Video Fill In the Blank using LR/RL LSTMs with Spatial-Temporal Attentions

Amir Mazaheri, Dong Zhang, and Mubarak Shah
Center for Research in Computer Vision, University of Central Florida, Orlando, FL 32816
amirmazaheri@cs.ucf.edu, dzhang@cs.ucf.edu, shah@crcv.ucf.edu

Abstract

*Given a video and a description sentence with one missing word, "**source sentence**", Video-Fill-In-the-Blank (VFIB) problem is to find the missing word automatically. The contextual information of the sentence, as well as visual cues from the video, are important to infer the missing word accurately. Since the source sentence is broken into two fragments: the sentence's left fragment (before the blank) and the sentence's right fragment (after the blank), traditional Recurrent Neural Networks cannot encode this structure accurately because of many possible variations of the missing word in terms of the location and type of the word in the source sentence. For example, a missing word can be the first word or be in the middle of the sentence and it can be a verb or an adjective. In this paper, we propose a framework to tackle the textual encoding: Two separate LSTMs (the LR and RL LSTMs) are employed to encode the left and right sentence fragments and a novel structure is introduced to combine each fragment with an* external memory *corresponding to the opposite fragments. For the visual encoding, end-to-end spatial and temporal attention models are employed to select discriminative visual representations to find the missing word. In the experiments, we demonstrate the superior performance of the proposed method on challenging VFIB problem. Furthermore, we introduce an extended and more generalized version of VFIB, which is not limited to a single blank. Our experiments indicate the generalization capability of our method in dealing with such more realistic scenarios.*

1. Introduction & Related Works

In computer vision, due to Deep Convolutional Neural Networks (CNNs) [1, 2, 3, 4] dramatic success has been achieved in detection (e.g. object detection [5]) and classification (e.g. action classification [6]). Likewise, Recurrent Neural Networks (RNN) [7, 8, 9] have been demonstrated to be very useful in Natural Language Processing (NLP) for language translation. Recently, new problems such as Visual Captioning (VC) [10, 11, 12] and Visual Question Answering (VQA) [13, 14, 15, 16, 17, 18] have drawn a lot of interest, as these are very challenging problems and extremely valuable for both computer vision and natural language processing. Both Visual Captioning and Visual Question Answering are related to the Video-Fill-in-the-Blank (VFIB) problem, which is addressed in this paper.

Single Blank:

He ___ up the steps of the stand and away. (Runs)

Multiple Blanks:

He ___ up the steps of the ___ and away. (Runs, Stand)

Figure 1. Two examples of the Video-Fill-in-the-Blank problem[19] with a single blank and multiple blanks.

Visual Captioning (VC) needs deep understanding of both visual (i.e. image or video) and textual cues. Recently, several approaches for VC have been introduced. Some of these algorithms focus on leveraging RNNs, which take visual input, focus on different regions of the image and leverage attention models to produce captions [12, 20]. Furthermore, a dense captioning method is proposed in [20], where for a given image, multiple captions are generated and for each caption a corresponding bounding box is produced. Visual Question Answering (VQA) has deep roots in textual question answering [21, 22, 23]. The goal of textual question answering is to answer a question based on a collection of sentences.

The major difference between VQA and VC problems is that in VQA there is a "question" as a sentence in addition to the image or video. This makes the problem harder since the question can be about details of the image or video; However, for the VC problem, any factual sentence about the image can be a correct caption. Some methods use a combination of the question and the image/video features through

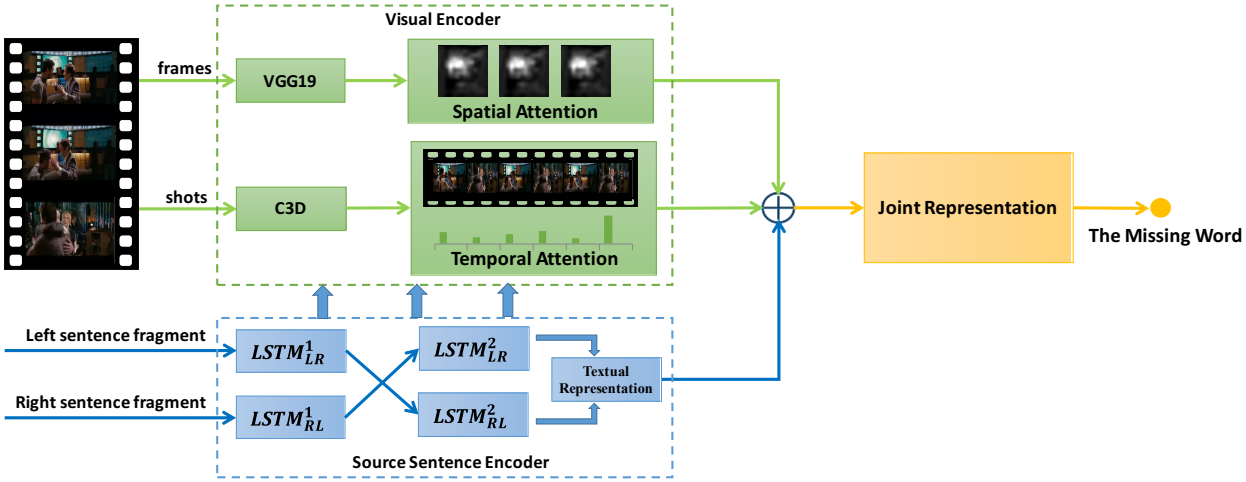

Figure 2. Our proposed method to solve Video Fill In the Blank (VFIB) problem. Source sentence encoding, spatial and temporal attention models are shown.

