Hierarchical Recurrent Neural Encoder for Video Representation with Application to Captioning

Pingbo Pan[§] Zhongwen Xu[†] Yi Yang[†] Fei Wu[§] Yueting Zhuang[§]
[§]Zhejiang University [†]University of Technology Sydney
{lighnt001,zhongwen.s.xu}@gmail.com yi.yang@uts.edu.au
{wufei,yzhuang}@cs.zju.edu.cn

Abstract

Recently, deep learning approach, especially deep Convolutional Neural Networks (ConvNets), have achieved overwhelming accuracy with fast processing speed for image classification. Incorporating temporal structure with deep ConvNets for video representation becomes a fundamental problem for video content analysis. In this paper, we propose a new approach, namely Hierarchical Recurrent Neural Encoder (HRNE), to exploit temporal information of videos. Compared to recent video representation inference approaches, this paper makes the following three contributions. First, our HRNE is able to efficiently exploit video temporal structure in a longer range by reducing the length of input information flow, and compositing multiple consecutive inputs at a higher level. Second, computation operations are significantly lessened while attaining more non-linearity. Third, HRNE is able to uncover temporal transitions between frame chunks with different granularities, i.e. *it can model the temporal transitions between frames as well as the transitions between segments. We apply the new method to video captioning where temporal information plays a crucial role. Experiments demonstrate that our method outperforms the state-of-the-art on video captioning benchmarks.*

1. Introduction

Incorporating temporal information into video representation has long been a fundamental problem in computer vision. Earlier works such as Dense Trajectories [39] and improved Dense Trajectories (iDT) [40] typically utilize optical flow to extract temporal information and hand-crafted features to model appearances and motions. With the recent success of deep Convolutional Neural Networks (ConvNets) both in efficiency and efficacy, we have witnessed a new trend in leveraging ConvNets to infer video representation. Xu *et al*. [41] propose to utilize Vector of Locally Aggregated Descriptors (VLAD) [13] to aggregate frame level ConvNet for video representation, which is unable to capture temporal structure. Simonyan and Zisserman [26] combine stacked optical flow frames and RGB streams to train ConvNets for video classification, which achieves comparable performance to iDT in action recognition. A limitation of two-stream ConvNets [26] and iDT [40] is that both algorithms require optical flow as input, which is expensive to extract (it takes usually 0.06 seconds to extract optical flow between a pair of frames [26]), but is only able to capture temporal information in video clips of short duration.

To avoid extracting optical flow, 3D ConvNets are proposed in [35] to generate a video representation, with emphasis on efficiency improvement. This approach, however, can only cope with 16 frames or so each time [35]. Very recently, Long Short-Term Memory (LSTM) [11] has been applied to video analysis [20], inspired by the general recurrent encoder-decoder framework [31]. A plausible feature of LSTM is that LSTM is capable of modeling data sequences. However, as this paper tries to cope with, there are still a few challenges remain unaddressed.

First, a large number of long-range dependencies are usually difficult to capture. Even though LSTM can deal with long video clips in principal, it has been reported that the favorable length of video clips to be feed into LSTM falls in the range of 30 to 80 frames [20, 37]. In order to model longer video clips while attaining similar good performance as in [20, 37], we propose to divide a long video clip into a few short frame chunks, feed the chunks into LSTM, and composite the LSTM outputs of the frame chunks into one vector, which can then be fed into another LSTM at a higher level to uncover the temporal information among the composited vectors over a longer duration. Such a hierarchical structure significantly reduces the length of input information flow but is still capable of exploiting temporal information over longer time afterwards at a higher level.

Second, additional non-linearity has been demonstrated helpful for improving model training for visual tasks such

as image and video classification [27, 31, 20]. A straightforward way of adding non-linearity into LSTM is stacking [31, 20]. Despite of the improved performance, a major disadvantage of stacking is that it introduces a long path from the input to the output video vector representation, thereby resulting in heavier computational cost. As we will discuss in details later, Hierarchical Recurrent Neural Encoder (HRNE) proposed in this paper dramatically shortens the path with the capability of adding non-linearity, providing a better trade-off between efficiency and effectiveness.

Third, video temporal structures are intrinsically layered. Suppose a video of birthday party consists of three actions, *e.g*., blowing candles, cutting cake, and eating cake. As the three actions usually take place sequentially, *i.e*., there are strong temporal dependencies among them, we need to appropriately model the temporal structure among the three actions. In the meantime, the temporal structure within each action should also be exploited. To this end, we need to model video temporal structure with multiple granularities. Unfortunately, straightforward implementation of LSTM can not achieve this goal.

The proposed HRNE framework models video temporal information using a hierarchical recurrent encoder and can effectively deal with the three aforementioned challenges. While HRNE is a generic video representation, we apply it to video captioning to test the performance, because temporal information plays a key role in video captioning. Two widely-used video captioning datasets, the Microsoft Research Video Description Corpus (MSVD) [5] and the Montreal Video Annotation Dataset (M-VAD) [34], are used in our experiments, which demonstrate the effectiveness of the proposed method.

