The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Unsupervised Deep Learning of Mid-Level Video Representation for Action Recognition

Jingyi Hou,[1] Xinxiao Wu,[1] Jin Chen,[1] Jiebo Luo,[2] Yunde Jia[1]
1. Beijing Laboratory of Intelligent Information Technology, School of Computer Science,
Beijing Institute of Technology, Beijing 100081, China
2. Department of Computer Science, University of Rochester, Rochester NY 14627, USA

Abstract

Current deep learning methods for action recognition rely heavily on large scale labeled video datasets. Manually annotating video datasets is laborious and may introduce unexpected bias to train complex deep models for learning video representation. In this paper, we propose an unsupervised deep learning method which employs unlabeled local spatial-temporal volumes extracted from action videos to learn mid-level video representation for action recognition. Specifically, our method simultaneously discovers mid-level semantic concepts by discriminative clustering and optimizes local spatial-temporal features by two relatively small and simple deep neural networks. The clustering generates semantic visual concepts that guide the training of the deep networks, and the networks in turn guarantee the robustness of the semantic concepts. Experiments on the HMDB51 and the UCF101 datasets demonstrate the superiority of the proposed method, even over several supervised learning methods.

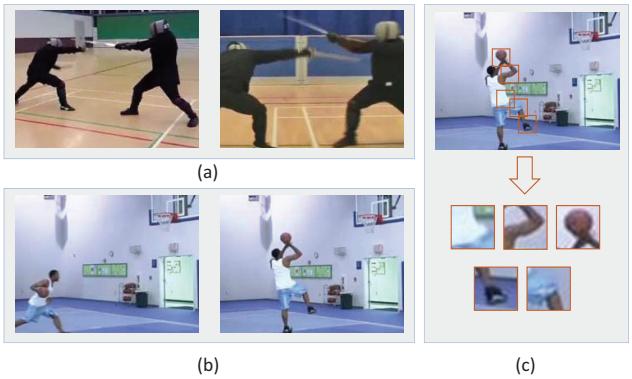

Figure 1: *Left*: Examples of problematic annotation (see text for explanation). *Right*: The proposed method learns mid-level action representation using local video volumes.

Introduction

Recently, representation learning via deep neural networks has proven to improve state-of-the-art performance dramatically for various computer vision tasks (Simonyan and Zisserman 2014; Szegedy et al. 2015; He et al. 2016; Lin et al. 2016). Despite the great success of deep neural networks on these image-based tasks, their superiorities are not that remarkable on video-based tasks, especially for the task of action recognition. While good performance of deep models depends on large amounts of labeled training samples, it is difficult to collect sufficient labeled data for action recognition because manually annotating a video is far more laborious and error-prone than an image.

Recently, more and more works focus on learning deep video representation in unsupervised ways without the need of annotated training data. Some approaches (Wiskott and Sejnowski 2002; Taylor et al. 2010; Le et al. 2011) use the convolutional operation to learn abstract and robust low-level spatio-temporal features of videos. Su et al. (2016) developed an unsupervised method to parse video sequences and hierarchically encoded the segmented features to represent actions instead of learning low-level features. Misra, Zitnick, and Hebert (2016) proposed to use an unsupervised sequential verification method to train a deep neural network for action recognition. The LSTM based methods (Du, Wang, and Wang 2015; Srivastava, Mansimov, and Salakhutdinov 2015) learn fixed-length global action features by reconstructing videos. All these methods of unsupervised learning for video representation are purely data-driven without considering the discriminative and semantic information in videos.

High quality labels can provide crucial semantic information and supposed to benefit learning more discriminative video representations. Actually, many classification criteria are so ambiguous that manually annotated action categories are perplexing and at different semantic levels. Using these kinds of labels as guidance to learn video representation might miss some spatial-temporal information of actions and even introduce bias. Here are two examples of unreasonable annotations: Figure 1(a) illustrates that different classes may in fact share similar meaning. The two frames are sampled from different videos which are classified as different actions, namely, "sword" and "fencing". Figure 1(b) shows the annotations at different semantic levels. The two frames are sampled from one video classified as "shoot ball" containing relatively low level semantic actions, "run" and "jump".

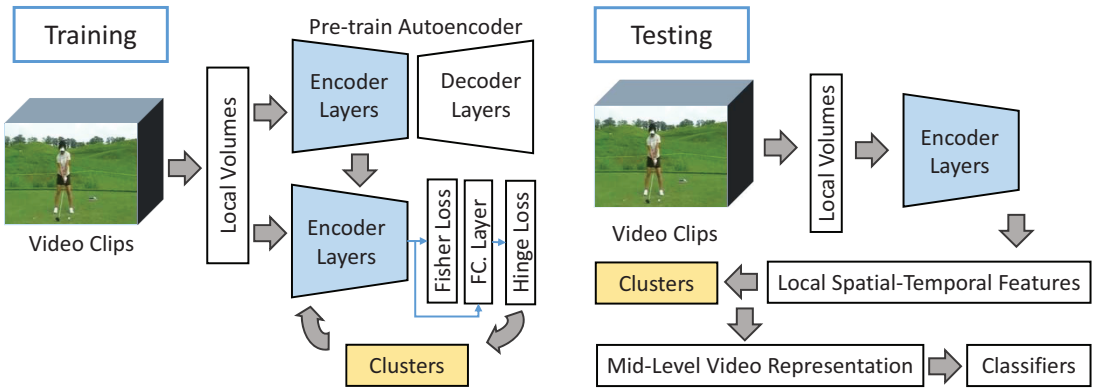

Figure 2: Overview of our approach. It simultaneously learns local spatial-temporal features and semantic concepts based on either local hand-crafted motion features or raw local volumes to generate mid-level video representation for action recognition.

