Video Paragraph Captioning Using Hierarchical Recurrent Neural Networks

Haonan Yu[1*] Jiang Wang[3] Zhiheng Huang[2*] Yi Yang[3] Wei Xu[3]
[1]Purdue University [2]Facebook
haonanu@gmail.com zhiheng@fb.com
[3]Baidu Research - Institute of Deep Learning
{wangjiang03,yangyi05,wei.xu}@baidu.com

Abstract

We present an approach that exploits hierarchical Recurrent Neural Networks (RNNs) to tackle the video captioning problem, i.e., generating one or multiple sentences to describe a realistic video. Our hierarchical framework contains a sentence generator and a paragraph generator. The sentence generator produces one simple short sentence that describes a specific short video interval. It exploits both temporal- and spatial-attention mechanisms to selectively focus on visual elements during generation. The paragraph generator captures the inter-sentence dependency by taking as input the sentential embedding produced by the sentence generator, combining it with the paragraph history, and outputting the new initial state for the sentence generator. We evaluate our approach on two large-scale benchmark datasets: YouTubeClips and TACoS-MultiLevel. The experiments demonstrate that our approach significantly outperforms the current state-of-the-art methods with BLEU@4 scores 0.499 and 0.305 respectively.

1. Introduction

In this paper, we consider the problem of video captioning, *i.e*. generating one or multiple sentences to describe the content of a video. The given video could be as general as those uploaded to YouTube, or it could be as specific as cooking videos with fine-grained activities. This ability to generate linguistic descriptions for unconstrained video is important because not only it is a critical step towards machine intelligence, but also it has many applications in daily scenarios such as video retrieval, automatic video subtitling, blind navigation, *etc*. Figure 1 shows some example sentences generated by our approach.

The video captioning problem has been studied for over one decade ever since the first rule-based system on describing human activities with natural language [23]. In a very limited setting, Kojima *et al.* designed some simple heuristics for identifying video objects and a set of rules for producing verbs and prepositions. A sentence is then generated by filling predefined templates with the recognized parts of speech. Following their work, several succeeding approaches [26, 20, 21, 15, 3] applied similar rule-based systems to datasets with larger numbers of objects and events, in different tasks and scenarios. With *ad hoc* rules, they manually establish the correspondence between linguistic terms and visual elements, and analyze the relations among the visual elements to generate sentences. Among them, the most complex rule-based system [3] supports a vocabulary of 118 lexical entries (including 48 verbs and 24 nouns).

To eliminate the tedious effort of rule engineering when the problem scales, some recent methods train statistical models for lexical entries, either in a fully [10, 14, 24, 42] or weakly [37, 36, 57, 55] supervised fashion. The statistical models of different parts of speech usually have different mathematical representations and training strategies (*e.g*., [14, 24]). With most of the manual effort gone, the training process exposes these methods to even larger datasets (*e.g*., YouTubeClips [6] and TACoS-MultiLevel [36]) which contain thousands of lexical entries and dozens of hours of videos. As a result, the video captioning task becomes much more challenging, and the generation performance of these methods is usually low on these large-scale datasets.

Since then, inspiring results have been achieved by a recent line of work [11, 48, 47, 32, 54, 56] which benefits from the rapid development of deep neural networks, especially Recurrent Neural Network (RNN). Applying RNN to translating visual sequence to natural language is largely inspired by the recent advances in Neural Machine Translation (NMT) [1, 43] in the natural language processing community. The idea is to treat the image sequence of a video as the "source text" and the corresponding caption as the target text. Given a sequence of deep convolutional features (*e.g*., VggNet [40] and C3D [45]) extracted from video frames, a compact representation of the video is obtained by: average pooling [48, 32], weighted average pooling with an attention model [56], or taking the last output from an RNN en-

*This work was done while the authors were at Baidu.

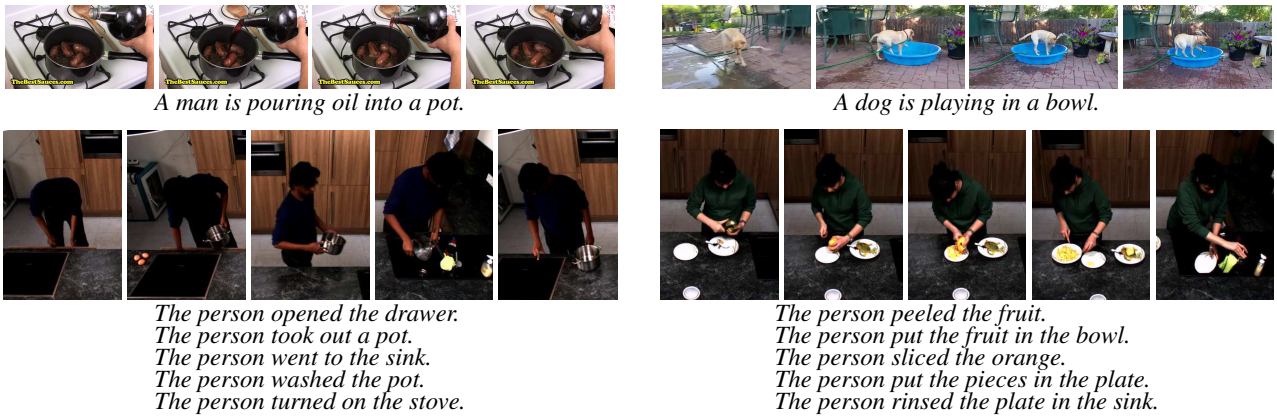


Figure 1. Some example sentences generated by our approach. The first row shows examples trained on YouTubeClips, where only one sentence is generated for each video. The second row shows examples trained on TACoS-MultiLevel, where paragraphs are generated.

coder which summarizes the feature sequence [11, 47, 54]. Then an RNN decoder accepts this compact representation and outputs a sentence of a variable length.

