CS230

Captioning Video Events with NetVLAD

Julio A. Martinez
ICME, Stanford University
juliomz@stanford.edu

Parker Miller
Law School, Stanford University
pmiller0@stanford.edu

Songze Li
ICME, Stanford University
songzeli0@stanford.edu

Abstract

Understanding video content has a variety of use cases from video query search to ad content matching. In this project we explore captioning events that occur in different segments of video. We start with a baseline implementation that was done for our other course (CS230) and use a different attention module known as NetVLAD to pool preprocessed C3D features that represent video segments. We are able to show NetVLAD yields better performance than our baseline by BLEU score on a small subset of the Activity Net Captions dataset.

1 Introduction

The objective in this work is to describe events that occur in videos using natural language. We seek to do this with a variant of learnable pooling known as NetVLAD. Several applications of this work are interesting, such as being able to search a video for a specific segment wherein a particular action occurs, or match the content of the video with ads that are relevant.

2 Related work

Activity recognition has historically involved using hidden Markov models and discriminative SVM models to describe events with a fixed label set [8, 9, 15, 2]. Some have used deep learning to further study video event actions [16]. In our approach we use similar methods, but our goal is to describe video events with natural language using a larger vocabulary. Traditionally, temporal action proposal methods used a sliding window approach [14], but later evolved to build proposal models that used dictionary learning [6] or RNN architecture [18] to locate potential action segments. These methods required processing for each video frame in the sliding window. Rather than use this sliding window approach, we output proposals for every time-step in a single pass of the video. We believe this model is more efficient. There have been various approaches to video captioning. Notably, a paper recommended describing videos with paragraphs, where an RNN generates hidden vectors that then initialize lower-level RNNs that generate individual sentences [17, 7]. Our work is similar to this approach, but we attempt to caption events using a different dataset [1], ActivityNet Captions.

3 Dataset and Features

For both courses we use the same dataset ActivityNet Captions. Although the dataset contains 20k videos which amount to 849 hours and 100k captions, we use approximately 10% of the dataset with 2000 videos to train with, 250 for dev, and 250 for testing. Since the entire dataset is large, we make sure to select videos in dev and test set whose captions have words with high frequency of appearance in the 2000 training videos.

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

3.1 Preprocessing

?For preprocessing we simply group the C3D frames by video and by event. Thus each video has a sequence of video segments which contain 500-dimensional C3D-PCA features representing the frames in the segment. To give context, these C3D features are extracted from a C3D model and then reduced using PCA. This is provided on the Dense Captioning Events in Videos webpage. Each segment outlining an event has a distinct duration, thus the number of frames and hence feature representation that correspond to an event varies. We limit the number of events to 10, and in each event we limit the number of feature frames to 2000, meaning each event could be up to 2000 frames long. For segments not meeting this requirement padding of zeros is added at the end. The number of words in a caption (the sentence length) is limited to 30. For captions not meeting these two requirements, we truncate or pad the sentence at the end accordingly. Finally words found in the label captions corresponding to these event segments are tokenized, meaning each word is mapped to a unique integer as done in [10]. The set of words plus three additional tokens, one to indicate <start> and one to indicate <end> of a caption, and padding <pad>, make up the vocabulary, which has cardinality 10,999.

4 Methods

This project involved implementing two network, first the baseline method found in [11] and second is a variation that uses NetVLAD (which is the portion done for this course, baseline was implemented to satisfy CS231N). The general pipeline goes as follows: Input: C3D Features $\rightarrow$ NetVLAD $\rightarrow$ 2-Layer LSTM $\rightarrow$. For understanding the baseline, see report from CS23N. The main difference is instead of learnable pooling as will be discussed they use max pooling over frame features and then attention to arrive a features vector that represent the segments.

4.1 Baseline Attention Module

The first step is the Attention step. In the baseline, C3D features for the video segments are max pooled temporally. Thus a sequence of 2000 500-dimensional features representing a segment are pooled into a single vector using the maximum values. These are then denoted h_i for the ith video segment. The max pooling is not mentioned in [11]; however, after meeting with Ranjay Krishna, it was clear this is the approach taken in that work. Sufficeth to say that the resulting vector is a concatenation of future, past, and present segment information that is of dimension 1500, 500 for past, present and future each. See CS231N paper for details.

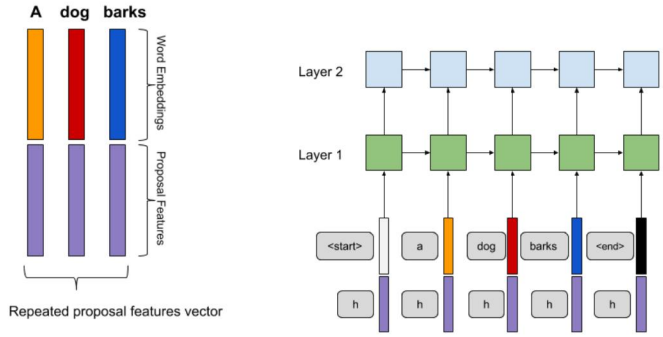
4.2 NetVLAD Attention Module

NetVLAD (i.e neural network of vector of locally added descriptors) is a form of attention and learnable pooling that has been used successfully in video-level tagging [12]. NetVLAD learns clusters in the input features and computes residuals from each temporal input feature to the cluster centers c, and multiplies this residual to a softmax. The number of clusters in NetVLAD is a hyperparameter and each cluster center is randomly initialized. Therefore the resulting clusters are found from the backprop step. Hence the best cluster centers for the model are found through training as discussed in [4]. These computed values are then aggregated temporally. This results in a matrix $VLAD$ which we flatten and denote the result as X

$$VLAD(j,k) = \sum_{i}^{N} softmax(h_i)(h_{ij} - c_{kj}) \tag{1}$$

$$CG(X) = \sigma(WX + b) * X. \tag{2}$$

Context Gating, an activation function, is a combination of an element-wise sigmoid with element-wise product to flattened $VLAD$ features X, as shown below. This introduces yet another quadratic relationship between the features. As opposed to the attention from baseline where video segment features are maxpooled, here NetVLAD and context gating are used to aggregate these feature frames in a nonlinear way. The output is a pooled representation of dimension 500 that is learned, as opposed to 1500 in the baseline. Figure 1 shows the attention module using NetVLAD and context gating.



(a) Tiling Concatenation (b) 2-Layer LSTM Input

Figure 2: 2 Figures side by side

4.3 Captioning Module

The output from the attention module is the input to the captioning module. The output from NetVLAD is a 500-dimensional vector, one which represents the entire event in a segment of a video.

