A Temporal Sequence Learning for Action Recognition and Prediction

Sangwoo Cho, Hassan Foroosh
University of Central Florida
swcho@knights.ucf.edu, foroosh@cs.ucf.edu

Abstract

In this work[1], we present a method to represent a video with a sequence of words, and learn the temporal sequencing of such words as the key information for predicting and recognizing human actions. We leverage core concepts from the Natural Language Processing (NLP) literature used in sentence classification to solve the problems of action prediction and action recognition. Each frame is converted into a word that is represented as a vector using the Bag of Visual Words (BoW) encoding method. The words are then combined into a sentence to represent the video, as a sentence. The sequence of words in different actions are learned with a simple but effective Temporal Convolutional Neural Network (T-CNN) that captures the temporal sequencing of information in a video sentence. We demonstrate that a key characteristic of the proposed method is its low-latency, i.e. its ability to predict an action accurately with a partial sequence (sentence). Experiments on two datasets, UCF101 and HMDB51 show that the method on average reaches 95% of its accuracy within half the video frames. Results, also demonstrate that our method achieves compatible state-of-the-art performance in action recognition (i.e. at the completion of the sentence) in addition to action prediction.

1. Introduction

Video-based action recognition is an active research area due to its important practical applications in many areas, such as video surveillance, behavior analysis, and human-computer interaction. The action recognition task is accomplished after acquiring the entire video, while action prediction is different in the sense that it aims at classifying the action with shortest possible latency, i.e. classify as early as possible as the frames come in. The capability of predicting an action early is crucial in both surveillance systems and human-computer interaction. The two tasks of action

prediction and recognition have often been researched separately under different settings and constraints.
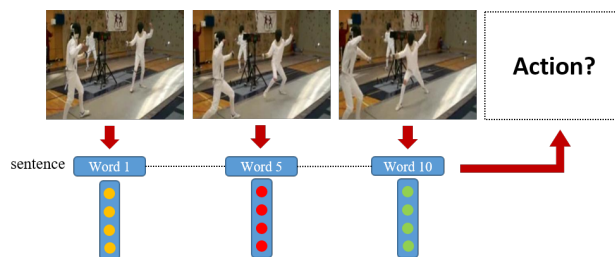

Figure 1: Given a partial or a full video frames, our goal is to classify the correct action. Each frame is converted to a corresponding word and the sequence of words is trained to predict an activity.

A video contains two important pieces of information: appearances and motions. These information are complementary, and therefore an accurate prediction relies on the ability to extract the information with low latency, i.e. as early as possible in the temporal sequence. However, extracting effective information (whether for prediction or recognition) is non-trivial due to a number of difficulties such as viewpoint changes, camera motions, and scale variations, to name a few. It is thus crucial to design an effective and generalized representation of a video. Convolutaional Neural Networks (ConvNets) [17] have been playing a key role in solving hard problems in various areas of computer vision, e.g. image classification [17, 9, 44] and human face recognition [29]. ConvNets also have been employed to solve the problem of action recognition [30, 14, 37, 22] in recent literature.

Data-driven supervised learning enables to achieve discriminating power and proper representation of a video from raw data. However, ConvNets for action recognition have not shown a significant performance gain over the methods utilizing hand-crafted features [40, 25]. We speculate that the main reason for the lack of big impact is that ConvNets employed in action recognition do not take full advantage of temporal sequencing or order. Recently some methods [39, 5] attempted to capture long-term temporal information. However, they require excessive computation

for a long video.

In this work, inspired by key ideas from NLP, we represent each frame as a word and a video as a sequence of words. The sequence of words, or a sentence, is a new video representation as shown in Fig. 1. We call the word as *ActionWord*. We use the standard BoW [24] framework to encode each visual feature as an assigned word in a codebook. The sequence of words then is learned with a simple but effective CNN architecture capturing the sequential order of temporal information. This method is flexible to input size, and hence is applicable to any length of videos. The capability to adopt a variable-size input, combined with low latency versus high accuracy makes the method particularly powerful for both action prediction and action recognition.

Our key contributions can thus be summarized as follows: (i) A new representation for video data as a sequence of words that inherently captures temporal order and sequencing of information. (ii) An effective ConvNet that learns such temporal sequencing to predict with low latency an action. (iii) The ability of the method to maintain state-of-the-art accuracy in both predition and recognition with the challenging datasets, such as UCF101 and HMDB51. (iv) The entire system is easy to implement and is trained with a small amount of computational cost compared to other methods employing ConvNets.

2. Related Work

Several works using ConvNets to acquire temporal information for action recognition have been studied. In [42], hand crafted features are used in the pooling layer of ConvNet to take advantage of both merits of hand-designed and deep learned features. Temporal information from optical flow is explicitly learned with ConvNets in [30] and the result is fused with the effect of the trained spatial (appearance) ConvNet. [6] merges the ConvNet architecture of the two streams ConvNets [30] to capture spatio-temporal information. Although the aforementioned approaches capture temporal information in small time windows, they fail to capture long-range temporal sequencing information that contains long-range ordered information.

Several works modeling a video-level representation or modeling long temporal information with ConvNets have also been investigated. [7] proposes a method that employs a ranking function to generate a video-wide representation that captures global temporal information. In [35], a HMM model is used to capture the appearance transitions and a max-margin method is employed for temporal information modeling in a video. [5, 35, 21] utilize LSTM [9] unit in their ConvNets and attempt to capture long-range temporal information. However, the most natural way of representing a video as long-range ordered temporal information is not fully exploited.

