Video Captioning using Temporal Deformable Convolution

Soumith, Puneeth, Sumanth, Bala

Abstract
Video captioning is a critical step towards machine intelligence and many applications such as video retrieval, video understanding, and automatic video subtitling. In this work, we talk about Temporal Deformorable Convolutional Endoder - Decoder architecture proposed in the paper [1] from CVPR 2019. This techique used convolutional layers to model temporal dependencies instead of RNNs and its other variants. We use this model to perform video captioning on You Cook 2 [2] dataset. We train and evaluate our model on this dataset and we present the results.

Keywords
Video Captioning, Temporal Deformable Convolution, You Cook 2

Contents

1. Introduction

Video captioning is the automatic generation of natural language sentences that describe the contents of a given video. It has applications in human-robot interaction, video understanding and video subtitling. Video captioning requires concepts from the fields of Computer Vision (CV) and Natural Language Processing (NLP). With advances in both these fields, a lot of work is being done towards video captioning [3]. In this work we have explored a fully convolutional network for video captioning proposed in the paper titled 'Temporal Deformable Convolutional Encoder-Decoder Networks for Video Captioning' [1]. The paper proposes a model using only convolutional networks to model time dependencies. Recently, CNNs have been used for sequence learning in many fields like speech recognition, translation etc and have performed well [4]. CNN allows a hierarchical learning process where in lower layer nearby input sequences interact and as we move downstream in the layers interactions of distant input sequences are learned as seen in 1. Also CNNs are extremely parllelizable, so they are fast. In this study, we explore the CNN model proposed in the paper. We train the model from scratch on You Cook 2 dataset. Since the authors haven't trained on this dataset, our aim is to see how the model performs on this dataset. This paper is structured as follows, next section gives a brief overview of the paper mentioned above followed by how we built the model and training procedure. We then talk about our experiments and results.

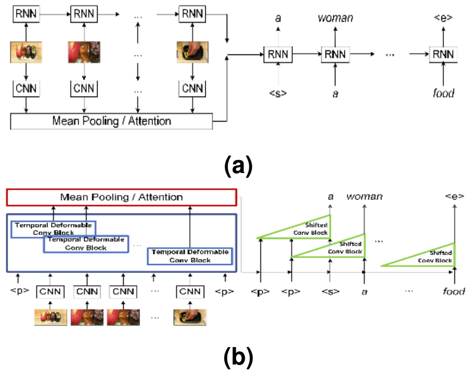


Figure 1. An illustration of (a) CNN or RNN plus RNN ar-chitecture and (b) Temporal Deformable ConvolutionalEncoder-Decoder Networks, for video captioning. [1]

2. Paper Summary

2.1 Problem Formulation

A video $\mathbf{V}$ is sampled into N_V denoted as $v = (v_1, v_2, .., v_{n_v})$ and $S = (w_1, w_2, w_3, ..., w_{N_v})$is the word sequence of the output of n_s words.
The temporal deformable convolutional encoder is used to

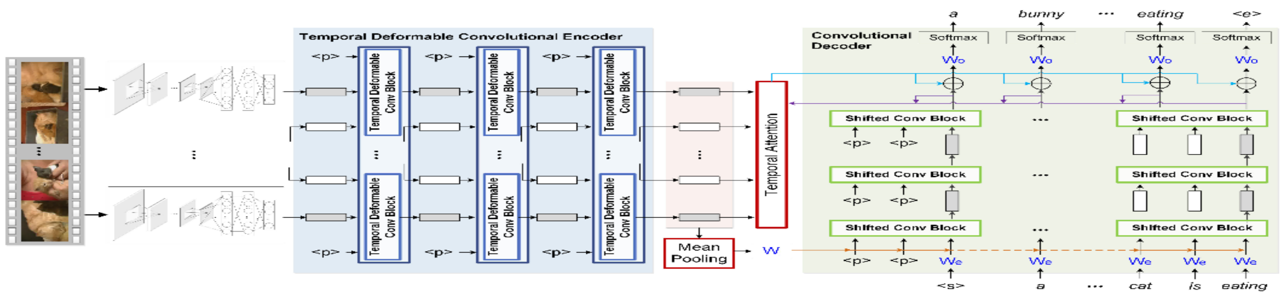


Figure 2. The overall architecture of our TDConvED that fully employs convolutions in both encoder and decoder networksfor video captioning. A standard CNN is firstly utilized to extract visual features of sampled frames/clips. Then, the features offrames/clips are fed into a temporal deformable convolutional encoder with stacked temporal deformable convolutional blocksto produce intermediate states of each frame/clip with contexts. The video-level representations are computed by mean poolingover all the intermediate states. After that, a convolutional decoder with stacked shifted convolutional blocks operates on theconcatenation of video-level representations and the embeddings of past words to generate the next word. In addition, a temporalattention mechanism tailored to such convolutional encoder-decoder structure is incorporated to boost video captioning. [1]

