Combining Frame and GOP Embeddings for Neural Video Representation: Supplementary Materials

Jens Eirik Saethre[1,2] Roberto Azevedo[1] Christopher Schroers[1]

[1]DisneyResearch|Studios, Switzerland
[2]ETH Zürich, Switzerland

jens.eirik@saethre.ch {roberto.azevedo,christopher.schroers}@disneyresearch.com

A. Additional Results

In this section, we present additional results for the evaluation of our proposed T-NeRV architecture. In particular, we show results from the main paper in terms of the MS-SSIM metric (Appendix A.1), rate-distortion curves for the individual UVG [6] sequences (Appendix A.2), Bjøntegaard statistics (Appendix A.3), video representation results on additional datasets (Appendix A.4), encoding and decoding times (Appendix A.5), qualitative results (Appendix A.6), and additional results on the byte allocation of compressed T-NeRV models (Appendix A.7).

A.1. MS-SSIM Results

Video Representation. Tab. 1 lists the video representation results in the two described settings in terms of the MS-SSIM metric, analogously to Tab. 2 in the main paper. The MS-SSIM results paint the same picture: T-NeRV achieves state-of-the-art results in both settings, with the most prominent improvements observed in dynamic videos. In particular, we want to highlight the MS-SSIM increase by 0.0423 on the *ReadySetGo* sequence in the second setting.

Ablations. Tab. 2 shows the ablation study with respect to the MS-SSIM metric. On the encoder side, the ablated models use only embeddings (E1), only feature grids (E2), no temporal augmentation (E3), replace T-NeRV's ConvNeXt block with HNeRV's tiny one (E4), and use HNeRV's tiny ConvNexT block while disabling temporal augmentation (E5). On the decoder side, we employ no flow-guided decoder (D1), no AdaIN layers in the T-NeRV blocks (D2), and regular 5×5 convolutions (D3). Again, the MS-SSIM results show the same trend as the PSNR results presented in Tab. 3 in the main paper, underlining the importance of each proposed component of the T-NeRV architecture.

Rate-Distortion Curve. The rate-distortion curve on the UVG dataset in terms of the MS-SSIM metric is depicted in Fig. 1. Similar to Fig. 5 from the main paper, it shows that T-NeRV outperforms all baselines across the entire BPP spectrum for this metric, too. Unfortunately, the authors of FFNeRV did not collect any MS-SSIM metrics, hence the omission from this plot.

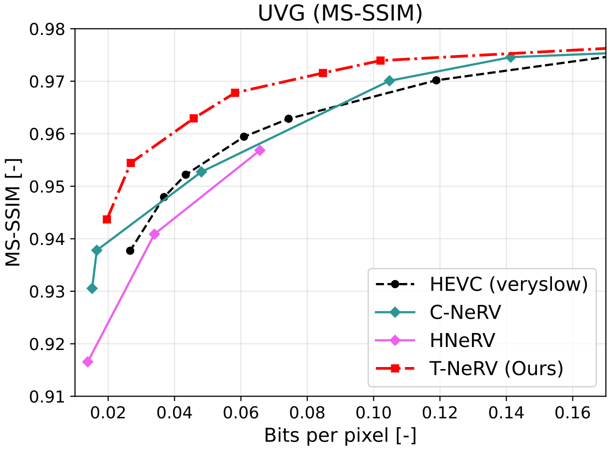


Figure 1. Rate-Distortion performance of T-NeRV on UVG in terms of MS-SSIM. Our method outperforms HEVC and the previous state-of-the-art video INRs across the entire BPP spectrum.

A.2. Per-Video Rate-Distortion Curves

We complement the rate-distortion analysis from the main paper (Fig. 5) with rate-distortion curves for the seven UVG videos in Fig. 2. As indicated, T-NeRV is the first video INR that outperforms HEVC (x265 with `veryslow` preset, no b-frames) on all video sequences. Similar to the video representation setting, the largest improvements in visual quality over previous INR-based methods can be observed in high-motion sequences, such as *Bosphorus*, *Jockey*, *ReadySetGo*, or *YachtRide*. On static videos, such as *Beauty*, *HoneyBee* and *ShakeNDRy*, T-NeRV can match and sometimes slightly outperform existing video INRs, which