LSTMs [24] and output the answer to the question. Some other methods combine the image features with the words one by one, for instance, the methods in [17, 22]([17] is an extension of [22]), use visual and textual information as input to the dynamic memory networks and convert it to a sequence of words through an RNN with an iterative attention process. In [18], a binary question answering (yes/no) on abstract (unreal) scenes is presented. Another binary answering problem which verifies existence of a relation between two concepts (for example: dogs eat ice cream) is proposed in [25], which uses CNNs to verify the relation. MovieQA [26] presents another form of VQA, where the input is a video and its subtitle. The authors in [26] use multiple choice questions and demonstrate that the visual information does not contribute to the final results.

Video Fill-In-the-Blank (VFIB) [27] is a relatively new problem and has a broad variety of real world applications such as visual guided report generation, corrupted data recovery, officer report automation for police departments, etc. VFIB is related to Video Question Answering (VQA) problem, however, it has some differences. VQA datasets usually have a bias to some specific forms of questions such as location, color, counting, etc. Therefore, the answers for each of these questions are limited to a small dictionary of words. For example, in DAQUAR question answering dataset [15], in many cases, the word "table" is the answer to "What is" question type, and "White" to "What color" questions [14]. On the other hand, in VFIB problem, there is no limit on the type of the missing word and the word can be a verb, adverb, adjective, etc. Furthermore, for VQA problems, the "question" is always a complete sentence. In this scenario, it is easier to encode the "question" in the model (for example, using standard models such as Recurrent Neural Networks) and then use it to predict the answer. Therefore, it is easy and straightforward to use off-the-shelf techniques; However, there is no "question" in the VFIB problem, so it is tricky to encode the source sentence using the standard encoding techniques. Also, the blank can be at any location in the sentence, namely, in the middle or very first or last word of the source sentence. Last but not least, for VQA problem, it is very expensive to collect datasets, hence limiting its practical applications. It is time-consuming since the human annotators have to generate questions and answers one by one while watching the videos. Some efforts have been made to make the question-answer generation automatic [14], but these approaches generate a lot of abnormal questions, and they tend to work well only on object related questions.

Due to lack of a complete sentence and the various types of missing words and also different possible locations of the missing word in the source sentence, the VFIB problem cannot be well-solved by traditional Recurrent Neural Network (e.g. LSTM). Therefore, we propose a new framework which leverages the source sentence structure and integrates the spatial and temporal attention modules to fully exploit the visual information.

This paper makes the following contributions: First, we propose a novel approach which can encode left and right sentence fragments. It encodes each fragment with a separate LSTM and feeds the output of each fragment to the opposite fragment (for example, from left fragment to the right fragment) as an external memory. This is very different from basic BiLSTM approaches [28, 7], where each of left and right fragments are encoded separately and then joined by a simple fusion technique (e.g. concatenation, summation, etc). Moreover, we show that our text encoding achieves superior performance over other methods. Second, we propose a novel framework for the spatial and temporal

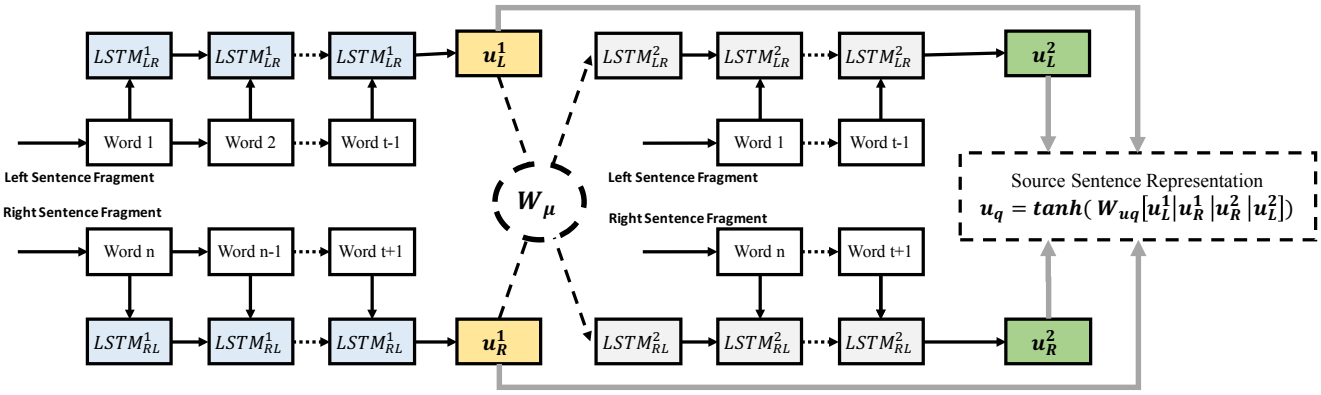

Figure 3. An illustration of the source sentence encoding approach. In the first stage, Each source sentence is formatted as two fragments, the left and right sentence fragments. Each sentence fragment is passed through an LSTM. In the second stage, left and right fragments' LSTMs are combined with a memory from the opposite side.

attention models, which are trained end-to-end. Third, different visual features are employed by the spatial and temporal attention models, which exploit the visual cues from different domains. Fourth, in our formulation we perform multiple features fusion in end-to-end fashion without any need for pre-training single modules like [29, 30], which deal with multi-modal problems. We also introduce a more general version of VFIB problem and employ source sentences with multiple blanks. We show that our method produces the superior results.

The organization of the rest of the paper is as follows: In Section 2 we introduce different components of the proposed method in detail; in Section 3, we show experimental results on both single blank and multiple blanks problems; and we conclude in Section 4.

2. Method

We formulate the VFIB problem as a word prediction problem with two sentence fragments (left sentence fragment "q_l", and right sentence fragment "q_r") and the video v:

$$\hat{b} = \arg\max_{b \in \beta} p(b|\mathbf{q}_l, \mathbf{q}_r, v, \theta), \quad (1)$$

where $\beta \subset V$ is the set of words to fill in the blank and V is the dictionary of all the words in our dataset. $\hat{b}$ is the prediction (the word to be filled in the blank), and θ is the model parameters. Our framework is depicted in Figure 2. The proposed approach consists of four components: 1) source sentence encoding, 2) spatial attention model, 3) temporal attention model, and 4) inference of the missing word. We discuss these four components in the following subsections.

