A Additional Implementation Details

A.1 Datasets

We use eight different video sequences to conduct experiments for representation ability. Here we provide a visualization of some random selected frame images from these videos in Fig. 6. It can be seen that the video sequences we used have diverse types of contents, and our method outperforms baseline method on all of these sequences.

Fig. 6. A visualization of frame images in the video sequences that been used in experiments. Name of each sequence is indicated on the left.

For the dataset split in Section. 5.7, we split each sequence at a ratio of 3 : 1. To be more specific, for every four consecutive frames, the last frame is

distributed into the "unseen" split while the other three frames are assigned to "seen" split. The normalized frame index is calculated before the partition, so we can use the frame index in "unseen" split to quantitatively experiment the temporal interpolation ability.

A.2 Network Architecture

We provide a detailed illustration of the architectures of 12.49M E-NeRV and 12.57M NeRV-L. Splitting the network structure into two parts: before and after the generation of feature map $\mathbf{f_t}$, in Fig. 7 we show the comparison of structure and parameters of original coupled MLP and our proposed disentangled network. The input and output dimensions of single-head-attention transformer block Φ is 256, and the dimension of MLP layer inside the Φ is 128. The transformer block F_θ is almost identical to Φ except it supports a multi-head-attention where number of heads equals 8. In Fig. 8 we show the comparison between the similar convolution stages and also our proposed branch for temporal instance normalization.

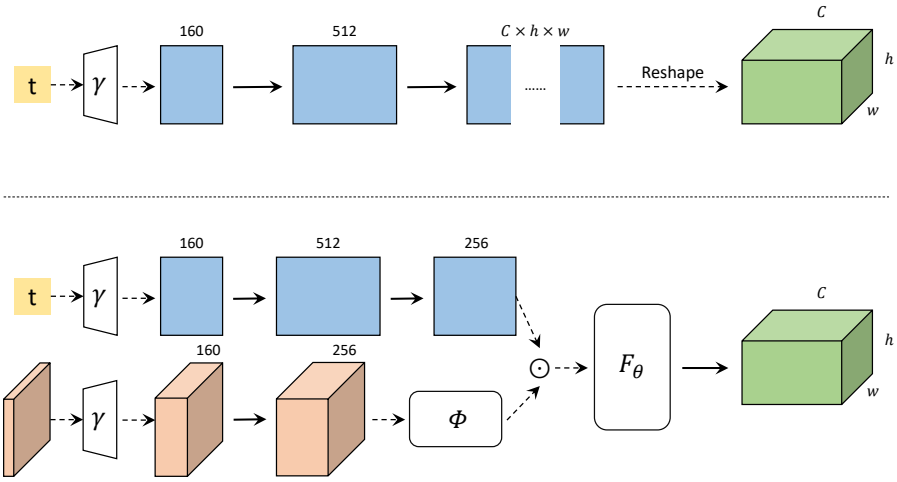

Fig. 7. Comparison of how to generate spatial-temporal feature map in NeRV and our method. The black solid arrows stand for MLP layers with learnable parameters. Our method greatly reduce the size of parameters in this stage by introducing the disentangled structure. Detailed description of Φ and F_θ is in A.2.

A.3 Training Details

For fair comparison, we trained all of our models following the schedule proposed in [2]. The maximum of learning rate is set to be 5e-4 and follows the

