Summarizing Videos with Attention

Arithmer Dynamics Div. Christian Saravia

2020/12/15

Christian Martin Saravia Hernandez

- Graduate School
  - Universidad Peruana de Ciencias Aplicadas
  - Bs. Software Engineering
- Former Job
  - Ficha Inc
    - Algorithm Development Engineer
      - Research & application of deep learning in computer vision problems
- Current Job
  - Application of machine learning / deep learning to computer vision problems
    - Object detection
    - Object classification
  - Research topics
    - Attention & Transformers

Purpose of this material

- Explore a solution to the task of video summarization using attention.

Contents

Motivation

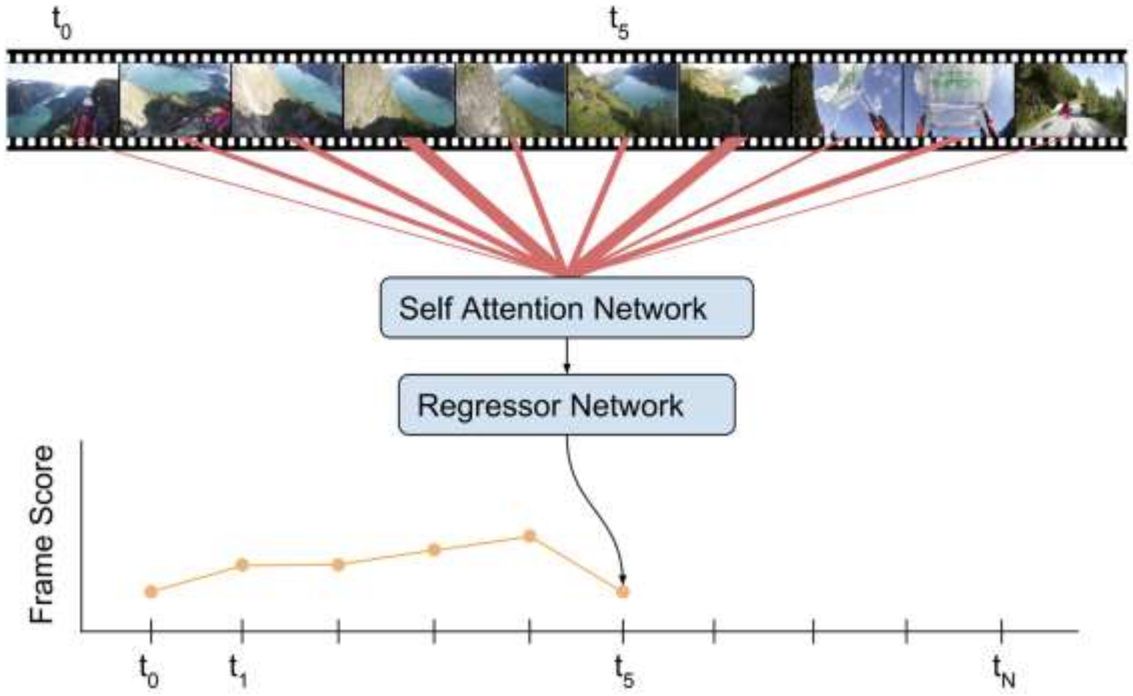
- Early video summarization methods were based on unsupervised methods, leveraging low level spatio-temporal features and dimensionality reduction with clustering techniques.Success of these methods solely stands on the ability to define **distance/cost functions between the keyshots/frames with respect to the original video.**

- Current state of the art methods for video summarization are based on recurrent encoder-decoder architectures, **usually with bi-directional LSTM or GRU and soft attention.** They are computationally demanding, especially in the bi-directional configuration.

Contribution

1. A novel approach to sequence to sequence transformation for video summarization **based on soft, self-attention mechanism**. In contrast, current state of the art relies on complex LSTM/GRU encoder-decoder methods.

1. A demonstration that a **recurrent network can be successfully replaced with simpler, attention mechanism** for the video summarization.

