Boosting Neural Representations for Videos with a Conditional Decoder

Xinjie Zhang[1,2*] Ren Yang[2†] Dailan He[3] Xingtong Ge[4]
Tongda Xu[5] Yan Wang[5] Hongwei Qin[2] Jun Zhang[1†]
[1]The Hong Kong University of Science and Technology [2]SenseTime Research
[3]The Chinese University of Hong Kong [4]Beijing Institute of Technology
[5]Institute for AI Industry Research (AIR), Tsinghua University
xzhangga@connect.ust.hk, r.yangchn@gmail.com, hedailan@link.cuhk.edu.hk, xingtong.ge@bit.edu.cn
x.tongda@nyu.edu, wangyan202199@163.com, qinhongwei@sensetime.com, eejzhang@ust.hk

Abstract

Implicit neural representations (INRs) have emerged as a promising approach for video storage and processing, showing remarkable versatility across various video tasks. However, existing methods often fail to fully leverage their representation capabilities, primarily due to inadequate alignment of intermediate features during target frame decoding. This paper introduces a universal boosting framework for current implicit video representation approaches. Specifically, we utilize a conditional decoder with a temporal-aware affine transform module, which uses the frame index as a prior condition to effectively align intermediate features with target frames. Besides, we introduce a sinusoidal NeRV-like block to generate diverse intermediate features and achieve a more balanced parameter distribution, thereby enhancing the model's capacity. With a high-frequency information-preserving reconstruction loss, our approach successfully boosts multiple baseline INRs in the reconstruction quality and convergence speed for video regression, and exhibits superior inpainting and interpolation results. Further, we integrate a consistent entropy minimization technique and develop video codecs based on these boosted INRs. Experiments on the UVG dataset confirm that our enhanced codecs significantly outperform baseline INRs and offer competitive rate-distortion performance compared to traditional and learning-based codecs. Code is available at https://github.com/Xinjie-Q/Boosting-NeRV.

1. Introduction

Implicit neural representations (INRs) are gaining widespread interest for their remarkable capability in accurately representing diverse multimedia signals, including audios [40, 46], images [31, 38], and 3D scenes [30, 33, 37]. They typically use a compactly parameterized neural network to learn an implicit continuous mapping that translates coordinates into target outputs (*e.g.*, RGB values, density). This new-fashioned neural representation has opened up a plethora of potential applications, ranging from data inpainting [24, 38, 52] and signal compression [44, 53, 54] to advanced generative models [41, 42].

Given their simplicity, compactness and efficiency, several studies have suggested applying INRs to video compression. Unlike traditional [45, 50] and recent neural

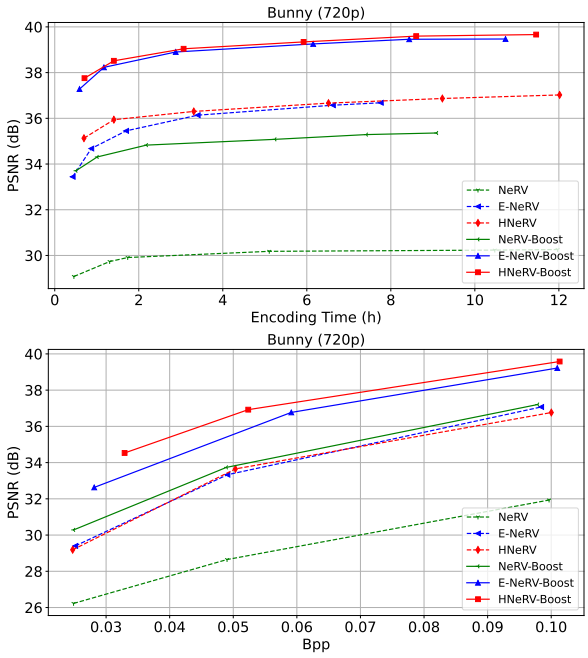


Figure 1. Video regression with different encoding time under 1.5M model size (left) and video compression with different model sizes(right). Our boosted methods achieve significantly better performance than the corresponding baselines.

*Work was done when Xinjie Zhang interned at SenseTime Research
†Corresponding author

[22, 25, 27, 39] video codecs that rely on a complex predictive coding paradigm with separate encoder and decoder components, NeRV [7] pioneers in representing videos as a function of the frame index t and formulating video compression as model-based overfitting and compression. This innovative method significantly simplifies the encoding and decoding processes. Built on this paradigm, a series of subsequent works [8, 9, 21, 23, 55] have devoted to designing more meaningful embeddings to improve the quality of video reconstruction.