2. Related Works

Dense Trajectories [39] and its improved version: improved Dense Trajectories [40] have dominated the filed of action recognition and general video classification tasks such as complex event detection. Dense Trajectories applies dense sampling to get the interest points along the video and then tracks the points in a short time period. Local descriptors such as HOG, HOF and MBH are extracted along the tracklets. Bag-of-Words (BoWs) [28] and Fisher vector encoding [25] are then applied to accumulate the local descriptors and generate the video representation.

Besides the hand-crafted visual features like Dense Trajectories, researchers have started exploring the Convolutional Neural Networks (ConvNets) on video representation recently. Karpathy *et al*. [15] first introduce ConvNets which are similar with Krizhevsky *et al*. [17] into video classification, and different fusion strategies are explored to combine information over the temporal domain in this work. In order to better capture temporal information in action recognition, Simonyan and Zisserman [26] propose to utilize stacked optical flow frames as inputs to train the ConvNets, which, together with the RGB stream, achieves comparable performance as the state-of-the-art hand-crafted features [40] on action recognition. Tran *et al*. [35] utilize 3D ConvNets to learn temporal information without optical flows, which is inspired by Ji *et al*. [14] and Simonyan and Zisserman [27]. Xu *et al*. [41] propose to utilize VLAD [13] aggregation on frame-level ConvNet features and it directly adapts ImageNet pretrained image classification model to video representation.

All these works mentioned above utilize either average pooling or encoding methods such as Fisher vector and VLAD over time to generate a global video feature from a set of local features. However, time dependency information is lost since average pooling and encoding methods always ignore the order of the input sequences, *i.e*., taking the local features as a *set* rather than a *sequence*. To tackle this problem, Ng *et al*. [20] introduce Long Short-Term Memory (LSTM) to model the temporal order, inspired by the general sequence to sequence learning neural model proposed by Sutskever *et al*. [31]. Stacked LSTM is applied in [20], where each layer of the LSTM retains the same time scale. Different from the standard approach which stackes LSTM layers into multilayered one and simply aims to introduce more non-linearity into the neural model, the *Hierarchical Recurrent Neural Encoder* proposed in this work aims to abstract the visual information at different time scales, and learns the visual features with multiple granularities.

In the application of video captioning, Donahue *et al*. [9] introduce the LSTM into this task by feeding the Conditional Random Field (CRF) outputs of objects, subjects, and verbs into the LSTM to generate video description. The same as [9], other works such as [38, 42, 21] utilize the LSTM essentially as a recurrent neural network *language model* to generate video descriptions, which conditions on either the average pooled frame-level features or the context vector linearly blended by the attention mechanism [1]. In contrast to these works, we study better video content understanding from the visual feature aspects instead of language modeling ones. Based on stacked LSTM, Venugopalan *et al*. [37] is the only attempt to utilize LSTMs as both visual encoder and language decoder in the video captioning task, which is inspired by the general neural encoder-decoder framework [31, 7] as well.

In the area of query suggestion, Sordoni *et al*. [29] propose a hierarchical recurrent neural network for context-aware query suggestion in a search engine. In this model, the text query in a session is firstly abstracted by one RNN layer into the query-level state, another RNN layer is used to learn session-level dependency and then, the session-level hidden states is utilized to make suggestions for users.

Contemporary to this work, Yu *et al*. [43] introduce a hierarchical RNN *decoder*, specifically Gated Recurrent Unit

(GRU) [7], into the video captioning system. A sentence generator consisting of a GRU layer conditions on visual feature, and then a paragraph generator accepts sentence vector and the context to generate paragraph level description, which essentially learns the time dependencies between sentences, and works on the language processing aspects. In contrast, this work is focusing on learning good *visual feature*, *i.e.*, the encoder part, but not the *language processing*, *i.e.*, the decoder part.

3. The Proposed Approach

We propose a Hierarchical Recurrent Neural Encoder (HRNE) model for video processing tasks. Assume we have n frames in the video, based on the HRNE model, we develop a general video encoder which takes the frame-level visual features from a video sequence $(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n)$ as input and outputs a single vector $\mathbf{v}$ as the representation for the whole video. For the video captioning task specifically, we keep the single layer LSTM decoder as a recurrent neural network language model [31], which conditions on the video feature vector $\mathbf{v}$, similar to previous works [37, 42].

3.1. The Recurrent Neural Network

The recurrent neural network is a natural extension of feedforward neural networks on modeling sequence. Given an input sequence $(\mathbf{x}_1, \mathbf{x}_2..., \mathbf{x}_n)$, a standard RNN computes the output sequence $(\mathbf{z}_1, \mathbf{z}_2..., \mathbf{z}_n)$ by iterating the following equations:

$$\mathbf{h}_t = \tanh(W_{hx}\mathbf{x}_t + W_{hh}\mathbf{h}_{t-1}), \quad (1)$$

$$\mathbf{z}_t = W_{zh}\mathbf{h}_t, \quad (2)$$

The RNN can map the inputs to the outputs whenever the alignment between inputs and outputs is provided. The standard RNN would work principally, but it is really difficult to train the standard RNN due to the vanishing gradient problem [3]. The Long Short-Term Memory (LSTM) is known to learn patterns with wider range temporal dependencies. We now introduce the LSTM model.