In order to learn more meaningful and discriminative video representation, we propose an unsupervised video representation learning method which can automatically discover mid-level semantic concepts providing latent interpretations of action video volumes. First, the action videos are divided into local volumes containing simple and discriminative semantic information as shown in Figure 1(c). The local volumes are then clustered and used to train two deep neural networks (i.e., a fully connected network and a fully convolutional network) iteratively to find the most optimal latent semantic concepts. Finally, the mid-level representation of action videos is obtained by sharing these concepts. The loss function of the deep neural networks integrates the hinge and the Fisher (Kan, Shan, and Chen 2016) losses to find optimal video concepts. The hinge loss reduces the inter-category confusions and ensures the discriminative property of the local features, and the Fisher loss draws features into low dimensional space while keeping discriminative properties to acquire compact video representations. Moreover, using local volumes as the input reduces the size of the neural networks and thus saves the computation resources.

Overall, our main contributions are:

- We propose an unsupervised deep learning of mid-level video representation for action recognition. To the best of our knowledge, it is the first work which learns the deep *mid-level* video representation in an unsupervised manner;
- We develop a novel iterative clustering algorithm such that discriminative video representation can be learned by relatively small scale deep models with fewer computing resources;
- Experiments on two well-known action recognition datasets, the HMDB51 (Kuehne et al. 2011) and UCF101 (Soomro, Zamir, and Shah 2012) benchmarks, show that our method outperforms the state-of-the-art methods based on either supervised or unsupervised learning.

Related work

Iterative clustering. The iterative clustering algorithm (Singh, Gupta, and Efros 2012) is a powerful unsupervised framework for mid-level representation, and its applications have shown its superiority on image-based tasks (Singh, Gupta, and Efros 2012; Huang, Chen, and Tang 2016). Singh, Gupta, and Efros (2012) proposed an iterative clustering algorithm by using SVM classifiers to discover discriminative mid-level concepts for scene classification. Huang *et al.* (Huang, Chen, and Tang 2016) used a two-stage pipeline which trains a deep neural network by using iterative clustering algorithm and fine-tunes the network with the triplet ranking loss for image retrieval and classification. Different from previous methods on image-based tasks, we develop a new iterative clustering algorithm to capture discriminative motion information in videos for action recognition. Compared with the previous methods either using hand-crafted features or learning features from clusters, our method jointly learns optimally compatible features and clusters by using deep networks to obtain precise and robust mid-level representations.

Mid-level video representation. Mid-level representation methods, such as Bag-of-words (Csurka et al. 2004), Fisher vector (Perronnin et al. 2010), and VLAD (Jégou et al. 2010), are commonly used to encode local spatial-temporal features in many works (Dollár et al. 2005; Wang et al. 2011; Wang and Schmid 2013; Jain, Jegou, and Bouthemy 2013) for action recognition. Recently, the hierarchical structured models are proposed to boost action recognition performances by capturing rich semantic information of the mid-level concepts. Liu, Kuipers, and Savarese (2011) used action attributes which can be either manually specified or learnt from the training data to represent human actions. Tang, Fei-Fei, and Koller (2012) proposed a model for automatically discovering action primitives based on clustering and discriminative selecting of primitives. Raptis and Sigal (2013) jointly learned a set of latent keyframes and their local temporal contexts by using a structural SVM. Lan et al. (2015) represented videos by a hierarchy of mid-level action elements with action-level supervision. Different from these

methods, we simultaneously optimize the low-level features and cluster to find more descriptive and discriminative mid-level video representation. The proposed two kinds of deep neural networks guarantee multiple semantic mid-level information are captured.

Our Method

Our goal is to learn visual concepts and local features for generating discriminative and descriptive action representation in an unsupervised way. The overview of the proposed framework is illustrated in Figure 2. In the training phase, to ensure the informativeness of the local features, we learn a fully connected autoencoder using the hand-crafted features of local volumes as input training data and a fully convolutional autoencoder using the local volumes as input. Integrated with the hinge loss and the Fisher loss, the learned encoder layers of the autoencoders are interactively fine-tuned by using the clusters as surrogate classes to maximize the separability of the local features. At the test time, we calculate the center of the top most discriminative volumes of each cluster as the video concept, and then apply the soft assignment Vector of Locally Aggregated Descriptors (VLAD-all) (Peng et al. 2016) to generate the final representation of action. The linear SVM classifiers are then used for action classification.

Pre-training

Given training videos, the local video volumes of interest are extracted as input of the deep neural networks. In this paper, we extract the local volumes along the Improved Trajectories (IT) (Wang and Schmid 2013), which remove volumes of camera motion by homography estimation. In the following, we introduce the two types of autoencoders.

