Leveraging Bitstream Metadata for Fast, Accurate, Generalized Compressed Video Quality Enhancement: Appendices

A. Selection of Compared Methods and HEVC Comparison

Our paper presents a new and significantly more realistic benchmark than prior works. This required a significant time investment to identify working code releases and retrain compared methods. Some readers may take issue with our particular selection of comparison methods in the body of the paper given that newer methods than those we compared to exist (although we are "current" up until last year at the time of writing), so we address this up-front here. When selecting comparison methods, we divided them into three groups:

Methods with readily available code MFQEv2 (2021), STDF (2020)

Methods with partial code RFDA (2021)

Methods with no code All newer methods (Xu *et al*. (2021), PSTQE (2021)) and MFQEv1 (2018)

The goal of this exercise is to prioritize which methods we can reasonably compare to. Methods with readily available code can be re-trained and evaluated in a pain-free manner. Methods with partial code required some time investment to correct the code. Methods with no code would require re-implementation which is both time consuming for us and tricky to do in a fair way to the original authors. Optimizing this time-effort-fairness objective precluded us from re-evaluating the methods with no public code.

We note that RFDA has a partial public code release which is not fully functional particularly for retraining (we encourage readers to verify this by visiting the code at `https://github.com/zhaominyiz/RFDA-PyTorch`). However, we were able to make the training code functional by filling in details from the paper or correcting the existing code. This was verified by reproducing the paper results to within rounding error reported in the paper. We also note that the PSTQE authors have released the model definition but this alone is not enough to completely train the model even including any training details described in the paper. We opted not to include PSTQE since the training procedure could not be guaranteed to be fair and it would require a significant engineering effort on our part.

In the spirit of further fair comparisons we now show HEVC constant QP results compared to all prior art known to us at the time of writing (we do not claim to have an exhaustive list, nor do we believe a large table necessary to demonstrate that our method works). Since this is the less realistic benchmark which was reported in previous papers we do not have to retrain and we can simply copy the numbers into Table 1 allowing for these additional comparisons. To produce our numbers, we followed the same procedure prescribed in prior works by training a separate model per CQP. In this setting, only the recent work by Xu *et al*. [1] is really competitive with our method albeit with the IQE module which adds significant computational overhead. Given that we cannot perform independent testing of their method, it is unclear if that would generalize to more realistic compression benchmarks. It is also unclear to us if our QP cross attention mechanism is contributing in a constructive way to the performance of our model when the QP map is constant for all frames. We respectfully remind the skeptical reader that benchmark results are one aspect of our overall work and are not, were never intended to be, and should not be, a primary contribution.

B. GAN Architecture Details

Chu *et al*. [9] introduce a temporally consistent formulation for video GAN discriminators. We adapt their idea in our GAN loss which is otherwise based on DCGAN [10]. The architecture is shown in Figure 1.

Our critic operates on triplets of the compressed frames C, network outputs O, and target frames T. The input frames are stacked channelwise, in other words $C_{0,\mathrm{R}}, C_{0,\mathrm{G}}, C_{0,\mathrm{B}}, ...C_{6,\mathrm{R}}, C_{6,\mathrm{G}}, C_{6,\mathrm{B}}$ for all frames in the GOP are stacked channel-wise along with the corresponding O or T frames to create a 42 channel input (3 channels per frame, times 7 frames, times 2). The task for the critic, then, is to determine whether the T or O channels are network outputs or uncompressed frames given the compressed reference frames and otherwise operates as standard DCGAN. The output is used in a Wassertein GAN loss [11].

This encourages temporal consistency because the critic is judging the entire 7-frame sequence as an example as opposed to other video GANs which treat each frame as an different example. In the later case, there may be situations in which one frame is judged to be more real than another, and yet all frames are equally "real" or "fake". This "reallness" or "fakeness" of the sequence *vs*. the frames is better captured by the formulation of Chu *et al*. [9]. Note that for our task we define "fake" as the restored network output and "real" as the uncompressed version of the image.

C. Compression Details

The MFQE [2] dataset is stored as a series of uncompressed raw (`.yuv`) videos of known resolution and frame count. We compress these videos using ffmpeg [12]. For H.264, we use the libx264 encoder with the following command:

Table 1. **HEVC Comparison**. We report ΔPSNR (dB) ↑ / ΔSSIM ↑, averaged over the MFQE [2] test split. This matches exactly with prior works. Note that STDF does not test on QP42. Best in **bold**, second best underlined.

