L-STAP: Learned Spatio-Temporal Adaptive Pooling for Video Captioning

Danny Francis
danny.francis@eurecom.fr
EURECOM
Biot, France

Benoit Huet
benoit.huet@eurecom.fr
EURECOM
Biot, France

ABSTRACT

Automatic video captioning can be used to enrich TV programs with textual informations on scenes. These informations can be useful for visually impaired people, but can also be used to enhance indexing and research of TV records. Video captioning can be seen as being more challenging than image captioning. In both cases, we have to tackle a challenging task where a visual object has to be analyzed, and translated into a textual description in natural language. However, analyzing videos requires not only to parse still images, but also to draw correspondences through time. Recent works in video captioning have intended to deal with these issues by separating spatial and temporal analysis of videos. In this paper, we propose a Learned Spatio-Temporal Adaptive Pooling (L-STAP) method that combines spatial and temporal analysis. More specifically, we first process a video frame-by-frame through a Convolutional Neural Network. Then, instead of applying an average pooling operation to reduce dimensionality, we apply our L-STAP, which attends to specific regions in a given frame based on what appeared in previous frames. Experiments on MSVD and MSR-VTT datasets show that our method outperforms state-of-the-art methods on the video captioning task in terms of several evaluation metrics.

CCS CONCEPTS

• **Information systems → Content analysis and feature selection**; • **Computing methodologies → Natural language generation**; • **Applied computing → Annotation**.

KEYWORDS

deep learning, neural networks, video captioning

ACM Reference Format:
Danny Francis and Benoit Huet. 2019. L-STAP: Learned Spatio-Temporal Adaptive Pooling for Video Captioning. In *1st International Workshop on AI for Smart TV Content Production, Access and Delivery (AI4TV '19), October 21, 2019, Nice, France.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3347449.3357484

1 INTRODUCTION

Automatic video captioning can be used to enrich TV programs with textual informations on scenes. These informations can be useful for visually impaired people, but can also be used to enhance indexing and research of TV records. The video captioning task consists in automatically generating short textual descriptions for videos. It is a challenging multimedia task as it requires to grasp all information contained in a video, such as objects, persons, context, actions, location, and to translate this information into text. This task can be compared to a translation task: except instead of translating a sequence of words in a source language into a sequence of words in a target language, the aim is to translate a sequence of frames into a sequence of words. Therefore, most of recent works in video captioning rely on the encoder-decoder framework proposed in [25], initially for text translation. In video captioning, the encoder aims at deriving a video representation. Recent advances in deep learning have shown to fit very well to that task. In particular, Convolutional Neural Networks (CNNs) have proved to give excellent results in producing highly descriptive image representations or video representations. The decoder part aims at generating a sentence based on the representation produced by the encoder. Long Short-Term Units (LSTMs) [12] and Gated Recurrent Units (GRUs) [5] are usually chosen for that task. Image captioning [30] and video captioning can seem to be similar tasks, as both of them require to "translate" a visual object into a textual one. However, video captioning poses a problem that makes it more challenging than image captioning: it requires to take into account temporality.

In [18], authors showed that for text translation tasks based on the encoder-decoder framework, results could be improved if the decoder attended to hidden states of the encoder based on its hidden states. Some other works showed that the same attention mechanisms could be applied to video captioning [10, 31, 38, 39]. The improvement induced by that attention mechanism can be interpreted as follows: when the decoder is predicting the next word of a sentence, it attends to relevant frames to perform that task accurately. Some other works have also shown that attending to relevant regions in a video during the encoding phase could lead to better representations of videos, and thus better results [32, 40]. However these works attend to local regions based on frame-level considerations, without taking into account previous frames. In our work, we aim at attending to relevant regions of a video based on previous frames, because the relevance of objects, persons or actions relies on the context in which they appear; and that context should be inferred from previous frames. More precisely, after deriving frame-level local features using the last convolutional layer of a ResNet-152 [11], we do not apply an average pooling to pool these local features. We process them by Learned

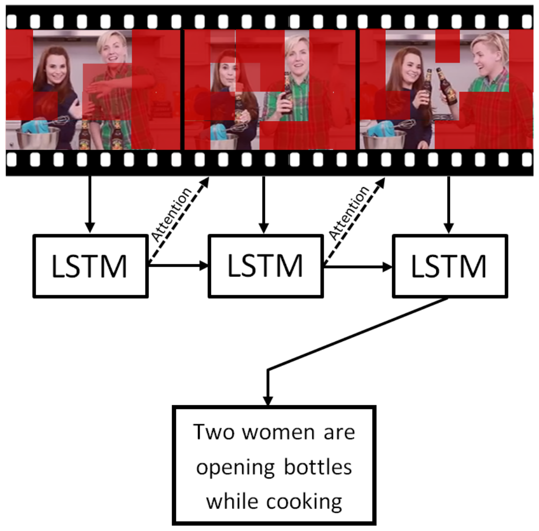


Figure 1: Overview of our L-STAP method. Frame-level local features are derived using a ResNet-152. Then, an LSTM processes these local features, and updates its hidden state by attending to them based on previous frames. The result is that space and time are jointly taken into account to build video representations.

