and enforce the matrix $\ell_{2,1}$ norm regularization on this weight matrix. We can use the $\ell_{2,1}$ norm to conduct a robust feature selection on the feature sequence as this regularization term can ensure the sparsity of the weight matrix in row.

Meanwhile, like the works of Kumar and Daume [31] and Zhou *et al.* [32], we decompose the overall model parameters into a latent task matrix and combination matrix. The columns of the latent task matrix represent the parameter vectors of latent tasks. Then, we reconstruct classifiers through linear combinations of these columns. Considering the ℓ_1 norm regularization in our optimization goal, we apply a modified stochastic gradient method (SGD-$L1$) [35] to train our model, and the overall parameters are optimized via an efficient alternative optimization strategy. This SGD-$L1$ method can efficiently train the ℓ_1-regularized models, and requires much less training time than proximal gradient methods like the accelerated proximal gradient method (APG) [36].

Our main contributions of this paper are as follows:

- We upgrade the normal MTL architecture to adapt to accept a feature sequence representation for each video input rather than the limited single feature representation.
- We uniquely combine 3D CNN, LSTM with an enhanced MTL mechanism, tailored to information sharing among videos. This promising structure digs more detailed information to share among videos.
- Experiments conducted on three benchmark datasets demonstrate that the proposed architecture outperform other MTL methods. Meanwhile, our MTL method also shows a significant improvement compared with the baseline deep models embedded in our architecture.

The remainder of this paper is organized as follows. Section II presents the related work. Section III and Section IV describe the structure of deep model and our enhanced MTL architecture, respectively. Experimental results are presented and analyzed in Section V. Finally, we conclude the study in Section VI.

II. RELATED WORK

A. MULTI-TASK LEARNING

Compared with learning each task independently [1]–[4], [34], multi-task learning is able to significantly improve generalization performance for the related task. In this paper, the key factor of applying MTL method in action recognition is that many action categories are highly correlated. Most previous methods proposed for multi-task learning are based on the assumption that all tasks are related [3], [4]. Obviously, this assumption is harsh and can debase the performance because some tasks are not similar. Then, some researchers assume that there are disjoint groups of tasks to address this problem [5], [6]. For example, Kang *et al.* [6] assume that tasks within each group lie in a low dimensional subspace through a regularization framework, and the subspaces shared by each group do not overlap. To solve this problem, the work of Kumar and Daume [31] allows the tasks in different groups to overlap with each other via representing all tasks as the linear combinations of some latent basic tasks. Furthermore, Zhou *et al.* [32] enforce ℓ_1 norm regularization on the parameter vectors of latent tasks to avoid sharing too much ''holistic'' information. Adbulnabi *et al.* [34] apply the MTL method to the multi-label tasks.

Our work is related to [31], [32], and [34], but it is different. First, our work aims to share more detailed information and we add a new weight matrix to share the similarities of video clips not just the whole videos. We also compare the performance of our method and these MTL

methods in the experiment part. Second, we apply a different optimization method which is a modified stochastic gradient method (SGD-$L1$) [35] to optimize the non-smooth convex optimization problems. Compared with other optimization methods like Accelerated Proximal Method (APG), SGD-$L1$ is quite efficient in the training time cost especially when we conduct our experiments on video data. We will explain the details in the next section.

B. HUMAN ACTION RECOGNITION

Many works of action recognition follow the orthodox paradigm of computer vision and pattern analysis. In computer vision, it is extremely successful to analyze static images with local hand-designed features. Some typical illustrations of such successful features are SIFT [37], SURF [38], ORB [39]. These methods to extract features of static images usually contain two steps: a feature detection step followed by a local feature description step. Many researchers extent those local features to calculate spatiotemporal interest points of videos in video action recognition. Similarly, The Harris-3D detector [7] and the Cuboid detector [40] are likely the most used interest points detection methods. For descriptors, popular methods are HOG [18], HOF [19], and HOG3D [41]. These features are the pre-processing step before applying them to any classification methods.

Moreover, the feature trajectory-based method [42], [43] is also successful for shallow video representations recently. The dense human trajectories by computing the information of optical flow in videos is also considered to calculate video feature vectors. Dalal *et al.* [20] extend this method to compute on the horizontal and vertical components of optical flow separately, and gain the Motion Boundary Histogram (MBH) feature. Wang and Schmid [15] apply the improved dense trajectories (iDT) to gain an impressive improvement.

C. DEEP LEARNING

Deep learning methods have been demonstrated as an effective class of methods for computer vision and pattern recognition [44]–[48]. Especially, after the remarkable results produced in ImageNet in 2012 [23], deep learning technology has actually became the common solution for computer vision. Although CNNs are considered as the technological frontier, LeCun *et al.* [49] have successfully developed the LeNet-5 (a CNN model) on MNIST dataset to recognize hand-written numbers as early as 1998. Limited to computing power at that time, CNNs can just be successfully applied in small dataset like MNIST, and CIFAR-10. Recently, with the improvement of hardware (especially GPUs), we have the capability to train a huge CNN network with stacking multiple convolutional and pooling layers. Encouraged by great results in domain of static images, there has also been a number of attempts to adapt CNNs to video recognition [24]–[26].

However, video recognition is more challenging compared to static images due to the difficulties of capturing both the spatial and temporal information of consecutive video frames [79]. Ji *et al.* [27] propose the 3D convolution operator to compute features from both spatial and temporal dimensions. Additionally, Karpathy *et al.* [54] speed up the training through a hybrid CNN structure containing a low-resolution context CNN and a high-resolution fovea CNN to implement the large-scale video classification. Moerover, the two stream CNNs method [28] is a representative work of deep learning which outperforms the iDT method [15]. Based on this work, Yue-Hei Ng *et al.* [53] combine the LSTM with this two stream CNNs and gains the state-of-art performance. The spatial stream performs action recognition from still video frames, while the temporal stream is trained to recognize action from motion in the form of dense optical flow. Furthermore, the work of Srivastava *et al.* [55] performs well in UCF-101 through applying a LSTM network to represent videos because of the long-range time sequence learning tasks capacity of LSTM.

In our work, we combine a 3D CNN and LSTM network to learn the video representation. The 3D CNN model is applied to video clips to capture features of video clips, whilst the LSTM network is used to understand the temporal correlation of these video clips.

III. METHODOLOGY

The overall structure of the proposed method is shown in Fig. 1, and the description of our structure will be partitioned into two sections. Our method has two stages to make action category predictions from raw videos. First, we train deep models to extract features of action videos whose structure is a combination of 3D CNNs and LSTM models. For each action category, we train a binary classifier via 3D CNNs and LSTM models, and these classifiers are trained independently. Second, the features generated from the second LSTM layer will be sent into our MTL model. After gaining the features in the last LSTM layer, we replace the fully-connected layer and the logistic regression layer with our enhanced MTL framework as shown in Fig 1. Meanwhile, the weight parameters learned in the logistic regression of the trained binary classifier will be used to initialize the parameters of weight matrix T_c and the latent task matrix L in the MTL model. Then the whole structure including 3D CNNs, LSTM and the parameters in MTL model will be end-to-end trained according to the loss function of the proposed MTL architecture.

IV. DEEP MODELS

A. 3D CNNs

Our goal is to gain a feature sequence representation for each video. Therefore, we split each video into several video clips. Each video is split into 25 clips, and 3D CNNs are applied into all those 25 video clips to generate 25 feature vectors for one video.

