CANF-VC: Conditional Augmented Normalizing Flows for Video Compression
Supplementary Materials

Yung-Han Ho[1], Chih-Peng Chang[1], Peng-Yu Chen[1], Alessandro Gnutti[2], and Wen-Hsiao Peng[1]

[1] Department of Computer Science, National Yang Ming Chiao Tung University, Taiwan wpeng@cs.nctu.edu.tw
[2] Department of Information Engineering, CNIT - University of Brescia, Italy alessandro.gnutti@unibs.it

This supplementary document provides additional materials to assist with the understanding of the performance and design of our CANF-VC. Specifically, it includes:

- CANF implementations
- Complexity characterization
- Rate-distortion curves with GOP 32
- Comparison with ELF-VC [10]
- Network details: CANF and motion extrapolation
- Temporal prior for motion coding
- Training strategy
- Subjective quality comparison
- Command lines for x265 and HM

1 CANF Implementations

Fig. A1 depicts two possible CANF implementations. Fig. A1a corresponds to the one presented in the main paper. It concatenates the motion-compensated reference frame x_c with the input frame x_t as input to all the encoding transforms. In comparison, the scheme in Fig. A1b additionally accepts x_c as input to all the decoding transforms. The former (decoding transforms w/o x_c) can be viewed as a special case of the latter (decoding transforms w/ x_c), which utilizes x_c for encoding transforms only. For the implementation of Fig. A1b, the latent code is decoded first to produce 16-channel features having the same spatial resolution as x_c. The resulting features are then concatenated with x_c before being processed further by the three convolution layers (the orange part in Fig. A1b) to complete the decoding transform. This implementation (Fig. A1b) has a slightly larger model size than Fig. A1a.

Table A1 presents the BD-rate comparison between the two CANF implementations. For experiments, we use the motion coder from DVC [7], while the inter-frame coder adopts the two different CANF implementations (Fig. A1a vs. Fig. A1b). It is seen that the more generalized implementation (decoding

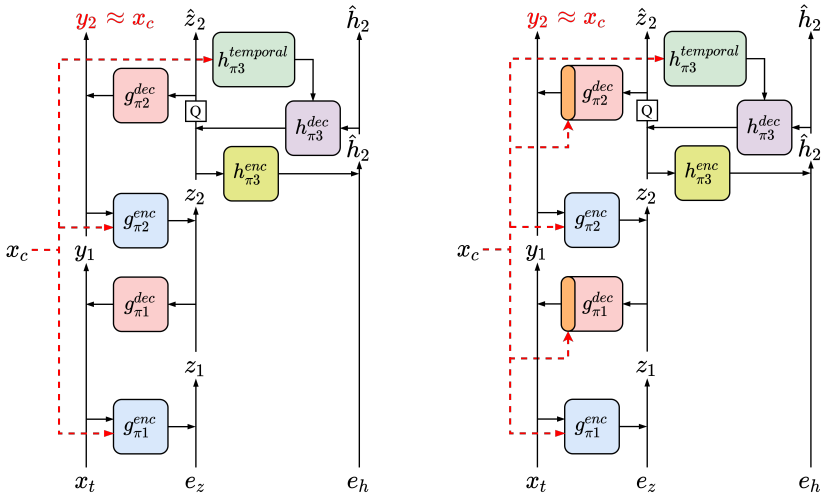


Fig. A1: Illustration of CANF implementations: (a) Decoding transforms w/o x_c and (b) Decoding transforms w/ x_c.

Table A1: BD-rate comparison between two CANF implementations for conditional inter-frame coding. The motion coder is from DVC [7]. The anchor is x265 in veryslow mode.

Implementations	UVG [9]	MCL-JCV [12]	HEVC-B [11]
Decoding transforms w/o x_c	-33.1%	-15.3%	-35.4%
Decoding transforms w/ x_c	-35.2%	-15.3%	-33.9%

transforms w/ x_c) has comparable performance to our current implementation (decoding transforms w/o x_c) on all three datasets. This justifies our choice of decoding transforms w/o x_c because of its comparable performance and simpler design.

