Enhancing and Re-Purposing TV Content for Trans-Vector Engagement (ReTV)
H2020 Research and Innovation Action - Grant Agreement No. 780656

Enhancing and Re-Purposing TV Content for Trans-Vector Engagement

Deliverable 3.2 (M20)

Content Adaptation, Re-purposing and Scheduling

Version 1.0

This document was produced in the context of the ReTV project supported by the European Commission under the H2020-ICT-2016-2017 Information & Communication Technologies Call Grant Agreement No 780656

DOCUMENT INFORMATION

Delivery Type	Report
Deliverable Number	3.2
Deliverable Title	Content Adaptation, Re-purposing and Scheduling
Due Date	M20
Submission Date	August 31, 2019
Work Package	WP3
Partners	GENISTAT, CERTH, MODUL Technology
Author(s)	Basil Philipp, Krzysztof Ciesielski (GENISTAT), Konstantinos Apostolidis, Evlampios Apostolidis, Alexandros Metsai, Eleni Adamantidou, Damianos Galanopoulos, Vasileios Mezaris (CERTH), Lyndon Nixon (MODUL)
Reviewer(s)	Martin Gordon (RBB)
Keywords	Content Adaptation, Content Re-purposing, Video Summarization, Content Scheduling
Dissemination Level	PU
Project Coordinator	MODUL Technology GmbH Am Kahlenberg 1, 1190 Vienna, Austria
Contact Details	Coordinator: Dr Lyndon Nixon (nixon@modultech.eu) R&D Manager: Prof Dr Arno Scharl (scharl@weblyzard.com) Innovation Manager: Bea Knecht (bea@zattoo.com)

Revisions

Version	Date	Author	Changes
0.1	30/6/18	V. Mezaris, K. Apostolidis	Created template and ToC
0.2	29/7/19	E. Apostolidis, E. Adamantidou, A. I. Metsai	Added parts related to learning-based video summarization
0.3	30/7/19	K. Apostolidis	Added parts related to non-learning-based video summarization
0.4	31/7/19	D. Galanopoulos, K. Apostolidis	Added parts related to text-to-video matching
0.5	15/8/19	K. Ciesielski, B. Philipp	Added section on viewer profiling
0.6	19/8/19	L. Nixon	Added update on Metadata and Vocabulary Interoperability
0.7	23/8/19	M. Gordon	QA review
0.8	26/8/19	K. Ciesielski, K. Apostolidis	Post-QA revision
0.9	27/8/19	L. Nixon	Corrections

Statement of Originality

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

This deliverable reflects only the authors' views and the European Union is not liable for any use that might be made of information contained therein.

Contents

EXECUTIVE SUMMARY

In the ReTV project, the objective of our work in WP3 is to deliver components for content adaptation, re-purposing, scheduling and recommendation based on the annotations, metrics and predictions of WPs 1 and 2. To support viewer-level recommendation and audience targeting, we define viewer profiles (T3.2). Since the data produced in the different WPs may follow different metadata models and vocabularies, we also ensure interoperability through the definitions of mappings of properties and values across metadata specifications (T3.1). This deliverable reports on the final results of these first two tasks, which are a precondition for the subsequent work in implementing re-purposing, recommendation and scheduling components. This deliverable also reports on the progress in providing those content re-purposing, recommendation and scheduling components (T3.3 and T3.4). These form integral parts of the TVP and are used in all scenarios defined in WP5 and WP6 where they will enable the automatic creation of summaries and finding the optimal publishing strategies for content.

ABBREVIATIONS LIST

Abbreviation	Description
API	Application Programming Interface: a set of functions and procedures that allow the creation of applications which access the features or data of an application or other service.
CNN	Convolutional Neural Network: a type of artificial neural network.
DCNN	Deep Convolutional Neural Network: a type of artificial neural network.
DCT	Discrete Cosine Transform: a transformation that expresses a finite sequence of data points in terms of a sum of cosine functions oscillating at different frequencies.
DTR	Dilated Temporal Relational unit: a convolutional module for capturing temporal dependencies on various time windows.
EPG	Electronic Program Guides: menu-based systems that provide users of television with continuously updated menus displaying broadcast programming or scheduling information for current and upcoming programming.
HTTP POST/GET	Types of method in the Hypertext Transfer Protocol (HTTP). The HTTP POST method is used to send data to a server to create/update a resource. The HTTP GET method is used to request data from a specified resource.
GAN	Generative Adversarial Network: a deep learning architecture where two separate neural networks compete against each other.
GRU	Gated Recurrent Unit: a type of recurrent neural network.
IPTV	Internet Protocol Television: is the delivery of television content over Internet Protocol (IP) networks.
JSON	JavaScript Object Notation: a data-interchange format.
KTS	Kernel Temporal Segmentation: a video fragmentation algorithm.
LSTM	Long Short Term Memory networks: a type of recurrent neural network.
MSE	Mean Squared Error.
MTL	Multi-task learning: a field of machine learning in which multiple learning tasks are solved at the same time, exploiting commonalities and differences across tasks.
OTT	Over The Top: content providers that distribute streaming media as a standalone product directly to viewers over the Internet, bypassing telecommunications that traditionally act as a distributor of such content.
RDF	Resource Description Framework: a method for conceptual description or modeling of information that is implemented in web resources.
REST	Representational State Transfer: an architectural style that defines a set of constraints to be used for creating web services.
RNN	Recurrent Neural Network: a type of an artificial neural network.
TVoD	Transactional Video on Demand: a distribution method by which customers pay for each individual piece of video on demand content.
URL	Uniform Resource Locator: a reference to a web resource that specifies its location on a computer network and a mechanism for retrieving it.
VAE	Variational Auto-Encoder: a generative neural network that models a data distribution.

1. Introduction

This document summarizes the current state of the WP3 tasks in the first phase of the ReTV project (month M20). Section 2 discusses task T3.1 (Metadata and Vocabulary Interoperability) led by MODUL Technology. Section 3 discusses task T3.2 (Viewer Profiling) led by Genistat. Section 4 discusses task T3.3 (Content Adaptation and Re-purposing) led by CERTH. The outputs of all tasks are preconditions for the progress in the task on content recommendation and scheduling (T3.4), which is reported in Section 5. Finally, Section 6 contains conclusions and the future outlook for all WP3 tasks.

2. Metadata and Vocabulary Interoperability

We have provided for a shared metadata model and vocabulary space in the work reported in deliverable D3.1, where we defined mappings between ReTV's content annotation model (D1.1) and other standards and specifications for Web multimedia and TV audiovisual content. In data collection, we have already implemented such a mapping in the case of Europeana Data Model (EDM) so that content items from the Europeana API can be ingested into our metadata repository. With regards to promoting best practices in media annotation, we presented ReTV at the EBU Metadata Network meeting (June 2019), the community which also uses and promotes the EBU Core metadata model. Shared vocabularies are also used as targets of annotation, as reported in deliverable D1.2 (Semantic Knowledge Base or SKB). We use the SKB as a local knowledge graph for both named entities (drawn from DBPedia and WikiData) and for lexical keywords (drawn from OmegaWiki), with links to external and globally unambiguous identifiers.

3. Viewer Profiling

As we stated in the previous deliverable (cf. D3.1), the crucial aspect of the viewer profiling is to make sure that all relevant entities recognized by prediction models (i.e. users behaviour and interests, as well as TV content, events and social media activities) are described in the same feature space and can be related to each other (for instance, that we can model the relationship between user interests and content categories or sport event disciplines). In this vein, all of the works for viewer profiling in the reported time period were executed in parallel with the recommendation models related tasks (cf. section 5). Specifically, we carried out a series of experiments to verify the hypothesis that implicit audience segmentation learned by a recommendation model can be used to create explicit segmentation, when the latter is required (e.g. for a direct targeting a subset of users with a given piece of advertisement).

However, it is worth stressing here that in most ReTV recommendation scenarios it is sufficient to operate on the implicit segmentation learned by a predictive model. Such implicit segmentation follows the requirement mentioned at the beginning of this section, i.e. it describes each user's segment in the feature space defined by content-related attributes. In other words, it is a behavioral segmentation (implicitly) reflecting users interests. One of the advantages of such an approach is that once the richer, extensive content annotation is integrated, the feature space describing user preferences is also automatically extended, reflecting additional aspects of interests. One example could be face recognition, when we can automatically discover that user is interested in particular actors or politicians. Another advantage of our approach is that it is a fuzzy segmentation, i.e. each user can belong to multiple segments with varying level of segment similarity. In our opinion, this is a more precise segmentation description, since naturally behavioral segments are not always disjoint (e.g. sport fans segment can overlap with some celebrity fans segment).

In the following sections we describe experiments that show the relationship between the implicit segmentation derived from the recommendation model (cf. section 5) and the explicitly measured users interest in various TV content categories.

3.1. Data

We took Zattoo users session data (the subset of 60k Swiss users over 3 months) for 7 channels of the public Swiss TV. For the first segmentation experiment, we didn't use any user or content related attributes. The only input information was the interaction between the user and the content (i.e. the programs watched). We measured the strength of the user-content information as the fraction of the program that has been watched (the number between 0 [instant zapping] and 1 [program fully watched]). For the evaluation part (but not for the model itself) we used EPG information about program categories. The goal was to verify whether the model is able to discover user interests patterns even though this information was not explicitly provided as an input.

3.2. The Modelling Approach

We treat segmentation as a counterpart to the recommendation model that tries to match content pieces (trailer, programs, social web posts etc) to the user/segment profile. We want user profiles and segment profiles to be fully exchangeable. I.e. described in the same space of

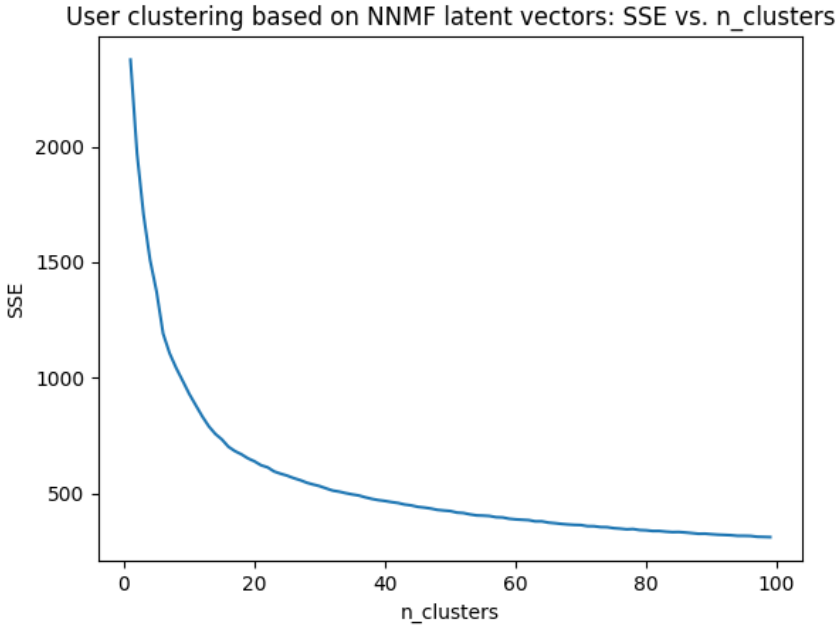

Figure 1: Inter-cluster variance (SSE) vs. the number of clusters

attributes. The advantage of such an approach in the ReTV context is that we can use more detailed information when available (e.g. for registered TV channel users), and more granular, privacy-preserving, segment information otherwise (e.g. for social media).

We built a collaborative filtering model based on non-negative matrix factorization. Thus, every user is represented by a vector in latent, low-dimensional space. NNMF model ($D = W * H$ [1]) can be treated as clustering model itself, where each row of H matrix represent individual user, and latent vector represents cluster membership (fuzzy clustering). We wanted to be more elastic with the number of segments (clusters), so we build 10-dimensional latent NNMF model and then clustered latent vectors representing users with K-Means algorithm. It should be noted that any other algorithm can be used here (esp. fuzzy C-Means, since we want to allow user to belong to multiple segments, or hierarchical algorithm if we need a hierarchy of segments). We used crisp clustering here only for simplicity of the presentation of results. Figure 1 presents the inter-cluster variance with respect to the different number of clusters. Its value stabilizes somewhere between 60 and 90 segments (much higher than latent space dimension).

3.3. Results and Discussion

As briefly mentioned earlier, for the evaluation part (but not for the model training), we built an explicit user profile, based on EPG data. The goal was to verify whether implicit information expressed only by user-content information (i.e. training data for clustering model) was also able to capture explicit attributes that can describe resulting segments.

After matching of the content (programs) with EPG information and aligning it with the data used to train the model (user-content interactions), each user was described by the distribution

[1] https://en.wikipedia.org/wiki/Non-negative_matrix_factorization

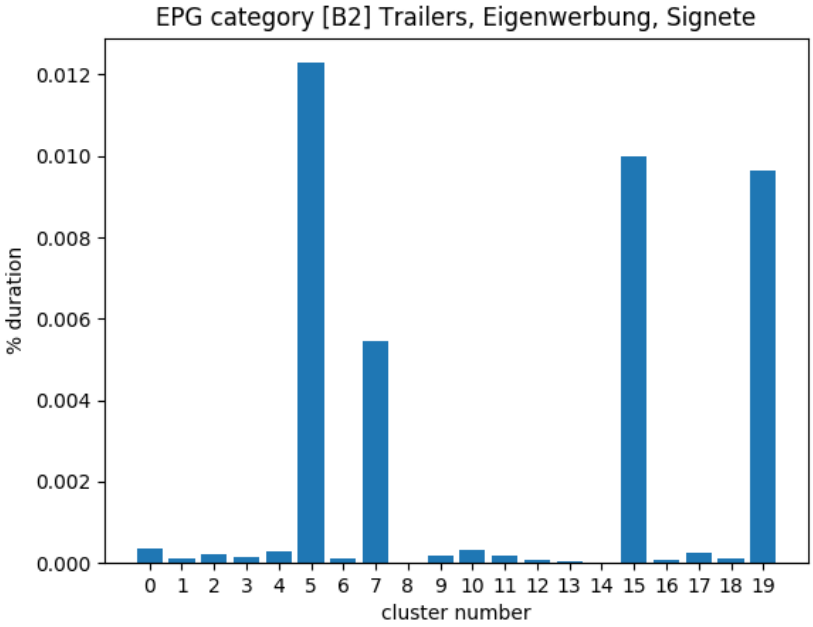

Figure 2: Trailers: user engagement measured as duration spent on a given content category

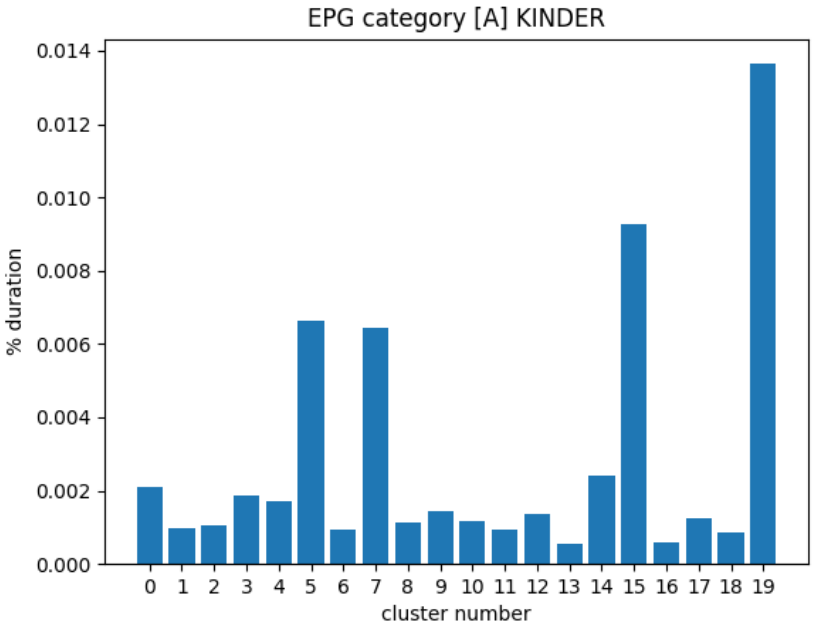

Figure 3: Kids programs: user engagement measured as duration spent on a given content category

of the EPG categories with which he/she interacted. In other words, we calculated the total duration spent on various EPG categories and normalized the values.

