Attention-based Densely Connected LSTM for Video Captioning

Yongqing Zhu, Shuqiang Jiang
Key Lab of Intelligent Information Processing, Institute of Computing Technology, CAS, Beijing, 100190, China
University of Chinese Academy of Sciences, Beijing, 100049, China
yongqing.zhu@vipl.ict.ac.cn,sqjiang@ict.ac.cn

ABSTRACT

Recurrent Neural Networks (RNNs), especially the Long Short-Term Memory (LSTM), have been widely used for video captioning, since they can cope with the temporal dependencies within both video frames and the corresponding descriptions. However, as the sequence gets longer, it becomes much harder to handle the temporal dependencies within the sequence. And in traditional LSTM, previously generated hidden states except the last one do not work directly to predict the current word. This may lead to the predicted word highly related to the last generated hidden state other than the overall context. To better capture long-range dependencies and directly leverage early generated hidden states, in this work, we propose a novel model named Attention-based Densely Connected Long Short-Term Memory (DenseLSTM). In DenseLSTM, to ensure maximum information flow, all previous cells are connected to the current cell, which makes the updating of the current state directly related to all its previous states. Furthermore, an attention mechanism is designed to model the impacts of different hidden states. Because each cell is directly connected with all its successive cells, each cell has direct access to the gradients from later ones. In this way, the long-range dependencies are more effectively captured. We perform experiments on two publicly used video captioning datasets: the Microsoft Video Description Corpus (MSVD) and the MSR-VTT, and experimental results illustrate the effectiveness of DenseLSTM.

CCS CONCEPTS

• **Computing methodologies** → **Natural language generation**; **Computer vision**.

KEYWORDS

Video Captioning, Deep Learning, LSTM, Attention Mechanism, Vision and Language

ACM Reference Format:
Yongqing Zhu, Shuqiang Jiang. 2019. Attention-based Densely Connected LSTM for Video Captioning. In *Proceedings of the 27th ACM International Conference on Multimedia (MM '19), October 21–25, 2019, Nice, France.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3343031.3350932

MM '19, October 21–25, 2019, Nice, France

ACM ISBN 978-1-4503-6889-6/19/10…$15.00
https://doi.org/10.1145/3343031.3350932

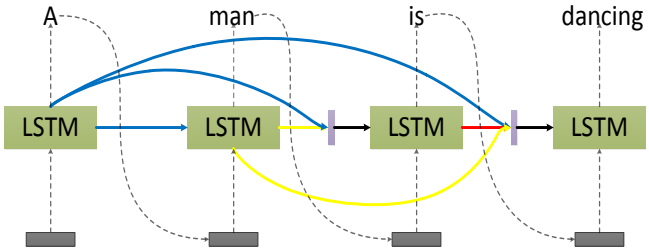

Figure 1: The topology of a four time step DenseLSTM. Each cell takes all preceding hidden states as input, and also passes its own output to the following recurrent units.

1 INTRODUCTION

With the development of the Internet and the social network service, videos have drawn increasing attention in recent years. The ability to automatically understand and summarize the video contents is highly desired for computers. Generating natural descriptions for videos known as video captioning is one of such important tasks. Early works like [3, 30] ususlly first identify the possible semantic contents, and generate a sentence based on the predefined templates. The performances of template-based methods are also highly related to the quality of predefined templates, which usually fail to model the richness of natural language.

Thanks to the development of recurrent neural networks, researchers have investigated on automatically describing video contents with a sentence. Many recent works [7, 8, 20, 21, 35] have illustrated the effectiveness of the recurrent neural networks especially the Long Short-Term Memory (LSTM) network because of the ability of capturing temporal dependencies within sequential data. Compared with traditional RNN, by introducing non-linear gating units to control information flow, LSTM can alleviate the problem of vanishing or exploding gradients to some extent, nevertheless the problem still exists. The long-range information can also be easily forgetton, as the path between the start and end of a sequence gets longer. When traditional LSTM is used as decoder to generate captions, previous hidden states except the last generated one do not work directly to predict the current word. The contexts within earlier generated hidden states should firstly pass through several LSTM cells before they are used to predict the target word. And due to the length of the path that signals have to traverse, those information may be insufficiently used and long-range dependencies are also hard to be modeled, which may make the predicted word more related to partial information other than the general context.

Different from [5, 16] which utilize CNNs to address the above issues, in this work, we propose a model named Attention-based Densely Connected LSTM (DenseLSTM). Figure 1 illustrates this

layout schematically. In DenseLSTM, each recurrent cell is connected with all preceding cells to maximize the usage of all previous generated information. When predicting the target word, the proposed model first reconstructs the temporal information kept in previous hidden states, then we feed the reconstructed hidden state instead of the last generated one into the current recurrent unit. And intuitively hidden states generated at different time have different impacts to predict the target word, attention mechanims are designed to automatically integrate information within different hidden states. By introducing direct connections between the current cell and all previous ones, all earlier generated informaiton can be directly used to predict the target words without having to go through multiple units, so the long-range information can persist until the last word is predicted. Since both earlier hidden states and the last one are directly utilized to predict the current word, more abundant informaiton is used and the predicted word is more related to the overall context. The model is named as Attention-based Densely Connected LSTM (DenseLSTM) for two reasons. One is that the architecture of this model is similar to DenseNet [12]. In this model, all previous LSTM cells including the last one are directly connected to the current one. The second is that attention mechanisms are used to fuse the information within different hidden states. As noted in [14, 15], in traditional LSTM, compared with short-term hidden state which is recomputed at each time step, long-term hidden state only computes increments on the basis of the last long-term hidden state, which makes long-term information persist for much longer compared with short-term information. So we only utilize dense connection to modify the short-term information in this work.

This work attempts to realign previous hidden states based on global context for the task of video captioning. In summary, this paper makes the following contributions:

- We propose a model named Attention-based Densely Connected LSTM (DenseLSTM), which introduces dense connections to recurrent neural networks. By directly incorporating previous hidden states, more complete information is utilized to predict each word, which makes the generated sentences more accurate. To learn the impacts of different hidden states, two attention based methods are designed.
- By using attention mechanism, our model also provides an insight to why earlier information is also important to predict the target word, compared with the latest generated information.
- The proposed model is evaluated on the MSVD dataset and MSR-VTT dataset. And the experimental results also have verified the effectiveness of our method.