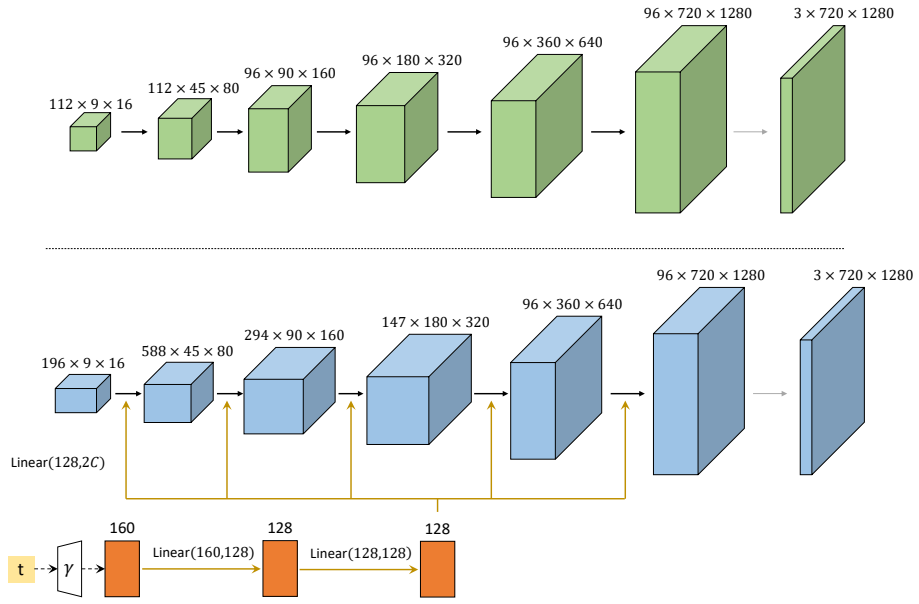

Fig. 8. Comparison of the convolution stage, and also our proposed temporal instance normalization module. The black solid arrows stand for 3×3 convolution, pixel-shuffle and activation to up-scale the feature map. We adopt our upgraded design (Section. 4.2) at the first block. The gray arrows stand for 1×1 convolution with a sigmoid function to get the desired 3-channel image. The brown arrows indicate the MLP layers for temporal instance normalization. Since we lower the size of first stage, in this stage we can increase the channel dimensions for much better performance.

"warmup-cosine-drop" schedule. The learning rate increases linearly in first 20% of the iterations and drops to 0 in a cosine schedule in the rest of the iterations. As mentioned in [2] that more iterations lead to better performance, we set the batchsize of 1 in all the experiments to maximize the nubmers of back propagation. All experiments are run with NVIDIA RTX2080ti, and since the batchsize is small, GPU with less memory can also support. We implement all of our experiments in PyTorch.

A.4 Metrics

For the representation ability, we adopt two metrics in our experiments: PSNR and MS-SSIM. These two metrics all measure the similarity between the ground truth frame and the frame image that network outputs. The PSNR stands for "Peak Signal-to-Noise Ratio", calculated on the basis of mean-square error of two images. The MS-SSIM stands for "Multiscale structural similarity", it is calculated on the basis of luminance, contrast and structure. The method with larger value of both metrics have better performance on reconstructing the frame image.

Table 5. Per-video quantitative results of alternative comparison. See Section. 5.4 for a detailed description of each alternative.

		PSNR↑							
	Size	Bunny	Beauty	Bosphorus	Bee	Jockey	SetGo	Shake	Yacht
NeRV-C_S	5.8M	35.97	35.24	33.95	39.88	34.07	27.00	34.98	28.67
Ours-1†	5.8M	38.94	35.81	36.34	40.97	36.95	30.25	36.58	31.44
NeRV-Split	7.2M	39.10	36.02	36.29	40.95	36.24	29.64	36.91	31.06
Ours-2†	7.2M	40.23	36.27	37.31	41.31	37.24	31.01	37.65	32.88
E-NeRV-MLP	12M	42.15	36.66	39.68	41.69	39.23	34.28	39.21	35.43
E-NeRV-Conv	12.5M	42.49	36.69	39.97	41.70	39.28	34.54	39.26	35.50
E-NeRV	12.5M	42.87	36.72	40.06	41.74	39.35	34.68	39.32	35.58
		MS-SSIM↑							
	Size	Bunny	Beauty	Bosphorus	Bee	Jockey	SetGo	Shake	Yacht
NeRV-C_S	5.8M	0.9843	0.9446	0.9567	0.9924	0.9509	0.9324	0.9667	0.9212
Ours-1†	5.8M	0.9920	0.9508	0.9760	0.9937	0.9710	0.9689	0.9790	0.9590
NeRV-Split	7.2M	0.9930	0.9512	0.9745	0.9937	0.9686	0.9629	0.9807	0.9548
Ours-2†	7.2M	0.9946	0.9588	0.9801	0.9941	0.9746	0.9736	0.9845	0.9652
E-NeRV-MLP	12M	0.9964	0.9633	0.9898	0.9945	0.9832	0.9877	0.9897	0.9841
E-NeRV-Conv	12.5M	0.9966	0.9638	0.9905	0.9946	0.9834	0.9885	0.9899	0.9843
E-NeRV	12.5M	0.9969	0.9887	0.9641	0.9906	0.9946	0.9835	0.9900	0.9846

B More Related Works

B.1 Differences between video INRs and video GANs

Video INR [43,27] is a certain type of representation that using network functions to implicitly fit the video sequence. The efficient frame-wise video INRs like NeRV [2] and E-NeRV proposed in this paper, generate 2-D frame from 1-D latent code, thus having similar architecture compared with video GAN methods. The difference is that GAN frameworks try to generate diverse videos from different random latent codes, while video INRs try to fit to one individual video precisely. DIfferent latent codes in our case stand for different frame index inputs. On the other hand, video GAN, like recently proposed DiGAN [62] can also adopt video INR as the representation, which inherits the advantage of continuous implicit functions.