Below we present only results for the EPG categories. However there are definitely more implicit patterns in the data that can be used to describe user segments, such as:

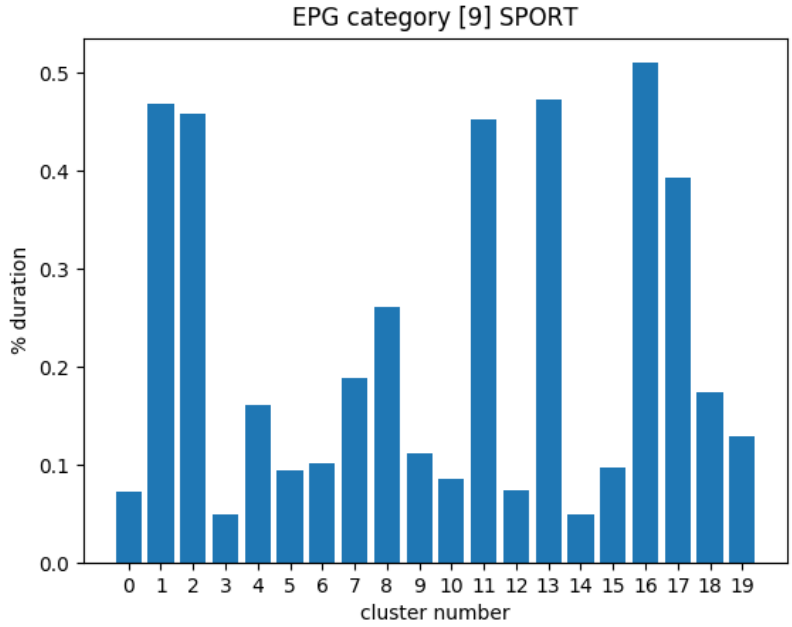

Figure 4: Sport: user engagement measured as duration spent on a given content category

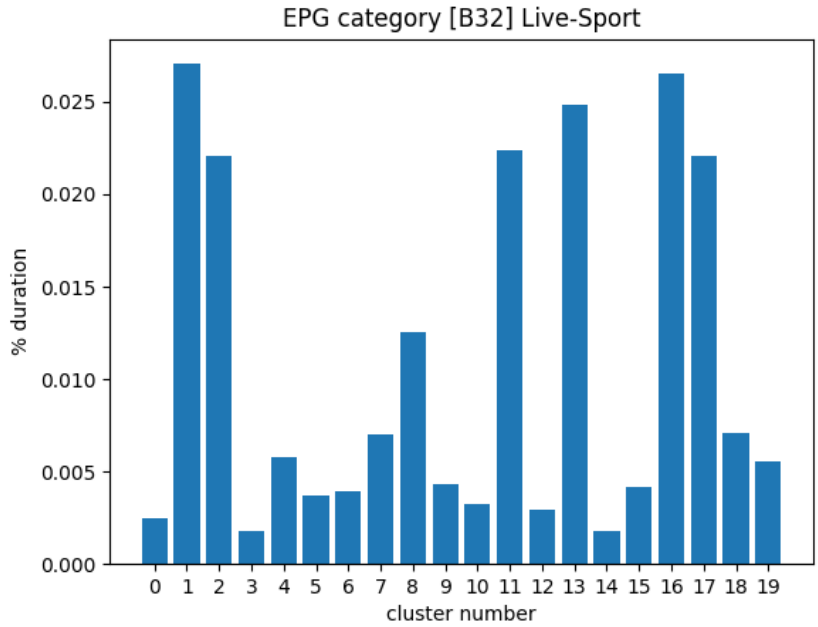

Figure 5: Sport ads: user engagement measured as duration spent on a given content category

- geographical location
- language
- typical viewing time (e.g. prime-time audiences) or day/day-of-week
- size and the structure of the household (esp. kids/no kids)

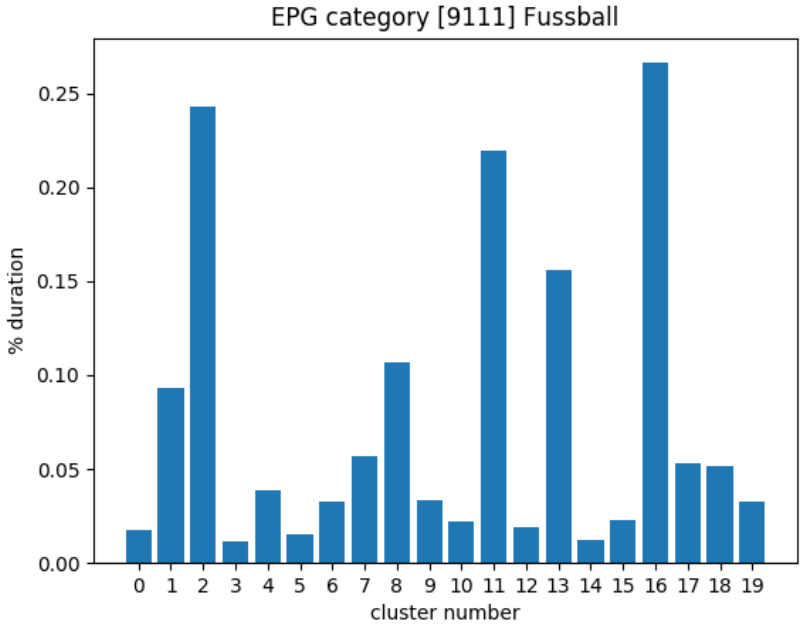

Figure 6: Football: user engagement measured as duration spent on a given content category

- socio-demographic attributes (age, gender)

The next step would be to execute similar analyses for viewing times, geolocation, socio-demographics and household structure. Such attributes can be used in the Content sWitch scenario (better targeting of trailers and ads, both segment-level and individual user level) as well as in the 4u2 chatbot scenario (e.g. segment-based recommendation for social media) - for more about the consumer scenarios, see deliverable D6.2.

For the sake of simplicity, we present results for 20 segments only (even though the optimal number of clusters is higher, as mentioned above).

For the Content sWitch scenario, we observe that only a few segments are in general interacting with trailers (cf. Figure 2).

We observe a few segments interested in kids content. We can assume that users from households with kids are included mostly in these segments (cf. Figure 3).

Sport is in general the most popular category, however we observe differences between segments (cf. Figure 4).

Not surprisingly, ads related to sport are correlated with interests in sport content (cf. Figure 5).

Going deeper into individual sport disciplines, we see not only that overall engagement differs between the sport disciplines, but also that there are segments more interested in one discipline, and segments more interested in multiple disciplines. For example in Figure 6, when comparing the distribution to the one shown in Figure 4, we can observe that segments 1 and 17 are generally interested in sport but less in football.

In conclusion, we see the close resemblance between the explicit and implicit segmentation learnt by the recommendation model. In section 5 we describe the underlying model in more

detail.

4. Content Adaptation and Re-Purposing

4.1. Problem Statement

The technologies developed in this task are focused on two directions: video adaptation and video content re-purposing. This adaptation is based on the characteristics of the vector that the video's shorter version will be published in, as well as the attributes of the target viewer segment, leveraging information from task T3.2 of ReTV.

Regarding the first direction, the video content is adapted by generating shorter versions utilizing video summarization methods which aim to provide a short visual summary that encapsulates the flow of the story and the essential parts of the full-length video. According to [57], there are two fundamental types of video summaries: static video summary (a.k.a. video storyboard), which is a collection of video frames extracted from the original video, and dynamic video skimming, which is a set of short video clips, joined in a sequence and played as a video. One advantage of a video skim over a set of key-frames is the ability to include audio and motion elements that offer a more natural story narration and potentially enhance both the expressiveness and the amount of information conveyed by the summary. Moreover, it is often more entertaining and interesting to watch a skim rather than a slide show of key-frames [40]. On the other hand, key-frame sets are not restricted by timing or synchronization issues and, therefore, they offer much more flexibility in terms of organization for browsing and navigation purposes [6, 59]. Such technologies were developed for the purpose of video adaptation and are described in this deliverable.

The provision of a concise summary that adequately conveys the main concept of the video enables the viewer to quickly understand the main messages conveyed by it without having to watch the entire video. Given the plethora of video content on the Web and the limited time that a viewer can spend on deciding whether to watch or skip a video, an effective video summary enables the time-efficient browsing of large video collections and increases the potential of a video to be consumed. Given the above, video summarization aims to provide a short visual summary that encapsulates the flow of the story and the essential parts of the full-length video.

The application domain of automatic video summarization is fairly wide and includes the use of such technologies by video sharing platforms that constantly aim to higher viewer engagement and content consumption. Furthermore, video content management systems of media organizations exploit video summarization techniques to allow effective indexing, browsing and retrieval of video content. Last but not least, techniques for parameterizable/personalizable video summarization that take into account the viewers' needs, enable effective sharing of video content across different channels (e.g. 4G/5G WANs, local LANs, etc. with various data transmission capacity) and presentation devices (e.g. desktops, laptops, tablets, smartphones), in forms (storyboards, skims, excerpts) that are tailored to the needs of the targeted audiences, thus facilitating content presentation and consumption.

Regarding the second direction, video content is re-purposed by enhancing it with external information and advertisements based on the target viewer's segment and vector characteristics. To this end, we developed a method for text-to-video matching. Such a method can be applied

for the targeted advertisement placement and advertisement replacement requirements of the ReTV project (i.e., matching the description of the ads with regions of the video suitable for ad placement).

4.2. State-of-the-Art Survey

4.2.1. Video Summarization

Several approaches were proposed over the last couple of decades, for addressing the task of video summarization. For a long time, the relevant research area was dominated by methods that select the key parts of the video (either key-frames for static video summaries, or key-fragments for generating dynamic video skims) based on the extraction and processing of low-level visual features from the video frames. These methods include: algorithms that assess the visual similarity over sequences of frames (e.g. [65, 68, 34, 15]; clustering-based techniques that group frames according to their visual similarity and extract key-frames from the clusters' centres (e.g. [23, 12, 68, 7, 1, 10]); dictionary learning approaches which aim to approximate the gap between low-level visual features and high-level visual semantics, and perform semantic-driven selection of the video summary parts (e.g. [17, 41, 42, 74]); and visual attention modeling that imitates the human attention mechanism that is used to spot the most important parts of the video for generating the summary (e.g. [29, 36, 14, 71]). Early supervised machine learning methods aimed to: capture the underlying frame selection criterion from human-created summaries to produce video summaries that meet human expectations (e.g. [24, 26]); exploit auxiliary information, such as the video title or metadata, to extract the semantically-related parts of the video (e.g. [33, 55, 51]); and directly optimize multiple objectives for video summarization, such as representativeness, relevance, importance, diversity, uniformity, storyness and actioness (e.g. [25, 38, 16]).

In addition to the above, and based on the processing capacity of the modern Graphic Processing Units and the rise of deep network architectures, a number of deep learning video summarization approaches were introduced, with the majority of them being trained in a supervised manner, i.e. using ground-truth summaries. In this context, the learning efficiency of Convolutional Neural Networks (CNN) was exploited for the needs of supervised video summarization. In [50] video summarization is addressed as a weakly-supervised learning problem and solved via a flexible deep 3D convolutional neural network architecture that learns the notion of importance using only video-level annotation. [54] tackles video summarization as a sequence labeling problem and performs key-frame-based video summarization using fully convolutional sequence models. [19] combines a soft, self-attention network with a two-layer fully connect network to process the CNN features of the video frames and compute frame-level importance scores that are subsequently used for key-fragment selection. [48] uses deep video features for encoding various levels of content semantics (e.g. objects, actions, scenes) and then, a deep neural network that maps videos and their descriptions to a common semantic space is jointly trained with associated pairs of videos and descriptions. Following, a summary is created by applying a clustering-based process on the extracted deep features from the video segments.

Other supervised techniques utilize advanced variations of Recurrent Neural Networks (RNN) to capture the temporal dependency over sequential data (Long Short-Term Memory (LSTM) units [27] and Gated Recurrent Units (GRU) [9] have shown remarkable performance in various problems that are inextricably linked to the temporal domain, such as speech recognition

and text transcription), several RNN-based architectures have been proposed for video summarization and represent the current state-of-the-art on this research field. [66] is the first work that introduced the use of LSTM to model the temporal dependency among frames and compute frame-level importance scores, while the diversity of the video summary was enhanced by modeling the pairwise frame repulsiveness using the Determinatal Point Process algorithm. [30] formulates video summarization as a sequence-to-sequence learning problem and proposes an LSTM-based encoder-decoder network with an intermediate attention layer to obtain video summaries. Another approach, that combines attention with generative adversarial learning is described in [22]. The typical encoder-decoder seq2seq model is replaced by a special attention-based seq2seq model that defines the different fragments of the video. The model is combined with a 3D-CNN classifier (the discriminator of the adversarial framework) which guides the training by judging whether a fragment is from a ground-truth or a generated summary. [20] goes one step further by introducing an architecture with memory augmented networks, which utilizes an external memory to record visual information of the whole video with high capacity. Thus, video summarization is tackled in a more global manner that involves the extraction of knowledge about the temporal interdependency across the entire video. [72] proposes a 2-layer LSTM architecture where the first layer extracts and encodes data about the video structure and the second layer uses this data to define the key-fragments of the video. This work is extended in [73] to exploit the shot-level temporal structure of the video and compute shot-level confidence scores for producing a key-shot-based summary of the video. [69] describes a Dilated Temporal Relational (DTR) Generative Adversarial Network (GAN), where the generator contains LSTM and DTR units to exploit long-range dependencies at different temporal windows, and the discriminator is trained via a 3-player loss to distinguish between the learned summary and a trivial summary consisting of randomly selected frames.

Contrary to the above supervised approaches, a few unsupervised methods for automatic video summarization were also proposed. [43] addresses video summarization by selecting a sparse subset of video frames that optimally represent the input video. For this, a deep summarizer network is trained to minimize the distance between training videos and a distribution of their summarizations through a generative adversarial framework. [63] follows a similar approach that aims to maximize the mutual information between the summary and the original video. A bi-directional LSTM network performs frame selection and a variational auto-encoder is used to learn video representation and acts as generator. Finally, an information-preserving metric between the summary and the original video is estimated by a trainable couple of discriminators and a cycle-consistent adversarial learning objective. [67] performs a similar comparison with [43] but in an abstract semantic space, based on the intuition that, if the summary preserves the important and relevant information in the original video, then the two embeddings (in the abstract space) are similar. [75] formulates video summarization as a sequential decision-making process and develops a (encoder-decoder) deep summarization network that learns to produce diverse and representative video summaries via policy-gradient reinforcement learning and a novel diversity-representativeness reward function. [70] suggests an approach that extracts key motions of appearing objects in the video, and learns to produce a fine-grained object-level video summarization in an unsupervised and online manner. The authors of [54] describe an unsupervised variation of their model, that aims to increase the visual diversity of the selected key-frames. Finally, [53] introduces a new formulation to learn video summarization from unpaired data. For this, sports highlights, movie trailers and other professionally edited summary videos available online are collected and used to guide a generative adversarial process that aims to learn a mapping function of a raw video to a human-like summary.

4.2.2. Text to Video Matching

Image-to-text matching has been an active research direction in the multimedia research community for many years. This particular problem deals with content from different modalities, thus a common solution is the transformation of the typical image/video and text representations into a new common embedding space, in which the similarity between image and text can be directly measurable. Recent approaches address the problem by using dual networks to embed both images and text into the new latent space, e.g., in [58] authors propose to learn the common latent space using coupled inputs and class regression.

An important driver of scientific development in image-to-text matching, specifically focusing on video (rather than still images), has been the Ad-hoc Video Search task (AVS) of TRECVID. In the AVS task, the majority of the state-of-the-art approaches depend on the available set of visual concept detectors. The input videos are annotated by these detectors and the textual queries are decomposed using linguistic analysis in order to be correlated with the detectors. In [44] multiple pre-trained DCNNs are used to detect visual concepts of interest in the video content and a set of complex linguistic rules are used to extract meaningful concepts for a specific query. More recent approaches utilize the power of deep neural networks in both textual and visual component. The advantage of these approaches is that they do not depend on the usage of several and heterogeneous visual concept detectors. They directly encode both visual and textual content in a common latent space. In [13] a network consisting of fully connected layers and RNN-based layers (Gated Recurrent Units) along with different textual representations is used in order to embed sentences into the visual spaces of the images. In [18] a method for learning a visual-semantic embedding for cross-modal retrieval is presented by exploiting hard negatives examples and ranking loss functions in order to improve the performance of cross-modal retrieval.