In our experiments, we use an efficient 3D convolutional neural network (C3D [57]) to extract feature of all the video clips. C3D is an efficient 3D CNN model to learning spatiotemporal features. It has 8 convolution layers, 5 pooling

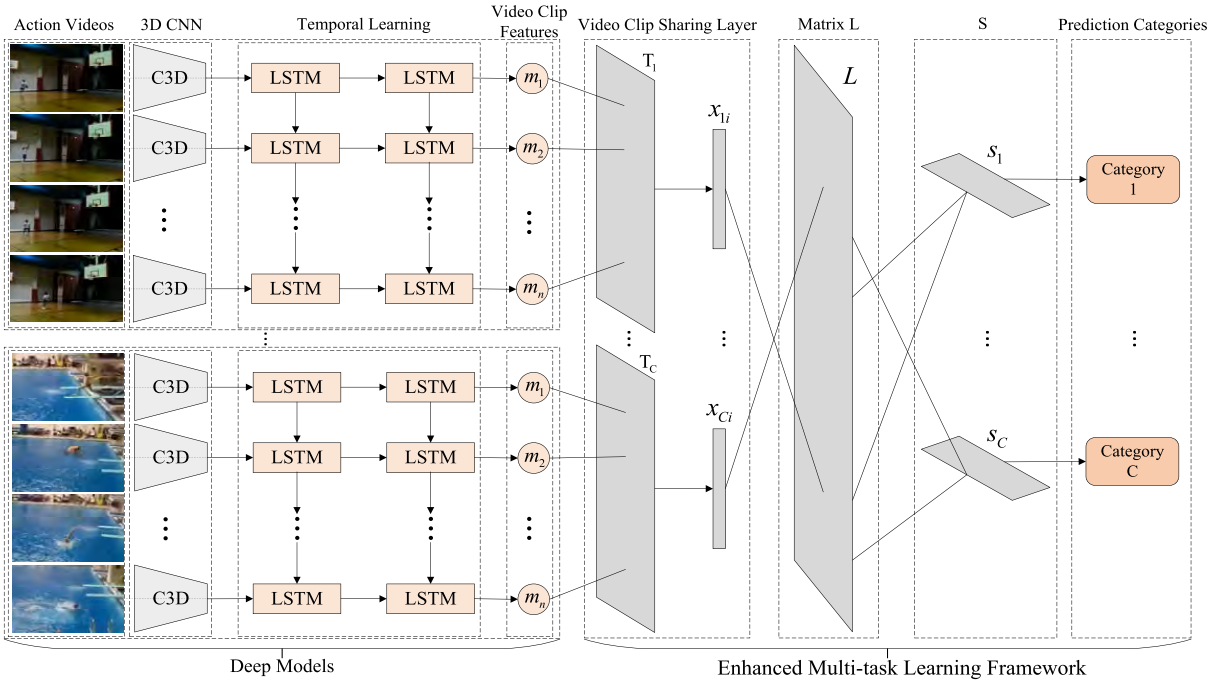


FIGURE 1. The overall structure of our system. First, we train 3D CNN and LSTM models to compute features from action videos as the representations of video. We apply C3D on video frames. $\{m_1, m_2,\ldots, m_n\}$ are the learned feature sequence by two LSTM layers of each video. Then, we build up a weight matrix T_c to get the weighted mean of the feature sequence generated from the LSTM model. We build $\{T_1, T_2,\ldots, T_C\}$ for all the c action categories. Matrix L contain the latent task of all the action categories C. S is a matrix, and its columns $\{s_1, s_2,\ldots, s_C\}$ represent the weights of linear combination for each category.

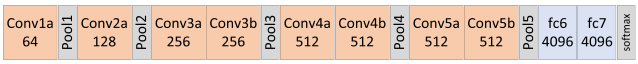


FIGURE 2. The structure of C3D network.

layers, followed by 2 fully connected layers, and a softmax output layer. The network architecture is presented in Fig. 2. Actually, the overall structure of C3D is quite similar to VGG-16 [58] but replaces all the 2D convolutional kernels with 3D convolutional kernels. All the 3D convolution kernels are $3 \times 3 \times 3$ with stride $1 \times 1 \times 1$. All the 3D pooling layers are $2 \times 2 \times 2$ with stride $2 \times 2 \times 2$ except for pool1 which has kernel size of $1 \times 2 \times 2$ and stride $1 \times 2 \times 2$ with the intention of preserving the temporal information in the early phase [57]. Each fully connected layer has 4096 output units. C3D requires the input size of $16 \times 112 \times 112$, so we will randomly select 16 frames in all the video clips and resize all the frames to 112×112.

B. LSTM MODELS

As mentioned above, we need to train a binary classifier for each action category. Fig. 3 shows the structure of the binary classifier for each category. We apply the LSTM network for binary classifiers, which is one of the recurrent neural network (RNN) models. The RNN is a neural network that is specialized for processing sequential inputs. RNNs can learn complex temporal dynamics by mapping input sequences to a sequence of hidden states, and hidden states to outputs. However, there exists the vanishing and exploding gradients problem when vanilla RNNs learn the long-term dynamics [60]. Then LSTM network provides a mechanism to forget the old state in its forget gate.

In our work, we build a two-layer LSTM network for each category to learn the temporal dynamic of videos based on features generated by the last pooling layer of C3D networks. In Fig. 3, $\{f_1, f_2,\ldots, f_n\}$ are the $n = 25$ features computed by C3D from 25 video clips in each video. Thus, from an input sequence $\{f_1, f_2,\ldots, f_n\}$, the memory cells in the two LSTM layers will produce a representation sequence $\{m_1, m_2,\ldots, m_n\}$. Next, we calculate the entry-wise products of u_i and this representation sequence m_i $(i = 1, 2, \ldots, n)$. $\{u_1, u_2,\ldots, u_n\}$ are all initialized to 1 and will be updated when train the binary classifiers. F represents the mean pooling feature vector using those entry-wise products, which is expressed as:

$$F = \frac{u_1 \cdot m_1 + u_2 \cdot m_2 + \ldots + u_n \cdot m_n}{n}. \quad (1)$$

Finally, this feature vector F will be fed into a logistic layer to complete the binary classifier for each action category, and w is the parameter vector of the last logistic regression layer. All the binary classifiers for each category are trained on data of each category, and all follow this process of training.

Since the feature sequence $\{m_1, m_2,\ldots, m_n\}$ generated by two LSTM layers represents the spatio-temporal features of 25 video clips in each video, the parameters $\{u_1, u_2,\ldots, u_n\}$ stores the weights of 25 video clips for each video.

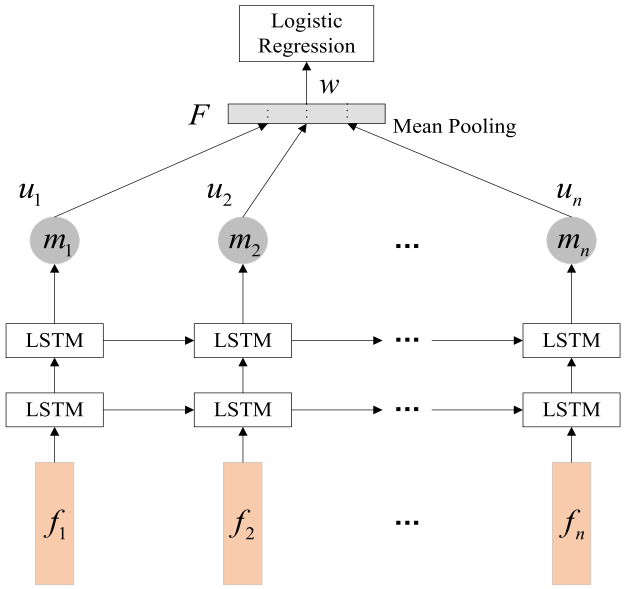


FIGURE 3. The structure of the binary classifier for each category. $\{f_1, f_2, \ldots, f_n\}$ are the $n = 25$ features which is computed by C3D from 25 video clips in each video. F represents the mean pooling feature vector using the weighted features $\{m_1, m_2, \ldots, m_n\}$, which are learned by two LSTM layers. w is the parameter vector of the last logistic regression layer.

The parameter w in the logistic regression layer is the classifier weight. Then, the weighted parameters $\{u_1, u_2, \ldots, u_n\}$ of video vector F and parameter w of logistic regression layer will be used to initialize the parameters of the following multi-task learning layer.

V. ENHANCED MULTI-TASK LEARNING MECHANISM

In the following subsections, we describe our novel multi-task learning mechanism. As the best of our knowledge, it is the first time to adopt the hybrid model of CNN and LSTM into the MTL mechanism, tailored to share knowledge among videos. As shown in Fig. 1, after completing the training of all binary classifiers for all categories, we take the outputs $\{m_1, m_2, \ldots, m_n\}$ of the last LSTM layer as the feature representation of each video. Those video feature representations will be used to train the whole network of our multi-task layer. Meanwhile, we show how to share the similar information across all categories, and limit the transmission of dissimilar information through norm regularization. Moreover, the alternative optimization strategy used to train the MTL framework is also explained in the following subsections.