2 RELATED WORK

Video Captioning

Early video captioning methods [3, 30] usually used predefined templates to generate video descriptions. They firstly identify the possible semantic contents with visual classifiers, and then feed the identified semantic contents in the templates. It is intitutive that the results of template-based methods are highly dependent on the quality of the predefined templates.

Thanks to the development of recurrent neural networks, most recent researchers usually applied recurrent neural networks to automatically model temporal dependencies within the input video frames and generate the corresponding sentences. Some works like [8, 36] used recurrent networks as decoder to generate the video descriptions given the visual contents. [36] used pre-trained CNN model to extract the visual features from video frames, mean pooled the visual features as the entire video feature, and then fed the visual information to an LSTM layer to generate the captions. [8] employed Conditional Random Field (CRF) to get semantic tuples, and then used an LSTM layer to translate the semantic tuples into a sentence.

To more effectively encode the temporal information within both the video clips and the descriptions, works like [4, 24, 35] used LSTM as both the encoder and the decoder. [35] is the first so-called sequence to sequence method in video captioning. They proposed a stacked LSTM architecture, where an encoder LSTM was used to encode video features and another was used to generate the corresponding caption based on the encoded visual features. Different from [35] which directly used LSTM as encoder, in [24], they proposed a hierarchical recurrent video encoder. Their method applied a way similar to a convolutional operation to embed the visual features of video frames. In [4], the proposed method could learn to adapt its temporal structure to the input data by training a boundary detector, which could put homogeneous visual features in the same chunk. They then used LSTM to encode the visual features in different chunks.

To exploit latent information within the given video clip, [26] proposed a method to predict visual attributes in videos, and introduced a method to fuse the detected visual attributes. [9] applied topic information in their work. They trained both the encoder and the decoder jointly by introducing an interpretive loss which minimized the distance between the visual feature and the topic representation. Different from [9, 26] which exploited the textual or semantic information, [7, 40] proposed methods to exploit visual or audio information. In [7], the authors proposed an encoder to fuse specific modalities like static image features, motion features and audio features. They then fed the fused features to the decoder to predict the captions. [40] also proposed a method that exploited multiple modalities (e.g., frame, motion, and audio streams). To more effectively combine different modalities, they trained modality-specific LSTMs to capture the intrinsic representations of individual modalities.

For video captioning, the input visual features are usually extracted from the whole frames, but there are also regions that contain more detailed information. [41, 45] proposed methods to utilize informative regions within the video clips. In [41], a method was introduced to incorporate temporal structure of global features and regions-of-interest features in parallel. In [45], Faster-RCNN was employed to extract object regions in frames and a dynamic programming based method was devised to generate possible region tubes. Then a decoder LSTM was used to decode the features within different region tubes. In order to optimize captioning models on the basis of test metrics (e.g. CIDEr-D, METEOR), methods [6, 37] based on Reinforcement Learning (RL) have also been proposed.

In this paper, our work focuses on the decoding stage. Based on recent video captioning works [4, 7, 24, 35], we follow the general

encoder-decoder framework, where an encoder is used to embed the input visual contents and the proposed DenseLSTM as the decoder to generate the target sentence.

DenseNet

Convolutional neural networks (CNNs) have become the most commonly used machine learning approach for most visual tasks. As CNNs become deep, a problem emerges: as information about input or gradient pass through many layers, it can vanish when it reaches the end (or begining) of the network. Recent works [11, 13, 17] have shown that convolutional networks can be substantially deeper and efficiently trained if shorter connections between the inputs and the outputs are embedded. In [12], they further introduced the Densely Connected Convolutional Network (DenseNet), where each layer is connected to every subsequent layers. And, in DenseNet, each layer obtains additional inputs from all preceding layers and passes its own feature-maps to all subsequent layers. Dense connections are proved have following advantages: they can alleviate the gradient-vanishing problem, strengthen feature propagation and encourage feature reuse. Similar to DenseNet in CNNs, DenseLSTM also introduces shorter connections between different neural units in LSTM. Different from DenseNet where the concatenated preceding feature maps are fed to subsequent layers, in DenseLSTM, the hidden states generated by earlier recurrent cells are fed as inputs to the successive cells via attention mechanisms.

Attention Mechanism

Attention mechanisms play an important role in both visual tasks and natural language processing tasks. In [10], to automatically search for parts of a source sentence relevant to the target word, they introduced an attention mechanism in neural machine translation. Compared with neural machine translation which translates from the source sentence to the target sentence, video captioning can be considered as a special version of translation that translates from the input video clips to the output natural language. Works like [7, 41, 42] also applied attention mechanisms in video captioning, where attention mechanisms are usually used to automatically fuse the outputs of the encoder.

An attention function can be described as mapping a query and a set of key-value pairs to an output. And self-attention is a special version of attention mechanisms, where both the query and key-value pairs come from the same space. Self-attention mechanism is usually used to model the intra dependencies within a sequence. In [2], they used self-attention mechanism to embed the inner dependencies within both the source sentences and the partially generated target sentences. In [1], they applied self-attention mechanism in memory slots, where each memory will attend over all of the other memories. In [18, 22], self-attention mechanism was used to embed both the video contents and the corresponding captions.

In DenseLSTM, all preceding recurrent cells are connected with subsequent cells. And information generated at different time should have different impacts on the current word. To effectively modulate and integrate different hidden states, self-attention mechanism and a task specific attention mechanism are employed.

3 METHOD

3.1 Long Short-Term Memory

Long Short-Term Memory [14], a variant of recurrent neural networks (RNNs), is discovered to work better for capturing temporal dependencies in sequential data. Given an input sequence $\{x_1, x_2, ..., x_n\}$, an LSTM could output two sequences: short-term hidden states $\{h_1, h_2, ..., h_n\}$ and long-term hidden states $\{c_1, c_2, ..., c_n\}$. Given inputs x_t, h_{t-1} and c_{t-1}, the updates of the $t-th$ LSTM unit are:

$$input\ gate\ :\ i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{1}$$
$$forget\ gate\ :\ f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{2}$$
$$output\ gate\ :\ o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{3}$$
$$cell\ input\ :\ g_t = \phi(W_g x_t + U_g h_{t-1} + b_g) \tag{4}$$
$$cell\ state\ :\ c_t = i_t \odot g_t + f_t \odot c_{t-1} \tag{5}$$
$$cell\ output\ :\ h_t = o_t \odot \phi(c_t) \tag{6}$$

where $\sigma(x)$ is logic sigmoid function, $\odot$ denotes the element-wise multiplication of two vectors and $\phi(x)$ is hyperbolic tangent function. W_*, U_* and b_* are the parameters to be learned. For the convenience of the reader, all the hidden states referred are short-term hidden states in the following parts.