4.3. Non-learning-based Video Summarization Approach

In the course of the ReTV project, and up to its 20th month, several different versions of video summarization methods were introduced and continuously adapted to newly presented challenges and requirements. This continuous effort can be outlined in three different milestone versions of our video summarization method: one that is based on semantic clustering (M6), a second one that ranks the video shots according to ReTV specific criteria and select segments from the top-ranked shots (M12), and finally a more elaborate learning-based one (M20). Each version was designed based on the feedback of several rounds of tests conducted by ReTV partners. In this section the non-learning-based versions (i.e., 1st and 2nd versions) are discussed, while in Section 4.4 the learning-based approach is examined.

In M6 of the ReTV project, a first baseline version of video summarization method was introduced. This method utilized the video fragmentation module of the Video Analysis (VA) service of WP1 to segment the video to sub-shots and select a single key-frame for each sub-shot. Then, using the concept-based annotation module of the VA service, semantic tags representing high-level concepts of the TRECVID-SIN video annotation concept pool [49] were assigned to the selected key-frames (the interested reader is referred to Section 4.2.3 of D1.1 regarding details on the concept-based detection method used for this). The inferred probabilities of the 323 concepts for each key-frame are treated as the feature vector of the key-frame, which in turn is used as input to the Affinity Propagation clustering algorithm [21]. For each resulting cluster, the key-frame which is closest to the centroid of the specific cluster is selected. The

clustering-based selection of key-frames is temporally sorted according to the position of the key-frames in the original video and then stitched together to produce the final summary of the video, in the Graphic Interchange Format (GIF) file format.

Table 1: Summary of the feedback collected from ReTV partners that tested the first version of the video summarization service.

Point ID	Description
#1.1	The static summarization format (i.e., a compilation of static key-frames) does not suit the ReTV project's requirements. Instead, a summarization format where the summary is comprised of actual video segments should be introduced.
#1.2	Instead of relying on the number of clusters estimated by the Affinity Propagation algorithm for the length of the summary, the user must be able to set a desired target duration. This is particularly important, since different summaries have to be generated for each considered target vector where they will later be published.
#1.3	Video segments with visual effects (e.g., TV news introductory animated graphics) or with overlaid text (e.g., lower third text banner found in TV news videos) should be avoided.
#1.4	Video segment which depict an anchorman or anchorwoman, TV news studios and TV news commentators should be avoided.
#1.5	Scenes were the camera does a close-up of a person speaking should be avoided.
#1.6	"Clean" key-frames frames should be preferred, i.e., video key-frames/segments with blurred content should be excluded.
#1.7	Scenes which repeat information already included in the summary should be excluded.

Following a first round of tests on the baseline video summarization method, and in accordance with the collected feedback (from the test done in WP6 as well as internal unofficial tests between ReTV partners), a new version of the video summarization method was designed based on shot ranking. According to this method, the video is segmented to shots using again the video fragmentation module of the VA service, and each shot is ranked with a value $r \in [0,1]$. The lower the r value, the more likely is the shot to be included in the summary. To rank each shot we use a variety of measures to meet the requirements collected from the testing of the previous summarization method. Specifically, to avoid selection of shots with visual effects (point #1.3), we utilize the concept detection module of the WP1 VA service, summing the probability scores of concepts *Synthetic_Images*, *Junk_Frame*, *graphic*, *Network_Logo*, *Overlaid_Text*, *Scene_Text*, *Text_labeling_people*, *Text_On_Artificial_Background* from the SIN concept pool. Our intuition is that the higher the probability score of these concepts for a shot, the higher will be the shot rank, thus the less likely to be included in the summary. Similarly, to avoid selection of shots with content related to specific parts of a TV news program (point #1.4), we take into consideration the probability scores of concepts *Anchorperson*, *Commentator_Or_Studio_Expert*, *Studio_With_Anchorperson*, *Female_Anchor*, *Male_Anchor*, *Female_Reporter*, *Male_Reporter* from the SIN concept pool. To avoid selection of "talking heads" shots (point #1.5), we utilized the probability scores of concepts *Face*, *Female-Human-Face-Closeup*, *Head_And_Shoulder*, *Speaking_To_Camera* from the SIN concept pool. In response to point #1.6 of the feedback, we introduced the computation of variance of the laplacian of each frame, a widely used method in the field of computer vision, to measure the blurriness of an image. To avoid selection of shots with blurry content we take into consideration the maximum value of this blurriness measure of all frames in the shot. Additionally, aiming to satisfy point #1.6 for moving content, i.e., avoid the selection of shots with extreme camera movement that might result in blurry content being included in the summary, we introduced the computation of Edge Change Ratio (ECR)

measure for all frames and we take into consideration the maximum value of the ECR measure of all frames in the shot.

The aforementioned procedure results in a ranked list of shots. The shot with the lowest rank is selected as a candidate to be included in the summary and is removed from the ranked list. We check if the candidate shot is visually similar to the set of already selected shots that will be included in the summary (point #1.7). To assess the visual similarity of shots we make use of the max pooled features of intermediate layers of the Places365 model, employed in the concept-detection module of the WP1 VA service and check whether the distance of two shots is above a pre-defined *min_visual_distance* threshold. Then, two different pre-defined thresholds, *min_shot_duration* and *max_shot_duration* are utilized, with the intend to not include very short or very long shots in the summary. The value of *min_shot_duration* was initially set to 1 seconds, while the value of *max_shot_duration* was set to 3. If the candidates shot's duration is greater than the *min_shot_duration* threshold and lower than the *max_shot_duration* threshold then the whole shot is included in the summary. If the candidates shot's duration is greater than the *max_shot_duration* then we select a *max_shot_duration* seconds segment of the shot for which its frames exhibit the minimum sum of blurriness and ECR measures, in an attempt to select the part of the shot with the most "clean" content (as defined in point #1.6). The procedure is repeated until the user-defined time budget for the summary is exhausted. Performing the summarization process on shot level, the generation of a dynamic summary (i.e. a skim video) is achieved (point #1.1).

Table 2: Summary of the feedback collected from ReTV partners that tested the second version of the video summarization service.

Point ID	Description
#2.1	A query-based summarization approach would help generate alternative versions of summaries for different target vectors and different viewer segments.
#2.2	A multi-video summarization approach would be useful specifically for NISV, e.g., generating a single summary for multiple video documentaries that concern biking.
#2.3	The rhythm of some generated summaries was characterized as "quite snappy", namely the shots were changing much too quickly. Since, this behaviour is desirable for certain scenarios, a user-defined parameter to control this should be introduced.
#2.4	For certain use-case scenarios, in the developed web graphical user interface when the user navigates over a thumbnail of a video, then a short version of the video summary must play.
#2.5	For certain use-case scenarios, a professional user must be able to re-organize an automatically generated summary suggestion, depending on his/her personal taste.
#2.6	The ability to include subtitles and background music in a video summary would be desirable.

After releasing the above described summarization method as a Video Summarization (VS) service, several rounds of subsequent tests took place. The further feedback of the ReTV content partners and the analysis of the corresponding technical requirements can be summarized in the following points (a summary of the feedback is also given in Table 2: Based on the requirements of point #2.1, the summarization method was extended with the ability to take into consideration a list a concept IDs from the adopted concept pools of ReTV. The algorithm creates a second ranked list, in a manner similar to how the ranked list is constructed for satisfying the first round of points. The lists are then combined by taking the geometric mean of the two ranks for each shot. This way, we achieve a query-based summarization approach, i.e. assigning more weight to the shots that contain the concepts of the user-provided

list. Similarly, in response to point #2.2 the summarization method was extended with the ability to receive multiple videos as input, considering the sum of shot sets of all videos as a sole video to produce a single summary. It should be noted that in the case of multi-video summarization, the chronological order of selected segments within the video is kept, while the order of the videos is not kept, i.e., videos are re-arranged based on their fitness to be included in the summary.

In response to point #2.3, a user-defined parameter that adjusts the rhythm of segments in the summary was introduced. In essence this parameter is a multiplier of the *max_shot_duration*, i.e., the maximum allowed duration of a shot segment that is selected to be included in the summary. Setting this parameter to a value well below 1 will result in the *max_shot_duration* threshold to be decreased and a summary with fast alternating segments will be generated. Setting this parameter to a value well above 1, will result in a summary whose segments will be longer, i.e., a summary with a slower pace of alternating segments. Regarding point #2.4, we extended the summarization algorithm to also generate a short version of the summary, by utilizing only the top three ranked shots, with a limited time budget of up to 6 seconds. This short version of the summary is referred to as *preview* in the following. Additionally, we select the key-frame of the lowest-ranked shot of it as a thumbnail. Note that, since all three outputs of the VS service, i.e. the summary, the preview and the thumbnail, start with the same (lowest ranked) shot, we make sure that the user has a smooth experience in the user interface in which the summary will be employed. For example, consider the following scenario: a user sees the thumbnail of a video; if she/he navigates the mouse pointer on the thumbnail, the preview will start to play; if she/he clicks on the thumbnail, the summary will start to play. Both the summary and the preview will start with content that is visually very similar to what the thumbnail depicts.

In response to point #2.5, a re-arrangement of the workflow of the summarization method was extensively discussed and realized. Specifically, after selecting the segments that the summary is comprised of, the VS service returns a summary script in the form of a JSON file, instead of the actual video summary file. This JSON file (i.e., *summary script*) contains an array with the start and end times of each selected segment. The summary script can be transferred to a web-page with a graphical user interface where the user can modify the selected segments and re-submit the script to a different endpoint of the VS service. This will in turn render the final summary to a video file. The API details for this workflow are analyzed in Section 4.7.

Finally, Comment #2.6 was discussed in ReTV meeting in Thessaloniki and it was decided that such features will be made available by the front-end that utilizes the VS service.

Table 3: Open issues for summarization.

Point ID	Description
#3.1	The core story must be traceable. Video segments which do not support the core story should be excluded.
#3.2	The summary must recognize the story of the video. The viewer must not feel as though relevant parts were left out.
#3.3	Certain summaries show uninteresting segments.

After all the feedback collected and analysed, and the corresponding adjustments and extensions on the implemented method, there are still some open issues (Table 3); It is evident that these revolve around a method being able to "understand" the "story" of the original video. Thus, research on a more elaborate method that takes into consideration in a better

way the temporal relation of frames was decided. Our efforts led to the implementation of a learning-based summarization approach and an extension of it, both of which are presented in the following sections.

4.4. Learning-based Video Summarization Approaches

4.4.1. Unsupervised Video Summarization Using Stepwise, Label-based Generative Adversarial Learning

Several methods have been proposed to automate video summarization, and the researchers' focus was recently attracted by deep learning architectures. In this direction, annotated datasets were built to facilitate training and evaluation. However, since video summarization is a highly-subjective task, we argue that supervised learning approaches, which rely on the use of a single ground-truth summary, cannot fully explore the learning potential of such architectures. Hence, we focused on developing an unsupervised learning-based method for video summarization.

The starting point of our work is the unsupervised method of [43]. This algorithm selects the video key-frames by minimizing the distance between the deep feature representations of the original video and a reconstructed version of it based on the selected key-frames. For this, a deep representation of the entire video frame sequence is created with the help of a bi-directional LSTM, which assigns a weight to each frame, and a variational auto-encoder (VAE). The former is used to capture the long-term dependencies over sequences of frames in both forward and backward direction. The latter is used to reveal the underlying structure of the frame/key-frame features (in its encoding part) and produce another representation of the video by drawing samples from the computed latent space (in its decoding part). The difficulty in defining a suitable threshold regarding the similarity between the reconstructed and the original video, directed Mahasseni et al. to the generative adversarial framework and the integration of a trainable discriminator network. The ultimate goal of this approach was to jointly train the frame selector and the variational auto-encoder in order to maximally confuse the discriminator, i.e. decrease discriminator's confidence in distinguishing the original from a reconstructed video, a condition that denotes a highly representative key-frame collection.

Building on this method, we scrutinized features of the architecture and the training process that could be fine-tuned to improve the model's performance. For this, we were based on a publicly available PyTorch implementation of a variation of this architecture [8], that was used for evaluating the performance of SUM-GAN on the summarization of 360° videos (see [37]). This variation contains a linear compression layer right before the frame selection component of the architecture. In the updated model (see Fig. 7), given a video of T frames and focusing on the t^{th} frame of this video, x_t represents the CNN feature vector, x'_t denotes the compressed feature vector, s_t refers to the computed importance score from the frame selector, w_t corresponds to the weighted feature vector ($s_t \otimes x'_t$), and $\hat{x}_t$ relates to the reconstructed feature vector by the variational auto-encoder.

In addition to the added linear layer, this variation follows a 3-step incremental training approach that updates specific parts of the network in each step. In particular, differently to the immediate update of the entire model based on the computed losses after a single forward pass of the architecture (see Alg. 1 in [43]), the implemented process:

- performs a 1^{st} forward pass over the entire model, computes the $L_{reconst}$, L_{prior} and

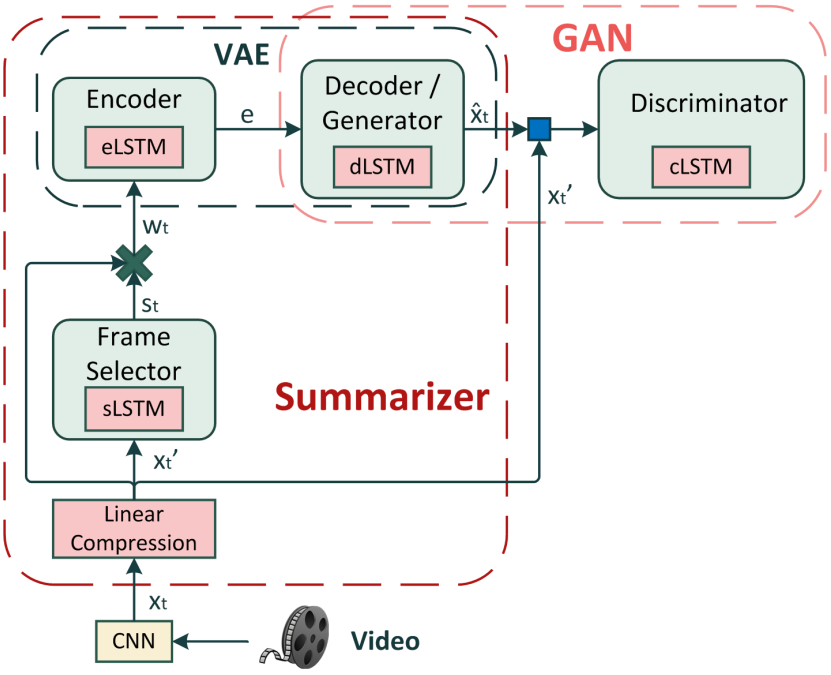

Figure 7: The developed SUM-GAN-sl architecture.

$L_{sparsity}$ losses, and updates only the frame selector, the encoder and the linear compression layer during the 1^{st} backward pass (top part of Fig. 8);

- performs a 2^{nd} forward pass of the partially updated model, computes the $L_{reconst}$ and L_{GAN} losses, and updates only the decoder and the linear compression layer during the 2^{nd} backward pass (middle part of Fig. 8);
- performs a 3^{rd} forward pass of the updated model, computes the L_{GAN} loss, and updates the linear compression layer and the discriminator during the 3^{rd} backward pass (bottom part of Fig. 8);

The aforementioned losses are computed similarly to [43]:

$$L_{reconst} = \left\| \varphi(\mathbf{x'}) - \varphi(\hat{\mathbf{x}}) \right\|^2 \tag{1}$$

where $\varphi(\mathbf{x'})$ is the output of the last hidden layer of cLSTM for compressed feature vectors of the original video ($\mathbf{x'} = \{x'_t\}_{t=1}^{T}$) and $\varphi(\hat{\mathbf{x}})$ is the output of the last hidden layer of cLSTM for the feature vectors of the summary-based reconstructed video ($\hat{\mathbf{x}} = \{\hat{x}_t\}_{t=1}^{T}$).

$$L_{prior} = D_{KL}(q(\mathbf{e}|\mathbf{x})||p(\mathbf{e})) \tag{2}$$

where $p(\mathbf{e})$ is a prior over the unobserved latent variable, $\mathbf{x}$ is the observed data, $q(\mathbf{e}|\mathbf{x})$ is the probability of observing $\mathbf{e}$ given $\mathbf{x}$, and D_{KL} denotes the Kullback-Leibler divergence. For efficient training we employ the re-parameterization trick proposed in [35].

$$L_{sparsity} = \left\| \frac{1}{T} \sum_{t=1}^{T} s_t - \sigma \right\|^2 \tag{3}$$

where T is the total number of video frames and σ is the regularization factor, a tunable hyper-parameter of the model.

$$L_{GAN} = log(p(\mathbf{x'})) + log(1 - p(\hat{\mathbf{x}})) + log(1 - p(\hat{\mathbf{x}}_\mathbf{p})) \tag{4}$$

where $p(\mathbf{x'})$, $p(\hat{\mathbf{x}})$ and $p(\hat{\mathbf{x}}_\mathbf{p})$ are probability scores (computed at the soft-max output of the discriminator) representing the discriminator's confidence when classifying the original video, the generated summary and the uniform summary respectively.