Action prediction is to recognize an action with a partial amount of video data. The task may be considered as a subset of the action recognition problem, in a sense that the input data is limited. [28] proposes the integral BoW and dynamic BoW to model an action in a particular stage. Sparse coding is used to compute activity likelihood of video segments [3]. A max-margin learning method for prediction is proposed in [3], where human activity is represented in a hierarchical way. [16, 10] employ structured SVM to detect an event and capture global and local dynamics of motions. However, the performance of the above methods are not comparable to our results and they are not applicable to large-scale datasets, such as UCF101 [33].

Our work is inspired by a key idea of sentence classification [46, 12, 13, 15] in NLP. We convert from the domain of images to a domain of words to represent each frame as a word and hence represent a video as a sequence of words, i.e. a sentence. In NLP, words in a sentence are often represented in the form of vectors, see for instance word2vec [20] and Glove [26]. In order to acquire a similar frame-level representation, we adopted the standard BoW [24] encoding method to handle large variability of motions and appearances in video data. It is worth noting, however, that our method can adopt any type of frame-level features to represent video frames as words.

Various ConvNet arichitectures [46, 12, 13, 15] have been taken into account for sentence classification. [13] utilizes dynamic pooling ConvNets for modeling sentences. In [12, 15], a simple 1D ConvNet is employed to classify sentences, and LSTM unit is additionally inserted in [46]. Similarly, we utilize a simple but effective ConvNet for learning video word sequencing for action prediction and recognition applicable to large-scale datasets.

3. Approach

In this section, we give a detailed description of the proposed word encoding and word sequence learning. The pipeline of our method is illustrated in Fig. 2.

3.1. BoW Framework for Word Representation

Feature Extraction: Since the approaches based on ConvNets [30, 31, 6, 42] recently have achieved competitive results, we utilize *deep-learned* features. In [30], a two-stream ConvNet is trained with stacked optical flows and frames. We follow the two-stream ConvNet method and extract N features $\{x_1, \cdots, x_N\}$, where $x_t \in \mathbb{R}^D$, every T frame from all videos using the two trained networks. The extracted features are the output vectors of fully connected (FC) layers on both ConvNets and the dimension is D. Note that the input frames of consecutive temporal features are overlapped by $(L - T)$ frames, when $L > T$, as we train the temporal network with L stacked frames. The temporal ConvNet is trained with $L = 10$ and T is set to 5 to consider partial overlap between consecutive temporal
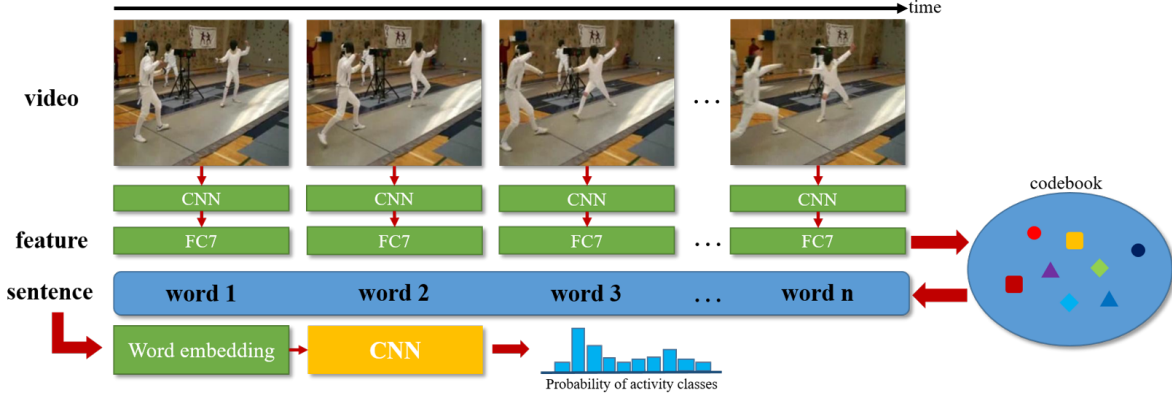
Figure 2: Pipeline of our method for action prediction/recognition. First, we extract features from video frames using a trained CNN. We then generate a codebook to assign each feature as *ActionWord* as explained in section 3.1. Finally, a sequence of *ActionWord*s is learned with a sequence learning CNN to classify actions, as described in section 3.2.
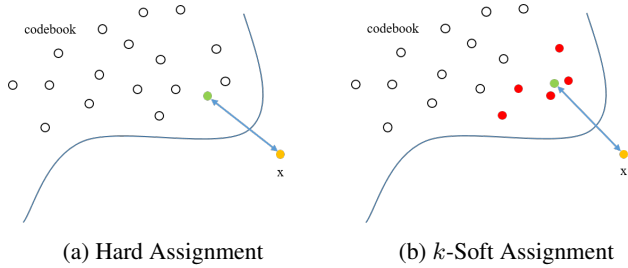

(a) Hard Assignment   (b) k-Soft Assignment

Figure 3: Feature encoding methods

features. Also, it should be noted that any frame-wise feature extraction techniques can be utilized to represent each frame as a vector.

Codebook Generation: A codebook is generated to represent each feature as an *ActionWord*. A typical choice for constructing the codebook is k-means [2] or Gaussian Mixture Model (GMM) [2]. In our method, we used the method of approximate k-means [27] to construct the codebook with all extracted features from training videos. The generated K clusters $\{c_1, \cdots, c_K\}$, where $c_k \in \mathbb{R}^D$, are employed to both training and testing videos.

