NIRVANA: Neural Implicit Representations of Videos with Adaptive Networks and Autoregressive Patch-wise Modeling: Supplementary Material

A. Additional Implementation Details

A.1. NeRV

We use the NeRV-L config from the original paper. The model takes in positional embedding of time coordinate as input. We set the number of sine levels to 80 and base value to 1.25. The hidden dimension of the 2-layer MLP in the beginning is set to 1024, with 128 output channels and an $8\times$ channel expansion for the first NeRV Block. We stack 5-NeRV blocks with upscale factors of $\{5, 3, 2, 2, 2\}$ for HD version and an additional block with $2\times$ upsampling for the 4K version. Following the standard training schedule, we use cosine learning rate schedule starting with $5e^{-4}$, with a warmup of 0.2. We use the combination of L1 and SSIM loss and train the model with a batch size of 1, as mentioned in the paper.

A.2. Entropy Loss and Weight Quantization

We follow the works of [6,8] for performing model compression with small variations. We represent our MLP layer weights as $\boldsymbol{W}_1, \boldsymbol{W}_2, ..., \boldsymbol{W}_L$ for L layers with the l^{th} layer weight matrix $\boldsymbol{W}_l \in \mathbb{R}^{O_l \times I_l}$ having a shape $O_l \times I_l$ consisting of continuous values. For each weight matrix, we maintain a corresponding flattened latent quantized representation vector $\widetilde{\boldsymbol{W}}_l \in \mathbb{Z}^{O_l I_l}$. $\widetilde{\boldsymbol{W}}_l$ consists of integer values for each corresponding element in $\boldsymbol{W}_l$. For ease of notation, we drop the subscript l.

We then maintain decoders $f_\phi(.)$ parameterized by parameters ϕ. The weight matrix $\boldsymbol{W}$ is then obtained from the quantized weights $\widetilde{\boldsymbol{W}}$ as $\boldsymbol{W} = f_\phi(\widetilde{\boldsymbol{W}})$. Prior works use matrices or vectors to represent the weight decoders while we use a single scalar ϕ as a parameter of the decoder. The weight matrix is thus simply, $\boldsymbol{W} = reshape(\phi\widetilde{\boldsymbol{W}})$, where the scale parameter ϕ is multiplied with individual values of $\widetilde{\boldsymbol{W}}$. We make the scale parameter learnable by passing gradients. Thus, it effectively controls the bit width of the weight matrix. We maintain separate decoders for each layer of the MLP.

To make the network fully differentiable, we maintain continuous surrogates $\widehat{\boldsymbol{W}}$ for the discrete latents $\widetilde{\boldsymbol{W}}$. During training, we simply round the surrogates to their nearest integer to obtain the discrete latents which are then passed to the decoder. We make the rounding operation differentiable using the straight-through estimator [3] to pass the gradients from $\widehat{\boldsymbol{W}}$ to $\widetilde{\boldsymbol{W}}$.

To reduce the entropy of the quantized latents, we use probability models from [2]. For each continuous surrogate $\widehat{\boldsymbol{W}}$, we maintain probability models $c_\theta(.)$ parameterized by θ which output the CDF of the latent distributions. Similar to prior works, we use uniform noise $n \sim \mathcal{U}\left(-\frac{1}{2}, \frac{1}{2}\right)$ as a substitute for quantization. The entropy of the model can now be minimized by minimizing the self-information $\boldsymbol{I}$ as follows:

$$I(\widetilde{\boldsymbol{W}}) = -\log_2(c_\theta(\widehat{\boldsymbol{W}} + n)). \tag{1}$$

This serves as the entropy regularization loss which controls the rate-distortion tradeoff. A higher entropy coefficient λ_I leads to more compressed latents (lower rate), but usually at the cost of PSNR (higher distortion). The network latents, decoder parameter, and probability model parameters are learnable and jointly optimized. Following prior works, we use a learning rate of $1e{-4}$ for the probability model weights and the same learning rate for the decoder weights. We set the learning rate of the latents to $5e{-4}$. All the parameters are optimized in an end-to-end manner with an Adam optimizer during training, thus requiring no post-hoc approaches. After training, we discard the probability models and use the frequency of each quantized value in the latent vector to obtain the probability tables required for arithmetic entropy coding. Note that the continuous surrogates are discarded and only their rounded discrete latents are stored using entropy coding. These latents can then be decoded using the probability tables. The decoder parameters and probability tables have almost no overhead compared to the overall model latents.