In addition, word embeddings are used with 512 units. This is done with the model end-to-end. Since each caption is limited to length 30, each segment corresponded to a caption of at most 30 words.

A 2-layer LSTM network is used to learned captioning structure. The LSTM run for 30 timesteps since these are the lengths of each caption. At each time step the LSTM is fed a concatenation of the word embedding and the NetVLAD output for the entire video segment. See figure 2(a) to see the concatenation process.

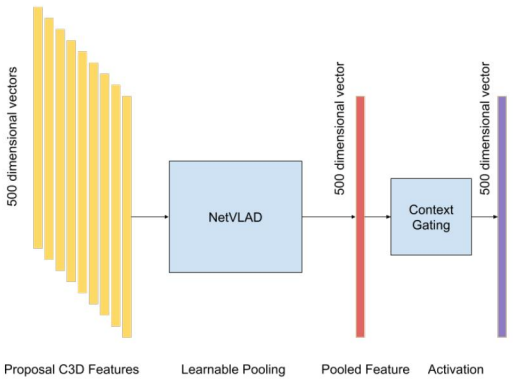


Figure 1: NetVLAD with Context Gating Pipeline

This concatenation is done by first tiling the segment representation 30 times, one for each word in the caption. This way each word embedding is concatenated to the same vector. This allows the network to have the same information of the entire video segment at each step producing words based on the visual information as well as words already predicted. The 2-Layer LSTM is set to have 512 hidden units and the hidden states are initialized to zero. A subsequent softmax layer is computed over each step of the 2-layer LSTM. This softmax layer has 10,999 units, in order to produce a score for each word in the vocabulary (i.e. size of vocab is 10,999). A diagram is found in figure 2(b) to illustrate the input at each time step during training to the two layer lstm.

4.4 Greedy Caption Generation

For this course we implemented greedy caption generation. During training time, all dimensions are fixed except for the length of captions in Tensorflow which we set at "None", this way we can iteratively increase the length as we generate captions. Thus at the first time step a concatenation of a start token and the segment feature are fed. The softmax of the LSTM produces scores for each word in the vocabulary. The next word is chosen by simply taking the maximum score, and the next input is the first input concatenated horizontally with the predicted output. This continues until an $< end >$ word embedding is reached. Keep in mind the LSTM is trained such that each caption starts with a $< start >$ word embedding. This mode generation allows us to output captions of different lengths which we can then compare to the ground truth.

Caption Generation is used only during the test stage, so that we can generate word by word based on the past corpus.

In addition to this greedy selection of words, we also implemented beam search for CS231N. Briefly, beams search expands upon greedy search by choosing a most likely sequence of words instead of just the most likely word one step at a time. We choose a beam size of 3 to see an improvement but still computationally achievable within a reasonable amount of time, since longer sequences are more expensive to compute.

4.5 Optimization and Loss Function

Cost function used is the mean cross entropy. We implemented our own cross entropy function as well to reduce effect of padding. After testing we found little effect from padding and decided to use tensorflow cross entropy loss. Initially longer caption sequences and larger number of video segments had more of an effect due to padding. By reducing these sequences, we were able to train our model much more effectively. Two optimizers were tried and tested. First gradient descent with momentum. Momentum was set to 0.9. We noticed the loss values overshoot and take a very long time to minimizing the loss. Instead Adam optimizer with default beta1=0.9,beta2=0.999, and epsilon=1e-08 proved to be much more effective. Learning rate was manually decreased by a factor of 0.1 starting at 0.01. You can see figure 3 where sudden drops occur are when learning rate was changed and thus improved learning speed.

5 Experiments

Here we report on results for both baseline of CS231N and NetVLAD. We tested our work on greedy and beam search. We also show some generated captions to illustrate their quality. For training we used the train and dev set to tune hyperparameters. We tested a 1 and 2 Layer LSTM. 1-layer was unable to drive the loss down like the 2-layer to reach a good performing model. Batch size was chosen to optimize GPU performance, this was set to 25. Note that for each example video, there are 10 segments, so each batch in reality has 250 video segments. A larger batch size was unable to fit due to memory constraints. Smaller batch sizes were much more stochastic but not more effective at lowering the loss. Just as in CS231N baseline, early stopping was used to select the best checkpoint or model to generate captions with and evaluate the test. Seeing our reference plot in figure 3 that at 70 epochs, the best model was chosen. In addition, NetVLAD required the number of clusters to train on. We experimented with 2^n, for n= 1, ..., 7. These tests are expensive to run, so we only ran our best estimates all the way. All others were run sufficiently to see differences in ability to reduce the loss. We ultimately chose 32 clusters which gave us good results.

6 Results

Table of results is shown in table 1. We evaluate captions based on BLEU metric using unigrams, bi-grams, 3-grams, and 4-grams which we denote as B1, B2, B3, and B4. We believe NetVLAD was effective in outperforming baseline due to its inclusion of nonlinearities. Before the LSTM, the baseline simply computes attention over proposals. So it is very effective at drawing together relationships of the past present and future segments in generating its captions. However, NetVLAD only focuses on each segment without future or previous knowledge, instead only knowledge of the segment itself. The feature that represents the segment however is pooled in a nonlinear way and passes through context gating with help find interactions that give context to the LSTM while. Another drawback of the baseline is that it simply

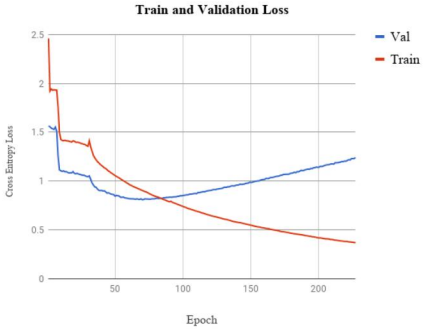


Figure 3: Loss over number of epochs in NetVLAD model

pools segments together. So although it has future and past information, that future and past information may be very ambiguous after the temporal max pooling is done. NetVLAD pools in such a way to reach its objective, to lower the loss. A more fair comparison would be to use entire training, dev, and test sets. We do expect however for all results to improve even more with the entire set.