Spatio-Temporal Adaptive Pooling (L-STAP). L-STAP attends to specific regions of a frame based on what occurred previously in the video. Our pooling method is learned because it is based on an LSTM whose parameters are learned. It is spatio-temporal because it takes into account space and time in a joint fashion. In addition, it is adaptive because the attention paid to local regions is based on previous hidden states of the LSTM; pooling depends not only on the processed frame but also on previous ones. A high-level schematic view of our proposed model is depicted in Figure 1.

We evaluated our results on two common datasets used for benchmarking video captioning tasks: MSVD [4] and MSR-VTT [36]. Results show that our model based on L-STAP outperforms state-of-the-art models in terms of several metrics. An ablation study also shows that our method leads to significant improvements with respect to state-of-the-art methods.

Our contributions can be summarized as follows: we propose a novel pooling method for video processing, which we evaluate on the video captioning task, even though it could be applied to any other task involving video processing, such as video classification. Moreover, we demonstrate the interest of our pooling method over usual approaches. The paper is organized as follows. In Section 2, we introduce previous works on video captioning. In Section 3, we present our model based on L-STAP. Section 4 is dedicated to experiments. We conclude the paper in Section 5.

2 RELATED WORK

Video captioning can be seen as a translation task: a sequence of frames, which can be compared to a sequence of words in a source language, have to be translated in a target language. Some pioneering works such as [23] make use of Statistical Machine Translation techniques to generate captions from videos. However nowadays, most of recent works on video captioning rely on Deep Learning techniques, and more particularly on the encoder-decoder framework that has been developed in [25] for text translation [8]. Moreover, attending to the hidden states of the encoder during the decoding phase has shown to give significant improvements in Neural Machine Translation [18], which have been confirmed in by [38] in the context of video captioning.

In some works, videos are split into frames, global features are derived for each frame using a CNN [11, 13, 24, 26], and the obtained features vectors are sequentially processed by the encoder [10, 14, 17, 19, 31, 34]. The drawback in such approaches is that spatial information is lost. In our approach, we aim at taking into account this spatial information.

Other approaches take into account locality. However, these approaches have some significant differences with our approach. In [38], the authors separate their model into two parts: a usual encoder-decoder based on global features of frames, and a 3D-CNN that derives a single representation for a whole video. The 3D-CNN they employ does take into account locality, but it has two major conceptual differences with respect to our method. First, it is based on handcrafted features, which do not provide as much semantic information as CNN features. Moreover, the pooling operations that are used to get their video representations are neither learned nor adaptive. In our approach, pooling takes into account the relevance of local features in a frame with respect to previous frames. In [39], authors use local features to trace semantic concepts along videos, which is conceptually different from our approach, as we aim to derive a video representation based on these local features. In [35], authors propose another method to compute trajectories through videos. In both papers, these trajectories are combined with global features to build video representations. In [32], local features are used to generate video representations. However, local features from different spatial locations are not related together, contrary to our work, which proposes to attend to local features based on all local features from previous frames. Eventually, some other works used 3D-CNN architectures [33] or convolutional RNNs [32] to relate local features through time. However, due to the nature of convolution operations, relations drawn through these methods remain local: they are not able to spatially relate objects from the video which are far from each other in a video for instance. Our method, as we will show in the following, is build to grasp jointly spatial and temporal information, by attending to relevant locations of a frame with respect to previous ones.

3 PROPOSED MODEL

Let us first formulate the problem we are to deal with. Given a video V, which is a sequence of T frames $(v^{(1)}, ..., v^{(T)})$, our goal is to derive a descriptive sentence $Y = (y_1, ..., y_L)$. The approach that we have followed is based on the encoder-decoder framework. The encoder first derives frame-level representations $(x^{(1)}, ..., x^{(T)}) = X$, and then pool these representations together to form frame-level video representations $(\overline{h}^{(1)}, ..., \overline{h}^{(T)})$. Based on these representations, the decoder reconstructs a descriptive sentence in a recurrent fashion. Figure 2 summarizes the important steps our model. In the