Method	HEVC CQP				
	22	27	32	37	42
MFQE 1.0 [2]	0.31 / 0.0019	0.40 / 0.0034	0.43 / 0.0058	0.46 / 0.0088	0.44 / 0.0130
MFQE 2.0 [3]	0.46 / 0.0027	0.49 / 0.0042	0.52 / 0.0068	0.56 / 0.0109	0.59 / 0.0165
STDF-R1 [4]	0.51 / 0.0027	0.59 / 0.0047	0.64 / 0.0077	0.65 / 0.0118	-
STDF-R3 [4]	0.63 / 0.0034	0.72 / 0.0057	0.86 / 0.0104	0.83 / 0.0151	-
RFDA [5]	0.76 / 0.0042	0.82 / 0.0068	0.87 / 0.0107	0.91 / 0.0162	0.82 / 0.0220
PSTQE [6]	0.55 / 0.0029	0.63 / 0.0052	0.67 / 0.0083	0.69 / 0.0125	0.69 / 0.0186
Xu *et al*.-SQE [1]	0.83 / 0.0046	0.92 / 0.0077	0.93 / 0.0116	0.85 / 0.0158	0.79 / 0.0218
S2SVR [7]	-	-	-	0.93 / 0.0176	-
Xu *et al*.-IQE [1]	0.96 / 0.0053	**1.09** / 0.0092	**1.08 / 0.0136**	**1.03** / 0.0190	**0.89** / 0.0241
BasicVSR++ [8]	0.90 / 0.0050	1.04 / 0.0091	1.06 / 0.0128	0.99 / 0.0178	-
MetaBit (Ours)	**1.20 / 0.0109**	0.99 / **0.0117**	0.99 / 0.0131	0.90 / **0.0232**	0.84 / **0.0294**

```
ffmpeg ffmpeg -video_size <WIDTH>x<HEIGHT> \
    -framerate 10 \
    -i <INPUT> \
    -preset medium \
    -vcodec libx264 \
    -crf <CRF> \
    -x264opts <OPTIONS>\
     [OUTPUT]
```

where `<OPTIONS>` is defined as:

```
keyint=7:min-keyint=7:no-scenecut:no-fast-pskip:me=esa:subme=7:bframes=0:aq-mode=2
```

This produces a compressed `.mp4` file for later use (to be read as-is or converted back to raw `.yuv` for compatibility with prior work). Many of these options are simply there to ensure a 7 frame GOP which is a requirement of the model we presented in the body of the paper (but **not** a requirement of the general method which we presented). Please note the `-framerate 10` argument: CRF is sensitive to framerate, so different framerates will incur different degradation strengths. Our choice of 10 was arbitrary and motivated primary by recommendation of Li *et al*. [13].

For H.264 in "streaming mode" (Appendix D) we use the following command in order to be consistent with deep-learning-based compression works:

```
ffmpeg -video_size <WIDTH>x<HEIGHT> \
    -framerate 30 \
    -i <INPUT> \
    -crf <CRF> \
    -preset medium \
    -vcodec libx264 \
    -pix_fmt yuv420p \
    -x264opts keyint_min=10000:bframes=0 \
    <OUTPUT>
```

Here, `keyint_min=10000` ensures that there is only a single I-frame per video.

For H.265, prior works evaluated on the HM reference codec, which is notoriously slow. For any models which we retrained on H.265 data, we instead use libx265 which incurs no appreciable change in degradation (Figure 2) and is significantly faster than HM. To generate these videos we used the following command:

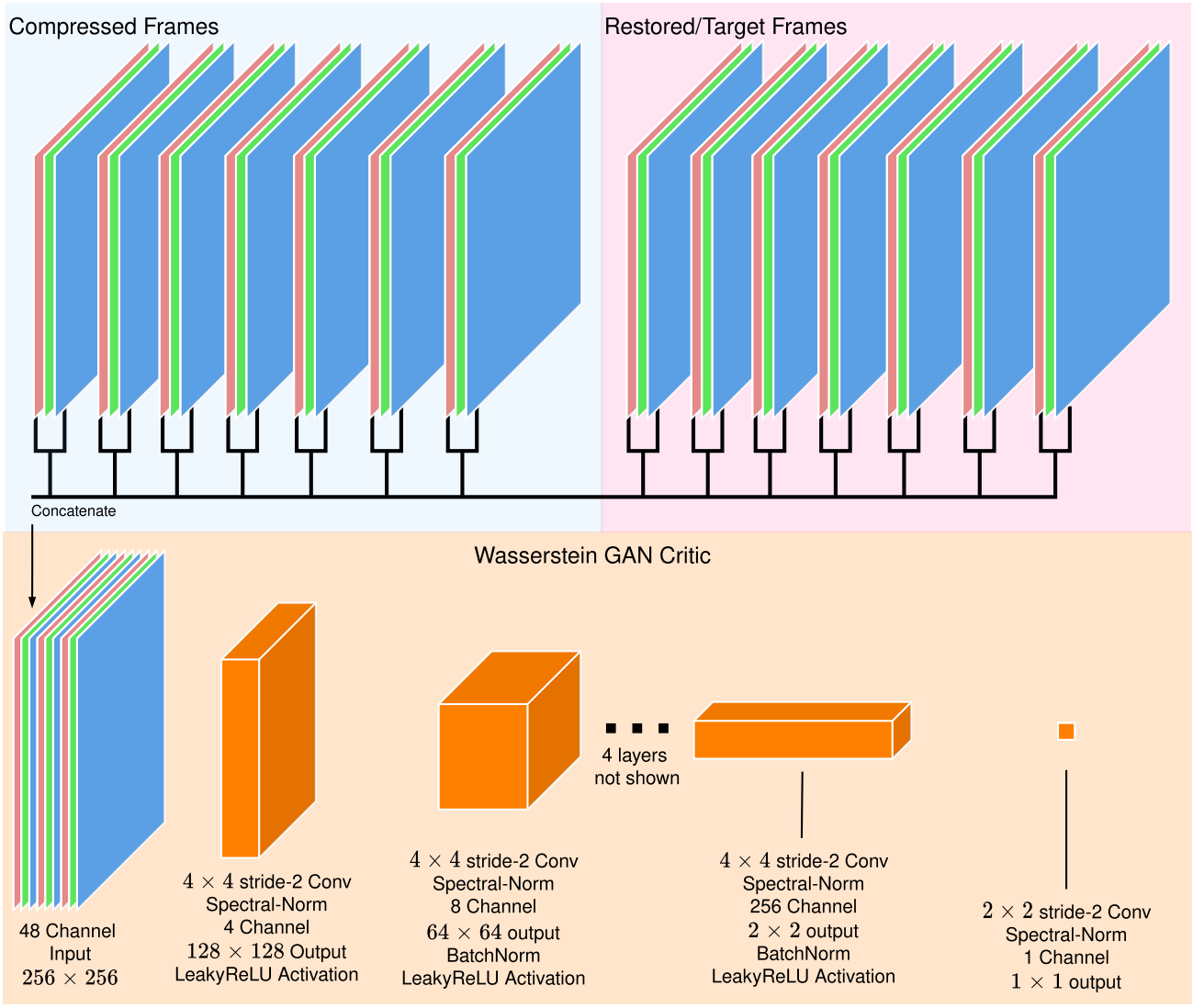

Figure 1. **Wasserstein GAN Critic.** Inputs (top) are RGB channels for all 7 frames stacked in the channel dimension. The compressed frames are concatenated with the restored or target frames for a 48 channel input. This is then processed by 8 downsampling convolutional layers as in DCGAN to produce the critic output.

```
ffmpeg -video_size <WIDTH>x<HEIGHT> \
    -i <INPUT> \
    -vcodec libx265 \
    -qp <QP> \
    -x265-params <OPTIONS> \
     <OUTPUT>
```

for `<OPTIONS>`:

```
keyint=7:min-keyint=7:no-scenecut:me=full:subme=7:bframes=0:qp=<QP>
```

Note that QP is specified twice and there is no longer a need to control for framerate. We strongly recommend that future works use libx265.

Figure 2. **Codec Comparison.** HM reference encoder *vs*. libx265 at QP37. While the artifacts produced by the different encoders are different, the overall perceptual quality is similar. This is likely because of CQP encoding which does not leave many options for the encoder to make intelligent rate-distortion decisions.