Bleu Mean Values on Test Set				
Model	B1	B2	B3	B4
Baseline-Greedy	0.531	0.516	0.523	0.540
Baseline-Beam	0.542	0.533	0.554	0.585
NetVLAD-Greed	0.537	0.531	0.520	0.519
NetVLAD-Beam	0.607	0.589	0.592	0.598

Table 1: BLEU mean values for n-grams of 1, 2, 3, and 4, for test set. set

7 Discussion

We would like to point out some key differences in the baseline from CS231N and NetVLAD. Figure 4 shows some generated captions using greedy output (the captioning generator we implemented for this class). This first example shows a man in the yard doing some yard. The ground truth is that he is giving instructions on lawn and garden care. However this is not apparent until one watches the video and hears the audio. Baseline from CS231N predicts similar words, but does not make sense. NetVLAD with CG on the other hand predicts dissimilar words but the content makes more sense "a man is shown wandering through the yard." While this is not particularly correct, it is reasonable to expect. A weakness of NetVLAD however is not being able to tie together context from previous video segments since it does not have this information in the LSTM. This is evident in the third example where the word "also" ties together information from the past in the baseline. NetVLAD is not able to pick this type of information. A really interesting part to note is that although baseline and NetVLAD has same number of training examples, notable difference are that NetVLAD learns new sentences that make sense while baseline is trying too much to be correct, could be sort of overfitting due to having very similar dev and train set. See example two, where baseline repeats many words such as "violin", "ending", and, "the" and never outputs an $< end >$ token. NetVLAD on the other hand produces still a very different caption but is 100% accurate from an description standpoint.

8 Conclusion/Future Work

In conclusion, we have shown improvement via BLEU scores using NetVLAD over the baseline, however only on 10% of dataset. Training on entire set would be more conclusive to test the effectiveness of our model. Future work is to consider METEOR and CIDEr metrics in addition to BLEU. Also extending this work to combine strengths of NetVLAD in pooling C3D features and baseline attention to pool proposals may prove very effective.

Figure 4: Generated Captions

9 Code

See Course Project GitHub Repository. All coding was written by project members with three exceptions: the deep action proposals (demo purposes only) are from Github from [5], the C3D features and labels are from json files on the Dense-Captioning Events in Videos website, and the NetVLAD is from the LOUPE directory from [13]. The deep learning framework used throughout the entire project was Tensorflow [3].

10 Contributions

The group had contributions from everyone in the group. Some group members focused more on strengths and backgrounds. Parker Miller played a role in report writing and idea generation, Julio Martinez in Tensorflow coding, cloud and GPU setup, and designing the neural network architecture including the use of NetVLAD. Songze Li helped with CS231N aspects which included the baseline and Beam Search Caption Generator. However, all group members contributed to the project.

Contributions for class are as follows:

- CS231N/CS230: Report Writing by Parker Miller, Julio Martinez, and Songze Li
- CS231N/CS230: Idea Generation: Parker Miller, Julio Martinez
- CS231N/CS230: Preprocessing by Julio Martinez and Songze Li
- CS 231N: Baseline started from no source code all in tensorflow by Julio Martinez
- CS 231N: Beam Search Caption Generator by Songze Li
- CS230: NetVLAD and Context Gating adaption and caption module to work with NetVLAD by Julio Martinez
- CS230: Greedy Caption Generator by Julio Martinez
- CS231N/CS230: BLEU scoring function by Julio Martinez

References

[1] W. Qiu A. Friedrich M. Pinkal A. Rohrbach, M. Rohrbach and B. Schiele. Coherent multi-sentence video description with variable level of detail. In *German Conference on Pattern Recognition*, pages 184–195, 2014.

[2] M. Ranjbar A. Vahdat, B. Gao and G. Mori. A discriminative key pose sequence model for recognizing human interactions. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Converence*, pages 1729–1736, 2011.

[3] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[4] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5297–5307, 2016.

[5] Victor Escorcia, Fabian Caba Heilbron, Juan Carlos Niebles, and Bernard Ghanem. *DAPs: Deep Action Proposals for Action Understanding*, pages 768–784. Springer International Publishing, Cham, 2016.

[6] J.C. Niebles F. Caba Heilbron and B. Ghanem. Fast temporal activity proposals for efficient detection of human actions in untrimmed videos. In *Proceedings of the IEEE International Conference on Computer Vision and Patter Recognition*, pages 1914–1923, 2016.

[7] Z. Huang Y. Yang H. Yu, J. Wang and W. Xu. Video paragraph captioning using hierarchical recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4584–4593, 2016.

[8] J. Ohya J. Yamato and K. Ishii. Recognizing human action in time-sequential images using hidden markov model. In *Computer Vision and Patter Recognition, 1992. Proceedings CVPR'92., 1992 IEEE Computer Society Conference*, pages 379–385, 1992.

[9] C.W. Chen J.C. Niebles and L. Fei-Fei. Modeling temporal structure of decomposable motion segments for activity classification. In *European conference on computer vision*, pages 392–405, 2010.

[10] Justin Johnson, Andrej Karpathy, and Fei-Fei Li. Densecap: Fully convolutional localization networks for dense captioning. *CoRR*, abs/1511.07571, 2015.

[11] Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. Dense-captioning events in videos. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 1, page 6, 2017.

[12] Antoine Miech, Ivan Laptev, and Josef Sivic. Learnable pooling with context gating for video classification. *arXiv preprint arXiv:1706.06905*, 2017.

[13] Antoine Miech, Ivan Laptev, and Josef Sivic. Learnable pooling with context gating for video classification. *arXiv:1706.06905*, 2017.

[14] J. Sivic F. Bach O. Duchenne, I. Laptev and J. Ponce. Automatic annotation of human actions in video. In *Computer Vision, 2009 IEEE 12th International Conference*, pages 1491–1498, 2009.

[15] H. Pirsiavash and D. Ramanan. Parsing videos of actions with segmental grammars. In *Proceedings of the IEEE International Conference on Computer Vision and Patter Recognition*, pages 612–619, 2014.

[16] L. Seidenari S. Karaman and A. Del Bimbo. Fast saliency based pooling of fisher encoded dense trajectories. In *ECCV THUMOS Workshop*, volume 1, 2014.

[17] L. Burget J. Cernocky T. Mikolov, M Karafiat and S. Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, page 3, 2010.

[18] J.C. Niebles V. Escorcia, F.C. Heilbron and B. Ghanem. Daps: Deep action proposals for action understanding. In *European Conference on Computer Vision*, pages 768–784, 2016.