3.2 Baseline Method

Our baseline method is similar to the architecture proposed by [10], where a bidirectional LSTM is used as encoder to embed visual inputs, and a traditional LSTM as decoder to generate the descriptions.

3.2.1 Encoder. Given an input video V with N_v sampled frames, we firstly use a pre-trained convolutional neural network (CNN) to extract the visual feature of each sampled frame. We denote the extracted visual features of V as $\{f_1, f_2, ...f_{N_v}\}$, where f_i is the feature of the $i-th$ frame. To encode the input video frames, the extracted visual features $\{f_1, f_2, ..., f_{N_v}\}$ are fed into the commonly used bidirectional LSTM:

$$\overrightarrow{h}_t = \overrightarrow{LSTM_f}(f_t, \overrightarrow{h}_{t-1}) \tag{7}$$
$$\overleftarrow{h}_t = \overleftarrow{LSTM_r}(f_t, \overleftarrow{h}_{t+1}) \tag{8}$$

where $\overrightarrow{LSTM_f}$ and $\overleftarrow{LSTM_r}$ are the forward LSTM and the reverse LSTM of the bidirectional LSTM respectively, $\overrightarrow{h}$ and $\overleftarrow{h}$ are the output hidden states.

To obtain the encoded feature of each frame, the output forward hidden states $\overrightarrow{h}$ and the reverse hidden states $\overleftarrow{h}$ are concatenated. In this way, the encoded feature of each frame summarizes the features of both the preceding frames and the following frames. We denote the outputs of the encoder as $\{h_1, h_2, ...h_{N_v}\}$, where $h_n = \left[\overrightarrow{h_n}^T; \overleftarrow{h_n}^T\right]^T$.

3.2.2 Decoder. To generate the corresponding captions, a traditional LSTM is employed as the decoder in our baseline method.

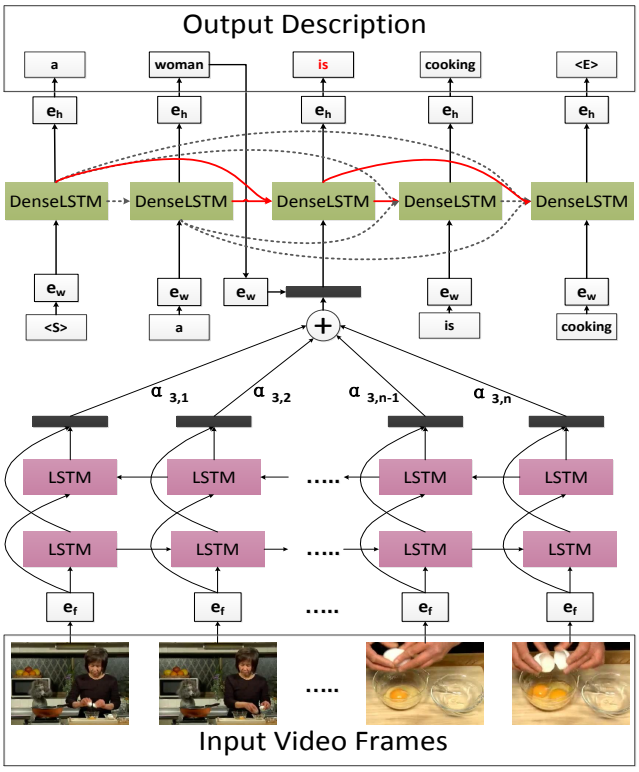

Figure 2: An overview of our framework with a bidirectional LSTM as encoder and the proposed DenseLSTM as decoder. In the above figure, we focus on the cell which is utilized to predict the word "is". It recives inputs from preceding decoder cells, and passes its outputs to subsequent cells. e_f and e_w denote the embedding layers which are employed to embed visual features and the input words. And e_h is the linear transformation layer used to predict the target word.

Let us denote the hidden states of the decoder as s. The hidden state s_t generated by the $t-th$ decoder cell is computed by

$$s_t = LSTM(w_t, s_{t-1}, V_t) \tag{9}$$

where w_t is the embedded word vector of the $t-th$ input word, s_{t-1} is the hidden state generated by the last decoder cell. And V_t is a weighted sum of encoder hidden states $\{h_1, h_2, ...h_{N_v}\}$:

$$V_t = \sum_{u=1}^{N_v} \alpha_{t,u} h_u \tag{10}$$

The attention weights $\{\alpha_{t,u}\}$ in Eq. (10) are trained to give higher weights to certain encoder hidden states which match the current input decoder state s_{t-1} better, and are computed as:

$$\alpha_{t,u} = \frac{exp(e_{t,u})}{\sum_{i=1}^{N_v} exp(e_{t,i})} \tag{11}$$

where the attention function $e_{t,u}$ is defined as:

$$e_{t,u} = w_a^T tanh(W_s s_{t-1} + W_h h_u + b_a) \tag{12}$$

where w_a, W_s, W_h and b_a are the parameters to be learned in attention layer.

3.3 The Proposed Method

The overall architecture is illustrated in Figure 2. We apply a bidirectional LSTM as encoder to obtain the encoded hidden states $\{h_1, h_2, ...h_{N_v}\}$ as described in the baseline method. Different from the baseline method where a traditional LSTM is utilized as decoder, the proposed DenseLSTM is used to generate the corresponding descriptions. In DenseLSTM, the outputs of all preceding recurrent cells are used as inputs to the current cell, which makes the paths between different cells shortened. By introducing short connections in LSTM, the gradients from successive units could directly flow into all preceding units, which could further facilitate the training of the model.

