Temporal Segment Captioning Network

Yu Mao
ymao1@andrew.cmu.edu

Sree Harsha Kalli
skalli@andrew.cmu.edu

Abstract

Videos, unlike images are much more complex to understand as changes in view point, speed of action, duration of video among others are significantly different across videos and make video understanding much more challenging. Thus any method for generating descriptions of a video must be robust to these variations and also account for the varying lengths of videos. In our project we employ a model that uses a stratified sampling strategy coupled with an LSTM module to generate captions for videos. We evaluate our model using standard scores such as BLEU, CIDEr, METEOR and show that not only does our model generate very rich sentences but also **beats the state of the art methods** in some scores while remaining competitive in other metrics. We have released our code at https://github.com/YuMao1993/ActivityNetVideoCaptioning.

1 Introduction

Since (1) used convolutional neural networks for image classification, almost all computer vision related problems related to images have been improved with the use of Convolutional Neural Networks. (2) uses a combination of CNN and LSTM to predict the next word in the sentence conditioned on the previous word. More recently with the tremendous improvements in the field of images, people have started looking at more challenging problems related to videos.While image captioning handles a variable number of outputs, video captioning is even more challenging since the input is also of variable length. There are several ways to deal with the variable input size such as a simple way is to average the per frame features but this would result in loss of rich temporal information and also since we know that the adjacent frames have lot of overlapping information, giving each feature vector the same weight doesn't work very well. One popular approach that is often used for videos is the C3D (6) feature vectors which is essentially a 3D convolution on a chunk of video data. (4) uses a CNN network along with an encoder and a decoder network to predict the sentence for a video. The problem of captioning videos is extremely challenging due to diverse set of scenes, objects and varying lengths of videos we have to describe. Our architecture draws its inspiration from (5).
In (5), the authors use a stratified sampling strategy to pick a fixed number of frames to train and test the network for an action recognition task. We extend similar sampling strategy to the captioning task. Our model is shown in 1. Using a stratified sampling strategy, a frame is picked at random from each segment and is passed through a pre-trained Inception network and the last fc layer features' are extracted. These features are concatenated and collapsed to a 1024 dimensional vector. This vector is concatenated with a word embedding of the previous word and is fed into the LSTM module to predict the next word. Our model deals with a variable length of inputs, models the temporal structure of the video and exploits the temporal consistency in the videos and also learns the language model to generate rich and long sentences. We work with Activity Net Captions data set (3) and use BLEU, CIDEr and METEOR scores to evaluate the learned model. We show that the learned model beats the state of the art models in CIDEr scores and remains competitive in other scores. We also provide a list of generated sentences for some sample videos.

