Sequence to Sequence Models for Generating Video Captions

Rafael A. Rivera-Soto
Stanford
I450 Serra Mall, Stanford, CA 94305
rivera43@stanford.edu

Juanita Ordóñez
Stanford
I450 Serra Mall, Stanford, CA 94305
ordonez2@stanford.edu

Abstract

Automatically generating natural language descriptions for videos poses a challenging problem for the computer vision community. In contrast to many of the image classification and object detection tasks, this problem requires a level of video understanding wherein a visual scene can be effectively translated into a natural language description of the events occurring therein.

In this work, we explore sequence to sequence models mainly used in neural machine translation and apply them in our context. The modern RESNET-50 and VGG-16 convolutional neural network architectures are used in conjunction with the LSTM recurrent neural network model. Our models are trained on video to text pairs and learn to associate a variable sized sequence of frames to a variable sized sequence of words. In this way, they learn to associate a sequence of visual features to a natural language description of such. The performance of our models are evaluated on the Microsoft Video Description Corpus (MSVD) and are quantified using the METEOR [8] metrics.

1. Introduction

Image captioning: the description of the content of images using natural language has been the subject of many research endeavors. A logical extension of this research problem is that of video captioning, wherein the goal is to generate a description of an entire video using natural language. Some of the applications of video captioning include but are not limited to: human-robot interaction, description of videos to the blind, video indexing and information retrieval. While image captioning requires a variable length output sequence, video captioning requires both a variable length input and a variable length output. As such, we explore the effectiveness of sequence to sequence models mainly used in neural machine translation to tackle this problem. The model needs to understand how various scenes in the video relate to words in natural language, this problem is thus an intersection of two very important fields in artificial intelligence: Computer Vision and Natural Language Processing.

Existing approaches to this problem can be divided in two main categories: template-based language models and sequence learning methods such as the one explored in this work. The former pre-defines a set of templates following specific grammar rules. This approach is highly dependent on the templates defined thus the generated sentence will always have a constant syntactic structure that follow that of the templates. In contrast, sequence learning models directly translate video content to a sentence without relying on pre-defined templates.

This works explores general sequence to sequence models for the generation of video captions. More specifically, our models takes as inputs a sequence of video frames which are then fed to a pre-trained convolutional neural network in order to extract the visual features. These visual features are fed into a recurrent neural network that is responsible for encoding information about the video sequence. This encoded information is then used by a decoder whose responsibility is to generate a natural language description. In this way, were able to handle both variable length inputs and outputs as well as two distinct types of data: images and text.

2. Related Works

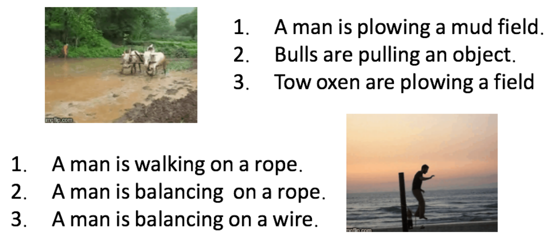
Early work annotated arbitrary short videos using off-the-shelf visual detectors without the engineering efforts required to build a domain-specific model. By a two-stage pipeline that first identifies the semantic content, such as subject, verb and object from a video (SVO); using a combination of visual object and activity detectors along with text-mined knowledge the semantic content probability for sentence generation phase is calculated, where the subject, verb and object are selected and plugged into a fixed template. [7] There are two major limitations to this approach. The first limitation is that the model is dependent on the fixed template which can result in having a small range of sentences outputted by the model, compared to human description it will end up lacking in richness. The second limitation is that they typically involve training individual classifiers to identify candidate objects, actions, and scenes. They then used a probabilistic graphical model to combine the visual confidences with a language model in order to estimate the most likely content in the video, which is then used to generate a sentence, thus it requires selecting a set of relevant objects and actions to recognize [18].