While promising results were achieved by these RNN methods, they only focus on generating a single sentence for a short video clip. So far the problem of generating multiple sentences or a paragraph for a long video has not been attempted by deep learning approaches. Some graphical-model methods, such as Rohrbach *et al.* [36], are able to generate multiple sentences, but their results are still far from perfect. The motivation of generating a paragraph is that most videos depict far more than just one event. Using only one short sentence to describe a semantically rich video usually yields uninformative and even boring results. For example, instead of saying *the person sliced the potatoes, cut the onions into pieces, and put the onions and potatoes into the pot*, a method that is only able to produce one short sentence would probably say *the person is cooking*.

Inspired by the recent progress of document modeling [27, 28] in natural language processing, we propose a hierarchical-RNN framework for describing a long video with a paragraph consisting of multiple sentences. The idea behind our hierarchical framework is that we want to exploit the temporal dependency among sentences in a paragraph, so that when producing the paragraph, the sentences are not generated independently. Instead, the generation of one sentence might be affected by the semantic context provided by the previous sentences. For example, in a video of cooking dishes, a sentence *the person peeled the potatoes* is more likely to occur, than the sentence *the person turned on the stove*, after the sentence *the person took out some potatoes from the fridge*. Towards this end, our hierarchical framework consists of two generators, *i.e.* a sentence generator and a paragraph generator, both of which use recurrent layers for language modeling. At the low level, the sentence generator produces single short sentences that describe specific time intervals and video regions. We exploit both temporal- and spatial-attention mechanisms to selectively focus on visual elements when generating a sentence. The embedding of the generated sentence is encoded by the output of the recurrent layer. At the high level, the paragraph generator takes the sentential embedding as input, and uses another recurrent layer to output the paragraph state, which is then used as the new initial state of the sentence generator (see Section 3). Figure 2 illustrates our overall framework. We evaluate our approach on two public datasets: YouTubeClips [6] and TACoS-MultiLevel [36]. We show that our approach significantly outperforms other state-of-the-art methods. To our knowledge, this is the first application of hierarchical RNN to video captioning task.

2. Related Work

Neural Machine Translation. The methods for NMT [18, 9, 1, 43, 27, 28] in computational linguistics generally follow the encoder-decoder paradigm. An encoder maps the source sentence to a fixed-length feature vector in the embedding space. A decoder then conditions on this vector to generate a translated sentence in the target language. On top of this paradigm, several improvements were proposed. Bahdanau *et al.* [1] proposed a soft attention model to do alignment during translation, so that their approach is able to focus on different parts of the source sentence when generating different translated words. Li *et al.* [27] and Lin *et al.* [28] employed hierarchical RNN to model the hierarchy of a document. Our approach is much similar to a neural machine translator with a simplified attention model and a hierarchical architecture.

Image captioning with RNNs. The first attempt of visual-to-text translation using RNNs was seen in the work of image captioning [29, 22, 19, 50, 8], which can be treated as a special case of video captioning when each video has a single frame and no temporal structure. As a result, image

captioning only requires computing object appearance features, but not action/motion features. The amount of data handled by an image captioning method is much (dozens of times) less than that handled by a video captioning method. The overall structure of an image captioner (instance-to-sequence) is also usually simpler than that of a video captioner (sequence-to-sequence). Some other methods, such as Park and Kim [34], addressed the problem of retrieving sentences from training database to describe a sequence of images. They proposed a local coherence model for fluent sentence transitions, which serves a similar purpose of our paragraph generator.

Video captioning with RNNs. The very early video captioning method [48] based on RNNs extends the image captioning methods by simply average pooling the video frames. Then the problem becomes exactly the same as image captioning. However, this strategy works only for short video clips where there is only one major event, usually appearing in one video shot from the beginning to the end. To avoid this issue, more sophisticated ways of encoding video features were proposed in later work, using either a recurrent encoder [11, 47, 54] or an attention model [56]. Our sentence generator is closely related to Yao *et al.* [56], in that we also use attention mechanism to selectively focus on video features. One difference between our framework and theirs is that we additionally exploit spatial attention. The other difference is that after weighing video features with attention weights, we do not condition the hidden state of our recurrent layer on the weighted features (Section 3.2).

3. Hierarchical RNN for Video Captioning

Our approach stacks a paragraph generator on top of a sentence generator. The sentence generator is built upon 1) a Recurrent Neural Network (RNN) for language modeling, 2) a multimodal layer [29] for integrating information from different sources, and 3) an attention model [56, 1] for selectively focusing on the input video features. The paragraph generator is simply another RNN which models the inter-sentence dependency. It receives the compact sentential representation encoded by the sentence generator, combines it with the paragraph history, and outputs a new initial state for the sentence generator. The RNNs exploited by the two generators incorporate the *Gated Recurrent Unit* (GRU) [9] which is a simplification of the Long Short-Term Memory (LSTM) architecture [16]. In the following, we first briefly review the RNN with the GRU (or the *gated RNN*), and then describe our framework in details.

3.1. Gated Recurrent Unit

A simple RNN [12] can be constructed by adding feedback connections to a feedforward network that consists of three layers: the input layer $\mathbf{x}$, the hidden layer $\mathbf{h}$, and the output layer $\mathbf{y}$. The network is updated by both the input and the previous recurrent hidden state as follows:

$$\begin{aligned} \mathbf{h}^t &= \phi\left(\mathbf{W}_h\mathbf{x}^t + \mathbf{U}_h\mathbf{h}^{t-1} + \mathbf{b}_h\right) && \text{(hidden state)} \\ \mathbf{y}^t &= \phi\left(\mathbf{U}_y\mathbf{h}^t + \mathbf{b}_y\right) && \text{(output)} \end{aligned}$$

where $\mathbf{W}, \mathbf{U}$ and $\mathbf{b}$ are weight matrices and biases to be learned, and $\phi(\cdot)$ are element-wise activation functions.