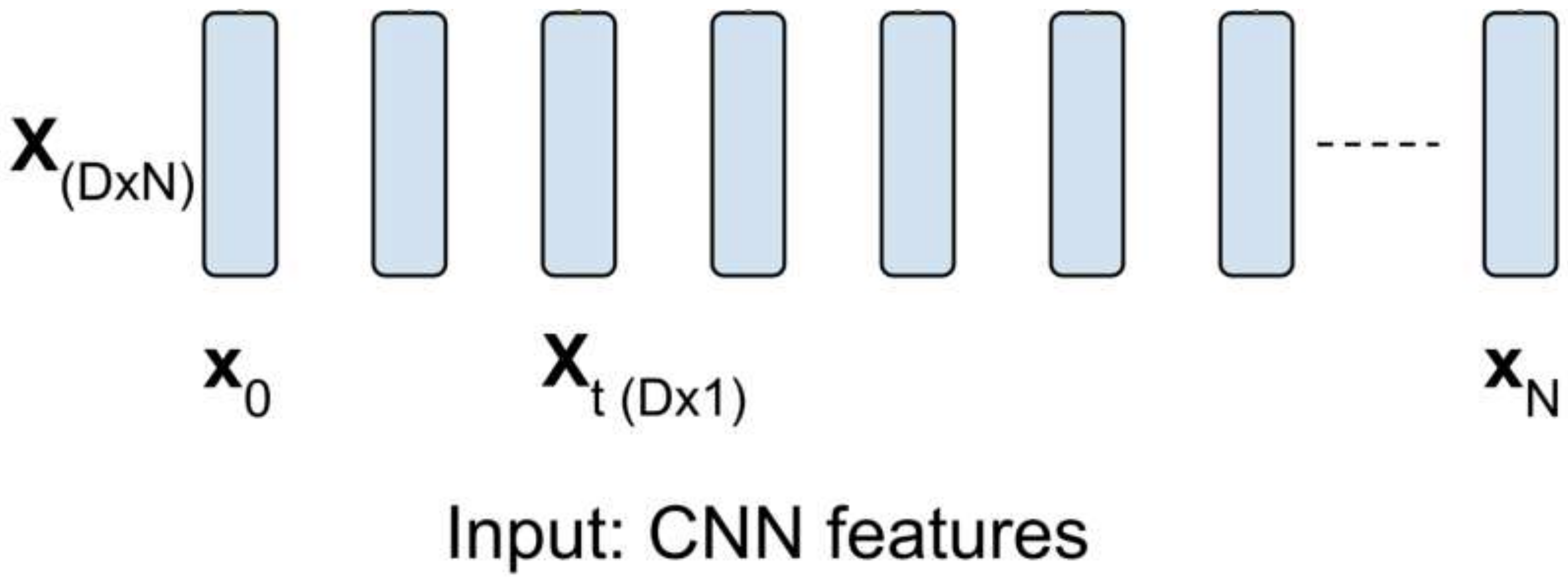


Dataset

- TVSum Dataset: https://github.com/yalesong/tvsum
- SumMe Dataset: https://gyglim.github.io/me/vsum/index.htm

Feature Extraction

- Given a time interval t, every 15 frames are collected in an ordered set X
- Each set then is used as input to GoogLeNet for feature extraction
- Then we extract the Pool 5 layer of GoogLeNet, which is a 1024 dimensional array (D = 1024).



Attention Network

- Unnormalized self-attention weight $e_{t,i}$ is calculated as an alignment between input feature X_t and the entire input sequence

$$e_{t,i} = s[(\boldsymbol{U}\boldsymbol{x}_i)^T(\boldsymbol{V}\boldsymbol{x}_t)] \qquad t = [0, N), \quad i = [0, N)$$