Model	Size	Resolution	Beauty	Bospho.	Honey.	Jockey	Ready.	Shake.	Yacht.	Avg.
NeRV [1]	12.6M	(720, 1280)	0.9527	0.9839	0.9941	0.9681	0.9594	0.9799	0.9579	0.9709
E-NeRV [5]	12.5M	(720, 1280)	0.9523	0.9898	0.9945	0.9721	0.9730	0.9873	0.9675	0.9766
C-NeRV [3]	12.5M	(720, 1280)	0.9523	0.9901	0.9944	0.9713	0.9709	0.9877	0.9680	0.9764
FFNeRV [4]	12.4M	(720, 1280)	0.9580	0.9900	**0.9947**	0.9785	0.9793	0.9863	0.9747	0.9802
HNeRV [2]	12.4M	(720, 1280)	0.8677	0.8366	0.9930	0.8554	0.9598	0.8787	0.8059	0.8853
DNeRV [9]	12.4M	(720, 1280)	0.9487	0.9742	0.9943	0.9556	0.9510	0.9848	0.9480	0.9652
T-NeRV (Ours)	12.4M	(720, 1280)	**0.9583**	**0.9912**	0.9946	**0.9830**	**0.9886**	**0.9886**	**0.9789**	**0.9833**
NeRV [1]	3.0M	(960, 1920)	0.8947	0.9411	0.9830	0.9040	0.8519	0.9348	0.8872	0.9138
E-NeRV [5]	3.0M	(960, 1920)	0.8951	0.9423	0.9841	0.8756	0.8417	0.9464	0.8942	0.9113
C-NeRV [3]	3.0M	(960, 1920)	0.8951	0.9434	0.9840	0.8735	0.8435	0.9433	0.8930	0.9108
FFNeRV [4]	3.0M	(960, 1920)	0.8977	0.9377	0.9841	0.9210	0.8881	0.9450	0.8963	0.9243
HNeRV [2]	3.0M	(960, 1920)	0.8949	0.9457	0.9844	0.8812	0.8444	0.9449	0.8917	0.9125
DNeRV [9]	3.0M	(960, 1920)	0.8938	0.9452	0.9844	0.8851	0.8526	0.9449	0.8866	0.9132
T-NeRV (Ours)	3.0M	(960, 1920)	**0.9160**	**0.9589**	**0.9887**	**0.9422**	**0.9304**	**0.9584**	**0.9139**	**0.9441**

Table 1. Video regression results in terms of MS-SSIM on UVG [6] in two settings proposed in [1] and [2]. Best results are marked in bold, second-best results are underlined. T-NeRV significantly outperforms all previous methods in both settings.

Model	Beauty	Bospho.	Honey.	Jockey	Ready.	Shake.	Yacht.	Avg.
T-NeRV	0.9464	**0.9816**	**0.9942**	**0.9752**	**0.9751**	0.9730	0.9616	**0.9724**
(E1)	**0.9469**	0.9789	0.9941	0.9737	0.9740	**0.9736**	0.9613	0.9718
(E2)	0.9468	0.9810	**0.9942**	0.9682	0.9693	0.9723	**0.9620**	0.9705
(E3)	**0.9469**	0.9791	**0.9942**	0.9708	0.9711	0.9731	0.9594	0.9707
(E4)	**0.9469**	0.9788	**0.9942**	0.9699	0.9710	0.9729	0.9591	0.9704
(E5)	0.9456	0.9780	**0.9942**	0.9678	0.9683	0.9717	0.9549	0.9686
(D1)	0.9466	0.9804	0.9941	0.9717	0.9713	0.9716	0.9541	0.9700
(D2)	0.9411	0.9527	0.9918	0.9702	0.9694	0.9629	0.9521	0.9629
(D3)	0.9467	0.9787	**0.9942**	0.9709	0.9709	0.9733	0.9592	0.9706

Table 2. Ablation study of T-NeRV in terms of MS-SSIM on the first 300 frames of UVG sequences. Best results are marked in bold.

Metric	x265 (veryslow)	C-NeRV	FFNeRV	HNeRV
BD-Rate	-39.09%	-31.62%	-21.87%	-24.15%
BD-PSNR	0.98	0.73	0.45	0.64

Table 3. Bjøntegaard statistics for T-NeRV.

Model	DAVIS			MCL-JCV		
	Size	PSNR	MS-SSIM	Size	PSNR	MS-SSIM
NeRV	2M	26.33	0.8506	5M	34.88	0.9700
E-NeRV	2M	28.42	0.9081	5M	36.79	0.9809
C-NeRV	2M	28.47	0.9088	5M	36.84	0.9807
FFNeRV	2M	27.25	0.8824	5M	36.30	0.9791
HNeRV	2M	26.53	0.8076	5M	30.67	0.8870
DNeRV	2M	28.21	0.8939	5M	36.44	0.9779
T-NeRV (Ours)	2M	**30.81**	**0.9377**	5M	**38.12**	**0.9854**

Table 4. Video regression results on DAVIS [7] and MCL-JCV [8]. Best results are marked in bold, second-best results are underlined.

already drastically outperform the HEVC baseline. Overall, this demonstrates the versatility of our proposed T-NeRV architecture, achieving state-of-the-art results on videos ranging from static to highly dynamic.

