CNeRV: Neural Visual Representation with Content-adaptive Embedding

Supplementary Material

A More Results

A.1 Embedding Quantization Results

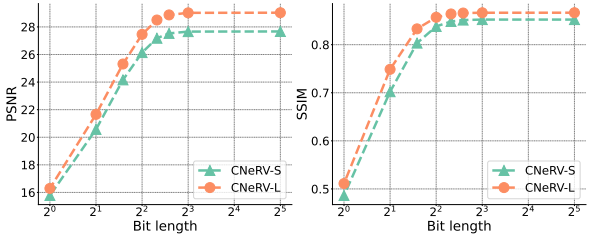

Figure 9: **Embedding quantization**, We evaluate reconstruction quality on unseen images with embedding quantization.

For unseen frames and with an 8-bit quantized model, we further quantize the block embedding, ranging from the original 32 bit to 1 bit, where where the image embedding can maintain most of its capacity with only 6 bits in Figure 9

A.2 More Autoencoder Results

We first provide more autoencoder results, where both seen PSNR and unseen PSNR improve as we increase the training images, embedding length, and model size.

Table 11: Autoencoder results with more training images, bigger model size, and longer embeddings.

Methods	Embed Length	Model Size	Training Images	PSNR Seen ↑	PSNR Unseen ↑
ConvAE	480	68M	2k	24.29	23.2
ConvAE	1000	68M	4k	26.6	25.92
ConvAE	48k	83M	4k	**29.28**	**29.28**

A.3 Sensitivity Analysis and Limitations

We conduct an ablation study on our content-adaptive encoding. For base value b in Equation 2, Table 12 shows that 1.15 performs better than 1.05 and 1.25. For frequency length P and Q, Table 13 shows results with 10, 15, and 20. Although 20 is better for seen PSNR, it does not further improve unseen PSNR and will introduce more encoding computation. Since we mainly focus on internal generalizability in this work, *i.e.*, unseen image PSNR,

Table 12: **Frequency value** b ablation

Base number	PSNR		MS-SSIM	
	Seen ↑	Unseen ↑	Seen ↑	Unseen ↑
1.05	32.6	25.94	0.915	0.8072
1.15	**33.83**	**26.85**	**0.9539**	**0.8242**
1.25	33.5	26.67	0.949	0.8217

Table 13: **Frequency length** ablation

Freq number	PSNR		MS-SSIM	
	Seen ↑	Unseen ↑	Seen ↑	Unseen ↑
10	32.75	26.3	0.9229	0.8151
15	33.83	**26.85**	0.9539	**0.8242**
20	**34.07**	26.84	**0.9565**	0.8242

Figure 10: Sensitivitiy for **Block number**

Block number	Embed length	PSNR		MS-SSIM	
		Seen ↑	Unseen ↑	Seen ↑	Unseen ↑
1×2	480	30.22	25.31	0.8975	0.8059
2×4	480	**33.83**	**26.85**	**0.9539**	**0.8242**
5×10	500	4.7	4.69	0.3344	0.3336
5×10	1000	34.49	27.63	0.9788	0.8597

Table 14: NeRV performance with shuffled frame index. It shows that the input embedding of frame index does not provide any meaningful information since shuffling the data does not impact the final performance.

	Dataset Size	PSNR		MS-SSIM	
		Seen ↑	Unseen ↑	Seen↑	Unseen ↑
Sequential	1k	34.6	12.57	0.9489	0.3495
Shuffle	1k	35.22	13.15	0.9554	0.3905
Sequential	2k	33.53	16.46	0.919	0.4933
Shuffle	2k	33.47	16.36	0.9193	0.4841
Sequential	4k	32.78	20.68	0.9077	0.6994
Shuffle	4k	32.45	20.24	0.9073	0.6859

we choose 15 as the default frequency length. For block numbers, Table 10 shows results for 1×2, 2×4, and 5×10. where total embedding length is computed by $M \times N \times L$. With 2×4 blocks, CNeRV reaches the best performance for both seen and unseen images. Note that when embedding length is too short (*e.g.*, 10 for block number 5×10), CNeRV fails to overfit; nevertheless, it can still perform reasonable reconstruction.

Limitation The main limitation of our method is that with a small training dataset, unseen PSNR still lags behind seen PSNR. Although more training images can alleviate this problem, CNeRV still cannot reconstruct unseen images with perfect fidelity in this work. This also limits its application for visual codec or compression methods.

A.4 NeRV Generalization Results

We provide more NeRV results on shuffled/sequential video frames in Table 14, with different number of training images on 'Big Buck Bunny'.