Other work adapted the recurrent neural network models, LSTM [5] and GRU [3], to act as decoder of the video clip and learned to generate the natural sentences instead using a specified template. One of the first approaches used pre-trained VGG-16 convolutional neural network, trained on the ImageNet corpus, to extract the visual information from each frame then mean pool to generate a single vector that describes the entire the video snippet. Then the video feature vector was used to initialize a stacked LSTMs hidden state where natural language is then generated. Due to the mean pooling of the videos frames features, temporal information is lost[19].

To take advantage of the frames sequence information, LSTM were introduced as sequence to sequence model where parts of it act as an encoder and the other part as a decoder. This idea was inspired by machine translation work, where the encoder receives a sentence in a language, while the decoder tries to translate the input sentence into another language [15]. For the video captioning problem, this approach was introduced as S2VT [18] where the first LSTM was used to read and encode a sequence of video frames and a second LSTM, conditioned on the last hidden state of the first, was used to generate a sentence. This approach was able to get a METEOR score of 29.8 on the MSVD [2] dataset whereas the mean pooling approach achieved a METEOR score of 27.7.

The datasets used today to tackle this problem tend to have scarce source of the text data, which can result in a restricted language model based on a small vocabulary. To tackle to this issue, instead learning a word embedding layer from scratch, the model initializes the embedding layer with weights pre-trained with Glove [14] on a large web unlabeled corpus, 6B tokens from Gigawood and Wikipedia 2014. Significant improvements were found on human evaluations of grammar while modestly improving the overall descriptive quality of sentences on the dataset [17].

Video scenes contain several shots that, although temporally consistent, have a different appearance. In this case we want to prevent the network from mixing the memory of the different shots [1]. Different Architecture of LSTM have been tested, currently the state of the art results on the MSVD dataset is the Hierarchical LSTM encoder [12]. Instead of the taking all the time steps output from the first LSTM into the second LSTM, only a chunk of the certain time steps are used by the second LSTM, which is decides with the output of the attention layer. This model has a METEOR score of 33.9.


1. A man is plowing a mud field.
2. Bulls are pulling an object.
3. Tow oxen are plowing a field

1. A man is walking on a rope.
2. A man is balancing on a rope.
3. A man is balancing on a wire.

Figure 1. Some samples from the MSVD Dataset

3. Dataset and Features

We perform all our experiments on the MSVD dataset. This video corpus is a collection of the 1970 YouTube snippets. The duration of each clip is between 10 seconds to 25 seconds, typically depicting a single activity or a short sequence [18]. Dataset descriptions come in multiple languages including English and are annotated by crowdsourcing Amazon Mechanical Turk. In general, the MSVD dataset has 80,827 sentences, 567,874 tokens, 12,594 vocabulary size, 10.2 average length of video clip and about 40 descriptions per video. We used previous work split of train, test, and validation set [1] along with the pre-processed captions, where characters are lowercase and punctuation was removed, we picked the longest caption per video. For our work, we tokenized the text data using the Natural Language Tool Kit, NLTK, word tokenizer [2]. We later pre-processed the text data into a sequence of integers, where each integer represents the index of the word in the vocabulary.

There are a different number of raw frames per video clip. It has been proved that LSTMs show good learning capabilities on sequences which are between 30 and 80 frames long. [11] And on the set of experiments from S2VT the author sampled every 10th of the frames, thus we sampled every tenth of the frame for each video clip. The end result on average is 40 frames per video, but each video has a different number of frames. During training, we don't truncate the number frames, to take advantage of all the video information from start to end. Some samples of the dataset can be seen in Figure 1.

[1]https://www.cs.utexas.edu/~vsub/naacl15_project.html
[2]http://www.nltk.org/

4. Methods

Our models take as input a set of video frames $(x_1, x_2, ..., x_n)$ and output a sequence of words $(y_1, y_2, ...y_m)$ representative of a natural language description. This problem is very similar to machine translation but differs in that the input is a sequence of frames instead of a sequence of words.