A.3. Bjøntegaard Statistics

We present Bjøntegaard statistics for the video compression experiments on UVG in Tab. 3. Each entry hereby denotes the increase in BD-PSNR or the decrease in BD-Rate that T-NeRV achieves over the respective baseline.

A.4. Results on Additional Datasets

In line with previous works [1, 2, 4] we mainly evaluate our proposed method on the UVG dataset. However, UVG videos are shot at 120 fps, resulting in very smooth videos even for the sequences that exhibit the highest degrees of motion, such as e.g. *ReadySetGo*. This characteristic does not apply to a large fraction of videos that are encoded and decoded in the real world. Consequently, the relative performance of INR-based methods on UVG might not necessarily reflect their performance on more common videos that are less smooth. To that end, we evaluate T-NeRV and the six baselines on two additional datasets: DAVIS validation [7] and MCL-JCV [8]. The corresponding results in terms of both PSNR and MS-SSIM are listed in table Tab. 4.

DAVIS validation. The DAVIS validation set consists of 20 sequences at 1080p of different lengths. Recall that the model size of hybrid methods such as HNeRV and our proposed T-NeRV depends on the number of frames to be encoded. For simplicity, we restrict ourselves to a single configuration for T-NeRV and each of the six baselines, therefore extracting the first 60 frames of the 11 sequences that comprise at least 60 frames. We then train all models with