In DenseLSTM, we maintain the components of traditional LSTM, which is described in section 3.1. When predicting the $t-th$ word, instead of directly feeding the hidden state of the last recurrent unit s_{t-1} to the current cell, we firstly reconstruct the outputs of all preceding cells $\{s_1, s_2, ...s_{t-1}\}$, and then feed the reconstructed hidden state into the model. Denote the reconstructed hidden state as A_t, the output of the $t-th$ decoder cell is computed by:

$$s_t = LSTM(w_t, A_t, V_t) \tag{13}$$

where A_t is computed by fusing $\{s_1, s_2, ...s_{t-1}\}$. We also use the reconstructed hidden state A_t to modulate the embedded visual features from hidden states $\{h_1, h_2, ...h_{N_v}\}$ obtained by the encoder:

$$e_{t,u} = w_a^T tanh(W_s A_t + W_h h_u + b_a) \tag{14}$$

$$\alpha_{t,u} = \frac{exp(e_{t,u})}{\sum_{i=1}^{N_v} exp(e_{t,i})} \tag{15}$$

$$V_t = \sum_{u=1}^{N_v} \alpha_{t,u} h_u \tag{16}$$

where w_a, W_s, W_h and b_a are the parameters to be learned in attention layer.

Different from [12, 13], where the feature maps are combined through summation or concatenation before they are fed to future layers, we propose two attention-based methods to effectively fuse $\{s_1, s_2, ...s_{t-1}\}$. And attention mechanisms are employed for the following reasons:

- Hidden states generated at different time have different context information, and different context information also has different effects to predict the current word. If we use methods like summation or concatenation to fuse previous hidden states, they would have the same weights to predict the target word.
- Hidden states generated by recurrent neural networks are hard to be interpreted. And it is also hard to identify the importance of a generated hidden state. But one superiority of attention mechanism is that it can automatically learn to give higher weights to more relevant parts during training.

According to the differences of the internal exponents, the models are named as Self-Attention DenseLSTM and Latest Guiding DenseLSTM respectively.

3.3.1 Self-Attention DenseLSTM. This method is directly inspired by [1, 2, 18], where self-attention mechanism is used to model the dependencies within the sequence. For self-attention mechanism, both the queries and key-value pairs come from the same space. For convenience, we denote the queries, keys, values as Q, K, V respectively. In Self-Attention DenseLSTM, Q, K and V are the same, where $Q = \{s_0, s_1, ...s_{t-1}\}$, $K = \{s_0, s_1, ...s_{t-1}\}$ and $V = \{s_0, s_1, ...s_{t-1}\}$. To compute the reconstructed decoder hidden state A_t, a simple linear projection is firstly used to embed queries Q and keys K. Next, we use the embedded queries to perform a scaled dot-product attention over the embedded keys. The returned scalars are then passed through a softmax function to produce a set of weights, which can be used to return the weighted sum of values V:

$$M_t = softmax((W^q Q)^T W^k K)V^T, \tag{17}$$

where W^q and W^k are parameters to be learned in the embedding layers. After the above attention layer, we add a simple fully connected feed-forward network as an activation layer, which is applied to each position in M_t separately and identically. The feed-forward layer consists of one linear projection layer and one ReLU layer:

$$FFN(x) = ReLU(xW_1 + b_1) \tag{18}$$

And A_t is obtained by a max-pooling layer:

$$A_t = max(FFN(M_t)) \tag{19}$$

Regardless of the batch size in practice, M_t is a matrix of size $t \times d_{model}$ and A_t is a vector of size $1 \times d_{model}$, which can be directly used in LSTM.

3.3.2 Latest Guiding DenseLSTM. When using recurrent neural networks as decoder, the input hidden state s_{t-1} implicitly contains the context information within $\{s_1, s_2, ...s_{t-2}\}$. So s_{t-1} can be considered as an abstraction of previous context. Compared with Self-Attention DenseLSTM where both the queries and key-value pairs are $\{s_1, s_2, ...s_{t-1}\}$, the query used to generate A_t is the latest generated s_{t-1} and the key-value pairs remain unchanged in Self-Attention DenseLSTM.

To reduce computation, in Latest Guiding DenseLSTM, A_t is a weighted sum of decoder hidden states $\{s_1, s_2, ...s_{t-1}\}$:

$$A_t = \sum_{u=1}^{t-1} \beta_{t,u} s_u \tag{20}$$

where $\{\beta_{t,u}\}$ acts as attention weights, and are computed as

$$\beta_{t,u} = \frac{exp(e_{t,u})}{\sum_{i=1}^{t-1} exp(e_{t,i})} \tag{21}$$

And the attention function $e_{t,u}$ is defined as:

$$e_{t,u} = w_a^T tanh(W_s s_{t-1} + W_h s_u + b_s) \tag{22}$$

where w_a, W_s, W_h and b_s are the parameters to be learned.

3.4 Sentence Generation

For a video clip v with a description $W = [w_1, w_2, ..., w_T]$ encoded with one-hot vectors ($1 - of - N$ coding, where N is the size of the word vocabulary), the video sentence generation problem can be formulated by minimizing the following energy loss function:

$$E(W, v) = -logP(W|v), \tag{23}$$

which is the negative log probability of the correct sentence given the input video. The log probability of a sentence is equal to the summation of the log probabilities over each word via chain rule, which can be written as:

$$logP(W|v) = \sum_{t=1}^{T} logP(w_t|v, w_0, w_1, ..., w_{t-1}) \tag{24}$$

Similar to other RNN based language models, we apply a softmax layer to compute the probability distribution of the current word over the word space,

$$P(w_t|w_0, w_1, ..., w_{t-1}, v) \propto exp(w_t^T W_y s_t) \tag{25}$$

where s_t is the output hidden state generated by the $t-th$ decoder cell, W_y is the parameters of the linear embedding layer. And the above energy loss function is optimized over the entire training dataset.

4 EXPERIMENTS

4.1 Datasets

We conduct experiments on two publicly used video captioning datasets: the Microsoft Video Description Corpus (MSVD) [30] and the MSR-VTT [39].

Microsoft Video Description Corpus (MSVD). The Microsoft Video Description Corpus contains 1,970 short video clips from YouTube. There are 80,839 human annotated sentences in total, with about 41 sentences per clip. On average, each sentence contains about 8 words. We adopt the same data splits as provided in [35], with 1,200 video clips for training, 100 video clips for validation and the rest 670 clips for testing.