A. LEARNING VIDEO CLIPS SHARING

Suppose we have C action categories and $A_c = \{(x_{ci}, y_{ci}) : i \in \{1, 2, \ldots, N_c\}\}$ be the training set for each task $c \in \{1, 2, \ldots, C\}$. N_c denotes the number of training examples of action categories c, whilst x_{ci} and y_{ci} represent the training data and labels of the training example i in category c. Since it is a binary classifier for each category, y_{ci} belongs to $\{0, 1\}$. Suppose x_{ci} represents the output generated from the last LSTM layer in each binary classier, such that we will have $\sum_{i=1}^{C} N_c$ training examples as the input for our multi-task learning network. In previous works of MTL framework [31], [32], x_{ci} is a fixed one-dimensional vector to represent each training example. However, it can not share the partial information of the whole sequential inputs. Considering videos are time-series data, we calculate the entry-wise products of weight parameters t_c^i and the sequential feature m_{ci}^i $(i = 1, 2, \ldots, n$, in our experiment we choose $n = 25$ video clips) for each video example, and x_{ci} (the training example i in category c) is the mean pooling of those entry-wise products, which is expressed as

$$x_{ci} = \frac{t_c^1 \cdot m_{ci}^1 + t_c^2 \cdot m_{ci}^2 + \ldots + t_c^n \cdot m_{ci}^n}{n}, \tag{2}$$

where $\{m_{ci}^1, m_{ci}^2, \ldots, m_{ci}^n\}$ denote the outputs of the second LSTM layer of input x_{ci}. n represents the number of video clips we choose in each video. $\{t_c^1, t_c^2, \ldots, t_c^n\}$ are the parameters of entry-wise products with sequential feature of LSTM. The parameters of $\{t_c^1, t_c^2, \ldots, t_c^n\}$ are the same for each category, initialized by the parameters of $\{u_1, u_2, \ldots, u_n\}$ of the corresponding binary classifier in Fig. 3. We use l to denote the length of m_{ci}^i, which refers to the length of the feature generated by LSTM. Since x_{ci} is the mean pooling of the entry-wise products of t_c^i and m_{ci}^i $(i = 1, 2, \ldots, n)$, the length d of x_{ci} should be the same as m_{ci}^i $(d = l)$. We stack $\{t_c^1, t_c^2, \ldots, t_c^n\}$ as columns of a matrix T_c, and $\{m_{ci}^1, m_{ci}^2, \ldots, m_{ci}^n\}$ as columns of a matrix M_{ci}. For simplicity, we represent Eq. 2 as

$$x_{ci} = T_c \odot M_{ci}, \tag{3}$$

where $T_c \odot M_{ci} = \frac{t_c^1 \cdot m_{ci}^1 + t_c^2 \cdot m_{ci}^2 + \ldots + t_c^n \cdot m_{ci}^n}{n}$. Since M_{ci} is the sequential feature representation of the input video, T_c can control the weights of different video clips in the final feature vector x_{ci}.

In our model, it is important to encourage partial video clip sharing across whole videos. To address this problem, we add the $\ell_{2,1}$ norm regularization as $\|T_c\|_{2,1}$ into the learning cost function of our whole multi-task learning network, which can encourage robust feature selection of T_c. Due to that $\ell_{2,1}$ norm can ensure the sparsity of the matrix T_c in row, $\|T_c\|_{2,1}$ can automatically select the video clip features of each video.

B. LEARNING LATENT TASKS SHARING

Our basic idea of sharing latent tasks in MTL is related with the works of Kumar and Daume [31] and Zhou *et al.* [32], but we enhance this framework to be well suited to process time-series data, which will be described in this subsection. As mentioned above, we will train a binary classifier for each action category. In Fig. 3, we use the parameter w to express the parameter vector of the last logistic regression layer. Consequently, for each category indexed by c, we define w_c as the parameter of last logistic regression layer. Suppose a weight matrix W for all action categories is made up of w_c $(c = 1, 2, \ldots, C)$, whose columns are stacked by those vectors.

Because of the length of the training examples x_{ci} is d, the size of matrix W will be $d \times C$.

We also want to construct classifiers for all categories in the multi-task layer as we did in the deep models. However, In the multi-task learning framework, we attempt to learn classifiers for all the action categories simultaneously instead of training classifiers independently in our deep models. In order to achieve this attempt, we make an important assumption that each classifier can be reconstructed from a number of shared latent tasks and each action category can be represented as a linear combination of these tasks. Let L denote the shared latent task matrix and S denote the matrix containing the weights of linear combination for each task. Then the weight matrix W can be decomposed as:

$$W = LS. \tag{4}$$

We assume there are k latent tasks for all the categories. In specific, L is a matrix of size $d \times k$, whose columns represent the latent task. At the same time, S is a matrix of size $k \times C$, whose columns s_c $(c = 1, 2, .., C)$ represent the weights of linear combination for each action category. In other words, the parameter w_c of our model for category c can be expressed as:

$$w_c = Ls_c. \tag{5}$$

The elements in s_c decide the weights of all the latent tasks for the action category c. Then different action categories are able to share similar visual information via the combination of the matrix L and S.

We apply norm regularization to learn a robust model and control the information sharing across all the categories. First, for the matrix L, we apply the Frobenius norm to improve the generalization ability of the system. It can avoid overfitting of our MTL model. Meanwhile, we use the ℓ_1 norm regularization of L to encourage the sparsity between latent tasks, which can control the information sharing across all the categories. Second, similarly, we control the sparsity patterns of matrix S to determine the number latent tasks sharing in different categories through the ℓ_1 norm regularization on S.

To gain the final cost function of the MTL layer, we first conduct the logistic regression of each binary classification in our deep model. As we mentioned in the previous section, we choose logistic regression for binary classifiers of each category in the last layer of LSTM. The logistic loss for $y_{ci} \in \{0, 1\}$ is defined as:

$$L(y_{ci}, w_c^T x_{ci}) = -\frac{1}{N_C}\left[\sum_{i=1}^{N_C} y_{ci} h_{w_c}(x_{ci}) + (1 - y_{ci})(1 - h_{w_c}(x_{ci}))\right], \tag{6}$$

where $h_{w_c}(x_{ci}) = 1/(1 + e^{-w_c^T x_{ci}})$. Due to Eq. 5, w_c can be expressed as Ls_c. Based on the above equations, the cost function can be expressed as the following form:

$$\min_{L,S} \sum_{c=1}^{C}\left[-\frac{1}{N_C}\left[\sum_{i=1}^{N_C} y_{ci} \log \frac{1}{1 + e^{-(Ls_c)^T x_{ci}}} + (1 - y_{ci}) \log(1 - \frac{1}{1 + e^{-(Ls_c)^T x_{ci}}})\right]\right] + \mu \|S\|_1 + \lambda \|L\|_F^2 + \gamma \|L\|_1, \tag{7}$$

where first term $\|S\|_1$ after the loss function is entry-wise ℓ_1 norm of the matrix S, enforcing the sparsity of the linear combination weight for each category. The parameter μ is the penalty term to control the level of sparsity in S. Similarly, the term $\|L\|_1$ denotes the ℓ_1 norm of the latent task matrix. At the same time, $\|L\|_F^2 = trace(LL^T)$ is the Frobenius norm of matrix L to avoid overfitting of our model. λ and γ are also the penalty terms of the Frobenius norm and ℓ_1 norm of L. We can build a linear classifier for each category through learning latent task matrix L and the combination weight matrix S in Eq. 7.

In this paper, we propose a novel MTL to enable the video clip information sharing. As shown in Eq. 3, every training example x_{ci} is the mean pooling of the columns of matrix T_c and M_{ci}. M_{ci} represents the feature matrix learning by our deep model of the training example i in category c, and T_c denotes video clip sharing weights for the action category c. We use the $\ell_{2,1}$ norm of matrix T_c to enforce the sparsity of the matrix T_c in row and automatically select the video clip feature of each video. Thus, based on Eq. 3 and Eq. 7, the cost function of our enhanced MTL framework takes the following form ultimately:

$$\min_{L,S,T} \sum_{c=1}^{C}\left[-\frac{1}{N_C}\left[\sum_{i=1}^{N_C} y_{ci} \log \frac{1}{1 + e^{-(Ls_c)^T (T_c \odot M_{ci})}} + (1 - y_{ci}) \log(1 - \frac{1}{1 + e^{-(Ls_c)^T (T_c \odot M_{ci})}})\right]\right] + \mu \|S\|_1 + \lambda \|L\|_F^2 + \gamma \|L\|_1 + \alpha \sum_{c=1}^{C} \|T_c\|_{2,1}. \tag{8}$$

Consequently, we can learn the latent task matrix L, the combination weight s_c for the action category c and the video clip sharing weights T_c for each category c to obtain a linear classifier for each category in our enhanced MTL framework.

