Machine-Learning-Based Method for Content-Adaptive Video Encoding

Sergey Zvezdakov*, Denis Kondranin† and Dmitry Vatolin‡
Faculty of Computational Mathematics and Cybernetics
Lomonosov Moscow State University
Moscow, Russia
Email: *szvezdakov@graphics.cs.msu.ru, †denis.kondranin@graphics.cs.msu.ru, ‡dmitriy@graphics.cs.msu.ru

***Abstract*—Video codecs have several dozen parameters that subtly affect the encoding rate, quality and size of the compressed video. Codec developers, as a rule, provide standard presets that on average yield acceptable performance for all videos, but for a given video, certain parameters may yield more efficient encoding. In this paper, we propose a new approach to predicting video codec presets to increase compression efficiency. Our effort involved collecting a new representative video-sequence dataset from Vimeo.com. An experimental evaluation showed relative bitrate decreases of 17.8% and 7.9%, respectively for the x264 and x265 codecs with standard options, all while maintaining quality and speed. Comparison with other methods revealed significantly faster automatic preset selection with a comparable improvement in results. Finally, our proposed content-adaptive method predicts presets that archive better performance than codec-developer presets from MSU Codec Comparison 2020 [1].**

***Index Terms*—video compression, multi-objective optimization, content adaptation, content-adaptive encoding, video dataset, x264, x265**

I. Introduction

According to the Ericsson Mobility Report 2020 [2], video delivery will consume about 76% of global wireless bandwidth by 2025. Currently, the share of video on the Internet is 63%. This huge fraction is due to the increasing duration of the transmitted content and the development of new high-definition formats. The massive amount of video data stimulates creation of video coding tools: new encoders and coding standards are emerging, and existing compression algorithms are becoming more complex. Modern video codecs have 50+ options, making it difficult for users to select an effective preset to meet their requirements. Therefore, most employ standard presets when encoding video.

The task of finding a suitable preset can be formulated as a multicriterion-optimization problem where the input video can strongly influence the solution. In this work, we studied the selection of a sufficient number of presets to provide the maximum possible compression rate using a new dataset. We also propose a method that adaptively predicts the best encoding preset for the input video from a predefined list. We exclude the encoded video's bitrate from the preset so the user can change it when using our solution.

II. Related work

Analyzing and improving the performance of video encoder presets can follow several directions. In [3], the authors propose finding the optimal values for eight H.264 options. First, their method predicts the main options for the encoder and decoder using linear regression; next, it uses the NSGA-II genetic algorithm [4], to select the optimal presets. A similar approach appears in [5], where the authors increased the number of experimental videos and the number of video codecs, then tested their approach on the H.265 codec. Another representative of classical techniques is [6]; it employs a greedy algorithm that iteratively varies one encoder option in a given preset and tries to achieve the Pareto optimal front.

The main goal of [7] is to predict the codec's output characteristics on the basis of other outputs. The researchers propose a method that runs the codec several times to train the machine learning model, allowing it to predict the results for other presets (such as video quality and encoding speed) and to choose the Pareto optimal set. The authors of [8] describe a technique that splits the database of video codec runs into several parts and selects the Pareto optimal presets for each part. This work also proposes an ML-based algorithm for classifying new videos into the groups from the first stage.

Features extracted from video are important to predicting the efficiency of codec presets. Therefore, the comparison from [1] as well as the work in [9] and [10] use SI and TI features to select videos for codec testing. The authors suggest that estimating these features enables estimation of video complexity for encoders, allowing them to create a representative set of videos. Also, a video feature analysis (including neural-network-based techniques) appears in [11].

Some efforts devoted to optimizing the encoding process emphasize choosing the optimal points at which to change resolution in the video bitrate ladder. In [10] the authors investigate optimal strategies for estimating the switching bitrate in such ladders. They tested their method on 2K and 4K resolutions with the VP9 encoder. Netflix employs a similar technique in its encoding pipeline, called the "Dynamic Optimizer" [12]. But unlike the usual bitrate ladders, the Dynamic Optimizer selects the optimal resolution and bitrate for every shot in each video.