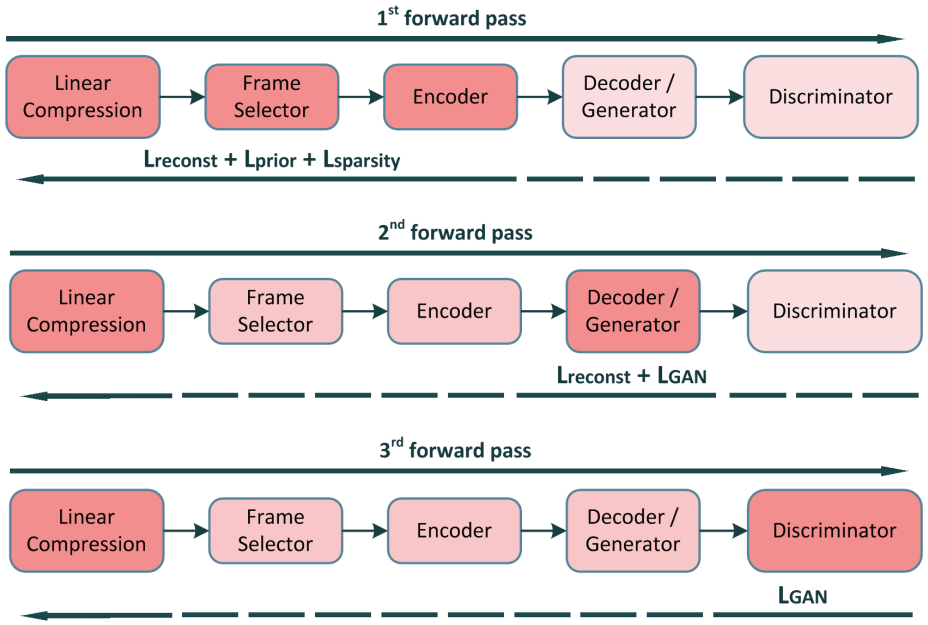

Figure 8: The different parts of the architecture are trained through a 3-step, incremental procedure that updates specific components of the model in each step. Updated components during a backward pass are indicated by dark-coloured boxes and solid line in the backpropagation arrow.

Given the above, we examined a different training strategy for the adversarial part of the model. The introduced learning approach was utilized in [52] for unsupervised representation learning with deep convolutional GANs, a method used for image generation. Driven by the effectiveness of this approach on training a network to generate realistic images from white noise, we transfer this methodology in our context. Our aim is to find a better equilibrium point between the generator and the discriminator, which means a better reconstruction of the video from the combination of the weighted frames and the learned distribution of data by the variational auto-encoder of the architecture. So, instead of using the L_{GAN} loss of the original SUM-GAN model, we follow a label-based approach, where label "1" is assigned to the original video and label "0" to the video summary. Given these labels, we introduce the following two losses:

$$L_{ORIG} = (1 - p(\mathbf{x'}))^2 \; and \; L_{SUM} = (p(\hat{\mathbf{x}}))^2 \tag{5}$$

The L_{ORIG} is used to minimize the Mean Squared Error (MSE) between the original video label and the computed probability when the discriminator is fed with the original video. Respectively, the L_{SUM} is used to minimize the MSE between the summary label and the computed probability when the discriminator is fed with the summary-based reconstruction of the video. Based on these losses, the training of the discriminator is performed in a stepwise manner, as depicted in Fig. 9 (top part). First, we pass the compressed feature vectors of the original video (x'_t, $t \in [1,T]$) through the discriminator (forward pass), calculate L_{ORIG} and then calculate the gradients (backward pass). Secondly, we pass the original video through the summarizer to create the reconstructed video ($\hat{x}_t$, $t \in [1,T]$), forward the latter to the discriminator, calculate L_{SUM} and then accumulate the gradients from both the original video

and the summary-based reconstructed one, with another backward pass. With the gradients accumulated, we call a step of the discriminator's optimizer. This incremental process enables a more fine-grained computation of the discriminator's gradients (compared with the training policy used in SUM-GAN), and helps the discriminator develop higher discrimination efficiency, thus performing better during the classification.

For training the generator, we introduce the following loss:

$$L_{GEN} = (1 - p(\hat{\mathbf{x}}))^2 \tag{6}$$

The L_{GEN} is used to minimize the MSE between the original video label and the computed probability when the discriminator is fed with the summary-based reconstruction of the video. By constantly trying to reduce the sum of $L_{reconst}$ and L_{GEN}, the generator aims to confuse the discriminator and make the summary-based reconstruction of the video indistinguishable from the original one.

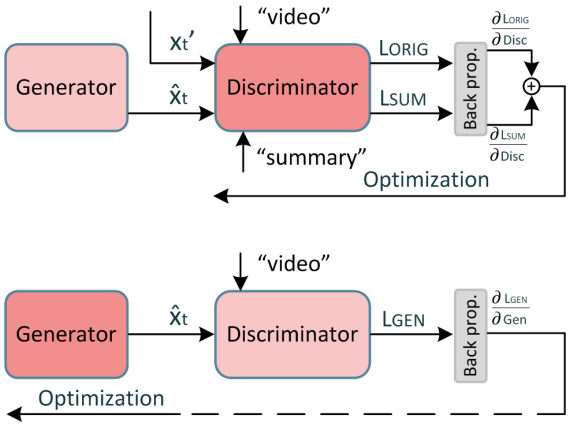

Figure 9: The stepwise, label-based training of the adversarial component of our model. Top part corresponds to the Discriminator and bottom part to the Generator.

Given the above described training strategy, the randomly generated summary used in the original SUM-GAN model to regularize learning of the discriminator is not needed any more in our variation. The authors of [43] claim that the use of the randomly generated summary enhances the discriminator's ability to distinguish between the original video and a summary-based reconstruction of it. Nevertheless, through this approach the discriminator learns to classify the random summary in the same class with the generated summary, thus restricting the discriminator's ability to make the distinction between an actual video summary and a randomly generated one. Based on this reasoning, we omit the use of a random summary for training our model.

Given a trained model, the components responsible for generating a summary for an unseen video are the linear compression layer and the frame selector. In particular, the CNN features of the video frames pass through the aforementioned components and an importance score is computed for each frame. Based on these scores, the key-fragments of the video are selected via the following procedure: the video is segmented using the KTS algorithm [51] (other approaches for shot or subshot segmentation, e.g. [2] and [3], could be used too); then, fragment-level importance scores are calculated by averaging the importance scores of each fragment's frames; and finally, the summary is generated by selecting the fragments that maximize the total importance score provided that the length of the summary does not exceed

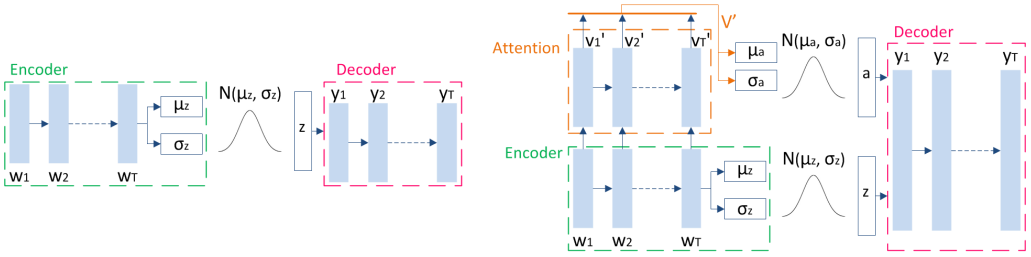

Figure 10: Going from the typical variational auto-encoder (left side) to the variational attention auto-encoder (right side).

15% of the original video duration. The latter requirement is adopted by several video summarization approaches (e.g. [30, 55, 66, 75]) and met by solving the following optimization problem:

$$max \sum_{i=1}^{N} a_i \cdot b_i, \; s.t. \sum_{i=1}^{N} a_i \cdot l_i \leq 0.15 \cdot L, \; a_i \in 0,1 \tag{7}$$

where N is the number of fragments, L is the length of the original video, 0.15 defines the upper limit for the summary duration, and given the $i-th$ fragment of the video, a_i is a binary value that indicates whether the fragment is selected or not, b_i is the computed fragment-level importance score, and l_i is the length of the fragment. The latter is the $0/1$ Knapsack problem.

4.4.2. Unsupervised Video Summarization Using Attention-driven Generative Adversarial Learning

The idea behind the use of an attention mechanism for video summarization is to mimic the way humans select the most representative pieces of a data sequence through a gradual decision-making approach that bases the selection of a piece of data on the previously seen ones. Inspired by [30], we examined two alternatives for integrating an attention mechanism into the SUM-GAN-sl model.

The first alternative involved the direct insertion of this mechanism within the variational auto-encoder of the architecture. A recent work (see [5]) that aimed to build a method for natural language modeling investigated different settings for this integration and described the bypassing effect that the traditional (deterministic) attention mechanism has on the VAE's functionality, since the latter has no impact in the process. To avoid this effect the authors of [5] proposed a variational attention mechanism where the attention vector is also modeled as Gaussian distributed random variables. Hence, the typical VAE is extended as shown in Fig. 10, in order to compute a latent variable also for the attention vector and use it when generating the decoded representation of the input sequence. Based on the above, we extended the SUM-GAN-sl model with variational attention, forming the SUM-GAN-VAAE architecture. In particular, the attention weights of each frame were considered as random variables and a latent space was computed by the variational auto-encoder for these values, too. Finally, the decoding part of this component was modified in order to update its hidden states based on both latent spaces (computed for the encoder's output and the attention values) during the reconstruction of the video.

The second alternative examined for integrating an attention mechanism to the SUM-GAN-sl model is based on the supervised attention-based encoder-decoder architecture of [30]. Since

deterministic attention bypasses the functionality of the VAE, the latter is no longer needed, and thus, it is entirely replaced by an attention auto-encoder (AAE) network. The overall architecture of the new model (called SUM-GAN-AAE) is presented in Fig. 11. Assuming again a video of T frames and based on the notations from Section 4.4.1, s_t, $t \in [1,T]$ refers to the computed importance score by the frame selector, w_t, $t \in [1,T]$ corresponds to the weighted feature vectors ($s_t \otimes x'_t$, where $\otimes$ denotes element-wise matrix multiplication) that are passed to the encoding part of the attention auto-encoder, and $\hat{x}_t$, $t \in [1,T]$ represents the feature vectors of the reconstructed video.

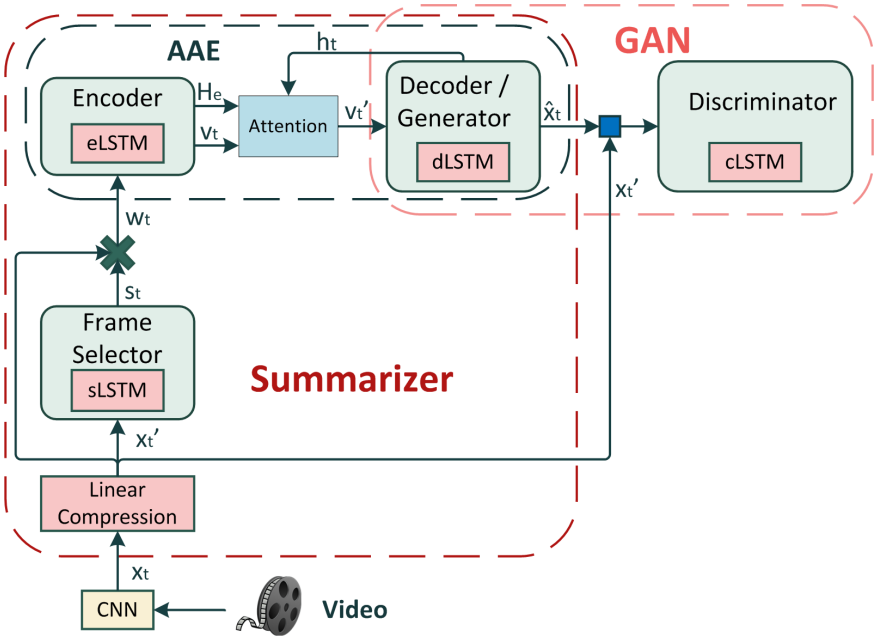

Figure 11: The developed SUM-GAN-AAE architecture.

Focusing on the introduced AAE module of the architecture (see Fig. 12), after feeding the weighted feature vectors to the encoder, the attention component receives the encoder output $\mathbf{V} = \{\nu_t, t \in [1,T]\}$ and the previous hidden state of the decoder h_t, then computes the attention weights e_t using a score function and finally applies a soft-max function to normalize them (a_t). In the first step of the decoding process, the attention component uses the hidden state of the last encoder's step ($\mathbf{H_e}$) instead of the previous hidden state of the decoder. Afterwards, the a_t weights are multiplied (the multiplication is denoted by "MM" in Fig. 12) with the encoder's output, producing the context vectors ν'_t, $t \in [1,T]$. The latter are fed to the decoder, which combines them with its output from the previous frame y_{t-1}, so as to reconstruct the initial video. The score function used in our implementation is a multiplicative one:

$$e_t^i = \nu_i^* W_a h_{t-1} \tag{8}$$

where ν_i^* is the transposed encoder output for the $i-th$ video frame, h_{t-1} is the hidden state of the decoder for $t-1$, W_a is a learnable parameter and e_t^i is the relevance score before the normalization. The final attention weights a_t^i are computed based on the following normalization:

$$a_t^i = \frac{\exp(e_t^i)}{\sum_{j=1}^{n} \exp(e_t^j)} \tag{9}$$

Our work on learning-based summarization has been submitted for publication to scientific

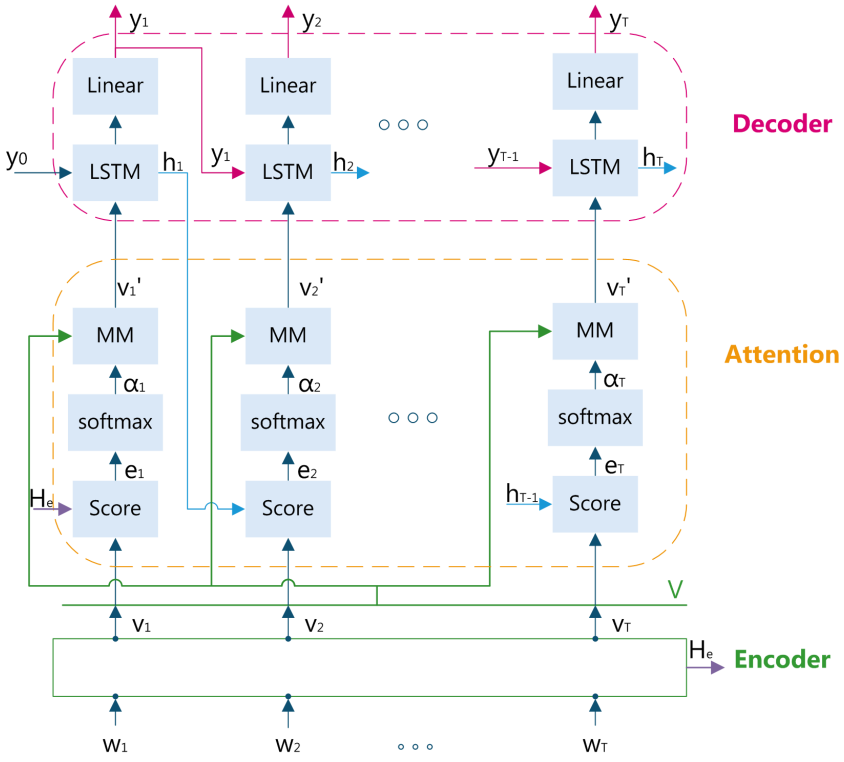

Figure 12: The attention auto-encoder. Decoding is performed in a stepwise manner which involves the corresponding step of the attention component.

conferences.

4.5. Text to Video Matching

For the purposes of text to video matching, we built a fully automatic dual encoding deep neural network architecture. Following the method that has been proposed in [13], we implemented an architecture that learns to represent a textual instance (e.g. a sentence) and a visual instance (i.e. a video key-frame) into a common feature space. Therefore, the correlation between a given text s_i and an image Im_j is directly comparable in the common space.