C. ALTERNATIVE OPTIMIZATION STRATEGY

Since this cost function is not jointly convex in L, S and T_c, we implement an alternative optimization strategy to optimize Eq. 8. We notice that it is convex in L for fixed S and T_c, and is convex in S for fixed L and T_c. Likewise, if L and S are fixed, it becomes convex over T_c. Then, for the fixed L and T_c, the optimization problem in terms of s_c is described

as follows:

$$\min_{S}\sum_{c=1}^{C}\left[-\frac{1}{N_C}\left[\sum_{i=1}^{N_C}y_{ci}\log\frac{1}{1+e^{-(Ls_c)^T(T_c\odot M_{ci})}}+(1-y_{ci})\log(1-\frac{1}{1+e^{-(Ls_c)^T(T_c\odot M_{ci})}})\right]\right]+\mu\|S\|_1. \quad (9)$$

Similarly, for the fixed L and S, the optimization problem can be decomposed in individual problems for T_c:

$$\min_{T}\sum_{c=1}^{C}\left[-\frac{1}{N_C}\left[\sum_{i=1}^{N_C}y_{ci}\log\frac{1}{1+e^{-(Ls_c)^T(T_c\odot M_{ci})}}+(1-y_{ci})\log(1-\frac{1}{1+e^{-(Ls_c)^T(T_c\odot M_{ci})}})\right]\right]+\alpha\sum_{c=1}^{C}\|T_c\|_{2,1}. \quad (10)$$

After fixing the S and T_c, the optimization problem is in term of L as follows:

$$\min_{L}\sum_{c=1}^{C}\left[-\frac{1}{N_C}\left[\sum_{i=1}^{N_C}y_{ci}\log\frac{1}{1+e^{-(Ls_c)^T(T_c\odot M_{ci})}}+(1-y_{ci})\log(1-\frac{1}{1+e^{-(Ls_c)^T(T_c\odot M_{ci})}})\right]\right]+\lambda\|L\|_F^2+\gamma\|L\|_1. \quad (11)$$

Although Eq. 9, Eq. 10 and Eq. 11 are convex functions for their own optimization variables, stochastic gradient descent methods (SGD) are not capable of solving such optimization problems. Due to the ℓ_1 norm regularization terms existing in all the above equations, all the optimization problems are non-smooth convex functions. However, compared with other optimization methods like Accelerated Proximal Method (APG), SGD is very attractive because it often needs less training time. Then, in this paper, we apply a modified stochastic gradient method (SGD-$L1$) to optimize these equations. SGD-$L1$ handles the non-smooth convex functions by penalizing the weights according to cumulative values for ℓ_1 penalty [35].

Moreover, we initialize matrix L and T_c via the parameters w_c and $\{u_1^c, u_2^c, \ldots, u_n^c\}$ of our binary classifiers in Fig. 3 for the training data of category c. We stack w_c of each category c to the weight matrix W, and compute top-k singular vectors through $W = U\Sigma V^T$. Then the matrix L is initialized to the top-k left singular vectors of W. Meanwhile, S is randomly initialized. All the columns of T_c are initialized by $\{u_1^c, u_2^c, \ldots, u_n^c\}$ of each category c. To sum up, Algorithm 1 shows the steps of our optimization strategy. The VS-MTL algorithm aims to optimize Eq. 8 by the alternative optimization strategy.

During the training processing, the forward propagation will generate the input for the multi-task loss layer from all our deep models. Then, the output is the overall model weight matrix W and the video clip sharing matrix T_c, where each column in W will be dedicated to its specific corresponding CNN model and is taken back in the backpagation pass alongside the gradients with respect to its input. This weight matrix W will be decomposed into two matrices L and S, where knowledge sharing is already explored through MTL between all the CNN models via L. Meanwhile, the video clip information will be shared by the matrix T_c. All our deep models can collaborate together as the end-to-end training when optimizing the MTL layer, which will consume too much time. In our experiments, due to the large number of videos, we freeze the training of the deep models and optimize the MTL loss function using the outputs generated from the C3D+LSTM models.

Algorithm 1 VS-MTL: Learning Video Data Sharing in MTL

Input: Features generated from deep models: M_c
Labels 0, 1 of training data: y_{ci}
Output: Category predictors matrix L, S, T_c

```
Learn binary classifiers independently for each action category.
Initialize the matrix L with the top-k left singular vectors of W.
Initialize the columns of T with {u_1^c, u_2^c, ..., u_n^c} of each category c.
Initialize the matrix S randomly.
while not converged do
    for c = 1 to C do
        for i = 1 to N_c do
            Solve Eq. 9 to obtain s_t.
            Solve Eq. 10 to obtain T_c.
        end for
    end for
    Construct matrix S = [s_1 s_2 ... s_C].
    Fix S and T_c, and solve Eq. 11 to obtain L.
end while
Return: the matrix L,
        the matrix S,
        the matrixes T_c (c ∈ {1, 2, ..., C}).
```

VI. EXPERIMENTS

We first train and validate our model via two challenging human action datasets. Then, the experimental results are analyzed in detail to illuminate the effects of our enhanced MTL framework. The experimental results show that our method can efficiently not only share latent task information in multiple action categories but also share the video clip information.

A. DATASETS

There are several publicly available datasets for human action recognition. In our work, we use three most common datasets, KTH [17], UCF101 [51] and HMDB51 [52].

KTH Dataset: KTH [17] is a lightweight human actions dataset with six types ($C = 6$) of human actions (walking, jogging, running, boxing, hand waving and hand clapping) and 2391 sequences. Each sequence has 25fps frame rate,

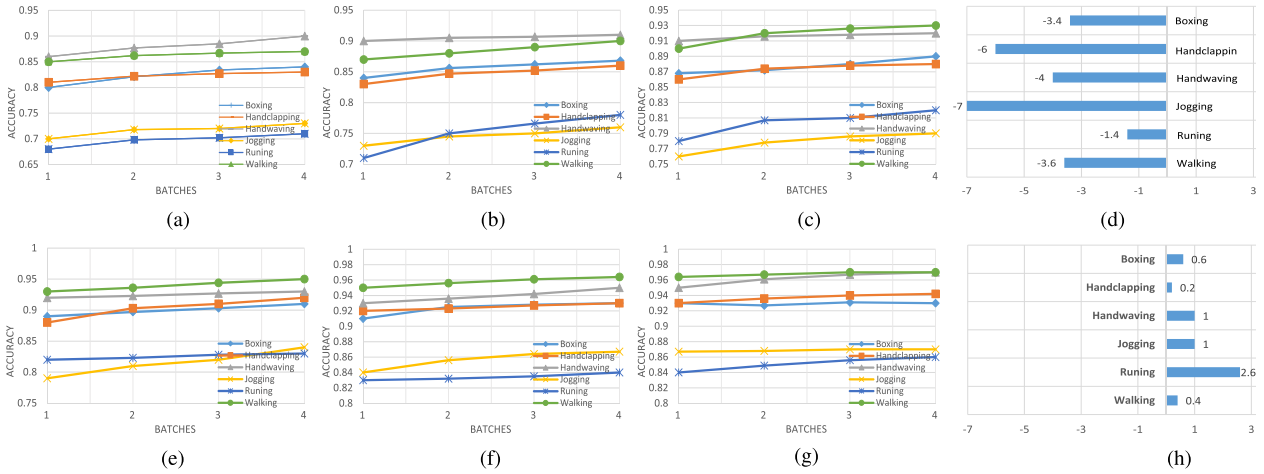


FIGURE 4. Prediction accuracy of the training process for our multi-task method. (a)-(d) belong to the first training epoch, and (e)-(h) belong to the second training epoch. For each row, the first figure is the training process of the parameter S. The second figure is the training process of the parameter T and the third figure is the training process of the parameter L. The last figure shows the gap of the final prediction accuracy of all categories compared with the single-task method. (a) Training on S. (b) Training on T. (c) Training on L. (d) Training on S. (e) Training on T. (f) Training on L.

a resolution of 160x120 pixels and a length of four seconds in average. Because of all sequences taken over homogeneous backgrounds with a static camera, KTH dataset is focused on the human body movements which would demonstrate the benefits of memory sharing. Besides, each action type is performed several times by 25 subjects in four different scenarios, and we split the dataset into a training set (8 persons), a validation set (8 persons) and a test set (9 persons) as the methods in [17].