A similar problem is under investigation in [13], consisting of two main parts. In the first part, the authors consider 10 simple video features to demonstrate that the possibility of adaptively predicting basic image encoding parameters for the JPEG and WebP codecs. In the second part, they apply a

TABLE I: Main characteristics of existing methods.

Paper	Codecs	Number of videos	Number of tested codec options	Encoder run necessary for prediction?	Predicted values
F. Al-Abri *et al.* [3]	JVT264	1	9	Yes	Enc_Complexity, RateDistortion, Dec_Complexity, Dec_Memory
V. Popov [6]	x264	3	7	Yes	$S(p,V)$, $Q(p,V)$ (SSIM-based)
M. Naccari *et al.* [9]	x265	16	0	Yes	BD-rate, Enc_Speed, Enc_Complexity
C. Chen *et al.* [10]	VP9	7966	1	Yes	SSIM
M. Al-Barwani *et al.* [5]	x264, x265	6	4	Yes	PSNR, Bitrate, Enc_Time
S. Zvezdakov *et al.* [7]	x264	351	6	Yes	$S(p,V)$, $Q(p,V)$ (SSIM-based)
"Dynamic Optimizer" [12]	VP9	∼1,000,000	2	Yes	VMAF
O. Murashko *et al.* [13]	x265	4	2	No	SSIM, Enc_Time, FileSize
R. Kazantsev *et al.* [8]	x264	351	6	No	$S(p,V)$, $Q(p,V)$ (SSIM-based)
Proposed	x264, x265	1145	42 (x264), 72 (x265)	No	$S(p,V)$, $Q(p,V)$ (SSIM-based)

similar method to predict video codec presets and extend it to adaptive video compression.

A comparison of these various methods appears in Table I.

III. Dataset creation

To analyze video codecs, we created our own dataset by running the codecs, as no public alternatives contain enough videos tested on varied encoder options. In this section, we describe our dataset and explain why we took this approach.

A. Video datasets

We wanted to create a representative dataset containing videos that codecs must handle in real situations. Using the selection technique from "MSU Video Codecs Comparison 2016" [14], we have over four years (2016–2019) analyzed more than one million videos and downloaded 7,036 unique high bitrate videos from Vimeo.com. Fig. 1 shows the bitrate distributions for our video dataset by year. We resized and cropped the 4K videos to Full HD resolution to avoid compression artifacts, and at shot transitions we cut all videos to samples using an approximate length of 500 frames. Next we randomly selected 1,060 videos from the set.

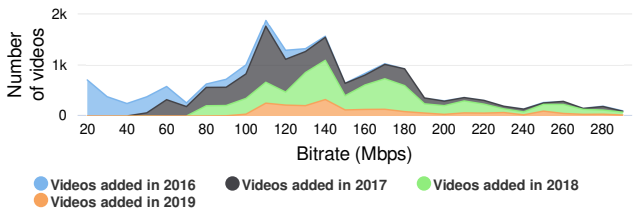


Fig. 1: Bitrate distribution for videos downloaded from Vimeo.com.

We also collected public datasets for codec testing; for this paper we used media.xiph.org (Full HD, Netflix) [15], SJTU 4K [16], JCT-VC video collection [17] and the UVG Dataset [18]. If the video resolution was higher than Full HD, we downsized it to Full HD.

To evaluate spatial and temporal complexity, we employ the technique from [10]. We also visualized the complexity of the videos collected from Vimeo.com and other public sets[1]. The distribution shows that some public datasets (like xiph.org "Full HD" collection or JCT-VC video collection) are concentrated in areas with high spatial and temporal complexity, although in real life codecs typically handle simpler videos.

Our final video dataset consists of 1145 videos. It contains most popular video types: animation, computer graphics, television programs and vlogs, sports events, drone footage, and more.