While the simple RNN is able to model temporal dependency for a small time gap, it usually fails to capture long-term temporal information. To address this issue, the GRU [9] is designed to adaptively remember and forget the past. Inside the unit, the hidden state is modulated by non-linear gates. Specifically, let $\odot$ denote the element-wise multiplication of two vectors, the GRU computes the hidden state $\mathbf{h}$ as:

$$\begin{aligned} \mathbf{r}^t &= \sigma(\mathbf{W}_r\mathbf{x}^t + \mathbf{U}_r\mathbf{h}^{t-1} + \mathbf{b}_r) && \text{(reset gate)} \\ \mathbf{z}^t &= \sigma(\mathbf{W}_z\mathbf{x}^t + \mathbf{U}_z\mathbf{h}^{t-1} + \mathbf{b}_z) && \text{(update gate)} \\ \widetilde{\mathbf{h}}^t &= \phi\left(\mathbf{W}_h\mathbf{x}^t + \mathbf{U}_h(\mathbf{r}^t \odot \mathbf{h}^{t-1}) + \mathbf{b}_h\right) \\ \mathbf{h}^t &= \mathbf{z}^t \odot \mathbf{h}^{t-1} + (1 - \mathbf{z}^t) \odot \widetilde{\mathbf{h}}^t && \text{(hidden state)} \end{aligned}$$

where $\sigma(\cdot)$ are element-wise Sigmoid functions. The reset gate $\mathbf{r}$ determines whether the hidden state wants to drop any information that will be irrelevant in the future. The update gate $\mathbf{z}$ controls how much information from the previous hidden state will be preserved for the current state. During the training of a gated RNN, the parameters can be estimated by Backpropagation Through Time (BPTT) [53] as in traditional RNN architectures.

3.2. Sentence Generator

The overall structure of our hierarchical RNN is illustrated in Figure 2. The sentence generator operates at every time step when a one-hot input (1-of-N encoding, where N is the vocabulary size) arrives at the embedding layer. The embedding layer converts the one-hot vector to a dense representation in a lower dimensional space by multiplying it with an embedding table ($512 \times N$), of which each row is a word embedding to be learned. The resulting word embedding is then input to our first RNN, *i.e.*, the recurrent layer I. This gated recurrent layer has 512 dimensions and acts similarly to those that are commonly employed by a variety of image/video captioning methods (*e.g.*, [47, 29, 56]), *i.e.*, modeling the syntax of a language. It updates its hidden state every time a new word arrives, and encodes the sentence semantics in a compact form up to the words that have been fed in. We set the activation function ϕ of this recurrent layer to be the Rectified Linear Unit (ReLU) [31], since it performs better than non-linear activation functions such as Sigmoid according to our observation.

As one branch, the output of the recurrent layer I is directed to the attention layers to compute attention weights for the features in the video feature pool. Our attention model is inspired by the recent soft-alignment method that

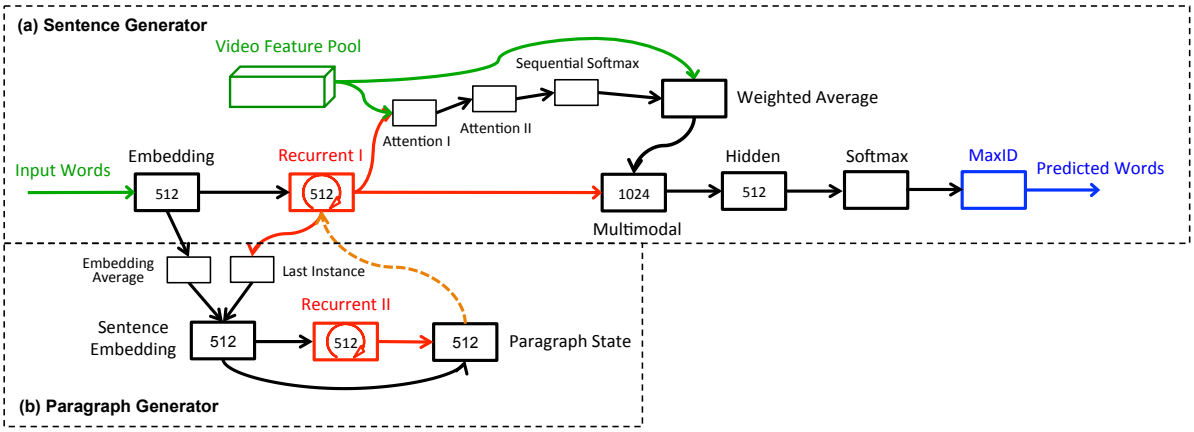


Figure 2. Our hierarchical RNN for video captioning. Green denotes the input to the framework, blue denotes the output, and red denotes the recurrent components. The orange arrow represents the reinitialization of the sentence generator with the current paragraph state. For simplicity, we only draw a single video feature pool in the figure. In fact, both appearance and action features go through a similar attention process before they are fed into the multimodal layer.

has been successfully applied in the context of Neural Machine Translation (NMT) [1], and was later adapted to video captioning by Yao *et al.* [56]. The difference between our model and the one used by Yao *et al.* is that their model only focuses on temporal attention. We additionally include spatial attention by computing features for multiple image patches at different locations on a video frame and pool the features together. This simple improvement is important when objects are small and difficult to be localized on some datasets (*e.g.*, TACoS-MultiLevel [36]). In this case, whole-frame-based video features will fail to capture the object information and multiple object proposals are needed for good performance (see Section 5 for details). Let the features in the pool be denoted as $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{KM}\}$, where M is the video length and K is the number of patches on each frame. We want to compute a set of weights $\{\beta_1^t, \beta_2^t, \ldots, \beta_{KM}^t\}$ for these features at each time step t such that $\sum_{m=1}^{KM} \beta_m^t = 1$. To do so, we first compute an attention score q_m^t for each frame m, conditioning on the previous hidden state $\mathbf{h}^{t-1}$:

$$q_m^t = \mathbf{w}^\top \phi(\mathbf{W}_q \mathbf{v}_m + \mathbf{U}_q \mathbf{h}^{t-1} + \mathbf{b}_q)$$

where $\mathbf{w}$, $\mathbf{W}_q$, $\mathbf{U}_q$, and $\mathbf{b}_q$ are the parameters shared by all the features at all the time steps, and ϕ is set to the element-wise Scaled Hyperbolic Tangent (stanh) function [25]: $1.7159 \cdot \tanh(\frac{2x}{3})$. The above computation is performed by the attention layers I and II in Figure 2(a), where the attention layer I projects the feature $\mathbf{v}$ and the hidden state $\mathbf{h}$ into a lower dimensional space whose dimension can range from 32 to 256. The attention layer II then further compresses the activation of the projected vector into a scalar, one for each feature. After this, we set up a sequential softmax layer to get the attention weights:

$$\beta_m^t = \exp\left(q_m^t\right) \Big/ \sum_{m'=1}^{KM} \exp\left(q_{m'}^t\right)$$