B.2 Discussion of video representation learning

Another line of related works lies in the self-supervised representation learning. Self-supervised learning (SSL) methods [55,12,3,5,51,11,4] focus on pretext tasks for visual pre-training from unlabelled data, well-studied tasks including contrastive learning [12,3,5] and masked image modeling [11,4]. For the video SSL, many works [54,18,25,48] also develop specific objectives for video pre-training which aims for the video-related down-stream tasks. For now, the SSL methods utilize network to extract the unique representation of each video, while

the video INR tries to fit each video and make the network weight as the representation. Despite of the difference, both methods focus on the representation of video sequence, and how can implicit representation be involved in representation learning, like SSL, is an exciting new direction.

C Baseline NeRV Details

As mentioned in Section. 5.1, for the network structures which are orthogonal to our motivation in NeRV, we follow their original settings. To be more specific, for the MLP, we place a GELU activation function following each linear layer, which in ablation study of NeRV[2] is proved to be superior of other activations. For convolution stages, each NeRV Block also ends with a GELU activation function. Additionally, we don't adopt any of the normalization modules in our model except for the temporal instance normalization. Neither of the Layernorm in transformer and Batchnorm in convolution is adopted. As analyzed in [2], the normalization layer reduces the over-fitting capability of the neural network, which is contradictory to the training objective of video INR.

D More Experiment Results

D.1 Alternative Comparison

Since the experiment results provided in Section. 5.4 are averaged over 8 different video sequences, here we provide the detailed results of each video in Table. 5. The results show the consistent relative relation among the alternatives on all videos.

D.2 Ablation Study

As experiment results provided in Section. 5.5 are also averaged over 8 different video sequences, the detailed results are provided in Table. 6 for a better look.

D.3 Schedule Comparison

In this section we show the comparison of our method and NeRV on all video sequences for different training epochs. As illustrated in Fig. 9, the PSNR of E-NeRV's output frame images surpasses NeRV's with a much longer 8× schedule. And we can further boost the performance with more epochs. Our method's representation ability shows strong advantage on both the video sequences with dynamic content like "Bunny" and the video sequences with more still frames like "Bee", which proves the significant improvement of our proposed video INR.

Table 6. Per-video results of ablation study. As adding the proposed modules, our method's performance increases gradually.

	Φ	F_θ	IN	PSNR↑ Bunny	Beauty	Bosphorus	Bee	Jockey	SetGo	Shake	Yacht
NeRV-L	-	-	-	39.63	31.86	36.06	37.35	41.23	38.14	37.22	32.45
Variant 1	✗	✗	✗	41.95	36.30	38.97	41.38	38.40	32.81	38.99	34.30
Variant 2	✓	✗	✗	42.16	36.32	39.29	41.55	38.71	33.16	39.06	34.44
Variant 3	✓	✓	✗	42.34	36.52	39.95	41.64	39.02	33.51	39.18	35.40
E-NeRV	✓	✓	✓	42.87	36.72	40.06	41.74	39.35	34.68	39.32	35.58
	Φ	F_θ	IN	MS-SSIM↑ Bunny	Beauty	Bosphorus	Bee	Jockey	SetGo	Shake	Yacht
NeRV-L	-	-	-	0.9931	0.9787	0.9562	0.9825	0.9941	0.9783	0.9883	0.9704
Variant 1	✗	✗	✗	0.9964	0.9570	0.9882	0.9941	0.9806	0.9837	0.9893	0.9805
Variant 2	✓	✗	✗	0.9966	0.9571	0.9892	0.9944	0.9814	0.9848	0.9896	0.9810
Variant 3	✓	✓	✗	0.9967	0.9605	0.9903	0.9945	0.9825	0.9861	0.9898	0.9832
E-NeRV	✓	✓	✓	0.9969	0.9887	0.9641	0.9906	0.9946	0.9835	0.9900	0.9846

D.4 Temporal frequency study

According to the experiments in Section. 5.7, we set two different frequency values $b = 1.05$ and $b = 1.25$, and conduct frame interpolation experiments on all the video sequences. From the results in Table. 7 we can see the conclusion that adjusting temporal frequency boost both performances on seen and unseen splits still remains, while same adjustment leads to performance drop on seen split for NeRV.