MSR-VTT. MSR-VTT is a dataset for general video captioning, which is collected from a wide variety of video categories. MSR-VTT contains of 10,000 video clips with 20 human annotated descriptions per video clip. We use the standard split as provided in [39], 6,513 video clips as training set, 497 video clips as validation set and the remaining 2,990 video clips as test set.

4.2 Preprocessing and training details

We extract static image features as well as motion features from input videos of both MSVD and MSR-VTT. To encode static image features, we use the pretrained VGGNet model [31] trained on the 1.2M image ILSVRC-2012 object classification subset of the Imagenet dataset [29] to extract the features of the sampled frames. To model video motion and activities, we use the pretrained C3D model [33] (trained on the Sports-1M dataset). For C3D network, the model outputs a feature of fixed length every 16 frames, which encodes motion features around the middle frame of the window. In this work, we use the window of default size with a stride of 6 frames.

Table 1: Experimental results on the MSVD dataset. The short name in the brackets indicates the features, where A denotes AlexNet, V denotes VGGNet, C denotes C3D, G denotes GoogleNet and R denotes ResNet50. And B@N, M and C are short for BLEU@N, METEOR and CIDEr-D respectively. All values are reported as percentage (%).

Medel	B@1	B@2	B@3	B@4	M	C
S2VT(V+A)[35]	-	-	-	-	29.8	-
LSTM-E(V+C)[25]	78.8	66.0	55.4	45.3	31.0	-
BAE(R+C)[4]	-	-	-	42.5	32.4	63.5
HRNE(G+C)[24]	81.1	68.6	57.8	46.7	**33.9**	-
h-RNN(V+C)[43]	81.5	70.4	60.4	49.9	32.6	65.8
Multimodal Attention(V+C)[7]	-	-	-	**52.4**	32.0	68.8
MA-LSTM(G+C)[40]	**82.3**	**71.1**	**61.8**	52.3	33.6	70.4
Baseline method(V+C)	79.2	67.0	57.0	47.0	32.8	71.5
Self-Attention DenseLSTM(V+C)	80.1	68.9	59.7	49.7	32.2	72.0
Latest guiding DenseLSTM(V+C)	80.7	69.5	60.1	50.4	32.9	**72.6**

To maintain the same granularity, for static image features, we use VGGNet to extract the middle frame within the sliding window of C3D. And the hidden activation vectors of fully connected layer fc7 are used for the static image features, which output a sequence of 4096-dimensional feature vectors. For the motion features, we use the fully connected layer fc6-1 in C3D, which also produces features of 4096 dimension. We then concatenate the static image features and motion features, which leads to 4096+4096-dimensional visual feature vectors. Instead of directly feeding the visual feature vectors into our model, a linear embedding layer is learned as the input of the model.

The annotated descriptions are converted to lower case and tokenized after removing punctuation characters. For MSVD, we use the vocabulary provided by [35]. For MSR-VTT, we retain only words which appear at least 5 times. During training, we add a special token <start_token> to present the beginning of the caption, and an end-of-sentence tag <end_token> at its end, so the model can deal with captions with variable length. When the model is used to generate the captions, the decoder is given a <start_token> as input for the first timestep, then the word with highest probability is sampled and fed to the next decoder unit, until the <end_token> is predicted. To input the words to our model, we firstly use traditional one-hot vector to present it, and then embed the one-hot vector to a lower dimensional space by applying a linear transformation layer.

For the encoder bidirectional LSTM, the forward and the reverse LSTM share the same parameters. And we set the hidden state size to 1024-dimension. The hidden state size of our decoder is also set to 1024-dimension. We use 512-dimension vectors to embed both the input words and visual features before they are fed into our model. For both MSVD and MSR-VTT datasets, the model is trained with a learning rate of 0.0001, and the parameters are initialized with a uniform distribution in the range [-0.05,0.05]. Training is performed by minimizing the log-likelihood loss via the Adam [28] optimizer with the default coefficients. And the mini-batch size is set to 32. To regularize the training and avoid overfitting, we apply the commonly used Dropout [44] with retain probability 0.5 for recurrent cells. For both the baseline method and the proposed methods, we unroll video encoder bidirectional to 40 time steps. For videos with fewer than 40 frames, we pad the remaining inputs with zeros. For longer videos, we only retain the first 40 frames. For the language decoder, we unroll the maximal time steps to 20. And similar trick is applied to ensure the word sequences within this limit.

To evaluate the similarity between the ground truth and the model generated sentences, we employ four popular metrics: BLEU [27], METEOR [19], ROUGE_L [23] and CIDEr-D [34]. To ensure a fair evaluation, we use the Microsoft CoCo evaluation toolkit[1]. And all values in our tables are reported as percentage (%).

4.3 Experimental Results on MSVD

On the MSVD dataset, we compare our method with the following methods: S2VT [35], LSTM-E [25], HRNE [24], Multimodal Attention [7], h-RNN [43], BAE [4] and MA-LSTM [40]. The above compared methods use visual features extracted from same or comparable models. For MSVD dataset, we use BLEU@1-4, METEOR and CIDEr-D to evaluate the generated sentences. Moreover, we use beam search with size 5 for all our methods, which is consistent with the above methods.

As shown in Table 1, the qualitative results across most metrices indicate that the proposed Self-Attention DenseLSTM and Latest Guiding DenseLSTM outperform the traditional LSTM, when used as decoder. And our methods are also comparable to those compared methods. It is also worth noting that both Self-Attention DenseLSTM and Latest Guiding DenseLSTM reach a better performance on CIDEr-D, even though some metrics like BLEU can not surpass methods like MA-LSTM [40]. The explanation noted by [7] for this problem is: in MSVD dataset, each video has multiple "ground-truth" descriptions, but some annotated "ground-truth" are incorrect. Since BLEU and METEOR scores do not consider frequency of words in the ground truth, they can be strongly affected. In contrast, CIDEr-D is a voting-based metric that is robust to errors in ground truth.