Finally, a single feature vector is obtained by weighted averaging: $\mathbf{u}^t = \sum_{m=1}^{KM} \beta_m^t \mathbf{v}_m$. The above process is a sophisticated version of the temporal mean pooling. It allows the sentence generator to selectively focus on a subset of the features during generation. Note that while only one feature channel is shown in Figure 2(a), our sentence generator in fact pumps features of several channels through the same attention process. Each feature channel has a different set of weights and biases to be learned. In our experiments, we employ two feature channels, one for object appearance and the other for action/motion. (Section 5).

After the attention process, the weighted sums of the video features are fed into the multimodal layer which has 1024 dimensions. The multimodal layer also receives the output of the recurrent layer I, thus connecting the vision component with the language model. Suppose we have two video feature channels, of which the weighted features output by the attention model are $\mathbf{u}_o^t$ and $\mathbf{u}_a^t$ respectively. The multimodal layer maps the two features, together with the hidden state $\mathbf{h}^t$ of the recurrent layer I, into a 1024 dimensional feature space and add them up:

$$\mathbf{m}^t = \phi(\mathbf{W}_{m,o}\mathbf{u}_o^t + \mathbf{W}_{m,a}\mathbf{u}_a^t + \mathbf{U}_m \mathbf{h}^t + \mathbf{b}_m)$$

where ϕ is set to the element-wise stanh function. To reduce overfitting, we add dropout [41] with a drop rate of 0.5 to this layer.

The multimodal layer is followed by a hidden layer and a softmax layer (see Figure 2(a)), both with the element-wise stanh function as their activation functions. The hidden layer has exactly the same dimension 512 with the

word embedding layer, and the softmax layer has a dimension that is equal to the size of the vocabulary which is dataset-dependent. Inspired by the transposed weight sharing scheme recently proposed by Mao *et al.* [30], we set the projection matrix from the hidden layer to the softmax layer as the transpose of the word embedding table. It has been shown that this strategy allows the use of a word embedding layer with a much larger dimension due to the parameter sharing, and helps regularize the word embedding table because of the matrix transpose. As the final step of the sentence generator, the maxid layer picks the index that points to the maximal value in the output of the softmax layer. The index is then treated as the predicted word id. Note that during test, the predicted word will be fed back to the sentence generator again as the next input word. While in the training, the next input word is always provided by the annotated sentence.

3.3. Paragraph Generator

The sentence generator above only handles one single sentence at a time. For the first sentence in the paragraph, the initial state of the recurrent layer I is set to all zeros, *i.e.*, $\mathbf{h}^0 = \mathbf{0}$. However, any sentence after that will have its initial state conditioned on the semantic context of all its preceding sentences. This semantic context is encoded by our paragraph generator.

During the generation of a sentence, an embedding average layer (see Figure 2(b)) accumulates all the word embeddings of the sentence and takes the average to get a compact embedding vector. The average strategy is inspired by the QA embedding [5] in which questions and answers are both represented as a combination of the embeddings of their individual words and/or symbols. We also take the last state of the recurrent layer I as a compact representation for the sentence, following the idea behind the Encoder-Decoder framework [9] in NMT. After that, the averaged embedding and the last recurrent state are concatenated together, and fully connected to the sentence embedding layer (512 dimensions) with stanh as the activation function. We treat the output of the sentence embedding layer as the final sentence representation.

The sentence embedding layer is linked to our second gated RNN (see Figure 2(b)). The recurrent layer II operates whenever a full sentence goes through the sentence generator and the sentence embedding is produced by the sentence embedding layer. Thus the two recurrent layers are asynchronous: while the recurrent layer I keeps updating its hidden state at every time step, the recurrent layer II only updates its hidden state when a full sentence has been processed. The recurrent layer II encodes the paragraph semantics in a compact form up to the sentences that have been fed in. Finally, we set up a paragraph state layer to combine the hidden state of the recurrent layer II and the sentence embedding. This paragraph state is used as the initial hidden state when the recurrent layer I is reinitialized for the next sentence. It essentially provides the sentence generator with the paragraph history so that the next sentence is produced in the context.

4. Training and Generation

We train all the components in our hierarchical framework together from scratch with randomly initialized parameters. We treat the activation value indexed by a training word w_t^n in the softmax layer of our sentence generator as the likelihood of generating that word:

$$P\left(w_t^n | s_{1:n-1}, w_{1:t-1}^n, \mathbf{V}\right)$$

given 1) all the preceding sentences $s_{1:n-1}$ in the paragraph, 2) all the previous words $w_{1:t-1}^n$ in the same sentence n, and 3) the corresponding video $\mathbf{V}$. The cost of generating that training word is then defined as the negative logarithm of the likelihood. We further define the cost of generating the whole paragraph $s_{1:N}$ (N is the number of sentences in the paragraph) as:

$$\begin{aligned}&\mathcal{PPL}(s_{1:N}|\mathbf{V})\\ &= -\sum_{n=1}^{N}\sum_{t=1}^{T_n}\log P\left(w_t^n | s_{1:n-1}, w_{1:t-1}^n, \mathbf{V}\right) \Big/ \sum_{n=1}^{N} T_n\end{aligned}$$

where T_n is the number of words in the sentence n. The above cost is in fact the *perplexity* of the paragraph given the video. Finally, the cost function over the entire training set is defined as:

$$\mathcal{PPL} = \sum_{y=1}^{Y}\left(\mathcal{PPL}(s_{1:N_y}^y|\mathbf{V}_y)\cdot\sum_{n=1}^{N_y} T_n^y\right) \Big/ \sum_{y=1}^{Y}\sum_{n=1}^{N_y} T_n^y \tag{1}$$

where Y is the total number of paragraphs in the training set. To reduce overfitting, L2 and L1 regularization terms are added to the above cost function. We use Backpropagation Through Time (BPTT) [53] to compute the gradients of the parameters and Stochastic Gradient Descent (SGD) to find the optimum. For better convergence, we divide the gradient by a running average of its recent magnitude according to the RMSPROP algorithm [44]. We set a small learning rate 10^{-4} to avoid the gradient explosion problem that is common in the training process of RNNs.