E Visualizations

E.1 Reconstructed Frames

In this section we show a qualitative comparison between our method and the baseline NeRV-L. As the visualization provided in Fig. 10, despite that NeRV can reconstruct the whole frame image to a good degree, our E-NeRV can further fix some detailed regions. For example, the red flag and elaborate wrinkle in the images of first row.

E.2 Temporal Interpolation

As visualized in Fig. 11, lowering the frequency value for temporal instance normalization can greatly boost E-NeRV's performance on temporal interpolation.

Table 7. An study of different encoding frequency for temporal interpolation.

Method	split	b	Bunny	Beauty	Bosphorus	Bee	Jockey	SetGo	Shake	Yacht
NeRV	seen	1.25	39.30	36.16	37.70	41.13	38.24	32.21	37.36	32.78
		1.05	39.06	36.05	37.35	40.94	37.84	31.50	37.09	32.44
	unseen	1.25	28.58	23.98	25.65	37.08	17.27	14.69	28.05	19.83
		1.05	33.70	26.55	29.55	39.90	18.24	15.73	30.40	22.23
E-NeRV	seen	1.25	42.52	36.96	40.71	41.72	39.78	35.44	39.58	36.32
		1.05	42.63	37.04	40.74	41.74	40.06	35.90	39.61	36.57
	unseen	1.25	29.32	24.26	26.30	37.92	17.37	14.95	28.56	20.26
		1.05	33.77	26.41	29.10	39.67	18.24	15.91	30.57	22.31

The results become more elaborate and less blurry compared to interpolation with original parameters.

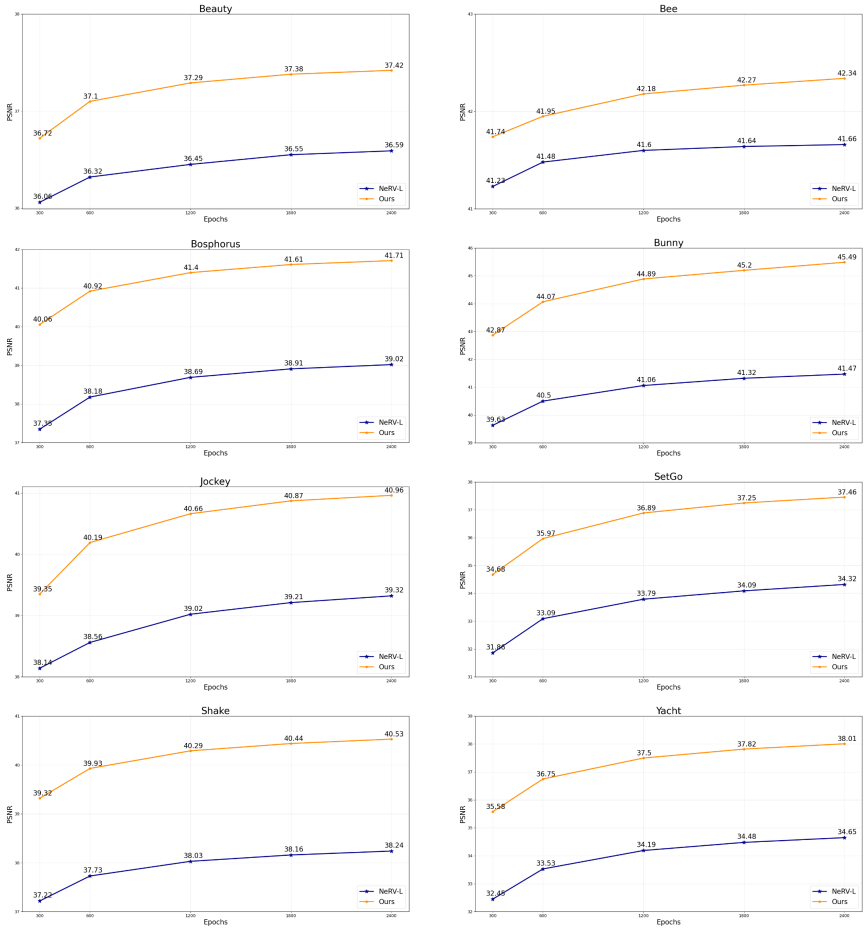

Fig. 9. The experiment results of different epochs. Our method's performance at 300 epochs consistently surpass the baseline's results at 2400 epochs on all the video sequences.