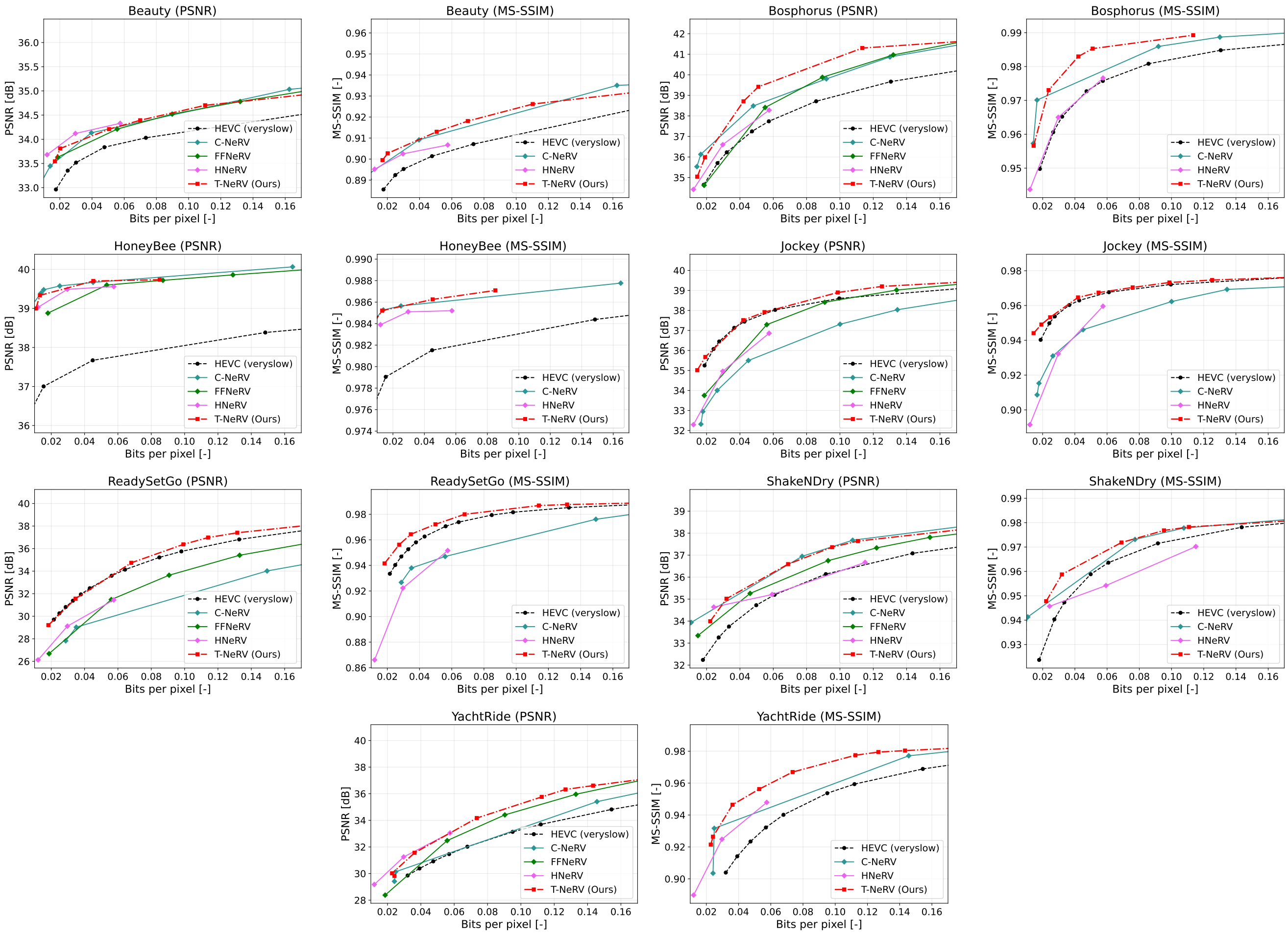


Figure 2. Rate-distortion curves of the seven UVG videos. This figure is best viewed digitally.

configurations of 2M parameters for 600 epochs. The results indicate that T-NeRV can handle non-smooth videos much better than all of the baselines, outperforming the second-best method by 2.34 dB of PSNR on average.

MCL-JCV. From the 30 videos in the MCL-JCV dataset, we select the 16 videos that were shot at 30 fps for a total of 150 frames per sequence at a resolution of 720p. We then employ configurations of 5M parameters for all models and train them for 300 epochs. Similarly to the DAVIS dataset, T-NeRV significantly outperforms all previous methods by a significant margin (1.28 dB of PSNR improvement over the second-best method). Interestingly, FFNeRV, arguably the second-best method on the UVG dataset, does not perform well on non-smooth videos. As discussed in Sec. 3.1 in the main paper, this can be attributed to FFNeRV's ill-suited dependence on the multi-resolution grid to extract frame-specific information. Recall that each feature in this grid is used in the forward pass of at least two frames due to the grid's interpolating behavior. While consecutive frames might be almost identical in smooth 120-fps sequences, they can differ significantly in less smooth videos, making it impossible for the multi-resolution grid to extract truly frame-specific information.

A.5. Encoding & Decoding Times

We list encoding and decoding times for the different methods on the DAVIS and MCL-JCV datasets in Tab. 5. Before discussing the results, we want to emphasize that decoding speed is arguably the more important metric as videos are encoded only once, often on specialized hardware, but decoded potentially millions of times on consumer devices. T-NeRV significantly improves on the decoding speeds of previous hybrid video INRs, decoding videos 2-3 times as fast as HNeRV and comfortably achieving real-time speeds. This is in contrast with neural video codecs (NVC) that often achieve less than 2 fps. While encoding is slower than previous approaches, we argue that this is warranted for the above reason as well as the significant increase in

Dataset	Resolution	Metric	NeRV	E-NeRV	C-NeRV	FFNeRV	HNeRV	DNeRV	T-NeRV
DAVIS	1080p	Enc. [min]	37	33	**31**	56	83	122	119
		Dec. [fps]	94.1	115.1	**117.1**	83.0	19.6	35.7	35.2
MCL-JCV	720p	Enc. [min]	**35**	43	39	46	97	117	103
		Dec. [fps]	96.3	107.6	**109.2**	95.4	17.5	22.3	48.6

Table 5. Encoding/Decoding speeds on an NVidia RTX 3090.

video quality. Moreover, T-NeRV outperforms all previous methods after significantly fewer epochs, e.g. by 0.33 dB of PSNR on DAVIS after only 50% of epochs.

A.6. Qualitative Results

We present qualitative results for the video compression task in Fig. 3, comparing patches of individual frames for three UVG sequences, where we employ FFNeRV as our baseline for a multitude of reasons. First, FFNeRV is arguably the best-performing of the previous video INRs, especially on challenging high-motion videos. Additionally, it covers a large BPP spectrum, and its implementation is available with detailed configurations for every data point.[1] We obtain FFNeRV frames by training the respective models on the reference implementation and then decoding the compressed frames.

Fig. 3 compares patches of three sequences that range from high-motion (*ReadySetGo*) to low-motion (*ShakeNDry*) with videos compressed at high bitrates (*ReadySetGo* and *ShakeNDry*) to very low bitrates (*Jockey*). We also vary the patch size from tiny ones (*ReadySetGo*) to very large ones (*Jockey*). In all of the shown patches, T-NeRV can extract details much more accurately than FFNeRV at the same or lower BPP values.

