AVT-VIBE – OVERVIEW OF TWO MODELS FOR THE ICIP 2024 GRAND CHALLENGE ON VIDEO COMPLEXITY

Steve Göring, Rakesh Rao Ramachandra Rao

Audiovisual Technology Group, Technische Universität Ilmenau, Germany;
Email: [steve.goering, rakesh-rao.ramachandra-rao]@tu-ilmenau.de

ABSTRACT

For the ICIP 2024 Grand Challenge on Video Complexity we submitted two models, which after the evaluation were ranked as second and third best models of overall 18 submissions. In the following brief overview, we describe the challenge and decisions which resulted in the two models. Both models are kept simple, and only rely on a recent ffmpeg installation and can be run with linux. The implementation is publicly available[1].

Index Terms— video complexity, grand challenge

1. INTRODUCTION

Spatial (within frame) and temporal (across frames) redundancies are the key for efficient video compression. Hence, it is important to estimate them for a video especially for encoding optimization in streaming scenarios. One way of estimating such redundancies is by measuring the complexity of the video. Traditionally, SI and TI [1] have been used for this purpose. However, SI and TI do not correlate well with the compressibility of the video [2, 3]. Thus, there is a need to develop measures for the complexity of the video to aid better compressibility. E.g., Satti et al. [3] propose a CRF-based complexity measure which has been shown to be a better measure of encoding complexity, or with DNNs and CRF in [4]

Another important factor in estimating video complexity is the computation speed. All the aforementioned approaches are not real time. VCA [5] tends to alleviate this problem by providing a computationally efficient approach to estimate the spatial and temporal complexity of a video. However, it also does not correlate well with the encoding complexity, e.g., estimating the bitrate for a given CRF value. Hence, there is a need for computationally efficient video complexity estimators that correlate well with encoding complexity. In this paper we propose two approaches to predict the accurate and computationally efficient bitrate complexity of a given video.

[1] https://github.com/Telecommunication-Telemedia-Assessment/avt_vibe

2. PRE-ANALYSIS

We ran various experiments to select the ffmpeg [6] parameters. In the following we use the video 2.mp4 of the Inter4K dataset [7] for demonstration. For each of the subsequent experiments we ran 20 repetitions for the time measurement. We varied the preset, the CRF parameter, and the amount of selected frames. All measurements have been performed on a Lenovo Thinkpad T14s with AMD Ryzen 7 PRO 6850U.

2.1. Preset Encoding Speed

H.264 offers 11 different presets (ultrafast to placebo[2]). We only included ultrafast, superfast, veryfast, faster, fast, medium, and slow to analyze the required processing time.

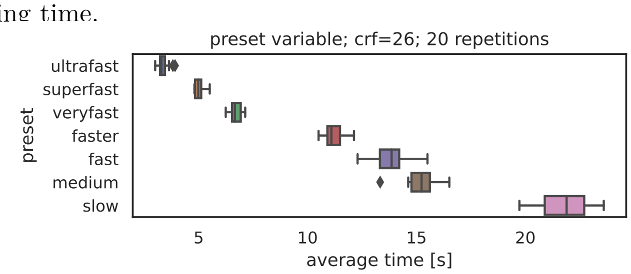

Fig. 1. Preset parameter compared to processing time.

In Fig. 1 it can be seen that the medium preset takes approximately 15 s while the superfast preset takes only $\approx 5\,s$ and ultrafast takes $\approx 3\,s$. Thus we selected for the AVT-VIBE-S superfast as the preset. For the model AVT-VIBE-R we selected the ultrafast preset.

2.2. CRF Setting

H.264 can have CRF values from 0 to 51[2], in our analysis we only selected even values in [20, 34] with a fixed preset of medium. The competition aimed for the bitrate at a CRF of 26, however CRF values around this value may result in similar bitrates. In the first instance we checked the processing time for varying CRF values.

[2] https://trac.ffmpeg.org/wiki/Encode/H.264

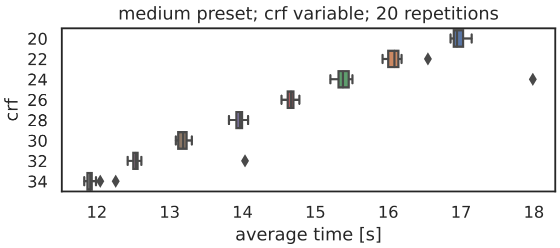

Fig. 2. CRF vs. processing time.

Results are shown in Fig. 2, here the overall time ranges from 12 to 17 seconds, while the target CRF of 26 is $\approx 15\,s$. For the CRF=28 the time is only slightly reduce to 14 s. For the AVT-VIBE-S , we selected CRF=28 also because it was minimally improving the performance for the bitrate estimation. AVT-VIBE-R uses the target CRF of 26.

2.3. Frame Sampling and other Parameters

We evaluated the speed gain for frame sampling (e.g. every n-th frame for $n \in \{5, 10, 20, 30, 60\}$), here, $n = 10$ showed the best tradeoff between speed and accuracy.

Our first prototypes were built around Python, however, we skipped Python for the final submission and all models just run with BASH due to time requirements. Furthermore, we added the keyframe interval of 10 for AVT-VIBE-S and tune parameter as zerolatency for both models. AVT-VIBE-R used a curve fitting approach, while AVT-VIBE-S just outputs the bitrate.

3. PERFORMANCE EVALUATION

For the performance evaluation considering bitrate estimation, we used two datasets, the first is the Inter4K dataset [7] (1k videos), as it was required by the competition. Furthermore, we used the 2160p videos from the YouTube UGC Dataset (YTUGC) [8] (125 videos; H.264 CRF=10 encoded). These videos were then trimmed to the first 5 seconds and we estimated the target bitrates with re-encoded using H.264 CRF=26 medium preset.