Fully connected autoencoder. The fully connected autoencoder is applied to initialize the network by reconstructing the input itself to retain sufficient information from the hand-crafted local features. We extract the HOG (Dalal and Triggs 2005), HOF (Dalal, Triggs, and Schmid 2006) and MBH (Dalal, Triggs, and Schmid 2006) features of local spatial-temporal video volumes as the input of our fully connected autoencoder. Figure 3 illustrates the architecture of our fully connected autoencoder. The fully connected autoencoder takes n input features $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{d \times n}$, where $\boldsymbol{x}_i \in \mathbb{R}^{d \times 1}$ is the ith d-dimensional IT feature of the local volume extracted from an action video in the training set. The transformation of the encoder, denoted as ϕ_e, maps each input feature into a hidden representation, and the transformation of the decoder, ϕ_d, attempts to map the hidden representation back to the input feature. We define the cost function of the autoencoder by using the Mean Squared Error (MSE)

$$\min_{\phi_e, \phi_d} \frac{1}{N} \sum_{i=1}^{N} \|\boldsymbol{x}_i - \phi_d(\phi_e(\boldsymbol{x}_i))\|_2^2, \tag{1}$$

where N is the size of minibatch.

Fully convolutional autoencoder. Even though the hand-crafted features are used, they might not be optimal for describing the local spatial-temporal volumes. Besides, the input volumes are small that some higher level structural information may be lost. Therefore, we attempt to learn the optimum and complementary local features directly from local volumes by using a fully convolutional autoencoder. Figure 4 shows the architecture of the fully convolutional autoencoder. The input of the fully convolutional autoencoder is a local volume set $\boldsymbol{V} = \{\boldsymbol{v_i} : i = 1, \ldots, n\}$, where $\boldsymbol{v_i} \in \mathbb{R}^{w \times h \times t}$ is the ith local volume with the width of w, the height of h and the length of t. The size of $\boldsymbol{v_i}$ is set to be the half of the original volume. Different from existing fully convolutional networks (Long, Shelhamer, and Darrell 2015; Dosovitskiy et al. 2015b), we use convolutional layers instead of the "deconvolutional" or "upconvolutional" layers as decoder, since the proposed decoder aims to restore information contained in the input features as much as possible, rather than generating new information for specific visual tasks such as image segmentation. As shown in Figure 4, the "Conv1+ReLU $5 \times 5 \times 3$ 16" denotes the convolutional layer with 16 filters where the kernel size is $5 \times 5 \times 3$ pixels and the ReLU is applied as the activation. The "Max Pooling $2 \times 2 \times 2$" denotes that the kernel size and stride of the max pooling layer are both $2 \times 2 \times 2$ pixels. Similar to the fully connected autoencoder, the MSE objective function is used as the reconstruction loss of the fully convolutional autoencoder. For iterative clustering, we concatenate a max pooling layer with the kernel size and stride of $2 \times 2 \times 1$ pixels to the "Conv3" layer.

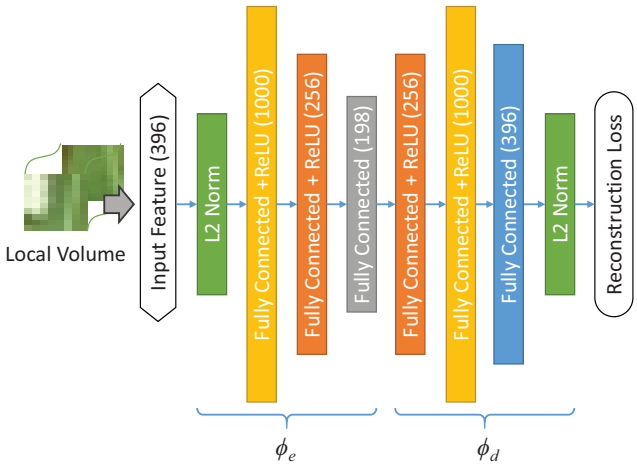

Figure 3: The scheme of the fully connected autoencoder. The input 396-dimensional IT feature extracted from the local volume is $l2$ normalized as in (Wang and Schmid 2013) and passed to the encoder layers, ϕ_e, and the decoder layers, ϕ_d, sequentially. Inside of each bracket is the number of neurons of the corresponding layer. The Rectified Linear Unit (ReLU) activations are applied to the hidden layers with 1,000 and 256 nodes.

Loss functions

Our goal is to learn representative, discriminative, compact video concepts and local features for mid-level representation, in order to reduce inter-category confusions during en-

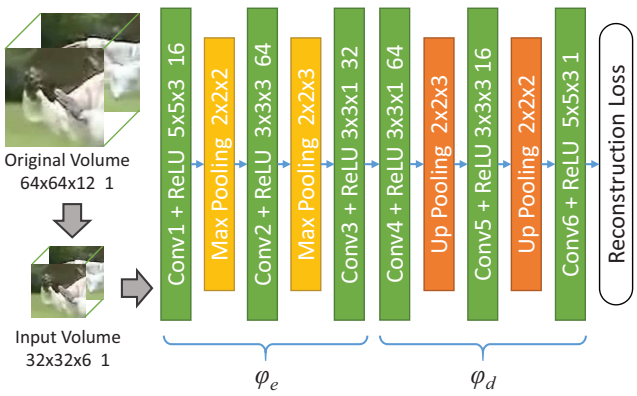

Figure 4: The scheme of the fully convolutional autoencoder. The transformation of the encoder of the fully convolutional autoencoder is denoted as φ_e, and φ_d refers to the decoder.