2.1. Source Sentence Encoding

In this section, we introduce how to encode the source sentence which consists of left and right fragments and a blank between them. Figure 3 shows an illustration of the proposed source sentence encoding approach. The blank can be anywhere in a source sentence, thus a single LSTM architecture will not work very well, since left or right fragment can be very short, and in many cases the missing word depends on both fragments. However, a simple LSTM will fail if one fragment is very short or both fragments are needed for prediction. Also, the blank can belong to any classes of words (e.g. verb, adjective, etc.). These complexities of the source sentence make the textual encoding difficult.

We treat the source sentence as two fragments; the left fragment from the first word to the word before the blank and the right fragment backward from the last word to the word after the blank. Our source sentence encoding module has three stages. In the first stage, we encode each of the left and right fragments separately with two independent LSTMs. In the second stage, we encode left and right fragments along with the encoded fragments from the opposite side in stage one. Namely, we use the encoded left fragment in the first stage as an external memory to encode the right fragment in stage two and vice versa. We call it "external memory", since it is computed using the opposite fragment. In fact, the external memory makes the model to understand each fragment better, since it has some information from the opposite fragment. Finally, the model learns to combine the output of both stages and generates the final source sentence's representation called u_q (Figure 3). Our approach has two major differences compared to BiLSTM [28, 7]. First, we use the the opposite fragments as an external memory for each sides, and second, our method learns how to combine the left and right encoded fragments. In the following, we provide more details.

Assume that the source sentence has n words, and the t'th word is missing. The left and right fragments' se-

quences can be embedded as:

$$\mathbf{q}_l^1 = \mathbf{W}_x^1[\mathbf{x}^1, \mathbf{x}^2, ..., \mathbf{x}^{t-1}]$$
$$\mathbf{q}_r^1 = \mathbf{W}_x^1[\mathbf{x}^n, \mathbf{x}^{n-1}, ..., \mathbf{x}^{t+1}], \quad (2)$$

where $\mathbf{x}^i \in \{0,1\}^{|V|}$ is an one-hot vector representation of i'th word in the source sentence, and $\mathbf{W}_x^1 \in \mathbb{R}^{c\times|V|}$ is a word embedding matrix ($|V|$ is the size of the dictionary, and c is the encoding dimension). $\mathbf{q}_l^1$ and $\mathbf{q}_r^1$ are two sequences where each element $q_{l_{(j)}}^1 \in \mathbb{R}^c$ is a continuous vector representing j'th word in q_l^1 sequence. We model the left and right sentence's fragments separately using two LSTMs:

$$\mathbf{u}_l^1 = LSTM_{LR}^1(\mathbf{q}_{l_{(i)}}^1), \quad (i = 1, ..., (t-1))$$
$$\mathbf{u}_r^1 = LSTM_{RL}^1(\mathbf{q}_{r_{(i)}}^1), \quad (i = 1, ..., (n-t)) \quad (3)$$

where $\mathbf{u}_l^1, \mathbf{u}_r^1 \in \mathbb{R}^h$ are the last hidden states from the "LR" and "RL" LSTMs respectively (Fig. 3) and h is the LSTMs' hidden state size. Since the missing word is related to both left and right fragments of a sentence, we have an extra stage to encode Left/Right fragments with respect to an external memory coming from the opposite side, namely Right/Left fragments. In this way, the first stage processes each of fragments separately and the second stage processes them with respect to opposite side's encoded representation.

$$\mathbf{q}_l^2 = [\mu_l, \mathbf{W}_x^2[\mathbf{x}^1, \mathbf{x}^2, ..., \mathbf{x}^{t-1}], \mu_l]$$
$$\mathbf{q}_r^2 = [\mu_r, \mathbf{W}_x^2[\mathbf{x}^n, \mathbf{x}^{n-1}, ..., \mathbf{x}^{t+1}], \mu_r], \quad (4)$$

where $\mathbf{W}_x^2$ is of the same size as $\mathbf{W}_x^1$ and $\mu_r, \mu_l \in \mathbb{R}^c$ are two external memory vectors obtained by:

$$\mu_r = \mathbf{u}_l^1 \mathbf{W}_\mu$$
$$\mu_l = \mathbf{u}_r^1 \mathbf{W}_\mu \quad (5)$$

where $\mathbf{W}_\mu \in \mathbb{R}^{h\times c}$ encodes the LSTMs outputs to memory vectors. $\mathbf{q}_l^2$ and $\mathbf{q}_r^2$ are two sequences while $|\mathbf{q}_l^2| - |\mathbf{q}_l^1| = |\mathbf{q}_r^2| - |\mathbf{q}_r^1| = 2$ because of the external memory vectors attached to them and each element of them is a continuous vector. Similar to Eq. 3, we encode these two sequences with two different LSTMs, namely $LSTM_{LR}^2$ and $LSTM_{RL}^2$ (Fig 3), to obtain $\mathbf{u}_l^2, \mathbf{u}_r^2 \in \mathbb{R}^h$ as encoded left and right fragments in the second stage. Note that; since $\mathbf{W}_x^1$ and $\mathbf{W}_x^2$ are two different matrices, LR/RL LSTMs of first and second stages, observe completely different sequence of vectors. Also, none of these four LSTMs share any parameters with the other ones.