4.1. Convolutional Neural Network for Visual Feature Extraction

The Resnet-50 [4] convolutional neural network architecture is used as a visual feature extractor prior to feeding the images to the encoder/decoder network. This architecture explicitly reformulates the layers as learning the residual function $F(x)$ which allows for efficient optimization when training deeper neural networks. These networks currently achieve state of the art performance on the ImageNet dataset. The network takes as input a 256x256x3 dimensional image and outputs a 2048-dimensional vector representative of the visual features of the image. This model was pre-trained on the ImageNet dataset prior to its use in this work. The function of this network can be interpreted as summarizing the most salient features of each image prior to feeding it to the encoder network. The process of extracting visual features using the pre-trained convolutional neural network is shown in Figure 2.

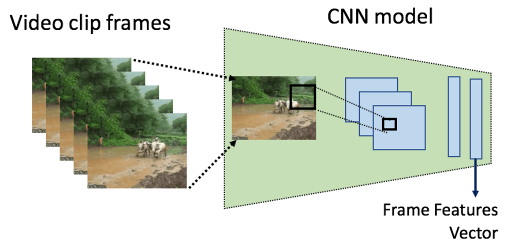


Figure 2. Extraction of Visual Features using a pre-trained CNN

4.2. Long Short-Term Memory Networks for Encoding and Decoding

In order to handle both variable-sized inputs and variable-sized outputs we explore the LSTM recurrent neural network architecture. The entire input sequence $(x_1, x_2, ..., x_n)$ is first encoded, summarizing the video into one hidden state vector which is used to decode a natural language description of the features shown.

For some input x_t at timestep t, the LSTM computes a hidden state h_t, a memory cell c_t and the next output o_t. The equations describing the LSTM are:

$$\begin{aligned} i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \\ f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \\ o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \\ g_t &= \tanh(W_{xg}x_t + W_{hg}h_{t-1} + b_g) \\ c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\ h_t &= o_t \odot \tanh(c_t) \end{aligned} \tag{1}$$

where σ is the sigmoid non-linearity, $\odot$ is an element-wise product and the weight matrices W and biases b are learnable parameters. During training, our encoder network takes an input sequence $(x_1, x_2, ..., x_n)$ and computes some sequence of hidden states $(h_1, h_2, ...h_n)$ and cell states $(c_1, c_2, ..., c_n)$.

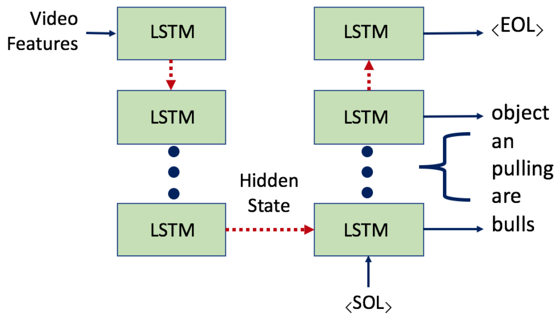


Figure 3. Encoder-Decoder Network

4.3. Word Embeddings

All the words that appeared in our dataset were tokenized and represented using an index that represents its position in the vocabulary. We add to the vocabulary the <SOL> and <EOL> tokens which signify the beginning and end of a caption. Another token: <UNK>, is added to represent uncommon words in our vocabulary.

In order to input these words into our models, we use an embedding layer to convert each of our words $(y_1, y_2, ..., y_m)$ to a vector representation $(w_1, w_2, ..., w_n)$. Our models learn these word embeddings using backpropagation. These word embeddings allow the model to learn the difference and relationship between the words in our training corpus.

4.4. Baseline

Our baseline approach is a simplified approach of the network described in [19]. We use the mean pooled VGG-16 features gathered by the authors and feed these in as the initial hidden state of a one-layered LSTM. The network then decodes an entire caption based on this single feature vector. The network can be seen pictorially on FIGURE N.