3.1. AVT-VIBE-S and AVT-VIBE-R

The final command of AVT-VIBE-S is shown in Listing 1 and AVT-VIBE-R in Listing 2 respectively.

```
#!/bin/bash
 if [[ "$1" == "" ]]; then
     echo "usage: ./est_sf_crf.sh <video>"
     exit 0
 fi
 res=$(ffmpeg -y -i "$1" -filter:v "select='not(mod(n\,10))',setpts=N/FRAME_RATE/
        ↪ TB" -c:v libx264 -x264opts keyint=10:min-keyint=10:scenecut=-1 -crf 28
        ↪ -preset superfast -tune zerolatency -f mp4 /dev/null 2>&1 | grep "
        ↪ libx264")

 bitrate=$(echo "$res" | grep "kb/s" | sed "s|.*kb.s:||g")

 echo "{\"video\": \"$1\", \"estimated_bitrate_[kb/s]\": $bitrate }"
```

Listing 1. AVT-VIBE-S

```
#!/bin/bash
# requires ffmpeg and bc installed
 if [[ "$1" == "" ]]; then
     echo "usage: ./est_uf_qp_sr_crf.sh <video>"
     exit 0
 fi
 res=$(ffmpeg -y -i "$1" -filter:v "select='not(mod(n\,10))',setpts=N/FRAME_RATE/
        ↪ TB" -c:v libx264 -crf 26 -preset ultrafast -tune zerolatency -f mp4 /
        ↪ dev/null 2>&1 | grep "libx264")

 bitrate=$(echo "$res" | grep "kb/s" | sed "s|.*kb.s:||g")
 skipratio=$(echo "$res" | grep "skip:" | sed "s|.*:||g" | sed "s|%||g")
 pframe_qp=$(echo "$res" | grep "frame P:" | sed "s|.*Avg QP:||g" | sed "s|size:.*
        ↪ ||g" | sed "s| *||g")

 a1=21749.2761
 b1=0.13837051
 c1=30121.3953
 d11=-31616.5783
 e1=80835.9569

 bitrate=$(echo "scale=0;$a1 + $b1*$bitrate + $c1 * l($pframe_qp / 51) + $d11*e(-
        ↪ e1* $skipratio/ 100 )" | bc -l)

 echo "{\"video\": \"$1\", \"estimated_bitrate_[kb/s]\": $bitrate }"
```

Listing 2. AVT-VIBE-R

3.2. Performance

Table 1. Performance values for both models considering Inter4K and YT-UGC datasets.

Dataset	Pearson	Kendall	Spearman	Model
Inter4K	0.783	0.623	0.812	AVT-VIBE-R
Inter4K	0.848	0.715	0.886	AVT-VIBE-S
YTUGC	0.838	0.707	0.876	AVT-VIBE-R
YTUGC	0.869	0.732	0.901	AVT-VIBE-S

In Table 1 the performance values for both models are summarized, indicating that both models have a similar high correlation. Within the competition the evaluation focused on speed and Pearson Correlation. AVT-VIBE-S takes $\approx 1.8\,s$ for 2.mp4 and AVT-VIBE-R takes $\approx 1.6\,s$ in contrast to the full encoding time of $\approx 15\,s$.

4. CONCLUSION AND FUTURE WORK

As part of the “Grand Challenge on Video Complexity” we developed and submitted two models to estimate bitrate complexity. In this paper, in addition to the algorithms of the submitted models, we also present a detailed analysis of how the different encoding parameters were chosen. Furthermore, the developed models were evaluated on a separate validation dataset created using videos from the YT-UGC dataset prior to submitting these models to the competition. This evaluation showed that the models performed well both in terms of accuracy measured using Pearson Correlation and computation speed. As future work, these models will be extended to be applicable to other codecs such as H.265, AV1, etc.

5. REFERENCES

[1] ITU-T Rec. P.910, “Subjective video quality assessment methods for multimedia applications,” 1999.

[2] Werner Robitza, Rakesh Rao Ramachandra Rao, Steve Göring, and Alexander Raake, “Impact of spatial and temporal information on video quality and compressibility,” in 13th International Conference on Quality of Multimedia Experience (QoMEX), 2021.

[3] Shahid Mahmood Satti, Matthias Obermann, Christian Schmidmer, and Michael Keyhl, “Video encoding complexity characterization,” in 14th International Conference on Quality of Multimedia Experience, QoMEX 2022, Lippstadt, Germany, September 5-7, 2022. 2022, pp. 1–4, IEEE.

[4] Francisco Micó-Enguídanos, Wilmer Moina-Rivera, Juan Gutiérrez-Aguado, and Miguel Garcia-Pineda, “Per-title and per-segment crf estimation using dnns for quality-based video coding,” Expert Systems with Applications, vol. 227, pp. 120289, 2023.

[5] Vignesh V Menon, Christian Feldmann, Hadi Amirpour, Mohammad Ghanbari, and Christian Timmerer, “Vca: video complexity analyzer,” in Proceedings of the 13th ACM Multimedia Systems Conference, New York, NY, USA, 2022, MMSys ’22, p. 259–264, Association for Computing Machinery.

[6] Suramya Tomar, “Converting video formats with ffmpeg,” Linux Journal, vol. 2006, no. 146, pp. 10, 2006.

[7] Alexandros Stergiou and Ronald Poppe, “Adapool: Exponential adaptive pooling for information-retaining downsampling,” arXiv preprint, 2021.

[8] Yilin Wang, Sasi Inguva, and Balu Adsumilli, “Youtube ugc dataset for video compression research,” in 2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP). IEEE, 2019, pp. 1–5.