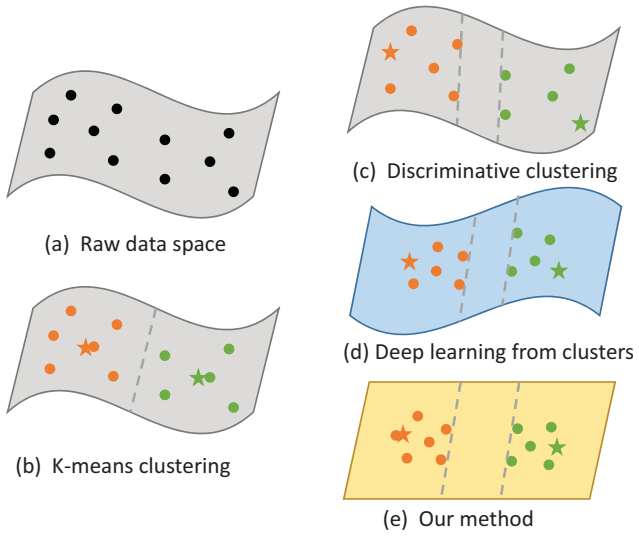

Figure 5: A toy example of different clustering methods. The star characteristic denotes the concept of each cluster. (a) The input data for clustering. (b) K-means minimize the within-cluster distance in an iterative fashion for clustering. (c) Using linear SVM classifiers to compute more discriminative and robust clusters (Singh, Gupta, and Efros 2012). (d) Based on (c), embedding a CNN with softmax loss to learn features from clusters (Huang, Chen, and Tang 2016). The learned features become closer in the same cluster, but the data space is unknown. (e) Our method learn discriminative clusters, and draw features into Euclidean space.

coding as well as to efficiently capture more information from complex action videos. Moreover, the local features are described and measured in Euclidean space, and the relationship between the learned video concepts can thus be easily measured by Euclidean distance. The Euclidean distance rather than latent measurement in learned space ensures that the two loss functions of the neural networks are compatible, and the video concepts are accurately encoded to represent action videos.

To this end, we jointly use the hinge loss and the Fisher loss as the guidance of the neural networks. The Fisher loss, which is known as the Rayleigh quotient objective and designed for cross-views face representation, is applied to learn discriminative and compact local descriptors for encoding of video representation by minimizing the within-class scatter and maximizing the between-class scatter of input samples, formulated by

$$L_{Fisher} = \frac{tr(\boldsymbol{S}_W)}{tr(\boldsymbol{S}_B)}, \tag{2}$$

where $\boldsymbol{S}_W$ and $\boldsymbol{S}_B$ denote the within-class and between-class scatters of the input data, respectively. $tr(\cdot)$ computes the trace of a matrix.

In this work, the Fisher loss is attached to the topmost encoder layers of the pre-trained fully connected and the fully convolutional networks, respectively. For the simplicity of exposition, we consistently denote $\boldsymbol{y}_j^k$ as the output descriptor of the topmost encoder layer of the two autoencoders $\phi_e(\boldsymbol{x}_i)$ and $\varphi_e(\boldsymbol{v}_i)$, where k is the surrogate label generated by clustering, and j denotes the jth descriptor in class k. Therefore, the within-class scatter of the output features of our networks is given by

$$\boldsymbol{S}_W = \sum_k \sum_j (\boldsymbol{y}_j^k - \boldsymbol{\mu}^k)(\boldsymbol{y}_j^k - \boldsymbol{\mu}^k)^{\mathrm{T}}, \tag{3}$$

where $\boldsymbol{\mu}^k = \frac{1}{n_k}\sum_{j=1}^{n_k} \boldsymbol{y}_j^k$ is the mean of class k with n_k samples. And the between-class scatter is given by

$$\boldsymbol{S}_B = \sum_k n_k(\boldsymbol{\mu} - \boldsymbol{\mu}^k)(\boldsymbol{\mu} - \boldsymbol{\mu}^k)^{\mathrm{T}}, \tag{4}$$

where $\boldsymbol{\mu} = \frac{\sum_k \sum_j \boldsymbol{y}_j^k}{\sum_k n_k}$ is the mean of all features. The Fisher loss has been proven to be differentiable with respect to the parameters by (Kan, Shan, and Chen 2016). Figure 5 conceptually illustrates the differences between the proposed clustering using a network with Fisher loss and other related methods.

Although the Fisher loss can learn discriminative features, its constraint on the discrepancy of different categories is so strong that other information of input data such as appearance information would be omitted during the training procedure of the networks. Therefore, the hinge loss with strong categorical supervision is proposed to relax these constraint. Moreover, the hinge loss with L2 regularization on the parameters can be applied as a discriminative classifier to generate the clusters. The hinge loss is formulated as

$$\begin{aligned} L_{Hinge} &= \sum_k \sum_i \max(0, 1 - d_i^c t_i^c) + \lambda \|\boldsymbol{W}\|_{\mathrm{F}}^2, \\ \text{s.t. } & \mathbf{1}^{\mathrm{T}} \boldsymbol{t}_i = 2 - K, \\ & t_i^k \in \{+1, -1\}, \end{aligned} \tag{5}$$

where $\boldsymbol{t}_i = [t_i^1, \ldots, t_i^K]^{\mathrm{T}}$ denotes the labels of K classes and $\|\boldsymbol{W}\|_{\mathrm{F}}^2$ returns the Frobenius norm of the matrix $\boldsymbol{W}$. $\boldsymbol{d}_i = [d_i^1, \ldots, d_i^K]^{\mathrm{T}}$ is the predict value, calculated by $\boldsymbol{d}_i = \boldsymbol{W}\boldsymbol{y}_i + \boldsymbol{b}$. $\boldsymbol{W}$ and $\boldsymbol{b}$ are the weight matrix and bias of a fully connected layer concatenating to the top-most layers of ϕ_e

and φ_e.