Codeword Assignment: For coding a video, every extracted video frame feature vector x needs to be mapped to one of the vectors in the codebook, i.e. to one *ActionWord* that best represents the frame-level visual information at time T. We consider two voting based assignment methods: Hard assignment (HA) [32] (or Vector Quantization) and soft assignment (SA) [38], and a direct assignment as described below.

- Hard Assignment: With HA, *ActionWord A*, is simply associated with its nearest codeword to the feature as shown in Fig. 3a. The nearest codeword is determined as the one best correlated with the feature vector x_n. The assigned word number (label) for each feature is a sequential

number from 1 to K.

$$A_{HA_i} = \underset{i}{\arg\min} \|x - c_i\|_2 \qquad (1)$$

where $i \in \{1, \cdots, K\}$ and a corresponding weight vector ω for each feature is associated with one of codewords based on the assigned number.

$$\omega_{x_{HA}} = c_i, \quad \text{where } i = A_{HA_i}. \qquad (2)$$

HA encoding enables reducing memory requirements by maintaining only codewords and the assigned codeword numbers instead of keeping all features. Moreover, the codeword can be ignored and initialized with random values when learning a sequence of assigned numbers. Thus, a video can be represented by a sequence of assigned numbers, leading to memory saving.

- Soft Assignment: The SA method considers k-nearest codewords to the feature. Fig. 3b illustrates an example of 5 nearest neighbor (NN) codewords (5-SA). Five red nearest codewords are correlated with the feature vector x and a weighted centroid vector colored in green is then computed for assignment. The weight vector ω is computed as follows.

$$\omega_{x_{SA}} = \sum_{j=1}^{K} \delta(x, c_j) \cdot c_j \cdot d_{\omega_j} \qquad (3)$$

where d_{ω_j} is the normalized inverse distance weight:

$$d_{\omega_j} = \frac{\delta(x, c_j) \exp(-\beta \|x - c_j\|_2^2)}{\sum_{j=1}^{K} \delta(x, c_j) \exp(-\beta \|x - c_j\|_2^2)} \qquad (4)$$

where $\delta(x, c_j)$ is the indicator function for the k-NN codewords of x:

$$\delta(x, c_j) = \begin{cases} 1, & \text{if } c_i \in k\text{-NN}(x), \\ 0, & \text{otherwise.} \end{cases} \qquad (5)$$

Thus, the computed weight vector ω gives the weighted centroid of k-NN codewords based on inverse distance between the feature and k nearest codewords. Each weight vector ω is unique, and therefore an assigned number for each weight vector ω is also unique. Hence, the total number of assigned numbers is the same as the total number of extracted features in a dataset.

$$A_{SA_i} = i, \quad \text{where } i \in \{1, \cdots, \text{N}\}. \tag{6}$$

When learning an *ActionWord* encoded with SA, random vector initialization of the weight vectors cannot be feasible as the assigned numbers are nothing but sequential numbers for each feature. Note that HA can be regarded as a special case of k-SA, where k is 1.

- **Direct Assignment**: Instead of computing the codebook, Direct Assignment (DA) encoding considers each video-frame feature as a weighted codeword and assign a unique number to it.

$$\omega_{\text{x}_{DA}} = \text{x} \tag{7}$$
$$A_{DA_i} = i, \quad \text{where } i \in \{1, \cdots, \text{N}\}. \tag{8}$$

Each frame feature vector is thus directly considered as an *ActionWord*. This method does not require codebook generation leading to reduced computation time, but the memory requirement increases.

3.2. Sequence Learning with Temporal ConvNet

With the proposed *ActionWord* coding, action prediction and action recognition can be regarded as classification problems for a partial sentence or a sentence. By leveraging the success of sentence classification using ConvNets [46, 12, 13, 15] in NLP, we apply similar ConvNet architectures to train and classify *ActionWord* sequences. We consider two ConvNet models: i) T-CNN, ii) Covolutional LSTM (C-LSTM).

Word Embedding: The sequence of *ActionWord*s is the input to the ConvNets shown in Fig. 4. Since the length of the sequence for each video is different, a word embedding layer is utilized to make the sequences of the same length. The length of each sequence l_i is truncated if $l_i > l_{max}$ whereas l_i is padded with a special codeword that corresponds to $v = [0, \cdots, 0]$ if $l_i < l_{max}$, where $v \in \mathbb{R}^D$ and l_{max} is a user-determined sequence length. The word embedding layer combines the corresponding weight vector ω based on the assigned word number, and generates an $D \times l_{max}$ matrix for each sequence. The weight vector can be initialized with a random number between -0.05 and 0.05 for the HA random initialization encoding method.

T-CNN Model: Fig. 4a shows the overall structure of the T-CNN Model. T-CNN consists of L one-dimensional convolution layers denoted by $C^l \in \mathbb{R}^{F_l \times T}$ in parallel where
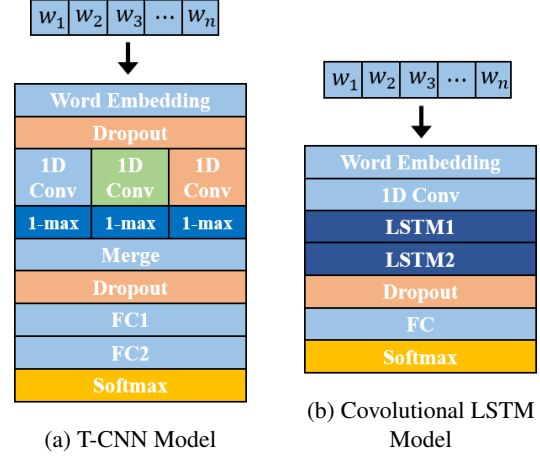