B. Encoders and presets

For our experiments, we selected two of the most popular open-source video encoders: x264 and x265. For each one we created a set of highly successful encoding presets. For this task we used option-popularity statistics[2] from [19] and [20], codec-developer presets from the MSU Codec Comparison 2016-2019, and our algorithm for iteratively discovering Pareto optimal presets. The final set consists of 686 presets for x264 and 424 presets for x265, which were obtained by using 42 and 72 codec options respectively.

At the Faculty of Computation Mathematics and Cybernetics of Lomonosov Moscow State University, we built a cluster of 253 identical computers, each based on an Intel Xeon E3-1245v3 processor with 16GB of RAM. Owing to a lack of SSDs, each computer had a RAM disk for storage. Using this cluster, we ran each preset on every video in our dataset. The test employed seven bitrates from 1 to 12 Mbps. In total we launched the x264 encoder about 16.5 million times and the x265 encoder about 7.1 million times.

Our measurement of objective video quality used the SSIM metric, since it works quickly and is less susceptible to hacking than VMAF [21]. For each preset p and video V, we calculated

[1]SI/TI distribution of datasets: https://papers.evt.guru/ml-preset-predictor/si-ti.html

[2]Interactive charts with collected statistics: https://papers.evt.guru/ml-preset-predictor/options-stats.html

the relative bitrate $Q(p, V)$ with same objective quality and relative encoding speed $S(p, V)$ of the encoded file using the method from [22].

IV. Proposed method

A. Problem definition

The main idea of content-adaptive encoding (CAE) is to replace a reference preset p_{ref} (by which we encoded all previous video groups) with other presets that will more efficiently encode an input video. For simplicity, we assume the relative encoding speed $S(p_{\text{ref}}, V)$ of the reference preset p_{ref} is satisfactory. We can then formulate CAE for video V using Eq. 1, where P is the set of all possible presets.

$$p_{\text{best}}(V) = \underset{p:p\in P:S(p,V)\leq(1+\epsilon)S(p_{\text{ref}},V)}{\operatorname{argmin}} Q(p, V) \qquad (1)$$

We added $\epsilon = 0.05$, a small constant, because encoding time always includes a little noise that is impossible to completely filter.

To solve this problem in a reasonable time, we can replace the ground-truth values $S(p, V)$ and $Q(p, V)$ with predicted values $\hat{S}(p, V)$ and $\hat{Q}(p, V)$ using any video codec model. This approach underwent testing in [7] but has two major drawbacks. First, it requires application of the model to all possible presets, a time-consuming task. Second, the predicted values ($\hat{S}(p, V)$ and $\hat{Q}(p, V)$) are insufficiently accurate, greatly reducing the effectiveness of the selected preset in many cases. Therefore, we decided to replace the regression problem with a classification problem.

B. Converting to classification

Our proposed method aims to solve the classification problem for each video and reference preset: the model must select one of several presets.

We assume it is possible to improve on the reference preset by choosing from a few predefined presets. To confirm this hypothesis, we conducted an experiment to determine the relationship between the size of the selected preset sample and the maximum possible improvement (for our dataset). Formally, this problem requires finding presets $p_1, ..., p_k$ for a set of videos H and for each k such that:

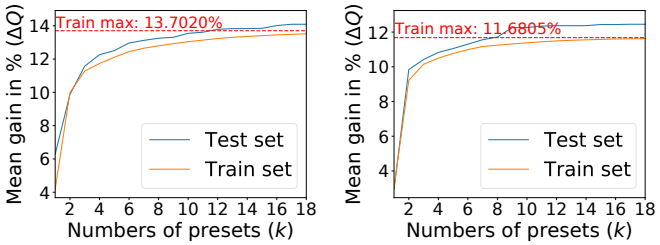
$$\underset{(p_1,...,p_k)\in P}{\operatorname{argmin}} \sum_{V\in H} \min_{i:S(p_i,V)\leq(1+\epsilon)S(p_{\text{ref}},V)} Q(p_i, V) \qquad (2)$$