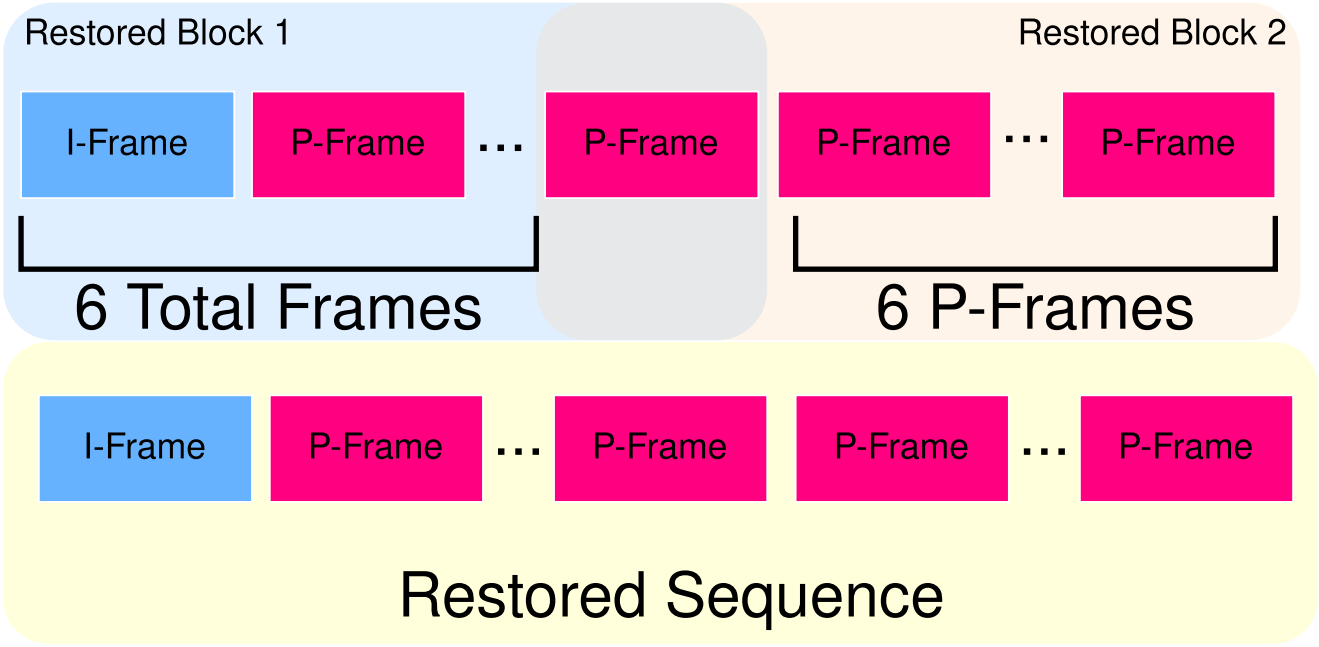

Figure 3. **Streaming Mode.** In streaming mode, the last restored P-frame in each block is cached and used instead of an I-frame for the next block.

D. Streaming Mode

In many streaming applications, only a single I-frame is ever transmitted, with every subsequent frame stored as a P-frame. This saves bandwidth and decreases latency at the expense of quality. The model as described in the body of the paper requires a 7-frame GOP with a single I-frame followed by 6 P-frames, however, our model can be easily modified to operate in what we call "streaming mode".

In streaming mode (Figure 3), our method performs a 7-frame restoration as usual on the first 7 frames which include the I-frame. The last restored P-frame (frame 7) is cached and used in place of the I-frame for the next 6 P-frames. This process is repeated for all subsequent groups of 6 P-frames, eliminating the need for a periodic high-information frame. This comes at a small cost to restoration performance however it also greatly reduces the bitrate of the H.264 videos. This mode was used to produce the rate-distortion comparisons to deep-learning based compression which requires aggressive settings to match the

bitrates reported in the compared works. Adding periodic I-frame incurs a large increase in bitrate. Note that this does not require retraining the model, it is only a change to the inference procedure.

E. Additional Qualitative Results

In this section we show additional qualitative results. These results are intended to showcase particular strengths and weaknesses we observed in our model, and are explained further in the figure captions. We encourage viewing the video files contained in our supplementary material to observe temporal consistency issues we noticed due to fluctuating compression artifacts.

1. **Dark Region** Figure 4 highlights a known failure mode of compression causing additional information loss in dark areas in an image.

2. **Crowd** Figure 5 shows our model performance on a dense crowd

3. **Texture Restoration** Figure 6 shows an additional result of our model generating a plausible reconstruction of an artificial texture

4. **Compression Artifacts** Figure 7; one particular failure mode we observed was compression artifacts, particularly chroma subsampling artifacts, mistaken as a degraded texture and restored by the GAN, creating texture where none exists in the original images

5. **Motion Blur** Figure 8. Another common occurrence is missing motion blur in reconstructed images. There are several issues that lead to this: 1) high motion frames are largely absent from the training data, 2) motion blur is largely destroyed by compression, and 3) the reconstruction loss is explicitly rewarded for generating sharp restorations, whereas in this case we actually *want* a blurry reconstruction.