(a) T-CNN Model (b) Covolutional LSTM Model

Figure 4: ConvNet Architectures

F_l is the number of convolution filters in the l-th layer and T is same as l_{max}. Each layer consists of temporal convolution, a non-linear activation, and global max (1-max) pooling across time. The collection of filters in each layer is defined as $W = \{W^{(i)}\}_{i=1}^{F_l}$ where $W^{(i)} \in \mathbb{R}^{d \times F_l}$ and a window of d duration. The corresponding bias vector is $b \in \mathbb{R}^{F_l}$. Given the input sequence of weight vectors, $\Omega \in \mathbb{R}^{D \times T}$, the activation C^l is computed such that

$$C^l = ReLU(\mathbb{W} * \Omega + b) \tag{9}$$

where $*$ is the convolution operator. The convoluted signals can be viewed as N-gram in a sentence, where N can be determined by the size of filters in the convolution layer. After the ReLU activation, the global max pooling is applied to get the largest signal from the activation. Each layer produces a ν vector where $\nu \in \mathbb{R}^{F_l}$ by concatenating the global max signals. All vectors from L layers are then concatenated generating a v vector where $v = \sum_{i=1}^{L} \nu_i$. The output size of the second FC layer is the number of class in a dataset and Softmax activation is applied in the end.

Covolutional LSTM Model: C-LSTM Model consists of a convolution layer and a long short-term memory recurrent neural network (LSTM) [9] designed for time-series data to learn long-term information. Fig. 4b shows the overall architecture of the C-LSTM. The multiple parallel convolution layer is not applied because the concatenation of the resulting vectors can break the original sequence for the input of the LSTM layer. The global max pooling layer is also omitted for the same reason. We retain the original order of the sequence and extract more descriptive representations by convolution computation for the sequence. The extracted local temporal information is fed into the LSTM layer and the LSTM layer outputs a video level representation that captures high level temporal information.

	UCF101	HMDB51
C	101	51
l_{train}	35.8 (4 / 354)	17.7 (2 / 211)
l_{test}	35.3 (4 / 177)	17.1 (3 / 128)
N	9537 (3783)	3570 (1530)

Table 1: Summary statistics of extracted features for each dataset. C: number of classes, l_{train}: average sequence length of training data (min / max), l_{test}: average sequence length of testing data (min / max), N: number of training(testing) sequences(or videos) for each dataset

4. Experiments

4.1. Dataset and Statistics

We test our method on two action video datasets, HMDB51 [18] and UCF101 [33]. The HMDB51 dataset consists of 51 action classes with 6,766 videos and more than 100 videos in each class. All videos are acquired from movies or Youtube, and contain various human activities, including interactions with other humans or objects. Each action class has 70 videos for training and 30 videos for testing. The UCF101 dataset consists of 101 action categories with 13,320 videos and at least 100 videos are involved in each class. All videos are gathered from Youtube.

Both datasets provide three training and testing splits. We used the first split of each dataset for validating our proposed models. The same parameters and models from split 1 are utilized for other two splits. Table 1 shows the statistics of sequence lengths on each dataset for our experiments. We extracted temporal features every 5 frames (T = 5) with 10 stacked input frames (L = 10) and spatial features every 5 frames.

4.2. Implementation Details

Training Two-ConvNets: We use the VGG-16 model [31] for two-stream ConvNets training. Both the temporal and the spatial network are initialized with the pre-trained weights trained with ImageNet [4]. The networks are then fine-tuned with each dataset.

For the training of the spatial network, we use dropout ratios of 0.8 for two FC layers. The input images are resized to make the smaller side as 256. We augment the input images by randomly cropping 224×224 sub-images from the four corners and the center of the original images and randomly flipping in horizontal direction. The learning rate is set to 10^{-3} initially and decreased by a factor of 10 when the validation error saturates.

For the training of the temporal network, we use dropout ratios of 0.9 for UCF101 and 0.9 and 0.8 for HMDB51. We pre-compute the optical flows using the TVL1 method [45] before training to improve the training speed. The optical flow input is stacked with L = 10 frames making a 224×224×20 sub-volume. Same data augmentation techniques are employed for the sub-volume and the learning rate is initialized with 5×10^{-4} and decreased in the same manner of the spatial network training. A mini-batch of 128 samples are employed at each iteration, but batch normalization method [11] is not used for all trainings.

Word Vector Representation: The dimension of temporal x_t and spatial x_s feature vectors is 4096. Since the two extracted feature vectors are complementary, we concatenate them with a data ratio r, resulting in a combined feature vector x.

$$x = PCA(x_{t(1:rD)}) \oplus PCA(x_{s(1:(1-r)D)}) \qquad (10)$$

where D is the dimension of x, $0 \leq r \leq 1$, $\oplus$ is a concatenation operation, and $PCA(x_{1:n})$ is to apply PCA to x and take the first n elements of the projected vector. The reduced dimension of x is $D' \in \{32, 64, 128, 256, 512, 1024\}$. We use the output vector of the penultimate FC (FC7) layer, since the performance with the FC7 vectors is consistently 2~3% better than the one with the first FC (FC6) layer. In addition, we take the output vector of FC7 with input images or optical flow images that are cropped in the center area making size of 224×224. For the SA and HA feature encoding method, we consider $K = \{5000, 10000, 20000\}$ as the size of codebook.

