Short Form Video Captioning with C3D and CLIP

Chenyang Dai
Stanford University
qddaichy@stanford.edu

Abstract

In this project, we build video captioning system for the short form videos focusing on Microsoft Research Video Description Corpus Dataset (MSVD) [1] and Microsoft Research Video to Text Dataset (MSR-VTT) [2]. We implement and train two encoder-decoder models using CNN and transformer approaches. The first model uses C3D [3] encoding and the second model uses CLIP [4] encoding while both models use GPT-2 [5] for deocoding. We explore model variations and optimization methods for the best performance. Specifically, there are three major contributions. Firstly, we implement both models from scratch and achieve acceptable performance. Secondly, we conduct various experiments regarding training strategies and find that pretraining with image captioning dataset and fine tuning specific layers with video captioning data set can greatly boost performance. Finally, we conclude that adopting beam search [6] and use more input frames can further improve the performance. Our best single captioning model achieved 46.39 BLEU-4 [7] score on MSVD test set and 36.45 BLEU-4 score score on MSR-VTT test set.

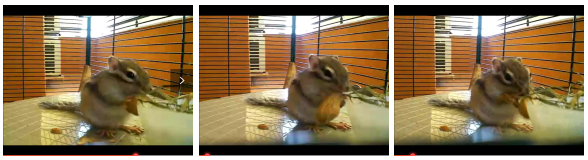
1. Introduction

Nowadays, video has became the number one media format on internet. Recent study by Cisco [8] shows that more than 80% of all internet traffic is on video. Moreover, short-form video grows dramatically in the past years and quickly became an important part of social media. With the rise of short-form videos, video captioning has become more and more important. Captioning has the ability to make video more accessible in multiple ways. It helps with video policy check; helps in the retrieval of the video efficiently through text and helps people to capture the information more easily.

Video captioning is a challenging task that traditional machine learning approaches tend to struggle with for the following reasons: First of all, there are limited datasets that have large number of videos and accompanying text captions, because videos are significantly more difficult and expensive to collect and annotate than images and captioning is a much more complex and ambiguous task for human annotators than classification. Secondly, video captioning requires accurate modeling of each video frame's semantics for any possible video content. Without a decent understanding of video frames, it is almost impossible to generate any meaningful text. Thirdly, video captioning requires the system to keep track of long term relation, as it may be necessary to combine information from different frames at different time steps to figure out the best caption. Finally, conditional text generation in natural language can be challenging. It requires the model to generate human like natural language while capturing the video semantics.

To define the problem more accurately, given a relatively short video (less than 3 minutes) as input, our model should be able to generate a short sentence in natural language that best describes the video content with no supervision or human input. For example:

Input Video:



Output:
a chipmunk is eating a nut

In this work, we built video captioning system for the MSVD and MSR-VTT dataset. The overall idea is to use the encoder-decoder architecture. Encoder processes frames out of the video and encodes the information into a feature vector. Decoder is responsible for generating the captioning text based on the encoded feature. For encoder, we focus on two approaches (C3D and pre-trained CLIP models) and their variations. For decoder we utilize pre-trained GPT-2 language model. We experiment with different designs to connect encoder and decoder. We also experiment with different data pre-processing and training strategies for the best performance.

Our best-performing model uses CLIP and GPT2 combi-

nation. The connection layer is a three layer MLP with tanh activation. We found that useing 20 GPT2 tokens as conditional input to GPT-2 works very well (see method section for details). For training, we find that pre-training with COCO [28] dataset and apply beam search [6] can greatly improve performance. Also, at fine tuning step, we realize that freeze CLIP model and only fine tune MLP with GPT2 give us the best result (see experiment section for details).

The best model achieve 50.05 BLEU-4 score on MSVD validation set and 46.39 on test set, which is + 13.09 BLEU-4 than the baseline model [9] (CNN encoder and LSTM [10] decoder approach).