2 Complexity Characterization

Table A2 presents the computational characteristics of different competing methods from the perspectives of multiply-accumulate (MAC) operations, encoding/decoding times for inference, and model sizes. It is to be noted that the prolonged encoding/decoding times of DCVC [5] are due to the use of an auto-regressive model for entropy coding. Our CANF-VC neither uses an auto-regressive model for motion coding nor uses it for inter-frame coding. Its larger MAC

Table A2: Complexity characterization in terms of MACs, encoding/decoding times, and model sizes. The MACs and runtimes of DVC [7] and DCVC [5] are collected by running the test code released by the respective authors on the same 1080Ti platform. The MACs are evaluated based on encoding a 1080p P-frame, while the encoding/decoding times are averaged over the first 100 P-frames of the Beauty sequence in UVG dataset.

Method	MACs	Encoding/Decoding Time	Model Size
DVC [7]	1725G	4.15 s/4.06 s	8.5M
DVC_Pro [8]	-	-/-	29M
FVC [4]	-	-/-	26M
DCVC [5]	2268G	7.70 s/32.90 s	8M
CANF-VC Lite	4012G	1.38 s/0.98 s	15M
CANF-VC	5088G	1.60 s/1.05 s	31M

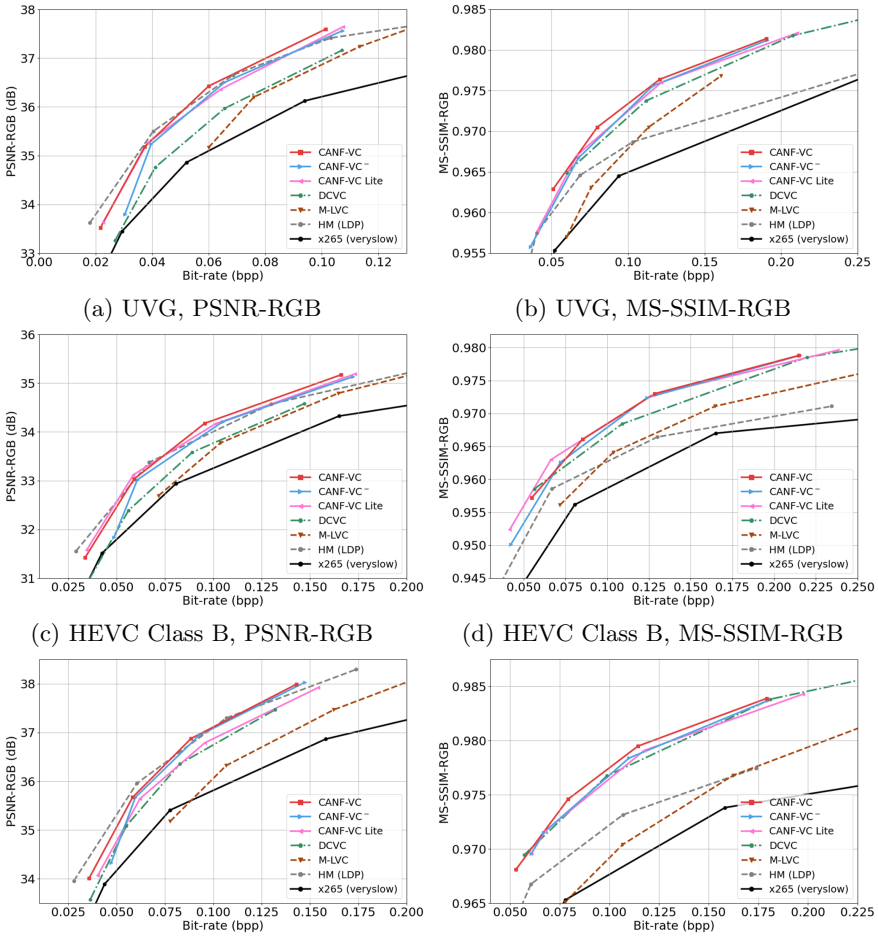
arises from stacking multiple autoencoding transforms. Nevertheless, its relatively short encoding/decoding times suggest that these autoencoding transforms are amenable to parallel computing. Lastly, we remark that the relatively longer encoding/decoding times of DVC [7] are due to their software implementation, particularly the entropy coding part. Our CANF-VC follows [2] to quantize the scale parameters from the hyperprior into 64 distinct values, enabling a fast table look-up to derive the probabilities for entropy coding. In contrast, DVC [7] does not quantize the scale parameters, and needs more time in evaluating higher-precision coding probabilities.