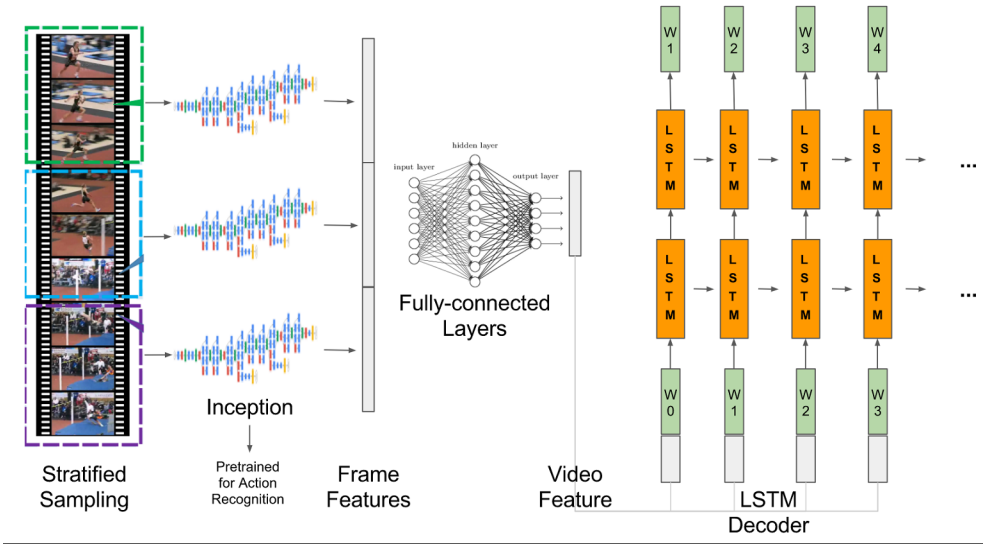


Figure 1: Architecture of Temporal Segment Captioning Network

2 Related Work

Early works on video captioning (7) tagged videos with metaddata and used a cluster of the captions and metadata to retrieve a caption for a video. With the emergence of deep learning, there has been a lot of work recently on generating captions for images as well as on videos. (2) solves the problem of image captioning by using an end to end CNN plus LSTM model. The input image is fed into the GoogleNet and the fc layer features are extracted and passed into the LSTM module in the first time step to predict the first word. In the next timesteps, only the word embeddings of the previous word are fed into the network. The reason for not feeding the image features in every time step was that since the LSTM module was overfitting easily, however in our language model, we found that for videos feeding in the features at every time step improves the model's performance partly because unlike images , videos have lot of data and are less likely to overfit.
(10) first pools the frame level features and then passes it through an LSTM encoder decoder network to generate the sentences. The main drawbck in this method is that the temporal ordering is ignored. (4) extracts feature vector for every frame and instead of pooling these features across frames, passes these features into an encoder lSTM. Once all the frames are over the decoder LSTM starts generating the words. The main shortcoming of these approaches is that consecutive frames are redundant in data and hence we need not use all the frames features to generate captions from the video.Hence inspired from (5), we use a stratified sampling strategy to pick certain number of frames during training and testing. More details about our architecture are explained in the below section.

3 Dataset

The dataset we choose to work on is ActivityNet (8). It has 10,024 training videos, 4926 validation videos, 5,044 testing videos. The training and test videos are divided into segments(with a start time and end time) with each segment having one action label out of 200 action classes and a caption as well. It is important to note that although a video can have multiple segments, each segment has the same action class. On an average each video in this data set has been annotated with an average of 3.65 sentences. Each sentence has an average length of 13.48 words.

4 Methods

Data Preprocessing

Video Trimming

Given an input video with its segments annotated, we first trim the video according to the segment start and end times. Hence for a given video V with S number of segments, we extract S number of trimmed videos according to their start and end times and each trimmed video has a corresponding caption associated with it. During training, we treat each trimmed video segments independently.

RGB Frames Augmentation

Data augmentation is a technique that can generate diverse training samples and prevent the model from overfitting. In our system, random cropping and horizontal flipping are used to augment training video frames.

Caption Preprocessing

For each sentences, we insert a <BEGIN> token at the front and append an <END> token at the end. Sentences are then tokenized using NLTK(18) and are converted to lower case form. Then each word is mapped to a unique index in our vocabulary. We constructed the vocabulary by selecting the most frequent words appeared in the dataset. Words that are not found in the vocabulary are replaced with a special <UNK> token.

Pretraining for Activity Recognition

Over the last few years it has been shown that CNN is capable of extracting rich and fix-length feature representations from the input images, so in our system, we adopted CNN as the encoder of the video in our system. It has also been shown that models with pretraining are generally more capable of representing visual concepts than models without pretraining. Therefore, we pretrained the network for activity recognition, which is the task of classifying a given sequence of video to one of the action classes. We think activity recognition is a very relevant task to video captioning. Because to describe the video in a natural way, one must first understand what the action is. Besides, (5) has shown that the trained model on activity recognition tends to focus on humans in the video and learns to model the long-range temporal structure with respect to the action class. Therefore, we followed (5) and trained a GoogLeNet(9) network for activity recognition on RGB frames of the video.

From frame features to video feature

Because videos come with various lengths, one challenge we are facing using image-based CNN is to aggregate the frame-wise deep features into a single video-level representation. Although it is very attempting to use an LSTM to learn a video-level feature embedding from frame-level feature embedding, one potential issue is vanishing gradient due the fact that most videos contain more than one thousand frames. In stead, we borrowed the idea from Temporal Segment Network(5) for action recognition to sample a fixed number of frames to represent the whole video. Specifically, we divide the video into fixed number of segments and randomly sample one frame from each segments. Then the deep features from each of the selected images are concatenated and fed through FC layers and converted to a feature representation of the video, as shown in [1].