The Ad-hoc Video Search (AVS) problem is formulated as follows: given a text query Q and a set of keyframes $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{N}$ extracted from a video collection $\mathbf{V} = \{\mathbf{v}_j\}_{j=1}^{M}$, where a number of keyframes $\mathbf{x}_i \in \mathbb{R}^d$ has been extracted from each shot of the videos in the collection. Our goal is to retrieve for query Q the k keyframes from $\mathbf{X}$ that are most closely related to it.

We built a deep neural network that projects a natural language textual sentence s and a video keyframe $\mathbf{x}_i$ into the common feature space $\phi(x) \in \mathbb{R}^d$. For that the s is encoded into three different components: i) bag-of-words, ii) word2vec [46] and, iii) GRU-based sequential modeling. The bow representation is expressed as $\mathbf{V}^s_{bow} = [v_{w_1}, v_{w_2}, \ldots, v_{w_c}]$, where v_{w_i} is the occurrences of the word w_i in s and c in the size of the specific vocabulary. Word2Vec represents every word w_i of a phrase or other piece of text as a continuous vector $\mathbf{V}^{w_i}_{word2vec} = [v_1, v_2, \ldots, v_n]$ in a low dimensional space. Finally, the GRU-based $\mathbf{V}_{GRU}$ representation inputs the v_t word embedding vector at the time t and the previous hidden state h_{t-1} and outputs the updated h_t. Then, the three different representations are concatenated

$\mathbf{V}(s) = [\mathbf{V}_{bow}; \mathbf{V}_{word2vec}; \mathbf{V}_{GRU}]$ and the new representation is forwarded in a sequence of k fully connected layers $fc_i(s) = \sigma(W_i fc_{i-1}(s) + b1)$ where $fc_1(s) = \sigma(W_1 ms(s) + b1)$. The k layer outputs the final representation $\phi(s)$.

We use a pre-trained DCNN model to extract features for every keyframe, specifically, the Resnet-152 trained at the ImageNet 11k dataset. Each keyframe is represented by the flattened output of the *pool5* layer. Then, similarly to sentence representation, each $\mathbf{I}$ is forwarded into a fully connected layer $fc_1(x) = \sigma(W_I \mathbf{I}(x) + b_I)$. The output of the $fc_1(x)$ is at the common space $\phi(x)$.

Given a sentence representation $\phi(s)$ and a visual representation $\phi(x)$ we use the Mean Squared Error (MSE) as the objective function in order to train our framework into the common space. The MSE loss achieves better performance in contrast to other cost functions (i.e. marginal ranking loss)[13]. The framework is trained to minimize the overall MSE loss is defined as:

$$\underset{\theta}{\operatorname{argmin}} \sum_{(s,x)\in D} \ell_{mse}(s, x; \theta)$$

, where $\ell_{mse}(s, x; \theta) = (\phi(s) - \phi(x))^2$ for a sentence-image pair (s, x), D the set of all available pairs (s, x) and θ stands for all trainable parameters.

To develop the AVS framework we used the Keras[2] open-source neural-network library with the TensorFlow backend[3]. The Root Mean Square Propagation (RMSProp) optimizer was used for training in mini-batches of 128 sentences-images pairs. RMSprop is a fast and very popular optimizer that is suitable for large datasets and mini-batch learning. Gradient explosion is a well known problem in recurrent networks and the gradients could became NaN due to numerical overflow. To avoid gradient explosion, gradients were clipped by their l_2 norm.

4.6. Results

4.6.1. Video Summarization

Experimental Setting

The performance of the developed learning-based video summarization approaches is evaluated on the SumMe [26] and TVSum [55] datasets. SumMe includes 25 videos of 1 to 6 min. duration, covering multiple events from both first-person and third-person view. The authors of the SumMe dataset selected a team of $15-18$ anonymous users as an annotators team. Each video has been annotated by this team in the form of key-fragments, and thus is associated to multiple fragment-level user summaries. Moreover, a single ground-truth summary in the form of frame-level importance scores (calculated by averaging the key-fragment user summaries per frame) is also provided. TVSum contains 50 videos of 1 to 5 min. duration, capturing 10 categories of the TRECVid Multimedia Event Detection dataset. The authors of the TVSum dataset selected a team of 20 anonymous users as an annotators team. Each video has been annotated by this team in the form of frame-level importance scores, while once again, a single ground-truth summary in the form of frame-level importance scores (computed after averaging all users' scores) is available.

[2] `https://keras.io/`
[3] `https://www.tensorflow.org/`

A large portion of the videos in SumMe and TVSum datasets are comprised of a single shot. During preliminary attempts on evaluating the early non-learning-based versions of our video summarization approaches we observed the very poor performance, by means of F-Score in the specific dataset. This is due to two factors: (i) these non-learning-based video summarization approaches are shot-based, thus they fail to analyse single-shot video content, and (ii) these early versions where developed strictly abiding to the defined ReTV requirements. We argue that they should not be evaluated in the context of a benchmark procedure designed for the video summarization research field. Therefore Section 4.6.1 solely deals with the evaluation of the developed learning-based video summarization approaches.

Regarding our evaluation approach and for fair comparison with other approaches, we adopt the key-fragment-based evaluation protocol from [66] (adopted by the majority of SoA approaches; see Tables 6 and 7). The similarity between an automatically generated (A) and a user summary (U) is computed by the F-Score (as percentage), where (P)recision and (R)ecall measure the temporal overlap ($\cap$) between the summaries ($||*||$ denotes duration):

$$F = 2 \times \frac{P \times R}{P + R} \times 100, \ with \ P = \frac{A \cap U}{||A||} \ and \ R = \frac{A \cap U}{||U||} \tag{10}$$

So, given a video, we compare the automatically generated summary with the available user summaries for this video, and compute an F-Score for each pair of generated and user summary. Then, we average the computed F-Scores (for TVSum) or keep the maximum of them (for SumMe, following [25]) and end up with the final F-Score for this video. The computed F-Scores for the entire set of testing videos are finally averaged to capture the algorithm's performance. This protocol is directly applicable on SumMe, as user annotations are already available in the form of key-fragments. For TVSum, frame-level annotations are converted to key-fragment annotations following [55, 66]. The videos are segmented using the KTS method [51], and fragment-level importance scores are computed by averaging the scores of each fragment's frames. Video fragments are ranked based on the computed scores and the Knapsack algorithm is used to select the key-fragments and form the summary, such that it does not exceed 15% of the video duration.

Last but not least, for fair comparison with a group of methods ([43, 30, 60, 69, 63]) that follow a different evaluation protocol, which involves the comparison of the generated video summary only with the single ground-truth summary for that video, we report our models' performance based on this approach too.

Videos were downsampled to 2 fps. For fair comparison with other works, feature extraction was based on the pool5 layer of GoogleNet [56] trained on ImageNet. The linear compression layer reduces the size of these vectors from 1024 to 500. Each component of the architecture is comprised of 2-layer LSTM, with 500 hidden units, while as in [43] the frame selector is a bi-directional LSTM. Training is based on the Adam optimizer and the learning rate for all components but the discriminator is 10^{-4}; for the latter one is 10^{-5}. Finally, we followed the standard 5-fold cross validation approach and we report the average performance over the 5 runs.

Preliminary Study on Datasets

Aiming to get some insights about the used datasets, we examined the following aspects:

- the efficiency of a randomly generated summary (frames' importance scores defined based on a uniform distribution of probabilities and the experiment was performed 100 times);

- the human performance, i.e. how well a human annotator would perform based on the preferences of the remaining annotators; this is a metric regarding the compatibility/agreement between the defined human summaries;
- an estimate about the highest performance on TVSum [4] according to the best human-generated summary (with the highest overlap) for each video of the dataset.

For completeness, in Table 4 we report the outcomes of our study using both criteria for calculating the video-level F-Scores, i.e. the maximum of the computed F-Scores in the case of SumMe, and the average of these scores in the case of TVSum. The results - which are consistent with the findings of a recently published study on these datasets [47] - clearly indicate that video summarization is a highly subjective task, as there is no ideal summary that exhibits significant overlap with all annotators' preferences, in both datasets. Moreover, the "average" metric in the case of TVSum shows that human performance is comparable with the efficiency of a randomly generated summary, and thus limits the available space for improvement. In particular, the best possible summary (i.e. a summary that matches the best human-generated summary for each different video of the dataset) results in a score that is approximately 10 units higher than the score of a random summary. Given the reasonable lack of an objective summary for a video, we argue that the "max" criterion is more suitable for assessing the performance of video summarization approaches. In this sense, the upper-bound with respect to video summarization efficiency will be 100% in both datasets, denoting that machine-generated summaries are indistinguishable from human-generated ones.

Table 4: Findings on the performance (F-Score (%)) of different types of summaries and the theoretical upper-bound of the SumMe and TVSum dataset, based on the "average" and "max" criterion.

	SumMe		TVSum	
	Average	Max	Average	Max
Random	18.1	39.9	53.9	75.5
Human Summaries	31.3	55.1	53.8	77.5
Best Possible	44.7	100.0	64.7	100.0

Evaluation Outcomes

The developed models were initially evaluated for several values of the regularization factor σ, ranging between 0.05 and 0.5. Greater values were not examined as the models' performance was significantly reduced in (at least) one of the datasets for the highest tested value. In Table 5 we report our findings focusing on the SUM-GAN-AAE model, which was proved to be the most effective one. As can be seen, this factor affects the model's efficiency (as reported in [43]) and thus, it needs fine-tuning. Moreover, the latter seems to be dataset-dependent, as the highest performance is achieved for different values of σ in each dataset. For fair comparison with other approaches that rely on a strictly defined set of (hyper-)parameters, in the following we refer to the SUM-GAN-AAE model with $\sigma = 0.05$, since the gain compared to the model's performance in SumMe for $\sigma = 0.3$, is higher than the observed mitigation in TVSum for $\sigma = 0.1$. Similarly, the best performing SUM-GAN-sl and SUM-GAN-VAAE models were observed for $\sigma = 0.05$ and $\sigma = 0.3$ respectively.

[4] Based on the "max" criterion, the upper-bound for SumMe is 100%, i.e. the generated summary perfectly matches with a human-generated summary.

Table 5: Performance (F-Score (%)) of SUM-GAN-AAE for different values of the regularization factor. Best performance is shown in bold.

	SumMe	TVSum
$\sigma = 0.05$	**48.8**	58.4
$\sigma = 0.1$	46.4	58.6
$\sigma = 0.15$	47.4	58.8
$\sigma = 0.3$	47.1	57.9
$\sigma = 0.5$	44.7	**59.2**

The results of the comparative evaluation of these models against the performance of a randomly generated summary [5] and of other state-of-the-art unsupervised approaches on SumMe and TVSum are reported in Table 6 [6]. The original SUM-GAN method is not listed in this table as it follows a different evaluation protocol, and the comparison with it is reported in the sequel (see Tables 8 and 9). These results show that: (i) the performance of a few SoA methods is comparable (or even worse) than that of a random summary generator; (ii) the best approach on TVSum (Tessellation) achieves random-level performance on SumMe, a fact that indicates it is a dataset-tailored technique; (iii) the best technique on SumMe (UnpairedVSN) performs slightly better than SUM-GAN-sl but not better than SUM-GAN-AAE, while it is clearly less competitive on TVSum; (iv) the introduction of variational attention reduces the efficiency of the SUM-GAN-sl model, possibly due to the difficulty in efficiently defining two latent spaces in parallel to the continuous update of the model's components during the training; (v) the replacement of the VAE with the AAE results in a noticeable performance improvement over the SUM-GAN-sl model. The latter indicates the contribution of the introduced attention mechanism in enhancing the decoder's ability to identify the most important frames to pay attention to, and in effectively guiding the learning of the adversarial component of the architecture. The applied training strategy efficiently backpropagates this knowledge to the frame selection component, resulting in a significantly improved performance compared to the SUM-GAN-sl model. On top of these findings, the SUM-GAN-AAE model performs consistently well in both datasets (being the best one on SumMe), and thus is the most competitive one among the compared approaches.

Table 6: Comparison (F-Score (%)) with different unsupervised video summarization approaches, on SumMe and TVSum. $+/-$ indicate better/worse performance compared to SUM-GAN-AAE.

	SumMe	TVSum
Random summary	39.9 $(-)$	53.9 $(-)$
Tessellation [32]	41.4 $(-)$	**64.1** $(+)$
DR-DSN [75]	41.4 $(-)$	57.6 $(-)$
Online Motion-AE [70]	37.7 $(-)$	51.5 $(-)$
UnpairedVSN [53]	47.5 $(-)$	55.6 $(-)$
SUM-GAN-sl	47.3 $(-)$	58.0 $(-)$
SUM-GAN-VAAE	45.7 $(-)$	57.6 $(-)$
SUM-GAN-AAE	**48.8**	58.4

In addition, a study of the training curves of the models' components (see Fig. 13) points out that the AAE contributes to much faster and more stable training of the model. In

[5] Importance scores were defined based on a uniform distribution of possibilities and the experiment was repeated 100 times.

[6] The scores for each method are from the corresponding paper.

particular, with respect to the generative adversarial part of the architecture, the L_{ORIG} and L_{SUM} losses which are responsible for training the discriminator (top part of Fig. 13) and the L_{GEN} loss which contributes to the training of the decoder (bottom left part of Fig. 13) are converging much earlier for the SUM-GAN-AAE model (compared to SUM-GAN-sl), while this convergence is kept stable for the remaining training period. This impact is much more pronounced after focusing on the training loss of the frame selector and the encoder of the developed models, where a rapid reduction of this loss is exhibited once again for the very early training epochs in the case of SUM-GAN-AAE. Finally, the faster training achieved by introducing the AAE component is demonstrated also after comparing the learning curves of the SUM-GAN-sl and SUM-GAN-AAE models. As presented in Fig. 14, both models start from approx. the performance of a randomly-generated summary and develop knowledge about the task (the fluctuation is reasonable due to the adversarial nature of the training), which results in a noticeable improvement of their summarization efficiency. Nevertheless, the training in the case of SUM-GAN-AAE is much faster, as the model reaches its peak performance before the 30^{th} training epoch, while the highest performance for the SUM-GAN-sl model is observed after the 90^{th} training epoch. A similar training efficiency was exhibited in the case of the TVSum dataset.

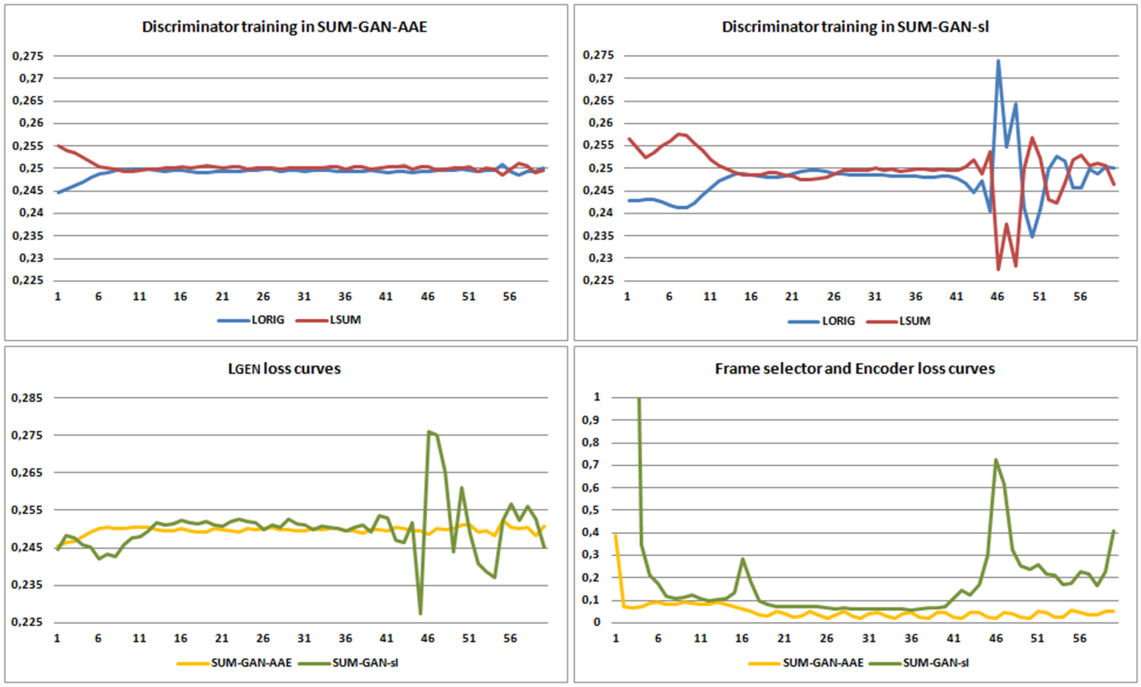

Figure 13: Loss curves of the discriminator (top), generator (bottom left), frame selector and encoder (bottom right) for the SUM-GAN-AAE and SUM-GAN-sl models. Horizontal axis denotes training epochs.