The core of the LSTM model is a memory cell $\mathbf{c}_t$ which records the history of the inputs observed up to that time step. $\mathbf{c}_t$ is a summation of the previous memory cell $\mathbf{c}_{t-1}$ modulated by a sigmoid gate $\mathbf{f}_t$, and $\mathbf{g}_t$, a function of previous hidden state and the current input modulated by another sigmoid gate $\mathbf{i}_t$. The sigmoid gates can be thought as knobs that LSTM learns to selectively forget its memory or accept current input. The cell has three gates. The input $\mathbf{i}_t$ gate controls whether the LSTM will consider current input $\mathbf{x}_t$. The forget gate $\mathbf{f}_t$ is used to control whether LSTM will forget the previous memory $\mathbf{c}_{t-1}$. The output gate $\mathbf{o}_t$ controls how much information will be transferred from memory $\mathbf{c}_t$ to hidden state $\mathbf{h}_t$. There are several widely used LSTM variants and we use the LSTM unit described in [44] in our model, which iterates as follows:

$$\mathbf{i}_t = \sigma(W_{ix}\mathbf{x}_t + W_{ih}\mathbf{h}_{t-1} + \mathbf{b}_i), \quad (3)$$

$$\mathbf{f}_t = \sigma(W_{fx}\mathbf{x}_t + W_{fh}\mathbf{h}_{t-1} + \mathbf{b}_f), \quad (4)$$

$$\mathbf{o}_t = \sigma(W_{ox}\mathbf{x}_t + W_{oh}\mathbf{h}_{t-1} + \mathbf{b}_o), \quad (5)$$

$$\mathbf{g}_t = \phi(W_{gx}\mathbf{x}_t + W_{gh}\mathbf{h}_{t-1} + \mathbf{b}_g), \quad (6)$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t, \quad (7)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \phi(\mathbf{c}_t), \quad (8)$$

where σ is the sigmoid function, ϕ is the hyperbolic tangent function $\tanh$, $\odot$ donates element-wise product, W_{*x} is the transform from the input to LSTM states, W_{*h} is the recurrent transformation matrix between the hidden states and $\mathbf{b}_*$ is the biases vector.

3.2. Hierarchical Recurrent Neural Encoder

It has been reported that adding more non-linearity is helpful for vision tasks [27]. The performance of LSTM can be improved if additional non-linearity is added. A straightforward way is stacking multiple layers, which, however, will increase computation operations. Inspired by the ConvNet operations in spatial domain, we propose a Hierarchical Recurrent Neural Encoder (HRNE) model. As shown in Figure 1, in a ConvNet model, a filter is used to explore the spatial visual information of an image by performing convolution calculation between image patch matrix I and a learnable filter matrix H:

$$y = \sum_{i=1}^{h}\sum_{j=1}^{w} I_{i,j}H_{i,j}, \quad (9)$$

where w denotes the number of columns of the filter matrix, h denotes the number of rows, $H_{i,j}$ denotes the matrix item located in the i-th row and j-th column and y is the convolution calculation result. The filter is applied over the whole image to generate the filtered image, which is further forwarded into the next layer. Similarly, in temporal domain, we introduce an additional layer, instead of stacking, by which only short LSTM chains need to be dealt with. The filters in ConvNet's convolutional layer are well suited for exploring local spatial structure. Analogously, using temporal filter to explore the local temporal structure is presumed to be beneficial since videos always consist of several incoherence clips.

The main difficulty of introducing additional layers into temporal modeling is finding a proper temporal filter. In spatial domain, the output of filter is independent from spatial location, and a matrix can be used as a filter. Differently, in temporal domain, there is certain temporal dependencies between consecutive items. As a result, a matrix is not sufficient to be used as a temporal filter. Since RNN is well

(a) Spatial convolutional operations of ConvNet (b) Temporal operations of Hierarchical Recurrent Neural Encoder

Figure 1: Analogical illustration of temporal operations of HRNE to spatial convolutional operations of ConvNet. In ConvNet a learnable filter is applied to each location to generate a filtered image which is further forwarded to the next layer. In HRNE, a learnable filter (*i.e.*, LSTM) along with attention mechanism is applied to each temporal time step to generate a sequence of video chunk vectors, which are further forwarded to the next layer.

suited for temporal dependency modeling, we adopt short RNN chains as the temporal filters in our HRNE model. Specifically, we use LSTM chains in this paper and take the mean of all LSTM chain's hidden states as the filtering result.