2. Related Work

Video captioning has a relatively long history of development. Researches mainly focus on modeling the information from the video and the relationship between video representations and the output text in natural language via an encoder-decoder framework for video captioning. Specifically, methods like PickNet [11], GRU-EVE [12] and Syntax-Aware Action Targeting(SAAT) [13] use a CNN or C3D based approach to refine video representations from a set of fixed video frame features and use a RNN (LSTM or GRU [14]) based decoder for text generation. As transformer take off, recent models like Video Vsion Transformer(VViT) [15], MultiModal Transformer (MMT) [16], TimeSformer [17] and SwinBert[18] abandon CNN and use a pure transformer based approach for both video modeling and text generation. In this work, we are experimenting with both CNN and transformer approaches. The goal is to better understand the pro and cons for each approach and to compare the performances.

2.1. CNN Approach

The classic model architecture using encoder-decoder idea combines a CNN-like encoder and a LSTM-like decoder as first proposed by S. Venugopalan et al [9] . In their work, the CNN is 2D and the feature maps from different frames are fused together using an average pooling layer before feeding into the decoder. This approach is the baseline model we are comparing with. The author mentioned the importance of pre-training using image data given the very limited video caption training data and the effectiveness of transfer learning between image data and video data. Based on their work, we adopt the idea of pre-training using image data in this project.

Unlike image data, video data has one additional time dimension in the input and traditional 2D CNN with a fusion layer cannot model the 3D structure very well. As a consequence, 3D ConvNets (C3D) was proposed by Du Tran et al. [3], which later became a good default choice to process stacked video frames. C3D is a deep 3D convolutional neural networks with a homogenous architecture containing 3 x 3 x 3 convolutional kernels followed by 2 x 2 x 2 pooling at each layer. 3D CNN extends the original 2D convolutional kernel and 2D pool kernel into 3D kernel to capture both spatial and temporal space. The C3D model achieved state-of-the-art performance on video classification problem at that time. In this project, one of the models we are experimenting with is using C3D as the video frames encoder.

2.2. Transformer Approach

Dosovitskiy et al. [19] first demonstrate that a pure-transformer based architecture can outperform its convolutional counterparts in ImageNet classification task [20]. Since then, pre-trained transformer based models are quickly being used in various vision tasks. Contrastive Language–Image Pre-training (CLIP) model [4] released by OpenAI is one of the state-of-the-art transformer based image encoding model so far. CLIP pre-trains an image encoder and a text encoder to predict which images were paired with which texts as the same time using a huge dataset (0.4 billion pairs). The model was proven to generalize very well on various vision related tasks such as image description matching, image search etc. In our project, we are also experimenting with CLIP to replace C3D as the video frames encoder.

Nowadays, various end to end pure transformer based models are being proposed and used for captioning tasks. The common high level idea among these models are that they completely get rid of convolutions and purely rely on attention mechanisms. SwinBert [31] built on top of Swin Transformer [18] by Microsoft is considered one of the best end to end video captioning model today. It outperforms other approaches by a large margin on 5 popular video captioning benchmarks. SwinBert uses a pre-trained Swin Transformer as the video encoder and a random initialized multimodal transformer as the text generator. The model is trained to perform video captioning end to endly. In their paper, the authors pointed out that using a high frame rate can boost the model performance and the high computation cost problem with a high frame rate can be solved by adding a spare attention mask to the multimodal transformer. In this project, we are also using SwinBert as the ceiling level benchmark we are comparing with.

2.3. Language Model Decoder

Decoding video feature into natural language is another important part of video captioning. Recent researches have demonstrated that transformer-based language models (TLM) with a large scale pretraining, such as BERT [21], ALBERT [22] and GPT-2 [5], have shown strong performance in a wide range of NLP tasks and currently dominate the field of NLP. Generative Pre-trained Transformer 2 (GPT-2) is a transformers model pretrained on a very large corpus of English data in a self-supervised fashion by

OpenAI. It was pretrained on the raw texts only to guess the next word in sentences, with no humans labelling them in any way. GPT-2 outperforms previous benchmarks for RNN/CNN/LSTM-based models in NLP area and only requires modest computation resource. Since the major focus of this work is on the vision encoding part, we adopted the pretrained GPT-2 model as the text generator for all our models and their variations.