Finally, the joint loss function is given by

$$L = \alpha L_{Hinge} + (1-\alpha) L_{Fisher}, \tag{6}$$

where α is a scalar to balance the two loss function.

Iterative algorithm

In this part, we will provide details about how we interactively cluster and fine-tune the networks by using surrogate labels to maximize the separability of the video concepts and local descriptors. To prevent over-fitting, we split the input video volume set $\boldsymbol{V}$ into M non-overlapping subsets $\{\boldsymbol{V}_1, ..., \boldsymbol{V}_M\}$ for cross-validation. For initializing the clusters, we employ k-means clustering of the features learned by the two pre-trained autoencoders (*i.e.*, $\phi_e(\boldsymbol{x}_i)$ and $\varphi_e(\boldsymbol{v}_i)$). Since we can not guarantee whether the initialization result of k-means is optimum or not, we generate as many clusters as we can, and cancel out clusters with very few members during the iterative procedure. The top p nearest to the clustering centers are selected as the initial set of clusters $\boldsymbol{K}$. In practice, removing useless clusters later during the iterative procedure can achieve satisfactory cluster result as well.

Given all detected video volumes in $\boldsymbol{V}_1$, we train the fully connected network ϕ_e and the 3D fully convolutional network φ_e with the hinge loss using the surrogate labels generated by the initial clustering. Then we fine-tune the networks with the joint loss using the initial set $\boldsymbol{K}$ as input. The predict value of the next volume set $\boldsymbol{V}_2$ are extracted from the output of top-most fully connected layer. Given the ith video volume whose predict value is $\boldsymbol{d}_i$, the class it belongs to is calculated by $\hat{k} = \arg\max_k d_i^k$. The score of the ith video volume is formulated as follows:

$$S_i = \begin{cases} d_i^{\hat{k}}, & if \ \forall k \neq \hat{k},\ d_i^k < 0 \ and \ d_i^{\hat{k}} > 0, \\ 0, & otherwise. \end{cases} \tag{7}$$

Through Eq. 7, each local feature is imposed on only one cluster to ensure the mid-level concepts are discriminative. Afterwards, we remove the samples with the score of 0 from $\boldsymbol{V}_2$. Meanwhile, we count the number of samples in each clusters and exclude the clusters with few members to eliminate "useless" and "impure" clusters, where "useless" means that the cluster fires rarely among the local volumes and "impure" means that a cluster converges to two or more concepts. The new clusters are then formed by choosing the samples with the top p highest scores from each cluster. The next set now becomes the training set, and the procedure is repeated until that the top p samples in each cluster do not change, *i.e.*, convergence.

The overall procedure of our method is shown in Algorithm 1.

Algorithm 1: Iterative clustering algorithm for deep video representation.

Input: IT feature sets $\{\boldsymbol{X}_m|_{m=1}^M\}$ and local spatial-temporal volume sets $\{\boldsymbol{V}_m|_{m=1}^M\}$.
Output: Video concepts $\boldsymbol{C}$ and deep local spatial-temporal features $\boldsymbol{Y}$.

```
Pre-train φ_e and ϕ_e;
Extract the initial features ϕ_e(X_1) and φ_e(V_1);
Calculate clusters K via k-means on ϕ_e(X_1) and φ_e(V_1);
repeat
    for m = 1 → M do
        Fine-tune ϕ_e and φ_e via K, X_m and V_m;
        Extract the predict value of X_{m+1} and V_{m+1} from
          the top-most fully connected layer of ϕ_e and φ_e;
        Calculate the new cluster K̃;
        K ⇐ K̃;
    end
until Convergence;
Extract local volume features Y from ϕ_e and φ_e;
Calculate C via the predict value of {X_m|_{m=1}^M} and
  {V_m|_{m=1}^M}.
```

Action recognition

Given the learned discriminative local features and video concepts, the soft assignment Vector of Locally Aggregated Descriptors (VLAD-all) (Peng et al. 2016) is employed to encode the local features and generate the mid-level representation of videos. We use a soft assignment encoding method instead of the local encoding methods because the learned features can be precisely measured in Euclidian distance. The VLAD-all is a kind of super vector based encoding method which encodes features to high dimensional descriptors with substantial information. The power, l_2 and intra normalization methods are applied to the VLAD-all. Finally, after obtaining the global representation of videos, we use the linear SVM classifiers for action recognition.

Experiments

Datasets

Extensive experiments are conducted on the HMDB51 (Kuehne et al. 2011) and the UCF101 (Soomro, Zamir, and Shah 2012) datasets to evaluate the performance of our method. The HMDB51 dataset comprises of 51 action categories with 6,766 realistic video clips collected from movies and the Internet. The UCF101 dataset consists of 101 action categories with 13,320 realistic video clips which are collected from YouTube.

We follow the standard evaluation protocols of the two datasets provided on (Kuehne et al. 2011) and (Soomro, Zamir, and Shah 2012) to calculate the average accuracy over the three splits into training and test data.