Training T-CNN Model for Sequence Learning: We use three (L=3) parallel 1D convolution layers whose filter sizes are 3,4,5 respectively and number of filters are 200. The first dropout rate and the second one are 0.2 and 0.8, respectively. Since the model is simple, we use a somewhat strong dropout rate to prevent from overfitting. The T-CNN model is trained with a mini-batch size of 64 and the training is terminated after 100 and 300 epochs for UCF101 and HMDB51, respectively.

Training C-LSTM Model for Sequence Learning: The filter size of the 1D convolution layer is 5 and its filter count is 200. The number of hidden units of the first and second LSTM layers is 100 and the dropout rate is set to 0.6. Training is terminated after 100 and 200 epochs for UCF101 and HMDB51, respectively. For both models, we use categorical cross entropy loss with Stochastic Grandient Descent and RMSProp [36] step updates, whose learning rate is initialized with 10^{-4}.

Tesing: Given the trained models (T-CNN, C-LSTM), we evaluate the accuracy with the full sequences for the action recognition task, as well as partial sequences for action prediction. Each video sequence is divided into 10 segments creating the following sequences for action predection [28, 16, 19, 10]: 0~10%, 0~20%, ⋯, 0~100%.

Running Time: The running time of our method is compared with MTSSVM [16], MSSC [3], and Two-stream Fusion [6] methods and the results are listed in Table 2. We executed authors' code on a 4.6GHz CPU with 32GB

RAM and one TITAN-X GPU. With a sequence of 512-dimension weight vectors, the training time is 51min(T-CNN) and 101min(C-LSTM) on UCF101, and 10min(T-CNN) and 67min(C-LSTM) on HMDB51. Note that the testing time takes a few seconds for each dataset. The T-CNN method is $170\times$, $507\times$, $425\times$ faster than MTSSVM, MSSC, Fusion methods, respectively on UCF101. For the HMDB51 dataset, the T-CNN method is $377\times$, $1150\times$, $945\times$ faster than MTSSVM, MSSC, Fusion methods, respectively. The C-LSTM method also spends much less time than compared methods. Note that training time of two-stream ConvNet and feature extraction is not included.

Methods	UCF101 (hrs)	HMDB51 (hrs)
MTSSVM [16]	145	83
MSSC [3]	431	253
Fusion [6] (15 epoch)	362	208
Ours (T-CNN)	0.85	0.22
Ours (C-LSTM)	1.68	1.12

Table 2: Training and testing time of comparison methods in hours on UCF101 and HMDB51.

4.3. Baseline of Two-stream ConvNets

Table 3 shows baseline accuracies for the spatial, temporal, two-stream networks on UCF101 and HMDB51. The value is averaged over three splits and two-stream results are obtain by averaging the prediction probabilities of the spatial and temporal ConvNets. The proposed methods leverage thes baseline two-strema ConvNet and show improvement by taking the temporal information into account.

	UCF101	HMDB51
Spatial	81.8	44.8
Temporal	84.9	55.0
Two-stream	90.1	61.4

Table 3: Baseline mean performance of spatial, temporal, and two-stream ConvNet on UCF101 and HMDB51. (VGG-16 CNN model is employed.)

4.4. Parameter Analysis

Effects of Dimension and Initialization of Weight Vector: We first investigate how the weight vector initialization and feature vector size affect the performance. We experiment by setting parameters: with equal data ratios ($r = 0.5$) for temporal and spatial features, with full testing sequences, and with $K = 20k$. Fig. 5 shows the results with the T-CNN model. The vectors initialized with weight vectors outperforms randomly initialized weight vectors on both datasets and the performance margin is smaller, as the
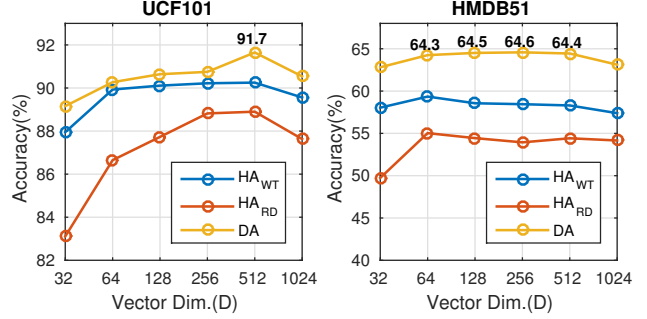

Figure 5: Accuracy based on different initialization and dimension of the weight vector ω. HA_{RD} and HA_{WT} denote random initialization and assigned codebook initialization, respectively.
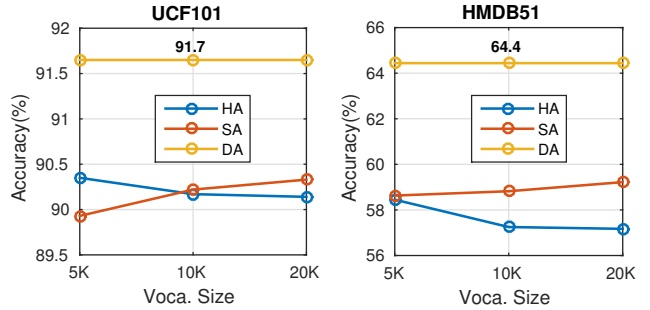

Figure 6: Accuracy based on different size of codebook and different encoding methods.

vector size increases. The randomly initialized vector takes about twice more epochs to be fully trained but data storage can be saved substantially.