CS231N Dense-Captioning Videos with Learnable Pooling

Songze Li
Stanford University, ICME
songze@stanford.edu

Julio A. Martinez
Stanford University, ICME
juliomz@stanford.edu

Parker Miller
Stanford University, Law School
pmiller0@stanford.edu

Abstract

Video content metadata have a variety of use cases such as video query search, ad content matching, and video multimedia editing. In this work the content metadata of interest are event descriptions. The objective is to take a video and produce captions that describe the events that occur given sequences of C3D feature representations of video frames. Our baseline implementation is from the Activity Net benchmark for dense captioning events in videos. We show that replacing the attention module in the baseline with NetVLAD can produce similar, and potentially better results. We evaluate predicted captions of our models using the Bleu metric with n-grams of $n = 1, 2, 3,$ *and* 4 *for direct, sampling and beam search predictions. A method to visualize the attention relationship between word and proposals is also proposed.*

1. Introduction

We investigate dense-captioning, or the description of events, in videos. The problem we face is how to better describe video events with natural language using past, present, and future events within a given video.

Several applications of this work are interesting, for instance, being able to search a video for a specific segment wherein a particular action occurs, or match the content of the video with ads that are relevant. In addition, the project is motivating as we plan to submit our work to the ActivityNet competition.

While in computer vision, training neural networks to recognize images has been done quite effectively, training them to understand sequences of images and the context of nearby events is a newer challenge. A neural network capable of this task could prove useful in data analytics, data automation, security, and other fields.

In an application setting, the inputs of our work are video frames, and the outputs are captions, or natural language that describe the events taking place in segments of the video where events take place. Specific to our model, the inputs are C3D feature representations of video segments extracted by a pre-trained C3D model and timestamps that correspond to those features.

Our objective is to produce captions that describe action events in videos by using learnable pooling methods that pay attention to past present and future actions.

2. Related Work

Dense captioning of video events brings together work done in temporal action proposals and video captioning. Activity recognition has historically involved using hidden Markov models and discriminative SVM models to describe events with a fixed label set [9, 10, 17, 2]. Some have used deep learning to further study video event actions [19]. In our approach we use similar methods, but our goal is to describe video events with natural language using a larger vocabulary.

Traditionally, temporal action proposal methods used a sliding window approach [16], but later evolved to build proposal models that used dictionary learning [4] or RNN architecture [22] to locate potential action segments. These methods required processing for each video frame in the sliding window. Rather than use this sliding window approach, we output proposals for every time-step in a single pass of the video. We believe this model is more efficient.

Beginning first in video retrieval with natural language [14, 18] and moving to video summarization [21, 5, 13, 3], the work done bridging video and language has historically used low-level features like color, motion, or model objects [7] to identify key segments. But these methods have often been limited by small vocabularies and have not effectively described video events.

There have been various approaches to video captioning. Notably, a paper recommended describing videos with paragraphs, where an RNN generates hidden vectors that then initialize lower-level RNNs that generate individual sentences [20, 6]. Our work is similar to this approach, but we attempt to localize events in time and we use a significantly larger dataset [1], ActivityNet Captions. Additionally, we use dense-image captioning to generate localized descriptions of an image [8] and attempt to train our model to recognize context, or the interdependency among events,

in a video.

3. Dataset and Features

We use a dataset provided by ActivityNet Captions. The dataset contains 20k videos amounting to 849 video hours with 100k total descriptions. Since we are limited in computing resources, we do not train on the entire dataset. Instead, we randomly select 2000 videos from train, and carefully find 250 videos for validation, and 250 for test set that are represented in the training set. We are able to select 250 validation and 250 test videos by setting a threshold on how frequent each word in the validation and test captions appears in the training set. These sets are similar assuming pixel C3D features also carry some similarity, however we did not measure the similarity across features. This can be done however using a cosine or 2-norm similarity measure.

3.1. Pre-Processing

The amount of work required to pre-process the data is significant.

First, the input to our dataset are C3D features. We extract C3D features and use PCA for dimensionality reduction to produce 500-dimensional vectors, each of which represent a specific frame in the video.

Second, we group C3D frames by video and for each video also by event. Thus for each video there is a sequence of events, each event contains a sequence of 500-dimensional C3D-PCA features that represent a certain time segment that start and end based on the labeled timestamps t_init and t_end. Since the duration from t_init to t_end varies, so does the number of 500-dimensional C3D-PCA features that represent an event. Number of events per video is limited to 10, and number of words in a caption is limited to 30. For videos not meeting these two requirements we truncate or pad at the end accordingly (for sequences of length less than requirement we add padding).

In addition, words found in the label captions corresponding to these event segments are tokenized, meaning each word is mapped to a unique integer as done in [11]. The set of words plus three additional tokens, one to indicate <start> and one to indicate <end> of a caption, and padding <pad>, make up the vocabulary which has cardinality 10,999.

4. Methods

We implement two networks. One is the baseline method found in [12] and second is a variation of this with NetVLAD. The baseline was implemented from scratch without any source code.

The general pipeline is as follows, Input: C3D Features → Attention Module → Captioning Module → with Caption Generation done independently to produce captions.

The network pipeline for the baseline network is shown in 1.

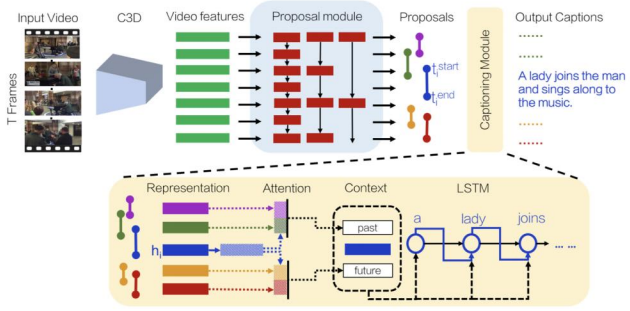


Figure 1: Baseline network structure

4.1. Baseline Attention Module

The first step in the model is the Attention step. We denote H as the event representation matrix composed of pooled C3D-PCA features. We let the ith column of H be h_i. Then each h_i is simply the max pooling across all C3D-PCA feature vectors for the ith proposal. The max pooling is not mentioned in [12] however after meeting with Ranjay Krishna it was clear this is the approach taken in that work. Thus, after the pooling, the ith column is a new single feature vector which represents the proposal.