3. Method

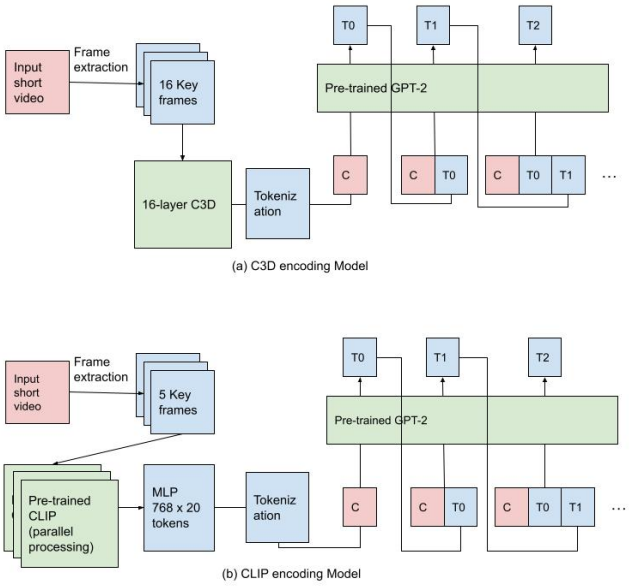


Figure 1. C3D based and CLIP based video captioning model structures

We mainly implemented two models: C3D based model and CLIP based model. Both approaches try to take advantages of open source large-scale pretrained transformer models (GPT-2 and CLIP) and apply modifications and fine tuning on top of them.

We implement our models using standard Pytorch library, Huggingface transformer library [23], and Open source CLIP library [24]. All the code in our submission is written by ourselves including data pre-processing logic, main models, training and fine tuning logic. The C3D implementation is inspired by an open source C3D project [25] and part of GPT-2 training logic is inspired by another open source project that fine tunes GPT-2 [26].

3.1. C3D Based Model

Figure 1 (a) shows the structure of the C3D based model. It takes a sequence of raw video frames as inputs and outputs a natural language description describing the given video. Due to the limited computation resources, we choose a fixed number 16 frames per video as input and the frame sample method is fixed-rate sampling. The entire model consists of two parts: C3D video encoder and GPT-2 decoder. First, we leverage C3D to extract the spatial and semantic information out of the video and encode the information into a 15360 dimension feature vector. The C3D has input size requirement as (C = 3, D = 16, W = 112, H =112). We implement scripts to squeeze and pre-process each training video into multiple training examples to meet the dimension requirement. Also, we remove the last softmax layer from C3D and take the feature vector from the last FC layers directly as the encoding result.

The C3D part consists of 8 3D convolutional layers, 5 pooling layers and 3 fully connected layers as shown in the Figure 2. The first convolution layer of size $1 \times 3 \times 3$ is followed by a pooling layer of size $1 \times 2 \times 2$. This is to preserve the temporal information in the first layer and build higher level representation of the temporal information at subsequent layers of the network. Every other convolution layer and pooling layer has a size of $3 \times 3 \times 3$ and $2 \times 2 \times 2$ where the strides are 1 and 2 respectively. The final fully connected layers have a size of 15360 dimensions. The reason behind this dimension choice is that GPT-2 language model does not take initial hidden state or conditional feature vectors as input. A common practice for GPT-2 to generate conditional output is to feed in prefix tokens at each time step. In our case, the 15360 dimension vector is then tokenized into 20 context tokens (each token has 768 dimension as required by GPT-2).

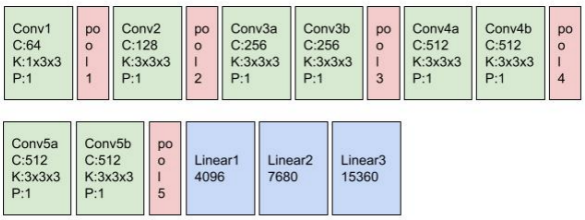


Figure 2. Modified C3D architecture (C: channels, K: kernal size, P: padding)

At training time, given the transformer nature of GPT-2, it performs masked self-attention. However, the default behavior of masked self-attention won't work because the additional 20 context tokens we want to use. As a result, we build training example and attention masks ourselves. We concatenate the 20 context tokens with the tokenized ground truth captioning text to from one training example. Then we pad all examples to the same length. For each example, we build the corresponding attention mask allowing the model to pay attention to context tokens while masking out all future tokens or pad tokens (Figure 3). This allows training to

utilize full parallelism feature of transformer. Finally, at test time, the behavior is slightly different. At test time, GPT-2 decoder only takes the 20 video representations context tokens as the initial inputs and outputs a natural language sentence via the standard sequence-to-sequence (seq2seq) generation flow. It will stop generation when max word limit (30 words) is reach or a stop token is encountered.