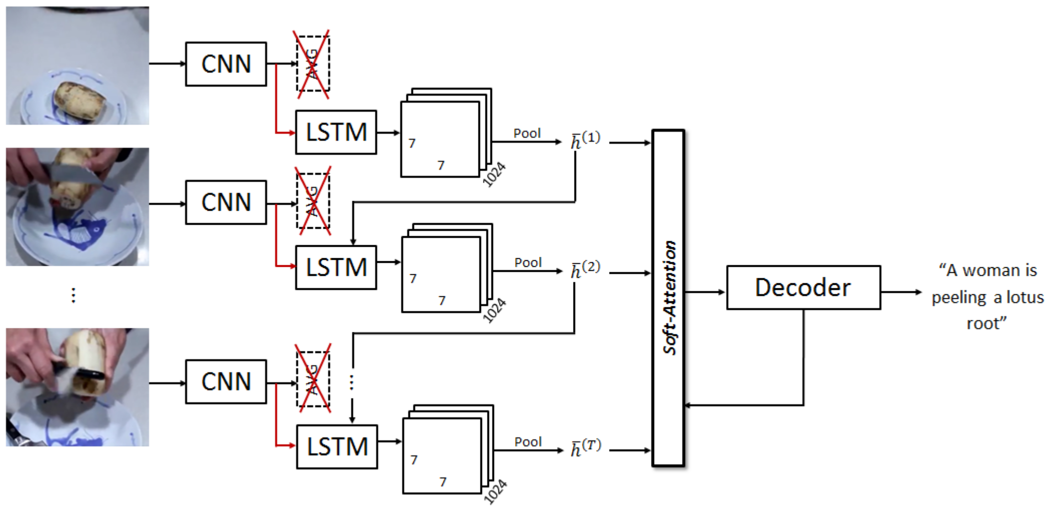


Figure 2: Illustration of our model, based on the proposed L-STAP method. Frames are processed sequentially by a CNN (a ResNet-152 in this case). However, instead of applying an average pooling on local features as some recent works do, we make use of an LSTM to capture time dependencies. Local hidden states are computed to obtain a 7x7x1024-dimensional tensor. These local hidden states are then pooled together (using average pooling or soft attention), and processed by an LSTM decoder to output a sentence.

following section, we will describe it in detail and report how we train it.

3.1 Grasping Spatio-Temporal Dependencies with L-STAP

As we stated above, the first step is to produce a representation of the input video. In the following subsections, we will explain how we derive frame-level features, and how we pool them together.

3.1.1 Image-Level Features. Given a video $V = (v^{(1)}, ..., v^{(T)})$, we need to derive features for each frame $v^{(t)}$. A common way to do so is to process each frame using a CNN, which has been previously pretrained on a large-scale dataset. In works such as [17], the outputs of the penultimate layer of a ResNet-152 have been chosen as frames representations, which consist of 2048-dimensional vectors. However, such representations discard locality, which results in loss of information. Therefore, in this work, we choose to take the output of the last convolutional layer of a ResNet-152. Thus, we obtain frame-level representations $(x^{(1)}, ..., x^{(T)}) = X$, where $x^{(t)} \in \mathbb{R}^{7\times7\times2048}$ for all t. The next step is to process these dense frame-level representations to derive compact frame-level representations, using the proposed L-STAP method instead of conventional pooling.

3.1.2 How L-STAP Works. L-STAP aims at replacing the average pooling operation after the last convolutional layer in a CNN, and to pool local features according to previous frames. The goal is to capture where important actions are occurring, and to discard locations that are not relevant to summarize what is happening in a video. For that purpose, we use an LSTM, taking local features as inputs, resulting in local hidden states, which are then combined in a way we will describe later in this subsection. More formally, given local features $x_{ij}^{(t)} \in \mathbb{R}^{2048}$, the aggregated local features $h_{ij}^{(t)}$ are computed recursively as follows:

$$i_{ij}^{(t)} = \sigma(W_{ix}x_{ij}^{(t)} + W_{ih}\bar{h}^{(t-1)} + b_i) \tag{1}$$

$$f_{ij}^{(t)} = \sigma(W_{fx}x_{ij}^{(t)} + W_{fh}\bar{h}^{(t-1)} + b_f) \tag{2}$$

$$o_{ij}^{(t)} = \sigma(W_{ox}x_{ij}^{(t)} + W_{oh}\bar{h}^{(t-1)} + b_o) \tag{3}$$

$$c_{ij}^{(t)} = f_{ij}^{(t)} \circ \bar{c}^{(t-1)} + i_{ij}^{(t)} \tanh(W_{cx}x_{ij}^{(t)} + W_{ch}\bar{h}^{(t-1)} + b_c) \tag{4}$$

$$h_{ij}^{(t)} = o_{ij}^{(t)} \circ \tanh(c_{ij}^{(t)}) \tag{5}$$

where W_{ix}, W_{ih}, b_i, W_{fx}, W_{fh}, b_f, W_{ox}, W_{oh}, b_o, W_{cx}, W_{ch} and b_c are trainable parameters, and $\bar{c}^{(t-1)}$ and $\bar{h}^{(t-1)}$ are respectively the memory cell and the hidden state of the LSTM. Please note that memory cells and hidden states are shared for computing all aggregated local features. The memory cell and the hidden state at time t are computed as follows:

$$\bar{c}^{(t)} = \sum_{i=1}^{7}\sum_{j=1}^{7} \alpha_{ij}^{(t)} c_{ij}^{(t)} \tag{6}$$

$$\bar{h}^{(t)} = \sum_{i=1}^{7}\sum_{j=1}^{7} \alpha_{ij}^{(t)} h_{ij}^{(t)} \tag{7}$$

where $\alpha_{ij}^{(t)}$ are local weights. In our work, we experimented with two types of local weights. We first tried to use uniform weights:

$$\alpha_{ij}^{(t)} = \frac{1}{7\times7} \tag{8}$$

which actually correspond to an average pooling of aggregated local features. The second solution that we tried was to derive local weights using an attention mechanism, as follows:

$$\tilde{\alpha}_{ij}^{(t)} = w^T \tanh(W_{\alpha x} x_{ij}^{(t)} + W_{\alpha h} \overline{h}^{(t-1)} + b_\alpha). \tag{9}$$

$$\alpha_{ij}^{(t)} = \frac{\exp(\tilde{\alpha}_{ij}^{(t)})}{\sum_{k=1}^{7} \sum_{l=1}^{7} \exp(\tilde{\alpha}_{kl}^{(t)})}, \tag{10}$$

where $W_{\alpha x}$, $W_{\alpha h}$, b_α are trainable parameters.