We first divide an input sequence $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$ into several chunks $(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n)$, $(\mathbf{x}_{1+s}, \mathbf{x}_{2+s}, \ldots, \mathbf{x}_{n+s})$, $\ldots$, $(\mathbf{x}_{T-n+1}, \mathbf{x}_{T-n+2}, \ldots, \mathbf{x}_T)$, where s is stride and it denotes the number of temporal units two adjacent chunks are apart. After inputting these subsequences into the LSTM filter, we will get a sequence of feature vectors $\mathbf{h}_1, \mathbf{h}_2, .., \mathbf{h}_{\lceil T/n \rceil}$, where $\lceil x \rceil$ denotes the least integer among those integers which are larger than x. Each feature vector in $\mathbf{h}_1, \mathbf{h}_2, .., \mathbf{h}_{\lceil T/n \rceil}$ gives a proper abstract of its corresponding clip. To get the feature vector of the whole video, we propose to use another LSTM layer to summarize all these feature vectors. We combine these two LSTM layers and build our HRNE model. The first LSTM layer serves as a filter and it is used to explore local temporal structure within subsequences. The second LSTM learns the temporal dependencies among subsequences. We note that more complex HRNE model could be adding more layers to build multiple time-scale abstraction of the visual information.

A large number of long-range dependencies are usually difficult to capture. Even though LSTM can deal with long video clips in principal, we compare HRNE with stacked multilayered LSTM in Figure 2. The red line in the Figure 2 shows how the input at $t = 1$ flows though the model to the final output. We are used to set the stride to be the same as the LSTM filter length. For an input sequence of length T and a LSTM filter of length n, the red line in HRNE model goes through $n + \lceil T/n \rceil$ LSTM units, which means the input at $t = 1$ will only flow through $n + \lceil T/n \rceil$ steps to the output rather than $T + 1$ steps if stacked RNN is used. If $T = 1,000$ and we set n to be 30, then HRNE will only go through 64 steps rather than 1,001 steps. Fewer steps an input will go through before it reaches the output means that it's easier to backtrack, so our HRNE is easier for stochastic gradient methods via Back-propagation Through Time (BPTT) to train.

Since the recently proposed soft attention mechanism from [1] has achieved great success in several sequence modeling tasks, we integrate the attention mechanism into our HRNE model. We next introduce the attention mechanism part.

The core of the soft attention mechanism is that instead of just inputting the original sequence $(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n)$ into a LSTM layer, dynamic weights are used to generate a new sequence $(\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_m)$:

$$\mathbf{v}_t = \sum_{i=1}^{n} \alpha_i^{(t)} \mathbf{x}_i, \tag{10}$$

where $\sum_{i=1}^{n} \alpha_i^{(t)} = 1$ and $\alpha_i^{(t)}$ will be calculated by an attention neural network at each time step $t = 1, 2, \ldots, m$.

The attention weight $\alpha_i^{(t)}$ actually measures the relevance between the i-th element $\mathbf{x}_i$ of the input sequence and the history information recorded by the LSTM $\mathbf{h}_{t-1}$. Hence a function is needed to calculate the relevance score:

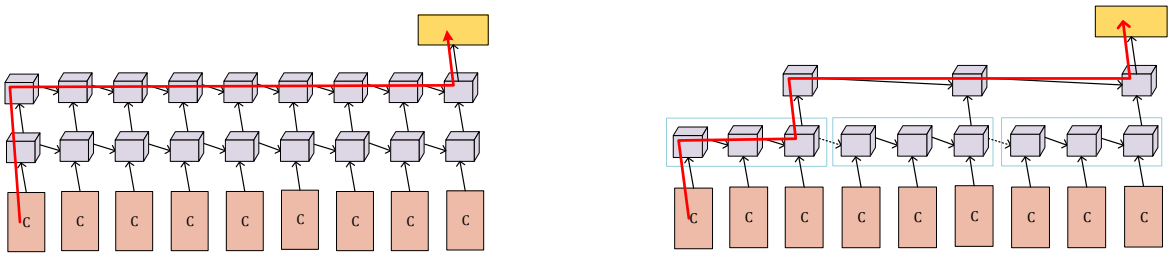
$$e_i^{(t)} = w^\top \tanh(W_a \mathbf{x}_i + U_a \mathbf{h}_{t-1} + b_a), \tag{11}$$

where w, W_a, U_a, b_a are all parameters and $\mathbf{h}_{t-1}$ is the hidden state of the LSTM at $(t-1)$-th time step.

We need to calculate $e_i^{(t)}$ for $i = 1, 2, ..., n$ and then $\alpha_i^{(t)}$ could be calculated by:

$$\alpha_i^{(t)} = \exp(e_i^{(t)}) / \sum_{j=1}^{n} \exp(e_j^{(t)}). \tag{12}$$

The attention mechanism could make the LSTM pay attention to different temporal locations of the input sequence



(a) Stacked LSTM video encoder

(b) Hierarchical Recurrent Neural Encoder

Figure 2: A comparison between stacked LSTM and the proposed Hierarchical Recurrent Neural Encoder. This figure takes a two layer hierarchy as an example to showcase. The red line in each subfigure shows one of the paths from the visual appearance input at $t = 1$ to the output video vector representation. There are 10 time steps in stacked LSTM and only 6 time steps in our model.

according to its backprop information, and when the input sequence and the output sequence are not aligned strictly, attention would especially be helpful. We add attention units in three different positions in our video caption model: between the visual input and the LSTM filter, between the output of the filter and the second LSTM layer, between the output of our HRNE and the description decoder.