UCF101 Dataset: UCF101 dataset [51] is an action dataset with realistic video clips, which is more challenging than KTH dataset because of camera motion, object appearance and pose, object scale, viewpoint, cluttered background, illumination conditions, etc. There are 101 action categories ($C = 101$) such as biking, high jump, playing piano, skiing, soccer juggling, etc., and about 13320 realistic video clips taken from youtube in total with a resolution of 320 × 240 pixels and a frame rate of 30 frames per second. For this dataset, we split the dataset into training and test set following "Three Train/Test Splits" [51] and repeat the experiment for 3 times.

HMDB51 Dataset: HMDB51 dataset [52] is collected from various sources, mostly from movies, and a small proportion from public databases such as the Prelinger archive, YouTube and Google videos. The dataset contains 6849 clips divided into 51 action categories, each containing a minimum of 101 clips. We also follow the evaluation strategy of three splits provided by the organizers for training and test data.

B. IMPLEMENTATION DETAILS

We implement our deep models and MTL framework based on Theano [61], including achieving stochastic gradient descent methods for $L1$ normalization (SGD-$L1$) [35] in Theano. All algorithms are performed on a server equipped with a 2.50 GHz Intel(R) Xeon(R) E5-2680 CPU, a NVIDIA(R) Tesla(TM) K80 GPU and 64G RAM.

1) DATA AUGMENTATION AND PREPROCESSING

We use several data preprocessing and augmentation strategies to preprocess videos on above datesets. For all datasets, we split each video into twenty-five clips as feature points in the time line, and in each clip, randomly select 16 frame in chronological order as the representation of this time slice, which means $n = 25$. Since videos are significantly more difficult to collect, data augmentation is used to reduce the effects of overfitting. Utilizing the temporal continuity of video, different frames are randomly selected in each video clip to generate new training examples.

2) DEEP MODEL TRAINING

We first extract video clip features via the 3D CNN (C3D), and our C3D model is pretrained on very huge video dataset which contains 1 million YouTube videos belonging to 487 classes. In our experiments, we use the output of last fully connected layer of C3D to represent video clip features, which is a 4096-dimensional vector. For each video, a sequence of 25 video clip features will be used as the input to the next two-layer LSTM network with 4096 hidden units in both layers to extract spatial features with size $k = 4096$. Then the output of the LSTM layer is multiplied by $\{u_1, u_2, \ldots, u_n\}$ with $n = 25$ and vector $u \in R^{4096}$, followed by a mean pooling and a logistic regression layer as illustrated in Fig. 3. The size of parameter w of logistic regression is 4096, which would be used to initialize matrix L in the MTL model.

When only training the deep models for binary classifiers, SGD (Stochastic Gradient Descent) is used with batch size 40 and learning rate 0.001. Besides, in order to prevent

over-fitting, we use some tricks such as ℓ_2 norm regularization with coefficient $\lambda = 0.0001$, dropout behind the LSTM layer, and early stop when the error does not decrease.

3) MTL SETTING AND OPTIMIZATION

After the training of binary classifiers, we set latent task number $k = 4096$, so the shared latent task matrix $L \in R^{4096\times4096}$ is initialized by the top 4096 left singular vectors of the weight matrix W stacked by each w_c. Besides, vector $s_c \in R^{4096}$ containing the weight of linear combination of latent tasks for the c-th single task is randomly initialized, and video clip sharing weight matrix $T_c \in R^{25\times4096}$ for the action category c is initialized by $\{u_1^c, u_2^c, \ldots, u_n^c\}$. Then we train the MTL model with $\mu = 0.01$ (the parameter of $\|S\|_1$), $\alpha = 0.001$ (the parameter of $\|T_c\|_{2,1}$), $\lambda = 0.001$ and $\gamma = 0.001$ (the ℓ_1 and Frobenius norm parameters of $\|L\|_1$), which are selected by the results of validation set.

C. EXPERIMENT RESULTS

1) EXPERIMENT RESULTS ON KTH

We test our method on KTH dataset at first, which just contains six types of human actions (walking, jogging, running, boxing, hand waving and hand clapping). It is not a very complex dataset. The experiment results are shown in Table 1 and 2. Single-task C3D + LSTM refers to our deep models without MTL layer, which are the 6 individual fine-tuned binary classifiers. To be specific, we input all the testing videos into the 6 individual fine-tuned binary classifiers for each category and gain the highest probability as the final prediction result. Multi-task C3D + LSTM is our whole framework, and all the binary classifiers can share information via our enhanced MTL framework. Different from the single-task model, the MTL framework shares knowledge among the 6 individual binary classifiers. As is shown in Table 1, we can see our multi-task model has gained a great performance compared with other methods (we chose the result in [27]). Meanwhile, the accuracy rate of our multi-task model raises more than 1% compared with the single-task model, which shows that our MTL layer is efficient to share the similar information on different action categories. We notice that there is a slight increase between the multi-task model and single-task model. This is due to this dataset is very small, and the accuracy rate is already quite high which will be hard to improve further. We can see our method achieves much better performance on a more complex dataset (UCF101) in the following.

TABLE 1. Comparison of Our Results Against Famous Action Recognition Methods on KTH.

Algorithm	KTH
Schüldt [17]	71.7%
Dollár [41]	81.2%
Niebles [65]	83.3%
Jhuang [16]	91.7%
Schindler [66]	92.7%
3D CNN [28]	90.2%
Single-task C3D + LSTM	92.3%
Multi-task C3D + LSTM	93.6%

The detailed accuracy rates of six types of human actions on KTH dataset are shown on Table 2. Compared with the performance of 3D CNN [27], our method performs better almost in all categories. We can see our framework shows the best in the action "Running". This may be because our framework can learn similar information from "Jogging" and "Walking". We also show the detailed training process of our MTL framework in Fig. 4. Since this is not a complex dataset, our model can be converged in two epochs in which the training process of parameters S, T and L is shown. In the beginning of training, the prediction accuracy declines compared with results of the single-task method. As showed in Fig. 4(d), our multi-task model still performs worse than the single-task method in the end of the first epoch. The prediction accuracies have a rise both in Fig. 4(b) and 4(f), which indicates the effect of our video clip sharing matrix T.

TABLE 2. Detailed Performance of Six Action Categories on KTH.

Algorithm	Boxing	Hand -clapping	Hand -waving	Jogging	Runing	Walking
3D CNN [28]	90%	94%	97%	84%	79%	97%
Single-task C3D + LSTM	94.5%	95%	97.3%	86%	84%	97%
Multi-task C3D + LSTM	95.4%	96.2%	97%	87.3%	87.8%	98%

2) EXPERIMENTAL RESULTS ON UCF101 AND HMDB51

UCF101 and HMDB51 are much harder than KTH since these two datasets contain much more videos and more complex background information. UCF101 includes more action categories and it is more difficult as compared with UCF50 [50] which is a subset of this dataset. As mentioned above, we first train the binary classifiers for each category in UCF101 and HMDB51, and directly use the features generated from those classifiers. After that, MTL framework is applied across all categories to share information in related tasks.

The overall performance of our method on UCF101 is shown in Table 3. Similarly with the setting in KTH, single-task C3D + LSTM refers to all the binary classifiers for each category, and the final prediction is made by the highest probability among those classifiers. Multi-task C3D + LSTM is our whole framework, and all the actions can share information through our enhanced MTL framework. As shown in Table 3, our model achieves competitive results compared with the state-of-art methods. The first line of table are the most representative traditional method for action recognition (iDT [15], [56]), and we also compare

FIGURE 5. Good and bad category prediction examples of UCF101 dataset are shown in (a) and (b). For each image in (a) and (b), we establish its top three action category prediction scores with our multi-task C3D + LSTM. The prediction scores are represented by the height of columns. Orange refers to the right action category for each image. (c) shows the changes of category prediction scores of multi-task C3D + LSTM with/without video clip information sharing. In the right below each image, they are action category prediction scores of our multi-task C3D + LSTM. In the left below each image, they are action category prediction scores of multi-task C3D + LSTM without video clip information sharing. Orange still refers to the right action category for each image. (a) Good categories prediction examples. (b) Bad categories prediction examples. (c) The performance of multi-task C3D + LSTM with/without video clip information sharing.

TABLE 3. Comparison of Our Results Against State-Of-The-Art Methods on UCF101 and HMDB51.