encode input frame sequence $v = (v_1, v_2, .., v_{n_v})$ into a set of context vectors $Z = (z_1, z_2, .., z_{n_v})$, which encode the contextual information among frames/clips sampled in a free-form temporal deformation along the input sequence.The context vectors Z are used by convolutional decoder to generate target output sentence S by capturing the long-term contextual relations among input video and previous output words.The Video captioning problem is formulated by minimizing the following energy loss.

$$E(v,S) = -logP_r(S/v)$$

The log probability of the sentence is measured by the sum of the log probabilities over each word, so the loss can be written as.

$E(v,S) = -\sum_{t=1}^{N_s} logP_r(w_t/v, w_0, .., w_{t-1})$ The temporal deformable convolutional encoder decoder architecture computes the joint probability $P_r(S/v)$ using convolutional layers.

2.2 Encoder

The encoder takes the source sequence as input and produces intermediate states to encode the semantic content. Here a temporal deformable convolutional block is created which applies temporal deformable convolution over the input sequence to capture the content across the frames sampled free-form temporal deformation.The free forward convolution in the encoder enables parallelization which allows in fast computation. Multiple temporal deformable convolutional blocks are stacked in the encoder to take content across a large number of input sequences.

The i^{th} temporal deformable convolutional block that is centered on the i^{th} frame of the video produces the output denoted as $P^l = (p_1^l, p_2^l, ..., p_{N_v}^l)$, where $p_i^l \in R^{D_r}$. Given the output sequence of the (l-1)-th bock the output of the lth block is obtained by taking the sequence of the (l-1)-th block and applying temporal deformable convolution (kernel size: k) plus a non-linearity unit on it.The temporal deformable convolution happens in a two-stage way. First a sub- sequence from output of (l-1)-th block,$X = (p_{i+r_1}^{l-1}, p_{i+r_2}^{l-1}, ..., p_{i+r_k}^{l-1})$ (i.e X is sub sequence of P^{l-1}) where r_n is the n^{th} element in $K = \{-k/2, .., 0, .., k/2\}$ is taken and one-dimensional convolution is applied to it, with parameters $W_f^l \in R^{k \times kD_r}$ as transformation matrix and $b_f^l \in R^k$. This produces a set of offsets $\Delta r^i = \{\Delta r_n^i\}_{n=1}^k \in R^k$

$$\Delta r^i = W_f^l[p_{i+r_1}^{l-1}, p_{i+r_2}^{l-1}, ..., p_{i+r_k}^{l-1}] + b_f^l$$

Where n-th element Δr_n^i in Δr^i denotes the measured offset for the n-th sample in the sub sequence X.The output of the temporal deformable convolution is obtained by applying the one-dimensional convolution to the samples which are argumented by their offsets.

$$o_i^l = W^l[p_{i+r_1+\Delta r_1^i}^{l-1}, p_{i+r_2+\Delta r_2^i}^{l-1}, ..., p_{i+r_k+\Delta r_k^i}^{l-1}] + b_f^l$$

Where $W_d^l \in R^{2k \times kD_r}$ is the transformation matrix and $b_d^l \in R^{2D_r}$ is the bias.Typically the offsets are fractional so the sample defined by $p_{i+r_k+\Delta r_k^i}^{l-1}$ is calculated by linear interpolation from the sample before and after it. The sample is defined by.

$$p_{i+r_k+\Delta r_k^i}^{l-1} = \sum_s B(s, i + r^k + \Delta r_k^i) p_s^{l-1}$$

Here s is all the integral positions of P^{l-1}, and $B(a,b) = max(0, 1 - |a-b|)$. Thus the output of temporal deformable convolution is twice the size of the output, so a gated linear unit(GLU) is applied over o_i^l.