B. Additional Dataset Details

B.1. UVG-4K

In addition to the datasets shown in Section 4 of the main paper, we show quantitative and qualitative results on 7 more videos from the UVG dataset at the 4K resolution: Twilight, Sunbath, CityAlley, FlowerFocus, FlowerKids, RiverBank, and RaceNight. We dub this dataset UVG-4K (Set 2).

B.2. Youtube-8M

We select 5 more videos from the Youtube-8M dataset, with varying video content to further test the ability of our model to encode longer videos. This is an extension of the experiments from Section 4.4 which consists of a single video (Mario Kart). We present the details of each video used in Table 1:

Video	Frames	Link
Mario Kart	4000	`http://bit.ly/3XjIvfR`
Dota	4261	`http://bit.ly/3Xf6Nru`
Ride	4000	`http://bit.ly/3TOEgWI`
Submarine	3626	`http://bit.ly/3EJKzGM`
Water Scooter	4199	`http://bit.ly/3OhF99h`
Mortal Kombat	3239	`http://bit.ly/3EdvNGU`

Table 1. **Details of Youtube-8M dataset.**

C. Video-wise comparison

We show additional quantitative results on UVG-4K (Set 2) and Youtube-8M mentioned above.

C.1. UVG-HD

We provide video-wise results of our approach along with that of NeRV [4]. We evaluate the video on the additional perceptual quality metrics of FLIP [1] and VMAF [7] as well, along with the standard PSNR. In addition to the image quality metrics, we also measure the tOF/tLP metrics proposed by [5] for measuring temporal consistency. tOF measures the L1-error between the optical flows from the predicted frames and the ground-truth frames while tLP uses the LPIPS metric instead. Results are summarized in Table 2. We see that we continue to obtain similar performance compared to NeRV in terms of these metrics while being $\sim 12\times$ faster. Also note the adaptive BPP of our method, which is based on the amount of motion in each video. In contrast, NeRV maintains a fixed BPP due to fixed model size (ShakeNDry shows twice the BPP due to half the number of frames). We observe a small drop in VMAF ($92.33 \rightarrow 91.14$) while maintaining similar value of FLIP (~ 0.0632) compared to NeRV. We marginally outperform NeRV based on the temporal metrics of tOF ($0.3167 \rightarrow 0.3077$) and tLP ($0.2125 \rightarrow 0.2032$).

C.2. UVG-4K

We provide video-wise results on the 2 sets of UVG at 4K resolution. For Set 1, we obtain comparable performance to NeRV while obtaining $\sim 6\times$ faster encoding speed. Similar to UVG-HD, we continue to show the benefits of adaptive compression, with static videos such as Honeybee showing lower levels of BPP (0.14) compared to the most dynamic video, ReadySteadyGo (0.41 BPP). For Set 2, we outperform NeRV by 1.5 PSNR while still obtaining 25% lower BPP $0.28 \rightarrow 0.21$. The PSNR drop of NeRV on the Twilight video is largely due to quantization at the fixed bit width of 20. Hand-tuning is necessary in order to maintain higher PSNR but at the cost of BPP. In contrast, our approach maintains the reconstruction quality for a variety of videos and adaptively quantizes for each video.

C.3. Youtube-8M

We now provide video-wise results of 5 videos picked from the Youtube-8M dataset at 1080p resolution. Details of the videos are provided in Table 1. Results are summarized in Table 4. We see that our reconstruction quality does not degrade with longer videos. This behavior is different from NeRV, which obtains a significant drop in PSNR (about -5.2). This is in line with the observations in Section 4.4 of the main paper, where we see that increasing number of frames results in drop of NeRV's reconstruction quality while we maintain similar levels of performance.

D. Additional Ablations

In addition to the ablations shown in Fig. 5, we analyze the effect of layer size and the number of layers of the MLP in our network when evaluating on the Jockey video of the UVG-HD dataset. Note that the default values of layer size is 512 and the number of layers is 5.

D.1. Effect of Layer Number

We increase the number of layers from 3 to 6, while keeping other parameters at their default values and varying the entropy coefficient λ_I for each curve as in Section 5. Results are summarized in Figure 1(a). We see that increasing the number of layers from 3 to 5 improves the tradeoff curve (shifts upwards) in the low BPP regime (<0.8). However, increasing it further shifts the curve upwards and to the right. This might be because the MLP network requires higher levels of non-linearity to learn a global representation for a group of 3 frames which typically contain significant motion in the case of Jockey. However, for 6 layers the network shifts the curve upwards and to the right, and we no longer obtain increase in PSNR at no cost of BPP.