However, there are several vital limitations hindering the potential of existing implicit video representations. **Firstly**, when decoding the t-th frame, the identity information in most works only relies on the t-th temporal embedding. This approach often struggles to align the intermediate features with the target frame. Although a few works [1, 14, 23] introduce the AdaIN [17] module to modulate intermediate features, this couples normalization and conditional affine transform. Its normalization operation might reduce the over-fitting capability of the neural network, resulting in limited performance gains (See Table 8). **Secondly**, while there are several studies [9, 21, 23] in refining NeRV's upsampling block for a more streamlined convolutional framework, the impact of activation layers on the model's representational ability remains underexplored. **Moreover**, most previous methods rely on the L2 loss [9, 55] or a combination of L1 and SSIM losses [7, 8, 21, 23] to overfit videos, but they often fail to preserve high-frequency information (*e.g.*, edges and fine details within each frame), thereby degrading the reconstruction quality (See Table 8). **Finally**, most video INRs follow NeRV to employ a three-step model compression pipeline (*i.e.*, pruning, quantization, and entropy coding) in the video compression task. Nevertheless, these components are optimized separately, which prevents INRs from achieving optimal coding efficiency. Although a few works [14, 28] have explored the joint optimization of quantization and entropy coding, they face critical challenges arising from inconsistencies in the entropy models employed during both training and inference stages, leading to sub-optimal rate-distortion (RD) performance (See Table 1 and Fig. 7). Hence, we argue that there is great potential to boost the performance by overcoming the challenges we mentioned above.

To this end, we propose a universal boosting framework based on a conditional decoder to deeply explore the representation performances of existing video INRs. **Firstly**, we introduce a temporal-aware affine transform (TAT) module that discards normalization to better align intermediate features with the target frame. It is achieved by using a pair of affine parameters $(\boldsymbol{\gamma}, \boldsymbol{\beta})$ derived from temporal embeddings. We further incorporate a residual block with two TAT layers to facilitate both feature alignment and information retention. By strategically alternating upsampling blocks and TAT residual blocks, our conditional decoder significantly boosts the model's representation capabilities. **Secondly**, as shown in Fig. 4, we find that the GELU layer in the NeRV-like block activates only a limited number of feature maps. To address this issue, we introduce a sinusoidal NeRV-like (SNeRV) block to replace the GELU layer with a SINE layer to generate more diverse features. By using a small kernel size in the SNeRV block and placing more SNeRV blocks in later upsampling stages, we achieve a more balanced parameter distribution across the network, which helps improve the model's capacity. **Moreover**, we integrate L1, MS-SSIM, and frequency domain losses as our optimization objectives during overfitting a video. This trio of loss functions promises to preserve intricate details in the reconstructed videos. **Finally**, we advocate a consistent entropy minimization (CEM) technique based on a network-free Gaussian entropy model with tiny metadata transmission overhead, which not only ensures the consistency of training and inference, but also captures the interrelationships between elements in each weight or embedding to accurately estimate the probability distribution.

In summary, our main contributions are three-fold: (1) We develop a universal boosting framework based on a novel temporal-aware conditional decoder to effectively improve the representation capabilities and accelerate the convergence speed compared to existing video INRs, shown in Fig. 1. (2) We further design a consistent entropy minimization scheme based on a parameter-efficient Gaussian entropy model to eliminate the discrepancy between training and inference. (3) Extensive experimental results demonstrate that our boosted video INRs achieves a remarkable performance improvement against various baselines on multiple tasks, including video regression, compression, inpainting and interpolation. Comprehensive ablations and analyses demonstrate the effectiveness of each proposed component.

2. Related Work

Implicit Video Representation. Recently, implicit video representations have engaged increasing interest due to their wide-ranging potential applications, such as video compression, inpainting and interpolation. Roughly, existing implicit video representations can be classified into two categories: (i) *Index-based* methods [1, 7, 14, 23, 28] take content-independent time vectors and/or spatial coordinates as inputs. They only rely on the neural network to store all video information. (ii) *Hybrid-based* methods [8, 9, 21, 47, 55] take content-relevant embeddings as inputs to provide a visual prior for the network, which reduces the learning difficulty of the model and thereby enhances its representation performance. These content-relevant embeddings can be derived from each frame using specific encoders [9, 47, 55] or generated through random initializa-