$$g(o_i^l) = A \otimes \sigma(B)$$

So the final output of a l-th temporal deformable convolu-

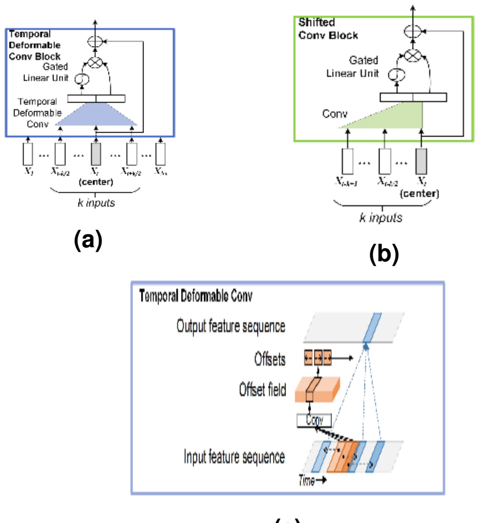


Figure 3. An illustration of (a) the temporal deformable convblock in encoder, (b) the shifted conv block in decoder, and(c) the temporal deformable convolution. [1]

tional block is $p_i^l = g(o_i^l) + p_i^{l-1}$.To make the length of the output and input of the block the left and right side of the input is padded with k/2 zeros. And by stacking several temporal deformable convolutional blocks over the input frame sequence we obtain final sequence of context vectors $Z = (z_1, z_2, ..., z_{N_v})$.

2.3 Decoder

The decoder is stacked with several shifted convolutional blocks to capture long-term context across the encoded context vectors of input video and output words (as shown in the fig).The center position of the convolution in shifted convolutional block is shifted with respect to the center of normal convolutions.

First mean pooling over all the context vectors $Z = (z_1, z_2, .., z_{N_v}$ and Z' is obtained.The Z' is transformed with linear mapping and concatenated with the with the embeddings of the input word at each time step, which will be set as the input of the first shifted convolutional block of the decoder.For a l-th shifted convolutional block in decoder at time step t the output sequence is denoted as $q^l = (q_0^l, q_1^l, ..., q_{N_s}^l)$,where $q_t^l \in R^{D_f}$. Here q^l is computed by taking a sub sequence from the output of the (l-1)-th shifted convolutional block and applying one-dimensional convolution(kernel size: k) plus a GLU and residual connections.The sub sequence is $X_q = (q_{t-k+1}^{l-1}, q_{t-k+2}^{l-1}, ..., q_t^{l-1})$ and the output of the convolutional block is.

$$q_t^l = g(W_l^q[q_{t-k+1}^{l-1}, q_{t-k+2}^{l-1}, ..., q_t^{l-1}] + b_l^q) + q_t^{l-1}$$

The left side of the input is padded with k-1 zeros so the its length is equal to output.The shifted convolutional blocks are stacked so that its can capture the context dependency of the all the previous words and predict current word.The output of the stacked shifted convolutional blocks is denoted as $h = (h_1, h_2, .., h_{N_s}$.

To include video level representations for predicting a word target word, a temporal attention mechanism is employed,this is to boost video captioning.At every time step t the attention mechanism generates a normalized attention distribution over Z' based on the current output h_t of the decoder.

$$a_i^t = W_a[\tanh(W_z z_i + W_h h_t + b_a)]$$
$$\lambda^t = softmax(a^t)$$

Here λ^tdenotes the normalized attention distribution and its i-th element λ_i^t is the attention probability of context vector z_i. Based on the λ^t a weighted sum is taken and denoted by $\hat{Z}_t$ for every time step.Then the weighted sum $\hat{Z}_t$ is transformed with a linear mapping and combined with the output intermediate state h_t, which is used to predict the next word through a softmax layer.

3. Data Set

We used YouCook2 dataset for our project. It consists of 2000 videos. The total video time is 176 hours. The duration of each video is around 5 minutes. Captions were given to each segment in a video. There are 6 to 8 video segments for each video. We first downloaded the videos from the URL's given in the annotations file(JSON format) for each video. We sampled these videos into sample images for each segment in a video as per the sampling frequency which is a hyper parameter. But the total videos size is around 130GB. When we tried to sample the video, it is taking a very long time to extract the sampled images. Instead we took a pre-computed ResNet feature file from their website. This is a frame wise resnet-34 features in .dat format for training, validation and testing. In our case, F = 500 for all the videos so the average sampling rate is 1.58 fps. During training, we temporally augment the data by sampling each video at most R = 10 times starting from frame $max(I/R, 1)$ r, where r = 0, 1, ..., R 1. These can be used to augment data.

