A PROJECT REPORT

ON

"DEEP LEARNING FOR VIDEO SUMMARIZATION"

BY

KUSHAGRA AGRAWAL 2014AAPS0334H

UNDER THE GUIDANCE OF

Prof. Anand M. Narasimhamurthy

Submitted in partial fulfillment of the requirements of

CS F376 Design Oriented Project

Birla Institute of Technology and Science

Pilani, Hyderabad

Aug - Dec 2017

Acknowledgements

I am profoundly grateful to **Prof. Anand M. Narasimhamurthy** for his expert guidance and continuous encouragement throughout to see that this project rights its target since its commencement to its completion.

We would like to express deepest appreciation towards **Prof. Tathagata Ray**, Head of Department of Computer Science and Information Systems, BITS Pilani Hyderabad Campus whose invaluable guidance supported me in completing this project.

At last I must express my sincere heartfelt gratitude to all the staff members of the department who helped me directly or indirectly during this course of work.

KUSHAGRA AGRAWAL

Birla Institute of Technology and Science-Pilani,
Hyderabad Campus

CERTIFICATE

This is certify that the project report entitled

Deep Learning for Video Summarization

submitted by

KUSHAGRA AGRAWAL   2014AAPS0334H

in partial fulfillment ofthe requirements of the course CS F376, Design Oriented Project Course, embodies the work done by him under my supervision and guidance.

Date: 27/11/2017

(Prof. Anand Narasimhamurthy)
Project Guide

ABSTRACT

With an explosion in multimedia content production due to ever increasing reach of internet, content in the form of videos is becoming increasingly commonplace, becoming the preferred method for the purpose of delivering content. Advent of social media and growth of video sharing websites, in particular Youtube, has only contributed to the increasing importance of videographic content. More content is uploaded to Youtube a day, than a person is capable of watching in his/her whole lifetime. With the emergence of video content as an effective mode of information propagation, automating the process of summarization a video has become paramount. Video Summarization, in recent times, has emerged as a challenging problem in the field of machine learning, which aims at automatically evaluating the content of a video, and generating a summary with the most relevant content of the video. Video summarization finds applications in generating highlights for sports events, trailers for movies and in general shortening video to the most relevant subsequences, allowing humans to browse large repository of videos efficiently. Video summarization is a challenging problem in multiple facets. There is no natural ordering of video summaries. Among a given set of video summaries, the best representation of the original video is highly subjective. The general objective in modern literature for video summarization aims at producing a summary, typically 5%-15% of the whole video, consisting of the most informative content from the original video. The content is usually represented in the form of key frames, or more appropriately as video skims. A good video summary depicts the synopsis of the original video, in a compact way depicting all important and relevant scenes/shots. Through this project, we review the major techniques in video summarization, and look at their performance on some recent datasets.

Contents

Chapter 1

Installation and Setup

1.1 Conda and pip

Anaconda is an easy-to-install free package manager, environment manager, Python distribution, and collection of over 720 open source packages offering free community support. The project code is in python and uses a number of libraries like $openCV$, $TensorFlow$, $Keras$, $SciPy$, $NumPy$ and $Scikit - learn$. To allow robust and compatible installation of these libraries and integrating them with the code, I installed anaconda.

It contains most of the machine learning (ML) and media processing libraries that may be required. Anaconda can be installed from a windows installer as well as by command-line. After that installing a library can be done by the command.

```
conda install
```

It also covers any dependencies needed for the installation of other packages. Any package not found under the umbrella of conda has conventionally been found in the metadata lists of pip which is a package management system used to install and manage software packages written in Python.

1.2 TensorFlow, Keras

Of late a number of Deep Learning libraries and packages have come up like Caffe, PyTorch, TensorFlow etc. All these are specially engineered for Deep Learning computations. It was found that Keras with a TensorFlow backed was the best choice for the project.

TensorFlow is an open source software library for ML and meets any needs for systems capable of building and training neural networks to detect and decipher patterns and correlations, analogous to the learning and reasoning which humans use.

It is built for Python and has a large support community. It is also a reputed package as a number of top companies constantly work on optimizing it better (Google Engineers Group among them).

Keras is built on top of Theano and TensorFlow and provides an easy interface for quick high level prototyping. Keras uses numpy and scipy, pyyaml, HDF5 and h5py, cuDNN (recommended if you use CNN) as dependencies.