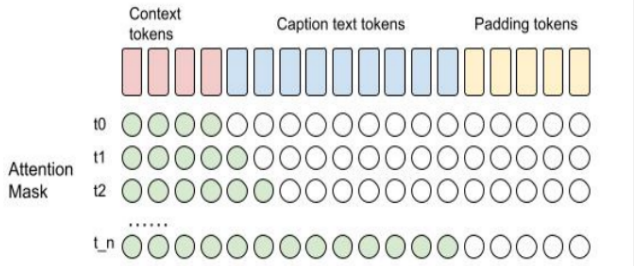


Figure 3. GPT-2 training example and attention mask

3.2. CLIP Based Model

To further benefit from open source large scale pre-trained models and to improve from the C3D approach. We build the improved model using CLIP as encoder backbone. Figure 1 bottom shows the architecture of our CLIP based model. The high level ideas are similar to the previous approach where CLIP encodes video frames and GPT-2 decodes the feature vector into natural language.

However, there are several important details applied in this model design in order for the backbone models to work for video captioning task. Firstly, given CLIP is designed to process one image at a time, it is not possible to process a stack of frames all at once. So we have custom logic to parallel process video frames individually. Given the limited accessible computation power, we only use 5 frames per video. Secondly, there is no fusion logic for different frames' CLIP feature vectors. We have a custom MLP layer that directly maps each frame feature to 4 GPT-2 tokens (4 x 5 frames = 20 tokens in total). The reason behind is that we want to defer the fusion logic to GPT-2 who has a lot more attention blocks and layers that can gradually fuse the information together. CLIP feature has a fixed dimension 512 per frame and the MLP layer projects it into 3072 dimensions through 3 fully connected layers and tanh activations (Figure 4). Finally, during fine tuning, we freeze the entire CLIP model because during one of our experiment, we realized that training with unfreezing CLIP can be too slow and can lead to potential over fitting problem. So we decided to freeze CLIP for all the subsequent training.

During initial training, we realized that there can be too many parameters and the end to end model is hard to train for this approach. After doing researches, we decide to

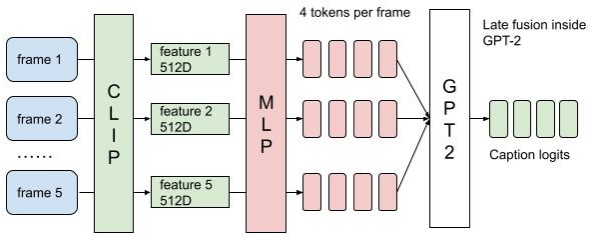


Figure 4. CLIP encoding based model (late fusion)

freeze the entire CLIP model parameters during training, which also allows us to pre-process each training video into a set of feature vectors to speed up training.

3.3. Beam Search

At text generation time, in addition to greedy search, we also implement the idea of beam search [6] as shown in Figure 5. Unlike greedy search where we took just the single best word at each position by looking at the softmax score of all possible words, beam search keeps track of the best M possible word sequences where M is called the beam width. Finally, the sequence with the highest conditional probability get selected.

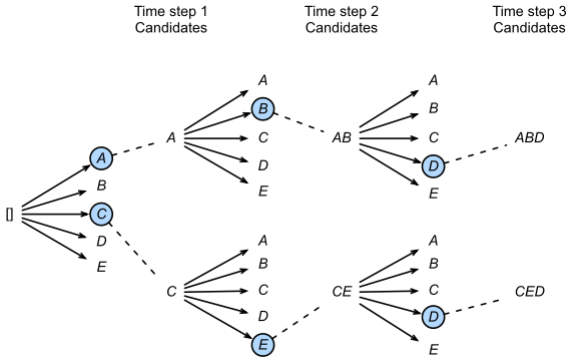


Figure 5. The process of beam search (beam width: 2)

3.4. Loss function

Usually, the loss function for captioning or text generation task is using the "shift by 1" trick, meaning at each time step, the loss is computed by comparing the predicted token (word) probability distribution with the next ground truth word and the loss function is usually cross-entroy loss. However, this approach does not work well for our case because we have 20 context tokens appended to the front of each ground truth caption token list and GPT-2 does not have an easy way to perform a "shift by 21" loss because the input tokens and output logits are designed to always have the same length. Specifically, output logits have shape

(batch_size, sequencelength, vocabsize).