3 Rate-Distortion Curves with GOP Size 32

Fig. A2 presents rate-distortion curves for HM [1], DCVC [5], M-LVC [6], and our CANF-VC under GOP size 32 (see Table 2 for their BD-rate figures and Section 4.2 for detailed discussion). Except HM, all the competing methods use ANFIC [3] as the intra-frame coder for a fair comparison.

In terms of PSNR-RGB, our CANF-VC models outperform DCVC and M-LVC, except for CANF-VC Lite, which performs comparably to DCVC on MCL-JCV dataset. In addition, CANF-VC$^-$ shows worse performance than the other two CANF-VC variants at low rates because it does not include conditional motion coding, which is critical to low-rate compression performance. In comparison with HM, our CANF-VC shows better results at high rates, but worse results at low rates.

In terms of MS-SSIM-RGB, our CANF-VC models show slightly better results on UVG [9] and HEVC class B [11] than DCVC [5], and comparable results on MCL-JCV [12].



(a) UVG, PSNR-RGB (b) UVG, MS-SSIM-RGB

(c) HEVC Class B, PSNR-RGB (d) HEVC Class B, MS-SSIM-RGB

(e) MCL-JCV, PSNR-RGB (f) MCL-JCV, MS-SSIM-RGB

Fig. A2: Comparison of rate-distortion curves on UVG, HEVC Class B, and MCL-JCV datasets for both PSNR and MS-SSIM. All the competing methods use ANFIC [3] as the intra-frame coder and are evaluated under the same setting, namely, 96-frame encoding with GOP size 32. Results for DCVC [5] and M-LVC [6] are produced by their released code.

4 Comparison with ELF-VC [10]

Table A3 presents separately the BD-rate comparison with ELF-VC [10] since ELF-VC [10] adopts a GOP size of 16, which is rarely used by the other competing methods. Note that the results of ELF-VC [10] are from their paper because its software is unavailable. Moreover, we note that ELF-VC [10] uses its own intra-frame coder, the details of which are unavailable. Under the same GOP

Table A3: BD-rate comparison under the same GOP size 16. The anchor is x265 in veryslow mode. Except ELF-VC [10], all the competing methods adopt ANFIC [3] as the intra-frame coder.

	BD-rate (%) PSNR-RGB		BD-rate (%) MS-SSIM-RGB	
	UVG	MCL-JCV	UVG	MCL-JCV
DCVC (ANFIC)	-19.5	-7.9	-47.4	-46.5
ELF-VC	-30.8	-11.7	-55.3	-53.9
CANF-VC Lite	-35.5	-12.0	-46.0	-44.4
CANF-VC-	-34.7	-13.5	-45.4	-43.9
CANF-VC	-41.0	-19.9	-50.3	-48.9

size and in terms of PSNR-RGB, we see that the superiority of our CANF-VC models to ELV-VC [10] and DCVC (with ANFIC as the intra-frame coder) is obvious. However, ELF-VC [10] achieves the best MS-SSIM-RGB results among all the competing methods. We remark that this comparison is to provide additional information; a fair comparison would require the software of ELF-VC [10] and more information about its intra-frame coder.

5 Network Details: CANF and Motion Extrapolation

Fig. A3 shows the network details of our CANF, where we choose $N = 128$ and $C = 128$, with M set to 192 for inter-frame coding and 128 for motion coding, respectively. Our CANF-VC Lite adopts $N = 72$ and $C = 128$, with $M = 128$ for both inter-frame and motion coding.

Fig. A4 depicts the network architecture of our U-Net-based motion extrapolation network.

6 Temporal Prior for Motion Coding

For conditional motion coding, our current implementation adopts the extrapolated image $warp(\hat{x}_{t-1}; f_c)$ for constructing the temporal prior (Section 3.3 of the main paper). Table A4 presents additional results for the case where the predicted flow map f_c is used instead. We see that the former achieves 8% more rate savings than the latter (i.e. using f_c to construct the temporal prior), which justifies our design choice. The reason may be that the flow map f_c is not as informative as $warp(\hat{x}_{t-1}; f_c)$, which contains more semantic and texture information.