D.2. Effect of Layer Size

We vary the layer size from 128 to 768 progressively, in steps of 128 for each of the 5 layers. Results are summarized in Figure 1(b). We see that increasing the layer size simply shifts the curve upwards and to the right, which is expected as a higher number of parameters leads to more

Video Name	NIRVANA (Ours)						NeRV					
	PSNR ↑	VMAF ↑	FLIP ↓	tOF ↓	tLP ↓	BPP ↓	PSNR ↑	VMAF ↑	FLIP ↓	tOF ↓	tLP ↓	BPP ↓
ReadySteadyGo	**35.43**	**98.04**	**0.0862**	**0.3348**	0.2231	1.26	34.59	96.95	**0.0862**	0.3604	**0.2089**	**0.81**
Bosphorus	**40.53**	**90.97**	**0.0549**	**0.1900**	**0.1371**	**0.68**	39.11	88.26	0.0621	0.1980	0.1654	0.81
Beauty	**35.77**	84.46	**0.0524**	**0.2716**	**0.3123**	0.96	34.57	**86.35**	0.0605	0.3352	0.3646	**0.81**
Honeybee	38.83	91.31	0.0505	0.0918	0.1511	**0.51**	**39.71**	**95.08**	**0.0419**	**0.0824**	**0.1491**	0.81
Jockey	37.56	93.07	0.0710	0.7303	0.2777	0.96	**38.16**	**95.76**	**0.0653**	**0.7001**	**0.2684**	**0.81**
Yachtride	**37.94**	**91.31**	**0.0668**	**0.3480**	**0.1604**	1.03	35.68	88.70	0.0786	0.4243	0.1881	**0.81**
ShakeNDry	37.82	88.81	0.0608	0.1875	0.1607	**0.76**	**39.68**	**95.20**	**0.0468**	**0.1165**	**0.1429**	1.61
Average	**37.70**	91.14	0.0632	**0.3077**	**0.2032**	**0.86**	37.35	**92.33**	**0.0631**	0.3167	0.2125	0.92

Table 2. **Video-wise performance on UVG-HD:** We show video-wise results of the 7 videos in UVG-HD and compare the reconstruction quality using the 3 image metrics: PSNR, VMAF, and FLIP, the 2 temporal metrics: tOF/tLP [5], along with compression rate measured by BPP. We see that we maintain similar performance as NeRV in all 5 metrics and BPP while having 12× faster encoding speed (as also shown in Table 1 of the main paper).

Video Name	NIRVANA (Ours)		NeRV	
	PSNR	BPP	PSNR	BPP
ReadySteadyGo	**33.85**	0.41	33.22	**0.24**
Bosphorus	38.71	**0.21**	**39.0**	0.24
Beauty	**31.96**	0.28	31.05	**0.24**
Honeybee	35.64	**0.14**	**36.36**	0.24
Jockey	35.05	0.30	**35.9**	**0.24**
Yachtride	**36.33**	0.33	35.05	**0.24**
ShakeNDry	34.78	**0.24**	**36.09**	0.49
Average	35.18	**0.27**	**35.23**	0.28

(a) UVG-4K (Set 1)

Video Name	NIRVANA (Ours)		NeRV	
	PSNR	BPP	PSNR	BPP
FlowerFocus	36.50	**0.12**	**37.08**	0.24
CityAlley	37.43	**0.17**	**38.39**	0.24
Twilight	**38.02**	**0.13**	20.99	0.24
FlowerKids	**34.62**	0.26	33.77	**0.24**
RiverBank	**33.83**	0.26	32.35	**0.24**
RaceNight	32.72	0.27	**32.92**	**0.24**
Sunbath	37.75	**0.24**	**44.17**	0.49
Average	**35.84**	**0.21**	34.23	0.28

(b) UVG-4K (Set 2)

Table 3. **Video-wise comparison on different sets of UVG-4K:** We show video-wise results on 2 different sets of 7 UVG videos at 4K resolution. Set-1 consists of videos from UVG-HD at 4K resolution while Set 2 consists of additional 7 videos from the dataset. We maintain similar performance in terms of PSNR and BPP as NeRV while also being ∼6× faster for both sets and being 6× faster in terms of encoding time.