During training, the network minimizes the negative log-likelihood loss of the predicted output sequence given the input vector. The gradients are distributed backwards us-

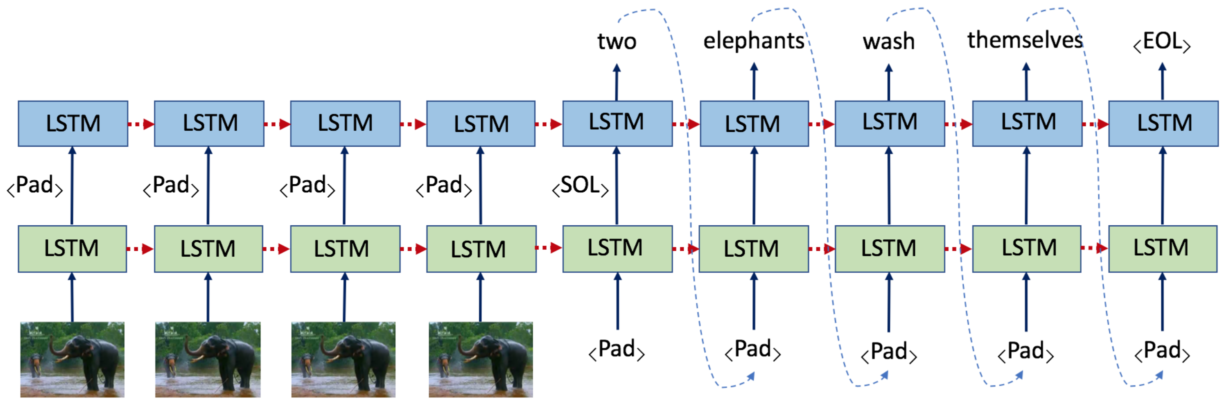


Figure 4. Stacked Encoder-Decoder Network

ing backpropagation through time and are updated using the Adam optimizer [6].

4.5. Encoder-Decoder Network

Our first model consists of a one layered LSTM that is used for both encoding the video frames and decoding a caption. We use the word embedding technique as a means of representing each word of our vocabulary in vector space.

This design puts the responsibility of both encoding the input features and decoding the natural language description in one set of weights which complicates the convergence of the network. This model is illustrated in Figure 3

4.5.1 Training Time Behavior

During the encoding stage, the visual features of each frame $(i_1, i_2, ..., i_n)$ is extracted using the RESNET-50 convolutional neural network. This results in a sequence of video features $(x_1, x_2, ..., x_n)$ which are fed into the LSTM, encoding the visual features into a hidden state representation h_t. Note that we don't calculate any loss during the encoding stage.

After we exhaust every visual feature of a given video, the model enters the decoding stage. The start of this is stage is signaled by feeding in the <SOL> token as input to the model. For each decoding timestep we feed in the ground truth word of the caption corresponding to the current video clip being analyzed. Each hidden representation h_t is passed through a linear layer which results in scores for each word in our vocabulary, this enables us to calculate the negative log-likelihood loss and use backpropagation through time to update all of our parameters.

4.5.2 Testing Time Behavior

The test time behavior of the model is mostly unchanged, except that we don't have the ground truth label. Instead, we pick the word with the highest score and feed it in to the model as input. We stop generating the caption either when the <EOL> token is produced by the model or a modifiable upper limit is reached.

4.6. Stacked Encoder-Decoder Network

We were inspired by the work done by [18] and decided to test this model for the video captioning task. The main difference between this model and the previous is the use of two stacked LSTM networks. Using a stacked approach, allows the first LSTM to focus more on the visual encoding task while the second LSTM focuses on the decoding task while still allowing parameters to be shared across both sides of the model. The model is illustrated in Figure 4.

4.6.1 Training Time Behavior

We process each video frame in the same manner described in 4.5.1 prior to feeding it in to the model. The first LSTM, takes in the video features and produces a hidden representation h_t. This hidden representation is then padded with zeros and sent to the second LSTM for further encoding. Similar to the previous model, no loss is calculated during the encoding stage. Once we exhaust every visual feature, the model enters the decoding stage. The start of this stage is signaled by feeding in the <SOL> token as input to the model. During this stage the first LSTM takes in as input a 2048 dimensional vector of zeros. The hidden state of the first LSTM is concatenated with the word embedding of the

I.

Ground Truth: a man slices tomatoes on a kitchen counter
Model: a man is slicing a peeled onion into four uniform slices of about 12 thickness each

II.