In addition, the performance on UCF101 increases as the feature vector dimension increases until 512 with both HA and DA. We speculate this trend occurs because more data is generally helpful but data of size larger than 512 can contain less important data from PCA, so the performance is degraded thereafter. Similar trend happens on the HMDB51 dataset, but no significant performance change is observed between feature vectors of 64 and 512. This means that our method is robust to the choice of the vector dimension results except the 32-dim vector which loses too much information.

Effects of Codebook Size and Encoding Methods. In this experiment, we observe the performance given different codebook sizes and encoding methods. The dimension of the feature vector is fixed to 512, since in the previous experiment the size 512 is found as the most optimal length. The data ratio r is set to 0.5. Fig. 6 shows the results with the T-CNN model. The performance of HA decreases as the codebook size increases, while the SA performance in-
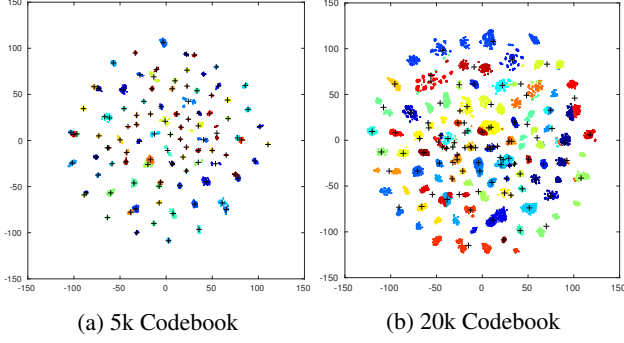
(a) 5k Codebook   (b) 20k Codebook

Figure 7: Visualization of 5k and 20k codebooks ($D = 2$) of UCF101 c. Each codebook is clustered with k-means ($k = 101$).

HMDB51	$r = 0.5$	$r = 0.625$	$r = 0.75$
64	65.2	65.2	66.0
128	65.0	65.6	65.0
256	64.6	65.7	64.6
512	64.8	**66.4**	65.1
UCF101	$r = 0.5$	$r = 0.625$	$r = 0.75$
512	91.5	91.8	**92.7**

Table 4: Performance based on different data ratios and feature dimensions on HMDB51 and UCF101 split 1.

creases with larger codebook. In order to investigate these trends, we reduce 128-dimensional 5k and 20k codebooks on UCF101 to 2-dimensional vectors respectively and cluster them with k-means, where $k = 101$. We employ the t-SNE dimensionality reduction technique [1], which is well suited for displaying high-dimensional data. As shown in Fig. 7, the 5k codebook has larger margin between clusters than the 20k codebook. Therefore, with HA, it is less likely to mislabel with the 5k codebook than the 20k codebook. On the other hand, with SA, the 5 NN codebooks can group more tightly with the 20k codebook, so the centroid of 5NN is likely to be closer to the original feature vector than the centroid in the 5k codebook. In any cases, since the performance gain of different codebook sizes is small, we can argue that our method is robust to the choice of the codebook size. Another distinctive observation is that DA outperforms other encoding methods with relatively large margin.

4.5. Optimal Data Ratio

The temporal and spatial feature vectors are concatenated based on the data ratio r in eq. (10). As shown in Table 3, the temporal network outperforms the spatial network on both datasets. In this analysis, we empirically find an optimal ratio that assigns higher weight to the temporal feature vector.
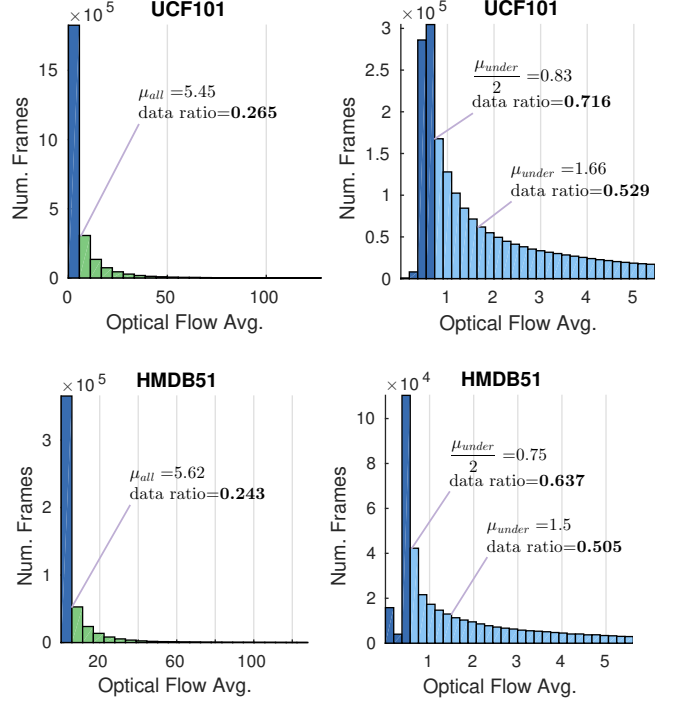

Figure 8: Histogram of average optical flow on UCF101 and HMDB51.