To fix this issue, we have to manually calculate the loss disregarding the first 20 positions in the output logits. Our final loss is calculated by only comparing the captioning text part using cross entropy loss while disregarding the positions for context tokens or the padding tokens as shown in Figure 6.

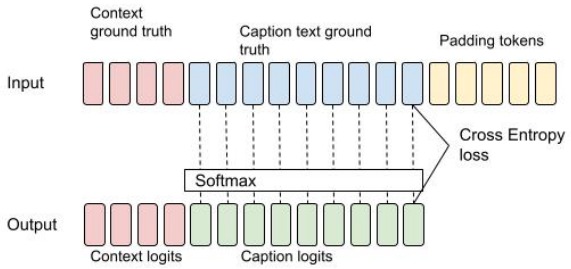


Figure 6. Loss calculation

4. Dataset

The dataset involved in this work are: COCO [28], MSVD [1] and MSR-VTT [2].

COCO stands for Common Objects in Context, as the image dataset was created with the goal of advancing image recognition. The COCO dataset is a large-scale object detection, segmentation, and captioning dataset published by Microsoft. In this work, we use the image captioning segment of 2014 version COCO dataset that contains about 83k images and corresponding captioning text. COCO dataset is quite clean and organized so we don't need to clean up the data. We use all the COCO data for pretraining.

MSVD (Microsoft Research Video Description Corpus) dataset consists of about 2k video snippets and 80k English sentences. It is collected by humans who watch a short video snippet and then summarize the action in a single sentence. Similar to the prior work [18] [29], we use the standard split: 1.2K training videos, 100 validation videos, and 670 test videos for performance measurement. The metrics in Experiments section is based on this split.

MSR-VTT (Microsoft Research Video to Text) is a large-scale dataset for the open domain video captioning, which consists of 9k video clips from 20 categories, and each video clip is annotated with 20 English sentences. MSR-VTT downloadable version comes with a train/test split, which has 6.5K training videos and 2.9K testing videos and we don't change this split,

Unfortunately, both MSVD and MSR-VTT appears to be a bit noisy and not very well maintained. The downloadable version of MSVD has a mismatching problem between videos and corresponding captioning text. Many of the sentences match to invalid video names or verse visa. We implemented python scripts to best match each video and text pair and remove the counterpart missing examples using regular expressions and hard coded rules. Also, we found that both datasets contain a few corrupted video files. We scanned all the videos using python library OpenCV to check the completeness of each training video and eliminated those broken examples to avoid impacting model training.

4.1. Data pre-processing

Extracting video frames on each training epoch is slow and repetitive. We pre-process MSVD and MSR-VTT using custom scripts into two versions to train the two different models. The first version contains 16 frames per video sampled with equal time interval. The interval is calculate as equally dividing the total number of frames by 15. Each sampled frame is down sampled to 112 x 112 using python PLI library. Then the 16 frames are stacked together to form a (16, 3, 112, 112) training example. In the end, we normalize the training data by subtracting mean from each color channel following common image data normalization practice. The second version is derived from the first version, where we further randomly sample 5 frames out of the 16 frames. Then each frame is encoded by the pretrained CLIP model to form a 512 dimension feature vector. The final shape for one training example in second version is (5, 512).

4.2. Data augmentation

We also applied two simple data augmentation methods to enrich the training set. Method 1: We combine each video with all provided captioning sentences to form multiple training pairs. For MSVD, each video has about 40 corresponding captioning sentences. We simply use all video-text combinations to form 40 training examples per video and the total number of training examples becomes about 40x1200 = 48k. Method 2: We randomly sample frames form each video clips with minimum and maximum time interval restrictions to generate more training examples per video. For each video, we try to generate 5 different sets of frames and the total number of training examples becomes 48k * 5 = 250k. Both augmentation methods are only applied on MSVD dataset due to time and storage limit.