Finally, a proper combination of $\mathbf{u}_l^1$, $\mathbf{u}_r^1$, $\mathbf{u}_l^2$ and $\mathbf{u}_r^2$ as the final representation of the source sentence is needed. We concatenate and combine them by a fully connected layer as the final representation of the source sentence:

$$\mathbf{u}_q = \tanh(\mathbf{W}_{uq}[\mathbf{u}_l^1|\mathbf{u}_r^1|\mathbf{u}_l^2|\mathbf{u}_r^2]), \quad (6)$$

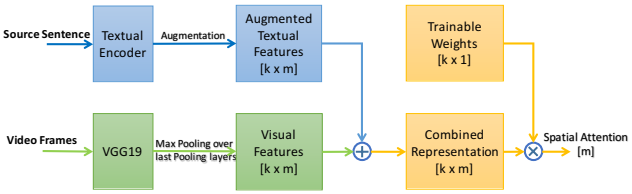

Figure 4. An illustration of the spatial attention model. The model assigns importance score to each region of an image based on the source sentence.

where $\mathbf{W}_{uq} \in \mathbb{R}^{d\times 4h}$ is a trainable weights matrix applied to learn a proper combination of four vectors. We refer $\mathbf{u}_q \in \mathbb{R}^d$ as the source sentence representation (textual feature) in the following sections and it is a bounded vector due to $\tanh(\cdot)$ activation function.

2.2. Spatial Attention Model

We use a CNN (i.e., VGG-19 [2], more details are provided in Section 3.4) to extract visual features. The output from the last pooling layer of CNNs can be considered as a feature map with spatial information about the input image. Figure 4 shows an illustration of the spatial attention model. First, we apply max-pooling over the raw CNN features (i.e the output from the last pooling layer of VGG-19 pre-trained network) from all video key-frames to obtain a spatial visual feature from the whole video:

$$\mathbf{\Phi}_F = \tanh(\mathbf{W}_f \Theta(\mathbf{\Phi}_f(f^t))|_{f^t \in F}), \quad (7)$$

where $\mathbf{\Phi}_f(\cdot)$ is the spatial visual feature map extraction function (more details are provided in Section 3.4), F represents all the video key-frames in v, f_t is a video frame at time t, $\Theta(\cdot)$ is the max-pooling function, $\mathbf{W}_f$ is a trainable transformation matrix, and $\mathbf{\Phi}_F \in \mathbb{R}^{d\times m}$ is the intermediate visual feature matrix where each column is a feature vector corresponding to a spatial region in the original video frames, d is the same as $\mathbf{u}_q$ in Eq.6, and m is the number of spatial regions in the video frame.

We use spatial attention model [31] to pool the intermediate visual features $\mathbf{\Phi}_F$. The first step is to combine the source sentence representation $\mathbf{u}_q$ with the intermediate visual features $\mathbf{\Phi}_F$:

$$\mathbf{\Psi}_F = \tanh((\mathbf{W}_F \mathbf{\Phi}_F) \oplus (\mathbf{W}_u \mathbf{u}_q + \mathbf{b}_u)), \quad (8)$$

where $\mathbf{W}_F \in \mathbb{R}^{k\times d}$ and $\mathbf{W}_u \in \mathbb{R}^{k\times d}$ are two transformations on the intermediate visual features and source sentence representation to align them and have the same dimensions. $\mathbf{b}_u$ is the bias term, and $\oplus$ is a summation between a matrix and an augmented vector (i.e. the source sentence representation $\mathbf{u}_q$ has to be expanded, or repeated m times, in order to have the same dimension as $\mathbf{W}_F \mathbf{\Phi}_F$. The matrix

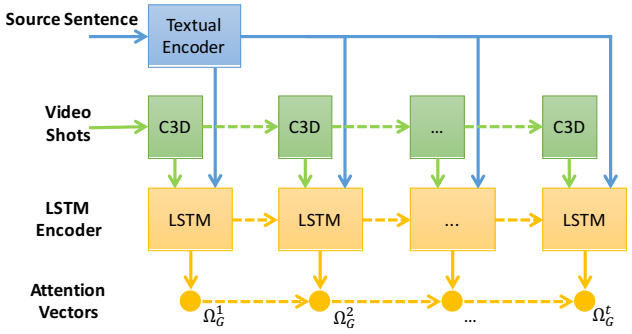

Figure 5. An illustration of the temporal attention model. An LSTM is used to find relevant shots based on the source sentence.

$\mathbf{\Psi}_F \in \mathbb{R}^{k \times m}$ is used to find the final attention scores over all the regions:

$$\mathbf{p}_{sp} = \text{softmax}(\mathbf{\Psi}_F^T \mathbf{w}_{sp}), \tag{9}$$

where $\mathbf{w}_{sp} \in \mathbb{R}^{k \times 1}$ is a trainable weight vector and $\mathbf{p}_{sp} \in \mathbb{R}^{m \times 1}$ is the spatial attention vector. The final spatial pooled visual vector is a weighted average over all m regional intermediate spatial feature vectors:

$$\mathbf{u}_{sp} = \Phi_F \mathbf{p}_{sp}, \tag{10}$$

where $\mathbf{u}_{sp} \in \mathbb{R}^d$ is the spatial pooled visual vector. It is a bounded vector since $\mathbf{\Phi}_F$ is bounded.