We tested this hypothesis on the x264 and x265 codecs using ΔQ to estimate the improvement:

$$\Delta Q = \frac{Q(p_{\text{ref}}, V) - Q(p_{\text{best}}(V), V)}{Q(p_{\text{ref}}, V)} * 100 \qquad (3)$$

where $p_{\text{best}}(V)$ is the proposed preset for video V and reference preset p_{ref} (from Eq. 1). We considered a subset of all videos as sets of videos H. In practice, ΔQ is the compression-ratio increase (in percent) relative to the standard preset (with the same quality and encoding time).

We split our encoder dataset into two parts. The training subset consisted of 973 videos, with the best presets $p_1, ..., p_k$ selected for each k. On the testing subset we only measured $\Delta Q(p_i)$; the size of test was 172 videos.



(a) Gain for *veryslow*, x264. (b) Gain for *veryslow*, x265.

Fig. 2: Quality improvement versus number of presets for the *veryslow* preset of x264 (a) and x265 (b). The red line represents the maximum training set gain. Our approach achieves 90% of the maximum gain at small numbers of presets ($k = [6, 10]$). As the number of presets increases, the gain also increases because more presets means a better fit for all video types.

Fig. 2 shows the results[3]. For each standard preset taken as a reference, just a few presets are sufficient. For example, in the case of the x265 *slower* preset, only 12 (of 424) presets are necessary to yield most of the gain. Also, the results demonstrate that these presets have not undergone retraining with a specific training sample because they exhibit same gain on the test subset.

Thus, content-adaptive encoding reduces to solving a classification problem. For each reference preset, we can train a model that will take video features as input and predict the best preset from a list of predefined presets chosen in accordance with Eq. 2.

C. Video features

To extract features from the input video, we selected the popular ones from other papers related to this topic. For further experiments, we chose the following:

- Spatial and temporal complexity from [10] (SI_{Codec}, TI_{Codec});
- Spatial and temporal complexity from [9] (SI, TI);
- Temporal complexity of video TI_{ME} (motion estimation from [23]);
- Blur values $blur$ from [24] and $blur_{\text{lap}}$ from [8].

All features except SI_{Codec} and TI_{Codec} are frame by frame, so we calculated the average and standard deviation for each one over the entire input video.

D. Hyperparameter selection and training details

To make a correct final comparison, we separated 53 videos from the xiph.org and JCT-VC datasets, omitting them from any further experiments. Our tests only involved the remaining 1,092 videos. All testing employed 5-fold cross-validation.

[3]Charts for other presets: https://papers.evt.guru/ml-preset-predictor/presets-gain-stats.html

TABLE II: Comparison of proposed method with the best possible improvement for x265 and x264 using created dataset.

	x264									
	ultrafast	*superfast*	*veryfast*	*faster*	*fast*	*medium*	*slow*	*slower*	*veryslow*	*placebo*
Proposed	71.86%	14.18%	8.63%	12.65%	12.80%	11.91%	9.88%	12.69%	9.93%	10.00%
Proposed, custom loss	72.74%	14.18%	9.80%	13.05%	13.55%	12.81%	9.54%	12.88%	9.85%	9.85%
Best (in our dataset)	73.72%	19.73%	13.73%	17.58%	18.50%	16.40%	13.69%	15.99%	13.68%	13.36%
	x265									
	ultrafast	*superfast*	*veryfast*	*faster*	*fast*	*medium*	*slow*	*slower*	*veryslow*	*placebo*
Proposed	11.83%	15.60%	3.75%	3.59%	2.78%	1.64%	7.86%	10.04%	8.76%	8.59%
Proposed, custom loss	12.09%	15.82%	3.82%	3.73%	4.69%	1.77%	8.60%	10.36%	9.28%	9.04%
Best (in our dataset)	14.66%	18.40%	6.49%	6.38%	6.44%	6.38%	11.67%	12.31%	11.88%	11.47%

Each model received as input a different subset of video features described in the previous subsection. We trained the models to predict for each input video the best preset from the set $p_1, ..., p_k$, which was obtained using Eq. 2 and k ranged from 2 to 15. Our tests have covered the following classifiers:

- C-Support Vector Classification [25];
- Random forests [25];
- CatBoost Gradient Boosting [26];
- XGBoost Gradient Boosting [27].