LSTM decoder

The most common challenge in training RNNs is to deal with the vanishing gradient problem. And one form of RNN called Long Short Term Memory(LSTM)(12) that introduces the idea of memory cells controled by 'gates', has been applied with great success in many language related tasks such as image captioning(2) and machine translation(11). The update rule of the LSTM module used in our system has the following definition.

$$
\begin{aligned}
f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) \\
\tilde{c}_t &= tanh(W_c x_t + U_c h_{t-1} + b_c) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\
h_t &= o_t \odot c_t \\
p_t &= softmax(W_p h_t + b_p)
\end{aligned}
$$

where f_t, i_t and o_t represents the values of forget gate, input gate and output gate at time t respectively. $sigma$ and $tanh$ are sigmoid and hyperbolic tangent non-linearity functions. As we can see these gates are used to control how much to forget the current cell state, to read in from the new cell state update values and to output as the new hidden state. Once we get the new hidden state, we can compute from it the distribution of the new words by applying a softmax layer.

We used a stack of two LSTMs as our decoder, where the output of the first LSTM is fed as the input to the second LSTM. We concatenate the current word embedding at every time step with the fixed video level feature and feed it into the LSTM as the input. Empirically, we found it works better than only feeding the video level feature at the first time step like (2). Our explanation is that this approach prevents the network from forgetting the content of the image though cell state update. If we denote the video level feature as f, the one-hot-encoding of the input word at time step t as s_t. The decoding procedure for computing the probability distribution of the next word can be written as

$$
\begin{aligned}
e_t &= W_e s_t \\
x_t &= \begin{pmatrix} e_t \\ f \end{pmatrix} \\
p_t &= LSTM(x_t)
\end{aligned}
$$

where W_e is the embedding layer, s_t is the current word, e_t is the embedding of the current word and x_t is the input to the LSTM.

Training

The loss function we used is the sum of negative log likelihood of the correct word at each step.

$$
L = -\sum_{t=1}^{N} log p_t(s_t)
$$

Due to limited time and computation resources, during training, we froze the weights of the pretrained GoogLeNet and simply used it as a feature extractor. We trained only the video feature embedding layer and LSTM layer.

Inference

During inference, we first compute the video-level feature using the sampling strategy as discussed earlier. Then we feed the LSTM with the concatenation of video-level feature and the embedding of <BEGIN> token and start to decode. At each time step, the LSTM decoder outputs the probability distribution for the net word. There are several ways to choose the next word. The first method is **argmax**, where we just choose the word with the highest probability. The second method is **sampling**, where we sample the next word according to the probability output of the LSTM decoder. The third method is **beam search**, where we maintain a fixed-size set of most promising candidates found up to time t to generate sentences up to time $t + 1$. During experiments, we found that argmax method gives the best performance and this is the selection method we adopted in all of our following experiments.

	FLICKER-8K			COMPOSITE		
Metric	Pearson	Spearman	Kendall	Pearson	Spearman	Kendall
CIDEr	0.60	0.56	0.45	0.32	0.42	0.32
METEOR	0.69	0.58	0.47	0.37	0.44	0.33
BLEU	0.59	0.44	0.35	0.34	0.38	0.28

Table 1: Pearson, Spearman and Kendall Correlation coefficients on FLICKER-8K and COMPOSITE datasets for BLEU, CIDEr, METEOR metrics with human annotations

5 Evaluation Metrics

For this project, we chose to compare scores of the following standard evaluation metrics which have been used in the literature to evaluate language modelling tasks. We evaluate our model using the below metrics on ActivityNet captions dataset and compare the results with the ones mentioned in (7), (4), (10), (16).

BLEU Bilingual Evaluation Understudy(BLEU) is defined as $p = \frac{m}{w}$ where w is the total words in the input sentence and m is capped at the maximum occurrences of that word in the reference. However, if we use this score then the metric will always favor short sentences. In order to account for this, a brevity penalty is applied.
Let r be the total length of the reference corpus, and c the total length of the translation corpus. If $c \leq r$, then a brevity penalty of $e^{1-\frac{r}{c}}$ is applied as detailed in (13)
There are several forms of this score called BLEU-n where n is the n gram ,length being accounted for.

METEOR The advantage of Metric for Evaluation of Translation with Explicit ORdering(METEOR) metric compared to BLEU score is two folds.