It also can be observed that Latest Guiding DenseLSTM outperforms Self-Attention DenseLSTM a little bit in MSVD. One possible reason for this phenomenon is that, when we use self-attention mechanism to fuse past hidden states, both the early generated

[1]https://github.com/tylin/coco-caption

Table 2: The importance of visual features, where V denotes features extracted by VGGNet and C denotes features extracted by C3D. And B@4, M, R, C are short for BLEU@4, METEOR, ROUGE_L and CIDEr-D respectively. SA-DenseLSTM and LG-DenseLSTM are short for Self-Attention DenseLSTM and Latest Guiding DenseLSTM respectively.

Method	Feature	B@4	M	R	C
SA-DenseLSTM	V	44.7	31.2	67.3	64.6
SA-DenseLSTM	C	45.8	30.9	67.9	64.0
LG-DenseLSTM	V	45.0	31.0	67.1	65.5
LG-DenseLSTM	C	46.3	31.5	68.2	64.4
SA-DenseLSTM	V+C	49.7	32.2	69.0	72.0
LG-DenseLSTM	V+C	**50.4**	**32.9**	**69.9**	**72.6**

hidden states and the latest generated hidden states are used as queries to modulate all hidden states and the importance of them are the same. For example, when predicting the $t-th$ word, both the s_1 and s_{t-1} are respectively utilized to modulate the temporal dependencies within $\{s_1, s_2, ...s_{t-1}\}$ via scaled dot attention and then stack the results in a matrix, which is then fed into the feed-forward and max-pooling layer. And the weights of s_1 and s_{t-1} are the same when they are used as queries. But the latest generated hidden states may play a more important role for the target word. In Latest Guiding DenseLSTM, it can be considered that both the s_1 and s_{t-1} are used as queries, but they have different weights, the weight for s_1 is 0 and the weight for s_{t-1} is 1. For different datasets, the performances of Self-Attention DenseLSTM and Latest Guiding DenseLSTM may have differences, but they are both effective as decoder to generate descriptions.

To evaluate the importance of different visual features, we also report the results of the proposed methods merely based on the object appearance features or the motion features, with other components of our framework unchanged. The results are shown in Table 2. Compared with the performances based on single visual modality, it can be observed that the performances are much improved when using both the static image features and motion features together. This is also consistent with previous works like [7, 43]. Another phenomenon is, compared with VGG features, using the C3D features has a higher performance on metrics like BLEU@4 and ROUGE_L, but a little bit lower on CIDEr-D for both the proposed methods.

To evaluate how the attention mechanisms contribute, we also devise two shallow fusion methods, which are also used to fuse previous hidden states. The first uses a mean-pooling layer to fuse hidden states, and the second method just simply adds the hidden states up. It is worth noting that these two methods do not have parameters to be trained. The results are shown in Table 3. Simple fusion methods and the baseline method have comparable performances, and even outperform baseline method on some metrics. More reasonable fusion methods like Latest Guiding DenseLSTM could further boost the performances.

The design of Self-Attention DenseLSTM is more complicated than Latest Guiding DenseLSTM. To illustrate the importance of each component within our Self-Attention DenseLSTM which is described in 3.3.1, we conduct the following four experiments:

Table 3: The importance of attention mechanism in our model. The results are obtained based on both the object features and the motion features. ADD and Mean refer to the methods that mean pools and adds up hidden states respectively.

Method	B@4	M	R	C
ADD	**50.4**	32.3	69.3	70.1
MEAN	48.8	32.6	69.2	71.4
baseline	47.0	32.8	68.6	71.5
SA-DenseLSTM	49.7	32.2	69.0	72.0
LG-DenseLSTM	**50.4**	**32.9**	**69.9**	**72.6**

Table 4: The importance of components within Self-Attention DenseLSTM.

Model	B@4	M	R	C
SA-DenseLSTM v1	48.1	32.1	**69.1**	71.4
SA-DenseLSTM v2	**49.7**	**32.2**	69.0	72.0
SA-DenseLSTM v3	49.6	31.8	68.7	70.9
SA-DenseLSTM v4	48.0	31.9	68.7	**72.7**

- Version 1: Only the scaled dot-product attention layer in Equation 17 and the max-pooling layer are employed to fuse the hidden states.
- Version 2: We maintain all the components in section 3.3.1, where the scaled dot-product attention, FFN (fully-connected feed-forward network) and the max-pooling layer are used.
- Version 3: Based on Version 2, we add another FFN (fully-connected feed-forward network) layer.
- Version 4: Based on Version 2, we use mean-pooling layer instead of the max-pooling layer.

The results are shown in Table 4. It can be observed that applying moderate FFN layers can boost performance, and mean-pooling layer does not have remarkable boost for the performance in this model. So we finally adopt the Version 2 as our Self-Attention DenseLSTM.

We also show some samples of MSVD in Figure 3. Both the baseline method and our proposed methods can generate relevant sentences, but the sentences generated by our proposed methods are usually more accurate.

4.4 Expermental Results on MSR-VTT

We also train and evaluate our model on MSR-VTT dataset. Different from MSVD dataset, audio information and the corresponding categories are also provided for videos in MSR-VTT, which indicate the possible activities of the given video clips. To simplify the proposed model and reduce the number of parameters, the audio and category information are not utilized in our models, even though they are proved valuable in works like [6, 40]. We compare our models with MP-LSTM [36], S2VT [35], LSTM-E [25], MA-LSTM [40], MCNN+MCF [38], hLSTMat [32]. For MSR-VTT dataset, we use BLEU@4, METEOR and CIDEr-D to evaluate the generated sentences. And the results are depicted in Table 5.

Figure 3: Examples of the MSVD dataset. The videos are represented by sampled frames, and the sentences are generated by 1) baseline method, 2) our proposed SA-DenseLSTM and LG-DenseLSTM. GT is short for ground truth.

Figure 4: Examples of the MSR-VTT dataset. The videos are represented by sampled frames, and the sentences are generated by 1) baseline method, 2) our proposed SA-DenseLSTM and LG-DenseLSTM. GT is short for ground truth.

Table 5: Performances of the proposed methods and the compared approaches on MSR-VTT, where B@4, M and C are short for BLEU@4, METEOR and CIDEr-D scores. V, G, C, R and A denotes VGGNet, GoogleNet, C3D, ResNet152 and Audio feature. All values are reported as percentage (%).