This matrix H has dimensions 500xK for each video, where K is the number of proposals (in this case K=10). In the paper [12] the attention module is computed as follows:

$$h_i^{past} = \frac{1}{Z^{past}} \sum_{j \neq i} \mathbb{1}\{f_j^{end} < f_i^{end}\} a_{ij} h_j \quad (1)$$

$$h_i^{future} = \frac{1}{Z^{future}} \sum_{j \neq i} \mathbb{1}\{f_j^{end} \geq f_i^{end}\} a_{ij} h_j \quad (2)$$

Where we have

$$Z_i^{future} = \sum_{j \neq i} \mathbb{1}\{f_j^{end} \geq f_i^{end}\} \quad (3)$$

$$w_i = w_a h_i + b_a \quad (4)$$

$$a_i j = w_i h_j \quad (5)$$

and the Z_i^{past} is analogous to Z_i^{future} for past data.

In order to compute this efficiently for one example, we vectorized this in the following way:

$$W = W_a H + b_a \quad (6)$$

$$A = W^T H \quad (7)$$

$$H^{past} = H(Ind^{past} * A)^T / Z^{past} \quad (8)$$

$$Z^{past} = \sum_{rows} (Ind^{past}) \quad (9)$$

Where * represents element-wise multiplication and Ind^{past} and Ind^{future} are matrices of 1's and 0's. $Ind^{past}_{ik} = 1$ indicates that $f^{end}_k < f^{end}_i$. This can now be run more efficiently in vectorized format in Tensorflow.

The attention module output $Hout$ is simply the concatenation of the pooled representations of the C3D features H^{past}, H, and H^{future}, each of size 500, resulting in a vector of size 1500 ($Hout$). This way, each column of $Hout$ includes information for the future, present, and past proposals as (h^{future}_i, h_i, h^{past}_i).

4.2. NetVLAD Attention Module

NetVLAD (neural network of vector of locally added descriptors) is a form of attention and learnable pooling that has been used successfully in video-level tagging [15]. NetVLAD computes clusters in the input features and computes residuals from each temporal input feature to the cluster centers c, and multiplies this residual to a softmax. These values are then aggregated temporally, resulting in a vector that draws a nonlinear relationship between the features it aggregates.

$$VLAD(j,k) = \sum_{i}^{N} softmax(h_i)(h_{ij} - c_{kj}). \quad (10)$$

Context Gating, is an activation function, in fact a combination of sigmoid element-wise product and the input features as shown below. This introduces yet another quadratic relationship between the features.

$$CG(X) = \sigma(WX + b) * X \quad (11)$$

As shown below, in this case, instead of max pooling the C3D-PCA feature vectors for a proposal, we use NetVLAD and context gating to aggregate them in a nonlinear way. Thus the result is a pooled representation that is learnable of dimension 500, as opposed to 1500 with baseline which concatenates future past and present feature,s each of dimension 500. Figure 2 shows the attention module using NetVLAD and context gating.

4.3. Captioning Module

In the captioning module we initialized word embeddings uniformly at random to map each word from its integer id to a vector using a fully connected word embedding layer. We choose 512 dimensions consistent with work in [12].

After the word embedding, where each word is mapped from a one-hot vector to a 512-dimensional representation which we denote $Zembed$.

For both baseline and NetVLAD attention schemes we concatenate the proposal feature vectors with the word embeddings. In baseline, each proposal feature vector is 1500,

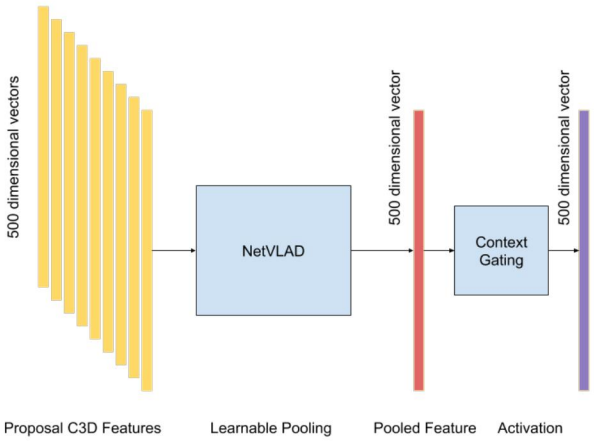


Figure 2: NetVLAD with Context Gating Pipeline

thus concatenating it to each 512-dimension word embedding in the caption leads to a 2012 dimensional vector. In NetVLAD + Context Gating, the proposal feature vector is 500-dimensional and hence the concatenation leads to 1012, in each case since the proposal feature is a single vector and the caption is a sequence of word embedding features, we tile the proposal feature vector to be same length as caption and then concatenate. This make the concatenation to each word embedding in the caption.

The next step in the caption module is the 2-layer LSTM. Each hidden state is set to be 512-dimensional as in [12]. Hidden states use zero initialization. A each time step of the LSTM the concatenation of the word embedding and the feature vector are fed in as input.

After a softmax is computed over each step of LSTM with 10,999 units (the number of classes) is computed to give scores for each class (word).

4.4. Caption Generation

Caption Generation is used during the test stage, so that we can generate work by word based on the past corpus. There are several ways to generate the caption in our model. A simple approximation is to use a greedy search that selects the most likely word at each step in the output sequence. This approach has the benefit that it is very fast, but the quality of the final output sequences may be far from optimal, because once if you get some wrong word it is harder to get back to the real label again.The second approach to do sampling, which mean we sample the next word by multinomial distribution sampling.

Another popular heuristic is the beam search that expands upon the greedy search and returns a list of most likely output sequences. Instead of greedily choosing the most likely next step as the sequence is constructed, the beam search expands all possible next steps and keeps the k

most likely, where k is a user-specified parameter and controls the number of beams or parallel searches through the sequence of probabilities. Common beam width values are 1 for a greedy search and values of 5 or 10 for common benchmark problems in machine translation. Larger beam widths result in better performance of a model as the multiple candidate sequences increase the likelihood of better matching a target sequence. This increased performance results in a decrease in decoding speed.

To make the generation process faster, we take the beam size 3.

4.5. Optimization and Loss Function

For the cost function the total cross entropy loss divided by the number of proposals and number of mini batches was used.

We tested two optimization algorithms. The first was gradient descent with momentum. This lead to slow progress. With such a large dataset there was not enough time to compute this all the way. We noticed the loss would start high and then progress slowly at first. We then tried the Adam Optimizer. This resulted in much quicker progress. However in both cases, train these large sets takes alot of computation and memory. The learning rate was initialized at 0.01 and reduced in a step-wise fashion by a decay rate of 0.95 every few epochs. This decay was done manually which corresponds to the kinks in the plot shown in figure 3.