- There is ordering of the words taken into consideration
- The score uses both precision and recall

The score is a harmonic mean of unigram precision and recall with recall weighted 9 times more than precision. The exact details and formula can be found in (14).

CIDEr Consensus-based Image Description Evaluation(CIDEr) is a more recent metric stated in (15) which shows high correlation with human evaluations on some datasets. The idea is similar to measuring the similarity of the majority of consensus of most people's descriptions.
All of the above mentioned metrics have some drawbacks and [1] shows the correlation of these metric's scores with those of human evaluations.

6 Results

Training Details

The size of the vocabulary we constructed is 4399, including inserted <BEGIN>, <END> and <UNK> tokens. The size of the vocabulary is chosen so that it contains 95% of the words appeared in the corpus. During training, we divided each video into 8 segments and randomly sampled one frame from each segments, from which the global_pool features were extracted, concatenated and then fed to the video embedding module. The video embedding module consists of one hidden layer with 1024 hidden units. We used $tanh$ as the non-linearity for the embedding module. The video embedding size is 512. The word embedding size is also 512. The hidden units of the two stacked LSTM modules are both 1024. To prevent the LSTM module from overfitting, we applied dropout(19) to the LSTM module and the dropout rate is set to 0.05. We used L2 regularization on the video embedding module with a strength of 1e-6. During training, we use stochastic gradient descent(SGD) with a batch size of 256. We trained the model for 50,000 iterations.

Quantitative Analysis

We evaluated the model using different metrics and in [2] are the results compared to other state of the art models for video caption generation. The code for the evaluation is obtained from (17).

Evaluation						
Method	BLEU@1	BLEU@2	BLEU@3	BLEU@4	METEOR	CIDEr
LSTM-YT(10)	18.22	7.43	3.24	1.24	6.56	14.86
S2VT (16)	20.35	8.99	4.60	2.62	7.85	20.97
H-RNN (4)	19.46	8.78	4.34	2.53	8.02	20.18
C3D+LSTM (3)	**26.45**	**13.48**	**7.12**	**3.98**	**9.46**	24.56
TSCN (ours)	18.20	7.32	2.93	1.22	9.07	**25.62**

Table 2: Comparison of our model to other models using standard evaluation metrics

From the results we can see that our model beats all the methods in CIDEr scores and beats the first three methods in METEOR score, while remaining competitive in other metrics. One important thing to notice about these automatic metrics for caption generation is that none of them are perfect. A detailed discussion can be found in (21).

Qualitative Analysis

We also show some qualitative evaluations of our model on some sample videos. in [2], we can see that the generated sentences are extremely good, long and have good grammar. We also showed some not so good captions generated from our model in [3]. The model makes some mistakes such as repeating the same sentence again, doesn't really understand what is possible and what is not, for example it says "a man is seen riding a kite down a hill" where as we know that a man can never ride a kite. Although there are some mistakes these are extremely reasonable mistakes that the model makes given the data that we have used during training.

7 Conclusion

In this paper, we proposed a novel architecture to deal with the varying lengths of the input videos to generate captions for the videos. In contrast to other methods, which make use of all the frames of a video, we use only some frames based on stratified sampling scheme thus exploiting the fact that successive frames in a video are redundant. Our model achieves state of the art results on Activity Net Captions data set and from the quantitative results, we can see that our model generates long sentences with very rich grammar and context.

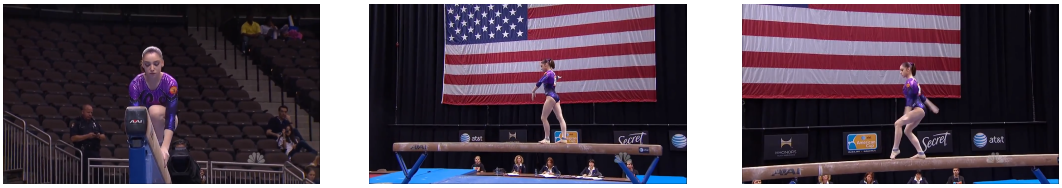
8 Future Work

We have a couple of things in mind that we would like to try out and some of them are listed below.