3.2 Encoding Videos

In our model, we encode videos using the L-STAP method we presented previously. We initialized the memory cell and the hidden state of the LSTM using the output of an I3D [3] (before the final softmax) which had been trained on Kinetics-600 [3]. More formally, if V is an input video:

$$c_{ij}^{(0)} = \tanh(W_c^{\mathrm{e}} \mathrm{e}(V) + b_c^{\mathrm{e}}) \tag{11}$$

$$h_{ij}^{(0)} = \tanh(W_h^{\mathrm{e}} \mathrm{e}(V) + b_h^{\mathrm{e}}) \tag{12}$$

where W_c^{e}, b_c^{e}, W_h^{e} and b_h^{e} are trainable parameters. The decoder produces $\overline{c}^{(T)}$ and $\overline{h}^{(T)}$ as outputs, where T is the length of the input video. These outputs will be used to initialize the sentence decoder that we will introduce in the next section.

3.3 Decoding Sentences

For decoding sentences, we chose to use an LSTM. In the following, we assume that sentences Y are represented by sequences of one-hot vectors $y_1, ..., y_L \in \mathbb{R}^N$ where N is the vocabulary size. The aim of the LSTM is to compute the probabilities $\mathrm{P}(y_l | y_{l-1}, ..., y_1, V; \theta)$ for $l \in \{1, ..., L\}$, where θ is the set of all parameters in the encoder and the decoder, and V an input video. In the following, we will describe formally how we compute these probabilities.

We initialize the memory cell and the hidden state of the decoder LSTM using the last memory cell and the last hidden state of the decoder:

$$c_0^d = \overline{c}^{(T)}, \tag{13}$$

$$h_0^d = \overline{h}^{(T)}. \tag{14}$$

It has been shown in [18] for text translation tasks that attending to hidden states of the encoder during the decoding phase improved results. Some works in video captioning have followed that approach successfully [37, 38]. We followed a similar approach for our decoding phase. More precisely, at each step l, we compute a weighted sum of hidden states of the encoder:

$$\varphi(\overline{h}, h_{l-1}^d) = \sum_{t=1}^{T} \beta_l^{(t)} \overline{h}^{(t)} \tag{15}$$

where $\beta_l^{(1)}, ..., \beta_l^{(T)}$ are computed as follows:

$$\tilde{\beta}_l^{(t)} = w_\beta^T \tanh(W_{\beta e} \overline{h}^{(t)} + W_{\beta h} h_{l-1}^d + b_\beta), \tag{16}$$

$$\beta_l^{(t)} = \frac{\exp(\tilde{\beta}_l^{(t)})}{\sum_{k=1}^{L} \exp(\tilde{\beta}_k^{(t)})}, \tag{17}$$

where $W_{\beta e}$, $W_{\beta h}$, b_β are trainable parameters. Assuming that the word y_{l-1} has been decoded at step $l-1$, we aim to decode y_l based on y_{l-1} and $\varphi(\overline{h}, h_{l-1}^d)$. For that purpose, we first compute a word embedding x_l^d:

$$w_l^d = W_{\mathrm{emb}} y_{l-1}, \tag{18}$$

where W_{emb} is a learned embedding matrix. Then, we concatenate w_l^d and $\varphi(\overline{h}, h_{l-1}^d)$ to obtain and x_l^d:

$$x_l^d = [w_l^d; \varphi(\overline{h}, h_{l-1}^d)] \tag{19}$$

Eventually, we input x_l^d to the decoder LSTM:

$$i_l^d = \sigma(W_{ix}^d x_l^d + W_{ih}^d h_{l-1}^d + b_i^d) \tag{20}$$

$$f_l^d = \sigma(W_{fx}^d x_l^d + W_{fh}^d h_{l-1}^d + b_f^d) \tag{21}$$

$$o_l^d = \sigma(W_{ox}^d x_l^d + W_{oh}^d h_{l-1}^d + b_o^d) \tag{22}$$

$$c_l^d = f_l^d \circ c_{l-1}^d + i_l^d \tanh(W_{cx}^d x_l^d + W_{ch}^d h_{l-1}^d + b_c^d) \tag{23}$$

$$h_l^d = o_l^d \circ \tanh(c_l^d) \tag{24}$$

where W_{ix}^d, W_{ih}^d, b_i^d, W_{fx}^d, W_{fh}^d, b_f^d, W_{ox}^d, W_{oh}^d, b_o^d, W_{cx}^d, W_{ch}^d and b_c^d are trainable parameters.

The last step is to infer a word y_l. For that purpose, we derive $\tilde{y}_l$ as follows:

$$\tilde{y}_l = \mathrm{softmax}(W_d h_l^d) \tag{25}$$

where W_d is a trainable parameter. We state that y_l is the one-hot vector corresponding to the maximum coordinate of $\tilde{y}_l$.

3.4 Training

Assuming that $y_1, ..., y_L$ correspond to ground-truth words, we aim to minimize the following cross-entropy loss:

$$\mathcal{L}_d(\theta) = -\sum_{l=1}^{L} \log \mathrm{P}(\tilde{y}_l | y_{l-1}, ..., y_1, V; \theta) \tag{26}$$

where V is a video corresponding to the caption $(y_1, ..., y_L)$.