3.3. Video Captioning

Our HRNE can be applied to several video processing tasks where feature vectors are required to represent videos. In this paper, we use video captioning, where temporal information plays an important role, to showcase the advantage of the proposed method.

We develop our video captioning model based on the general sequence to sequence model [31], *i.e.*, encoder-decoder framework, which is same as the previous works [42, 37]. We use the general video encoder to map video sequences to feature vectors and then one-layer LSTM decoder conditioned on the video feature vector to generate description for the video.

The overall objective function we are optimizing is the log-likelihood over the whole training set,

$$\max_{\mathbf{\Theta}} \sum_{t=1}^{T} \log \Pr(\mathbf{y}_t|\mathbf{z}, \mathbf{y}_{t-1}; \mathbf{\Theta}), \tag{13}$$

where $\mathbf{y}_t$ is a one-hot vector (1-of-N coding, where N is the size of the word vocabulary) used to represent the word at the t-th time step, $\mathbf{z}$ is the feature vector output by the video encoder and $\mathbf{\Theta}$ represents the video captioning model's parameters.

Similar to most recurrent neural network language models, we utilize a softmax layer to model the probability distribution of the next word over the word space, *i.e.*,

$$\Pr(\mathbf{y}_t|\mathbf{z}, \mathbf{y}_{t-1}; \mathbf{\Theta}) \propto \exp(\mathbf{y}_t^\top W_y \mathbf{s}_t), \tag{14}$$

where

$$\mathbf{s}_t = \tanh(W_z\mathbf{z} + W_h\mathbf{h}_t + W_e\mathbf{y}_{t-1} + \mathbf{b}), \tag{15}$$

and W_y, W_z, W_h, W_e and $\mathbf{b}$ are all the parameters.

Eqn (15) is an instance of deep output layer proposed in Pascanu *et al*. [23] and we find incorporating the deep output layer helps the model to converge faster and gets better performance. To make the model more robust, we adopt the Maxout [10] scheme to calculate $\mathbf{s}_t$.

4. Experimental Setup

We utilize two standard video captioning benchmarks to validate the performance of our proposed method in the experiments: the widely used Microsoft Video Description Corpus (MSVD) [5] and one recently proposed dataset the Montreal Video Annotation Dataset (M-VAD) [34].

4.1. The Datasets

The Microsoft Video Description Corpus (MSVD): The Microsoft Video Description Corpus (MSVD) [5] contains 1,970 videos with multiple descriptions labeled by the Amazon Mechanical Turkers. Annotators are requested to provide a single sentence description to a picked up short clips. The total number of clip-description pairs is about 80,000. The original dataset consists of multi-lingual descriptions while we only focus on the English description as the previous works [38, 37, 42]. We utilize the standard splits provided in [38] for fair comparisons with state-of-the-art video captioning systems [38, 37, 42], which separate the original dataset into training, validation and testing with 1,200 clips, 100 clips, and the remaining clips, respectively.

The Montreal Video Annotation Dataset (M-VAD): The Montreal Video Annotation Dataset (M-VAD) is a newly collected large-scale video description dataset from the DVD descriptive video service (DVS) narrations. There

are 92 DVD movies in the M-VAD dataset, which is further divided into 49,000 video clips. Each clip in the video has one corresponding narration as the groundtruth of the clip description. Since the narrations are generated in a semi-automatically transcribed way, the grammar used in the description is much more complicated than the one in MSVD. Same as previous works [42, 37], we utilize the standard splits provided in [34], which consists of 39,000 clips in the training set, 5,000 clips in the validation set, and 5,000 clips in the testing set.

4.2. Preprocessing

Visual Features: We use GoogLeNet [32, 12] to extract the frame-level features in our experiment. All the videos' lengths are kept to 200 frames. For a video with more than 200 frames, we drop the extra frames. For a video without enough frames, we pad zero frames. These are common approaches to ensure all the videos have the same length [38, 43]. Instead of directly inputting the features into HRNE, we learn a linear embedding of the features as the input of our model.

Description preprocessing: We convert all descriptions to lower case, and use the PTBTokenizer in Stanford CoreNLP tools[1] [19] to tokenize sentences and remove punctuation. This yields a vocabulary of 12,976 in size for the MSVD dataset and a vocabulary of 15,567 in size for the M-VAD dataset.

4.3. Evaluation Metrics

Several standard metrics such as BLEU [22], METEOR [8], ROUGE-L [18] and CIDEr [36] are used commonly for evaluating visual captioning tasks, mainly following the machine translation field. The authors of [36] evaluated the above four metrics in terms of the consistency with human judgment, and found that METEOR is always better than BLEU and ROUGE. Thus, *METEOR is used as the main metric in the evaluation*. We utilize the Microsoft COCO evaluation server [6] to obtain all the results reported in this paper, which makes our results directly comparable with the previous works.

4.4. Compared Algorithms

- FGM [33]: It first obtains confidences on subject, verb, object and scene elements. Then a factor graph model is used to infer the most likely (subject, verb, object) tuple in the video. Finally it generates sentence based on a template.