Model	B@4	M	C
MP-LSTM(G+C+A)[36]	35.7	25.6	38.1
S2VT(G+C+A)[35]	36.0	26.0	39.1
LSTM-E(G+C+A)[25]	36.1	25.8	38.5
hLSTMat(R)[32]	**38.3**	26.3	-
MA-LSTM(G+C+A)[40]	36.5	26.5	41.0
MCNN+MCF(R)[38]	38.1	**27.2**	42.1
Baseline method(V+C)	37.0	26.1	41.4
Latest Guiding DenseLSTM(V+C)	37.6	26.2	42.0
Self-Attention DenseLSTM(V+C)	38.1	26.6	**42.8**

As shown in Table 5, the proposed Latest Guiding DenseLSTM and Self-Attention DenseLSTM outperform traditional LSTM when used as decoder for video captioning, which is consistent with the results on MSVD. Compared with the above methods, the proposed methods achieve better performance on both BLEU@4 and CIDEr-D, which further verfied the effectiveness of our methods. We also show some generated sentences in Figure 4.

4.5 Sentence Generation Analysis

By computing the attention weights of previous generated hidden states with the proposed Latest Guiding DenseLSTM, the model conveys two important information. First, in most cases, the information generated by the last recurrent cell is usually the most important to predict the current word. Second, compared with the last generated hidden state, some earlier generated hidden states may also have equal or even higher importance. We show the weights of different hidden states in Figure 5. The weights are computed based on the overall test split of MSR-VTT dataset. To obtain the above weights distribution in Figure 5, we can firstly obtain the weights distribution for every video clip, and then calculate the

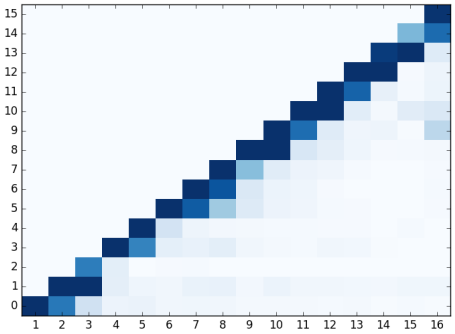

Figure 5: The weights of previously generated hidden states based on the test split of MSR-VTT dataset. If the color of an element is deeper than others, it means the corresponding hidden state is more important.

average value for each position. For each column in Figure 5, each element indicates the effect of the hidden state generated at that time. More specifically, the element within coordinate (x, y) means the average importance of the hidden state generated by the $y-th$ cell when it is utilized to predict the $x-th$ word.

5 CONCLUSION

In this work, we propose a model named DenseLSTM to sufficiently utilize previous generated hidden states. To further enhance the fusion of previous hidden states, two attention mechanisms are designed. By computing the attention weights, our method also reveals why earlier generated information is also important to predict the current word. On two commonly used datasets, the experimental results demonstrate the effectiveness of the proposed approaches.

ACKNOWLEDGEMENT

This work was supported in part by the National Natural Science Foundation of China under Grant 61532018, in part by Beijing Natural Science Foundation under Grant L182054, in part by National Program for Special Support of Eminent Professionals and National Program for Support of Top-notch Young Professionals.

REFERENCES

[1] Santoro Adam, Faulkner Ryan, Raposo David, Rae Jack, Chrzanowski Mike, Weber Theophane, Wierstra Daan, Vinyals Oriol, Pascanu Razvan, and Lillicrap Timothy. 2018. Relational recurrent neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.). Curran Associates, Inc., 7299–7310. http://papers.nips.cc/paper/7960-relational-recurrent-neural-networks.pdf

[2] Vaswani Ashish, Shazeer Noam, Parmar Niki, Uszkoreit Jakob, Jones Llion, Gomez Aidan N, Kaiser Łukasz, and Polosukhin Illia. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems (NIPS)*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 5998–6008. http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

[3] Kojima Atsuhiro, Tamura Takeshi, and Fukunaga Kunio. 2002. Natural Language Description of Human Activities from Video Images Based on Concept Hierarchy of Actions. *International Journal of Computer Vision (IJCV)* (2002).

[4] Lorenzo Baraldi, Costantino Grana, and Rita Cucchiara. 2017. Hierarchical Boundary-Aware Neural Encoder for Video Captioning. In *Internaltional Conference on Computer Vision and Pattern Recogintion (CVPR)*.

[5] Jingwen Chen, Yingwei Pan, Yehao Li, Ting Yao, Hongyang Chao, and Tao Mei. 2019. Temporal Deformable Convolutional Encoder-Decoder Networks for Video Captioning. In *Association for the Advance of Artificial Intelligence (AAAI)*.

[6] Yangyu Chen, Shuhui Wang, Weigang Zhang, and Qingming Huang. 2018. Less Is More: Picking Informative Frames for Video Captioning. In *The European Conference on Computer Vision (ECCV)*.

[7] Hori Chiori, Hori Takaaki, Lee Teng-Yok, Ziming Zhang, Harsham Bret, Hershey John R., Marks Tim K., and Sumi Kazuhiko. 2017. Attention-Based Multimodal Fusion for Video Description. In *International Conference on Computer Vision (ICCV)*.

[8] Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. 2015. Long-term Recurrent Convolutional Networks for Visual Recognition and Description. In *Internaltional Conference on Computer Vision and Pattern Recogintion (CVPR)*.

[9] Yinpeng Dong, Hang Su, Jun Zhu, and Bo Zhang. 2017. Improving Interpretability of Deep Neural Networks with Semantic Information. In *Internaltional Conference on Computer Vision and Pattern Recogintion (CVPR)*.

[10] Bahdanau Dzmitry, Cho Kyunghyun, and Bengio Yoshua. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*.

[11] Huang Gao, Sun Yu, Liu Zhuang, Sedra Daniel, and Weinberger Kilian. 2016. Deep Networks with Stochastic Depth. In *European Conference on Computer Vision (ECCV)*.

[12] Huang Gao, Liu Zhuang, van der Maaten Laurens, and Weinberger Kilian Q. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[14] Sepp Hochreiter and Jurgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* (1997).

[15] Ba Jimmy, Hinton Geoffrey E, Mnih Volodymyr, Leibo Joel Z, and Ionescu Catalin. 2016. Using Fast Weights to Attend to the Recent Past. In *Neural Information Processing Systems (NIPS)*. 4331–4339.