For each standard x264 and x265 preset, we used a grid search to select the best configuration of video features, classifiers (and parameters), and number of predefined presets. In most cases, configurations with XGBoost exhibited the best results, so we used only that classifier in further experiments.

For example, the best results for the *veryslow* preset of x265 were for the XGBoost classifier with 100 estimators and seven predefined presets. On the other hand, for x265 *superfast* the best model requires 12 predefined presets.

E. Customizing the loss function

XGBoost uses the softmax objective function and cross-entropy loss for multiclass classification. But our classification problem is nonstandard, and the error will differ greatly when choosing the wrong preset: it can be either insignificant (if the selected preset works in almost the same way as the ground-truth preset) or huge (if the predicted preset is not very similar to the ground-truth preset).

The standard cross-entropy loss is presented in Eq. 4, where $y^{(j)}$ is a true class label and $prob_{ji}$ is the probability that the model returned.

$$L = \frac{1}{N}\sum_{j=1}^{N}\sum_{i=1}^{k}\mathbb{1}\{y^{(j)} = i\} * \log(prob_{ji}) \tag{4}$$

To accommodate the differences between predefined presets $p_1, ..., p_k$ we replaced $\mathbb{1}\{y^{(j)} = i\}$ with y_{ji} (Eq. 5), where p_{ref} is the reference preset, p_x is the preset for label x and V_j is jth video in the datatet.

$$y_{ji} = \begin{cases} 0, \text{if } Q(p_{\text{ref}}, V_j) \leqslant Q(p_i, V_j) \\ 1, \text{if } y^{(j)} = i \\ Q(p_i, V_j)/Q(p_{y^{(j)}}, V_j) - 1, \text{otherwise} \end{cases} \tag{5}$$

F. Results

Table II compares the proposed method with the maximum possible improvement for a given dataset. For x265 with the *faster* preset, our method can reduce the file size by 3.73%. The proposed loss function in some cases enables a further 1-2% reduction.

For x264, we achieved a much greater improvement relative to the reference presets. Given the *slower* preset, the filesize decrease is 12.88%. Moreover, given the *placebo* preset, the encoding time decreased by 68.09% when the file size decreased by 9.85%. We analyzed cases where our method mistakenly chose a preset; usually, this situation occurs when the video contains transitions between shots that differ in type (for example, from titles to sporting events).

V. Comparison with other methods

We compared our method with analogs using the dataset from [8] which contains 351 videos. This dataset contains the Cartesian product of six options for x264. Table III shows that our method exhibits comparable results and occasionally works better than slow methods [6], [4], [7]. It also delivers quality superior to that of the fast method [8]. For the *faster* preset, however, our method failed to significantly improve the video codec's performance.

TABLE III: Average bitrate savings of the Pareto optimal presets versus standard presets, along with execution time for each method using dataset from [8] (351 video, x264).