After the parameters are learned, we perform the generation with Beam Search. Suppose that we use a beam width of L. The beam search process starts with the BOS (begin-of-sentence) symbol w_{BOS} (*i.e.*, w_0) which is treated as a 1-word sequence with zero cost at $t = 0$. Assume that at any time step t, there are at most L t-word sequences that were previously selected with the lowest sequence costs

(a sequence cost is the sum of the word costs in that sequence). For each of the t-word sequences, given its last word as input, the sentence generator calculates the cost of the next word $-\log P(w_t|w_{1:t-1}, \mathbf{V})$ and the sequence cost if the word is appended to the sequence. Then from all the $t+1$-word sequences expanded from the existing t-word sequences, we pick the top L with the lowest sequence costs.

Of the new $t+1$-word sequences, any one that is a complete sentence (*i.e.*, the last word w_{t+1} is the EOS (end-of-sentence) symbol w_{EOS}) will be removed from the search tree. It will be put into our sentence pool if 1) there are less than J ($J \leq L$) sentences in the pool or, 2) its sequence cost is lower than one of the J sentences in the pool. In the second case, the sentence with the highest cost will be removed from the pool, replaced by the new added sentence. Also of the new $t+1$-word sequences, any one that has a higher sequence cost than all of the J sentences in the pool will be removed from the search tree. The reason is that expanding a word sequence monotonically increases its cost. The beam search process stops when there is no word sequence to be expanded in the next time step. In the end, J candidate sentences will be generated for post-processing and evaluation.

After this, the generation process goes on by picking the sentence with the lowest cost from the J candidate sentences. This sentence is fed into our paragraph generator which reinitializes the sentence generator. The sentence generator then accepts a new BOS and again produces J candidate sentences. This whole process stops when the sentence received by the paragraph generator is the EOP (end-of-paragraph) which consists of only the BOS and the EOS. Finally, we will have a paragraph that is a sequence of lists, each list with J sentences. In our experiments, we set $L = J = 5$. Excluding the calculation of visual features, the average computational time for the sentence generator to produce top 5 candidate sentences with a beam width of 5 is 0.15 seconds, on a single thread with CPU Intel(R) Core(TM) i7-5960X @ 3.00GHz.

5. Experiments

We evaluate our approach on two benchmark datasets: YouTubeClips [6] and TACoS-MultiLevel [36].

YouTubeClips This dataset consists of $1,967$ short video clips (9 seconds on average) downloaded from YouTube. The video clips are open-domain, containing different people, animals, actions, scenarios, landscapes, *etc*. Each video clip is annotated with multiple parallel sentences by different turkers. There are $80,839$ sentences in total, with about 41 annotated sentences per clip. Each sentence on average contains about 8 words. The words contained in all the sentences constitute a vocabulary of $12,766$ unique lexical entries. We adopt the train and test splits provided by Guadarrama *et al.* [14], where $1,297$ and 670 videos are used for training and testing respectively. It should be noted that while multiple sentences are annotated for each video clip, they are *parallel* and *independent* in the temporal extent, *i.e.*, the sentences describe exactly the same video interval, from the beginning to the end of the video. As a result, we use this dataset as a special test case for our approach, when the paragraph length $N = 1$.

TACoS-MultiLevel This dataset consists of 185 long videos (6 minutes on average) filmed in an indoor environment. The videos are closed-domain, containing different actors, fine-grained activities, and small interacting objects in daily cooking scenarios. Each video is annotated by multiple turkers. A turker annotates a sequence of temporal intervals across the video, pairing every interval with a single short sentence. There are $16,145$ distinct intervals and $52,478$ sentences in total, with about 87 intervals and 284 sentences per video. The sentences were originally preprocessed so that they all have the past tense, and different gender specific identifiers were substituted with "*the person*". Each sentence on average contains about 8 words. The words contained in all the sentences constitute a vocabulary of $2,864$ unique lexical entries. We adopt the train and test splits used by Rohrbach *et al.* [36], where 143 and 42 videos are used for training and testing respectively. Note that the cooking activities in this dataset have strong temporal dependencies. Such dependency in a video is implied by the sequence of intervals annotated by the same turker on that video. Following Donahue *et al.* [11] and Rohrbach *et al.* [36], we employ the interval information to align our sentences in the paragraph during both training and generation. This dataset is used as a general test case for our approach, when the paragraph length $N > 1$.

To model video object appearance, we use the pre-trained VggNet [40] (on the ImageNet dataset [38]) for both datasets. Since the objects in YouTubeClips are usually prominent, we only extract one VggNet feature for each entire frame. This results in only temporal attention in our sentence generator (*i.e.*, $K = 1$ in Section 3.2). For TACoS-MultiLevel, the interacting objects are usually quite small and difficult to be localized. To solve this problem, both Donahue *et al.* [11] and Rohrbach *et al.* [36] designed a specialized hand detector. Once the hand regions are detected, they extract features in the neighborhood to represent the interacting objects. Instead of trying to accurately locate hands which requires a lot of engineering effort as in their case, we rely on a simple routine to obtain multiple object proposals. We first use Optical Flow [13] to roughly detect a bounding box for the actor in each frame. We then extract K image patches of size 220×220 along the lower part of the box border, where every two neighboring patches have an overlap of half their size. Our simple observation is that these patches together have a high recall of containing the interacting objects while the actor is cooking. Finally, we

	B@1	B@2	B@3	B@4	M	C
LSTM-YT [48]	-	-	-	0.333	0.291	-
S2VT [47]	-	-	-	-	0.298	-
MM-VDN [54]	-	-	-	0.376	0.290	-
TA [56]	0.800	0.647	0.526	0.419	0.296	0.517
LSTM-E [32]	0.788	0.660	0.554	0.453	0.310	-
h-RNN-Vgg	0.773	0.645	0.546	0.443	0.311	0.621
h-RNN-C3D	0.797	0.679	0.579	0.474	0.303	0.536
h-RNN (Ours)	**0.815**	**0.704**	**0.604**	**0.499**	**0.326**	**0.658**

Table 1. Results on YouTubeClips, where B, M, and C are short for BLEU, METEOR, and CIDEr respectively.