[16] Gehring Jonas, Auli Michael, Grangier David, Yarats Denis, and N.Dauphin Yann. 2017. Convolutional Sequence to Sequence Learning. In *International Conference on Machine Learning (ICML)*.

[17] Rupesh Kumar Srivastava, Klaus Greff, and JÃŒrgen Schmidhuber. 2015. Training Very Deep Networks. In *Neural Information Processing Systems (NIPS)*.

[18] Kim Kyung-Min, Choi Seong-Ho, Kim Jin-Hwa, and Zhang Byoung-Tak. 2018. Multimodal Dual Attention Memory for Video Story Question Answering. In *The European Conference on Computer Vision (ECCV)*.

[19] Alon Lavie and Abhaya Agarwal. 2005. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

[20] Xiangyang Li and Shuqiang Jiang. 2018. Bundled Object Context for Referring Expressions. In *IEEE Transactions on Multimedia, (TMM)*.

[21] Xiangyang Li and Shuqiang Jiang. 2019. Know More Say Less: Image Captioning Based on Scene Graphs. In *IEEE Transactions on Multimedia, (TMM)*.

[22] Xiangpeng Li, Jingkuan Song, Lianli Gao, Xianglong Liu, Wenbing Huang, Xiangnan He, and Chuang Gan. 2019. Beyond RNNs: Positional Self-Attention with Co-Attention for Video Question Answering. In *Association for the Advance of Artificial Intelligence (AAAI)*.

[23] Chin-Yew Lin. 2004. Rouge: a package for automatic evaluation of summaries. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

[24] Pingbo Pan, Zhongwen Xu, Yi Yang, Fei Wu, and Yueting Zhuang. 2016. Hierarchical recurrent neural encoder for video representation with application to captioning. In *Internaltional Conference on Computer Vision and Pattern Recogintion (CVPR)*.

[25] Yingwei Pan, Tao Mei, Ting Yao, Houqiang Li, and Yong Rui. 2016. Jointly Modeling Embedding and Translation to Bridge Video and Language. In *Internaltional Conference on Computer Vision and Pattern Recogintion (CVPR)*.

[26] Yingwei Pan, Ting Yao, Houqiang Li, and Tao Mei. 2017. Video Captioning with Transferred Semantic Attributes. In *Internaltional Conference on Computer Vision and Pattern Recogintion (CVPR)*.

[27] Kishore Papineni, Salim Roukos, Todd Ward, and Wei jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

[28] Diederik P.Kingma and Jimmy Lei Ba. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*.

[29] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)* 115, 3 (2015), 211–252. https://doi.org/10.1007/s11263-015-0816-y

[30] Guadarrama Sergio, Krishnamoorthy Niveda, Malkarnenkar Girish, Venugopalan Subhashini, Mooney Raymond, Darrell Trevor, and Saenko Kate. 2013. YouTube2Text: Recognizing and Describing Arbitrary Activities Using Semantic Hierarchies and Zero-Shot Recognition. In *International Conference on Computer Vision (ICCV)*.

[31] Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*.

[32] Jingkuan Song, Lianli Gao, Zhao Guo, Wu Liu, Dongxiang Zhang, and Hengtao Shen. 2017. Hierarchical LSTM with Adjusted Temporal Attention for Video Captioning. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, (IJCAI)*.

[33] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. 2015. Learning Spatiotemporal Features with 3D Convolutional Networks. In *Internaltional Conference on Computer Vision and Pattern Recogintion (CVPR)*.

[34] Ramakrishna Vedantam, C.Lawrence Zitnick, and Devi Parikh. 2015. CIDEr: Consensus-based image description evaluation. In *Internaltional Conference on Computer Vision and Pattern Recogintion (CVPR)*.

[35] Subhashini Venugopalan, Marcus Rohrbach, Jeff Donahue, Raymond Mooney, Trevor Darrell, and Kate Saenko. 2015. Sequence to Sequence – Video to Text. In *International Conference on Computer Vision (ICCV)*.

[36] Subhashini Venugopalan, Huijuan Xu, Jeff Donahue, Rohrbach Marcus, Mooney Raymond, and Saenko Kate. 2015. Translating Videos to Natural Language Using Deep Recurrent Neural Networks. In *The North American Chapter of the Association for Computational Linguistics (NAACL)*.

[37] Xin Wang, Wenhu Chen, Jiawei Wu, YuanFang Wang, and William Yang Wang. 2018. Video Captioning via Hierarchical Reinforcement Learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[38] Aming Wu and Yahong Han. 2018. Multi-modal Circulant Fusion for Video-to-Language and Backward. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, (IJCAI)*.

[39] Jun Xu, Tao Mei, Ting Yao, and Yong Rui. 2016. Msr-vtt: A large video description dataset for bridging video and language. In *Internaltional Conference on Computer Vision and Pattern Recogintion (CVPR)*.

[40] Jun Xu, Ting Yao, Yongdong Zhang, and Tao Mei. 2017. Learning Multimodal Attention LSTM Networks for Video Captioning. In *Proceedings of the ACM International Conference on Multimedia (ACM MM)*. 9.

[41] Ziwei Yang, Yahong Han, and Zheng Wang. 2017. Catching the Temporal Regions-of-Interest for Video Captioning. In *Proceedings of the ACM International Conference on Multimedia (ACM MM)*.

[42] Li Yao, Atousa Torabi, Kyunghyun Cho, Nicolas Ballas, Christopher Pal, Hugo Larochelle, and Aaron Courville. 2015. Describing videos by exploiting temporal structure. In *International Conference on Computer Vision (ICCV)*.

[43] Haonan Yu, Jiang Wang, Zhiheng Huang, Yi Yang, and Wei Xu. 2016. Video Paragraph Captioning using Hierarchical Recurrent Neural Networks. In *Internaltional Conference on Computer Vision and Pattern Recogintion (CVPR)*.

[44] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent Neural Network Regularization. *CoRR* abs/1409.2329 (2014). arXiv:1409.2329 http://arxiv.org/abs/1409.2329

[45] Bin Zhao, Xuelong Li, and Xiaoqiang Lu. 2018. Video Captioning with Tube Features. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, (IJCAI)*. International Joint Conferences on Artificial Intelligence Organization, 1177–1183. https://doi.org/10.24963/ijcai.2018/164