5. Experiments

5.1. Best model

Our best performing model is CLIP encoding + custom MLP + GPT2 decoding approach with beam search, COCO image captioning pretraining, data augmentation and MLP and GPT2 fine tuning. It achieved 46.4 BLEU-4 score on MSVD test set and 36.5 BLEU-4 score on MSR-VTT test set which outperforms our baseline model (vanilla CNN + LSTM approach) by a large margin. It also achieved com-

Model	MSVD (BLEU-4)	MSR-VTT (BLEU-4)
CNN-LSTM (2015, baseline)	33.3	
PickNet (2018)	46.1	38.9
GRU-EVE (2019)	47.9	38.3
STG-KD (2020) [27]	52.2	40.5
SWIN BERT (state of the art)	58.2	41.9
CLIP+GPT2 (Our Model)	**46.39**	**36.45**

Table 1. Comparison with previous methods on the test split of MSVD and MSRVTT

parable performance to some previously considered state-of-art models (Table 1).

5.2. C3D and CLIP encoding

We first compared the performance of the two encoding approaches by only training on augmented MSVD training set. For C3D, we don't freeze any layers and perform training end to endly. For CLIP, we freeze all but the MLP layers. For both models, we use Adam [30] optimizer with β_1 = 0.8, β_2 = 0.999 and $\epsilon = 10^{-7}$. We used batch size of 16 and learning rate of 0.001. Both models are trained for 20 epochs with the models that have best validation scores kept. Our initial thoughts was that C3D model would perform better because it gets 16 frames per video whereas CLIP model only get 5 frames per video. However, it turns out CLIP encoding outperforms C3D encoding in our experiment even with all other layers freezed (Table 2). We believe that it is because of the power of CLIP's large scale pre-training. After training on 0.4 billion image-text pairs, CLIP is really able to capture the semantics of different video frames and compress the information into the CLIP vector space.

	Validation Set	Test Set
No pretrain C3D + GPT-2	34.08	30.67
No pretrain CLIP + GPT-2	**37.14**	**33.54**

Table 2. Comparison between C3D and CLIP encoding

5.3. Fine tuning different layers

For CLIP encoding model, since both CLIP and GPT-2 are pre-trained, we have the option to fine tuning either GPT-2 or CLIP or both or neither. In this experiment, we compared the performance for different fine tuning strategies with augmented MSVD training set. The learning rate is set to 0.0002 in this experiment because fine tuning usually uses a smaller learning rate and the results are shown in table 3.

Based on the results, training time is significantly longer if we fine tune CLIP on a single GPU. This observation meets our expectation because fine tuning CLIP needs CLIP to first process 5 frames for every training video. On the other hand, freezing CLIP allows us to pre-process training videos into 5 feature vectors which can save a lot of time. Regarding the performance, we observe that fine tuning GPT-2 can greatly improve performance on both validation and test set. We hypothesis the reason behind is that GPT-2 was pretrained on random text all over the internet, fine tuning can allow it to generate more captioning-like text using descriptive tone. Surprisingly, fine tuning CLIP does not give us much improvement on test set even though the validation set score goes up a bit. We think this is because CLIP is already doing a good job on capturing the image semantics, further fine tuning it may lead to over fit training and validation data. Consequently, we decided to freeze CLIP layers during training.

Fine tune Strategy	MSVD Validation	MSVD Test	Training Time per epoch
Only MLP	37.14	33.54	$\sim$ 6 mins
MLP + GPT-2	40.97	**37.70**	$\sim$ 7 mins
CLIP + MLP + GPT-2	**42.65**	37.38	$\sim$ 38 mins
CLIP + MLP	39.32	34.10	$\sim$ 38 mins

Table 3. Results for fine tuning different layers

5.4. Pretraining

Even though GPT-2 and CLIP are already pretrained models, they are trained on different tasks with different data distribution. Given the limited training video and caption text examples available, we experiment with pre-training with COCO image captioning dataset to see if it helps with video captioning task. For each image example in COCO, we first use CLIP to encode it to 512 dimension feature vector. Then we stack the same feature vector 5 times to fake a training video example. Another way to view this training example is that the video is very static and

have many similar frames. Last, we use these "fake" examples to train the model to generate the same captioning text as the image caption. During pretraining, we use learning rate 0.0002 and during final fine tuning, we use learning rate 0.0001. Due to time constrain, the pre-training is only performed for CLIP based model and the results are shown in table 4.