In addition to that, some works have shown that regularizing the cross-entropy loss with a matching loss between video encodings and ground-truth sentences could improve results by bridging the semantic gap between them [10, 17]. As reported in Section 4.4, such improvement has been noticed in our experiments. The matching model we employed is described in the following. Let us assume that $Y = (y_1, ..., y_L)$ is a sentence corresponding to a video V. First, we translate this sequence of one-hot vectors into a sequence of word embeddings $(x_1^s, ..., x_L^s)$ using the matrix W_{emb} from Section 3.3. Then, we compute a sentence embedding $\psi(Y)$ by processing this sequence of word embeddings into another LSTM: each word embedding is entered sequentially as an input to that LSTM, and $\psi(Y)$ is defined to be its last hidden state. We want the initialization of the decoder to be as close as possible to an accurate representation of its corresponding sentence. Therefore, if

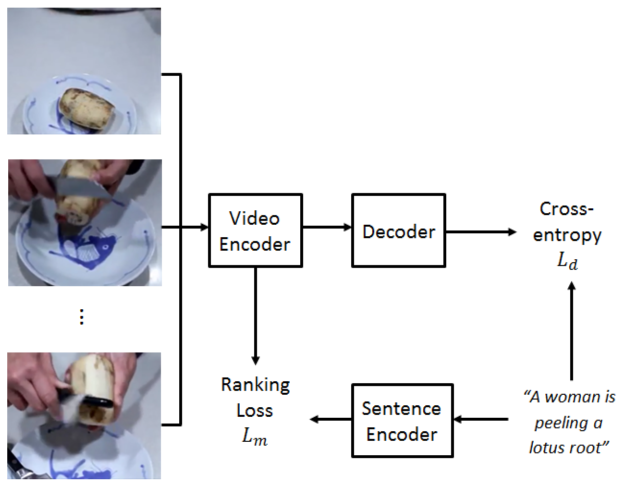

Video Encoder
Decoder
Cross-entropy $\mathcal{L}_d$
Ranking Loss $\mathcal{L}_m$
Sentence Encoder
"A woman is peeling a lotus root"

Figure 3: Overview of our training losses. The first training loss is the Cross-Entropy loss, which aims to make the probability distribution of sentences in the training set and the probability distribution of the inferred sentences match. The second one is a ranking loss, aiming to bridge the semantic gap between video representations and sentences.

$\varphi(V) = \overline{h}^{(T)}$ is the initial hidden state of the decoder, we will aim to minimize the following ranking loss from [9]:

$$\mathcal{L}_m(\theta) = \max_{\overline{V} \neq V} \left(\max(0, \alpha - S(\varphi(V), \psi(Y)) + S(\varphi(\overline{V}), \psi(Y))) \right) + \max_{\overline{Y} \neq Y} \left(\max(0, \alpha - S(\varphi(V), \psi(Y)) + S(\varphi(V), \psi(\overline{Y}))) \right) \tag{27}$$

where $\overline{V}$ is a negative video sample, and $\overline{Y}$ is a negative sentence sample coming from another video than V. The final loss is the following:

$$\mathcal{L}(\theta) = \mathcal{L}_d(\theta) + \lambda \mathcal{L}_m(\theta) \tag{28}$$

where λ is a hyperparameter that we set to 0.4 according to results on validation.

4 EXPERIMENTS

4.1 Datasets

We evaluated our models on two video captioning datasets: MSVD [4] and MSR-VTT [36]. MSVD is a dataset composed of 1,970 videos from YouTube, which have been annotated by Amazon Mechanical Turks (AMT). Each video has approximately 40 captions in English. We split that dataset following [29]: 1,200 videos for training, 100 videos for validation and 670 videos for testing. MSR-VTT is a similar dataset, but with much more videos, and less captions per video. It is composed of 10,000 videos, and 20 captions per video. Following [36], we split that dataset into 6,513 videos for training, 497 videos for validation and 2,990 videos for testing.

For both datasets, we uniformly sampled 30 frames per video as done in [40], and extracted features for each frame based on the last convolutional layer of a ResNet-152 [11], which had been trained on the image-text matching task on MSCOCO [16], after pre-training on ImageNet-1000 [6] following [9]. In addition, we extracted activity features for each video using an I3D pretrained on Kinetics-600 [3]. For MSVD, we converted sentences to lowercase and removed special characters, which lead to a vocabulary of about 14k words. We converted each word into an integer, and cut sentences after the thirtieth word if their lengths were higher than thirty. The same approach for MSR-VTT lead to a much bigger vocabulary size of about 29k words. Therefore, we kept only the 15k most common words, and replaced all the others by an <UNK> token. We applied the same process otherwise.

4.2 Implementation Details

Our models have been implemented with the TensorFlow framework [1]. We use 1024-dimensional LSTMs in both encoder and decoder. Soft attention spaces are 256-dimensional. Word embeddings are 300-dimensional.

We trained our model using the RMSProp algorithm [27], with decay = 0.9, momentum = 0.0 and epsilon = 1e-10. Batch size is set to 64. Learning rate is 1e-4, and we apply gradient clipping to a threshold of 5. Eventually, we apply dropout on the output of the decoder (before the prediction layer) with a rate of 0.5 to avoid overfitting.