4.6. Evaluation

BLEU uses a modified form of precision to compare a candidate or hypothesis caption against a reference caption. In this case the hypothesis is the generated caption and the reference is the ground truth caption. We use BLEU with uni-grams, bi-grams, 3-grams, and 4-grams which we denote B1, B2, B3, and B4 respectively. The BLEU score is only computed on test data generated captions.

4.7. Code

See Course Project GitHub Repository.

5. Experiments, Results & Discussion

5.1. Experiment Details

In this section we show our experiments and results for the two main networks tested, the baseline and the NetVLAD adaptation of the baseline, which we have discussed thoroughly above. In addition to these networks, caption generators are also tested, greedy caption generator and beam search. Here we share a few examples to give a qualitative analysis of the results, and give a discussion about our choices of various hyperparameters.

To ensure each model is capable of learning, a first approach was to run the models on a small subset of the data. We chose 100 videos from the training set. We ran 1000 iterations (this is computationally possible with 100 samples). This allowed the models to reduce loss to close to zero.

In addition, hyperparameters tuned in these experiments were the number of LSTM layers, dropout, batch size, and learning rate. Since we were limited in computing memory, a batch size of 32 was large enough. Note that for each video there are several video segments, in this case 10, this batch size of 32 results in a 32*10 = 320 number of video segments, which could be thought of as the true batch size. We tried batch size of 1 (i.e. 1*10 = 100 video segments), although much more stochastic, which is the size used in [12], this value was too small to be computed in a practical amount of time for this course, and does not take advantage of GPU computing power since it relies more on sequential learning in the for loop. 1-layer LSTM was tried before 2 layers, however, this network wasn't capable of learning well as the loss had a difficult time becoming as small as the 1-layer network. In addition, a dropout wrapper applied over the LSTM parameters was experimented with to better generalize. However, we didn't see much upside to using dropout with keep probability of 0.5. You can see the source code to verify dropout is implemented, we simply did not ended up using it, i.e dropout keep probability was set to 1.0. Learning rate was adjusted manually by keeping a close eye on the loss as it progressed, starting at 0.01, and slowly decreasing it as needed as loss approached 0.0.

In order to not overfit, we trained our models until the respective validation loss started to go up. This is referred to as early stopping. As shown in figure 3, the optimal model for NetVLAD model, for example, was at 70 epochs. Using checkpoints, we were easily able to recover the best model to test with.

5.2. Quantitative Results

We have found that NetVLAD is able to do slightly better at captioning video segments than the baseline model. This may be due to both the fact that NetVLAD learns how to best pool features together, and also since Context Gating add an extra activation. The baseline may lose information in pooling a segment of features together, which may result in a which is difficult to understand. The reason max pooling may work in Convolutional Neural Networks is due to still retaining understandable features. However if all features in an image were pooled into 1 feature, this may not result useful, the same may apply to segment max pooling.

NetVLAD performed slightly better overall than the best performing captions, with respect to BLEU scores, when using beam search. However both both baseline and NetVLAD yielded better BLEU scores for captions gener-

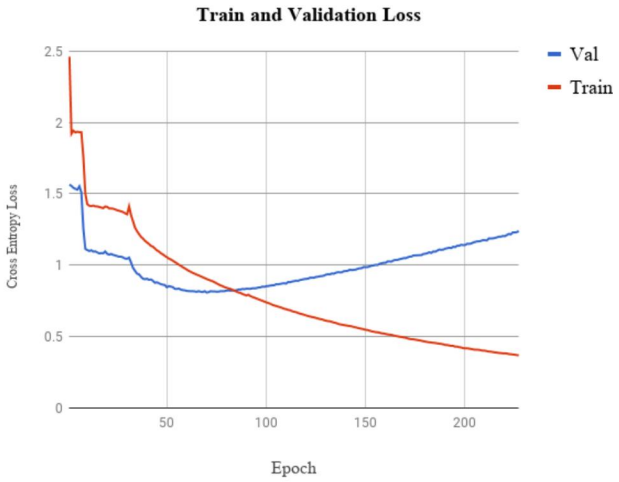


Figure 3: Loss over number of epochs in NetVLAD model.

ated with beam search vs greedy search.

Bleu Mean Values on Test Set				
Model	B1	B2	B3	B4
Baseline-Greedy	0.531	0.516	0.523	0.540
Baseline-Beam	0.542	0.533	0.554	0.585
NetVLAD-Greed	0.537	0.531	0.520	0.519
NetVLAD-Beam	0.607	0.589	0.592	0.598

Table 1: Bleu mean values for n-grams of 1, 2, 3, and 4, for test set. set

Because we are only able to train on a portion of the dataset, we expect our baseline results to be compromised in comparison to those in [12]. However, because we intend to only use this network as a baseline, we feel it could still act as an effective measurement against our adjusted model. Our hope is that the adjusted model's performance will improve upon the performance of the baseline model.

5.3. Qualitative Results

Below are shown a few selected example of NetVLAD and Baseline captions predicted using beam search. As can be seen, it seems baseline has a more difficult time generalizing the context and instead focuses more on correct word prediction, while NetVLAD approach may result in different words but in more meaningful expression such as man "wandering through the yard" in the first example, or "smiling once down into the camera". While not entirely the ground truth, it is reasonable. Both cases suffer from giving as good results at discussed in [12]. This is expected as the number of training examples is only 10% of the entire dataset used in that work.

Our main motivation for using NetVLAD was that it received 1st place in the Youtube-8M challenge in 2017 for video-level tagging. NetVLAD performed so well mainly because of the nonlinear pooling and context gating which adds yet another quadratic relationship between the features. This nonlinear pooling and activation allows the network to gather in the context of the entire proposal more effectively than a simple max pooling and linear attention as done by the baseline. This is our best judgement for why NetVLAD is able to come up with more variation in sentences as compared to ground truth and baseline, but still make sense. In the first example, "a man is shown wandering," brings up many questions, one of which is that the ground truth obviously includes information that is only noticeable from the audio. Thus by looking at the video, it is difficult to determine whether or not this is an instructional video or just some man "wandering" around in his yard trimming the hedges, hence the NetVLAD output.

5.4. Attention visualization and Analysis

To capture the context from all other neighboring events in our proposals, there are attention mechanisms in our network, which allowed the present proposals to "look into" past proposals and future proposals to "find" the connection and thus generate the captions. In our work the concatenation of $[h^{past}, h, h^{future}]$ and word embedding is used as input to cells of LSTM. In this way, every word is a representation of the whole proposals, which is encoded in the output of LSTM.However, There is few literature about visualization of the video understanding and about which part of the proposals is the attention paid by the word generated by each cell of LSTM. An idea of visualization these attention mechanism is first shown here and the results are discussed.