	B@1	B@2	B@3	B@4	M	C
CRF-T [37]	0.564	0.447	0.332	0.253	0.260	1.248
CRF-M [36]	0.584	0.467	0.352	0.273	0.272	1.347
LRCN [11]	0.593	0.482	0.370	0.292	0.282	1.534
h-RNN-Vgg	0.561	0.445	0.329	0.256	0.260	1.267
h-RNN-DT	0.557	0.451	0.346	0.274	0.261	1.400
RNN-sent	0.568	0.469	0.367	0.295	0.278	1.580
RNN-cat	0.605	0.489	0.376	0.297	0.284	1.555
h-RNN (Ours)	**0.608**	**0.496**	**0.385**	**0.305**	**0.287**	**1.602**

Table 2. Results on TACoS-MultiLevel, where B, M, and C are short for BLEU, METEOR, and CIDEr respectively.

compute the VggNet feature for each patch and pool all the patch features. When $K > 1$, the above routine leads to both temporal and spatial attention in our sentence generator. In practice, we find that a small value of K (*e.g.*, $3 \sim 5$) is enough to yield good performance.

To model video motion and activities, we use the pre-trained C3D [45] (on the Sports-1M dataset [19]) for YouTubeClips. The C3D net reads in a video and outputs a fixed-length feature vector every 16 frames. Thus when applying the attention model to the C3D feature pool, we set $K = 1$ and divide M by 16 (Section 3.2). For the TACoS-MultiLevel dataset, since the cooking activities are fine-grained, the same model trained on sports videos does not work well. Alternatively we compute the Dense Trajectories [51] for each video interval and encode them with the Fisher vector [17]. For the attention model, we set $K = 1$ and $M = 1$.

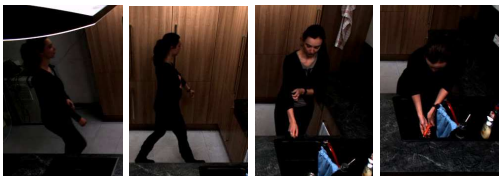
We employ three different evaluation metrics: BLEU [33], METEOR [2], and CIDEr [46]. Because the YouTubeClips dataset was tested on by most existing video-captioning methods, the prior results of all the three metrics have been reported. The TACoS-MultiLevel dataset is relatively new and only the BLEU scores were reported in the previous work. We compute the other metrics for the comparison methods based on the generated sentences that come with the dataset. Generally, the higher the metric scores are, the better the generated sentence correlates with human judgment. We use the evaluation script provided by Chen *et al.* [7] to compute scores on both datasets.

5.1. Results

We compare our approach (h-RNN) on YouTubeClips with six state-of-the-art methods: LSTM-YT [48], S2VT [47], MM-VDN [54], TA [56], and LSTM-E [32]. Note that in this experiment a single sentence is generated for each video. Thus only our sentence generator is evaluated in comparison to others. To evaluate the importance of our video features, we also report the results of two baseline methods: h-RNN-Vgg and h-RNN-C3D. The former uses only the object appearance feature and the latter uses only the motion feature, with other components of our framework unchanged. The evaluation results are shown in Table 1. We can see that our approach performs much better than the comparison methods, in all the three metrics. The improvements on the most recent state-of-the-art method (*i.e.*, LSTM-E [32]) are $\frac{0.499-0.453}{0.453} = 10.15\%$ in the BLEU@4 score, and $\frac{0.326-0.310}{0.310} = 5.16\%$ in the METEOR score. Since LSTM-E also exploits VggNet and C3D features, this demonstrates that our sentence generator framework is superior to their joint embedding framework. Moreover, although TA [56] also employs temporal attention, our approach produces much better results due to the fact that the hidden state of our RNN is not conditioned on the video features. Instead, the video features are directly input to our multimodal layer. Our approach also outperforms the two baseline methods by large margins, indicating that both video features are indeed crucial in the video captioning task.

We compare our approach on TACoS-MultiLevel with three state-of-the-art methods: CRF-T [37], CRF-M [36], and LRCN [11]. Like above, we have two baseline methods h-RNN-Vgg and h-RNN-DT which use only the appearance and motion features respectively. We also add another two baseline methods RNN-sent and RNN-cat that have no hierarchy (*i.e.*, with only the sentence generator, but not the paragraph generator). RNN-sent is trained and tested on individual video clips that are segmented from the original 185 long videos according to the annotated intervals. The initial state of the sentence generator is set to zero for each sentence. As a result, sentences are trained and generated independently. RNN-cat initializes the sentence generator with zero only for the first sentence in a paragraph. Then the sentence generator maintains its state for the following sentences until the end of the paragraph. This concatenation strategy for training a paragraph has been exploited in a recent neural conversational model [49]. We use RNN-send and RNN-cat to evaluate the importance of our hierarchical structure.