- Average pooling + LSTM decoder [38] (denoted as Mean pool): It uses the average pooling frame-level feature to represent the whole video. Then LSTM is utilized as a recurrent language model to produce the description given the visual feature.

- S2VT [37]: It first introduces stacked LSTM as an encoder-decoder model to video captioning tasks. It consists of two phases. In the first phase, it serves as a video encoder and in the second phase, it stops accepting video sequence and begins generating video descriptions.

- Temporal Attention [42] (SA): It applies attention mechanism on temporal locations and then utilizes the recurrent language model LSTM to generate the video description.

- LSTM embdding [21] (LSTM-E): It uses embedding layers to project the visual feature and text feature into one space, with a modified loss between description and visual features.

- Paragraph RNN decoder [43] (p-RNN): It introduces a hierarchical structure in *decoder* for language processing and introduce the paragraph description in addition to the standard sentence description.

4.5. Training Details

In the training phase, we add a begin-of-sentence tag <BOS> to start each sentence and an end-of-sentence tag <EOS> to end each sentence, so that our captioning model can deal with sentences of varying lengths. In the testing phase, we input <BOS> into video decoder to start generating video descriptions and during each step, we choose the word with the maximun probability after softmax until we reach <EOS>.

We adopt different parameter settings to train different datasets.When we are training on MSVD, we use the following settings: All the LSTM units are set to 1,024, the visual feature embedding size and the word embedding size are set as 512 empirically. When training on M-VAD, we find our HRNE is easier to overfit than in MSVD, so we set all the LSTM units to be 512 and still keep the visual feature embedding size and the word embedding size to be half of the number of LSTM units. As the videos in the two datasets are very short, a two-layer HRNE is sufficient to capture the temporal structure of videos. Nevertheless, one may use HRNE with more layers to deal with longer videos.

The length of the LSTM chain at the bottom layer is 8, and we set the stride to be 8 in all the experiments. We set the size of mini-batch as 128. We apply the first-order optimizer ADAM to minimize the negative log-likelihood loss for the training process and we set the learning rate $\eta = 2 \times 10^{-4}$, the decay parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$ as defaulted in Kingman and Ba [16], which generally shows good performance and does not need heavily tuned. Since

[1] version 3.4.1

we observe serious overfitting problems when training our model on M-VAD dataset, we apply the simple yet effective neural model regularization method Dropout [30] with rate of 0.5 on the input and the output of LSTMs but not on the recurrent transitions as suggested by Zaremba *et al*. [45]. We find that the proposed model has better generalization ability in this way, empirically. More details of parameter choice can be found in the supplementary material.

We train the model for 400 epoches, or stop the training until the evaluation metric does not improve on the validation set. We utilize Theano [2, 4] framework to conduct our experiments.

5. Experimental Results

We evaluate our HRNE model on video captioning on both MSVD and M-VAD. We report results on MSVD in Table 1 and Table 2. We firstly report the results only using static frame-level features in Table 1. We additionally compare our HRNE with GoogLeNet feature to other video captioning systems which combine multiple ConvNet features in Table 2. We also report the result of our HRNE with fusion of GoogLeNet feature and C3D feature [35]. Lastly, we conduct the experiment on the more challenging dataset M-VAD, and report the results in Table 4 .

5.1. Experiment results on the MSVD dataset

We report experiment results where only static frame-level features are used in Table 1 on the MSVD dataset. Our method achieves better result than S2VT, which indicates our hierarchical structure increases the learning capability and enables our model encode richer temporal information of multiple granularities. Both Mean pool and SA ignore temporal dependencies along video sequences. They adopt the weighted averages of frame-level features to represent videos. Our HRNE outperforms both Mean pool and SA, due to the exploration of temporal information of videos. Hierarchical description decoder is adopted in p-RNN to generate complex descriptions, while our HRNE has better performance than p-RNN, which indicates exploring temporal information of videos is more important for video captioning. To further improve our HRNE, we add attention mechanism, which again improves its performance.

We additionally compare our HRNE to other video captioning systems with fusion in Table 2. We first compare our HRNE with only GoogLeNet feature to systems which combine multiple ConvNet features. Our HRNE achieves the best result in METEOR. It means although adding more features helps improve video captioning systems' performance, our method still achieves the best performance. This result confirms the effectiveness of our HRNE. We notice that p-RNN outperforms our HRNE in terms of BLEU. However, our method outperforms p-RNN in almost all other cases (see Table 1 and Table 2) and, more importantly, as demonstrated in [36], METEOR is more reliable than BLEU. Our HRNE with fusion of GoogLeNet faeture and C3D feature indicates adding more features can improve the performance of our method, which is consistent with [21, 43].