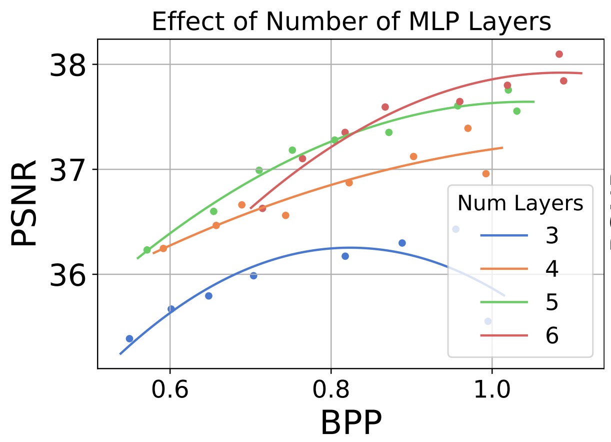
Video Name	NIRVANA (Ours)		NeRV	
	PSNR	BPP	PSNR	BPP
Dota	**38.03**	0.62	35.53	**0.34**
Ride	**36.65**	1.09	29.74	**0.36**
Submarine	**38.48**	0.69	33.64	**0.40**
Water Scooter	**37.79**	0.84	30.46	**0.34**
Mortal Kombat	**36.02**	1.03	31.36	**0.45**
Average	**37.39**	0.85	32.14	**0.38**

Table 4. **Results on Youtube-8M videos with long duration.** We provide video-wise results on 5 videos picked from the Youtube-8M datasets with approximately 4000 frames compared to the typical 600 from UVG. Still we maintain PSNR/BPP with no change in hyperparameters, whereas NeRV shows large degradation in performance for the same network and similar encoding times.

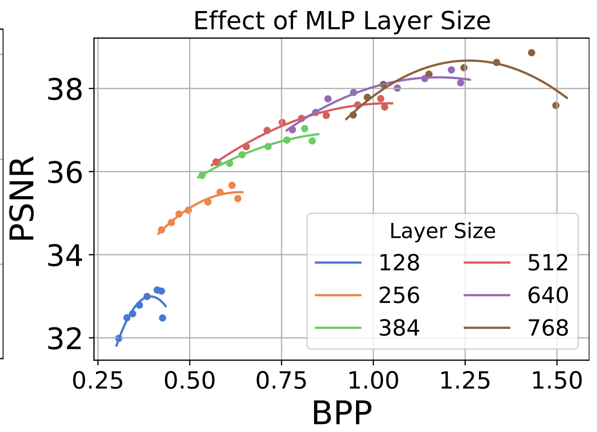
representation capability of the network at the cost of more parameters. While increasing the number of layers increases number of parameters as well, a similar tradeoff is not present in that case up to a certain level, suggesting that a minimum number of non-linearities/activation functions are important to achieve the optimal tradeoff.

D.3. Effect of video content

Section 4.5 in the paper shows the capability of our approach to adapt to video content based on varying stability. To further illustrate this, we visualize the correlation between BPP and L1-error (average L1 norm between pixel values of two frame groups) in Fig. 2. We see that for the Jockey video, lower L1-error between 2 subsequent frame groups shows direct correlation with the BPP required for storing that frame-group. This is to be expected as lower frame residuals reduces the entropy of the quantized resid-



(a)



(b)

Figure 1. Increasing number of layers improves the PSNR/BPP curve upto 5 layers in the lower BPP regime (<0.8). Increasing layer size shifts the PSNR/BPP curve upwards and to the right as representation capacity increases along with more parameters.

Iterations	Honeybee		Jockey	
	PSNR	BPP	PSNR	BPP
500	37.93	**0.35**	35.97	**0.85**
1000	38.25	0.37	36.82	0.90
1500	38.72	0.50	37.25	0.93
2000	**38.83**	0.51	**37.56**	0.96

Table 5. **Convergence.** We vary number of training iterations for each frame group in the Honeybee (static) and Jockey (dynamic) videos from UVG-HD. Honeybee achieves faster convergence showing that stable videos can be encoded faster.