Our unsupervised SUM-GAN-sl and SUM-GAN-AAE models were compared against the performance of supervised approaches for video summarization (which is a comparison that is rather unfair to the developed unsupervised models). From the data presented in Table 7, it seems that: (i) the two best methods in TVSum (MAVS and Tessellationsup respectively) are highly-adapted to this dataset, as they exhibit random-level performance on SumMe; (ii) only a few supervised methods clearly surpass the performance of a randomly-generated summary on both datasets, with VASNet being the best among them. The performance of the latter methods ranges from 44.1 to 49.7 in SumMe, and from 56.1 to 61.4 on TVSum. Hence, the

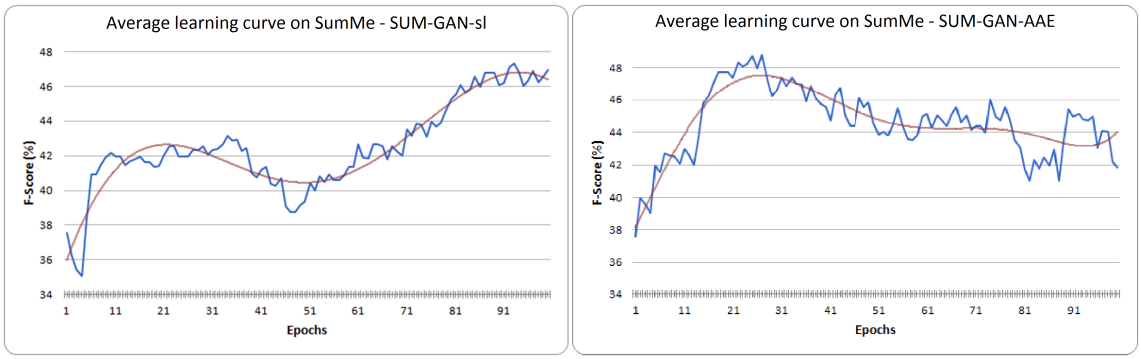

Figure 14: In blue, the average (over 5 splits) learning curves of the proposed models on SumMe. In red, the computed 6-order polynomial that approximates the learning curves.

performance of our SUM-GAN-sl (47.3 on SumMe and 58.0 on TVSum) and SUM-GAN-AAE (48.8 on SumMe and 58.4 on TVSum) models makes our unsupervised methods comparable with state-of-the-art supervised techniques for video summarization.

Table 7: Comparison (F-Score (%)) of our *unsupervised* methods with *supervised* video summarization approaches on SumMe and TVSum. $+/-/=$ indicate better/worse/equal performance compared to SUM-GAN-AAE.

	SumMe	TVSum		SumMe	TVSum
Random summary	39.9 $(-)$	53.9 $(-)$	MAVS [20]	40.3 $(-)$	**66.8** $(+)$
vsLSTM [66]	37.6 $(-)$	54.2 $(-)$	SUM-FCN [54]	47.5 $(-)$	56.8 $(-)$
dppLSTM [66]	38.6 $(-)$	54.7 $(-)$	SUM-DeepLab [54]	48.8 $(=)$	58.4 $(=)$
H-RNN [72] [7]	41.1 $(-)$	57.7 $(-)$	DR-DSNsup [75]	42.1 $(-)$	58.1 $(-)$
Tessellationsup [32]	37.2 $(-)$	63.4 $(+)$	ActionRanking [16]	40.1 $(-)$	56.3 $(-)$
HSA-RNN [73]	44.1 $(-)$	59.8 $(+)$	UnpairedVSNpsup [53]	48.0 $(-)$	56.1 $(-)$
DQSN [76]	-	58.6 $(+)$	VASNet [19]	**49.7** $(+)$	61.4 $(+)$
DSSE [64]	-	57.0 $(-)$	SUM-GAN-sl	47.3 $(-)$	58.0 $(-)$
			SUM-GAN-AAE	48.8	58.4

Finally, for fair comparison with approaches evaluated using the single ground-truth summaries of each video of SumMe and TVSum (i.e. the different evaluation protocol adopted in [22, 30, 43, 60, 69, 63]), we assessed our models via this approach as well. Once again, we considered different values for the regularization factor σ, to examine its impact on the models' efficiency according to this evaluation protocol and make our findings comparable with the ones in [43]. The results reported in Table 8 indicate that the methods' performance is, indeed, affected by the value of σ, while the effect of this hyper-parameter depends on the evaluation approach (best performance when using multiple human summaries was observed for $\sigma = 0.1$ on SUM-GAN-sl and for $\sigma = 0.05$ on SUM-GAN-AAE). Moreover, our methods clearly outperform the original SUM-GAN model on both datasets, even for the same value of σ. Finally, the comparison of the best performing instance of our model (for $\sigma = 0.5$) with other techniques that follow this evaluation protocol, indicates the superiority of the proposed approach in both datasets (see Table 9 [6]).

Given the findings reported in this section, the SUM-GAN-AAE model is qualified as the proposed method for video summarization in ReTV, at this stage of the project (month M20).

[7] Performance reported in a subsequent work of the authors (see [73]).

Table 8: Comparison (F-Score (%)) of the best performing SUM-GAN model (based on the score reported in [43]) with the performance of SUM-GAN-sl and SUM-GAN-AAE for different values of the regularization factor σ. Best performance for each developed model is shown in bold.

	σ	SumMe	TVSum
SUM-GAN	0.3	38.7	50.8
SUM-GAN-sl	0.1	38.1	61.0
	0.3	45.2	62.4
	0.5	**46.8**	**65.3**
SUM-GAN-AAE	0.1	45.9	**65.0**
	0.3	46.6	64.6
	0.5	**47.7**	64.3

Table 9: Comparison (F-Score (%)) of video summarization approaches on SumMe and TVSum, using a single ground-truth summary for each video. Unsupervised methods marked with asterisk. $+/-$ indicate better/worse performance compared to SUM-GAN-AAE.

	SumMe	TVSum
* SUM-GAN [43]	38.7 (−)	50.8 (−)
* SUM-GANdpp [43]	39.1 (−)	51.7 (−)
SUM-GANsup [43]	41.7 (−)	56.3 (−)
SASUM [60]	45.3 (−)	58.2 (−)
DTR-GAN [69]	44.6 (−)	59.1 (−)
A-AVS [30]	43.9 (−)	59.4 (−)
M-AVS [30]	44.4 (−)	61.0 (−)
AALVS [22]	46.2 (−)	63.6 (−)
* Cycle-SUM [63]	41.9 (−)	57.6 (−)
* SUM-GAN-sl	46.8 (−)	**65.3** (+)
* SUM-GAN-AAE	**47.7**	64.3

It should also be noted that the time required for generating a summary using this method ranges from 5% to 25% of the video's duration, on a PC with a TitanXP GPU. These times include the necessary keyframe extraction and processing step; thus, the SUM-GAN-AAE method is quite fast. When it comes to summarizing a previously un-processed video (thus, one for which the WP1 Video Analysis service has not already been run) this method is therefore at least 4 times faster than the implementation of the non-learning-based method presented earlier in this section, since the latter requires getting the video through the WP1 Video Analysis (VA) service first; the VA service takes roughly as much time as the video's duration for processing it (Arguably, this is also implementation-related; one could strip down the VA service to extract only the features needed for summarization, and this could speed it up by a factor of 2, i.e. requiring about 50% of the video's duration for it to run; but even so, for processing an un-seen video the SUM-GAN-AAE method would still be at least 2 times faster).

4.6.2. Text to Video Matching

Experimental Setting

To train the AVS framework we combined two different datasets, the TGIF [39], and the MSR-VTT [61]. The TGIF dataset contains approximately 100k short animated GIFs with one short

description for every GIF. We extract the keyframes by sampling each video every 0.5 seconds, obtaining from this procedure 820.000 frames. The MSR-VTT dataset consists of 10.000 short video clips, and each video is followed by 20 short descriptions. We extract keyframes as in the case of the TGIF dataset obtaining 239.537 keyframes.

Our experiments were performed on the TRECVID AVS 2016 (AVS16) [4] dataset that consists of approximately 600 hours of internet archive videos and is evaluated on 30 queries. Automated shot boundaries are provided by TRECVID. Ground-truth annotated training data does not exist for these queries. We analyze our results in terms of mean extended inferred average precision (MXinfAP), which is an approximation of the mean average precision suitable for the partial ground-truth that accompanies the TRECVID dataset [62].

Evaluation Outcomes

In Table 10 the results of the developed method for ad-hoc video search at the AVS16 dataset is presented. Two different variations of applying this method on the AVS16 dataset are evaluated. In the $\text{AVS16}_{\text{original}}$ one frame per shot is sampled, resulting in 335.944 frames, while in the $\text{AVS16}_{\text{dense}}$ we sample each shot with up to 15 frames per shot resulting in 1.231.143 frames. As expected the implemented method performs better when more than one keyframes are examined for each shot, i.e., in the $\text{AVS16}_{\text{dense}}$ setup. In this deliverable, we presented an early version of the text-to-video matching system which is meant to be used as a baseline for future improvements and upgrades. Future improvements will include examining deeper neural network architectures, improved loss functions as well as enhanced textual and visual representations.

Table 10: Evaluation results of the AVS method.

Evaluation Dataset	MXinfAP
$\text{AVS16}_{\text{original}}$	4.21
$\text{AVS16}_{\text{dense}}$	**4.82**

4.7. Video Summarization Component, Workflow and API

The CERTH Video Summarization (VS) component is a REST service that:

- Retrieves the video file, the segmentation data and the features extracted from videos previously analyzed by the WP1 Video Analysis (VA) service. The necessary data are retrieved from the GENISTAT repository where the results of the WP1 VA service are stored.
- Summarizes the video, selecting appropriate segments until the target duration is reached, and produces a summarization script in JSON format.
- The summarization script is submitted back to the service (via a Render call) and the video summaries and mouse-over preview videos (in gif and mp4 formats) are rendered.

Alternatively, in order to support different scenario uses, and if the service is called including the {"end2end":1} flag in the HTTP post body, then:

- The video file is retrieved.

- The video is analyzed from scratch, by performing temporal segmentation and extracting features (invoking the WP1 VA service).
- Summarizes the video, selecting segments until the target duration is reached, and produces a summarization script in JSON format.
- The summary script as well as the video previews are rendered to video files and are available in the CERTH server for the user to download.

The component works in an asynchronous way, thus, to use the service, there are three types of call:

- Start call: an HTTP POST call (i.e. has a BODY) which submits the input video and initiates a new session.
- Status call: an HTTP GET call that queries the status of a session.
- Results calls: an HTTP GET call that retrieves various information about a successfully completed session.

Start Call

```
HTTP POST http://retv.iti.gr:8090/va
```

Obligatory JSON structured arguments in the POST call body (one of the following):

- "video_analysis_sessions": use a single (or multiple) video analysis session id(s) from which to fetch video analysis data for the summarization.
- "video_urls": use a single (or multiple) video URL(s) from a web page as input to the service. The video will be downloaded from the web page and analyzed[8]. This can also be a Google Drive link or any link that points to any downloadable video file. Use this argument only when you want to run an end-2-end session (i.e., include the "end2end":1 string in your JSON body.

The URLS must be submitted as a JSON structured list (even if submitting a single video analysis session id or a single video URL, for example: {"video_analysis_sessions":["1234567890123456"]}).

Optional JSON structured arguments in the POST call body (can be optionally utilized and used in any order):

- "end2end" (int, [0,1], default=0): The usual call of this service takes as input the features already extracted from the video analysis service and produces a summary script. Setting this to 1, will force the service to perform video analysis, from scratch and then, create the summarization script and render the summaries and previews.
- "target_duration" (float, default=30.0): The target duration of the summary, in seconds. At the current implementation, the resulting summary duration may diverge from the set target duration ± 2 seconds. Additionally, the set target duration cannot be more than $1/2$ of the original video duration, e.g., for a 1-minute video you cannot request

[8] For a complete list of supported sites, see `https://ytdl-org.github.io/youtube-dl/supportedsites.html`

a summary of more than 30 seconds. If the target duration does not abide by this rule then it will be automatically adjusted by the service.

- "white_concepts" (string, default is an empty string): Use this for concept-based summarization, to provide a list of desired concept ids. The algorithm favours the selection of segments that contain one or more of the listed concepts. The syntax is that of a comma-separated list of items. Each item contains a pool-id string and a concept-id integer separated by the ":" character. For example, if you want to use segments that contain fire trucks and fire stations, you can search the concept pools list (available as shared documents in the project's repository) locate the Fire_Truck concept in the SIN concept pool and fire_station concept in the places concept pool, and then call the summarization service by including the following argument: "white_concepts":"sin:115,places:145".
- "black_concepts" (string, default is an empty string): Use this for concept-based summarization, to provide a list a concept ids. The algorithm avoid the selection of segments that contain the listed concepts. The syntax is the same as in the case of "white_concepts" parameter.
- "rhythm" (float, [0.25, 2.0], default=1.0): Controls the rhythm of segments in the summary ("snappiness" of the final summary). In essence this is a float which is multiplied with the maximum allowed duration of a shot segment that is selected to be included in the summary. Setting this to 0.1 will produce a summary with very fast alternating segments (some segments might be less than a second). Setting this to 2.0 will produce a summary where some segments might be quite long (up to 10 seconds).
- "auto_render" (int, [0,1], default=0): There are cases (e.g., 4u2 use-case) in which the summary script is not needed. Setting this parameter to 1, will force the service to render the previews and the summary directly. You can then use a GET results call to retrieve these from this session (no need to issue a separate render call). The summary script will still be available in this session, so you can still retrieve it, modify it and submit it to a different render session to produce a different summary. Note that end2end parameter regards the video analysis. If you want to perform video analysis, summarization and rendering in a single session, you should specify "end2end":1 AND "auto_render":1.
- "preview_target_duration" (float, [3.0,8.0], default=5.0): Target duration of the preview, in seconds.
- "preview_static_frame_duration" (float, [0.5,3.0], default=1.0): Duration of each selected key-frame in the static preview, in seconds.
- "utilize_story_percent" (float, [0,1], default=1.0): This parameter allows the exclusion of the last parts of the original video from consideration when creating the summary. Most times a "spoiler" might reside in the final shots of a video. This parameter allows the user to exclude such segments. For example setting "Utilize_story_percent"=0.75 will make sure that no shot from the last quarter of the video will be used.

The start POST call returns a JSON file. If the call is successful, the JSON file contains the following fields:

- "message": "The call has been received"
- "session": <session>

The <session> is a unique id of the call used later to get status or results.

If the call is NOT successful, the JSON file contains the following field with a self-explanatory message. A list of the messages received during a failed start POST call is the following:

```
'Invalid white_concepts syntax'
'Invalid black_concepts syntax'
'Unknown pool "XXXX"'
'Invalid index "XXXX" for pool "YYYY"'
'Invalid end2end syntax. Value must be 0 or 1'
'Cannot set both video_analysis_sessions and end2end'
'Cannot set end2end=1 without submitting video_urls'
'Either video_analysis_sessions or end2end has to be set'
```

Render Call

```
HTTP POST http://retv.iti.gr:8090/render
```

As a body in the HTTP post call, use the whole (possibly modified) JSON summary script that the summarization service produces. This will render the summaries and the previews in the current session folder. To retrieve these see the Results Call paragraph.

Status Call

```
HTTP GET http://retv.iti.gr:8090/<session>/status
```

where <session> is the unique id received when calling the service.

The status call returns a JSON file. If the call is successful, the JSON file contains the fields "status" and "message". If the message field is "The status you requested does not exist", please check that you provided a valid session ID. The status field will contain various messages throughout the procedure of the summarization. If a message containing the word "FAILED" has been received then there was an error during the summarization process. If the "VIDEO SUMMARIZATION COMPLETED" message has been received, you can proceed to make the results GET calls.

Results Calls

The various GET calls to retrieve information about a completed session, are the following:

- ```
  http://retv.iti.gr:8090/<session>/summary/script
  ```

  Retrieves the summarization script as a JSON file (ss.json). This JSON also contains some information about the video analysis session (where the original video can be found) and the links to all constructed previews.

- ```
  http://retv.iti.gr:8090/<session>/summary/mp4
  ```

Retrieves the video summary as an mp4 video file (summary.mp4). Note that for this call to succeed you must either have submitted a summarization script or have submitted the summarization script creation using the "end2end":1 .