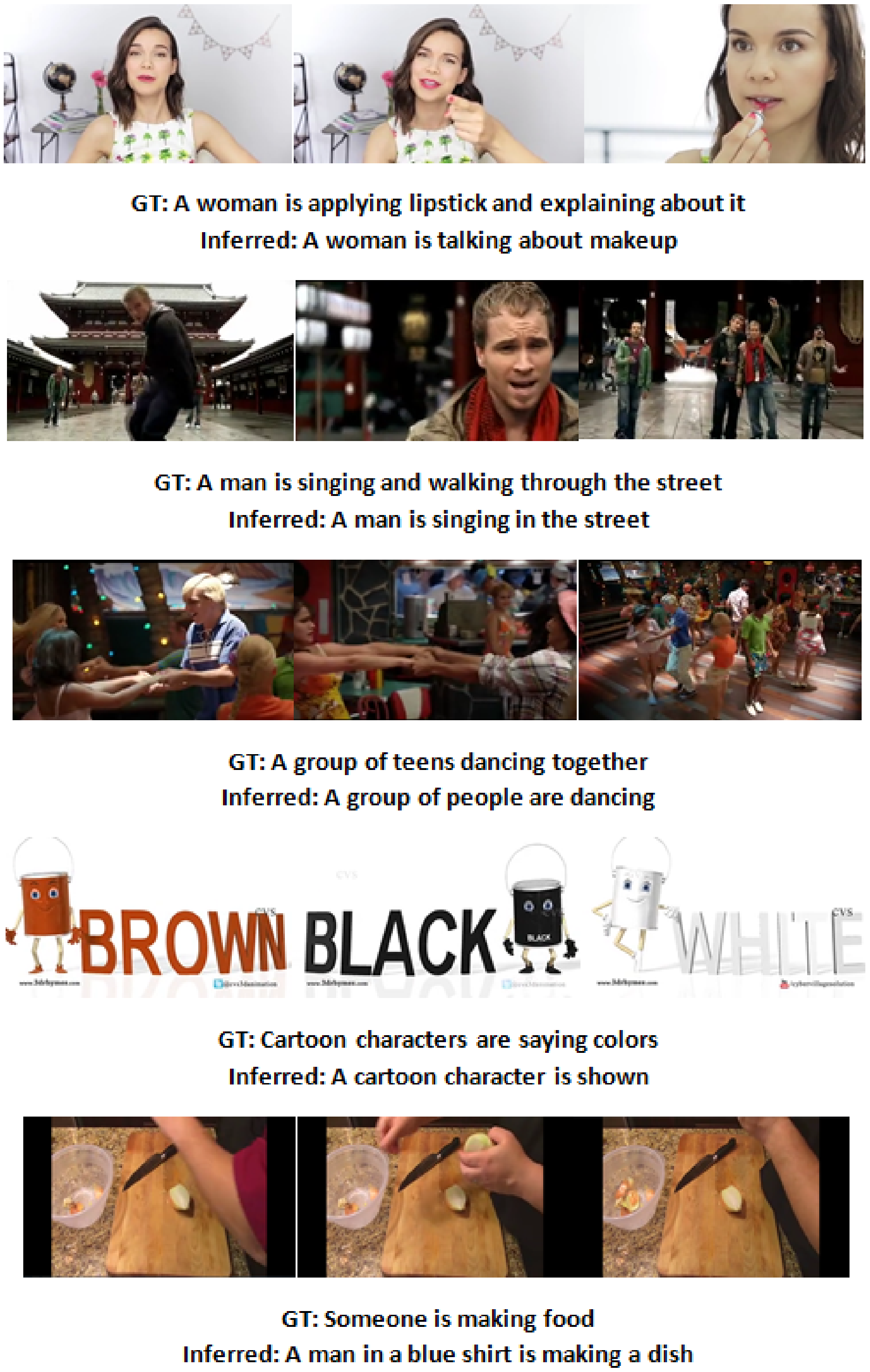
4.3 Results on MSVD and MSR-VTT

We evaluated our models in terms of BLEU [20], ROUGE [15], METEOR [7] and CIDEr [28] scores, which are metrics commonly used to evaluate automated captioning tasks. We compared them to the following recent models for video captioning. Our results on MSVD are presented in Table 1. Results on MSR-VTT are presented in Table 2.

On MSVD, it can be noticed that L-STAP achieves the best results on six out of seven metrics. It is also relevant to mention that E2E [14], which achieves better CIDEr results than our model, has been trained using reinforcement learning techniques to be optimized regarding that CIDEr metric. Works on image captioning and video captioning have shown that significant improvements could be done using such techniques [2, 22, 34], at the price of much longer training times. We did not use reinforcement learning to train our models, instead we use cross-entropy minimization which has the advantage of being fast and simpler to implement.

Results on MSR-VTT show that our model outperforms models trained using a cross-entropy loss on two metrics out of four (METEOR and ROUGE). HRL [34] obtains better results overall, however it makes use of reinforcement learning techniques, which leads to better results as stated in the previous paragraph.