HMDB51		UCF101	
iDT+FV [41]	57.2	iDT+FV [23]	85.9
iDT+HSV [24]	61.1	iDT+HSV [24]	87.9
VideoDarwin [7]	63.7	LRCN [5]	82.9
Two stream [30]	59.4	Two stream [30]	88.0
TDD+FV [42]	63.2	TDD+FV [42]	90.3
KVMF [47]	63.3	KVMF [47]	**93.1**
Fusion [6]	65.4	Fusion [6]	**92.5**
Transformation [43]	62.0	Transformation [43]	92.4
Ours(C-LSTM)	62.4	Ours(C-LSTM)	90.9
Ours(T-CNN)	**66.3**	Ours(CNN)	**92.5**

Table 5: Action recognition performance comparison with State-of-the-art. (mean over three splits)

First, we compute the frame-wise average of optical flow magnitudes along the two axis as follows:

$$f_i = \frac{1}{2} \left(\sum_{k=1}^{P} abs(f_{u_{i,k}} - 128)/P + \sum_{k=1}^{P} abs(f_{v_{i,k}} - 128)/P \right)$$

where f_i is the average optical flow for the i-th frame in the video, P is the total number of pixels in the i-th frame, and f_u, f_v are the horizontal and the vertical optical flow values, respectively. Of course, the intuition is that frames with higher motion information can be identified using f_i.

We explain the choice of r using the histograms of f_i shown in Fig. 8. The left column shows that the frames

UCF101	0-10%	0-20%	0-30%	0-40%	0-50%	0-60%	0-70%	0-80%	0-90%	0-100%
MOS [10]	–	35.0	–	37.1	–	39.4	–	40.3	–	40.9
SMMED[10]	–	40.6	–	40.6	–	40.6	–	40.6	–	40.6
Fusion [6]	**82.8**	85.5	87.5	88.8	89.2	90.4	90.7	91.0	91.5	92.5
Ours	82.2	**86.7**	**88.5**	**89.5**	**90.1**	**91.0**	**91.5**	**91.9**	**92.4**	**92.5**
HMDB51	0-10%	0-20%	0-30%	0-40%	0-50%	0-60%	0-70%	0-80%	0-90%	0-100%
Fusion [6]	**44.8**	51.5	54.5	58.0	61.0	62.9	64.9	65.2	65.4	65.4
Ours	38.8	**51.6**	**57.6**	**60.5**	**62.9**	**64.6**	**65.6**	**66.2**	**66.3**	**66.3**

Table 6: Action Prediction performance on UCF101 and HMDB51.

in the green colored bins contain more motion cues than the frames in the blue colored bins. Also, majority of the frames fall below the mean of the f_i across all frames, i.e. μ_{all}. These are frames that contain less motion information, and hence provide more spatial appearance information. A first order estimate of r could then be given by the ratio of frames above μ_{all} over total number of frames. However, since motion is a stronger cue,it is reasonable to assume that better estimates of r would be given by the first quartile or the half of the first quartile. Therefore, consider the graphs on the right column of Fig. 8, which show the histograms of f_i only for frames whose average optical flow is smaller than μ_{all}. We compute the mean of these lower histograms, denoted as μ_{under}, which determine the first quartile of the original histogram. Better estimates of the ratio r are then given by the ratio of frames above μ_{under} or $\mu_{under}/2$ over the total number of frames.

In our experiments, we found that the ratio r given by μ_{under} is 0.529 on UCF101 and 0.505 on HMDB51 meaning that μ_{under} is close to median of the average optical flows. The estimate based on $\mu_{under}/2$, resulted in $\sim$0.75 for UCF101 and $\sim$0.625 for HMDB51. One observation is that the UCF101 dataset involves many sports and exercise videos [8] that generally contain larger motions, while the HMDB51 dataset consists of simple action videos [8] that have moderate motion. The ratios computed with $\mu_{under}/2$ support this observation. Results using DA and T-CNN for these ratios are shown in Table 4. The best performance is achieved with estimates based on $\mu_{under}/2$, confirming that the estimated ratios are reliable.

4.6. Action Recognition Performance

Table 5 shows action recognition results of recent state-of-the-art methods. Our best result outperforms other methods by 0.9% on HMDB51 and is compatible on UCF101. We conjecture that [47] outperforms ours because they utilize GoogLeNet [34] with batch normalization [11], which is a deeper network than VGG-16 [31]. Our result is on par with Fusion [6] on UCF101 but its computational efficiency is much better due to the fast-trainable network as shown in Table 2. The C-LSTM model, however, does not learn much comparing with the baseline accuracy. We speculate this is

because the temporal 1D convolution without pooling does not represent a video effectively. Applying 1D convolution followed by max pooling over several small segments may boost the performance for the C-LSTM model.

4.7. Action Prediction Performance

The goal in action prediction is the same as in action recognition, except that the input test video is not a full video. Our method can take a variable size input so the partial input can be readily handled. In order to compare with a method using T-CNN, we evaluate Fusion [6] with the partial test video frames. We follow their testing procedure by taking 5 uniformly spaced frames from the given range. The horizontally flipped input frame is augmented and the entire frame is used.

Table 6 show the action prediction results with comparing methods. Our results consistently outperform the Fusion method as well as the previous best results: MOS and SMMED [7]. We observe an interesting trend, in the sense that our result is only outperformed by Fusion in the first 10% range. We conjecture two reasons about the result: the length of the sequence is too short to be fully trained, and noisy words are inserted to the sequence especially on HMDB51. On the other hand, our method rapidly reaches to full accuracy with partial data. The prediction results with half-video data reach 95% and 97% of full accuracy for the HMDB51 and UCF101, respectively. Also, the performance with 90% of frames is almost identical to full accuracy. These observations show that our method is well suitable to detect actions with partial data.