Model	METEOR	B@1	B@2	B@3	B@4
FGM [33]	23.9	-	-	-	-
Mean pool [42]	28.7	-	-	-	38.7
SA [42]	29.0	-	-	-	40.3
S2VT [37]	29.2	-	-	-	-
LSTM-E [21]	29.5	74.9	60.9	50.6	40.2
p-RNN [43]	31.1	77.3	64.5	54.6	**44.3**
HRNE	32.1	78.4	66.1	**55.1**	43.6
HRNE with attention	**33.1**	**79.2**	**66.3**	**55.1**	43.8

Table 1: Experiment results on the MSVD dataset. We compare our method with the baselines using *static frame-level features only* in this table.

Model	METEOR	B@1	B@2	B@3	B@4
Mean pool-(G) [42]	28.7	-	-	-	38.7
S2VT-(V)-(A) [37]	29.8	-	-	-	-
SA-(G)-(C) [42]	29.6	-	-	-	41.9
LSTM-E-(A) [21]	28.3	74.5	59.8	49.3	38.9
LSTM-E-(V) [21]	29.5	74.9	60.9	50.6	40.2
LSTM-E-(C) [21]	29.9	75.7	62.3	52.0	41.7
LSTM-E-(V)-(C) [21]	31.0	78.8	66.0	55.4	45.3
p-RNN-(V) [43]	31.1	77.3	64.5	54.6	44.3
p-RNN-(C) [43]	30.3	79.7	67.9	57.9	47.4
p-RNN-(V)-(C) [43]	32.6	**81.5**	**70.4**	**60.4**	**49.9**
HRNE-(G)	32.1	78.4	66.1	55.1	43.6
HRNE with attention-(G)	33.1	79.2	66.3	55.1	43.8
HRNE with attention-(C)	31.0	74.9	61.0	49.9	39.3
HRNE with attention-(G)-(C)	**33.9**	81.1	68.6	57.8	46.7

Table 2: Experiment results on the MSVD dataset with fusion. (A) denotes AlexNet, (V) denotes VGGNet, (C) denotes C3D and (G) denotes GoogLeNet in the model's name.

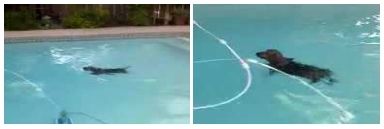
Compared with stacked LSTM, our HRNE can significantly reduce the computation operations. We provide experiment analysis in the supplementary material.

In Table 3, we show a few examples of the descriptions generated by our method. We notice that our HRNE can generate an accurate description of the video even in some difficult cases. In addition, the results with the attention mechanism is generally better than those without the attention mechanism, which is consistent with the results reported in Table 1 and Table 2.

5.2. Experiment results on the M-VAD dataset

Table 4 reports the results on M-VAD. Compared with MSVD, M-VAD is a more challenging dataset, because it contains more visual concepts and complex sentence structures. Since the result on BLEU metric is close to 0[2], we do not consider BLEU metric in this experiment. Our HRNE achieves 5.8% in METEOR, which outperforms both S2VT

[2]SA [42] achieves only 0.7% BLEU-4 on this dataset.



HRNE: A man is swimming in the water.
HRNE with attention: A dog is swimming.
Ground truth: A dog is swimming in a pool.

HRNE A man is playing a guitar
HRNE with attention: A man is playing a guitar.
Ground truth: A boy is playing a guitar.

HRNE: A woman is adding noodles into a pot
HRNE with attention: A woman is cooking.
Ground truth: A woman dips a shrimp in batter.

HRNE: A man is playing a guitar.
HRNE with attention: A man is playing a guitar
Ground truth: A man plays a guitar.

HRNE: A group of people are dancing.
HRNE with attention: A group of people are dancing
Ground truth: A group of young girls are dancing on stage.

HRNE: A person is preparing an egg.
HRNE with attention: A woman is peeling a mango.
Ground truth: A mango is being sliced.

HRNE: A girl is talking.
HRNE with attention: A young girl is talking on a telephone
Ground truth: A woman dials a cell phone.

HRNE: A man is riding a bike.
HRNE with attention: A man is riding a bike
Ground truth: A biker rides along the beach.

HRNE: A man is doing a machine.
HRNE with attention: A man is doing a dance.
Ground truth: A basketball player is doing a hook shot.

Table 3: Example results on the MSVD Youtube video dataset. We present the video descriptions generated by our HRNE.

Model	METEOR
SA-GoogLeNet+3D-CNN [42]	5.7
SA-GoogLeNet+3D-CNN [42][4]	4.1
S2VT-RGB(VGG) [37]	6.7
HRNE	5.8
HRNE (with attention)	**6.8**

Table 4: Experiment results on the M-VAD dataset.

and SA[3]. After adding the attention mechanism, our performance (in METEOR) is further improved from 5.8% to 6.8%. Such performance even outperforms S2VT which combines M-VAD and MPII-MD [24] for training. Because combining two datasets introduces much more training data than just one dataset as the standard setting we used for training, this result again validates the effectiveness of our HRNE.

6. Conclusions and Future Work

In this paper, we proposed a new method, namely Hierarchical Recurrent Neural Encoder (HRNE), to generate video representation with emphasis on temporal modeling. Compared to existing approaches, the proposed HRNE is more capable of video modeling because 1) HRNE reduces the length of input information flow and exploits temporal structure in longer range at a higher level; 2) more non-linearity and flexibility are added in HRNE; and 3) HRNE exploits temporal transitions with multiple granularities. Extensive experiments in video captioning demonstrate the efficacy of HRNE.

