Applications of Convolutional Neural Networks on Multi-modal Video Summarization

Vasileios Savvas Papagrigoriou[1], Apostolos Dimoulakis[1]

[1] Department of Informatics, Aristotle University of Thessaloniki

Abstract :

High length videos can sometimes contain segments that lack contextual relevance. When a viewer intends to simply understand the overall context of a video that contains a lot of irrelevant video segments, this can result in a waste of resources and time for them. Video summarization is the task of reducing a video to its most important segments. In our work, we have performed an ablation study, by investigating and comparing 4 lightweight video summarization models. Our analysis focuses on simple convolutional neural networks capable of modeling the importance of individual video frames with a post-processing step that selects keyshots depending on these importance scores. To perform our study we have utilized supervised learning that was relied on a benchmark dataset. Although our models are outperformed by the state-of-the-art with a significant gap, our current work is a reliable starting point for the task of video summarization. Additionally we offer a clean code that can be further developed to reach higher standards and perform additional studies in the field.

Index Terms: Video, Audio, Image, Multimodal Data, Convolutional Neural Networks.

1 Introduction

Video summarization is a complex task involving multiple modalities. The task's objective is finding brief video segments (i.e. clips), that can effectively convey its meaning, without requiring the viewer to watch the entire video. Due to the large volume of online videos, there is a pressing need for automatic video summarization algorithms. Research in the field shows that deep neural networks have taken the spotlight in terms of their effectiveness on the task of automatic video summarization [1, 2]. Hence, motivated by that fact, we have invoked a set of simple neural networks and supervised learning, to perform an ablation study and investigate their performance on that task. For our study, we have selected 4 simple convolutional neural networks (CNNs), two of which utilize both audio and visual information, while the rest are based only on visual information. The selected trainable models are trained on a subset of the benchmark dataset. These models are limited in processing the video on the frame level, considering exactly one frame at the time and are responsible for producing a score that measures how significant a frame is in general (i.e. without considering the associated video's context). The inferred scores are then processed by an additional optimization algorithm that produces the keyframes which collectively make up the summary.

2 Related Work

There are various deep learning models that perform well in the task of video summarization. Many of these models are capable of capturing the temporal dependencies and are reasonably preferred over others. Examples of such models utilize CNNs to map the input frames a latent space, in conjunction with either LSTMs [3] or attention-based blocks [4, 2]. A 3D U-Net is utilized by [5] to encode spatio-temporal information which is then processed by an agent that acts by either accepting or rejecting the frames it encounters. Their method allows for the production of both supervised and unsupervised learning models. Other more direct approaches, consider video summarization as a sequence-to-sequence, supervised learning task and utilize encoder-decoder architectures, where the input is the raw video and the output is the summary [4]. Another proposed training process [6] of such encoder-decoder pipelines utilize a latent space to model video context. The latent space is used for the a measurement of the semantic discrepancy between the raw video with the predicted summary in that latent space, and by capturing this discrepancy, the model can consequently learn much more effectively. Additionally there are approaches that utilize the video's audio as well [1]. Our approach differs significantly from all these approaches as we limit the training to individual frames, without considering video context. However our approach offers the advantages of notably lower training time, reduced memory requirement and simplicity.

3 Methods

We have devised a simple pipeline that receives a video and produces a summary (see Fig. 1). The pipeline is composed of 3 main components:

- Pre-processing,
- Trainable Model,
- Post-processing

and their lower level components were primarily inspired by [2, 1]. Each of these components was carefully designed following simple video processing principles. We refer to the trainable model responsible for the generation of frame importance values as AVM.

3.1 Pre-processing

Suppose that the raw input video is composed of N_{raw} frames, each with shape $3 \times H_{\text{raw}} \times W_{\text{raw}}$. Each frame is downscaled to 40×40. Since a lot of the local images are identical, the video is sampled at 1 frame per second. All the pixels are subjected to min-max normalization (per frame). These transformations produce a refined pixel tensor $\mathcal{I}$ with shape $N \times 3 \times H \times W$ where $H = W = 40$. Also since the raw video's audio signal tends to be large ($\sim$ 22 kHz sampling rate), we have applied dimensionality reduction to that signal as well. The audio is mapped to Mel-Frequency Cepstral Coefficients (MFCCs), which are a more compact represen-