2.3. Temporal Attention Model

Temporal dynamics of a video and the motion information plays a significant role in video understanding, especially in actions and events understanding. However, in our spatial-pooled visual representation presented in previous section, the whole video is represented as one vector, and there is no temporal information. Therefore, we propose to model the temporal dynamics of the video using a temporal attention model. Figure 5 shows an illustration of this component in our approach. We divide a video into a number of shots and represent the shots as below:

$$\mathbf{\Phi}_G = \tanh(\mathbf{W}_g[\mathbf{\Phi}_g(g^1), \mathbf{\Phi}_g(g^2), ..., \mathbf{\Phi}_g(g^{|G|})]), \tag{11}$$

where $\mathbf{\Phi}_g(\cdot)$ is the feature extraction function (C3D [32], which encodes the temporal information. More details are provided in Section 3.4), G represents the set of video shots and g^i is the ith video shot. $\mathbf{W}_g \in \mathbb{R}^{d \times z}$ is a transformation matrix, where z is the original dimension of the feature vector $\mathbf{\Phi}_g(g^i)$, and k is the encoding dimension. We combine the video shots representation $\mathbf{\Phi}_G \in \mathbb{R}^{d \times |G|}$ and the source sentence representation $\mathbf{u}_q$ as:

$$\mathbf{\Psi}_G = \tanh((\mathbf{W}_G \mathbf{\Phi}_G) \oplus (\mathbf{W}_u \mathbf{u}_q + b_u)), \tag{12}$$

where $\mathbf{W}_G \in \mathbb{R}^{k \times d}$, is a mapping for shot representation $\mathbf{\Phi}_G$. We share $\mathbf{W}_u$ and $\mathbf{b}_u$ with the spatial attention model in Eq. 8. Similar to Eq. 8, in order to apply the summation $\oplus$, $\mathbf{u}_q$ is repeated $|G|$ times to have the same number of columns as $\mathbf{\Phi}_G$. Each column of matrix $\mathbf{\Psi}_G \in \mathbb{R}^{k \times |G|}$ is the combination of a single shot and the source sentence. An LSTM is employed to model the dynamics between shots and the source sentence:

$$\mathbf{\Omega}_G^i = \text{LSTM}(\Psi_G^i) \quad (i = 1...|G|), \tag{13}$$

where $\mathbf{\Psi}_G^i$ is the i'th column of $\mathbf{\Psi}_G$. The output of this LSTM is a sequence of attention vectors corresponding to each shot $\mathbf{\Omega}_G = [\Omega_G^1, \Omega_G^2, ..., \Omega_G^{|G|}]$ (See Fig. 5). However, we need to make probabilities out of these vectors and for this purpose we simply use a $softmax$ operator:

$$\mathbf{p}_{tp} = softmax(\mathbf{\Omega}_G \mathbf{w}_{tp}), \tag{14}$$

where $\mathbf{w}_{tp} \in \mathbb{R}^{k \times 1}$ is a trainable weight vector and $\mathbf{p}_{tp} \in \mathbb{R}^{|G| \times 1}$ is the temporal attention of the shots and the final temporal-pooled representation is a weighted average over all the shot features $\mathbf{\Phi}_G$ with the attention model:

$$\mathbf{u}_{tp} = \mathbf{\Phi}_G \mathbf{p}_{tp}, \tag{15}$$

where $\mathbf{u}_{tp} \in \mathbb{R}^d$ is of the same dimension as the spatial-pooled features and the source sentence representation. Its values are also bounded since $\mathbf{\Phi}_G$ is obtained by passing through a $\tanh(\cdot)$ activation in Eq. 11. Using this temporal attention model, we capture the dynamics of the visual features which are related to the source sentence representation.

2.4. Inference of the Missing Word

Here, we discuss how to infer the missing word or fill in the blank. Let β be the vocabulary to fill in the blank ($|\beta|$ as its size). We aim to find a probability for each word candidate in β, which needs a joint representation (summation fusion [31]) of all three components as mentioned earlier:

$$\mathbf{u} = [\mathbf{u}_q + \mathbf{u}_{sp} + \mathbf{u}_{tp}], \tag{16}$$

where $\mathbf{u} \in \mathbb{R}^d$ is an joint representation of all three features: source sentence representations, spatially- and temporally-pooled visual representations. Note that all three vectors $\mathbf{u}_q$, $\mathbf{u}_{sp}$ and $\mathbf{u}_{tp}$ in equations 6, 10 and 15 are bounded, since they have been obtained by passing through a $\tanh(\cdot)$ activation. They also have the same dimension d and this makes the summation fusion [31] applicable and effective. For the final inference of the missing word, we compute a probability of each candidate word as follows:

$$P_{blank} = softmax(\mathbf{W}_{blank} \mathbf{u}), \tag{17}$$

where $\mathbf{W}_{blank} \in \mathbb{R}^{|\beta| \times d}$. This is followed by a multinomial logistic regression "$softmax$" to find the probabilities vector $\mathbf{P}_{blank} \in \mathbb{R}^{|\beta|}$. Based on Eq. 1, the final answer is:

$$\hat{b} = \arg\max_{b \in \beta} P_{blank}(b). \tag{18}$$

3. Experiments

We perform experiments on two datasets: the original LSMDC Dataset [27] to evaluate the single blank VFIB problem (i.e. there is only one blank in the source sentence), and an extended LSMDC Movie Dataset to evaluate our performance on the multiple blanks VFIB problem (i.e. there are multiple blanks in the sentence).

3.1. LSMDC Movie Dataset (Single Blank)

In this set of experiments, we use the movie dataset [27, 19, 33], which has been used in Large Scale Movie Description and Understanding Challenge (LSMDC) [27, 34]. Movies are a rich source of visual information and become much more valuable when proper textual meta-data is provided. Movies benefit from many textual data like the subtitle, audio descriptions and also movie synopsis. LSMDC dataset consists of respectively "$91,908$", "$6,542$", "$10,053$" and "$9,578$" movie clips as Training, Validation, Public and Private Test sets. We use the standard splits provided by [27, 34]. Each clip comes with a sentence annotated by an expert. There can be multiple source sentences built for one clip. We use respectively "$296,960$", "$21,689$" and "$30,349$" samples as training, validation and test as the standard split provided by [34].