7 Training Strategy

Table A5 summarizes our training steps in three major phases. The first phase uses uncompressed, original frames as inputs to the motion estimation and the

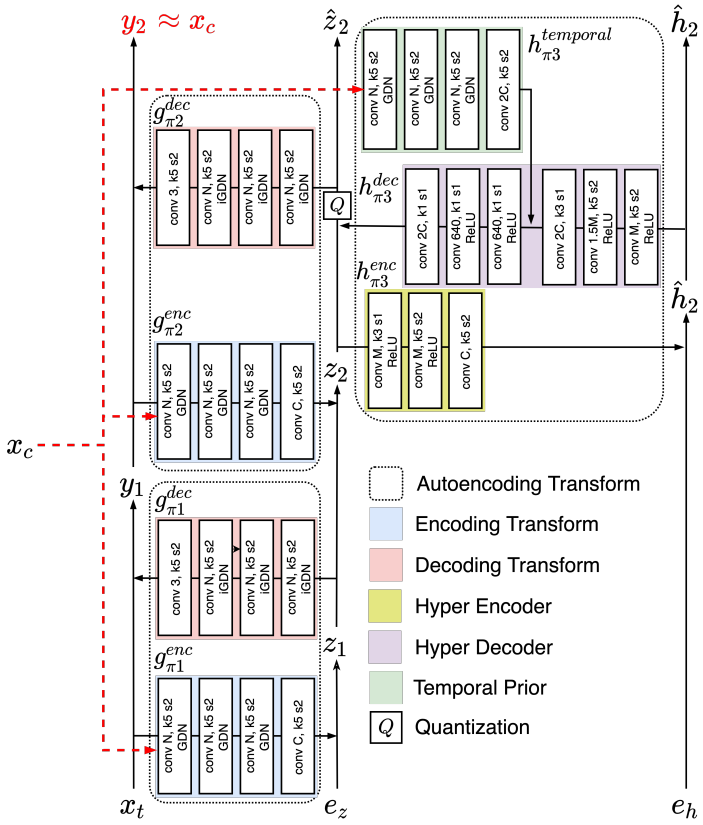


Fig. A3: Network details of our CANF-based coder.

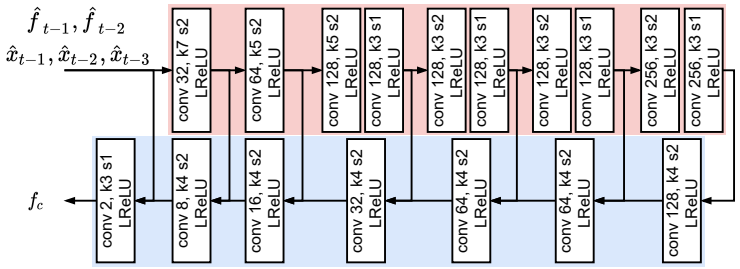


Fig. A4: Network details of our U-Net-based motion extrapolation network.

motion extrapolation networks, in order to pre-train these networks. We then freeze them until the last three steps.

In the second (2-frame training) phase, we train the P-frame coder by encoding one P-frame with its reference frame being an uncompressed I-frame. In

Table A4: Comparison of different temporal priors for the motion coder.

Cond. Variable of Temporal Prior	BD-Rate (%)
$warp(\hat{x}_{t-1}; f_c)$	-42.5%
f_c	-34.4%

Table A5: Details of our training strategy.

Phase	Training Parts	Loss	lr	Epochs
Pre-training	motion estimation and motion extrapolation networks	D		
2-frame (IP) training	motion coder	D+λR	1e-4	10
	motion coder and motion compensation network	D+λR	1e-4	10
	Inter-frame coder	D+λR	1e-4	8
	motion coder, motion compensation network, and inter-frame coder	D+λR	1e-4	5
5-frame (IPPPP) training	motion coder, motion compensation network, and inter-frame coder	D+λR	1e-4	5
	motion coder, motion compensation network, and inter-frame coder	D+λR	5e-5	5
	All networks	D+λR	2.5e-5	5
	All networks	D+λR	5e-5	1
	All networks	D+λR	2.5e-5	1

this phase, the uncompressed frames are used as inputs to the motion estimation and the motion extrapolation networks. We first train the motion coder and the motion compensation network. Subsequently, when the inter-frame coder is involved for training, we fix the motion coder and the motion compensation network for 8 epochs, followed by training jointly the inter-frame and the motion coders for another 5 epochs.