Where,

N: Number of frames

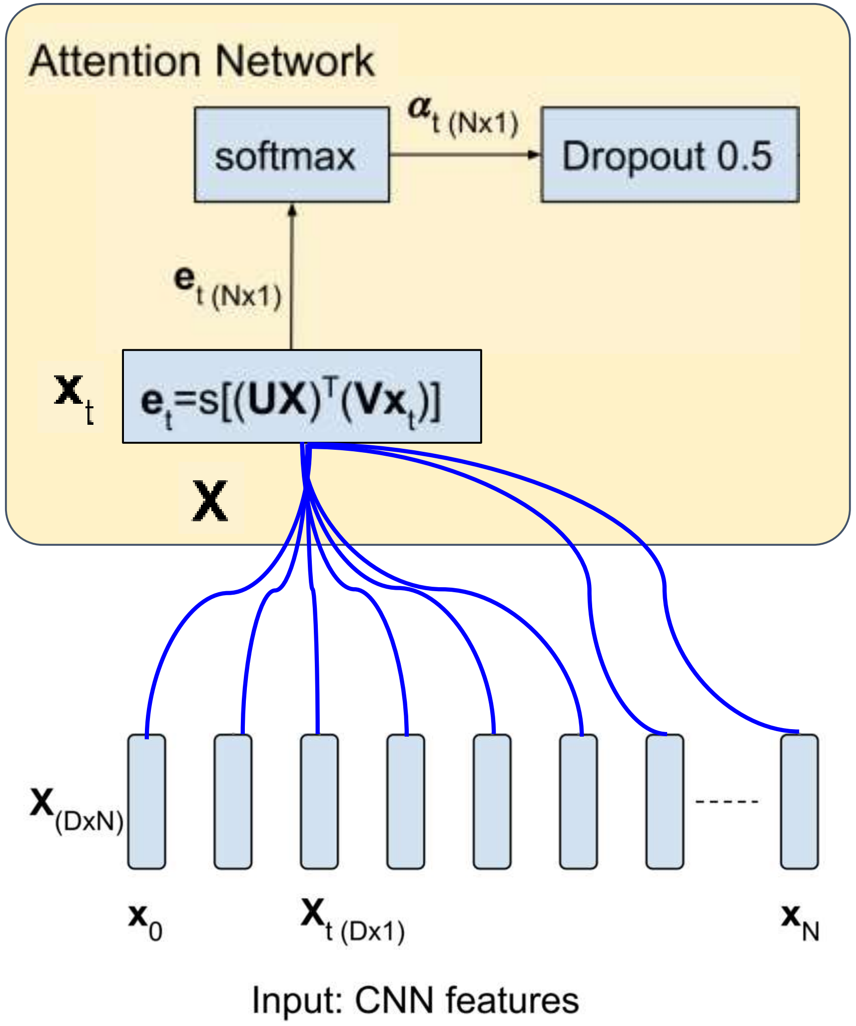
U, V: Network weight matrices estimated together with other parameters of the network during optimization

s: Scale paramenter

- The attention weights α_t are true probabilities representing the importance of input features with respect to the desired frame level score at the time t

$$\alpha_{t,i} = \frac{\exp(e_{t,i})}{\sum_{k=1}^{N}\exp(e_{t,k})}$$

Attention Network



Regressor Network

- Linear transformation C is then applied to each input and the results then weighted with attention vector α_t and averaged.
- The output is a context vector c_t which is used for the final frame score regression.

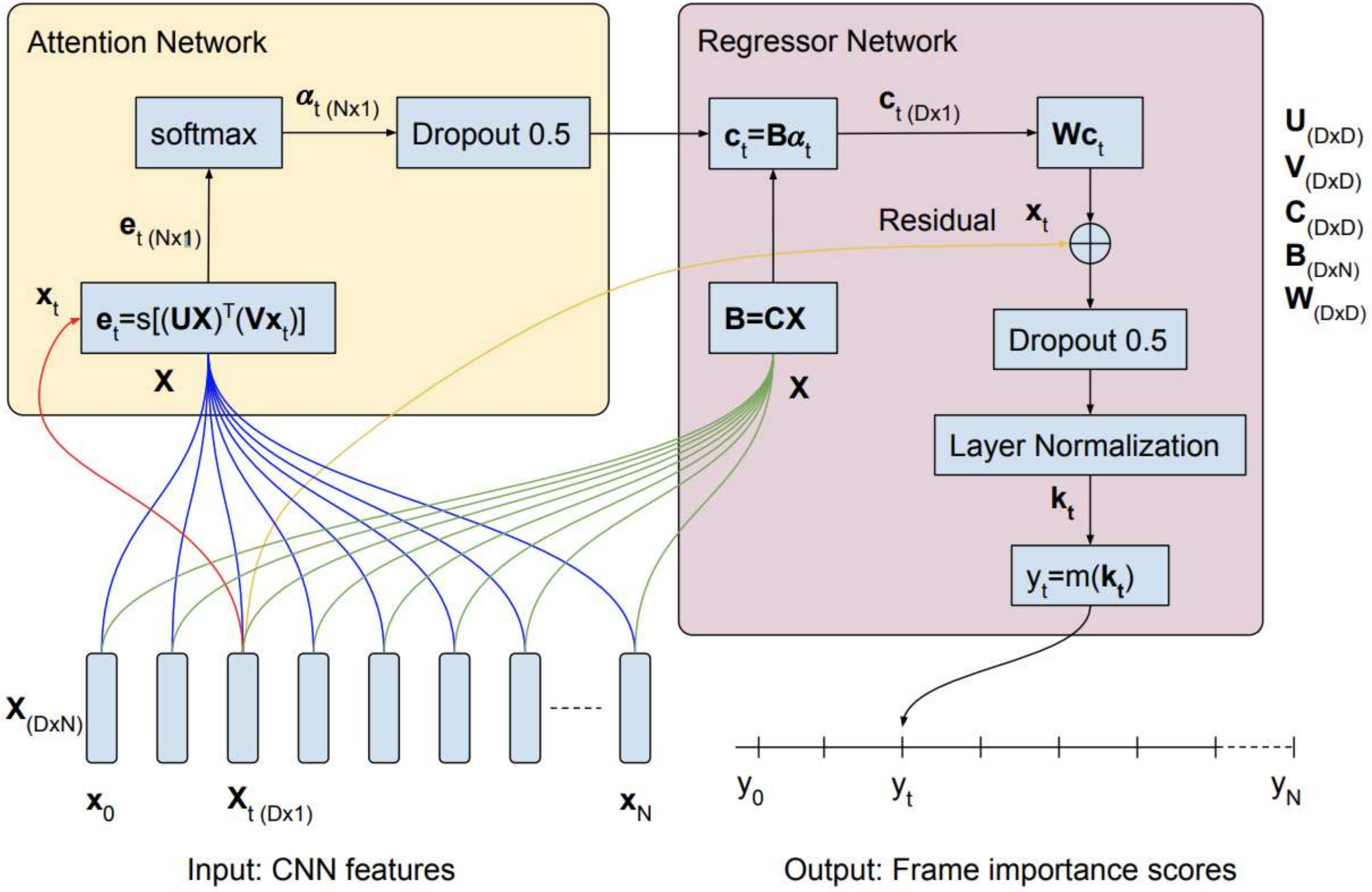
$$\boldsymbol{b}_i = \boldsymbol{C}\boldsymbol{x}_i$$