Video processing is a space-expensive activity and so would require the GPU RAM to be fully utilized. As the machine used for the project has a Nvidia graphic card, the CUDA library drivers were best suited as they abstract away the GPU dependence to give smooth computation. The cuDNN library provides fast and efficient functions for Deep Learning Algorithms .

For the installation of the packages, conda command is fully equipped. It takes cares and installs all dependencies for TensorFlow and Keras.

Chapter 2

Relevant Work

Video Summarization has gained a lot of attention in recent times. The task of video summarization can be accomplished in primarily two ways: *Key Frame Extraction* and *Video Skimming*. A lot of the initial work in video summarization has been done in the domain of unsupervised learning, but the recent trend has shifted towards learning from how humans generate video summaries, leading it into the supervised domains. A brief review of the work done in the video summarization is done according to the broad categorization above.

2.1 Key Frame Extraction

Key frame extraction, as the name suggests, is to choose the most informative frames from the video. These indexed frames are supposed to be the best ones that summarize the video. The key frame extraction is primarily used to obtain static summaries.

2.2 Video Skims

Using key frames to summarize a video might be useful for automatically analyzing the content of the video, but it produces a discontinuous and rather unpleasant summary for human viewing. This calls for summarizing a video in the form of skims of the video. This however is a complex task and often is more difficult to achieve for user videos which lack structure. The semantic meaning is frequently required in such cases. Some work in this region, like [15], tries to learn the meaning of the video by detecting motion of the camera and dividing the [16] tries to do so by detecting and clustering low-level features, and detecting differences between frames. Often the original algorithms for key frame extraction can be be extended to video skimming by choosing a continuous set of frames around key frames to produce a video skim.

Chapter 3

Datasets

3.1 SumMe

[1] created this benchmark with an intention to automate summary evaluation for present and future video summarization techniques, and have a common benchmark to compare it on. This dataset consists of *25 videos* which are single-shot and range in the length from *1-6 minutes*. The dataset contains summaries created by *15 to 18 users* with the constraint in length being that the summaries should be *5% to 15%* of the original video. Tge dataset also has the average scores of a frame being included in the summary from this data. The benchmark comes along with an automatic video summary evaluation scripts, which computes the f-score for given set of frames chosen by the algorithm. The f-score is measured against the human generated summaries. The dataset was created using crowd sourced annotation.

3.2 TVSum50

The videos in SumMe dataset contain only one shot, as they are home videos recorded using a single camera, which is a major limitation of the dataset. [2] created this TVSum50, in order to test algorithms based on shot detection. This dataset contains 50 videos from 10 wide range of categories. The dataset also focusses on title based summarization, and supplies a representative image for each video, which can be used for summarization.

Chapter 4

Methods Used

4.1 Uniform Sampling

Uniform Sampling is one of the most primitive algorithms for video summarization, which basically selects every k^{th} frame in the summary, where k is defined by the summary length requirement. Taking the typical summary size as 15% of the original video, this that every 6^{th} or 7^{th} is required to be chosen in the final summary. Video summarization, being the complex task it is, even uniform sampling remains important in context of video summarization. The approach based on superframes as specified in [1] is discarded for the simpler approach mentioned here.

4.2 VSUMM

First proposed in [3], the technique has been one of the fundamental techniques in video summarization in the unsupervised setup. The algorithm uses the standard K-means algorithm to cluster features extracted from each frame. *Colour histograms* are proposed to be used in [3]. Colour histograms are 3-D tensors, where each pixel's values in the RGB channels determines the bin it goes into. Since each channel value ranges in $0-255$, usually, 16 bins are taken for each channel resulting in a $16x16x16$ tensor. Due to computational reasons, a simplified version of this histogram was computed, where each channel was treated separately, resulting in feature vectors for each frame belonging to R^{48}. The approach in [1] for clustering is slightly different. But, the simplified colour histograms give comparable performance to the true colour histograms. The *features extracted from VGG16 at the* 2^{nd} *fully connected layer* were tried as features as well. Since the original features belong to R^{4096}, the features were reduced in dimensionality using PCA to R^{500}.

One of the initial things studied VSUMM was the effect of pre-sampling, that is only a fraction of the frames were considered for summary. This was computationally necessary as taking all the frames increased compute time substantially. As the sequence of the frames are strongly correlated, *sampling at low rates was shown to have no effect on the final results.* For the rest of the algorithm, only every 5^{th} fram was taken for consideration. (Taking a high sampling rate such as every 30^{th} frame often meant that a large enough summary cannot be obtained for evaluation. Also summary quality started to reduce at such high rates. So, 5 was chosen.)