A.7. Compression as Fine-Tuning

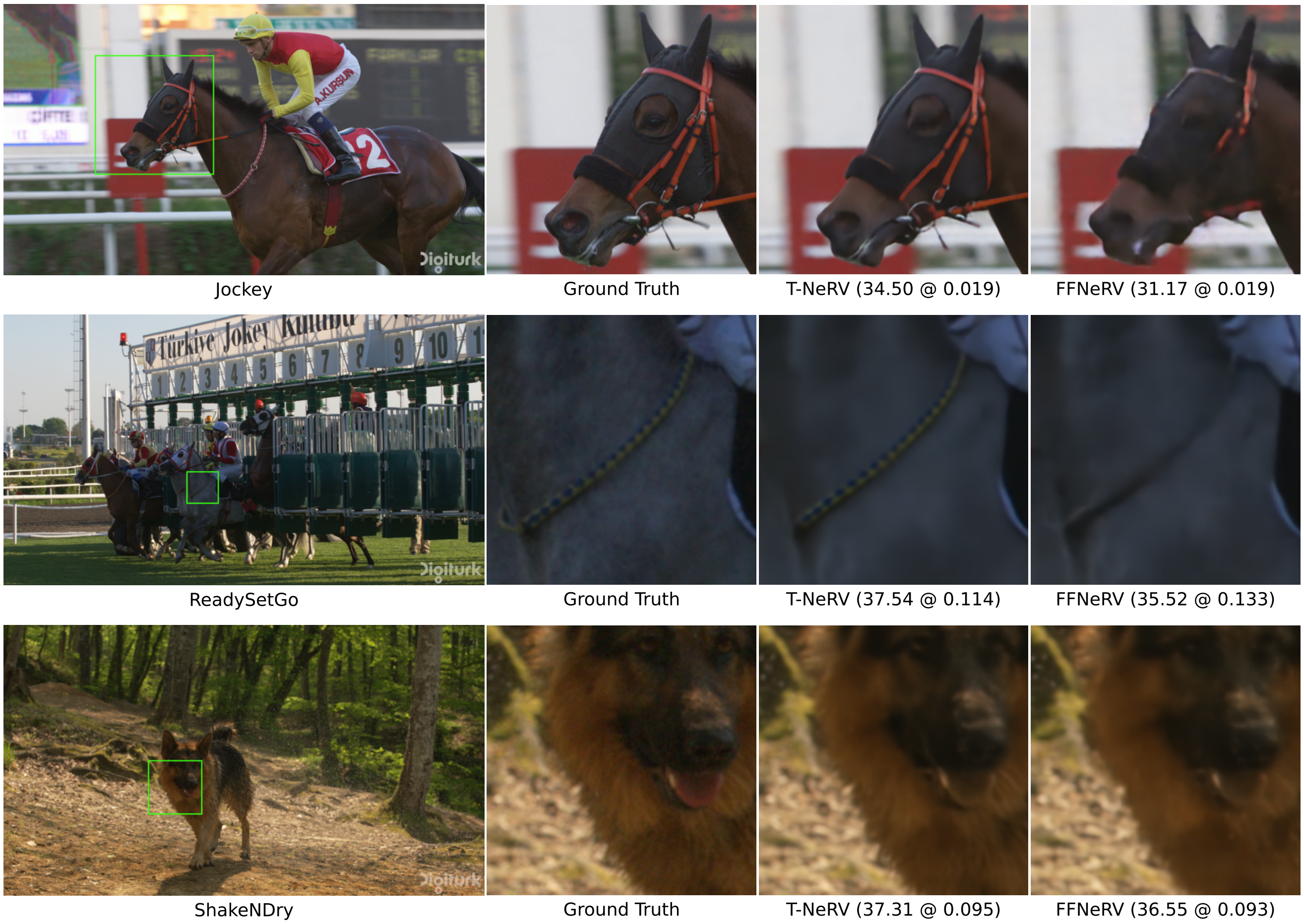
In the main paper, we showed how T-NeRV could automatically fine-tune itself with respect to the target content by allocating more or fewer bits to key components, such as the feature grids. To that end, we showed the byte distribution of six videos encoded with 9.3M parameter T-NeRV configurations and $\lambda = 1$ (cf. Fig. 6). The configuration for this model is listed in Tab. 8 and allocates approximately 48%, 27%, and 24% of the total number of parameters to the decoder, the feature grids, and the frame-specific embeddings, respectively. We expand on this analysis by showing the byte distribution at different compression rates that we obtain by varying $\lambda \in \{0.01, 0.1, 0.5, 1.0\}$ while keeping the model size constant.