Algorithm	UCF101	HMDB51
iDT [15], [58]	84.7%	57.2%
C3D [59], [82]	85.2%	51.6%
Two-stream ConvNet [29]	88.1%	59.4%
Two-stream + LSTM [70]	88.6%	-
Temporal Pyramid Pooling CN [73]	89.1%	66.8%
Long-term temporal ConvNet [68]	91.7%	64.8%
Two-stream fusion [71]	92.5%	65.4%
$\text{LTC}_{Flow+RGB}$ + IDT [72]	92.7%	67.2%
Key-volume mining CNN [69]	93.1%	63.3%
L^2STM [83]	93.6%	66.2%
TSN (3 modalities) [74]	94.2%	69.4%
FV-VAE [75]	94.2%	-
Single-task C3D + LSTM	88.9%	63.6%
Multi-task C3D + LSTM	93.4%	68.9%

our MTL method with many up-to-date deep learning methods.

For the performance on UCF101 dataset, although our multi-task C3D + LSTM can achieve very competitive results (93.4%) among those state-of-the-art methods, the performance is still lower than that of TSN (94.2%) and FV-VAE (94.2%). However, those methods fuse several different deep models and use lots of tricks to gain such high accuracy, which consumes too much time and requires for huge computing resources. Also, for the performance on HMDB51, our multi-task C3D + LSTM achieves state-of-the-art accuracy (68.9%) which also sight worse than the TSN (69.4%). Even compared with some more recent works [70], [71], [81], our multi-task C3D + LSTM can also outperform those state-of-the-art. In this paper, we focus on proposing a novel MTL framework. As shown in Table 3, our multi-task C3D + LSTM method brings a clear raise (4.5% on UCF101 and 5.3%, respectively) compared with the single-task C3D + LSTM method.

Those results demonstrate that our method can efficiently share latent information in different action categories, and improves the performance of the single task learning methods.

To evaluate the effects of the video clip length in the LSTM layer, we test different settings (15 and 35) but not only 25 in UCF101. The multi-task C3D + LSTM with 15 and 35 video clips achieve 92.5% and 90.7%, respectively. We find that too short or too long video clip length will both reduce performance. Actually, too short video clip length can not fully share the video clip information. It makes our multi-task method degenerate into [32] when the video clip length is set to 1. Meanwhile, too long video chip length will share too much redundant information.

FIGURE 6. Classification accuracy of each category on UCF101 dataset. Blue indicates the classification accuracy of single-task C3D + LSTM and red indicates the classification accuracy of multi-task C3D + LSTM. Meanwhile, green indicates the classification accuracy of the multi-task C3D + LSTM without video clip information sharing.

3) DETAILED ANALYSIS ON UCF101

We show the evaluation of some actions of UCF101 datasets in Fig. 5. Good and bad category prediction examples are shown in Fig. 5(a) and Fig. 5(b). For each image, we establish the top three action category prediction scores with our multi-task C3D + LSTM. It can be seen that similar actions can disturb the prediction of classifiers. For example, as ApplyEyeMakeup and ApplyLipstick are very similar which both belong to the make-up action, our classifiers may feel confused.

Meanwhile, we show the classification accuracy of all category on UCF101 dataset in Fig. 6. This figure shows the results in detail including the performance of the single-task C3D + LSTM and multi-task C3D + LSTM. Meanwhile, we also add the comparison of multi-task C3D + LSTM without our video clip information sharing method. The detailed analysis of the comparison of multi-task C3D + LSTM without video clip information sharing will be introduced in the following section, while we focus on the comparison of single-task and multi-task C3D + LSTM here. It can be seen from Fig. 6 that the classification accuracy has been greatly improved on almost all the categories with 6% gain in average, especially on the related categories. For example, ApplyEyeMakeup and ApplyLipstick both belong to the make-up action which are related to each other, thus their performances raise more than 15% gain due to MTL sharing. Other related categories like PlayingGuitar, PlayingPiano and PlayingViolin share the same rules. On the contrary, the performance of a few categories (like Biliards and Surfing) even decrease, perhaps because that there are some useless interference information shared among categories.

D. EFFECT OF VIDEO CLIP INFORMATION SHARING MECHANISM

To explain the influence of our MTL mechanism especially our video clip information sharing mechanism, we conduct two experiments: 1) removing our proposed layer for sharing video clip information, and 2) comparing the proposed model with other efficient MTL methods. For the first setting, we remove the T_c matrix in our model to inactivate our video clip information sharing mechanism. We test this model on UCF101 and show the results in Fig. 5(c) and Fig. 6. In Fig. 5(c), we show some examples of the differences of categories prediction scores among multi-task C3D + LSTM with/without video clip information sharing. We can see that some wrong predictions from the multi-task C3D + LSTM

without video clip information sharing become right because our video clip information sharing mechanism can learn more detailed information among similar actions. Moreover, the detailed results for each category in UCF101 are shown in Fig. 6. Orange columns indicate the classification accuracy of our multi-task C3D + LSTM and gray columns indicate the classification accuracy of the multi-task C3D + LSTM without video clip information sharing. This figure prove the video clip sharing mechanism can be efficient to improve the performance of the MTL method. It is clear that almost all prediction accuracy rates of categories decline. However, there also exist some abnormal examples like BabyCrawling and BrushingTeeth. For those examples, the multi-task C3D + LSTM without video clip sharing even shows better than the framework with the video clip sharing. The overall performance is shown in Table 4. We can see the performance of the multi-task learning framework without the video clip information sharing mechanism drops approximately 2% than our model.

TABLE 4. Effect of Video Clip Information Sharing.

Algorithm	Performance
Multi-task C3D + LSTM without video clip information sharing	91.1%
Multi-task C3D + LSTM with video clip information sharing	93.4%

TABLE 5. Comparison of Different MTL Methods.

Algorithm	Performance
GO-MTL [32]	88.6%
MTL with Sharing Latent Tasks [33]	90.3%
Our method	**93.4%**

For the second setting, we compare our model with other MTL methods on the UCF101 dataset in Table 5. For a fair comparison, all the methods in Table 5 use the same C3D + LSTM model to extract the video features as the input for the MTL framework. Actually, our method and the MTL with Sharing Latent Tasks of [32] both share the basic MTL framework of GO-MTL in [31] which can be regarded as the baseline model. The cost function of our enhanced MTL framework is presented in Eq. 8. Compared with the MTL with Sharing Latent Tasks of [32], our method adds the video clip sharing mechanism with the $\ell_{2,1}$ norm of $\alpha \sum_{c=1}^{C} \|T_c\|_{2,1}$. At the same time, the MTL with Sharing Latent Tasks in [32] adds the $\gamma\|L\|_1$ on the baseline GO-MTL of [31]. Since the two comparison method do not have the video clip sharing mechanism, we directly use the mean average of the 25 feature vectors as the single feature representation of each video. As shown in this table, we can see the effects of different regularization items in the MTL framework. It shows that our MTL algorithm is quite efficient for sharing knowledge among the video data.

VII. CONCLUSION

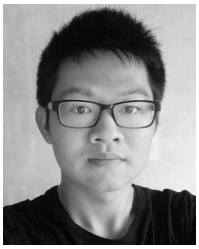
We have proposed a novel MTL mechanism to allow video clip information shared across related tasks instead of the whole information of videos. To the best of our knowledge, it is the first time to combine a spatiotemporal deep models (3D CNN plus LSTM) together with the MTL method. Meanwhile, we propose the video clip sharing mechanism tailored to information sharing among videos. We evaluate the proposed enhanced MTL method on two popular video datasets and achieve competitive results compared with the state-of-art methods on those datasets, which demonstrates that our method can encourage the efficient knowledge sharing of similar parts of action videos. Moreover, the proposed MTL method is not limited on the video data but can be applied for all the spatiotemporal data. For example, it can be also used to deal with the medical images like the CT or MRI images as 3D CNNs are proved to be suitable for these inputs.