Initial approach to the problem was to generate a key frame of 15% length, was to set $K = 15 * frames/100$ in K-means algorithm. The results for this approach are listed in VSUMM column using color histograms, using VGG16 features at fc7 in VSUMM+VGG16 column. The VGG16 approach was discarded to color histograms, because there was no substantial gain, and VGG16 features were computationally expensive to compute.

There are two problems with the above approach. After pre-sampling at every 5^{th} frame, the number of choices to be included in the 15% summary is only the 20% of original video. The second problem is that it produces discontinuous set of frames, which does not correlate well with how humans would generate video summary, giving relatively poorer scores. As suggested in [1], video skims of approximately 1.8 seconds centred around some key frames was used to generate summary. This slight modification reduced the number of key frames to be chosen using K-means clustering, and at the same time allowed for more continuous video summaries. The results are shown to improve using this technique in the VSUMM skims column.

Chapter 5

Results and Proposed Future Work

5.1 Results

The results shown are the F-scores of various models and humans on the SumMe dataset [1]. From the standard benchmarks shown below, we see that the algorithms are nowhere close to the upper bound, or even the best human. No algorithm dominates the results, i.e. outperform the other algorithms on most videos. This is because even in this small set of data, we have a wide variety of videos. Some algorithms suit a specific type of videos while fail miserably at others. In the future, a mixture of experts model could be trained to widen the range of videos that the model can deal with.

5.1.1 Standard Benchmarks

Video Name	Random	Upper Bound	Worst Human	Mean Human	Best Human	Uniform	Cluster	Attr.	Gygli
Base jumping	0.144	0.398	0.113	0.257	0.396	0.168	0.109	0.194	0.121
Bike Polo	0.134	0.503	0.19	0.322	0.436	0.058	0.13	0.076	0.356
Scuba	0.138	0.387	0.109	0.217	0.302	0.162	0.135	0.2	0.184
Valparaiso Downhill	0.142	0.427	0.148	0.217	0.4	0.154	0.154	0.231	0.242
Bearpark climbing	0.147	0.33	0.129	0.217	0.267	0.152	0.158	0.227	0.118
Bus in Rock Tunnel	0.135	0.359	0.126	0.217	0.27	0.124	0.102	0.112	0.135
Car railcrossing	0.14	0.515	0.245	0.217	0.454	0.146	0.146	0.064	0.362

Cockpit Landing	0.136	0.443	0.11	0.217	0.366	0.129	0.156	0.116	0.172
Cooking	0.145	0.528	0.273	0.217	0.496	0.171	0.139	0.118	0.321
Eiffel Tower	0.13	0.467	0.233	0.312	0.426	0.166	0.179	0.136	0.295
Excavators river crossing	0.144	0.411	0.108	0.303	0.397	0.131	0.163	0.041	0.189
Jumps	0.149	0.611	0.214	0.483	0.569	0.052	0.298	0.243	0.427
Kids playing in leaves	0.139	0.394	0.141	0.289	0.416	0.209	0.165	0.084	0.089
Playing on water slide	0.134	0.34	0.139	0.195	0.284	0.186	0.141	0.124	0.2
Saving dolphines	0.144	0.313	0.095	0.188	0.242	0.165	0.214	0.154	0.145
St Maarten Landing	0.143	0.624	0.365	0.496	0.606	0.092	0.096	0.419	0.313
Statue of Liberty	0.122	0.332	0.096	0.184	0.28	0.143	0.125	0.083	0.192
Uncut Evening Flight	0.131	0.506	0.206	0.35	0.421	0.122	0.098	0.299	0.271
paluma jump	0.139	0.662	0.346	0.509	0.642	0.132	0.072	0.028	0.181
playing ball	0.145	0.403	0.19	0.271	0.364	0.179	0.176	0.14	0.174
Notre Dame	0.137	0.36	0.179	0.231	0.287	0.124	0.141	0.138	0.235
Air Force One	0.144	0.49	0.185	0.332	0.457	0.161	0.143	0.215	0.318
Fire Domino	0.145	0.514	0.17	0.394	0.517	0.233	0.349	0.252	0.13
car over camera	0.134	0.49	0.214	0.346	0.418	0.099	0.296	0.201	0.372
Paintball	0.127	0.55	0.145	0.399	0.503	0.109	0.198	0.281	0.32
mean relative to upper bound	0.139 31%	0.454 100%	0.179 39%	0.311 68%	0.409 90%	0.143 31 %	0.163 36 %	0.167 37 %	0.234 52 %
relative to average human	45%	146%	58%	100%	131%	46 %	53 %	54 %	75 %