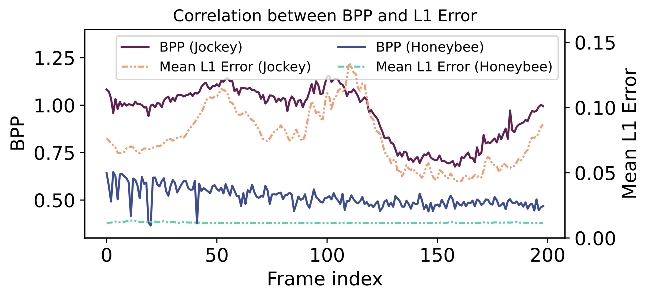


Figure 2. BPP correlates with L1 error: lower L1 error gives lower BPP.

ual weights as well and subsequently, lower BPP.

In addition to BPP, we also analyze the effect of convergence speed for various types of videos. We vary number of iterations for training networks for each frame group for the Honeybee and Jockey videos and visualize the results in Table 5. Stable videos such as Honeybee converge faster with only a 0.9 dB PSNR drop for $4\times$ encoding speedup from 2000 to 500 iterations while dynamic videos such as Jockey obtain a larger $1.5dB$ PSNR drop at similar encoding speedups. Due to our autoregressive modeling, the initialization of the network weights from the previous frame group provides a good solution for stable videos with little inter frame shift in comparison to dynamic ones.

E. Denoising

To test our method on downstream applications, we choose the task of denoising. Given a noisy video, our method is capable of removing the noisy patterns without any explicit supervision. We train and test our method on videos with various noise patterns and observe that out method outperforms all classical filter baselines. Results are shown in table 6. We outperform classically filters such as Mean/Median/Gaussian filter for a variety of noises such as Black/White/Salt and Pepper showing the efficacy of our representations to be applied to other tasks as well.

Denoising Method	Black	White	Salt& Pepper	Random	Average
Baseline	27.5	28.29	27.95	30.95	28.74
Mean Filter	29.11	29.06	29.10	29.63	29.22
Median Filter	33.89	33.84	33.87	33.89	33.87
Gaussian Filter	30.27	30.14	30.23	30.99	30.41
NIRVANA	**37.18**	**37.19**	**37.21**	**37.22**	**37.20**

Table 6. **Results for video denoising.** We outperform classical denoising filters by a large margin for different types of noises.

F. Qualitative Results

In Figure 3 we qualitatively visualize the reconstruction results for 3 videos from Set 2 of UVG-4K and 2 videos from Set 1. We obtain higher-quality and more faithful reconstructions while preserving more details at similar or even lower BPP compared to NeRV; *e.g.*, Twilight ($0.24 \rightarrow 0.13$), RiverBank ($0.24 \rightarrow 0.26$), CityAlley ($0.24 \rightarrow 0.17$). Notice the bird which is reconstructed by our approach in Twilight (top), or finer details of the branches in RiverBank (second), or maintaining the right color information of the door and the people's shirts in CityAlley (third) or the red dot in Yachtride (fourth). We continue to maintain important information in the images such as the number on the signboard of ReadySetGo (bottom) while NeRV fails to capture these fine details.

References

[1] Pontus Andersson, Jim Nilsson, Tomas Akenine-Möller, Magnus Oskarsson, Kalle Åström, and Mark D. Fairchild. Flip: A difference evaluator for alternating images. *Proc. ACM Comput. Graph. Interact. Tech.*, 3(2), aug 2020.

[2] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. *arXiv preprint arXiv:1802.01436*, 2018.

[3] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[4] Hao Chen, Bo He, Hanyu Wang, Yixuan Ren, Ser Nam Lim, and Abhinav Shrivastava. Nerv: Neural representations for videos. *Advances in Neural Information Processing Systems*, 34:21557–21568, 2021.

[5] Mengyu Chu, You Xie, Jonas Mayer, Laura Leal-Taixé, and Nils Thuerey. Learning temporal coherence via self-supervision for gan-based video generation. *ACM Transactions on Graphics (TOG)*, 39(4):75–1, 2020.

[6] Sharath Girish, Kamal Gupta, Saurabh Singh, and Abhinav Shrivastava. Lilnetx: Lightweight networks with extreme model compression and structured sparsification. *ArXiv*, abs/2204.02965, 2022.

[7] Zhi Li, Anne Aaron, Ioannis Katsavounidis, Anush Moorthy, and Megha Manohara. Toward A Practical Perceptual Video Quality Metric, 2016.

[8] D. Oktay et al. Scalable model compression by entropy penalized reparameterization. In *ICLR*, 2020.