REFERENCES

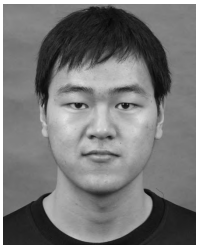
[1] A. Torralba, K.-P. Murphy, and W.-T. Freeman, "Sharing features: Efficient boosting procedures for multiclass object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, vol. 2, Jul. 2004, pp. 762–769.
[2] J. Zhang, Z. Ghahramani, and Y. Yang, "Learning multiple related tasks using latent independent component analysis," in *Proc. Adv. Neural Inf. Process. Syst.*, 2005, pp. 1585–1592.
[3] T. Evgeniou and M. Pontil, "Regularized multi–task learning," in *Proc. SIGKDD*, Aug. 2004, pp. 109–117.
[4] P. Rai and H. Daumé III, "Infinite predictor subspace models for multitask learning," in *Proc. AISTATS*, Mar. 2010, pp. 613–620.
[5] L. Jacob, J.-P. Vert, and F.-R. Bach, "Clustered multi-task learning: A convex formulation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 745–752.
[6] Z. Kang, K. Grauman, and F. Sha, "Learning with whom to share in multi-task feature learning," in *Proc. Int. Conf. Mach. Learn.*, Jun. 2011, pp. 521–528.
[7] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld, "Learning realistic human actions from movies," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2008, pp. 1–8.
[8] J. Hou, X. Wu, Y. Sun, and Y. Jia, "Content-attention representation by factorized action-scene network for action recognition," *IEEE Trans. Multimedia*, vol. 20, no. 6, pp. 1537–1547, Jun. 2017.
[9] X. Zhen, F. Zheng, L. Shao, X. Cao, and D. Xu, "Supervised local descriptor learning for human action recognition," *IEEE Trans. Multimedia*, vol. 19, no. 9, pp. 2056–2065, Sep. 2017.
[10] Y. Shi, Y. Tian, Y. Wang, and T. Huang, "Sequential deep trajectory descriptor for action recognition with three-stream CNN," *IEEE Trans. Multimedia*, vol. 19, no. 7, pp. 1510–1520, Jul. 2017.
[11] Y. Yang, C. Deng, S. Gao, W. Liu, D. Tao, and X. Gao, "Discriminative multi-instance multitask learning for 3D action recognition," *IEEE Trans. Multimedia*, vol. 19, no. 3, pp. 519–529, Mar. 2017.
[12] S. Wang, Z. Ma, Y. Yang, X. Li, C. Pang, and A.-G. Hauptmann, "Semi-supervised multiple feature analysis for action recognition," *IEEE Trans. Multimedia*, vol. 16, no. 2, pp. 289–298, Feb. 2014.
[13] X. Cai, W. Zhou, L. Wu, J. Luo, and H. Li, "Effective active skeleton representation for low latency human action recognition," *IEEE Trans. Multimedia*, vol. 18, no. 2, pp. 141–154, Feb. 2016.
[14] M.-A. Tahir *et al.*, "A robust and scalable visual category and action recognition system using kernel discriminant analysis with spectral regression," *IEEE Trans. Multimedia*, vol. 15, no. 7, pp. 1653–1664, Nov. 2013.
[15] H. Wang and C. Schmid, "Action recognition with improved trajectories," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 3551–3558.
[16] Y. Yang, J. Song, Z. Huang, Z. Ma, N. Sebe, and A. G. Hauptmann, "Multi-feature fusion via hierarchical regression for multimedia analysis," *IEEE Trans. Multimedia*, vol. 15, no. 3, pp. 572–581, Apr. 2013.

[17] C. Schuldt, I. Laptev, and B. Caputo, "Recognizing human actions: A local SVM approach," in *Proc. 17th Int. Conf. Pattern Recognit.*, vol. 3. Aug. 2004, pp. 32–36.
[18] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2005, vol. 1, no. 1, pp. 886–893.
[19] R. Chaudhry, A. Ravichandran, G. Hager, and R. Vidal, "Histograms of oriented optical flow and binet-cauchy kernels on nonlinear dynamical systems for the recognition of human actions," in *Proc. IEEE Int. Conf. Comput. Vis.*, Jun. 2009, pp. 1932–1939.
[20] N. Dalal, B. Triggs, and C. Schmid, "Human detection using oriented histograms of flow and appearance," in *Proc. Eur. Conf. Comput. Vis.*, 2006, pp. 428–441.
[21] D.-A. Freedman, *Statistical Models: Theory and Practice*. Cambridge, U.K.: Cambridge Univ. Press, 2009.
[22] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.
[23] A. Krizhevsky, I. Sutskever, and G.-E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst*, 2012, pp. 1097–1105.
[24] X. Wang, L. Gao, P. Wang, X. Sun, and X. Liu, "Two-stream 3-D convNet fusion for action recognition in videos with arbitrary size and length," *IEEE Trans. Multimedia*, vol. 20, no. 3, pp. 634–644, Mar. 2017.
[25] H.-J. Kim, J.-S. Lee, and H.-S. Yang, "Human action recognition using a modified convolutional neural network," in *Proc. Int. Symp. Neural Netw.*, 2007, pp. 715–723.
[26] C. Jia, M. Shao, and Y. Fu, "Sparse canonical temporal alignment with deep tensor decomposition for action recognition," *IEEE Trans. Image Process.*, vol. 26, no. 2, pp. 738–750, Feb. 2017.
[27] S. Ji, W. Xu, M. Yang, and K. Yu, "3D convolutional neural networks for human action recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 221–231, Jan. 2013.
[28] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 568–576.
[29] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[30] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Cognit. Modeling*, vol. 5, no. 3, p. 1, 1988.
[31] A. Kumar and H. Daume, "Learning task grouping and overlap in multi-task learning," in *Proc. Int. Conf. Mach. Learn.*, Jun. 2012, pp. 1383–1390.
[32] Q. Zhou, G. Wang, K. Jia, and Q. Zhao, "Learning to share latent tasks for action recognition," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 2264–2271.
[33] Y. Yan, E. Ricci, G. Liu, and N. Sebe, "Egocentric daily activity recognition via multitask clustering," *IEEE Trans. Image Process.*, vol. 24, no. 10, pp. 2984–2995, Oct. 2015.
[34] A.-H. Adbulnabi, G. Wang, J. Lu, and K. Jia, "Multi-task CNN model for attribute prediction," *IEEE Trans. Multimedia*, vol. 17, no. 11, pp. 1949–1959, Nov. 2015.
[35] Y. Tsuruoka, J.-I. Tsujii, and S. Ananiadou, "Stochastic gradient descent training for l1-regularized log-linear models with cumulative penalty," in *Proc. Annu. Meeting Assoc. Comput. Linguistics*, Aug. 2009, pp. 477–485.
[36] K.-C. Toh and S. Yun, "An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems," *Pacific J. Optim.*, vol. 6, no. 3, pp. 615–640, Nov. 2010.
[37] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2. Sep. 1999, pp. 1150–1157.
[38] H. Bay, T. Tuytelaars, and L. van Gool, "Surf: Speeded up robust features," in *Proc. Eur. Conf. Comput. Vis.*, 2006, pp. 404–417.
[39] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proc. IEEE Int. Conf. Comput. Vis.*, Nov. 2011, pp. 2564–2571.
[40] P. Dollär, V. Rabaud, G. Cottrell, and S. Belongie, "Behavior recognition via sparse spatio-temporal features," in *Proc. IEEE Int. Workshop Vis. Surveill. Perform. Eval. Tracking Surveill.*, Oct. 2005, pp. 65–72.
[41] A. Klaser, M. Marszaék, and C. Schmid, "A spatio-temporal descriptor based on 3d-gradients," in *Proc. Brit. Mach. Vis. Conf.*, vol. 275, Sep. 2008, pp. 1–10.
[42] H. Wang and A. Kläser, C. Schmid, and C.-L. Liu, "Action recognition by dense trajectories," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2011, pp. 3169–3176.
[43] R. Messing, C. Pal, and H. Kautz, "Activity recognition using the velocity histories of tracked keypoints," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2009, pp. 104–111.
[44] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.
[45] Z. Ma, H. Yu, W. Chen, and J. Guo, "Short utterance based speech language identification in intelligent vehicles with time-scale modifications and deep bottleneck features," *IEEE Trans. Veh. Technol.*, vol. 68, no. 1, pp. 121–128, Jan. 2019.
[46] Z. Ma, Y. Lai, W. B. Kleijn, L. Wang, Y. Z. Song, and J. Guo, "Variational Bayesian learning for dirichlet process mixture of inverted dirichlet distributions in non-Gaussian image feature modeling," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 2, pp. 449–463, Feb. 2018.
[47] Z. Ma, J.-H. Xue, A. Leijon, Z.-H. Tan, Z. Yang, and J. Guo, "Decorrelation of neutral vector variables: Theory and applications," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 1, pp. 129–143, Jan. 2018.
[48] Z. Ma, A. E. Teschendorff, A. Leijon, Y. Qiao, H. Zhang, and J. Guo, "Variational Bayesian matrix factorization for bounded support data," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 4, pp. 876–889, Apr. 2015.
[49] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[50] K. K. Reddy and M. Shah, "Recognizing 50 human action categories of Web videos," *Mach. Vis. Appl.*, vol. 24, no. 5, pp. 971–981, 2013.
[51] K. Soomro, A.-R. Zamir, and M. Shah. (2012). "UCF101: A dataset of 101 human actions classes from videos in the wild." [Oline]. Available: https://arxiv.org/abs/1212.0402
[52] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "HMDB: A large video database for human motion recognition," in *Proc. IEEE Int. Conf. Comput. Vis.*, Nov. 2011, pp. 1–10.
[53] Y. J. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 4694–4702.
[54] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 1725–1732.
[55] N. Srivastava, E. Mansimov, and R. Salakhutdinov, "Unsupervised learning of video representations using lstms," in *Proc. Int. Conf. Mach. Learn.*, Jun. 2015, pp. 843–852.
[56] H. Wang and C. Schmid, "Lear-inria submission for the thumos workshop," in *Proc. ICCV Workshop Action Recognit. Large Number Classes*, Jan. 2013, pp. 1–8.
[57] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3D convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 4489–4497.
[58] K. Simonyan and A. Zisserman. (2014). "Very deep convolutional networks for large-scale image recognition." [Online]. Available: https://arxiv.org/pdf/1409.1556
[59] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.
[60] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," Tech. Rep., 2001.
[61] J. Bergstra *et al.*, "Theano: A CPU and GPU math compiler in python," in *Proc. Python Sci. Conf.*, Jun. 2010, pp. 1–7.
[62] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 1–15.
[63] J. C. Niebles, W. Hongcheng, and L. Fei-Fei, "Unsupervised learning of human action categories using spatial-temporal words," *Int. Jour. Comput. Vis.*, vol. 79, no. 3, pp. 299–318, Sep. 2008.
[64] K. Schindler and L. van Gool, "Action snippets: How many frames does human action recognition require," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2008, pp. 1–8.
[65] L. Sun, K. Jia, D.-Y. Yeung, and B. E. Shi, "Human action recognition using factorized spatio-temporal convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 45–65.
[66] G. Varol, I. Laptev, and C. Schmid. (2016). "Long-term temporal convolutions for action recognition." [Online]. Available: https://arxiv.org/pdf/1409.1556
[67] W. Zhu, J. Hu, G. Sun, X. Cao, and Y. Qiao, "A key volume mining deep framework for action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 1991–1999.