3.1.1 Quantitative Results

Here, we compare our proposed method with other approaches and baselines to show its superior performance for the VFIB task. We have chosen some of these baselines from methods for visual question answering problem, which are applicable to this problem as well. The comparison table (Table 1) has four parts. The first part is for methods which only use the text to find the missing word; the second part is for methods which just use the video; the third part is for methods which use both text and video; and the last part is for different configurations of the proposed method. We report the accuracy (same as in [34]) of each method which is the ratio of number of missing words that are inferred correctly to the total number of blanks. Here are some details about these methods:

LSTM Left/Right Sentence fills the blank by just looking at the left/right fragment of the missing word. This experiment shows that both fragments are equally important. **BiLSTM** finds the missing word based on a BiLSTM [7], which encodes the input sentence using two different LSTMs; one takes the input from the last word to the first word and the other one in the reverse. The blank word is recovered based on BiLSTM's output in missing word location. **Our Sentence Embedding** as described in section 2.1. This approach finds the missing word by using just vector $\mathbf{u}_q$, without any visual features. For a fair comparison with BiLSTM method, we have fixed the LSTM cells sizes and also the word embedding lengths in all the experiments. The authors in [27] report a few baselines using **GoogleNet** [3] and **C3D** [32] features. The difference between the baselines in[27] and ours, shows the actual importance of our attention models and integration of the textual encoding and visual modules in our method. **Video+ Textual Encoding** corresponds to "IMG+LSTM" in [14]. Key-frames are passed through the VGG-19 pre-trained network to extract $4,096$ dimensional vector of "fc7" layer for each key-frame. Then, a max-pooling over all the features of all frames will generate a video feature vector and the rest of steps are the same as explained in [14]. We have used simple BiLSTM instead of LSTM to deal with two fragments of sentence in VFIB. **2Videos+ Textual Encoding** [14], sim-

Method	Accuracy
Text Only	
Random Guess	0.006
LSTM Left Sentence	0.155
LSTM Right Sentence	0.165
BiLSTM	0.320
Our Sentence Encoding (w/o Second Stage)	0.340
Our Sentence Encoding	**0.367**
Human [27]	*0.302*
Video Only	
BiLSTM Just Video	0.055
Text + Video	
GoogleNet-2D [27]	0.349
C3D [27]	0.345
GoogleNet-2D-Finetuned [27]	0.353
GoogleNet-2D + C3D-Finetuned [27]	0.357
Video + Textual Encoding [14]	0.341
2Videos + Textual Encoding [14]	0.350
Ask Your Neurons [24]	0.332
SNUVL [35]	0.380
SNUVL (Ensembled Model) [35]	0.407
Human [27]	*0.687*
Ours	
Single Model (VGG19 + C3D w/o Attention)	0.378
Single Model (w/o Spatial Attention)	0.390
Single Model (w/o Temporal Attention)	0.392
Single Model (w/o Second Stage)	0.396
Single Model (w/o LR/RL LSTMs)	0.387
Single Model	**0.406**
Ensembled Model	**0.434**

Table 1. Results on "Movie Fill-in-the-Blank" dataset.

ilar to previous case, uses two different representations of the video. One is attached to the beginning of each fragment and the other one to the end. **Ask Your Neurons** [24] encodes the visual CNN feature and concatenate with each of words and pass them through left and right LSTMs one by one. The answer is inferred based on the last output of LSTMs. **SNUVL** [35] is the best reported method on LSMDC FIB. It uses a concept detection method over the videos, following by an attention model over the detected concepts, to find the missing word. **Ensemble model** [36] is a technique to boost the performance, when the optimization process reaches different local optima, based on random factors like initialization. We train the model multiple times with different initializations and sum the final scores from all the trained models.

We also test our model performance by removing each of components, namely spatial attention, temporal attention and also replacing our source sentence encoding with the BiLSTM in baselines. We also show the results of our text encoder **without Second Stage**, by removing u_l^2 and u_r^2 from Eq. 6. In one more complementary experiment, we try our textual encoding with C3D and VGG19 features without any attentions. These experiments show that all the components contribute to final results.

3.2. Multiple Blanks VFIB

In this section, we explore a harder version of the VFIB problem where more than one word is missing from the sentence. We have generated a new dataset based on the original LSMDC Movie Dataset by inserting multiple blanks in the sentences, and we call it the "Extended LSMDC Movie Dataset". To be specific, we remove all the words which have appeared at least once as a blank in the original LSMDC dataset from all the sentence. In this case, most of sentences have more than one blank and this makes the LSMDC dataset suitable to be extended for multiple blanks problem. In Figure 6, we show some statistics about the number of blanks in sentences. About 79.3% of the sentences have more than one blank. To clarify, in each sentence, there are known number of blanks (with known locations), but there are various number of blanks in different sentences. For multiple blanks' experiments, we include all the sentences with one or more blanks in all sets and also for the evaluation, we consider equal value for all the blanks.

We employ two strategies to encode the source sentence for the multiple blanks VFIB problem. For the first one, we consider the left and right fragments of a missing word, as a fragment from that word to the next blank, or if there is no other blank, to the end of the sentence (both left and right fragments are used). For example, for the source sentence *"She took her Blank1 out of the garage and Blank2 at the house for a moment."*. The left phrase of "Blank1" is *"She took her"* and right phrase is *"out of the garage and"*

Method	Accuracy
Baselines	
Random Guess	0.006
Left LSTM (Masking)	0.104
Bi-LSTM (Masking)	0.156
2Videos + Textual (Subdivision) [14]	0.136
2Videos + Textual (Masking) [14]	0.177
Ours	
Text Only (Subdivision)	0.136
Text + Video (Subdivision)	0.148
Text Only (Masking)	0.180
Text + Video (Masking)	**0.201**

Table 2. Results on the LSMDC Dataset (Multiple Blanks). Our method has superior results.