Table 15: **Interpolation** results on different datasets. We show PSNR of unseen images with pixel interpolation and embedding interpolation. Besides, we also show results of ground truth embedding

Dataset	Pixel Interpolation	GT Embedding	Embedding Interpolation
UVG	30.14	28.76	28.88
MCL	28.05	26.85	26.33
Bunny	27.64	26.98	24.94

A.5 Embedding interpolation

We first provide interpolation results in pixel space and embedding space, together with ground truth embedding results, as shown in Table 15. Note that our interpolated embedding shows comparable PSNR with ground truth embedding on unseen images, which clearly demonstrates the internal generalization of our content-adaptive embedding.

A.6 Main Results with MS-SSIM

We also provide a detailed main results, with MS-SSIM, in Table 16, 17, 18, 19, and 20.

A.7 Embedding Analysis

We also perform 3 measurements to compare the embeddings produced by CNeRV to prior work. We compute uniformity to examine the distribution of each set of embeddings on the hypersphere [51].

$$U = \log \mathop{\mathbb{E}}_{x,y \sim p_{\text{data}}} \left[e^{-t||(||f(x)||_2 - ||f(y)||_2)||_2^2} \right] \tag{5}$$

With the same metric, we compute distances between neighbors for each method, to measure the extent to which each method encodes similar representations for neighboring frames. We also compute normalized distance, which is the neighbor distance (average distance between neighboring embeddings) divided by the uniformity (average distance between all embeddings). This normalized distance gives a more accurate reflection of the extent to which a given method's embeddings reflect a semantic connection between neighboring frames.

We compute linear centered kernel alignment (CKA) [8] to compare the similarity between embeddings of different methods in a pairwise fashion, as introduced by [20]. To compute this, we first obtain the matrices containing the embeddings for two different methods, such as NeRV and CNeRV, which we represent X and Y. We then compute the Gram matrices of the embedding matrices: $K = XX^T$, $L = YY^T$. The CKA value is given by the normalized Hilbert-Schmidt Independence Criterion (HSIC) [16] in Equation 6.

$$\text{CKA}(K,L) = \frac{\text{HSIC}(K,L)}{\sqrt{\text{HSIC}(K,K)\text{HSIC}(L,L)}} \tag{6}$$

As Table 21 shows, while CNeRV's content adaptive embeddings are more tightly clustered than the embeddings for the other methods, CNeRV's neighboring embeddings are still

Table 16: Results on different **video datasets**

Method	Dataset	Embed Length	Total Size	PSNR Seen	PSNR Unseen ↑	PSNR Gap ↓	MS-SSIM Seen	MS-SSIM Unseen ↑	MS-SSIM Gap ↓
NeRV	UVG	480	64M	36.05	23.66	12.39	0.9823	0.7314	0.2509
CNeRV	UVG	480	64M	35.83	**28.76**	**7.07**	0.9789	**0.8739**	**0.105**
NeRV	Bunny	480	64M	33.53	16.46	17.07	0.919	0.4933	0.4257
CNeRV	Bunny	480	64M	33.83	**26.85**	**6.98**	0.9539	**0.8242**	**0.1297**
NeRV	MCL	480	64M	34.83	19.44	15.39	0.9815	0.5824	0.3991
CNeRV	MCL	480	64M	34.67	**26.98**	**7.69**	0.978	**0.8229**	**0.1551**

Table 17: Results on different **model sizes**

Method	Model Size	Embed Length	Total Size	PSNR Seen	PSNR Unseen ↑	PSNR Gap ↓	MS-SSIM Seen	MS-SSIM Unseen ↑	MS-SSIM Gap ↓
NeRV	Small	480	32M	31	16.72	14.28	0.8977	0.5289	0.3688
CNeRV	Small	480	32M	31.33	**26.41**	**4.92**	0.911	**0.8198**	**0.0912**
NeRV	Medium	480	64M	33.53	16.46	17.07	0.919	0.4933	0.4257
CNeRV	Medium	480	64M	33.83	**26.85**	**6.98**	0.9539	**0.8242**	**0.1297**
NeRV	Large	480	97M	35.32	16.04	19.28	0.9485	0.4737	0.4748
CNeRV	Large	480	97M	35.5	**27.08**	**8.42**	0.9682	**0.8235**	**0.1447**