[68] J. Y.-H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Dec. 2015, pp. 169–179.
[69] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 1933–1941.
[70] V. Gül, L. Ivan, and S. Cordelia, "Long-term temporal convolutions for action recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 6, pp. 1510–1017, Jun. 2017.
[71] P. Wang, Y. Cao, C. Shen, L. Liu, and H. T. Shen, "Temporal pyramid pooling-based convolutional neural network for action recognition," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27, no. 12, pp. 2613–2622, Dec. 2018.
[72] L. Wang *et al.*, "Temporal segment networks: towards good practices for deep action recognition," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 404–417.
[73] Z. Qiu, T. Yao, and T. Mei, "Deep quantization: encoding convolutional activations with deep generative model," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Sep. 2017, pp. 145–165.
[74] Y. Yang, Z. Ma, Y. Yang, F. Nie, and H. T. Shen, "Multitask spectral clustering by exploring intertask correlation," *IEEE Trans. Cybern.*, vol. 45, no. 5, pp. 1083–1094, 2015.
[75] C. Zhang, X. Ouyang, and P. Patras, "ZipNet-GAN: Inferring fine-grained mobile traffic patterns via a generative adversarial neural network," in *Proc. 13th Int. Conf. Emerg. Netw. EXperiments Technol.*, Nov. 2017, pp. 363–375.
[76] X. Ouyang *et al.*, "Audio-visual emotion recognition using deep transfer learning and multiple temporal models," in *Proc. ICMI*, Sep. 2017, pp. 577–582.
[77] L. Ding, W. Fang, H. Luo, L. Peter, B. Zhong, and X. Ouyang, "A deep hybrid learning model to detect unsafe behavior: Integrating convolution neural networks and long short-term memory," *Autom. Construct.*, vol. 86, pp. 118–124, Feb. 2018.
[78] X. Ouyang, Y. Cheng, Y. Jiang, C.-L. Li, and P. Zhou. (2018). "Pedestrian-synthesis-GAN: Generating pedestrian data in real Scene and beyond." [Online]. [Online]. Available: https://arxiv.org/abs/1804.02047
[79] Y. Yang, Z.-J. Zha, Y. Gao, X. Zhu, and T.-S. Chua, "Exploiting Web images for semantic video indexing via robust sample-specific loss," *IEEE Trans. Multimedia*, vol. 16, no. 6, pp. 1677–1689, Oct. 2014.
[80] S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy. (2017). "Rethinking spatiotemporal feature learning for video understanding." [Online]. [Online]. Available: https://arxiv.org/abs/:1712.04851
[81] L. Sun, K. Jia, K. Chen, D.-Y. Yeung, B.-E. Shi, and S. Savarese, "Lattice long short-term memory for human action recognition," in *Proc. IEEE Int. Conf. Comput. Vis.*, Jul. 2017, pp. 145–175.



XI OUYANG received the B.S. degree from the School of Electronic Information and Communications, Huazhong University of Science and Technology (HUST), Wuhan, China, in 2015, and the M.S. degree from the Department of Electronics and Information Engineering, HUST, in 2018. He is currently pursuing the Ph.D. degree with the School of Biomedical Engineering, Med-X Research Institute, Shanghai Jiao Tong University, Shanghai, China. He was a Research Intern with the Panasonic Research and Development Center Singapore, from 2016 to 2017, where he focused on deep learning research. His current research interests include medical image analysis, deep learning, and machine learning.



SHUANGJIE XU received the B.S. degree from the Department of Electronics and Information Engineering, Huazhong University of Science and Technology, Wuhan, China, in 2016. He is currently pursuing the M.S. degree with the School of Electronic Information and Communications, Huazhong University of Science and Technology. His research interests include computer vision, machine learning, and deep learning as they apply to image and video analysis and understanding.

CHAOYUN ZHANG received the B.S. degree from the School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan, and the M.S. degree from the School of Informatics, The University of Edinburgh, Edinburgh, U.K., where he is currently pursuing the Ph.D. degree. His current research interests include deep learning, mobile big data, and pattern recognition.

PAN ZHOU received the B.S. degree (Hons.) in the advanced class from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2006, the M.S. degree from the Department of Electronics and Information Engineering, HUST, in 2008, and the Ph.D. degree from the School of Electrical and Computer Engineering, Georgia Institute of Technology (Georgia Tech), Atlanta, USA, in 2011. He was a Senior Technical Member with Oracle America Inc., Boston, MA, USA, from 2011 to 2013, where he focused on Hadoop and distributed storage system for big data analytics with the Oracle Cloud Platform. He is currently an Associate Professor with the School of Electronic Information and Communications, HUST. His current research interests include communication and information networks, security and privacy, machine learning, and big data. He received the Merit Research Award from HUST for his master study.

YANG YANG received the bachelor's degree from Jilin University, in 2006, the master's degree from Peking University, in 2009, and the Ph.D. degree from The University of Queensland, Australia, in 2012, under the supervision of Prof. H. T. Shen and Prof. X. Zhou. During 2012–2014, he was a Research Fellow with the National University of Singapore, under the supervision of Prof. T.-S. Chua. He is currently with the University of Electronic Science and Technology of China.

GUANGHUI LIU received the M.Sc. and Ph.D. degrees in electronic engineering from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2002 and 2005, respectively. In 2005, he joined Samsung Electronics, South Korea, as a Senior Engineer. In 2009, he became an Associate Professor with the School of Electronics Engineering, UESTC, where currently he has been a Full Professor, since 2014. He is currently with the School of Information and Communication Engineering, UESTC. His general research interests include digital signal processing and telecommunications, with emphasis on digital video processing and transmission, machine learning, and OFDM techniques. In these areas, he has published tens of papers in refereed journals or conferences, and holds more than 40 patents (six U.S. granted patents). He served as the Publication Chair for the IEEE ISPACS-2010 and the IEEE VCIP-2016.

XUELONG LI (M'02–SM'07–F'12) is currently a Full Professor with the Center for Optical Imagery Analysis and Learning (OPTIMAL), School of Computer Science, Northwestern Polytechnical University, Xi'an, China.