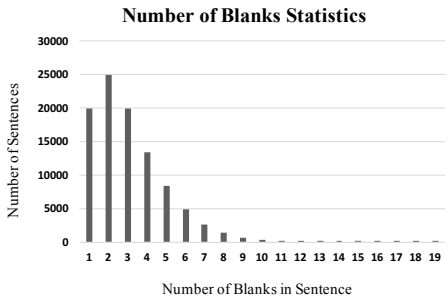

Figure 6. Number of sentences as a function of the number of blanks in training set of LSMDC.

and we find the left and right phrases for "Blank2" with the same approach as well. We call it the **"Subdivision"** approach since it makes multiple fragments out of the source sentence and each blank has one left and one right fragment. The second approach is to remove all other blanks and treat them as left and right fragments as normal. In our example, the left fragment of "Blank2" is *"She took her out of the garage"* and the right fragment of the "Blank2" is *"out of the garage and at the house for a moment"*. In this case, we deal with each blank similar to single blank problem and we just ignore other blanks in each of left and right fragments. We call it **"Masking"** approach since we are masking the other missing words from each fragment. After finding left and right fragments based on any of these approaches, we can apply our method or any other baselines (Table 2).

3.3. Qualitative Results

In Fig. 7, we show some qualitative results. For generating the attention map, we have reshaped the $\mathbf{p}_{sp} \in \mathbb{R}^{m=196}$ in Eq. 9 into a 14×14 matrix, then up-sampled it back to the original frames size. We smooth the attention map by a Gaussian filter and also suppress low intensity pixels to be zero. Brighter parts have higher attention score than

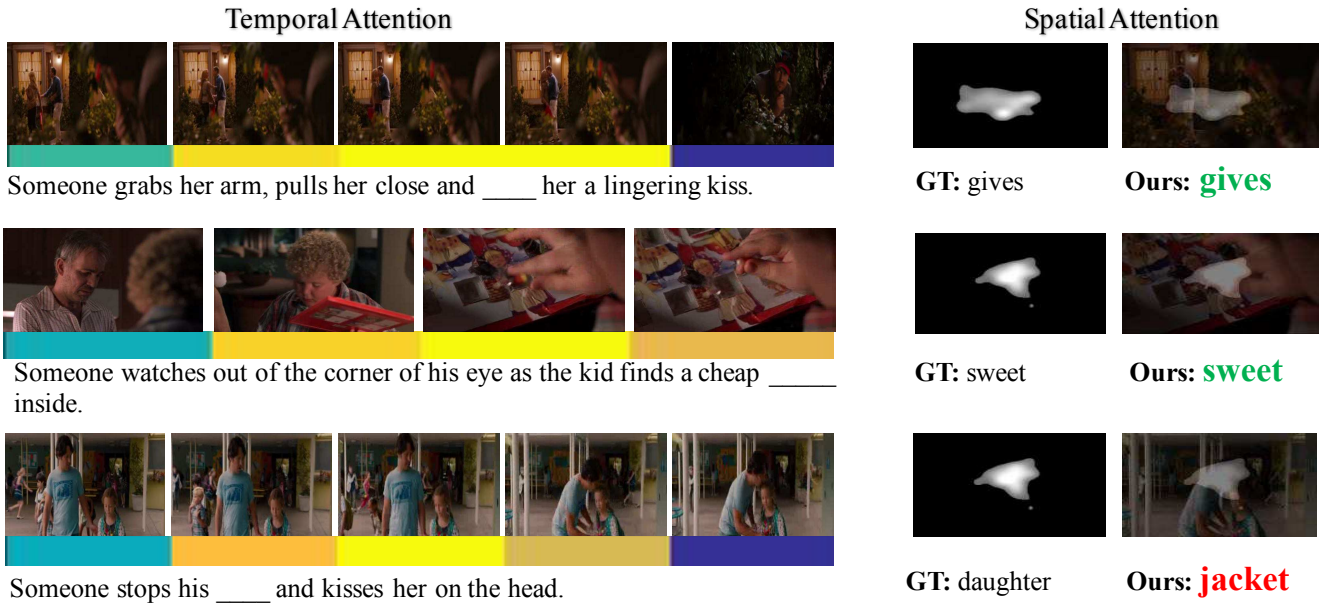

Figure 7. On the left we show representative frames from different shots. The colors below the frames show the temporal attention: yellow/blue means the most/least attention. On the right, we show an spatial attention map obtained by our method and also we show the attention map on one of selected key-frames.

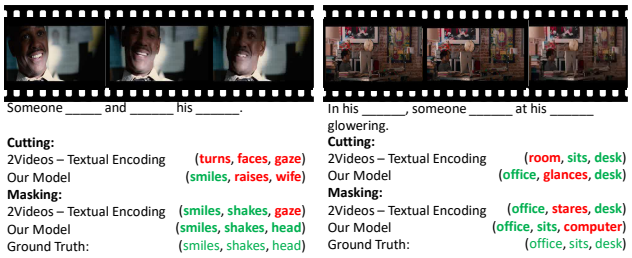

Figure 8. Examples for Multiple Blank VFIB problem which requires a higher level of video and text alignment to find all the missing words correctly.

the darker parts. For the temporal attention model, we extract the $\mathbf{p}_{tp}$ vector as the temporal attention. In Fig. 7, we show one frame from each shot and the color bar under the sequence of shots shows the attention scores. Yellow and blue respectively represent the maximum and minimum attentions. In Fig. 8 we provide examples of multiple blanks VFIB and predicted missing words using different methods.

3.4. Implementation Details

We use VGG-19 [37] network, pre-trained on ImageNet [5], and extract last pooling layer ("pool5") as our spatial visual features consumed in section 2.2. The output feature map is a $14 \times 14 \times 512$ matrix which can be reshaped as a 196×512 matrix and each of 512 dimensional vectors are representing a 32×32 pixels region of input frame. We believe any other very deep CNN network like GoogLeNet [3] or ResNet [4] can produce similar results. We extract and pass the frames through this network with 2fps rate. For temporal attention in section 2.3, we use pre-trained 3D CNN (C3D) network [32] pre-trained on [38] and followed settings defined in [32]. We extract the "fc6" output of the network for each 16 frames (one shot) of videos. We assume each video has 10 shots. For shorter videos, we use all-zero vectors for remaining shots and for longer ones we uniformly select 10 shots.