	MSVD Validation B-4	MSVD Test B-4	MSR-VTT Test B-4
No pretrain CLIP + GPT-2	40.97	37.70	28.60
Pretrain CLIP + GPT-2	**46.37**	**42.32**	**32.27**

Table 4. Comparison between pretraining and no pretraining

Clearly, pretraining boosts performance by a very large margin and it aligns with our assumption. After all, even with data augmentation, 1.2K training videos is still too small, the 80k COCO training data can better generalize the model and better tune the parameters towards captioning task. After pre-training and fine tuning, we don't believe our model overfits too much because the score difference between validation set and test is within reasonable range and we employ dropout in the MLP layer to further prevent over fitting.

5.5. Number of frames per video

We also briefly investigate the impact of number of video frames on performance at fine tuning time and the results are shown in table 5.

Number of Frames	MSVD Val B-4	MSVD Test B-4
1	37.85	35.32
2	40.04	39.32
3	44.98	39.92
4	46.35	41.78
5	**46.37**	**42.32**

Table 5. Comparison among different input frames

Since most videos in MSVD dataset do not involve a lot of motion, it is not very obvious that more input frames can provide better results. But for motion heavy videos, we believe the performance improvement with respect to number of frames will be more significant.

5.6. Beam search and greedy search

Next, we experiment with greedy search and different beam search widths at text generation step and the results are shown in table 6.

Method	MSVD Val B-4	MSVD Test B-4
Greedy Search	46.37	42.32
Beam search (width =3)	48.11	44.44
Beam search (width =4)	49.96	45.94
Beam search (width =5)	**50.05**	**46.39**

Table 6. Comparison among different text sample methods

We find that using beam search can further improve the performance on top of greedy search. The beam width 5 benefits the performance most. Beam width larger than 5 may be much slower and the improvement is flatten out.

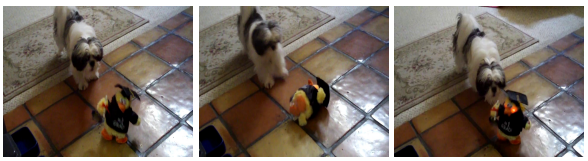
5.7. Results

In the end, we would like to show some impressive results our best model generates from unseen test examples as well as for some random short videos we found online to better understand what the model has learnt.

Input Video:

Output:
a dolphin is swimming in the water.

Input Video:



Output:
a dog is playing with a toy.

Input Video:

Output:
Two girls are dancing.

Input Video:

Output:
a woman is cleaning a tub.

Input Video:

Output:
A man is climbing a rock wall.

Form the above examples, we can tell that the best model is able to caption videos under different scenarios and for various contents. For the first example, even though it is hard to "see" what's under the water, the model is able to guess it is likely to be a dolphin. For the second and third examples, the model is able to tell there are multiple important objects (a dog and a toy) or multiple people (two girls) appear in the video. Moreover, the model is able to generate a high quality caption even if the third example is in black and white. For the last two examples, the model is able to capture different human motions like cleaning and climbing which is very impressive.

6. Conclusion and Future Work

In conclusion, we implemented two variations of encoder-decoder architecture for video captioning using C3D encoding or CLIP encoding and GPT-2 decoding. We experiment with different training strategies and various methods to improve the model performance. Specifically, we discover that conducting pretraining with image captioning dataset can greatly improve the performance for video captioning. We also find that fine tuning more layers with a larger dataset can usually improve performance. But since CLIP is already very good at modeling image semantics, further fine tuning it could lead to over fitting. In addition, beam search and more input frames can help with performance to some extend.

Video captioning used to be a very complicated task that requires a lot of model training and fine tuning. Also, lacking of high quality training data is a major blocker in video related model training. Nowadays, thanks to large scale pretrained transformer based models and transfer learning, it is possible to fine tune powerful open source models on various tasks and achieve impressive performance on modest computation power.

Due to time and resource constraint, we didn't explore MSRV-TT data set a lot. Also the input to our best model is only 5 frames per video. In the future, we would like to scale up the input frame size to CLIP and experiment more with dataset that contains videos with more motions. One idea we have is that instead of using a MLP to connect encoder and decoder, we would like to try if using several self attention blocks can combine information more effectively. Moreover, we didn't get a chance to do too much hyper-parameters search, We would like to explore better hyper-parameters to further improve the performance.