To better understand how the caption is generated, it is helpful to find how the output from LSTM cells H_{out} attention to the hidden state representation of the image $[h^{past}, h, h^{future}]$. Borrowed from the idea of attention mechanism widely used in NLP. We took the dot product between H_{out} and h^{past}, h, h^{future} individually and get the attention score of each word to the past, current and future proposal,$Att_{Past}, Att_{Current}, Att_{Future}$. Since it is hard to figure out what is the meaning of the vectors in hidden presentation of C3D features , we add the attention scores for each hidden presentation and normalize to make it a value between 0 and 1, and we call it "Attention Level". "Attention Level" is used to see how each word pay attention to past or future proposal relatively and thus to help us understanding how the caption is generated.

Two typical examples are selected to show the attention mechanism. The caption for the first proposal in Figure 5 is "the man grabs a third empty glass and moves it". If we look into the curve for attention level, we can see the at-

Figure 4: Generated Captions for Baseline and NetVLAD

tention for past and current is moving up for word "third", while the attention for future is going down. It is very interesting because it seems like the hidden state was counting the number of glasses the man had picked during the past and current proposals and neglecting the future event in the future proposals, because the future proposals has no relationship of the number of glasses in current proposal. The similar behavior could also been seen in 6, we can see from proposal one for the word "holding", the attention for the future and current is moving up compared to the past. Similar behavior happens in proposal two, the attention to current for word "seen" and "on" is high compared to past and future, which means these words is "seeing" the current proposals more, which could also be intuitively understood.

6. Conclusion and Future Work

In conclusion, we have effectively shown that NetVLAD can slightly improve BLEU scores for video events given the event segment frame features. However, this was trained on only 10% of the dataset and validated and tested in 250 videos. Although it performed good enough to see a few good results, the performance in our view is not significant enough to warrant one vs the other nor is the test a conclusive enough test to understand the actual performance. In addition the metrics used may not be sufficient enough to validate the performance for all use cases. It remains to be seen whether or not NetVLAD is as effective over baseline at the full 20k videos. In future work we could also consider other metrics which could expose further weaknesses or strengths of NetVLAD and baseline methods. In addition, the paper did not implement a proposal module. Extending this work to include proposals iteratively against a attention and captioning modules the way the baseline was able to do effectively would be a more fair comparison.

Some future work could also be to exploit both the strengths of the baseline and the NetVLAD model. We could use NetVLAD to pool features, and baseline Attention to pool proposals and keep the same captioning module.

In addition, because of the limited time of training for this project, we need more time to train the whole training set and test the whole validation sets to see see the effectiveness of our model. Secondly, several improvement could to be done, especially for the captioning model. For example, bi-direction LSTM, and how to incorporate the caption generation mechanism into our training process. All of these improvement are expected to increase the effectiveness of our model.