In the third (5-frame training) phase, we use 5 frames (IPPPP) as a basic training unit for forward propagation. However, in updating the P-frame coder, we stop the gradient at each reference frame so that the gradient will not back-propagate through reference frames. In this phase, the previously compressed frames are used as the reference frames (including I-frames) and are input to the motion estimation and the motion extrapolation networks. We train the inter-frame coder, the motion coder, and the motion compensation network for 10 epochs. Lastly, we fine-tune all the networks, including the motion estimation and the motion extrapolation networks, for another 7 epochs with learning rate decay.

8 Subjective Quality Comparison

Fig. A5 provides more subjective quality comparisons between CANF-VC and DCVC (ANFIC). Results are provided for models trained with PSNR and MS-SSIM. It is seen that our CANF-VC better preserves the shape of the objects than DCVC (ANFIC) (cf. the face of the jockey in the first row, the hair in the

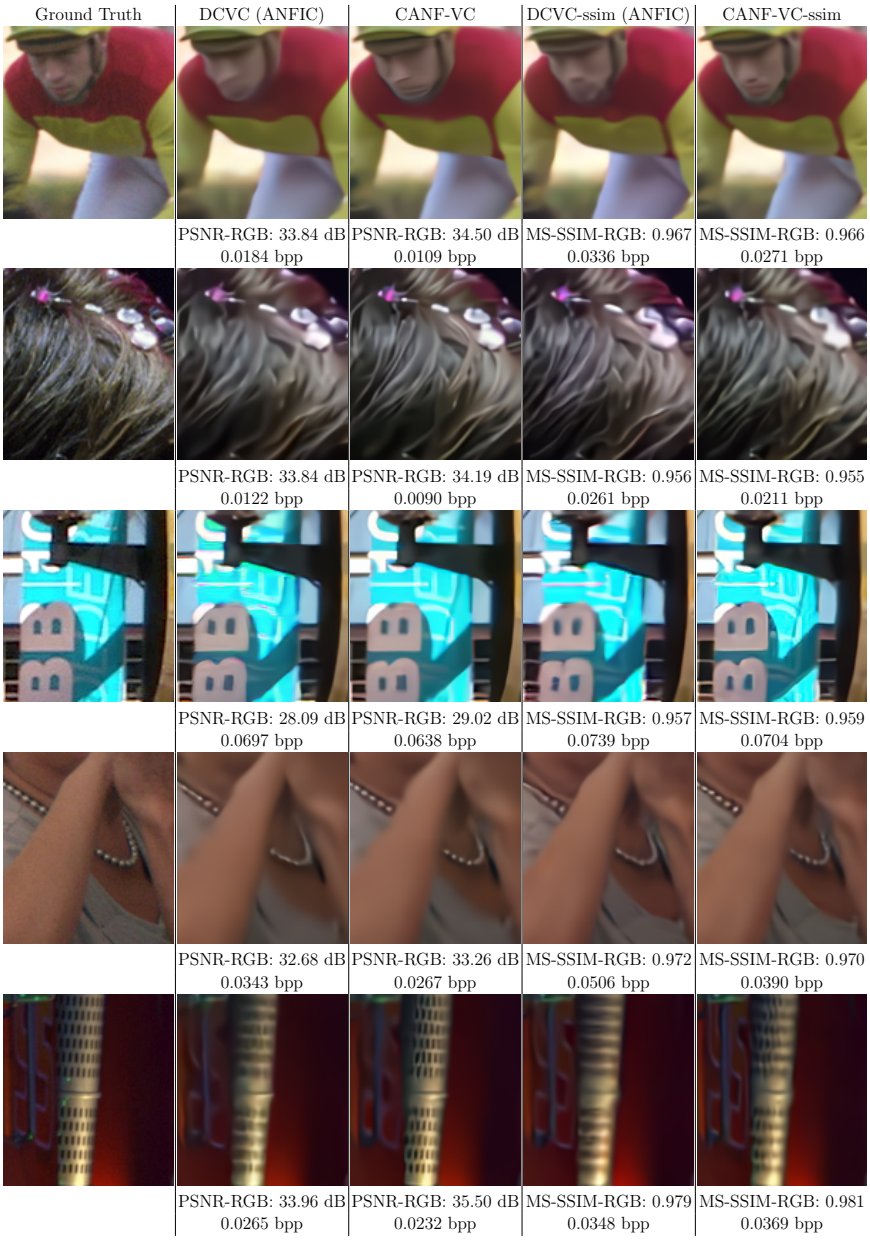


Fig. A5: Subjective quality comparison between CANF-VC and DCVC (ANFIC). The suffix "-ssim" indicates that the models are trained with MS-SSIM.

second row, the letters "DE" in the third row, the necklace in the forth row, and the textured pattern in the last row).