$$\boldsymbol{c}_t = \sum_{i=1}^{N} \alpha_{t,i}\boldsymbol{b}_i \qquad \boldsymbol{c}_t \in \mathbb{R}^D$$

- The context vector c_t is then projected by a single layer, fully connected network with linear activation and residual sum followed by dropout and layer normalization.

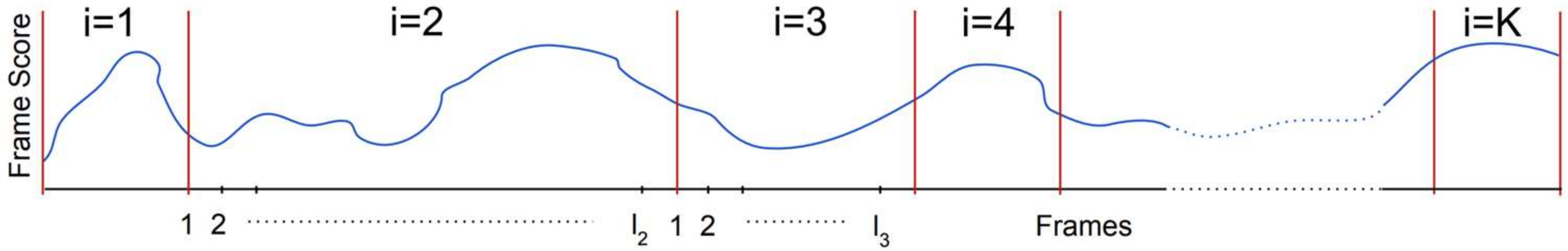
$$\boldsymbol{k}_t = norm(dropout(\boldsymbol{W}\boldsymbol{c}_t + \boldsymbol{x}_t))$$