The results on TACoS-MultiLevel are shown in Table 2. Our approach outperforms the state-of-the-art methods, including the very recently proposed one (*i.e.*, LRCN) with an improvement of $\frac{0.305-0.292}{0.292} = 4.45\%$ in the BLEU@4



RNN-sent:	*The person entered the kitchen.* *The person went to the refrigerator.* *The person placed the cucumber on the cutting board.* *The person rinsed the cutting board.*
h-RNN:	*The person walked into the kitchen.* *The person went to the refrigerator.* *The person walked over to the sink.* *The person rinsed the carrot in the sink.*

RNN-sent:	*The person took out a cutting board from the drawer.* *The person got a knife and a cutting board from the drawer.* *The person cut the ends off the cutting board.*
h-RNN:	*The person took out a cutting board.* *The person got a knife from the drawer.* *The person cut the cucumber on the cutting board.*

Figure 3. Examples of generated paragraphs. Red indicates incorrect sentences produced by RNN-sent and green shows the ones generated by our h-RNN in the corresponding time intervals. In the first example, our hierarchical model successfully captures the high likelihood of the event *walk to the sink* after the event *open the refrigerator*. In the second example, RNN-sent generates the event *take the cutting board* twice due to the fact that the sentences in the paragraph are produced independently. In contrast, our hierarchical model avoids this mistake.

score. Given that our strategy of extracting object regions is relatively simple compared to the sophisticated hand detector [11, 36], we expect to have even better performance if our object localization is improved. Our method is also superior to all the baseline methods. Although RNN-cat models temporal dependency among sentences by sentence-level concatenation, it performs worse than our hierarchical architecture. Again, it shows that both the video features and the hierarchical structure are crucial in our task. Figure 3 illustrates some example paragraphs generated by our approach on TACoS-MultiLevel.

To further demonstrate that our method h-RNN generates better sentences than RNN-cat in general, we perform human evaluation to compare these two methods on TACoS-MultiLevel. Specifically, we discard $1,166$ test video intervals, each of which has exactly the same sentence generated by RNN-cat and h-RNN. This results in a total number of $4,314 - 1,166 = 3,148$ video intervals for human evaluation. We then put the video intervals and the generated sentences on Amazon Mechanical Turk (AMT). Each video interval is paired with one sentence generated by RNN-cat and the other by h-RNN, side by side. For each video interval, we ask one turker to select the sentence that better describes the video content. The turker also has a third choice if he believes that both sentences are equally good or bad. In the end, we obtained 773 selections for h-RNN and 472 selections for RNN-cat, with a gap of 301 selections. Thus h-RNN has at least $\frac{301}{472+3069} = 8.50\%$ improvement over RNN-cat.

h-RNN	RNN-cat	Equally good or bad	Total
773	472	3069	4314

5.2. Discussions and Limitations

Although our approach is able to produce paragraphs for video and has achieved encouraging results, it is subject to several limitations. First, our object detection routine has difficulty handling very small objects. Most of our failure cases on TACoS-MultiLevel produce incorrect object names in the sentences, *e.g.*, confusing small objects that have similar shapes or appearances (*cucumber vs. carrot*, *mango vs. orange*, *kiwi vs. avocado*, *etc.*). See Figure 1 for a concrete example: *sliced the orange* should really be *sliced the mango*. Accurately detecting small objects (sometimes with occlusion) in complex video scenarios still remains an open problem. Second, the sentential information flows unidirectionally through the paragraph recurrent layer, from the beginning of the paragraph to the end, but not also in the reverse way. Misleading information will be potentially passed down when the first several sentences in a paragraph are generated incorrectly. Using bidirectional RNN [39, 52] for sentence generation is still an open problem. Lastly, our approach suffers from a known problem as in most other image/video captioning methods, namely, there is discrepancy between the objective function used by training and the one used by generation. The training process predicts the next word given the previous words from groundtruth, while the generation process conditions the prediction on the ones previously generated by itself. This problem is amplified in our hierarchical framework where the paragraph generator conditions on groundtruth sentences during training but on generated ones during generation. A potential cure for this would be adding Scheduled Sampling [4] to the training process, where one randomly selects between the true previous words and the words generated by the model. Another solution might be to directly optimize the metric (*e.g.*, BLEU) used at test time [35].

6. Conclusion

We have proposed a hierarchical-RNN framework for video paragraph captioning. The framework models inter-sentence dependency to generate a sequence of sentences given video data. The experiments show that our approach is able to generate a paragraph for a long video and achieves the state-of-the-art results on two large-scale datasets.

Acknowledgments

The primary author would like to thank Baidu Research for providing the summer internship.

References

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

[2] S. Banerjee and A. Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, June 2005.

[3] A. Barbu, A. Bridge, Z. Burchill, D. Coroian, S. Dickinson, S. Fidler, A. Michaux, S. Mussman, N. Siddharth, D. Salvi, L. Schmidt, J. Shangguan, J. M. Siskind, J. Waggoner, S. Wang, J. Wei, Y. Yin, and Z. Zhang. Video in sentences out. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 102–112, 2012.

[4] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179, 2015.

[5] A. Bordes, S. Chopra, and J. Weston. Question answering with subgraph embeddings. In *Conference on Empirical Methods in Natural Language Processing*, pages 615–620, 2014.

[6] D. L. Chen and W. B. Dolan. Collecting highly parallel data for paraphrase evaluation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL-2011)*, Portland, OR, June 2011.

[7] X. Chen, H. Fang, T. Lin, R. Vedantam, S. Gupta, P. Dollár, and C. L. Zitnick. Microsoft COCO captions: Data collection and evaluation server. *CoRR*, abs/1504.00325, 2015.

[8] X. Chen and C. L. Zitnick. Learning a recurrent visual representation for image caption generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[9] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing*, 2014.

[10] P. Das, C. Xu, R. F. Doell, and J. J. Corso. A thousand frames in just a few words: Lingual description of videos through latent topics and sparse object stitching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2634–2641, 2013.

[11] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[12] J. L. Elman. Finding structure in time. *COGNITIVE SCIENCE*, 14(2):179–211, 1990.

[13] G. Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Proceedings of the 13th Scandinavian Conference on Image Analysis*, pages 363–370, 2003.

[14] S. Guadarrama, N. Krishnamoorthy, G. Malkarnenkar, S. Venugopalan, T. D. R. Mooney, and K. Saenko. Youtube2text: Recognizing and describing arbitrary activities using semantic hierarchies and zero-shot recognition. In *ICCV'13 Int. Conf. on Computer Vision 2013*, December 2013.