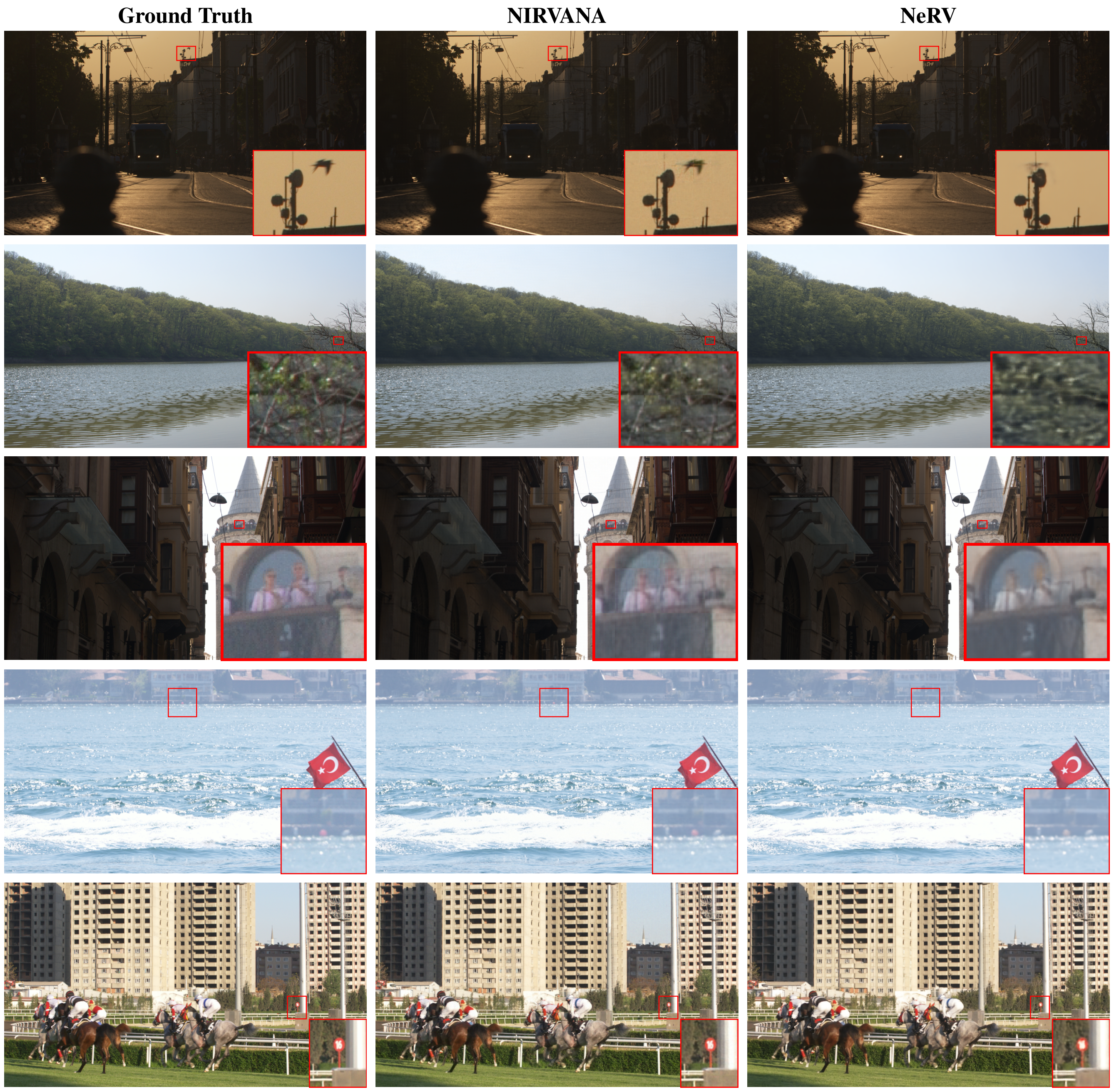


Figure 3. **Qualitative results from UVG-4K:** (Left) Ground truth video frames. (Center) Reconstruction from NIRVANA. (Right) Reconstruction from NeRV. Top to bottom: We show additional examples where NIRVANA is able to preserve the image fidelity after reconstruction, such as the bird in Twilight (top), the tree in RiverBank (second), humans in CityAlley (third) and signboards in ReadySetGo (bottom).