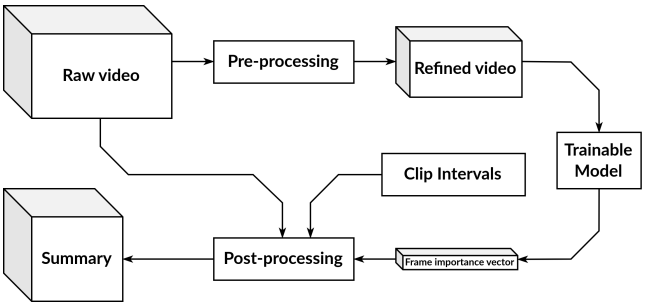

Figure 1. A high abstraction of the video summarization pipeline that we have invoked.

tation of the audio signal [7]. To describe the process, the first step is the segmentation of the audio samples. Each such segment is synchronized with the mentioned group of skipped frames (30 in total) iteratively. Hence for each such frame group's time-slot with index t, the corresponding audio signal is transformed into a separate MFCCs matrix $\mathcal{A}_t$. The matrices shape is $N_{\text{coef}} \times q$. Now since q depends on the transformation algorithm and on the input signal's size, we have invoked cubic interpolation to resize each column's length into $B := 30$ for uniformity.

3.2 Trainable Models

For the estimation of the importance score for each frame, we are based on a baseline neural network model we named as AVM. The importance score is a measure of semantic relevance of the corresponding frame with the video's context. The model (see Fig. 3) receives a frame along with the frame's corresponding MFCCs matrix and infers an importance score for that frame's time slot, specifically for the subsequent temporal indices $\{t, t+1, \ldots, t+29\}$. The model's input is initially split into two separate branches, namely VisBl and AudBl. Both of these branches are convolutional blocks, with VisBl being responsible for the refinement of visual descriptors while AudBl for the audio descriptors. VisBl is composed of 2D layers while AudBl of 1D layers (e.g. 1D convolutional layer). In AudBl the sliding windows overlap with the entire temporal axis and is performed as shown in the example of Fig. 2.

After the visual and audio descriptor vectors are computed, they are concatenated. Then the concatenated vector is being processed by a multi-layer perceptron block. Since the output's value set is an ordered set, it is most appropriate to handle this as a regression problem. Therefore we have decided to define the output layer as a fully connected layer with activation function $4 \cdot \sigma + 1$, where σ is the sigmoid function. The value set of this output neuron is $[1, 5]$. For our study we have used 4 variant model architectures of AVM specified by Table 1. From these models, VM and CVM do not include the AudBl block since they are not intended to process audio. Additionally CAVM and CVM are classifier models where each of the 5 output neurons is a probability for one of the 5 possible importance levels.

3.3 Post-processing

Since the refined video has a higher frame rate than that of the raw video's, the importance vector is expanded. To describe the expansion process, all importance values in the corresponding importance vector are replicated 29 additional times in order to

Table 1
Selected Configurations

Model Name	Loss Function	Audio Processing	Activation	Output size
AVM	MSE	True	$4 \cdot \sigma + 1$	1
VM	MSE	False	$4 \cdot \sigma + 1$	1
CAVM	Cat. Cross Entropy	True	Softmax	5
CVM	Cat. Cross Entropy	False	Softmax	5

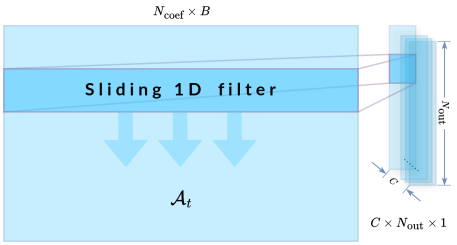

Figure 2. Demonstration of 1D convolution in AudBl, where the number of output channels is set to C. The left matrix is the MFCCs matrix received as an input, and the right tensor contains the resulting descriptors.

match the raw video's frame rate (see Fig. 4). The resulting vector matches the raw video's frame rate and thus its temporal length. We denote the expanded importance vector as v. Given the vector containing the frames' importance scores, we can now consider a segmentation of the video into clip intervals. We assume that these are all closed intervals. For each clip interval $I_j \subseteq [0, N_{\text{raw}} - 1]$ with interval index j, we assign a weight value defined as the length of the given interval

$$c_j := \max(I_j) - \min(I_j),$$

and an importance value which is the importance summation of all the contained frames

$$w_j := \sum_{t \in I_j \cap \mathbb{N}} v_t$$

The task is to find the optimal clips in order to produce the summarized video. The underlying process takes into account the clip intervals, their weights and their inferred importances. Specifically the task of video summarization is now reduced to the following Knapsack 0-1 [8] problem:

> Find a value $x_j \in \{0, 1\}$ for every interval with index j, that maximizes
>
> $$\sum_j (x_j \cdot c_j) \text{ satisfying } \sum_j (x_j \cdot w_j) \leq \lfloor r \cdot N_{\text{raw}} \rfloor$$

If x_j is set to 1, then the corresponding clip will be kept in the summary, otherwise it will not. A deterministic optimization algorithm is used to solve this problem. The resulting summary is the set containing the most relevant scenes from the given segmentation, without the total summary length surpassing some r fraction of the raw video's length. In our case we have set r as 15%. This approach was inherited by [2].

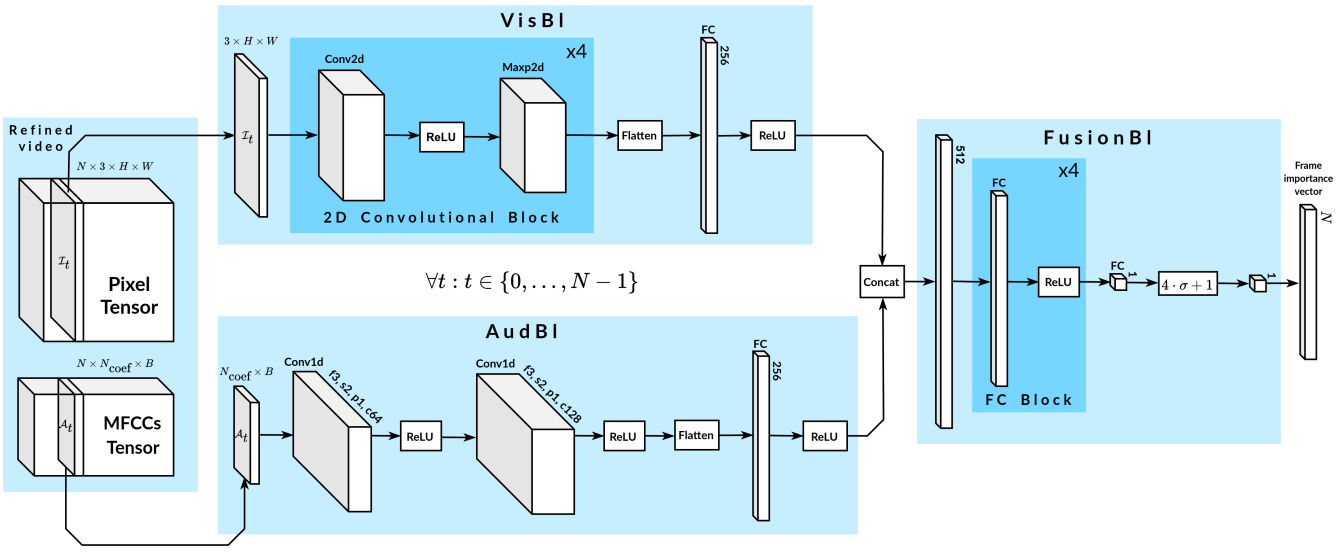

Figure 3. The pipeline of AVM. Conv2d and Conv1d are 2D and 1D convolutional layers respectively, where the integers right next to f, s, p and c are their window size, stride, padding and number of filters in that order. Maxp2d is a 2D max pooling layer and FC is a fully connected layer.

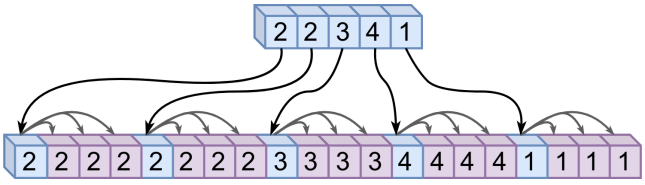

Figure 4. Diagrammatic depiction of how the frame importance vector is expanded. The upper vector is the importance vector generated by AVM for the refined video. The lower vector is the expanded importance vector, with length equal to that of the raw video. In this example the refined video has length 5 and the raw video has length 20.