References

[1] David Chen and William Dolan. Collecting highly parallel data for paraphrase evaluation. In ACL, 2011.

[2] Jun Xu, Tao Mei, Ting Yao, and Yong Rui. Msr-vtt: A large video description dataset for bridging video and language. In CVPR, 2016.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In CVPR, 2016.

[4] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision. arXiv:2103.00020, 2021.

[5] Alec Radford, Jeff Wu, Rewon Child, D. Luan, Dario Amodei and Ilya Sutskever. Language Models are Unsupervised Multitask Learners. 2019.

[6] Sam Wiseman and Alexander M. Rush. Sequence-to-Sequence Learning as Beam-Search Optimization. arXiv:2103.00020, 2021.arXiv:1606.02960, 2016.

[7] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In ACL, 2002.

[8] Video Will Grow to 82

[9] Subhashini Venugopalan, Huijuan Xu, Jeff Donahue, Marcus Rohrbach, Raymond Mooney and Kate Saenko. Translating Videos to Natural Language Using Deep Recurrent Neural Networks. arXiv:1412.4729. 2014.

[10] Sepp Hochreiter and Jurgen Schmidhuber. LONG SHORT-TERM MEMORY. 1997.

[11] Yangyu Chen, Shuhui Wang, Weigang Zhang, and Qingming Huang. Less is more: Picking informative frames for video captioning. In ECCV, 2018.

[12] Nayyer Aafaq, Naveed Akhtar, Wei Liu, Syed Zulqarnain Gilani, and Ajmal Mian. Spatio-temporal dynamics and semantic attribute enriched visual encoding for video captioning. In CVPR, 2019.

[13] Qi Zheng, Chaoyue Wang, and Dacheng Tao. Syntax-aware action targeting for video captioning. In CVPR, 2020.

[14] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau and Yoshua Bengio. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. arXiv:1409.1259. 2014.

[15] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić and Cordelia Schmid. ViViT: A Video Vision Transformer. arXiv:2103.15691. 2021.

[16] Jie Lei, Licheng Yu, Tamara L Berg, and Mohit Bansal. Tvr: A large-scale dataset for video-subtitle moment retrieval. In ECCV, 2020.

[17] Gedas Bertasius, Heng Wang and Lorenzo Torresani. Is Space-Time Attention All You Need for Video Understanding? arXiv:2102.05095. 2021.

[18] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin and Baining Guo. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. arXiv:2103.14030. 2021.

[19] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In ICLR, 2020.

[20] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. IJCV, 2015.

[21] Jacob Devlin, Ming-Wei Chang, Kenton Lee and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805. 2018.

[22] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. arXiv: 1909.11942, 2020.

[23] Thomas Wolf, Julien Chaumond, Lysandre Debut, Victor Sanh, Clement Delangue, Anthony Moi, Pierric Cistac, Morgan Funtowicz, Joe Davison, Sam Shleifer, et al. Transformers: State-of-the-art natural language processing. In EMNLP: System Demonstrations, 2020.

[24] CLIP Official Open Source Link. https://github.com/openai/CLIP

[25] C3D Open Source Project Link. https://github.com/DavideA/c3d-pytorch

[26] GPT-2 Open Source Project Link. https://github.com/parvathysarat/gpt2-text-generation

[27] Boxiao Pan, Haoye Cai, De-An Huang, Kuan-Hui Lee, Adrien Gaidon, Ehsan Adeli, and Juan Carlos Niebles. Spatio-temporal graph for video captioning with knowledge distillation. In CVPR, 2020.

[28] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick and Piotr Dollár. Microsoft COCO: Common Objects in Context. arXiv:1405.0312. 2015.

[29] Bairui Wang, Lin Ma, Wei Zhang, Wenhao Jiang, Jingwen Wang, and Wei Liu. Controllable video captioning with pos sequence guidance based on gated fusion network. In ICCV, 2019

[30] Diederik P. Kingma, Jimmy Ba. Adam. A Method for Stochastic Optimization. arXiv: 1412.6980. 2017.