Implementation details

The input IT features of the fully connected network are extracted following the setup of (Wang and Schmid 2013), so that the dimension of each IT feature is 396. The centers of the input volumes are sampled at the middle of the trajectories, and the size of each volume is $16 \times 16 \times 12$. The number of cross-validation sets is $m = 10$. The number of the topmost samples selected from each cluster during iteration is $p = 100$. The initial number of clusters is set to 600 for the fully connected autoencoder and 1,000 for the fully convolutional autoencoder, and after optimizing, it

Methods	HMDB51	UCF101
K-means clustering	54.7	85.5
Discriminatively clustering	56.5	86.4
Deep learning from clusters	62.6	88.1
Our method	66.8	90.3

Table 1: Comparison with different clustering methods for action recognition (accuracy, %) on the HMDB51 dataset and UCF101 dataset.

comes to $q = 499$ and 762, respectively. Practically, the algorithm converges in 4 iterations. The dimension of the output of the proposed networks is set to 198 and 512. As for the linear SVMs, the value of the penalty parameter is chosen among $10^{-3}, 10^{-2}, 10^{-1}, 1, 10^{1}, 10^{2}, 10^{3}$. The training of the deep networks is implemented on a single NVIDIA TITAN X GPU with the memory of 12G.

Evaluation on unsupervised clustering methods

Several unsupervised clustering methods (Singh, Gupta, and Efros 2012; Dosovitskiy et al. 2015a) are adopted to compare with our method for action recognition, including k-means clustering, discriminative clustering, deep learning from clusters. The VLAD-all encoding is applied after clustering to obtain global representation of action videos, and the linear SVM is used for action classification. Some of the details and parameter settings used in those proposed methods are briefly described as follows.

- **K-means clustering:** The Principal Component Analysis (PCA) is used to reduce the dimension of l2 normalized IT features from 396 to 198. The number of clusters is set to 512, which is recommended as the most efficient choice for VLAD-all encoding in (Peng et al. 2016).
- **Discriminative clustering:** The input features are the same with ours, and the algorithm in (Singh, Gupta, and Efros 2012) is applied to find the mid-level concepts.
- **Deep learning from clusters:** The input features and the networks ϕ_e are the same with ours, and the softmax loss is used. The clustering algorithm proposed by (Singh, Gupta, and Efros 2012) is applied to learn and select the mid-level concepts.

Table 1 shows the performances of the aforementioned clustering methods for action recognition. As can be seen, our method outperforms other methods. Comparing rows 1 and 2 of Table 1, the results of discriminative clustering are better than k-means clustering, which demonstrates the importance of discriminative clusters. But the performances of these two methods are both hand-crafted feature based lag far behind our method (row 4) and the method of deep learning from clusters (row3) which are deep learning based. It verifies that learning representation with semantic guidance leads to better action recognition performance. The improvement of our method compared with the method of deep learning from clusters indicates that using unified metric can remedy the inaccurate measurement.

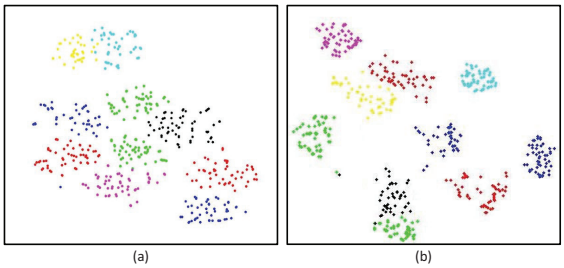

Figure 6: Visualization of local video representations clustered by different methods. (a) The IT features clustered by k-means. (b) Deep local representation clustered by our method. Best viewed in color.

Adaptivity Evaluation

We aim to learn generic and practical deep networks by training only once. To this end, the proposed model should be adaptive. Since the background clutter of videos in the HMDB51 dataset is more than that in the UCF101 dataset, the HMDB51 dataset contains richer motion information than the UCF101 dataset. Thus, we extract the local spatial-temporal volumes from video clips in the HMDB51 dataset to train the proposed networks, and use the trained networks to extract features of the UCF101 dataset to evaluate the adaptivity of our networks.

Firstly, we qualitatively evaluate the adaptiveness of the deep networks by visualizing the extracted features using method of tSNE (Maaten and Hinton 2008). As shown in Figure 6, we randomly selected 500 features in 10 clusters for each visualization. Figure 6 (a) demonstrates the IT features extracted from videos in the UCF101 dataset clustered by k-means and Figure 6 (b) shows the features extracted from videos in the UCF101 dataset clustered by ϕ_e with the output of hinge loss trained on the HMDB51 dataset. It is obvious that our clusters are more discriminative, which clearly validates the effectiveness and adaptivity of the proposed network.

Secondly, for quantitative analysis, we compare the action recognition performance on the UCF101 dataset of video representation encoded by local features extracted from the two ϕ_es respectively trained on the UCF101 dataset and the HMDB51 dataset. Using videos in the UCF101 dataset as training samples achieves an average accuracy of 90.3%, and using the HMDB51 dataset achieves 88.3%. The latter performance is almost on par with the former, which demonstrates the adaptivity of the the proposed model.