Last but not least, the proposed video representation is generic which can be applied to a wide range of video analysis applications. We will explore the application of the encoder on video classification in the future work, which plugs with a softmax classifier upon the encoder and video labels instead of the LSTM language decoder in this work to validate the generalization capability of this framework.

[3]Only S2VT and SA have reported result on this challenging dataset.

[4][37] notes that [42] achieves 4.1% METEOR with the same evaluation script as [37], while the 5.7% METEOR reported in [42] is caused by different tokenization.

7. Acknowledgement

This work was in part supported by the 973 Program(No.2012CB316400), the NSFC (No. U1509206),the China Knowledge Centre for Engineering Sciences and Technology (CKCEST); in part supported by the Data to Decisions Cooperative Research Centre www.d2dcrc.com; in part supported by the ARC DECRA and DP.

References

[1] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

[2] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio. Theano: new features and speed improvements. NIPS Workshop, 2012.

[3] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166, 1994.

[4] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: a CPU and GPU math expression compiler. In *SciPy)*, June 2010.

[5] D. L. Chen and W. B. Dolan. Collecting highly parallel data for paraphrase evaluation. In *ACL*, 2011.

[6] X. Chen, H. Fang, T.-Y. Lin, R. Vedantam, S. Gupta, P. Dollar, and C. L. Zitnick. Microsoft COCO captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*, 2015.

[7] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*, 2015.

[8] M. Denkowski and A. Lavie. Meteor universal: Language specific translation evaluation for any target language. In *EACL*, 2014.

[9] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015.

[10] I. Goodfellow, D. Warde-farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. In *ICML*, 2013.

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

[12] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ICML*, 2015.

[13] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *CVPR*, 2010.

[14] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *TPAMI*, 35(1):221–231, 2013.

[15] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.

[16] D. Kingma and J. Ba. ADAM: A method for stochastic optimization. In *ICLR*, 2015.

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012.

[18] C.-Y. Lin. ROUGE: A package for automatic evaluation of summaries. In *ACL workshop*, 2004.

[19] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. The Stanford CoreNLP natural language processing toolkit. In *ACL*, 2014.

[20] J. Y.-H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, 2015.

[21] Y. Pan, T. Mei, T. Yao, H. Li, and Y. Rui. Jointly modeling embedding and translation to bridge video and language. *CVPR*, 2016.

[22] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. BLEU: a method for automatic evaluation of machine translation. In *ACL*, 2002.

[23] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*, 2013.

[24] A. Rohrbach, M. Rohrbach, N. Tandon, and B. Schiele. A dataset for movie description. In *CVPR*, 2015.

[25] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek. Image classification with the fisher vector: Theory and practice. *IJCV*, 105(3):222–245, 2013.

[26] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014.

[27] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[28] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *CVPR*, 2003.

[29] A. Sordoni, Y. Bengio, H. Vahabi, C. Lioma, J. G. Simonsen, and J.-Y. Nie. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *CIKM*, 2015.

[30] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958, 2014.

[31] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014.

[32] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[33] J. Thomason, S. Venugopalan, S. Guadarrama, K. Saenko, and R. Mooney. Integrating language and vision to generate natural language descriptions of videos in the wild. In *COLING*, 2014.

[34] A. Torabi, C. Pal, H. Larochelle, and A. Courville. Using descriptive video services to create a large data source for video annotation research. *arXiv preprint arXiv:1503.01070*, 2015.

[35] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri. C3D: generic features for video analysis. In *ICCV*, 2015.

[36] R. Vedantam, C. L. Zitnick, and D. Parikh. CIDEr: Consensus-based image description evaluation. In *CVPR*, 2015.

[37] S. Venugopalan, M. Rohrbach, J. Donahue, R. J. Mooney, T. Darrell, and K. Saenko. Sequence to sequence - video to text. *ICCV*, 2015.

[38] S. Venugopalan, H. Xu, J. Donahue, M. Rohrbach, R. Mooney, and K. Saenko. Translating videos to natural language using deep recurrent neural networks. In *NAACL-HLT*, 2015.

[39] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Action recognition by dense trajectories. In *CVPR*, 2011.

[40] H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV*, 2013.

[41] Z. Xu, Y. Yang, and A. G. Hauptmann. A discriminative CNN video representation for event detection. In *CVPR*, 2015.

[42] L. Yao, A. Torabi, K. Cho, N. Ballas, C. Pal, H. Larochelle, and A. Courville. Describing videos by exploiting temporal structure. In *ICCV*, 2015.

[43] H. Yu, J. Wang, Z. Huang, Y. Yang, and W. Xu. Video paragraph captioning using hierarchical recurrent neural networks. *CVPR*, 2016.

[44] W. Zaremba and I. Sutskever. Learning to execute. In *ICLR*, 2015.

[45] W. Zaremba, I. Sutskever, and O. Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.