- http://retv.iti.gr:8090/<session>/preview/gif

  Retrieves the constructed video preview in gif format (preview.gif).

- http://retv.iti.gr:8090/<session>/preview/mp4

  Retrieves the constructed video preview in mp4 format (preview.mp4).

- http://retv.iti.gr:8090/<session>/preview/static

  Retrieves the constructed video static preview in gif format (static.gif).

- http://retv.iti.gr:8090/<session>/thumbnail

  Retrieves the video's most characteristic keyframe (thumbnail) as a jpg file (summary_-thumbnail.jpg). Note that this must be used with a render <session>.

- http://retv.iti.gr:8090/<session>/vs_time

  Retrieves the duration of the summarization process in seconds as well as a ratio to the original video duration.

- http://retv.iti.gr:8090/<session>/rs_time

  Retrieves the duration of the summarization process in seconds as well as a ratio to the original video duration.

- http://retv.iti.gr:8090/<session>/log

  Retrieves a compact log of the summarization process (for debugging and error tracking purposes).

- http://retv.iti.gr:8090/<session>/full_log

  Retrieves the full log of the summarization process (for debugging and error tracking purposes).

All sessions that are older than 48 hours are automatically deleted. After that time, any status or results GET calls for these sessions will return the message "The status/results you requested does/do not exist".

Component Testing and Software Quality Assessment

The VS REST service and the summarization method that is implemented in the service have undergone extensive testing, both by their developers and by other ReTV partners that use the service for generating video summaries. The generated summaries have been assessed by users in ReTV, both within the consortium (as discussed in Section 4.3) and as part of the user testing performed in relation to the use cases of the project, and the preliminary results are very encouraging. Concerning unit testing, similarly to the VA service of WP1, several hundred such tests were conducted to cover all the different possible call configurations of the service (given the number of either mandatory or optional parameters of the REST service API, as detailed earlier in this section, and also cover edge cases (such as very short or very long videos submitted for summarization). The proper exchange of information with the WP1 VA service, for the types of calls of the VS service that require the previous correct execution of the VA one, was also tested. In terms of stress testing, the VS REST service implements a queuing mechanism similar to that of the VA service; this mechanism was also tested with a few hundred summarization requests that were submitted and were successfully queued and eventually processed. The VS service is mature for use in ReTV; however, given its planned transition to using a learning-based summarization method in the near future, further testing and software quality assessment of it will need to be performed to assess the impact of the method updates to the service. Such testing will continue, as updates are introduced, throughout the life of WP3.

5. Content Recommendation and Scheduling

5.1. Content Recommendation Service Overview

A classic video recommendation service takes user preferences as input and returns content that best matches those preferences. Our content adaptation service not only recommends videos but also automatically adapts them. We developed a concise domain-specific language based on JSON to describe adaptations. We currently support two types of adaptations: Shortening and Merging.

Shortening describes the act of trimming the start and end of a video segment. We use this functionality in the Content sWitch scenario in order to make the replacement trailer fit into the slot of the original trailer.

Merging is the process of combining multiple videos into a single one, which is useful if more than one show covers a topic that is of interest to a user. Our content adaptation service then selects multiple videos that fit user preferences. It then passes those videos to the dedicated summarisation service described in Section 4.

We use a Field-aware Factorisation Machines (FFM) model [31] to find candidate videos for a given viewer. The advantage of FFM compared to other methods is that it allows us to exploit a large number of features efficiently. This extendability is a valuable property, as we expand the number of content features we extract from the video content. It also models the interactions between the individual feature values which allows us to discover viewing patterns. For instance, young men are interested in sports content, and fans of tennis are also interested in golf but not swimming.

Currently, our content features revolve around genres. The source of such features can be an EPG provider; however, we can categorise any content on our own. Our approach is an unsupervised categorisation, based on speech-to-text transcripts and multilingual word embeddings [11][9]. The topics taxonomy we use is a subset of EBUCore[10]. We categorise each word in a sentence independently, and the final categorisation is a majority voting result. Note that a single program can belong to multiple categories with different weights. The automatic approach has an advantage over standard EPG as it can categorise individual parts of a program. This is particularly important for thematically heterogeneous content, such as news shows where only a subset of the program might be of relevance to a specific viewer.

We are focusing on the two use cases, both described in detail in D6.2, namely:

1. 4u2 Chatbot, providing user with an up-to-date video content reflecting her explicitly declared interests
2. Content sWitch, replacing in-stream video trailers with the trailers best suited for the user behavioural profile (implicit interests)

While the resulting videos for the Use Case 1 and Use Case 2 are vastly different, we see this as the application of the same two-step process:

1. Find the best videos for a user under a set of business constraints.
2. Adapt those videos to fit the context.

[9] https://github.com/facebookresearch/MUSE
[10] https://tech.ebu.ch/MetadataEbuCore

Sample business constraints that we can encode are a maximum and minimum length. More advanced business constraints could be to bias the recommendation towards an editorial focus.

We can train our recommendation model on both types of input: implicit user profile, i.e. behavioural data, and explicitly provided user preferences.

In the next iteration, we plan to extend this system to support richer automatic content adaption.

5.2. Content Scheduling Service Overview

The content scheduling service is a part of the Content Wizard, described in detail in D5.2, section 4.2. Currently, the workflow for the manual scheduling of posts is implemented. Social media posts can be scheduled to be posted to multiple publication vectors at once, for example to Twitter and Facebook. Automatic scheduling is not yet integrated but will be in the final version of the Content Wizard. We describe our approach to the automatic scheduling below.

We plan to use a model developed in T2.4 (cf. D2.2). This is the forecasting model based on ensembles of decision trees (random forests) that in addition to the standard seasonality features also includes content-related features as well as event-related features. The latter features allow for modelling irregularities in the future audience numbers that can be attributed either to the special event (usually sports) or to the features of the content. We can use such a prediction model to find an optimal publication time for some TV content (on the linear channel, e.g. Web stream). In order to achieve such an optimization, we score a given piece of content – and its corresponding content- and event-based features – for all publication time candidates and take the one that maximizes the audience. It is worth to mention that other targets are also possible, e.g. we can optimize not for the total/general audience, but for some segment of audience, e.g. people from a given location.

After achieving promising results with the content- and event-feature augmented forecasting models, we plan to focus on the automatic publication time optimization in the near future.

5.3. Audience Profiling for the Content Recommendation

Another feature for audience profiling is to segment the audience by viewing preferences [45]. Viewing preferences can be learnt directly from past audience data, i.e. preferences about what channels are watched on what day at what time. Assuming the preferences of the audience remains fundamentally the same, future audience interests can be predicted. We already noted that the type of content in the TV programming can also be a feature for a learning model, so that preferences represent what content is preferred by the audience. An advantage of the content-based audience profiling is that the preferences can be learnt across all TV channels rather than assuming every TV channel would have its own, individual and entirely separate viewing patterns. In other words, we can consider the recommendation task to determine the likely percentage of the total audience (the sum of all individual viewers in our audience data) to watch a piece of TV content on a given channel at a given time.

We benefit from having access to data about viewers and their viewing sessions from Zattoo, anonymized and provided in a pre-aggregated form, so that we can not reconstruct a single viewers TV viewing. In our case, we want to forecast the total audience for a piece of TV content as the aggregation of audience segments learnt by a recommendation model from the past audience data which would watch that content. The intention is not to segment viewers using sets of multiple, individual categorical interests (e.g. soccer fan, murder mystery fan etc.) which could be determined by an explicit collection of interests (e.g. as part of a viewer's profile) as this could lead to wildly simplistic matching (e.g. TV program has category Sport, so is watched by all Sports fans). Rather, our model should determine implicitly the likelihood of each viewer to watch a particular category of TV content. This means every viewer implicitly belongs to some audience segment, i.e. a group of viewers who share a similar likelihood to watch TV across each category. We expect this to allow us to capture more complex relationships between viewing patterns such as there are Sports fans who are just as likely to watch News whereas others would only watch Sports. Now for a future broadcast time point, we can determine if a viewer is likely to be watching TV at that time and if so, given the content across channels, which content they are most likely to be watching. Finally, we use the implicit audience segments (fuzzy clusters learned by the model) to place different viewer groups on different channels, according to their interests.

We experimented with two modeling approaches:

1. Baseline model: standard collaborative filtering based on Non-Negative Matrix Factorization (NNMF) [28]. This model does not use any additional content- or event-related features. It just observes the interactions between users and content (TV programs).
2. Field-aware Factorization Machines (FFM) model [31]. This model is as an extension of the basic factorization model, and it allows us to test the additional features (content- and event-based). Most importantly, it models the interactions between the individual feature values as a dot-product of the associated weighted vectors.

Collaborative filtering is traditionally used in recommendation. Indeed, our starting point for using these approaches was to build a model of viewer preferences for content recommendation. For a given set of TV content options, we wanted to recommend which TV content the viewer is most likely to watch. We have developed two TV content recommendation scenarios:

1. Content sWitch: we replace a "general audience" program trailer in the TV stream with a trailer personalized to the user's interests. The replacement is done in real-time in the IP stream and takes into account the lengths of the original and replaced trailer. Personalized content needs to be adapted to the original trailer duration, either by cutting it or through more advanced content summarization. In the first version we only use replacement trailers that are a couple of seconds too long at most, and then naively cut off the end. This works well in practice, as most trailers end with content that is not crucial to understand the complete message. In future versions we will use the content adaptation service to shorten the trailers in a way that respects the overall story line.
2. 4u2 Chatbot: within a preferred messaging app (e.g. Telegram or Whatsapp) the user subscribes to a set of content categories. Personalised video summaries (e.g. snippets of last nights' programming) are sent to the user on a regular basis.

In the first scenario, we track audience behavior and train the model that learns the interaction patterns between users (and their associated attributes) and individual content pieces (and their associated features, such as category). In the second scenario, we only have a general, explicitly

provided list of user categories, so the input information is less detailed and static (unless user modifies his or her profile). This allows us to compare recommendation in these two contexts - one where the user can be identified by an anonymous log-in, the other where the user is not identifiable across sessions and we can only use the explicitly provided information.

The recommendation model is however also a prediction model, since it learns for any choice of TV content the likelihood of that content being watched by any audience segment. Here, rather than having multiple TV content items and a single audience segment (that the target viewer of a recommendation belongs to), we would consider a single TV content item and calculate the likelihood of being watched across all audience segments.

It should be noted that the model is trained on very sparse data (since every user's viewing pattern covers only a small part of the total broadcast TV content) and it requires to fit a high number of parameters (each feature value, e.g. each user identifier, is associated with a vector of weights in a low-dimensional latent space). FFM models are also prone to overfitting and require (a) careful training with the evaluation and test datasets and optimization early-stopping if the train/evaluation metrics diverge, as well as (b) proper optimization of model hyperparameters. We applied the Bayesian hyperparameter optimization approach.

For the early-stopping, we used the options available in xLearn library that provides fast implementation of FFM models. We include a measurement of the strength of the interaction between the user and the content (i.e. our target value to be modeled). However, explicit feedback from the user regarding how satisfied/engaged he or she is with a given program is missing from our viewing data. Therefore we based our model on the fraction of the program that the user watched. The assumption is that the more of the program the user has watched, the more relevant it was for her or him. On the other end, zapping between programs generates low target values that are considered as (implicit) negative feedback. We do not consider total watching duration since this would introduce bias and promote some content categories (e.g. movies are usually much longer than TV series or news).

We trained the recommendation model with our categorized audience data and the implicit audience segments. There are two types of metrics that are involved in the recommendation model training:

1. Metrics that are optimized during model fitting phase;
2. Metrics that we use to evaluate when the model is good enough for our purposes.

For the model optimization, we used the standard metric provided by xLearn library[11], log-loss (equivalent to cross-entropy). It should be noted that our approach is based on providing a single content item recommendation to a given user. So the metric that optimizes only the top of the program ranking (instead of optimizing for all users data) is preferred here. In the future, we'd like to experiment with metrics such as WARP (cf. e.g. [12]).

For the model evaluation, presented below, we applied the three standard metrics:

- MAE (mean average error) between the observed targets and the model-recommended values,
- Pearson linear correlation between the observed targets and the model-recommended values

[11] https://github.com/aksnzhy/xlearn

[12] https://medium.com/@gabrieltseng/intro-to-warp-loss-automatic-differentiation-and-pytorch-b6aa5083187a

Table 11: Recommendation model with TV content categorization as additional feature compared to the benchmark

Model	MAE	Pearson	Spearman
NNMF	0.45	0.32	0.35
FFM (no EPG)	0.25	0.73	0.75
FFM (coarse EPG)	0.22	0.75	0.77
FFM (detailed EPG)	**0.18**	**0.79**	**0.8**

- Spearman rank correlations.

Especially the last metric is relevant for our scenarios, since we are not concerned with absolute values predicted by our model, but rather with having relevant content pieces at the top of the recommendations ranking.

Table 11 shows that, compared to the baseline model (NNMF), a feature-based model (FFM) could already establish significantly better results which were also moderately improved by the additional of coarse or detailed TV programming content categories.

We compared the baseline model (NNMF, not using any content or user features, beside their identifiers) with various variants of FFM models (using the attributes described in the previous section):

- NNMF model had MAE error 0.45 (the smaller the better) and rank correlation between recommended and actually watched programs only 0.35 (the larger the better)
- The best FFM model (with hyperparameters optimization applied) without additional attributes achieved MAE 0.25 and rank correlation 0.75
- The model with just one additional attribute (EPG category for the TV content) achieved MAE 0.18 and rank correlation 0.8. It was slightly worse in case of the less-detailed EPG metadata: MAE 0.22 and rank correlation 0.77. It shows the importance of providing the model with high-quality content metadata.

Interestingly, the model without additional attributes was also more prone to overfitting. It may be because the differences between training and evaluation datasets are driven by factors which are not explicitly observed in the data (i.e. content-related attributes). Models for the chatbot scenario (where the input is the set of interests explicitly provided by a user, instead of user behavioral data - detailed interactions with content - as in the case of Content sWitch) were - as expected - slightly worse than the model in the Content sWitch scenarios: MAE 0.23 (vs. 0.18 for Content sWitch) and rank correlation 0.72 (vs. 0.8 for the Content sWitch). Still, the results are much better than the baseline or the model without any additional attributes provided. In both use case scenarios, the model took advantage of the interactions between the content features and the (implicit or explicit) user interests.

Future work is to test the recommendation model for audience prediction, aggregating audience segments that are most likely to watch a piece of future TV content. Planned model improvements include:

- Using WARP instead of log-loss optimization - this will focus on the top of the recommendation ranking, instead of the complete ranking
- Testing if explicit audience segmentation improves the model (e.g. k-means clustering of viewers by watching preference) compared to the current, implicit fuzzy approach

- Including temporal features in the model (with the assumption that the current user session provides a better context for recommendations than previous sessions).

5.4. Future Work and Conclusions

The currently tested recommendation model uses the following attributes:

- user: identifier, behavioral profile (the percentage of the time spent on each individual EPG category, which can be also viewed as implicit fuzzy segmentation of users)
- program: identifier, main EPG category (e.g. sport), detailed EPG genre (e.g. discipline) (actual values depending on the EPG metadata provider)

The recommendation modeling could still be improved as results vary greatly between EPG categories. In general, the model works best for the popular categories such as sport, since we also have most training data for such categories. In parallel, we work on extending the additional attributes with events. As noted earlier (also cf. D2.2 deliverable), audience data contains anomalies which can be to a large extent attributed to events (sport events in particular) broadcast on TV. We will address this by explicitly adding event features into the model and are still learning how to model the events to represent their features for optimal recommendation, analogously to the approach that we took for the forecasting model (cf. D2.2). The big advantage of the FFM model is that it is able to model the interactions between the various feature values, so it automatically learns, e.g. that sport events mostly affect the behaviour of a sport-predisposed audience segment. Similar to users and content, we add an event identifier and set of event features (the more detailed the better, including temporal and geographical features of relevance). Later we are interested in also adding:

- behavioural viewership patterns (hours of the day, days of the week) in order to be able to find not only a proper content but also optimal engagement time, and
- more advanced content features such as face detection with Deep Neural Networks. It could help to fine-grain user preferences even more, capturing user interest in a given TV presenter or an actor.