Regressor Network



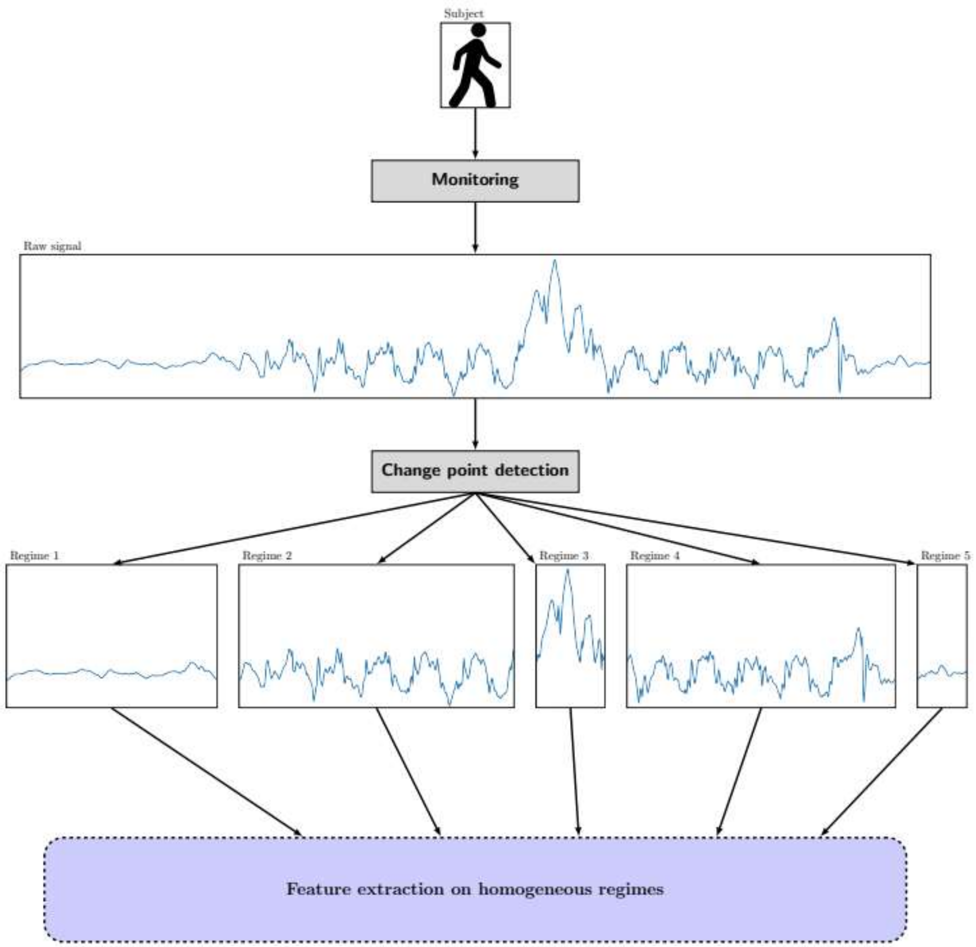
Inference

- The output of the model VASNet is a probability of importance per frame
- This probability must be analyzed in the range of the scene it corresponds
- However to get the number of frames is relative per video
- The problem to find the frames where a change a scene exist is called **changepoint detection.**
- For the datasets used, the changepoints (cps) are already calculated by using KTS algorithm with hyperparameter tuning