Comparison with the state-of-the-art

Finally, we compare our method with the state-of-the-art methods on the UCF101 and HMDB51 datasets. Here, besides applying the features of ϕ_e and φ_e for action recognition respectively, we combine these two kinds of features to further improve the action recognition performance. The combination is implemented by the Generalized Multiple Kernel Learning (GMKL) (Varma and Babu 2009) with linear kernels. Table 2 shows that the combination of the fea-

Methods	HMDB51	UCF101
Wang and Schmid 2013	57.2	-
Lan et al. 2015b	65.4	89.1
Srivastava et al. 2015	-	84.3
Misra et al. 2016	29.9	-
Tran et al. 2015	-	90.4*
Bilen et al. 2016	65.2*	89.1*
Feichtenhofer et al. 2016	69.2*	93.5*
Wang et al. 2016	69.4*	94.2*
Kar et al. 2017	66.9*	93.2*
Feichtenhofer et al. 2017	72.2*	94.9*
Ours (ϕ_e)	66.8	90.3
Ours (φ_e)	69.0	91.2
Ours ($\phi_e + \varphi_e$)	73.8	93.6
Ours ($\phi_e + \varphi_e$) + RGB	74.1	95.3

Table 2: Comparison with the state-of-the-art methods (accuracy, %), both unsupervised and supervised (indicated by *), on the HMDB51 dataset and UCF101 dataset.

tures taken from ϕ_e and φ_e improve the action recognition accuracy indeed. The probable reason is that the fully connected auto-encoder captures lower-order motion information of hand-crafted features, while the fully convolutional auto-encoder offers higher-order information by multiple convolution operations. Accordingly, the two kinds of features are complementary for better representing action videos.

The state-of-the-art methods of action recognition methods in Table 2 for comparison are divided into three groups: (1) Shallow methods using hand-crafted features (Wang and Schmid 2013 and Lan et al. 2015b.); (2) Unsupervised learning method (Srivastava et al. 2015 and Misra et al. 2016);(3) Supervised deep learning methods (Tran et al. 2015, Bilen et al. 2016, Feichtenhofer et al. 2016, Wang et al. 2016, Kar et al. 2017, and Feichtenhofer et al. 2017).

It is apparent that our method outperforms all the shallow methods which demonstrates that our mid-level video representation learned by deep neural networks is more discriminative than the representations which only encode IT features without deep learning. Compared with the unsupervised methods (Srivastava et al. 2015 and Misra et al. 2016) using sequential information of videos for global action representation, our method achieves a significant improvement on action recognition performances. It attributes to the proposed simple and small deep learning models which can precisely discover more discriminative information from videos and are less likely to overfit.

Also as shown in Table 2, it is interesting and encouraging to observe that our unsupervised method performs better than the supervised methods of (Tran et al. 2015, Bilen et al. 2016 and Kar et al. 2017). Tran et al. 2015 combined features taken from 3D CNNs trained on a large scale supervised video dataset with the IT features, while our method does not need such a large scale labeled video set to train 3D network but can learn models of good adaptiveness as well. Bilen et al. 2016 developed dynamic images to summarize motion information and used 2D CNNs to learn video representation. Kar et al. 2017 proposed an adaptive temporal pooling method for action recognition. They fused two kinds of motion information, namely the IT features and their deeply learned features, to achieve good action recognition performances. The better action recognition performances achieved by fusing the features of ϕ_e and φ_e indicates the efficiency of our method on representing action videos.

Among the supervised methods in Table 2, the multi-stream networks based methods (Feichtenhofer et al. 2016, Wang et al. 2016, and Feichtenhofer et al. 2017) use not only the motion information but also the frame-level information acquired from the RGB features. It can be inferred that the RGB features helps improve the performance of action recognition. We thus enrich our method by adding the RGB features which are extracted from the top-most layer of the ResNet-152 (He et al. 2016) followed by the average pooling operation. The results show that our method with the RGB features improves the recognition accuracy, and outperforms all the state-of-the-art methods.

Conclusions

In this paper, we have proposed a novel unsupervised deep learning method for action recognition by jointly clustering and learning features via deep neural networks. The deep networks learn robust and discriminative local features with the guidance of higher semantic information, and the higher semantic information is generated by the clustering. By iteratively performing the clustering and the training of the networks, the discriminative power of the learned deep features can be highly enhanced for globally representing action videos. Extensive experiments on action recognition benchmarks have demonstrated the superiority of the proposed approach.

Acknowledgments

This work was supported in part by the Natural Science Foundation of China (NSFC) under Grants No.61673062 and No.61472038.

References

Bilen, H.; Fernando, B.; Gavves, E.; Vedaldi, A.; and Gould, S. 2016. Dynamic image networks for action recognition. In *CVPR*, 3034–3042.

Csurka, G.; Dance, C.; Fan, L.; Willamowski, J.; and Bray, C. 2004. Visual categorization with bags of keypoints. In *Workshop on ECCV*, volume 1, 1–2. Prague.

Dalal, N., and Triggs, B. 2005. Histograms of oriented gradients for human detection. In *CVPR*, volume 1, 886–893. IEEE.

Dalal, N.; Triggs, B.; and Schmid, C. 2006. Human detection using oriented histograms of flow and appearance. In *ECCV*, 428–441. Springer.

Dollár, P.; Rabaud, V.; Cottrell, G.; and Belongie, S. 2005. Behavior recognition via sparse spatio-temporal features. In

Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on, 65–72. IEEE.

Dosovitskiy, A.; Fischer, P.; Springenberg, J.; Riedmiller, M.; and Brox, T. 2015a. Discriminative unsupervised feature learning with exemplar convolutional neural networks. *IEEE TPAMI* 1(10):1–1.