In conclusion, we have learnt that in audience prediction we can improve forecasts (cf. D2.2) and recommendations by taking into account the category of the TV content. While we have seen in the data how specific events cause significant anomalies in audience interests trends, we are still learning how best to incorporate event knowledge into our recommendation model. The sparsity and irregularity of events as part of overall audience measurement is a limitation. We also can implicitly segment the (actual or predicted) audience and use this in TV content recommendation. We found that the TV program category and overall content popularity as learnt by the recommendation model is even more important than an individual user profile. This may be considered a positive aspect of the model, since for a new user it allows to partially alleviate the cold-start problem (i.e. to recommend generally popular content rather than a random one, and iteratively learn the user preferences). We are now testing the accuracy of this recommendation model in predicting future audiences by aggregating the audience segments likely to watch a piece of future TV content. In general, we have found that AI models with additional features do work better but in terms of feature selection, content-based features have proven more effective to date compared to audience-based and then to event-based. The predictive analytics will be used in the ReTV project to provide tools for media organisations to help them publish the right content on the right channel at the right time. Two scenarios

demonstrate how AI enabled audience prediction and profiling can power new innovative TV content recommendation services for TV viewers.

6. Conclusion and Outlook

In the context of viewer profiling, we investigated the implicit segmentation derived from the recommendation model and evaluated its relevance as a way to measure user interests in various content categories (either provided by EPG metadata or automatically learned by a model).

In terms of video adaptation and re-purposing, we presented in this deliverable three methods for video summarization; one non-learning-based that relies primarily on shot clustering, and two unsupervised learning-based methods that are realized by new deep learning architectures that we proposed. We also presented our early experiments on text-to-video matching. Concerning future work, we continue to work on improving the learning-based summarization paradigm by examining further modifications to the deep network architecture, as we consider this to be a very promising and advantageous approach to summarization. We will also investigate how we can combine the advantages of the learning-based and non-learning-based approaches by introducing explicit video editor's desires (e.g., no anchorperson shots in the summary) in the learning-based paradigm. On the software implementation side, these developments will also be reflected in the Video Summarization REST service, which will be updated accordingly. Furthermore, we continue to work on improving the results of text-to-video matching and we will employ it for the task of targeted advertisement placement in the video.

In the context of content recommendation, we researched on how the content and event-related features can be used to improve the quality of the standard, baseline models based on collaborative filtering (such as factorization machines). We found out that both types of features are very relevant, both in terms of recommendations, as well as the audience forecasting (cf. D2.2). Our next steps will focus on extending it even further with more fine-grained content-related features (derived from video annotation tools, cf. WP1) and more extensive set of event-based features.

In context of content scheduling, we have a manual scheduling implemented as the part of Content Wizard scenario (cf. D5.2, section 4.2). Currently we start to work on the model-based publication time optimization, based on the forecasting model developed in T2.4.

References

[1] J. Almeida, N. J. Leite, and R. d. S. Torres. Vison: Video summarization for online applications. *Pattern Recogn. Lett.*, 33(4):397–409, Mar. 2012.

[2] E. Apostolidis and V. Mezaris. Fast shot segmentation combining global and local visual descriptors. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6583–6587. IEEE, 2014.

[3] K. Apostolidis, E. Apostolidis, and V. Mezaris. A motion-driven approach for fine-grained temporal segmentation of user-generated videos. In K. Schoeffmann, T. H. Chalidabhongse, C. W. Ngo, S. Aramvith, N. E. O'Connor, Y.-S. Ho, M. Gabbouj, and A. Elgammal, editors, *MultiMedia Modeling*, pages 29–41, Cham, 2018. Springer International Publishing.

[4] G. Awad, J. Fiscus, and M. M. et al. Trecvid 2016: Evaluating video search, video event detection, localization, and hyperlinking. In *TRECVID 2016 Workshop*. NIST, USA, 2016.

[5] H. Bahuleyan, L. Mou, O. Vechtomova, and P. Poupart. Variational attention for sequence-to-sequence models. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1672–1682, Santa Fe, New Mexico, USA, Aug. 2018. Association for Computational Linguistics.

[6] J. Calic, D. P. Gibson, and N. W. Campbell. Efficient layout of comic-like video summaries. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(7):931–936, July 2007.

[7] P. P. K. Chan, H. Yu, W. W. Y. Ng, and D. S. Yeung. A novel method to reduce redundancy in adaptive threshold clustering key frame extraction systems. In *2011 International Conference on Machine Learning and Cybernetics*, volume 4, pages 1637–1642, July 2011.

[8] J. Cho. PyTorch Implementation of SUM-GAN from "Unsupervised Video Summarization with Adversarial LSTM Networks", 2017. (last accessed on July 10, 2019).

[9] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, Oct. 2014. Association for Computational Linguistics.

[10] W. Chu, Y. Song, and A. Jaimes. Video co-summarization: Video summarization by visual co-occurrence. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3584–3592, June 2015.

[11] A. Conneau, G. Lample, M. Ranzato, L. Denoyer, and H. Jégou. Word translation without parallel data. *CoRR*, abs/1710.04087, 2017.

[12] S. E. F. de Avila, A. P. B. a. Lopes, A. da Luz, Jr., and A. de Albuquerque Araújo. Vsumm: A mechanism designed to produce static video summaries and a novel evaluation method. *Pattern Recogn. Lett.*, 32(1):56–68, Jan. 2011.

[13] J. Dong, X. Li, and C. G. M. Snoek. Predicting visual features from text for image and video caption retrieval. *IEEE Transactions on Multimedia*, 20(12):3377–3388, Dec 2018.

[14] N. Ejaz, I. Mehmood, and S. W. Baik. Feature aggregation based visual attention model for video summarization. *Computers & Electrical Engineering*, 40(3):993 – 1005, 2014. Special Issue on Image and Video Processing.

[15] N. Ejaz, T. B. Tariq, and S. W. Baik. Adaptive key frame extraction for video summarization using an aggregation mechanism. *Journal of Visual Communication and Image Representation*, 23(7):1031 – 1040, 2012.

[16] M. Elfeki and A. Borji. Video summarization via actionness ranking. In *IEEE Winter Conference on Applications of Computer Vision (WACV), Waikoloa Village, HI, USA, January 7-11, 2019*, pages 754–763, Jan 2019.

[17] E. Elhamifar, G. Sapiro, and R. Vidal. See all by looking at a few: Sparse modeling for finding representative objects. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1600–1607, June 2012.

[18] F. Faghri, D. J. Fleet, J. R. Kiros, and S. Fidler. Vse++: Improving visual-semantic embeddings with hard negatives. 2018.

[19] J. Fajtl, H. S. Sokeh, V. Argyriou, D. Monekosso, and P. Remagnino. Summarizing videos with attention. In G. Carneiro and S. You, editors, *Computer Vision – ACCV 2018 Workshops*, pages 39–54, Cham, 2019. Springer International Publishing.

[20] L. Feng, Z. Li, Z. Kuang, and W. Zhang. Extractive video summarizer with memory augmented neural networks. In *Proceedings of the 26th ACM International Conference on Multimedia*, MM '18, pages 976–983, New York, NY, USA, 2018. ACM.

[21] B. J. Frey and D. Dueck. Clustering by passing messages between data points. *science*, 315(5814):972–976, 2007.

[22] T. Fu, S. Tai, and H. Chen. Attentive and adversarial learning for video summarization. In *IEEE Winter Conference on Applications of Computer Vision (WACV), Waikoloa Village, HI, USA, January 7-11, 2019*, pages 1579–1587, 2019.

[23] M. Furini, F. Geraci, M. Montangero, and M. Pellegrini. Stimo: Still and moving video storyboard for the web scenario. *Multimedia Tools Appl.*, 46(1):47–69, Jan. 2010.

[24] B. Gong, W.-L. Chao, K. Grauman, and F. Sha. Diverse sequential subset selection for supervised video summarization. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 2069–2077, Cambridge, MA, USA, 2014. MIT Press.

[25] M. Gygli, H. Grabner, and L. V. Gool. Video summarization by learning submodular mixtures of objectives. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3090–3098, June 2015.

[26] M. Gygli, H. Grabner, H. Riemenschneider, and L. Van Gool. Creating summaries from user videos. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 505–520, Cham, 2014. Springer International Publishing.

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

[28] P. O. Hoyer. Non-negative matrix factorization with sparseness constraints. 5(Nov):1457–1469, 2004.

[29] j. peng and q. xiaolin. Keyframe-based video summary using visual attention clues. *IEEE MultiMedia*, 17(2):64–73, April 2010.

[30] Z. Ji, K. Xiong, Y. Pang, and X. Li. Video summarization with attention-based encoder-decoder networks. *IEEE Transactions on Circuits and Systems for Video Technology*, pages 1–1, 2019.

[31] Y. Juan, Y. Zhuang, W. S. Chin, and C. J. Lin. Field-aware factorization machines for ctr prediction. pages 43–50, 2016.

[32] D. Kaufman, G. Levi, T. Hassner, and L. Wolf. Temporal tessellation: A unified approach for video analysis. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 94–104, Oct 2017.

[33] A. Khosla, R. Hamid, C. Lin, and N. Sundaresan. Large-scale video summarization using web-image priors. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2698–2705, June 2013.

[34] C. Kim and J.-N. Hwang. Object-based video abstraction for video surveillance systems. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(12):1128–1138, Dec 2002.

[35] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[36] J.-L. Lai and Y. Yi. Key frame extraction based on visual attention model. *Journal of Visual Communication and Image Representation*, 23(1):114 – 125, 2012.

[37] S. Lee, J. Sung, Y. Yu, and G. Kim. A Memory Network Approach for Story-based Temporal Summarization of 360 Videos. In *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[38] X. Li, B. Zhao, and X. Lu. A general framework for edited video and raw video summarization. *IEEE Transactions on Image Processing*, 26(8):3652–3664, Aug 2017.

[39] Y. Li, Y. Song, L. Cao, J. Tetreault, L. Goldberg, A. Jaimes, and J. Luo. TGIF: A New Dataset and Benchmark on Animated GIF Description. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[40] Y. Li, T. Zhang, and D. Tretter. An overview of video abstraction techniques. *Hewlett Packard, Technical Reports*, 01 2001.

[41] S. Lu, Z. Wang, T. Mei, G. Guan, and D. D. Feng. A bag-of-importance model with locality-constrained coding based feature learning lt;newline/ gt;for video summarization. *IEEE Transactions on Multimedia*, 16(6):1497–1509, Oct 2014.

[42] M. Ma, S. Mei, S. Wan, Z. Wang, and D. Feng. Video summarization via nonlinear sparse dictionary selection. *IEEE Access*, 7:11763–11774, 2019.

[43] B. Mahasseni, M. Lam, and S. Todorovic. Unsupervised video summarization with adversarial lstm networks. pages 2982–2991, 2017.

[44] F. Markatopoulou, D. Galanopoulos, V. Mezaris, and I. Patras. Query and keyframe representations for ad-hoc video search. pages 407–411, 06 2017.

[45] D. Meyer and R. J. Hyndman. The accuracy of television network rating forecasts: The effects of data aggregation and alternative models. 1(3):147–âĂŞ155, 2006.

[46] T. Mikolov, I. Sutskever, and K. Chen et al. Distributed representations of words and phrases and their compositionality. In *26th Int. Conf. on Neural Information Processing Systems*, NIPS'13, pages 3111–3119, USA, 2013. Curran Associates.

[47] M. Otani, Y. Nakahima, E. Rahtu, and J. Heikkilä. Rethinking the evaluation of video summaries. In *2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[48] M. Otani, Y. Nakashima, E. Rahtu, J. Heikkilä, and N. Yokoya. Video summarization using deep semantic features. In *The 13th Asian Conference on Computer Vision (ACCV'16)*, 2016.

[49] P. Over, J. Fiscus, G. Sanders, D. Joy, M. Michel, G. Awad, A. Smeaton, W. Kraaij, and G. Quénot. Trecvid 2014–an overview of the goals, tasks, data, evaluation mechanisms and metrics. 2014.

[50] R. Panda, A. Das, Z. Wu, J. Ernst, and A. K. Roy-Chowdhury. Weakly supervised summarization of web videos. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 3677–3686, Oct 2017.

[51] D. Potapov, M. Douze, Z. Harchaoui, and C. Schmid. Category-specific video summarization. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 540–555, Cham, 2014. Springer International Publishing.

[52] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *2016 International Conference on Learning Representations (ICLR)*, 2016.

[53] M. Rochan and Y. Wang. Video summarization by learning from unpaired data. In *2019 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[54] M. Rochan, L. Ye, and Y. Wang. Video summarization using fully convolutional sequence networks. In V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, editors, *Computer Vision – ECCV 2018*, pages 358–374, Cham, 2018. Springer International Publishing.

[55] Y. Song, J. Vallmitjana, A. Stent, and A. Jaimes. Tvsum: Summarizing web videos using titles. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5179–5187, June 2015.

[56] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, June 2015.

[57] B. T. Truong and S. Venkatesh. Video abstraction: A systematic review and classification. *ACM Trans. Multimedia Comput. Commun. Appl.*, 3(1), Feb. 2007.

[58] K. Wang, R. He, W. Wang, L. Wang, and T. Tan. Learning coupled feature spaces for cross-modal matching. In *2013 IEEE International Conference on Computer Vision*, pages 2088–2095, Dec 2013.

[59] T. Wang, T. Mei, X. Hua, X. Liu, and H. Zhou. Video collage: A novel presentation of video sequence. In *2007 IEEE International Conference on Multimedia and Expo*, pages 1479–1482, July 2007.

[60] H. Wei, B. Ni, Y. Yan, H. Yu, X. Yang, and C. Yao. Video summarization via semantic attended networks. In *2018 AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[61] J. Xu, T. Mei, T. Yao, and Y. Rui. Msr-vtt: A large video description dataset for bridging video and language. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[62] E. Yilmaz, E. Kanoulas, and J. A. Aslam. A simple and efficient sampling method for estimating ap and ndcg. In *31st ACM SIGIR Int. Conf. on Research and Development in Information Retrieval*, pages 603–610, USA, 2008. ACM.

[63] L. Yuan, F. E. H. Tay, P. Li, L. Zhou, and J. Feng. Cycle-SUM: Cycle-Consistent Adversarial LSTM Networks for Unsupervised Video Summarization. In *2019 AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

[64] Y. Yuan, T. Mei, P. Cui, and W. Zhu. Video summarization by learning deep side semantic embedding. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(1):226–237, Jan 2019.

[65] H. Zhang, J. Wu, D. Zhong, and S. W. Smoliar. An integrated system for content-based video retrieval and browsing. *Pattern Recognition*, 30:643–658, 1997.

[66] K. Zhang, W.-L. Chao, F. Sha, and K. Grauman. Video summarization with long short-term memory. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, *Computer Vision – ECCV 2016*, pages 766–782, Cham, 2016. Springer International Publishing.

[67] K. Zhang, K. Grauman, and F. Sha. Retrospective encoders for video summarization. In V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, editors, *Computer Vision – ECCV 2018*, pages 391–408, Cham, 2018. Springer International Publishing.

[68] X.-D. Zhang, T.-Y. Liu, K.-T. Lo, and J. Feng. Dynamic selection and effective compression of key frames for video abstraction. *Pattern Recogn. Lett.*, 24(9-10):1523–1532, June 2003.

[69] Y. Zhang, M. Kampffmeyer, X. Liang, D. Zhang, M. Tan, and E. P. Xing. Dtr-gan: Dilated temporal relational adversarial network for video summarization. *CoRR*, abs/1804.11228, 2018.

[70] Y. Zhang, X. Liang, D. Zhang, M. Tan, and E. P. Xing. Unsupervised object-level video summarization with online motion auto-encoder. *Pattern Recognition Letters*, 2018.

[71] Y. Zhang, R. Tao, and Y. Wang. Motion-state-adaptive video summarization via spatiotemporal analysis. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(6):1340–1352, June 2017.

[72] B. Zhao, X. Li, and X. Lu. Hierarchical recurrent neural network for video summarization. In *Proceedings of the 2017 ACM on Multimedia Conference*, MM '17, pages 863–871, New York, NY, USA, 2017. ACM.

[73] B. Zhao, X. Li, and X. Lu. Hsa-rnn: Hierarchical structure-adaptive rnn for video summarization. In *2018 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '18, 2018.

[74] B. Zhao and E. P. Xing. Quasi real-time summarization for consumer videos. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2513–2520, June 2014.

[75] K. Zhou and Y. Qiao. Deep reinforcement learning for unsupervised video summarization with diversity-representativeness reward. In *2018 AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[76] K. Zhou, T. Xiang, and A. Cavallaro. Video summarisation by classification with deep reinforcement learning. In *2018 British Machine Vision Conference (BMVC)*, 2018.