Table 18: Results on different **video resolutions**

Method	Video Resolution	Embed Length	Total Size	PSNR Seen	PSNR Unseen ↑	PSNR Gap ↓	MS-SSIM Seen	MS-SSIM Unseen ↑	MS-SSIM Gap ↓
NeRV	240*480	480	60M	37.14	16.9	20.24	0.9929	0.5142	0.4787
CNeRV	240*480	480	60M	36.99	**27.97**	**9.02**	0.9923	**0.8532**	**0.1391**
NeRV	480*960	480	64M	33.53	16.46	17.07	0.919	0.4933	0.4257
CNeRV	480*960	480	64M	33.83	**26.85**	**6.98**	0.9539	**0.8242**	**0.1297**
NeRV	960*1920	480	67M	32.06	16.06	16	0.902	0.5496	0.3524
CNeRV	960*1920	480	67M	32.4	**26.15**	**6.25**	0.9057	**0.8118**	**0.0939**

Table 19: Results on different **training data size**

Method	Training Images	Embed Length	Total Size	PSNR Seen	PSNR Unseen ↑	PSNR Gap ↓	MS-SSIM Seen	MS-SSIM Unseen ↑	MS-SSIM Gap ↓
NeRV	1k	480	64M	34.6	12.57	22.03	0.9489	0.3495	0.5994
CNeRV	1k	480	64M	34.78	**26.41**	**8.37**	0.9754	**0.813**	**0.1624**
NeRV	2k	480	64M	33.53	16.46	17.07	0.919	0.4933	0.4257
CNeRV	2k	480	64M	33.83	**26.85**	**6.98**	0.9539	**0.8242**	**0.1297**
NeRV	4k	480	64M	32.78	20.68	12.1	0.9077	0.6994	0.2083
CNeRV	4k	480	64M	32.94	**27.75**	**5.19**	0.9274	**0.8396**	**0.0878**

Table 20: Results on different **image datasets**

Method	Dataset	Embed Length	Total Size	PSNR Seen	PSNR Unseen ↑	PSNR Gap ↓	MS-SSIM Seen	MS-SSIM Unseen ↑	MS-SSIM Gap ↓
NeRV	Celeb	240	33M	27.44	11.27	16.17	0.9548	0.4397	0.5151
CNeRV	Celeb	240	33M	27.42	**21.34**	**6.08**	0.9536	**0.7879**	**0.1657**
NeRV	Flower	240	35M	27	11.29	15.71	0.9028	0.2538	0.649
CNeRV	Flower	240	36M	27.04	**18.54**	**8.5**	0.91	**0.5491**	**0.3609**

relatively close to each other. CNeRV thus utilizes very little of the embedding space to generate meaningful and generalizable embeddings, and this is also consistent with embedding quantization results as in Figure 9.

Figure 11 further reflects the difference between CNeRV embeddings and those of other methods. Whereas the convolutional autoencoders and the pixel-wise neural representations generate somewhat similar embeddings, CNeRV is quite unique by contrast. This shows how radically different our method is from those already in the literature, in spite of its comparable performance and desirable properties.

Table 21: **Embedding distribution on hypersphere.** CNeRV embeddings tend to cluster very tightly together, with low uniformity compared to other methods. CNeRV neighbor frame embeddings are still reasonably close together, in stark contrast to NeRV.

Methods	Uniformity			Neighbor	Normalized
	Seen ↓	Unseen ↓	All ↓	Distance ↓	Distance ↓
FFN [44]	1.18	1.17	1.18	0.09	0.07
NeRF [31]	1.46	1.46	1.46	0.10	0.07
MLF [29]	1.39	1.38	1.38	0.14	0.10
ConvAE	2.00	1.97	1.99	0.14	**0.07**
ConvVAE	2.71	2.71	2.71	0.21	0.08
NeRV [5]	3.97	3.96	3.97	3.60	0.91
CNeRV	**0.23**	**0.23**	**0.23**	**0.03**	0.15

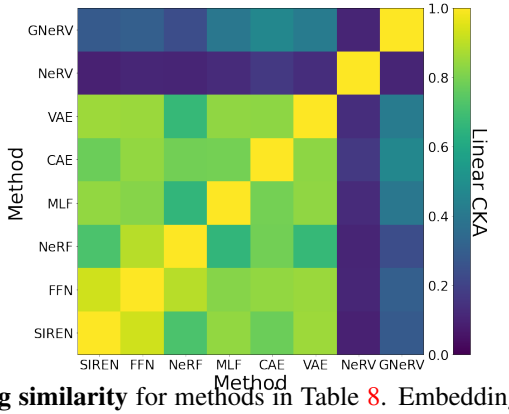

Figure 11: **Embedding similarity** for methods in Table 8. Embeddings of NeRV and CNeRV are quite different from other methods.