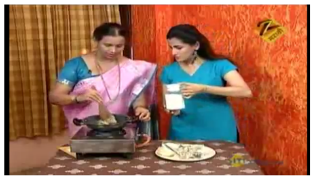
Ground Truth: a man is playing a guitar on a stage outside
Model: a man is standing in front of a microphone holding something in his hands and he appears to be singing or talking into the microphone

III.

Ground Truth: a woman is standing on the back of a motorbike while the male driver lifts the bike on to one wheel
Model: a woman is standing on a bicycle on a bicycle up a car off a woman in a white colored

Ground Truth: two elephants wash themselves in a river
Model: a panda climbs up a threetiered cake and another panda is sitting and looking at the cake

Ground Truth: a boy made a skull talk by moving the mouth with his hands
Model: a man is cutting a boiled potato into small pieces



Ground Truth: one woman is cooking some kind of coated meat in a pan on a hotplate
Model: a woman is standing on a bicycle on a bicycle on a block of a car off a car off

Figure 5. In the left, qualitative results for the mean pool model. In the middle, results for the stacked encoder-decoder model. In the right, results for the single encoder-decoder model.

	RNN-Hidden-Size	Embedding-Size
Mean Pool Model	512	128
One-Layered Encoder-Decoder	512	128
Stacked Encoder-Decoder	1024	128

Table 1. Hyper-Parameters used during the experiments.

current word being analyzed and send to the second LSTM for decoding. The hidden representations of the second LSTM are then passed through a linear layer which results in the scores for each of the words in our vocabulary. These scores allow us to calculate the negative log-likelihood loss and use backpropagation through time to update all of our parameters.

4.6.2 Test Time Behavior

The test time behavior of this model is identical to that discussed in 4.5.2, except that instead of one LSTM we have two thus extra care must be taken to ensure that the inputs of each of the LSTM's are correct.

5. Experiments

All of the models aforementioned were implemented from scratch using the Pytorch [3] deep learning library. We ran experiments with each of the models on the MSVD dataset using 1200 videos for training, 100 for validation and 670 for testing. This dataset split has been used in previous video captioning work as mentioned in 3.

The hyper-parameters are outlined in Table 1. A wide range of learning-rate and learning rate decay values were used, the results show those that did the best on the validation and test set.

5.1. Evaluation Metrics

To evaluate the generated sentences, we use the METEOR scores against all ground truth sentences. This score is computed based on the alignment between a given model generated sentence and set of ground truth sentences. Alignments are based on exact, stem, synonym from WordNet [10], and paraphrase matches between words and phrases. Once all the alignments have been produced between the model generated sentence and the ground truth sentences for this pair of translations is computes F-Mean, which combines the precision, P, and recall, R. The recall is a ratio between the number of mapped unigrams from the reference to ground truth unigrams over the total number of the unigrams generated by the model. Similarly the precision calculates the the ratio with respect to the total number of unigrams in the ground truth [8]. METEOR uses harmonic mean of P and 9R.

[3]`http://pytorch.org/`

$$Fmean = \frac{10PR}{R + 9P} \quad (2)$$

METEOR computes a penalty for a given alignment, to take into consideration longer matches. Where the penalty increases as the number of chunks from model generated sentence increases. Where later the METEOR score is computed as follows:

$$Penalty = 0.5 * (\frac{\#chunks}{\#unigrams\_matched}) \quad (3)$$

$$Score = Fmean * (1 - Penalty) \quad (4)$$

Even though other metrics such BLEU [13], ROUGE [9], and CIDEr [16], the authors of CIDEr found that METEOR in terms of the consistency with human judgment is always better than BLEU, and ROUGE [12]. The authors of CIDEr reported that CIDEr and METEOR are always more accurate, especially when the number of references captions are low. CIDEr is comparable to METEOR when the number of ground truth sentence are large [18].

We utilize the Microsoft COCO evaluation scripts to obtain all the results reported in this paper, which makes our results directly comparable with the previous works[4].