4 Results

4.1 Dataset

Since we relied on supervised learning models, we have selected a labeled video dataset to train these models. The selected TVsum50 dataset [9] is composed of 50 YouTube [1] videos of varying resolution and frame rate. Each video frame is paired with 20 importance scores annotated by various workers from Amazon's Mechanical Turk [2]. These importance scores are integer numbers from 1 up to 5. This score is used as a measure of relevance with the corresponding video's context, hence the higher the importance score, the more relevant a frame is to the video. We have also included a set of clip intervals taken from [2]. The set of intervals is a segmentation of the dataset's video frames in the temporal axis of each included video. Such segmentations can be created using traditional video processing algorithms. TVSum50 serves as a popular benchmark dataset and is extensively utilized by researchers to compare their models and perform ablation studies. The models were trained based on one such video, with another one used as the test set. The IDs of these videos are:

- Train video: `37rzWOQsNIw`,
- Test video: `RBCABdttQmI`.

For each video we have 20 annotators, and to generate a single ground truth value for each frame, we have taken the average of these 20 importance scores for every frame.

4.2 Evaluation Metrics

To provide an intuitive measure for the performance of a given video summarization model, we have used F-score [2]. To define this metric for this task, suppose that P is the set containing all the predicted summary frames and G the set containing all the ground truth summary frames. Then to construct the F-score measure, we first define precision and recall as follows

$$\text{Prec} := \frac{|P \cap G|}{|P|}, \quad \text{Rec} := \frac{|P \cap G|}{|G|}.$$

Now can now define F-score as the latter quantities' harmonic mean, i.e.

$$\text{F-score} := 2 \cdot \frac{\text{Prec} \cdot \text{Rec}}{\text{Prec} + \text{Rec}}.$$

Each of these metrics are computed between the predicted frame-indices and each of the 20 annotators' summary, based on their importance vector and the mentioned post-processing component. After that we computed the average of F-scores (F-score Avg) and the maximum of these F-scores (F-score Max), to get single metric value for each instance's inferred summary.

4.3 Experiments

The experiments were carried out on a PC equipped with Intel i5-12500H processor and Nvidia RTX 4060M (Laptop) graphics card. The system also included 40GB of DDR4 memory and ran Ubuntu 22.04.1 operating system. All scripts were written in Python 3.10.12 in conjunction with PyTorch version 2.1.0.

For the training process, each model is iteratively optimized based on Adam [10]. For each training step, the model's trainable parameters are updated based on batches of 10 frames. To

[1] https://www.youtube.com
[2] https://www.mturk.com

Table 2
Experiments for the Ablation Study

Model Name	Train			Test			Optimal Epoch	Training Time (s)	Impr.
	Loss	F-score Avg	F-score Max	Loss	F-score Avg	F-score Max			
AVM	0.1768	0.3647	0.8125	0.3327	0.1488	0.2813	48	949.3	0.1667
VM	0.2057	0.3773	0.8689	0.3986	0.1723	0.4540	45	753.3	0.1793
CAVM	1.2254	0.2634	0.7082	1.1466	0.1418	0.4100	0	37.3	0.0221
CVM	1.6126	0.1979	0.6305	1.6121	0.1418	0.4100	`init`	24.7	0.0000