Total Videos	Train	Validation	Test
2000	1333	457	210

Each video is sampled into 500 frames and ResNet features (features from last fully-connected layer of dimension 512 of ResNet34) is given for each frame. Each frame has a feature vector of size 512.

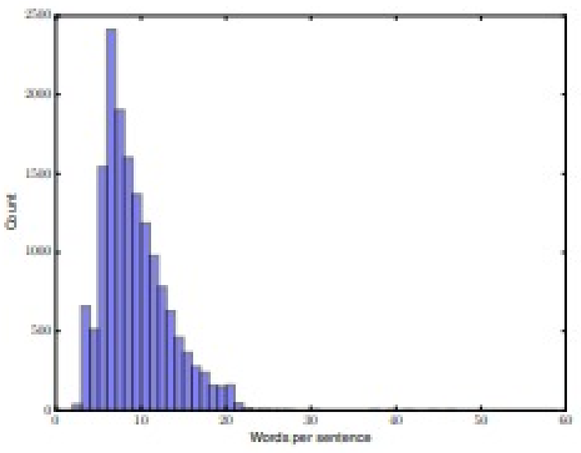
4. Training Procedure

4.1 Model

Model has been adopted from implementation provided by the authors. Since the code base is still under development, we have only used the temporal deformable conv blocks (encoder and decoder) from the original implementation and we have build the data loading and evaluation systems. This also simplified the task of making the model incorporate our dataset (You Cook 2) since the original model was not built to work with this dataset. We have built data loaders to work with the dataset such that they can be parameterised with various settings that we have used for our experiments. Our final model uses two levels of encoders and decoders (in the architecture discussed in the previous section) and all the encoding dimensions (in both encoder and decoder) and embedding dimensions are set at 256.

4.2 Training and Experiments

Model is trained for 30 epochs. Adam optimizer is used during the back-propagation with cross entropy as loss function. The complete model and its training has been implemented using pyTorch. BLEU score has been used as the metric for the model which is available in nltk library of python.
The max length of the sentence found to be 237 from the inspection of the dataset. But this didn't produce the desired results. Later after analysing the different lengths of sentences present, we found out that the sentences within the length of 20 are more. So we used this as the maximum length and were able to generate good results.



(d) Distribution of number of words per sentence.

Figure 4. As you can see the distribution of the length of the dataset, we have cut the max length of sentence to 20. [2]

The initial number of frames extracted per clip was 5. This was not that effective. After this was changed to 10 which generated good results. We hoped we would get much good results if we used 15, but the results generated were much worse than the values generated with 10.

BLEU score

BLEU (Bilingual Evaluation Understudy) is a measurement of the differences between an automatic translation and one or more human-created reference translations of the same source sentence.
The BLEU algorithm compares consecutive phrases of the automatic translation with the consecutive phrases it finds in the reference translation, and counts the number of matches, in a weighted fashion. These matches are position independent. A higher match degree indicates a higher degree of similarity with the reference translation, and higher score. Intelligibility and grammatical correctness are not taken into account.
In a more mathematical sense, BLEU score works by counting matching n-grams in the candidate translation to n-grams in the reference text, where 1-gram or unigram would be each token and a bigram comparison would be each word pair. The comparison is made regardless of word order. BLEU score ranges from 0 to 1.

5. Results

We compared our model with the two models Bi-LSTM + Temporal Attention and Masked Transformer.

Method	B@4	M
Bi-LSTM + Temporal Attention	0.87	8.15
Masked Transfomer	1.42	11.20
Our Model	**1.26**	**9.45**

The Bleu score of our model is more when compared to the RNN based approach.

frames pre sample	B@4
5	0.63
10	**1.26**
15	1.02

References

[1] Jingwen Chen, Yingwei Pan, Yehao Li, Ting Yao, Hongyang Chao, and Tao Mei. Temporal deformable convolutional encoder-decoder networks for video captioning, 2019.

[2] Luowei Zhou, Chenliang Xu, and Jason J Corso. Towards automatic learning of procedures from web instructional videos. In *AAAI Conference on Artificial Intelligence*, pages 7590–7598, 2018.

[3] Md. Zakir Hossain, Ferdous Sohel, Mohd Fairuz Shiratuddin, and Hamid Laga. A comprehensive survey of deep learning for image captioning, 2018.

[4] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning, 2017.