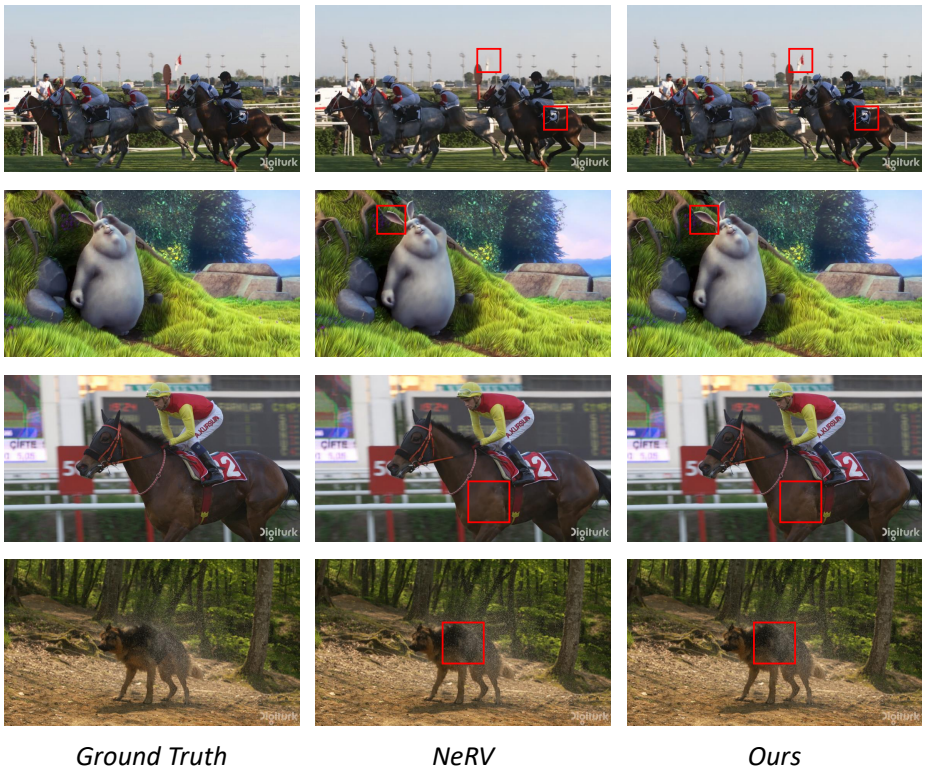

Fig. 10. Visualization of reconstructed frames. We use red rectangles to highlight the detailed regions that NeRV fails to synthesis while our method succeeds.

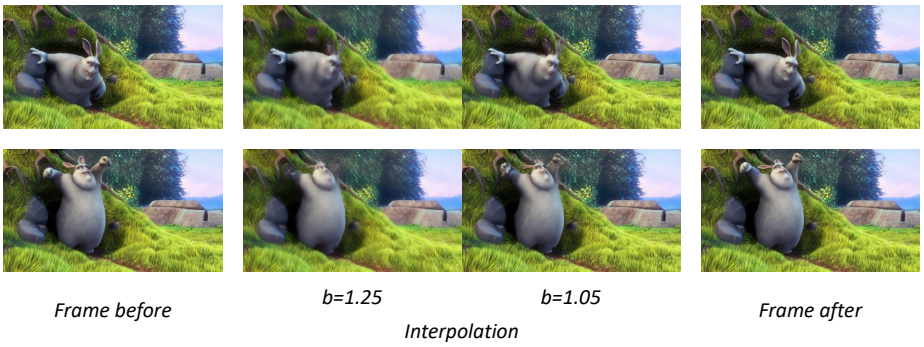

Fig. 11. Visualization of frame interpolation. Adjusting the frequency for temporal instance normalization modules provides a more precise interpolation.