Preset	Popov [6]	NSGA-II [4]	Zvezdakov [7]	Kazantsev [8]	Proposed
faster	8.0%	**15.9%**	11.0%	15.8%	3.0%
fast	29.0%	**30.2%**	30.2%	21.3%	28.6%
medium	28.4%	29.7%	**29.8%**	21.8%	**29.8%**
slow	34.9%	34.9%	34.9%	27.7%	**35.0%**
slower	32.2%	32.2%	32.2%	24.9%	**32.7%**
veryslow	28.3%	**29.0%**	28.7%	9.7%	28.9%
placebo	29.0%	28.3%	29.4%	10.5%	**29.5%**
Time	2.78 h.	3.8 h.	2.14 h.	735.5 s.	**468.4 s.**

To show the practical applicability of our proposed method, we tested it on videos from the JCT-VC and xiph.org collections. At the beginning of this research, we separated these videos from our test datasets and didn't use them to train

our models. Also, we added developer presets from [1] as a baseline. We chose only two standard presets for Table IV, because only two developer presets appear in [1]. For each developer preset, we selected the standard preset closest to it in encoding speed. More detailed comparison results are available online[4]. The comparison results indicate that our proposed method yields presets that work more efficiently than presets chosen by codec developers. For the x265 *medium* preset, the maximum improvement is small due to the peculiarities of the dataset, since among all our 424 presets it is impossible to achieve greater improvement.

TABLE IV: Average bitrate savings for presets obtained using each method versus standard presets (xiph.org and JCT-VC datasets, 49 videos, x264 and x265).

	x264		x265	
	slow	*veryslow*	*medium*	*veryslow*
Codec developers [1]	2.97%	0.87%	0.0%	1.96%
Proposed	7.65%	5.54%	0.75%	5.11%

VI. Conclusion

This paper presents a novel method of content-adaptive video encoding. Our method is independent of the video codec's architecture and implementation and is applicable to a variety of codecs and coding standards. It works with numerous options, does not require running a video codec and has undergone testing on more videos than previous works have done. Our experimental comparison on videos from real users demonstrated a 4–15% bitrate savings over the x264 and x265 standard presets while delivering similar compressed video quality and encoding time.

We would like to thank to the Faculty of Computational Mathematics and Cybernetics of Lomonosov Moscow State University for providing the computing resources that enabled our investigation. This work was partially supported by the Russian Foundation for Basic Research under Grant 19-01-00785a.

References

[1] D. Kulikov, M. Erofeev, A. Antsiferova, E. Sklyarov, A. Yakovenko, and N. Safonov. *HEVC/AV1 Video Codecs Comparison 2020*. Dec 07, 2020. Accessed on: Jan. 15, 2021. [Online]. Available: https://www.compression.ru/video/codec_comparison/hevc_2020/

[2] *Ericsson Mobility Report November 2020*. Mar 9, 2020. Accessed on: Jan. 15, 2021. [Online]. Available: https://www.ericsson.com/49da93/assets/local/mobility-report/documents/2020/june2020-ericsson-mobility-report.pdf

[3] F. Al-Abri, X. Li, E. A. Edirisinghe, and C. Grecos, "A novel framework for multi-objective optimization of video codecs," in *2009 International Conference on CyberWorlds*. IEEE, 2009, pp. 195–202.

[4] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[5] M. Al-Barwani and E. Edirisinghe, "A machine learning based framework for parameter based multi-objective optimisation of a h.265 video codec," in *2016 Future Technologies Conference (FTC)*. IEEE, 2016, pp. 553–559.

[4]Comparison results on JCT-VC and xiph.org datasets: https://papers.evt.guru/ml-preset-predictor/xiph-comparison.html

[6] V. Popov, "Automatic method of choosing pareto optimal video codec's parameters," Master's thesis, Lomonosov Moscow State University, 2009.

[7] S. V. Zvezdakov and D. S. Vatolin, "Building a x264 video codec model," in *Innovative technologies in cinema and education: IV International Symposium*. VGIK Moscow, 2017, pp. 56–65.

[8] R. Kazantsev, S. Zvezdakov, and D. Vatolin, "Machine-learning-based method for finding optimal video-codec configurations using physical input-video features," in *2020 Data Compression Conference (DCC)*. IEEE, 2020, pp. 374–374.

[9] M. Naccari, A. Gabriellini, M. Mrak, S. G. Blasi, I. Zupancic, and E. Izquierdo, "Hevc coding optimisation for ultra high definition television services," in *2015 Picture Coding Symposium (PCS)*. IEEE, 2015, pp. 20–24.