We report some qualitative results of our model on MSR-VTT in Figure 4. On the second video, the man who is singing appears during a very limited amount of time. This shows that our model has been able to attend to important frames to identify what the main action of the video was. In the first video, a woman starts talking about makeup, and then puts some lipstick on her lips. The caption generated by our model shows that it has been able to draw a relation between the first and the second parts of the video. Moreover, the lipstick is applied on a very localized part of the video frames: we can infer that our model could efficiently attend to the right part of the frame to generate a caption. The fourth video



GT: A woman is applying lipstick and explaining about it
Inferred: A woman is talking about makeup

GT: A man is singing and walking through the street
Inferred: A man is singing in the street

GT: A group of teens dancing together
Inferred: A group of people are dancing

GT: Cartoon characters are saying colors
Inferred: A cartoon character is shown

GT: Someone is making food
Inferred: A man in a blue shirt is making a dish

Figure 4: Some qualitative results of L-STAP on MSR-VTT.

Model	Bleu-1	Bleu-2	Bleu-3	Bleu-4	ROUGE	METEOR	CIDEr
TSL [35]	-	-	-	51.7	-	34.0	74.9
RecNet [31]	-	-	-	52.3	69.8	34.1	80.3
mGRU [21]	82.5	72.2	63.3	53.8	-	34.5	81.2
AGHA [40]	83.1	73.0	64.3	**55.1**	-	35.3	83.3
SAM [32]	-	-	-	54.0	-	35.3	87.4
E2E* [14]	-	-	-	50.3	70.8	34.1	87.5
SibNet [17]	-	-	-	54.2	71.7	34.8	**88.2**
L-STAP (Ours)	**84.0**	**74.1**	**64.5**	**55.1**	**72.7**	**35.4**	86.7

Table 1: Results on the MSVD dataset. The * sign means that the model is using reinforcement learning techniques to optimize over the CIDEr metric. Best results are in bold characters.

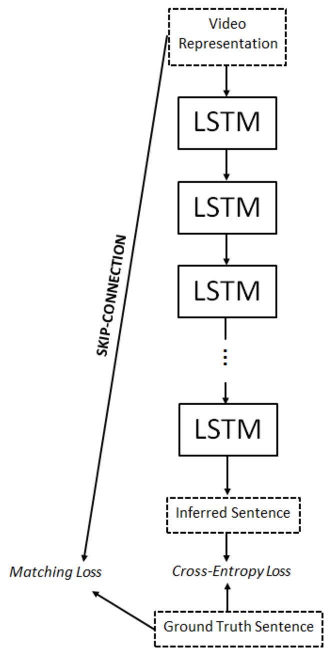


Figure 5: Our second interpretation about the efficiency of the second term of our loss function. Skip connections between video representations and ground-truth sentences improve results.

shows that results could be improved by adding sound processing to our model: it was not possible from the video only to know that colors were said.

4.4 Ablation Study

Results of an ablation study on the MSVD dataset are reported in Table 3. The encoder we used in our baseline model is an Long-term Recurrent Convolutional Network (LRCN) [8]. As shown in previous works such as [10, 17], adding a component to the training loss to make video representations match sentence representations improves results. Two interpretations can be given to these results. A first one is that adding a ranking loss to match video representations and sentence representations helps bridging the semantic gap between these two modalities. A second one could be that propagating the gradient across all the layers of the decoder could make it vanish through depth. Thus, adding a matching loss to the cross-entropy loss could be seen as a skip-connection between the sentence to be generated and the video representation used by the decoder. We illustrate that second interpretation in Figure 5.

Replacing the average pooling at the end of a CNN by our L-STAP induces a major improvement with respect to all metrics as reported in Table 3. On top of that, results shown in Table 1 demonstrate that L-STAP leads to better results than other models based on local features such as AGHA and SAM, and results shown in both Table 1 and Table 2 show the interest of L-STAP over average pooling.

We can notice in Table 3 that using a soft-attention mechanism to pool local hidden states in the encoder does not provide significant improvements over average pooling for all metrics except from CIDEr. Our interpretation is that the LSTM of the encoder can learn to attend to relevant local features by itself: before applying the average pooling, attention has already been drawn quite efficiently.

5 CONCLUSION

Video captioning is a way for TV broadcasters to enhance user experience, in particular regarding accessibility. In this paper, we presented a novel Learned Spatio-Temporal Adaptive Pooling (L-STAP) method for video captioning. It consists in taking into account spatial and temporal information jointly in a video to produce good video representations. As we have shown, these video representations can be successfully used to perform automated video captioning. We demonstrated the quality of our models based on L-STAP by comparing them with state-of-the-art models on MSVD and MSR-VTT, which are two video captioning datasets. On top of that, we assessed the interest of L-STAP through an ablation study. Although this paper concentrates on video captioning we believe that the proposed L-STAP method could be also applied to other video-related tasks such as video classification.

ACKNOWLEDGMENTS

This work was partially funded by ANR (the French National Research Agency) via the ANTRACT project and the European H2020 research and innovation programme via the project MeMAD (GA780069).

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al.

Model	Bleu-4	ROUGE	METEOR	CIDEr
RecNet [31]	39.1	59.3	26.6	42.7
E2E* [14]	40.1	61.0	27.0	48.3
SibNet [17]	40.9	60.2	27.5	47.5
HRL* [34]	**41.3**	**61.7**	**28.7**	**48.0**
L-STAP (Ours)	40.7	61.2	27.6	44.8

Table 2: Results on the MSR-VTT dataset. The * sign means that the model is using reinforcement learning techniques to optimize over the CIDEr metric. Best results are in bold characters.