[15] P. Hanckmann, K. Schutte, and G. J. Burghouts. Automated textual descriptions for a wide range of video events with 48 human actions. In *Proceedings of the European Conference on Computer Vision Workshops and Demonstrations*, pages 372–380, 2012.

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

[17] H. Jegou, F. Perronnin, M. Douze, J. Sánchez, P. Perez, and C. Schmid. Aggregating local image descriptors into compact codes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(9):1704–1716, Sept. 2012.

[18] N. Kalchbrenner and P. Blunsom. Recurrent continuous translation models. In *Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, 2013.

[19] 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.

[20] M. U. G. Khan, L. Zhang, and Y. Gotoh. Human focused video description. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 1480–1487, 2011.

[21] M. U. G. Khan, L. Zhang, and Y. Gotoh. Towards coherent natural language description of video streams. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 664–671, 2011.

[22] R. Kiros, R. Salakhutdinov, and R. S. Zemel. Unifying visual-semantic embeddings with multimodal neural language models. In *NIPS Deep Learning Workshop*, 2014.

[23] A. Kojima, T. Tamura, and K. Fukunaga. Natural language description of human activities from video images based on concept hierarchy of actions. *International Journal of Computer Vision*, 50(2):171–184, 2002.

[24] N. Krishnamoorthy, G. Malkarnenkar, R. J. Mooney, K. Saenko, and S. Guadarrama. Generating natural-language video descriptions using text-mined knowledge. In *AAAI Conference on Artificial Intelligence*, pages 541–547, 2013.

[25] Y. LeCun, L. Bottou, G. Orr, and K. Müller. Efficient backprop. In *Neural Networks: Tricks of the Trade*, page 546. 1998.

[26] M. W. Lee, A. Hakeem, N. Haering, and S.-C. Zhu. SAVE: A framework for semantic annotation of visual events. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–8, 2008.

[27] J. Li, M. Luong, and D. Jurafsky. A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 1106–1115, 2015.

[28] R. Lin, S. Liu, M. Yang, M. Li, M. Zhou, and S. Li. Hierarchical recurrent neural network for document modeling. pages 899–907. Conference on Empirical Methods in Natural Language Processing, Sept. 2015.

[29] J. Mao, W. Xu, Y. Yang, J. Wang, Z. Huang, and A. Yuille. Deep captioning with multimodal recurrent neural networks (m-rnn). *ICLR*, 2015.

[30] J. Mao, W. Xu, Y. Yang, J. Wang, Z. Huang, and A. L. Yuille. Learning like a child: Fast novel visual concept learning from sentence descriptions of images. 2015.

[31] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, pages 807–814, 2010.

[32] Y. Pan, T. Mei, T. Yao, H. Li, and Y. Rui. Jointly modeling embedding and translation to bridge video and language. *CoRR*, abs/1505.01861, 2015.

[33] K. Papineni, S. Roukos, T. Ward, and W. jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318, 2002.

[34] C. C. Park and G. Kim. Expressing an image stream with a sequence of natural sentences. In *Advances in Neural Information Processing Systems*, pages 73–81, 2015.

[35] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba. Sequence level training with recurrent neural networks. *CoRR*, abs/1511.06732, 2015.

[36] A. Rohrbach, M. Rohrbach, W. Qiu, A. Friedrich, M. Pinkal, and B. Schiele. Coherent multi-sentence video description with variable level of detail. In *German Conference on Pattern Recognition (GCPR)*, September 2014.

[37] M. Rohrbach, W. Qiu, I. Titov, S. Thater, M. Pinkal, and B. Schiele. Translating video content to natural language descriptions. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 433–440, 2013.

[38] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, pages 1–42, Apr. 2015.

[39] M. Schuster and K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, Nov. 1997.

[40] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2014.

[41] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[42] C. Sun and R. Nevatia. Semantic aware video transcription using random forest classifiers. In *Proceedings of the European Conference on Computer Vision*, pages 772–786, 2014.

[43] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, 2014.

[44] 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*, 4, 2012.

[45] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri. C3D: generic features for video analysis. In *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

[46] R. Vedantam, C. L. Zitnick, and D. Parikh. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4566–4575, 2015.

[47] S. Venugopalan, M. Rohrbach, J. Donahue, R. J. Mooney, T. Darrell, and K. Saenko. Sequence to sequence - video to text. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4534–4542, 2015.

[48] S. Venugopalan, H. Xu, J. Donahue, M. Rohrbach, R. J. Mooney, and K. Saenko. Translating videos to natural language using deep recurrent neural networks. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*, pages 1494–1504, 2015.

[49] O. Vinyals and Q. V. Le. A neural conversational model. In *ICML Deep Learning Workshop*, 2015.

[50] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164, 2015.

[51] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Action Recognition by Dense Trajectories. In *IEEE Conference on Computer Vision & Pattern Recognition*, pages 3169–3176, June 2011.

[52] T. Wen, M. Gasic, N. Mrksic, P. Su, D. Vandyke, and S. J. Young. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Conference on Empirical Methods in Natural Language Processing*, 2015.

[53] P. Werbos. Backpropagation through time: what does it do and how to do it. In *Proceedings of IEEE*, volume 78, pages 1550–1560, 1990.

[54] H. Xu, S. Venugopalan, V. Ramanishka, M. Rohrbach, and K. Saenko. A multi-scale multiple instance video description network. *CoRR*, abs/1505.05914, 2015.

[55] R. Xu, C. Xiong, W. Chen, and J. J. Corso. Jointly modeling deep video and compositional text to bridge vision and language in a unified framework. In *Proceedings of AAAI Conference on Artificial Intelligence*, 2015.

[56] L. Yao, A. Torabi, K. Cho, N. Ballas, C. Pal, H. Larochelle, and A. Courville. Describing videos by exploiting temporal structure. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4507–4515, 2015.

[57] H. Yu and J. M. Siskind. Learning to describe video with weak supervision by exploiting negative sentential information. In *AAAI Conference on Artificial Intelligence*, pages 3855–3863, Jan. 2015.