- In the above architecture, we have used the frame features extracted from a pretrained model on activity recognition, we would like to do an end to end training instead which would help learn better frame level features.
- During inference, we are following the same sampling strategy as in training. To reduce variance, one potential improvement is to sample the video multiple times and fuse the video embeddings computed. Or, we can run inference multiple times and select the best candidate sentences.
- In addition to Inception, we would want to use better architectures such as ResNet, Inception-V4, C3D and I3D.
- We are also looking into exploring the use of optical flow as an input modality in addition to the RGB input.
- It has been shown that attention(22) mechanism can improve the performance of image captioning task, we want to improve the system's performance by incorporating attention.

(a) Human Caption: Players play lacrosse on a field
Generated Caption: The players continue to play and the ref blocks the ball and the other team scored in the field.
URL: https://youtu.be/VV6l4fIIlAs

(b) Human Caption: He is dancing as he plays the sax
Generated Caption: The man continues playing the instrument and ends by looking off into the distance.
URL: https://youtu.be/c3VSxrbSeRc



(c) Human Caption: A young girl wearing a purple leotard jumps on a balance beam and begins to do her routine.
Generated Caption: The girl then jumps over the beam and begins performing a gymnastics routine on the mat.
URL: https://youtu.be/vzewFg1ZtY8

Figure 2: Some examples of good captions generated

(a) Human Caption: we see a man talking.
Generated Caption: A man is seen speaking to the camera and leads into a man speaking to the camera.
URL: https://youtu.be/kfed4WJ4ZeE

(b) Human Caption: A large kite is seen flying in the sky with a small group of people underneath.
Generated Caption: A man is seen riding a kite down a hill while others watch on the side.
URL: https://youtu.be/Vntvo-QNMpk

Figure 3: Some examples of mistakes the model makes

References

[1] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

[2] Vinyals, Oriol, et al. "Show and tell: A neural image caption generator." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[3] Krishna, Ranjay, et al. "Dense-Captioning Events in Videos." arXiv preprint arXiv:1705.00754 (2017).

[4] Venugopalan, Subhashini, et al. "Sequence to sequence-video to text." Proceedings of the IEEE international conference on computer vision. 2015.

[5] Wang, Limin, et al. "Temporal segment networks: Towards good practices for deep action recognition." European Conference on Computer Vision. Springer International Publishing, 2016.

[6] Tran, Du, et al. "C3D: generic features for video analysis." CoRR, abs/1412.0767 2.7 (2014): 8.

[7] Aradhye, Hrishikesh, George Toderici, and Jay Yagnik. "Video2text: Learning to annotate video content." Data Mining Workshops, 2009. ICDMW'09. IEEE International Conference on. IEEE, 2009.

[8] Caba Heilbron, Fabian, et al. "Activitynet: A large-scale video benchmark for human activity understanding." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015.

[9] Szegedy, Christian, et al. "Going deeper with convolutions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[10] Venugopalan, Subhashini, et al. "Translating videos to natural language using deep recurrent neural networks." arXiv preprint arXiv:1412.4729 (2014).

[11] Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. "Sequence to sequence learning with neural networks." Advances in neural information processing systems. 2014.

[12] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.

[13] Papineni, Kishore, et al. "BLEU: a method for automatic evaluation of machine translation." Proceedings of the 40th annual meeting on association for computational linguistics. Association for Computational Linguistics, 2002.

[14] Denkowski, Michael, and Alon Lavie. "Meteor universal: Language specific translation evaluation for any target language." Proceedings of the ninth workshop on statistical machine translation. 2014.

[15] Vedantam, Ramakrishna, C. Lawrence Zitnick, and Devi Parikh. "Cider: Consensus-based image description evaluation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[16] Yu, Haonan, et al. "Video paragraph captioning using hierarchical recurrent neural networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[17] https://github.com/ranjaykrishna/densevid_eval

[18] Bird, Steven. "NLTK: the natural language toolkit." Proceedings of the COLING/ACL on Interactive presentation sessions. Association for Computational Linguistics, 2006.

[19] Zaremba, Wojciech, Ilya Sutskever, and Oriol Vinyals. "Recurrent neural network regularization." arXiv preprint arXiv:1409.2329 (2014).

[20] http://activity-net.org/challenges/2017/guidelines.html

[21] Kilickaya, Mert, et al. "Re-evaluating automatic metrics for image captioning." arXiv preprint arXiv:1612.07600 (2016).

[22] Witten, Ian H., et al. Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann, 2016. APA