4. Conclusion

We proposed a new method for the Video-Fill-in-the-Blank (VFIB) problem which leverages the "source sentence" structure and also spatial-temporal attention models. We have introduced "external memory" to deal with the complexity of "source sentence" in VFIB problem. We have achieved superior performance over all other reported methods. Also, an extension and more general version of VFIB which deals with multiple blanks in sentences, is introduced and discussed.

Acknowledgements. The first author of this article was supported under a Nielsen Fellowship; we want to thank them for their generous support.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012. 1

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014. 1, 4

[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *CVPR*, 2015. 1, 6, 8

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015. 1, 8

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009. 1, 8

[6] K. Soomro, A. R. Zamir, and M. Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild," *arXiv preprint arXiv:1212.0402*, 2012. 1

[7] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, 1997. 1, 2, 3, 6

[8] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, 1997. 1

[9] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014. 1

[10] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *ICML*, 2015. 1

[11] S. Venugopalan, H. Xu, J. Donahue, M. Rohrbach, R. Mooney, and K. Saenko, "Translating videos to natural language using deep recurrent neural networks," *arXiv preprint arXiv:1412.4729*, 2014. 1

[12] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *CVPR*, 2015. 1

[13] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. Lawrence Zitnick, and D. Parikh, "Vqa: Visual question answering," in *ICCV*, 2015. 1

[14] M. Ren, R. Kiros, and R. Zemel, "Exploring models and data for image question answering," in *NIPS*, 2015. 1, 2, 6, 7

[15] M. Malinowski and M. Fritz, "A multi-world approach to question answering about real-world scenes based on uncertain input," in *NIPS*, 2014. 1, 2

[16] A. Agrawal, J. Lu, S. Antol, M. Mitchell, C. L. Zitnick, D. Batra, and D. Parikh, "Vqa: Visual question answering," *arXiv preprint arXiv:1505.00468*, 2015. 1

[17] C. Xiong, S. Merity, and R. Socher, "Dynamic memory networks for visual and textual question answering," *arXiv preprint arXiv:1603.01417*, 2016. 1, 2

[18] P. Zhang, Y. Goyal, D. Summers-Stay, D. Batra, and D. Parikh, "Yin and yang: Balancing and answering binary visual questions," *arXiv preprint arXiv:1511.05099*, 2015. 1, 2

[19] A. Rohrbach, M. Rohrbach, N. Tandon, and B. Schiele, "A dataset for movie description," in *CVPR*, 2015. 1, 6

[20] J. Johnson, A. Karpathy, and L. Fei-Fei, "Densecap: Fully convolutional localization networks for dense captioning," *arXiv preprint arXiv:1511.07571*, 2015. 1

[21] A. Bordes, N. Usunier, S. Chopra, and J. Weston, "Large-scale simple question answering with memory networks," *arXiv preprint arXiv:1506.02075*, 2015. 1

[22] A. Kumar, O. Irsoy, J. Su, J. Bradbury, R. English, B. Pierce, P. Ondruska, I. Gulrajani, and R. Socher, "Ask me anything: Dynamic memory networks for natural language processing," *arXiv preprint arXiv:1506.07285*, 2015. 1, 2

[23] J. Weston, S. Chopra, and A. Bordes, "Memory networks," *arXiv preprint arXiv:1410.3916*, 2014. 1

[24] M. Malinowski, M. Rohrbach, and M. Fritz, "Ask your neurons: A neural-based approach to answering questions about images," in *CVPR*, 2015. 2, 6, 7

[25] F. Sadeghi, S. K. Divvala, and A. Farhadi, "Viske: Visual knowledge extraction and question answering by visual verification of relation phrases," in *CVPR*, 2015. 2

[26] M. Tapaswi, Y. Zhu, R. Stiefelhagen, A. Torralba, R. Urtasun, and S. Fidler, "Movieqa: Understanding stories in movies through question-answering," in *CVPR*, 2016. 2

[27] T. Maharaj, N. Ballas, A. Courville, and C. Pal, "A dataset and exploration of models for understanding video data through fill-in-the-blank question-answering," *arXiv preprint arXiv:1611.07810*, 2016. 2, 6

[28] M. Berglund, T. Raiko, M. Honkala, L. Kärkkäinen, A. Vetek, and J. T. Karhunen, "Bidirectional recurrent neural networks as generative models," in *NIPS*, 2015. 2, 3

[29] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, "Multimodal deep learning," in *ICML*, 2011. 3

[30] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," in *CVPR*, 2016. 3

[31] Z. Yang, X. He, J. Gao, L. Deng, and A. Smola, "Stacked attention networks for image question answering," *arXiv preprint arXiv:1511.02274*, 2015. 4, 5

[32] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *ICCV*, 2015. 5, 6, 8

[33] A. Torabi, C. Pal, H. Larochelle, and A. Courville, "Using descriptive video services to create a large data source for video annotation research," *arXiv preprint*, 2015. 6

[34] A. Rohrbach, A. Torabi, T. Maharaj, M. Rohrbach, C. Pal, A. Courville, and B. Schiele, "Movie fill-in-the-blank dataset," 2016. 6

[35] Y. Yu, H. Ko, J. Choi, and G. Kim, "Video captioning and retrieval models with semantic attention," *arXiv preprint arXiv:1610.02947*, 2016. 6, 7

[36] D. Opitz and R. Maclin, "Popular ensemble methods: An empirical study," *Journal of Artificial Intelligence Research*, vol. 11, 1999. 7

[37] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014. 8

[38] 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. 8