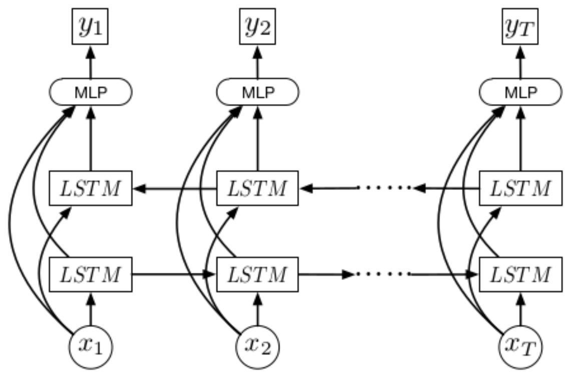
5.1.2 Results

Video Name	Gygli	VSUMM + VGG16	VSUMM Skims	Uniform
Base Jumping	0.121	0.083297	0.081826	0.085364
Bike Polo	0.356	0.078986	0.084036	0.074112
Scuba	0.184	0.145599	0.145578	0.145059
Valparaiso Downhill	0.242	0.201909	0.203090	0.19899
Bearpark Climbing	0.118	0.156292	0.148718	0.160377
Bus in Rock Tunnel	0.135	0.029249	0.031561	0.030199
Car railcrossing	0.362	0.387079	0.392653	0.363804
Cockpit Landing	0.172	0.096354	0.101712	0.089413
Cooking	0.321	0.023047	0.021330	0.023748
Eiffel Tower	0.295	0.123511	0.122409	0.119034
Excavators river crossing	0.189	0.326915	0.328802	0.328008
Jumps	0.427	0.174026	0.153630	0.176264
Kids playing in leaves	0.089	0.424418	0.438694	0.426775
Playing on water slide	0.2	0.174321	0.177334	0.168675
Saving dolphines	0.145	0.229369	0.230782	0.212642
St Maarten Landing	0.313	0.039482	0.038585	0.040343
Statue of Liberty	0.192	0.070949	0.074132	0.068651
Uncut Evening Flight	0.271	0.251676	0.249076	0.253156
paluma jump	0.181	0.047268	0.044189	0.048565
playing ball	0.174	0.258244	0.274092	0.239955
Notre Dame	0.235	0.223917	0.217358	0.229265

Air Force One	0.318	0.064999	0.064230	0.066812
Fire Domino	0.13	0.004711	0.014701	0.002603
car over camera	0.372	0.038549	0.040747	0.035693
Paintball	0.32	0.233006	0.233086	0.224332
mean	0.234	0.155	0.162	0.152
relative to upper bound	52 %	34.44&	35.75&	34%
relative to average human	75 %	49.84%	51.70%	49%

5.2 Future Work

As the models with which experiments were carried out are not giving very promising results, I aim to work on a LSTM model for video summarization as mentioned in [4]. The figure is as follows:



[4] mentions very promising results given by the above model. I hope to emulate the same to further my understanding of video summarization.

Chapter 6

Conclusion

Through the project, I learnt the following

- Structuring a research problem by surveying the existing literature and then developing an intuition about to approach solving the problem in concern.
- The role played by features and how engineering features for the task at hand is important to learn a good model.
- Techniques used for summarization include CNN's for feature extraction, Uniform sampling, colour histogram.

References

[1] M. Gygli, H. Grabner, H. Riemenschneider, and L. Van Gool,Creating summaries from user videos, in *European conference on computer vision.* Springer, 2014, pp. 505-520.

[2] Y. Song, J. Vallmitjana, A. Stent, and A. Jaimes, Tvsum: Summarizing web videos using titles, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*2015, pp. 51795187.

[3] S. E. F. De Avila, A. P. B. Lopes, A. da Luz, and A. de Albuquerque Arajo, Vsumm: A mechanism designed to produce static video summaries and a novel evaluation method, *Pattern Recognition Letters*, vol. 32, no. 1, pp. 5668, 2011.

[4] Ke Zhang, Wei-Lun Chao, Fei Sha, and Kristen Grauman, ”Video Summarization with Long Short-term Memory,” in *Proceedings of the European Conference on Computer Vision (ECCV),* 2016, pp. 1-17

[5] (2017) Video summarization repo. [Online]. Available: repo link