Fig. 4 plots the byte distribution for all four λ values, with the legend shown on the right-most figure. The impact that the compression rate λ has on the byte distribution is apparent: At a low compression rate of $\lambda = 0.01$, the byte distribution is similar to the parameter allocation, except for the case of the very static *HoneyBee* sequence, where the small amount of GOP-specific information already leads to much fewer bits being spent on the feature grids. As we increase λ and compress more, the relative size of the decoder and, more significantly, the feature grids start to decrease while the relative size of the embeddings rises drastically. At $\lambda = 0.5$, the byte distributions start to reveal the nature of the encoded content, with models spending more bits on the decoder and the feature grids representing high-motion sequences, while static video sequences lead to distributions with fewer bits spent on the grids and the decoder. This effect is even more visible at $\lambda = 1$, where the relative size of both the feature grids and the decoder correlate almost perfectly with the degree of motion present in the video sequence. Additionally, we can also observe that the reduction in total model size from $\lambda = 0.01$ to $\lambda = 1$ is the largest for static sequences and the lowest for high-motion sequences, which is in line with our expectation that dynamic videos are more difficult to compress than static videos.

T-NeRV Configuration: Video Representation (12.4M)			
Feature Grids	Temporal Resolutions		$(64, 128, 256)$
	Feature Size		$(30, 9, 16)$
Frame Encoder	Embedding Size		$(35, 9, 16)$
	ConvNeXt	Strides	$(5, 2, 2, 2, 2)$
		Blocks	$(3, 3, 9, 6, 3)$
		Channels	$(3, 64, 128, 256, 512, 35)$
	Temporal Augmentation	Pos. Enc.	$(1.25, 80)$
		MLP Dims	$(160, 512, 512, 5040)$
		MLP Activation	GELU
Temp. MLP	Positional Encoding		$(1.25, 80)$
	MLP Dims		$(160, 256, 256, 128)$
Decoder	Channels	Pre-Block	$(125, 231)$
		T-NeRV Blocks	$(231, 192, 160, 134, 111, 93)$
		Head Layer (3)	$(134, 12)$
		Head Layer (5)	$(93, 5)$
	Upsampling Factors		$(5, 2, 2, 2, 2)$
	Aggregation Window		$\{-2, -1, 1, 2\}$

Table 6. T-NeRV configuration for video representation (12.4M).

[1] This is unlike HNeRV, which only covers the low BPP regime and does not provide architectural details for the individual data points.



Jockey

Ground Truth

T-NeRV (34.50 @ 0.019)

FFNeRV (31.17 @ 0.019)

ReadySetGo

Ground Truth

T-NeRV (37.54 @ 0.114)

FFNeRV (35.52 @ 0.133)

ShakeNDry

Ground Truth

T-NeRV (37.31 @ 0.095)

FFNeRV (36.55 @ 0.093)

Figure 3. Qualitative compression results comparing T-NeRV against FFNeRV and the ground truth. Frames are annotated with PSNR @ BPP, where the PSNR is computed for the corresponding frame. This figure is best viewed digitally.

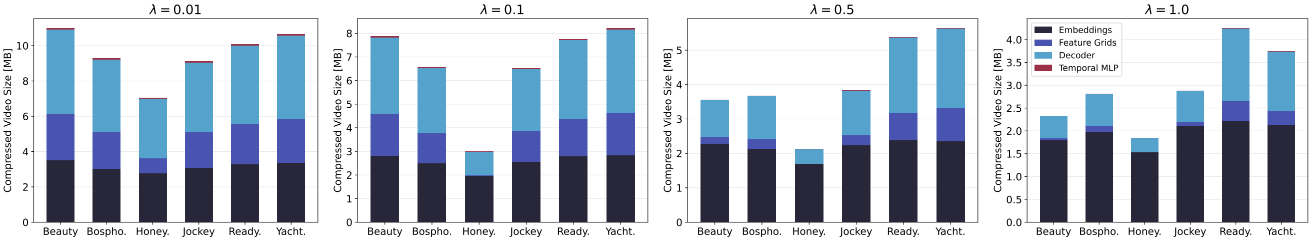


Figure 4. Byte distribution in videos represented by a T-NeRV model with 9.3M parameters that were compressed with different λ values. Higher values of λ amount to higher levels of compression. This figure is best viewed digitally.

B. Model Configurations

In this section, we list the configurations that were used for the video representation and compression tasks.

Video Representation. The T-NeRV configurations that were used for the two video representation experiments are listed in Tab. 6 and Tab. 7, respectively. Observe that we use a smaller ConvNeXt encoder (i.e., fewer channels) to speed up training in the second setting. Additionally, the spatial dimensions of the features and embeddings are set to $(h, w) = (8, 16)$ to fit videos with a 2:1 aspect ratio.