Dosovitskiy, A.; Fischer, P.; Ilg, E.; Hausser, P.; Hazirbas, C.; Golkov, V.; van der Smagt, P.; Cremers, D.; and Brox, T. 2015b. Flownet: Learning optical flow with convolutional networks. In *ICCV*, 2758–2766.

Du, Y.; Wang, W.; and Wang, L. 2015. Hierarchical recurrent neural network for skeleton based action recognition. In *CVPR*, 1110–1118.

Feichtenhofer, C.; Pinz, A.; and Wildes, R. P. 2017. Spatiotemporal residual networks for video action recognition. In *CVPR*. IEEE.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*, 770–778.

Huang, C.; Chen, C. L.; and Tang, X. 2016. Unsupervised learning of discriminative attributes and visual representations. In *CVPR*, 5175–5184. IEEE.

Jain, M.; Jegou, H.; and Bouthemy, P. 2013. Better exploiting motion for better action recognition. In *CVPR*, 2555–2562.

Jégou, H.; Douze, M.; Schmid, C.; and Pérez, P. 2010. Aggregating local descriptors into a compact image representation. In *CVPR*, 3304–3311. IEEE.

Kan, M.; Shan, S.; and Chen, X. 2016. Multi-view deep network for cross-view classification. In *CVPR*, 4847–4855.

Kar, A.; Rai, N.; Sikka, K.; and Sharma, G. 2017. Adascan: Adaptive scan pooling in deep convolutional neural networks for human action recognition in videos. In *CVPR*.

Kuehne, H.; Jhuang, H.; Garrote, E.; Poggio, T.; and Serre, T. 2011. Hmdb: a large video database for human motion recognition. In *ICCV*, 2556–2563. IEEE.

Lan, T.; Zhu, Y.; Roshan Zamir, A.; and Savarese, S. 2015. Action recognition by hierarchical mid-level action elements. In *ECCV*, 4552–4560.

Le, Q. V.; Zou, W. Y.; Yeung, S. Y.; and Ng, A. Y. 2011. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *CVPR*, 3361–3368. IEEE.

Lin, T.-Y.; Doállr, P.; Girshick, R.; He, K.; Hariharan, B.; and Belongie, S. 2016. Feature pyramid networks for object detection. *arXiv preprint arXiv:1612.03144*.

Liu, J.; Kuipers, B.; and Savarese, S. 2011. Recognizing human actions by attributes. In *CVPR*, 3337–3344. IEEE.

Long, J.; Shelhamer, E.; and Darrell, T. 2015. Fully convolutional networks for semantic segmentation. In *CVPR*, 3431–3440.

Maaten, L. v. d., and Hinton, G. 2008. Visualizing data using t-sne. *JMLR* 9(Nov):2579–2605.

Misra, I.; Zitnick, C. L.; and Hebert, M. 2016. Unsupervised learning using sequential verification for action recognition.

Peng, X.; Wang, L.; Wang, X.; and Qiao, Y. 2016. Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice. *CVIU* 150:109–125.

Perronnin, F.; Liu, Y.; Sánchez, J.; and Poirier, H. 2010. Large-scale image retrieval with compressed fisher vectors. In *CVPR*, 3384–3391. IEEE.

Raptis, M., and Sigal, L. 2013. Poselet key-framing: A model for human activity recognition. In *CVPR*, 2650–2657.

Simonyan, K., and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Singh, S.; Gupta, A.; and Efros, A. A. 2012. Unsupervised discovery of mid-level discriminative patches. In *ECCV*, 73–86. Springer.

Soomro, K.; Zamir, A. R.; and Shah, M. 2012. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*.

Srivastava, N.; Mansimov, E.; and Salakhutdinov, R. 2015. Unsupervised learning of video representations using lstms. In *ICML*, 843–852.

Su, B.; Zhou, J.; Ding, X.; Wang, H.; and Wu, Y. 2016. Hierarchical dynamic parsing and encoding for action recognition. In *ECCV*, 202–217. Springer.

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. In *CVPR*, 1–9. IEEE.

Tang, K.; Fei-Fei, L.; and Koller, D. 2012. Learning latent temporal structure for complex event detection. In *CVPR*, 1250–1257. IEEE.

Taylor, G. W.; Fergus, R.; LeCun, Y.; and Bregler, C. 2010. Convolutional learning of spatio-temporal features. In *ECCV*, 140–153. Springer.

Tran, D.; Bourdev, L.; Fergus, R.; Torresani, L.; and Paluri, M. 2015. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 4489–4497. IEEE.

Varma, M., and Babu, B. R. 2009. More generality in efficient multiple kernel learning. In *ICML*, 1065–1072. ACM.

Wang, H., and Schmid, C. 2013. Action recognition with improved trajectories. In *ICCV*, 3551–3558.

Wang, H.; Kläser, A.; Schmid, C.; and Liu, C.-L. 2011. Action recognition by dense trajectories. In *CVPR*, 3169–3176. IEEE.

Wang, L.; Xiong, Y.; Wang, Z.; Qiao, Y.; Lin, D.; Tang, X.; and Van Gool, L. 2016. Temporal segment networks: towards good practices for deep action recognition. In *ECCV*, 20–36. Springer.

Wiskott, L., and Sejnowski, T. J. 2002. Slow feature analysis: Unsupervised learning of invariances. *Neural computation* 14(4):715–770.