6. Discussion

Our results are shown on Table 2. Comparing the METEOR metric, we can see that the stacked encoder-decoder network performed above the other two models. We hypothesize that this is due to the fact that each layer of the model is able to take care of one particular task: encoding or decoding. Separating the tasks in this way, allows the development of specialized parameters for understanding video representations and producing natural language description therein. It also allows for better network convergence since the network doesn't have to juggle between two tasks with the same set of weights. Not only did this model perform better quantitatively, but it also generated better qualitative results as can be seen in Figure 5; here, we see that the model was able to generate a caption that is more descriptive than the ground truth caption.

The mean pool model performed better than the single layer encoder-decoder network as seen in Table 2. We hypothesize that this due to the fact that the model only needs to associate a single visual feature vector to a caption while the single layered model must juggle the encoding and decoding tasks with the same set of weights. A lot of the sentences generated by the single layered encoder-decoder network were incoherent and repetitive, this might be an artifact of juggling two tasks with the same set of weights.

[4]`https://github.com/tylin/coco-caption`

Model	Test Score	Validation Score
Stacked Encoder-Decoder	20.1	20.7
Single Encoder-Decoder	17.1	18.0
Mean Pool Model	17.4	17.9

Table 2. Results in validation and test set for each of our models. The METEOR metric was used to quantify the performance.

7. Conclusions

In summary, we explored a simple mean pool model and two sequence to sequence models commonly used in video captioning and compared their performance both quantitatively and qualitatively. We experimentally validated the instability of a single layer encoder-decoder network in performing the video captioning task. Although these single layered architectures are used in machine translation, we believe that the intersection of two types of data: visual and text complicates the problem and warrants the use of a two-layered approach. An observation made in recent work [1] [12] points out that many of these videos shift their visual center. Because of this, modern architectures that achieve state of the art results have explored more complicated encoding mechanisms where this shift in visual center is taken into consideration, coupled with an attention layer mechanism in a hierarchical fashion.

References

[1] L. Baraldi, C. Grana, and R. Cucchiara. Hierarchical boundary-aware neural encoder for video captioning. *CoRR*, abs/1611.09312, 2016.

[2] D. L. Chen and W. B. Dolan. 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*, HLT '11, pages 190–200, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[3] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio. Gated feedback recurrent neural networks. *CoRR*, abs/1502.02367, 2015.

[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[5] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, Nov. 1997.

[6] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[7] N. Krishnamoorthy, G. Malkarnenkar, R. Mooney, K. Saenko, and S. Guadarrama. Generating natural-language video descriptions using text-mined knowledge. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, AAAI'13, pages 541–547. AAAI Press, 2013.

[8] A. Lavie and A. Agarwal. Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07, pages 228–231,

Stroudsburg, PA, USA, 2007. Association for Computational Linguistics.

[9] C.-Y. Lin. Rouge: a package for automatic evaluation of summaries. July 2004.

[10] G. A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, Nov. 1995.

[11] J. Y. Ng, M. J. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. *CoRR*, abs/1503.08909, 2015.

[12] P. Pan, Z. Xu, Y. Yang, F. Wu, and Y. Zhuang. Hierarchical recurrent neural encoder for video representation with application to captioning. *CoRR*, abs/1511.03476, 2015.

[13] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: A methhod for automatically evaluating machine translation, 2002.

[14] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[15] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014.

[16] R. Vedantam, C. L. Zitnick, and D. Parikh. Cider: Consensus-based image description evaluation. *CoRR*, abs/1411.5726, 2014.

[17] S. Venugopalan, L. A. Hendricks, R. J. Mooney, and K. Saenko. Improving lstm-based video description with linguistic knowledge mined from text. *CoRR*, abs/1604.01729, 2016.

[18] S. Venugopalan, M. Rohrbach, J. Donahue, R. J. Mooney, T. Darrell, and K. Saenko. Sequence to sequence - video to text. *CoRR*, abs/1505.00487, 2015.

[19] S. Venugopalan, H. Xu, J. Donahue, M. Rohrbach, R. J. Mooney, and K. Saenko. Translating videos to natural language using deep recurrent neural networks. *CoRR*, abs/1412.4729, 2014.