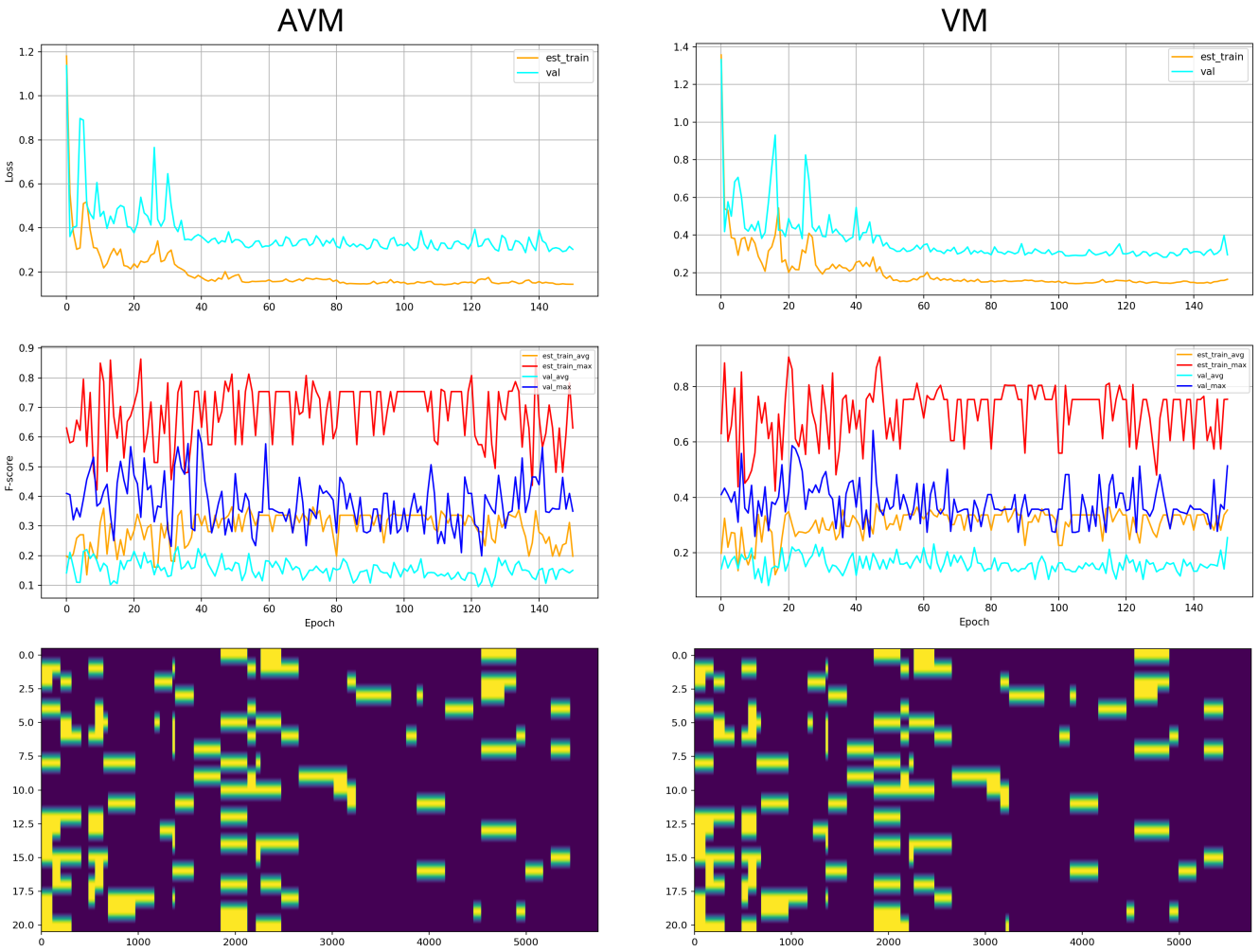

Figure 5. The first two rows show the performances of AVM and VM throughout their training. In the performance plots' legends, "train" means that the corresponding metric was computed on the train video, while "val" means that the corresponding metric was computed on the validation or test video. Also "max" and "avg" implies that the corresponding F-scores are F-score Max and F-score Avg respectively. Regarding the final row, it demonstrates the final model performance on the test video. In the latter diagrams, each of the first 20 rows correspond to an annotator's predicted summary, with the final row being the summary infered by the associated model, and the column axis being the frame index. In each case, the selected frame indices are yellow colored, while the dropped frames being purple.

compensate for any potential overfitting phenomena during training, we employ ensemble learning by including dropout layers directly after every fully connected layer with a drop rate of 20%. For the training of AVM and VM the loss function that is utilized is MSE, while CAVM and CVM are optimized based on the Categorical Cross Entropy (See Table 1). The evaluation scores, along with other training information of the mentioned models, can be observed in Table 2. For each training, we have selected a model's state (from some epoch) in which the F-score Avg was maximized on the train set. We consider these models to be the optimal ones. The last column of this table mentions the improvement of the optimal model compared to the initialized model in terms of the F-score Avg (train), for each architecture. Also when the optimal epoch is set to `init`, it means that the initialized model was the optimal one. We have also provided the performance plots for the trainings of AVM and VM along with a diagram of the final models' inference on the test video in Fig. 5.