Video Compression. For video compression, we use configurations with strides $(5, 3, 2, 2, 2)$ to encode videos at 1920×1080 pixels. One configuration with 9.3M parame-

T-NeRV Configuration: Video Representation (3.0M)			
Feature Grids	Temporal Resolutions		$(64, 128, 256)$
	Feature Size		$(11, 8, 16)$
Frame Encoder	Embedding Size		$(8, 8, 16)$
	ConvNeXt	Strides	$(5, 3, 2, 2, 2)$
		Blocks	$(3, 3, 9, 6, 3)$
		Channels	$(3, 64, 128, 256, 256, 8)$
	Temporal Augmentation	Pos. Enc.	$(1.25, 80)$
		MLP Dims	$(160, 512, 512, 1024)$
		MLP Activation	GELU
Temp. MLP	Positional Encoding		$(1.25, 80)$
	MLP Dims		$(160, 256, 256, 128)$
Decoder	Channels	Pre-Block	$(41, 102)$
		T-NeRV Blocks	$(102, 85, 71, 59, 49, 41)$
		Head Layer (3)	$(59, 12)$
		Head Layer (5)	$(41, 5)$
	Upsampling Factors		$(5, 3, 2, 2, 2)$
	Aggregation Window		$\{-2, -1, 1, 2\}$

Table 7. T-NeRV configuration for video representation (3.0M).

T-NeRV Configuration: Video Compression (9.3M)			
Feature Grids	Temporal Resolutions		$(64, 128, 256)$
	Feature Size		$(40, 9, 16)$
Frame Encoder	Embedding Size		$(26, 9, 16)$
	ConvNeXt	Strides	$(5, 3, 2, 2, 2)$
		Blocks	$(3, 3, 9, 6, 3)$
		Channels	$(3, 64, 128, 256, 256, 26)$
	Temporal Augmentation	Pos. Enc.	$(1.25, 80)$
		MLP Dims	$(160, 512, 512, 3744)$
		MLP Activation	GELU
Temp. MLP	Positional Encoding		$(1.25, 80)$
	MLP Dims		$(160, 256, 256, 128)$
Decoder	Channels	Pre-Block	$(146, 160)$
		T-NeRV Blocks	$(160, 136, 112, 96, 80, 64)$
		Head Layer (3)	$(96, 12)$
		Head Layer (5)	$(64, 5)$
	Upsampling Factors		$(5, 3, 2, 2, 2)$
	Aggregation Window		$\{-2, -1, 1, 2\}$

Table 8. T-NeRV configuration for video compression (9.3M).

ters, which we referenced in Appendix A.7, is described in Tab. 8.

C. Implementation of Video INR Baselines

Finally, we describe the process of implementing the six video INR baselines (NeRV [1], E-NeRV [5], C-NeRV [3], FFNeRV [4], HNeRV [2], and DNeRV [9]) in a common framework for the video representation experiments.

C.1. Motivation

As mentioned in the main paper, the representational capacity of video INRs has so far been assessed in two distinct settings: *Content-agnostic* methods (NeRV, E-NeRV, C-NeRV, FFNeRV) have been optimized for a setting where a large model (12.4M parameters) represents videos at a resolution of 1280×720 pixels, while *content-aware* approaches (HNeRV, DNeRV) employed much smaller models with 3M parameters to represent videos at a resolution of 1920×960 pixels, i.e. at a 2:1 aspect ratio.

Thus, we want to compare T-NeRV to the six baselines in both of these settings. However, no results are available for (i) content-agnostic methods in the second setting and (ii) content-aware methods in the first setting. The current content-aware methods, i.e., HNeRV and DNeRV, can also not directly be evaluated in the first setting as they can only fit videos with a 2:1 aspect ratio. Moreover, there are no open-source implementations of C-NeRV and DNeRV.

Consequently, we implement all of the six baselines in a common framework that we use to evaluate video representation performance. In the following, we describe our methods to derive suitable configurations for the cases (i) and (ii) from above.

C.2. 720p at 12.4M Parameters

Content-Agnostic Methods. The four content-agnostic methods were optimized for the first setting. Thus, we use their respective default configurations for our evaluation.