5. Conclusion

We proposed an effective and efficient sequence learning method that captures global temporal sequencing information of a video. This is achieved by means of a new video representation as a sequence of visual words (a sentence). By training a ConvNet to learn the sequences corresponding to different actions, we are able to accurately identify an action or predict it from a partial sentence. The ConvNet architecture is simple and can be trained with minimum computational cost. We also demonstrate how important hyper-

parameters such as data ratio are determined automatically. These parameters play significant roles in improving the accuracy. We achieve compatible state-of-the-art results on both action recognition and action prediction.

References

[1] M. Balamurali and A. Melkumyan. t-sne based visualisation and clustering of geological domain. In *Neural Information Processing - 23rd International Conference, ICONIP 2016, Kyoto, Japan, October 16-21, 2016, Proceedings, Part IV*, pages 565–572, 2016.

[2] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[3] Y. Cao, D. Barrett, A. Barbu, S. Narayanaswamy, H. Yu, A. Michaux, Y. Lin, S. Dickinson, J. Mark Siskind, and S. Wang. Recognize human activities from partially observed videos. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013.

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[5] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, T. Darrell, and K. Saenko. Long-term recurrent convolutional networks for visual recognition and description. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 2625–2634, 2015.

[6] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. *CoRR*, abs/1604.06573, 2016.

[7] B. Fernando, E. Gavves, J. O. M., A. Ghodrati, and T. Tuytelaars. Modeling video evolution for action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 5378–5387, 2015.

[8] F. C. Heilbron, V. Escorcia, B. Ghanem, and J. C. Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 961–970, 2015.

[9] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, Nov. 1997.

[10] D. Huang, S. Yao, Y. Wang, and F. D. la Torre. Sequential max-margin event detectors. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part III*, pages 410–424, 2014.

[11] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[12] R. Johnson and T. Zhang. Effective use of word order for text categorization with convolutional neural networks. *CoRR*, abs/1412.1058, 2014.

[13] N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A convolutional neural network for modelling sentences. *CoRR*, abs/1404.2188, 2014.

[14] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, pages 1725–1732, Washington, DC, USA, 2014. IEEE Computer Society.

[15] Y. Kim. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, 2014.

[16] Y. Kong, D. Kit, and Y. Fu. A discriminative model with multiple temporal scales for action prediction. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*, pages 596–611, 2014.

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, pages 1106–1114, 2012.

[18] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. HMDB: a large video database for human motion recognition. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011.

[19] T. Lan, T. Chen, and S. Savarese. A hierarchical representation for future action prediction. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part III*, pages 689–704, 2014.

[20] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[21] J. Y. Ng, M. J. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. *CoRR*, abs/1503.08909, 2015.

[22] J. C. Niebles, C. Chen, and F. Li. Modeling temporal structure of decomposable motion segments for activity classification. In *Computer Vision - ECCV 2010, 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part II*, pages 392–405, 2010.

[23] D. Oneata, J. Verbeek, and C. Schmid. The LEAR submission at Thumos 2014, 2014. -.

[24] X. Peng, L. Wang, X. Wang, and Y. Qiao. Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice. *Computer Vision and Image Understanding*, 150:109–125, 2016.

[25] X. Peng, C. Zou, Y. Qiao, and Q. Peng. Action recognition with stacked fisher vectors. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*, pages 581–595, 2014.

[26] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[27] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[28] M. S. Ryoo. Human activity prediction: Early recognition of ongoing activities from streaming videos. *2011 IEEE International Conference on Computer Vision (ICCV 2011)*, 00(undefined):1036–1043, 2011.

[29] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. *CoRR*, abs/1503.03832, 2015.

[30] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. *CoRR*, abs/1406.2199, 2014.

[31] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[32] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proceedings of the International Conference on Computer Vision*, volume 2, pages 1470–1477, Oct. 2003.

[33] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *CoRR*, abs/1212.0402, 2012.

[34] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 1–9, 2015.

[35] K. D. Tang, F. Li, and D. Koller. Learning latent temporal structure for complex event detection. In *2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012*, pages 1250–1257, 2012.

[36] T. Tieleman and G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

[37] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 4489–4497, 2015.

[38] J. C. van Gemert, C. J. Veenman, A. W. M. Smeulders, and J. M. Geusebroek. Visual word ambiguity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(7):1271–1283, 2010.

[39] G. Varol, I. Laptev, and C. Schmid. Long-term temporal convolutions for action recognition. *CoRR*, abs/1604.04494, 2016.

[40] H. Wang and C. Schmid. Action Recognition with Improved Trajectories. In *ICCV 2013 - IEEE International Conference on Computer Vision*, pages 3551–3558, Sydney, Australia, Dec. 2013. IEEE.

[41] H. Wang and C. Schmid. Action recognition with improved trajectories. In *IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013*, pages 3551–3558, 2013.

[42] L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 4305–4314, 2015.

[43] X. Wang, A. Farhadi, and A. Gupta. Actions ˜ transformations. *CoRR*, abs/1512.00795, 2015.

[44] Y. Xiong, K. Zhu, D. Lin, and X. Tang. Recognize complex events from static images by fusing deep channels. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 1600–1609, 2015.

[45] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime tv-l1 optical flow. In *In Ann. Symp. German Association Patt. Recogn*, pages 214–223, 2007.

[46] C. Zhou, C. Sun, Z. Liu, and F. C. M. Lau. A C-LSTM neural network for text classification. *CoRR*, abs/1511.08630, 2015.

[47] W. Zhu, J. Hu, G. Sun, X. Cao, and Y. Qiao. A key volume mining deep framework for action recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 1991–1999, 2016.