Model	Bleu-1	Bleu-2	Bleu-3	Bleu-4	ROUGE	METEOR	CIDEr
Baseline	83.2	72.5	63.1	52.7	71.4	34.1	79.5
Baseline + matching	83.4	72.8	63.3	53.3	71.2	34.5	82.2
L-STAP (avg) + matching	84.1	74.0	65.0	55.1	72.3	35.4	84.3
L-STAP (attention) + matching	84.0	74.1	64.5	55.1	72.7	35.4	86.8

Table 3: Results of ablation study on MSVD. Results show that a significant improvement can be reached using our Learned Spatio-Temporal Adaptive Pooling instead of the usual average pooling. Pooling hidden states of the encoder using soft-attention (line 4) instead of average pooling (line 3) does not always improve results. Our interpretation of that outcome is that the LSTM actually performs a kind of attention on local features before local hidden states are pooled together.

2016. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*. 265–283.

[2] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 6077–6086.

[3] Joao Carreira and Andrew Zisserman. 2017. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 6299–6308.

[4] David L Chen and William B Dolan. 2011. Collecting highly parallel data for paraphrase evaluation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, 190–200.

[5] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.

[7] Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the ninth workshop on statistical machine translation*. 376–380.

[8] Jeffrey 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 *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2625–2634.

[9] Fartash Faghri, David J Fleet, Jamie Ryan Kiros, and Sanja Fidler. 2017. Vse++: Improving visual-semantic embeddings with hard negatives. *arXiv preprint arXiv:1707.05612* (2017).

[10] Zhao Guo, Lianli Gao, Jingkuan Song, Xing Xu, Jie Shao, and Heng Tao Shen. 2016. Attention-based LSTM with semantic consistency for videos captioning. In *Proceedings of the 24th ACM international conference on Multimedia*. ACM, 357–361.

[11] 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*. 770–778.

[12] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[14] Lijun Li and Boqing Gong. 2019. End-to-end video captioning with multitask reinforcement learning. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 339–348.

[15] Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out* (2004).

[16] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*. Springer, 740–755.

[17] Sheng Liu, Zhou Ren, and Junsong Yuan. 2018. SibNet: Sibling Convolutional Encoder for Video Captioning. In *2018 ACM Multimedia Conference on Multimedia Conference*. ACM, 1425–1434.

[18] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* (2015).

[19] Yingwei Pan, Ting Yao, Houqiang Li, and Tao Mei. 2017. Video captioning with transferred semantic attributes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 6504–6512.

[20] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, 311–318.

[21] Ramakanth Pasunuru and Mohit Bansal. 2017. Multi-task video captioning with video and entailment generation. *arXiv preprint arXiv:1704.07489* (2017).

[22] Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7008–7024.

[23] Marcus Rohrbach, Wei Qiu, Ivan Titov, Stefan Thater, Manfred Pinkal, and Bernt Schiele. 2013. Translating video content to natural language descriptions. In *Proceedings of the IEEE International Conference on Computer Vision*. 433–440.

[24] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[25] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. 3104–3112.

[26] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1–9.

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

[28] Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4566–4575.

[29] Subhashini Venugopalan, Marcus Rohrbach, Jeffrey Donahue, Raymond Mooney, Trevor Darrell, and Kate Saenko. 2015. Sequence to sequence-video to text. In *Proceedings of the IEEE international conference on computer vision*. 4534–4542.

[30] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3156–3164.

[31] Bairui Wang, Lin Ma, Wei Zhang, and Wei Liu. 2018. Reconstruction network for video captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7622–7631.

[32] Huiyun Wang, Youjiang Xu, and Yahong Han. 2018. Spotting and aggregating salient regions for video captioning. In *2018 ACM Multimedia Conference on Multimedia Conference*. ACM, 1519–1526.

[33] Junbo Wang, Wei Wang, Yan Huang, Liang Wang, and Tieniu Tan. 2018. M3: multimodal memory modelling for video captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7512–7520.

[34] Xin Wang, Wenhu Chen, Jiawei Wu, Yuan-Fang Wang, and William Yang Wang. 2018. Video captioning via hierarchical reinforcement learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4213–4222.

[35] Xian Wu, Guanbin Li, Qingxing Cao, Qingge Ji, and Liang Lin. 2018. Interpretable Video Captioning via Trajectory Structured Localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 6829–6837.

[36] Jun Xu, Tao Mei, Ting Yao, and Yong Rui. 2016. Msr-vtt: A large video description dataset for bridging video and language. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5288–5296.

[37] Ziwei Yang, Yahong Han, and Zheng Wang. 2017. Catching the temporal regions-of-interest for video captioning. In *Proceedings of the 25th ACM international conference on Multimedia*. ACM, 146–153.

[38] Li Yao, Atousa Torabi, Kyunghyun Cho, Nicolas Ballas, Christopher Pal, Hugo Larochelle, and Aaron Courville. 2015. Describing videos by exploiting temporal structure. In *Proceedings of the IEEE international conference on computer vision*. 4507–4515.

[39] Youngjae Yu, Hyungjin Ko, Jongwook Choi, and Gunhee Kim. 2017. End-to-end concept word detection for video captioning, retrieval, and question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3165–3173.

[40] Junchao Zhang and Yuxin Peng. 2019. Hierarchical Vision-Language Alignment for Video Captioning. In *International Conference on Multimedia Modeling*. Springer, 42–54.