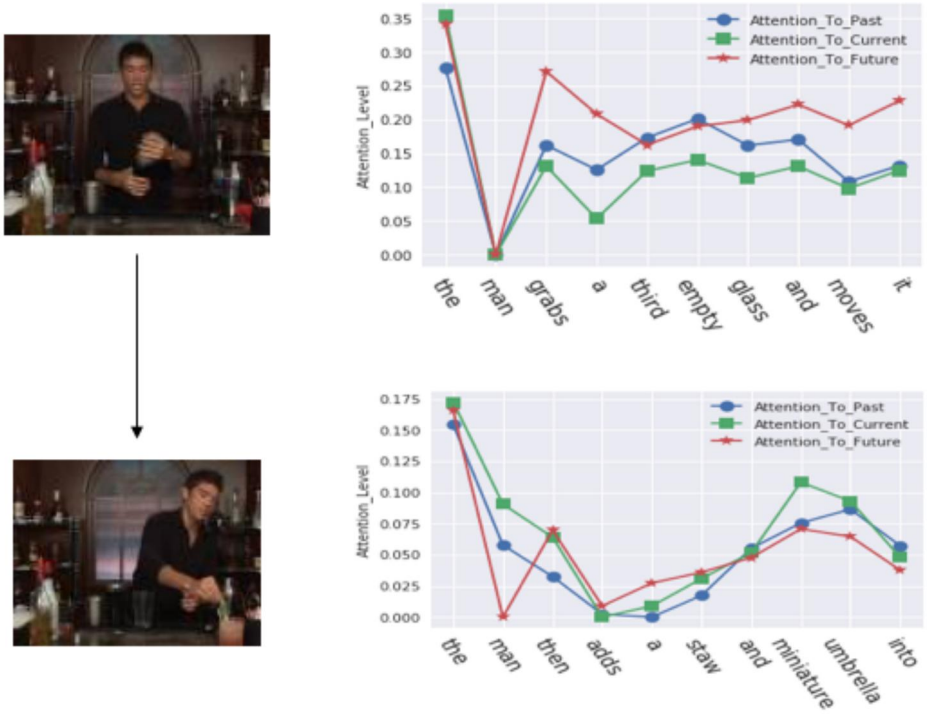


Figure 5: Attention analysis:A man making cocktail

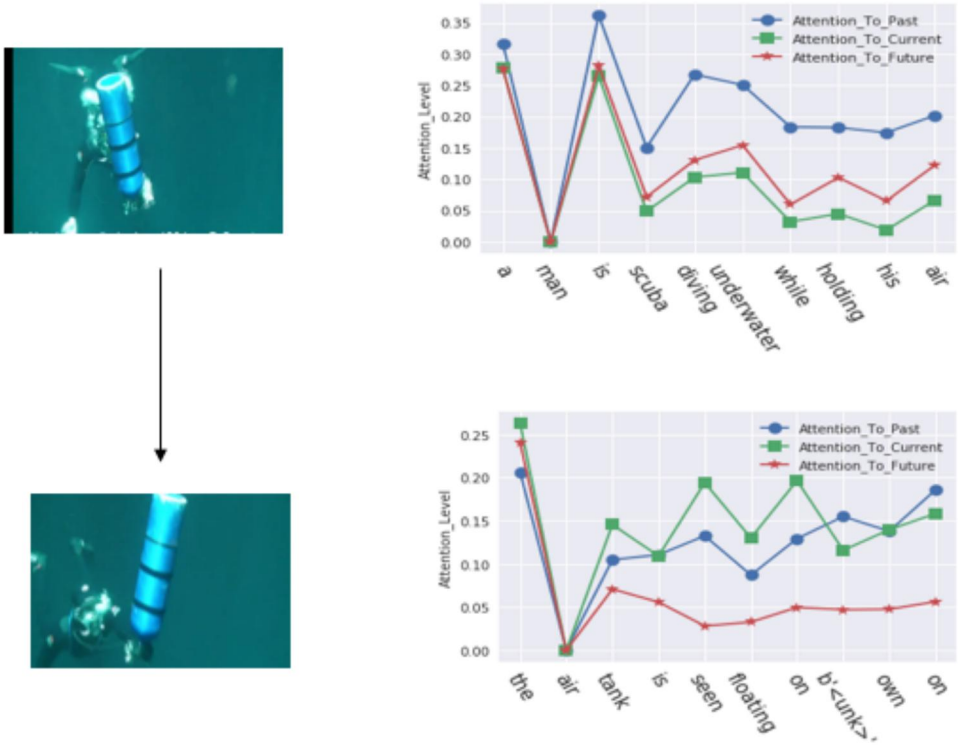


Figure 6: Attention analysis:A man scuba diving with air tank

7. Contributions

The group had equal contributions from everyone in the group. Some group members focused more on strengths and backgrounds. Parker Miller played a significant role in report writing and idea generation, Julio Martinez in tensorflow coding, cloud and GPU setup, and Songze Li on natural language architecture and caption generation. However, all group members contributed to the entirety of the project.

Contributions for class are as follows:

- CS231N/CS230: Preprocessing
- CS 231N: Baseline started from no source code all in tensorflow.
- CS 231N: Beam Search Caption Generator
- CS230: NetVLAD and Context Gating adaption and caption module to work with NetVLAD
- CS230: Greedy Caption Generator
- CS231N/CS230: Performance Analysis
- CS231N/CS230: BLEU scoring function
- CS230: Visualization Analysis

8. Code

All code in this work was written by the project members Julio Martinez, Songze Li, and Parker Miller with three exceptions, deep action proposals (daps, which is only used for demo purposes and not in this paper, i.e. get proposals from a video to feed into this network to predict captions) was available on GitHub, C3D features and corresponding labels are available in json files on website for DenseCaptioning Events in Videos, and also the NetVLAD class was available on GitHub. daps and the NetVLAD class are found in daps and the LOUPE directories respectively. Everything outside of these three including their mode of use was written by project members.

References

[1] W. Q. A. F. M. P. A. Rohrbach, M. Rohrbach and B. Schiele. Coherent multi-sentence video description with variable level of detail. In *German Conference on Pattern Recognition*, pages 184–195, 2014. 1

[2] M. R. A. Vahdat, B. Gao and G. Mori. A discriminative key pose sequence model for recognizing human interactions. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Converence*, pages 1729–1736, 2011. 1

[3] O. Boiman and M. Irani. Detecting irregularities in images and in video. In *International Journal of Computer Vision*, pages 17–31, 2007. 1

[4] J. N. F. Caba Heilbron and B. Ghanem. Fast temporal activity proposals for efficient detection of human actions in untrimmed videos. In *Proceedings of the IEEE International Conference on Computer Vision and Patter Recognition*, pages 1914–1923, 2016. 1

[5] S. L. D. W. M. G. H. Yang, B. Wang and B. Guo. Unsupervised extraction of video highlights via robust recurrent auto-encoders. In *Proceedings of the IEEE Conference on Computer Vision*, pages 4633–4641, 2015. 1

[6] Z. H. Y. Y. H. Yu, J. Wang and W. Xu. Video paragraph captioning using hierarchical recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4584–4593, 2016. 1

[7] D. Z. H.J. Zhang, J. Wu and S. Smoliar. An integrated system for content-based video retrieval and browsing. In *Pattern Recognition*, pages 643–658, 1997. 1

[8] A. K. J. Johnson and L. Fei-Fei. Densecap: Fully convolutional localization networks for dense captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4565–4574, 2016. 1

[9] J. O. J. Yamato and K. Ishii. Recognizing human action in time-sequential images using hidden markov model. In *Computer Vision and Patter Recognition, 1992. Proceedings CVPR'92., 1992 IEEE Computer Society Conference*, pages 379–385, 1992. 1

[10] C. C. J.C. Niebles and L. Fei-Fei. Modeling temporal structure of decomposable motion segments for activity classification. In *European conference on computer vision*, pages 392–405, 2010. 1

[11] J. Johnson, A. Karpathy, and F. Li. Densecap: Fully convolutional localization networks for dense captioning. *CoRR*, abs/1511.07571, 2015. 2

[12] R. Krishna, K. Hata, F. Ren, L. Fei-Fei, and J. C. Niebles. Dense-captioning events in videos. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 1, page 6, 2017. 2, 3, 4, 5

[13] H. G. M. Gygli and L. V. Gool. Video summarization by learning submodular mixtures of objectives. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3090–3098, 2015. 1

[14] E. R. J. H. M. Otani, Y. Nakashima and N. Yokoya. Learning joint representations of videos and sentences with web image search. In *European Conference on Computer Vision*, pages 651–667, 2016. 1

[15] A. Miech, I. Laptev, and J. Sivic. Learnable pooling with context gating for video classification. *arXiv preprint arXiv:1706.06905*, 2017. 3

[16] J. S. F. B. O. Duchenne, I. Laptev and J. Ponce. Automatic annotation of human actions in video. In *Computer Vision, 2009 IEEE 12th International Conference*, pages 1491–1498, 2009. 1

[17] H. Pirsiavash and D. Ramanan. Parsing videos of actions with segmental grammars. In *Proceedings of the IEEE International Conference on Computer Vision and Patter Recognition*, pages 612–619, 2014. 1

[18] W. C. R. Xu, C. Xiong and J. Corso. Jointly modeling deep video and compositional text to bridge vision and language in a unified framework. In *AAAI*, volume 5, page 6, 2015. 1

[19] L. S. S. Karaman and A. D. Bimbo. Fast saliency based pooling of fisher encoded dense trajectories. In *ECCV THUMOS Workshop*, volume 1, 2014. 1

[20] L. B. J. C. T. Mikolov, M Karafiat and S. Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, page 3, 2010. 1

[21] T. M. T. Yao and Y. Rui. Highlight detection with pairwise deep ranking for first-person video summarization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 982–990, 2016. 1

[22] J. N. V. Escorcia, F.C. Heilbron and B. Ghanem. Daps: Deep action proposals for action understanding. In *European Conference on Computer Vision*, pages 768–784, 2016. 1