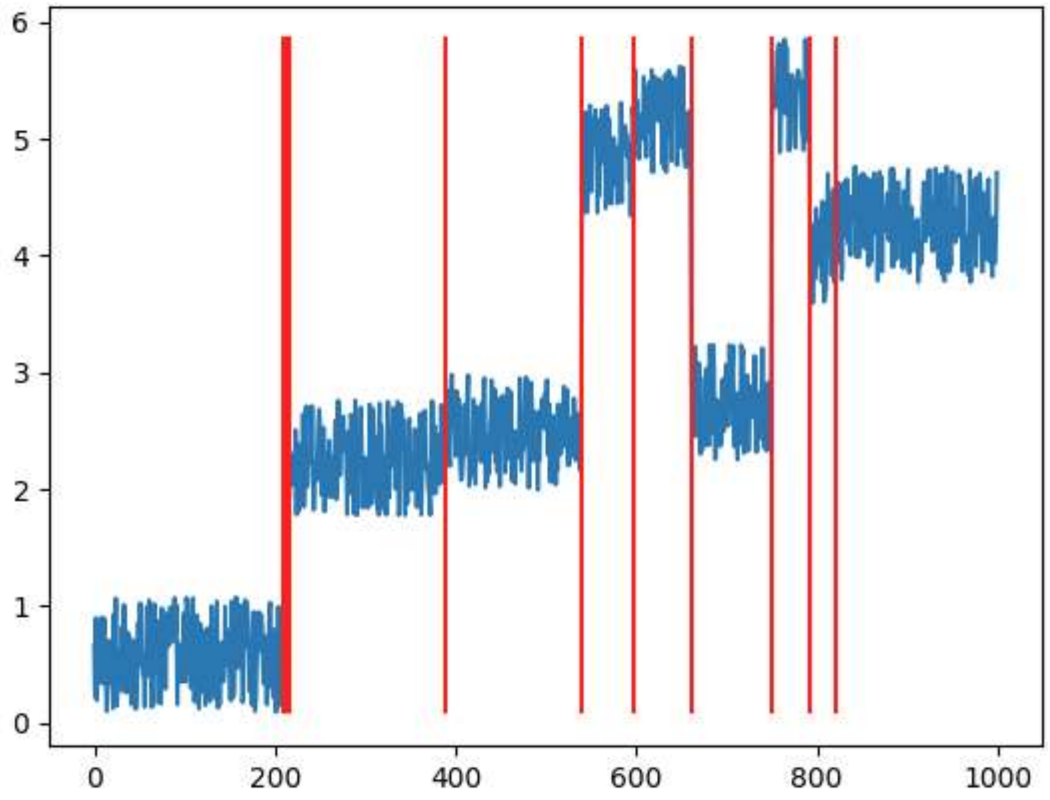
Changepoint detection

- In statistical analysis, change detection or change point detection tries to identify times when the probability distribution of a stochastic process or time series changes. In general the problem concerns both detecting whether or not a change has occurred, or whether several changes might have occurred, and identifying the times of any such changes.



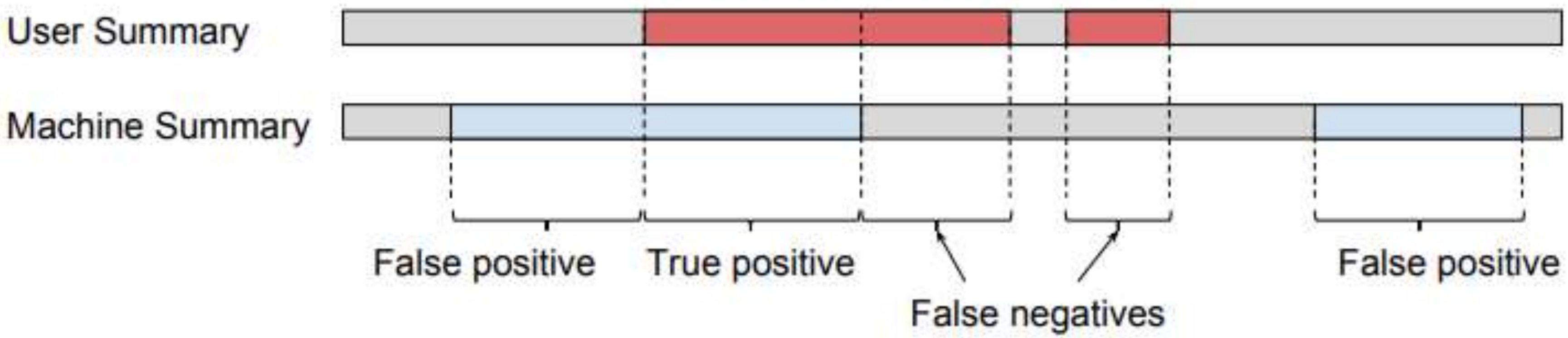
Kernel Temporal Segmentation (KTS)

- Kernel Temporal Segmentation (KTS) method splits the video into a set of non-intersecting temporal segments.
- It treats the cps detection as a dynamic programming problem.
- The method is fast and accurate when combined with high-dimensional descriptors.



Measuring method

P: Precision
R: Recall
F Score: [2 * P * R / (P + R)] * 100



Dataset Results

Method	SumMe		TvSum	
	Canonical	Augmented	Canonical	Augmented
dppLSTM [40]	38.6	42.9	54.7	59.6
M-AVS [15]	44.4	46.1	61.0	61.8
DR-DSN$_{sup}$ [42]	42.1	43.9	58.1	59.8
SUM-GAN$_{sup}$ [23]	41.7	43.6	56.3	61.2
SASUM$_{sup}$ [35]	45.3	-	58.2	-
Human	64.2	-	63.7	-
VASNet (proposed method)	**49.71**	**51.09**	**61.42**	**62.37**

Dataset Results

- Long video: https://youtu.be/873CBVbPJVE
- Summarize: https://youtu.be/weW4memH3Dg
- Full playlist: https://www.youtube.com/playlist?list=PLEdpjt8KmmQMfQEat4HvuIxORwiO9q9DB

Reference

- VASNet: https://arxiv.org/pdf/1812.01969.pdf
- VASNet official implementation: https://github.com/ok1zjf/VASNet
- KTS implementation: https://github.com/TatsuyaShirakawa/KTS
- Video summarization datasets and review: https://hal.inria.fr/hal-01022967/PDF/video_summarization.pdf
- Issue on testing on own videos: https://github.com/ok1zjf/VASNet/issues/2