9 Command Lines for X265 and HM

Following [5], we use FFmpeg to generate the compressed videos through x265 with veryslow mode. Given an uncompressed video "input.yuv" of size W × H, the command line for x265 encoding is as follows: *ffmpeg -pix fmt yuv420p -s W ×H -r FR -i input.yuv -vframes N -c:v libx265 -preset veryslow -tune zerolatency -x265-params "qp=Q:keyint=GOP:verbose=1" output.mkv*, where FR, N, Q, GOP represent the frame rate, the number of frames to be encoded, the quantization parameter and the GOP size, respectively. Q is set to 19, 22, 27, 32, 37. For the common test protocol, we choose GOP to be 10 for HEVC Class B and 12 for the other datasets.

For the encoding with HM, given an uncompressed video "input.yuv" of size W × H, we use the *encoder_lowdelay_P_main.cfg* configuration file [1] with the following parameters: InputFile=input.yuv, FrameRate=FR, SourceWidth=W, SourceHeight=H, FramesToBeEncoded=N, IntraPeriod=32, GOPSize=8, DecodingRefreshType=2, and QP=Q, where FR, N, Q represent the frame rate, the number of frames to be encoded, and the quantization parameter, respectively. Q is set to 17, 22, 24, 27, 32.

References

1. Hm reference software for hevc. https://vcgit.hhi.fraunhofer.de/Zhu/HM/-/blob/HM-16.22/cfg/encoder_lowdelay_P_main.cfg, accessed: 2022-03-10
2. Ballé, J., Minnen, D., Singh, S., Hwang, S.J., Johnston, N.: Variational image compression with a scale hyperprior. In: International Conference on Learning Representations (2018)
3. Ho, Y.H., Chan, C.C., Peng, W.H., Hang, H.M., Domański, M.: Anfic: Image compression using augmented normalizing flows. IEEE Open Journal of Circuits and Systems **2**, 613–626 (2021)
4. Hu, Z., Lu, G., Xu, D.: Fvc: A new framework towards deep video compression in feature space. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1502–1511 (2021)
5. Li, J., Li, B., Lu, Y.: Deep contextual video compression. Advances in Neural Information Processing Systems (2021)
6. Lin, J., Liu, D., Li, H., Wu, F.: M-lvc: multiple frames prediction for learned video compression. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3546–3554 (2020)
7. Lu, G., Ouyang, W., Xu, D., Zhang, X., Cai, C., Gao, Z.: Dvc: An end-to-end deep video compression framework. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11006–11015 (2019)
8. Lu, G., Zhang, X., Ouyang, W., Chen, L., Gao, Z., Xu, D.: An end-to-end learning framework for video compression. IEEE transactions on Pattern Analysis and Machine Intelligence (2020)
9. Mercat, A., Viitanen, M., Vanne, J.: Uvg dataset: 50/120fps 4k sequences for video codec analysis and development. In: Proceedings of the 11th ACM Multimedia Systems Conference. pp. 297–302 (2020)

10. Rippel, O., Anderson, A.G., Tatwawadi, K., Nair, S., Lytle, C., Bourdev, L.: Elf-vc: Efficient learned flexible-rate video coding. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 14479–14488 (October 2021)
11. Sullivan, G.J., Ohm, J.R., Han, W.J., Wiegand, T.: Overview of the high efficiency video coding (hevc) standard. IEEE Transactions on circuits and systems for video technology **22**(12), 1649–1668 (2012)
12. Wang, H., Gan, W., Hu, S., Lin, J.Y., Jin, L., Song, L., Wang, P., Katsavounidis, I., Aaron, A., Kuo, C.C.J.: Mcl-jcv: a jnd-based h. 264/avc video quality assessment dataset. In: 2016 IEEE International Conference on Image Processing (ICIP). pp. 1509–1513. IEEE (2016)