[10] C. Chen, S. Inguva, A. Rankin, and A. Kokaram, "A subjective study for the design of multi-resolution abr video streams with the vp9 codec," *Electronic Imaging*, vol. 2016, no. 2, pp. 1–5, 2016.

[11] M. Covell, M. Arjovsky, Y.-c. Lin, and A. Kokaram, "Optimizing transcoder quality targets using a neural network with an embedded bitrate model," *Electronic Imaging*, vol. 2016, no. 2, pp. 1–7, 2016.

[12] I. Katsavounidi. *Dynamic optimizer — a perceptual video encoding optimization framework*. Mar 5, 2018. Accessed on: Jan. 15, 2021. [Online]. Available: https://netflixtechblog.com/dynamic-optimizer-a-perceptual-video-encoding-optimization-framework-e19f1e3a277f

[13] O. Murashko, J. Thomson, and H. Leather, "Predicting and optimizing image compression," in *Proceedings of the 24th ACM international conference on Multimedia*. ACM, 2016, pp. 665–669.

[14] D. Vatolin, D. Kulikov, M. Erofeev, S. Dolganov, and S. Zvezdakov. *Eleventh MSU Video Codecs Comparison*. Aug 22, 2016. Accessed on: Jan. 15, 2021. [Online]. Available: https://www.compression.ru/video/codec_comparison/hevc_2016/

[15] *Xiph.org Video Test Media*. Accessed on: Jan. 15, 2021. [Online]. Available: https://media.xiph.org/video/derf/

[16] L. Song, X. Tang, W. Zhang, X. Yang, and P. Xia, "The sjtu 4k video sequence dataset," in *2013 Fifth International Workshop on Quality of Multimedia Experience (QoMEX)*. IEEE, 2013, pp. 34–35.

[17] F. Bossen, "Common HM test conditions and software reference configurations (JCTVC-L1100)," *Joint Collaborative Team on Video Coding*, 2013.

[18] A. Mercat, M. Viitanen, and J. Vanne, "Uvg dataset: 50/120fps 4k sequences for video codec analysis and development," in *Proceedings of the 11th ACM Multimedia Systems Conference*, 2020, pp. 297–302.

[19] *Doom9's Forum*. Accessed on: Jan. 15, 2021. [Online]. Available: http://forum.doom9.org

[20] *VideoHelp Forum*. Accessed on: Jan. 15, 2021. [Online]. Available: https://forum.videohelp.com

[21] A. Zvezdakova, S. Zvezdakov, D. Kulikov, and D. Vatolin, "Hacking vmaf with video color and contrast distortion," in *CEUR Workshop Proceedings*, 2019, pp. 53–57.

[22] A. V. Zvezdakova, D. L. Kulikov, S. V. Zvezdakov, and D. S. Vatolin, "Bsq-rate: a new approach for video-codec performance comparison and drawbacks of current solutions," *Programming and Computer Software*, vol. 46, pp. 183–194, 2020.

[23] K. Simonyan, S. Grishin, D. Vatolin, and D. Popov, "Fast video super-resolution via classification," in *2008 15th IEEE international conference on image processing*. IEEE, 2008, pp. 349–352.

[24] F. Crete, T. Dolmiere, P. Ladret, and M. Nicolas, "The blur effect: perception and estimation with a new no-reference perceptual blur metric," in *Human vision and electronic imaging XII*, vol. 6492. International Society for Optics and Photonics, 2007, p. 64920I.

[25] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[26] L. Prokhorenkova, G. Gusev, A. Vorobev, A. Dorogush, and A. Gulin, "Catboost: Unbiased boosting with categorical features," in *Advances in Neural Information Processing Systems*, 2018, pp. 6638–6648.

[27] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: ACM, 2016, pp. 785–794. [Online]. Available: http://doi.acm.org/10.1145/2939672.2939785