6. **Artificial** Figure 9; in this scene from the short film "Big Buck Bunny", the frame is restored quite accurately despite a lack of artificial training data.

Figure 4. **Dark Region.** Crop from 2560×1600 "People on Street". The dark region, is poorly preserved by compression. Our GAN restoration struggles to cope with the massive information loss in this region.

Figure 5. **Crowd.** Crop from 2560×1600 "People on Street". The image shows an extremely dense crowd. Despite the chaotic nature, our GAN is able to produce a good restoration although there is detail missing.

Figure 6. **Texture Restoration.** Crop from 1920×1080 "Cactus". The texture on the background is destroyed by compression. Our GAN reconstructs a reasonable approximation to the true texture.

Figure 7. **Compression Artifacts Mistaken for Texture.** Crop from 1920×1080 "Cactus". The compressed image exhibits strong chroma subsampling artifacts (lower right corner). These are mistaken by the GAN is a texture and restored as such.

Figure 8. **Motion Blur.** Crop from 1920×1080 "Cactus". The tiger exhibits high motion which presents itself in the target frame as motion blur. This blur is destroyed by compression and is not able to be restored by the GAN loss. The GAN loss is also "rewarded" for sharp edges which would make reconstructing blurry objects difficult. As an aside, note the additional detail on the background objects in the GAN image when compared to the compressed image.

Figure 9. **Artificial.** Crop from 1920×1080 "Big Buck Bunny". This artificial scene is restored accurately despite a lack of artificial training data. Note the grass and tree textures, sharp edges, removal of blocking on the flower, and preservation of the smooth sky region.

References

[1] Y. Xu, M. Zhao, J. Liu, *et al.*, "Boosting the performance of video compression artifact reduction with reference frame proposals and frequency domain information," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 213–222.

[2] R. Yang, M. Xu, Z. Wang, and T. Li, "Multi-frame quality enhancement for compressed video," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, Jun. 2018, pp. 6664–6673, ISBN: 978-1-5386-6420-9. DOI: 10.1109/CVPR.2018.00697. [Online]. Available: https://ieeexplore.ieee.org/document/8578795/.

[3] Q. Xing, Z. Guan, M. Xu, R. Yang, T. Liu, and Z. Wang, "Mfqe 2.0: A new approach for multi-frame quality enhancement on compressed video," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 3, pp. 949–963, Mar. 2021, arXiv: 1902.09707, ISSN: 0162-8828, 2160-9292, 1939-3539. DOI: 10.1109/TPAMI.2019.2944806.

[4] J. Deng, L. Wang, S. Pu, and C. Zhuo, "Spatio-temporal deformable convolution for compressed video quality enhancement," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 0707, pp. 10 696–10 703, Apr. 2020, ISSN: 2374-3468. DOI: 10.1609/aaai.v34i07.6697.

[5] M. Zhao, Y. Xu, and S. Zhou, "Recursive fusion and deformable spatiotemporal attention for video compression artifact reduction," in *Proceedings of the 29th ACM International Conference on Multimedia*, 2021, pp. 5646–5654.

[6] Q. Ding, L. Shen, L. Yu, H. Yang, and M. Xu, "Patch-wise spatial-temporal quality enhancement for hevc compressed video," *IEEE Transactions on Image Processing*, vol. 30, pp. 6459–6472, 2021.

[7] J. Lin, X. Hu, Y. Cai, *et al.*, "Unsupervised flow-aligned sequence-to-sequence learning for video restoration," in *International Conference on Machine Learning*, PMLR, 2022, pp. 13 394–13 404.

[8] K. C. Chan, S. Zhou, X. Xu, and C. C. Loy, "Basicvsr++: Improving video super-resolution with enhanced propagation and alignment," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 5972–5981.

[9] M. Chu, Y. Xie, J. Mayer, L. Leal-Taixé, and N. Thuerey, "Learning temporal coherence via self-supervision for gan-based video generation," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 4, pp. 75–1, 2020.

[10] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[11] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning*, PMLR, 2017, pp. 214–223.

[12] S. Tomar, "Converting video formats with ffmpeg," *Linux Journal*, vol. 2006, no. 146, p. 10, 2006.

[13] Y. Li, P. Jin, F. Yang, C. Liu, M.-H. Yang, and P. Milanfar, "Comisr: Compression-informed video super-resolution," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2021, pp. 2543–2552.