5 Discussion

From Table 2, it can be observed that the classifier models CAVM and CVM have obviously failed to improve. In contrast, the regression models AVM and VM have been improved, and if we consider the loss diagrams of Fig 5, we can see that they gradually improved throughout the first 40 epochs. The discrepancy between the train and validation/test losses is an obvious sign of data shift because they have almost identical trends. Since the train metrics are all consistently worse on the test set compared to the training, we can accept that the model overfits. To compare AVM with VM, audio did not seem to contribute to performance.

6 Conclusion

Classifier CNNs do not seem to adapt well to the task of video summarization. Regression CNNs on the other hand indicate that they can perform better. We have concluded that the large performance gap between the state-of-the-art and our approach, is due to the fact that we do not model the video's context. However we also do not consider temporal dependencies, which may be another explanation for the performance gap. Our current work is a reliable starting point for our proposed models' improvement. Additionally we offer a clean code that can be further developed to reach higher standards and perform additional studies in the field.

7 Future Improvements

7.1 Replacement with Deeper Neural Networks

- **Temporal Dependencies**: Utilize deeper neural networks that incorporate temporal dependencies to infer an importance score. This can be achieved using:
  - **3D Convolutional Layers**: These layers can process temporal information across multiple frames, capturing motion and changes over time.
  - **Attention Mechanisms**: Attention layers can focus on important parts of the video sequence, improving the relevance of the selected frames.
  - **Recurrent Neural Networks (RNNs)**: Incorporate RNNs such as LSTM or GRU to capture sequential dependencies across frames.
- **Hybrid Models**: Develop hybrid models combining CNNs with RNNs or Transformers to leverage both spatial and temporal features effectively.

7.2 Context Encoders

- **Latent Space Context Encoding**: Encode the video's context in some latent space and consider the discrepancy of the the ground truth with the predicted summaries using a similarity function during training. By adding the measured discrepancy to the total loss function, we expect to yield improved results.
- **Title-Based Context Encoding**: Encode the video's context based solely on its title. This additional context can be integrated into the training process to improve the model's understanding of the video's theme and subject matter, leading to better summarization quality.

References

[1] Bin Zhao, Maoguo Gong, and Xuelong Li. *AudioVisual Video Summarization.* 2021. arXiv: 2105.07667 [cs.CV].

[2] Evlampios Apostolidis et al. "Summarizing Videos using Concentrated Attention and Considering the Uniqueness and Diversity of the Video Frames". In: ICMR '22. Newark, NJ, USA: Association for Computing Machinery, 2022, pp. 407–415. ISBN: 9781450392389. DOI: 10.1145/3512527.3531404. URL: https://doi.org/10.1145/3512527.3531404.

[3] Ke Zhang et al. "Video Summarization with Long Short-Term Memory". In: vol. 9911. Oct. 2016, pp. 766–782. ISBN: 978-3-319-46477-0. DOI: 10.1007/978-3-319-46478-7_47.

[4] Zhong Ji et al. *Video Summarization with Attention-Based Encoder-Decoder Networks.* 2018. arXiv: 1708.09545 [cs.CV].

[5] Tianrui Liu et al. "Video Summarization Through Reinforcement Learning With a 3D Spatio-Temporal U-Net". In: *IEEE Transactions on Image Processing* 31 (2022), pp. 1573–1586. ISSN: 1941-0042. DOI: 10.1109/tip.2022.3143699. URL: http://dx.doi.org/10.1109/TIP.2022.3143699.

[6] Ke Zhang, Kristen Grauman, and Fei Sha. "Retrospective Encoders for Video Summarization: 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VIII". In: Sept. 2018, pp. 391–408. ISBN: 978-3-030-01236-6. DOI: 10.1007/978-3-030-01237-3_24.

[7] Zrar Kh. Abdul and Abdulbasit K. Al-Talabani. "Mel Frequency Cepstral Coefficient and its Applications: A Review". In: *IEEE Access* 10 (2022), pp. 122136–122158. DOI: 10.1109/ACCESS.2022.3223444.

[8] "The Knapsack Problem". In: *Combinatorial Optimization: Theory and Algorithms.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 439–448. ISBN: 978-3-540-71844-4. DOI: 10.1007/978-3-540-71844-4_17. URL: https://doi.org/10.1007/978-3-540-71844-4_17.

[9] Yale Song et al. "TVSum: Summarizing web videos using titles". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 2015, pp. 5179–5187. DOI: 10.1109/CVPR.2015.7299154.

[10] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization.* 2017. arXiv: 1412.6980 [cs.LG].