Content-Aware Methods. HNeRV employs tiny embeddings of dimensions $(c, h, w) = (16, 2, 4)$ in combination with strides $(5, 4, 4, 3, 2)$ to represent videos with a resolution of 1920×960 pixels. The remaining parameters are allocated to the decoder.[2] To allow HNeRV to represent 16:9 videos, we change the embedding dimension to $(16, 9, 16)$ and the strides to $(5, 2, 2, 2, 2)$ to arrive at the target resolution of 1280×720 pixels after upsampling. The remaining parameters are allocated to the decoder. Since DNeRV employs the same encoder and frame embeddings as HNeRV, we repeat this procedure for DNeRV. Additionally, we resize the difference embedding from $(2, 40, 80)$ to $(2, 45, 80)$ to reflect the change in the aspect ratio from 2:1 to 16:9. The collaborative content unit (CCU) is implemented as specified by equation 5 in [9], and the remaining parameters are allocated to the decoder.

C.3. 960p at 3M Parameters

Content-Agnostic Methods. We modify the dimensions of the latent feature from $(c, 9, 16)$ to $(c, 8, 16)$ to represent videos with a 2:1 aspect ratio with the four content-agnostic methods. To reflect the higher video resolution, we change the upsampling factors or strides from $(5, 2, 2, 2, 2)$ to $(5, 3, 2, 2, 2)$. To scale the default configurations from 12.4M to 3M parameters, we scale the size of the individual components while keeping the overall distribution of parameters to the components intact. For instance, if method X allocates 50%, 30%, and 20% of its parameters to components A, B, and C in the default 12.4M parameter config-

[2] This behavior is explicitly built into the HNeRV reference implementation. The embedding dimension depends only on the video resolution, but not on the model size.

uration, we retain the same percentages at the 3M parameter configuration.

Content-Aware Methods. The two content-aware methods provide default configurations for the second setting. However, in the case of DNeRV, the reported model size does not account for the size of the difference embeddings, which consist of more than 3.8M parameters. We remedy this issue by reducing the dimensions of the difference embeddings from $(2, 40, 80)$ to $(2, 10, 20)$ while shrinking the decoder accordingly to adhere to the 3M parameter limit. As a consequence, the CCU is inserted after the first DNeRV upsampling block as opposed to the second block for the dimensions to match.

C.4. Verifying our Implementations

Before evaluating our implementations of the six baselines for the video representation task, we verified the correctness of our implementations by reproducing either published results or results obtained from the reference implementation. We successfully validated our implementations for all of the video INRs except for DNeRV. Despite following the detailed model description in the paper and spending significant amounts of time trying to reproduce the published results, we were unable to obtain results in a similar range as the reported ones.

‘

References

[1] Hao Chen, Bo He, Hanyu Wang, Yixuan Ren, Ser Nam Lim, and Abhinav Shrivastava. NeRV: Neural Representations for Videos. In *Advances in Neural Information Processing Systems*, pages 21557–21568. Curran Associates, Inc., 2021. 2, 6

[2] Hao Chen, Matthew Gwilliam, Ser-Nam Lim, and Abhinav Shrivastava. HNeRV: A Hybrid Neural Representation for Videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10270–10279, 2023. 2, 6

[3] Carlos Gomes, Roberto Azevedo, and Christopher Schroers. Video Compression With Entropy-Constrained Neural Representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18497–18506, 2023. 2, 6

[4] Joo Chan Lee, Daniel Rho, Jong Hwan Ko, and Eunbyung Park. FFNeRV: Flow-Guided Frame-Wise Neural Representations for Videos. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 7859—-7870, New York, NY, USA, 2023. Association for Computing Machinery. 2, 6

[5] Zizhang Li, Mengmeng Wang, Huaijin Pi, Kechun Xu, Jianbiao Mei, and Yong Liu. E-NeRV: Expedite Neural Video Representation With Disentangled Spatial-Temporal Context. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXV*, pages 267—284, Berlin, Heidelberg, 2022. Springer-Verlag. 2, 6

[6] Alexandre Mercat, Marko Viitanen, and Jarno Vanne. UVG dataset: 50/120fps 4K sequences for video codec analysis and development. In *MMSys 2020 - Proceedings of the 2020 Multimedia Systems Conference*, 2020. 1, 2

[7] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016. 2

[8] Haiqiang Wang, Weihao Gan, Sudeng Hu, Joe Yuchieh Lin, Lina Jin, Longguang Song, Ping Wang, Ioannis Katsavounidis, Anne Aaron, and C. C.Jay Kuo. MCL-JCV: A JND-based H.264/AVC video quality assessment dataset. In *Proceedings - International Conference on Image Processing, ICIP*, 2016. 2

[9] Qi Zhao, M. Salman Asif, and Zhan Ma. DNeRV: Modeling Inherent Dynamics via Difference Neural Representation for Videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2031–2040, 2023. 2, 6