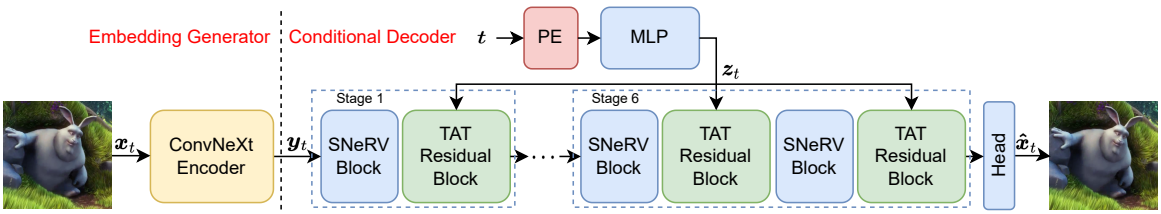


Figure 2. Our proposed HNeRV-Boost framework with the conditional decoder. The content-relevant embedding $\boldsymbol{y}_t$ expands its channel dimensions in stage 1 and upsamples in stages 2 to 6. The final three stages stack two SNeRV blocks with small kernel sizes to get fewer parameters, where the former upsamples features and the latter refines the upsampled features.

tion and backpropagation [8, 21]. In this paper, we focus on designing a universal conditional decoder framework to effectively boost the representation performance of existing video INRs, thus setting a new benchmark in the field.

Network Conditioning. Early methods in conditional feature modulation rely on conditional normalization (CN). CN replaces the feature-wise transformation in normalization layers with affine parameters generated from external information, which has shown its effectiveness in applications like style transfer [11, 13, 17], semantic image synthesis [34], and denoising [18]. A pivotal contribution by Perez *et al.* [36] demonstrates that it is possible to directly modulate intermediate features in a network without undergoing normalization. This technique has been widely used in super-resolution [16, 48], compression [43], and restoration [49]. Instead of using CN to modify the distribution of intermediate features as in previous INR studies [1, 14, 23], we introduce a conditional affine transformation without normalization to achieve a more precise alignment of intermediate features with the target frame, potentially improving the reconstruction quality of existing video INRs.

INR for Video Compression. As implicit neural networks generally fit videos using the model weights, it offers a novel perspective on video compression by translating it into model compression. Most video INRs [1, 20, 21, 23] follow the three-step compression pipeline of NeRV [7]: (i) model pruning, such as global unstructured pruning, with fine-tuning to reduce the model size; (ii) post-training quantization or quantization-aware training to lower the precision of each weight; (iii) entropy coding to minimize the statistical correlation of coded symbols. Unfortunately, optimizing these components separately leads to sub-optimal coding efficiency. Thus, Gomes *et al.* [14] and Maiya *et al.* [28] have applied entropy minimization [2, 3, 32] to improve the video INR compression. During training, they estimate the bitrate of quantized weights using a small neural network to model each weight's distribution. But in the inference stage, this neural entropy model is replaced with either a context-adaptive binary arithmetic coder (CABAC) or an arithmetic coder using a fixed statistical frequency table. This switch in entropy models between training and inference leads to the sub-optimal RD performance. Moreover, these methods focus on compressing model weights, overlooking the significance of content-relevant embeddings in hybrid-based video INRs. To bridge these gaps, we propose a consistent entropy minimization technique to unify the weight and embedding compression. Consequently, our scheme is exceptionally suited for any video INR compression, marking a significant stride in this domain.

3. Method

As shown in Fig. 2, our proposed boosted video representation architecture comprises two primary components: an embedding generator and a conditional decoder. The choice of an embedding generator depends on the specific video INR model. For clarity and ease of understanding, we take the hybrid-based representation model HNeRV [9] as an example. It is worth mentioning that our boosting framework can be easily generalized to other representation models (*e.g.*, NeRV [7], E-NeRV [23]) by selecting appropriate embedding generators, with more details given in appendix.

3.1. Overview

Let $\mathcal{X} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_T\}$ denote a video sequence, where $\boldsymbol{x}_t \in \mathbb{R}^{H\times W\times 3}$ is the frame at timestamp t with height H and width W. Following HNeRV [9], we use ConvNeXt blocks [26] to build a video-specific encoder E that maps each individual frame $\boldsymbol{x}_t$ to a compact embedding $\boldsymbol{y}_t \in \mathbb{R}^{h\times w\times d}$ with d representing the embedding dimension. Take a 1080×1920 video as an example, we set $h = \frac{H}{120}$ and $w = \frac{W}{120}$. Note that the identity information in the original HNeRV is confined to the input embedding $\boldsymbol{y}_t$. We instead introduce a frame reconstruction network F conditioned on the temporal embedding $\boldsymbol{z}_t$ to effectively align the intermediate features with identity information. Specifically, the frame index t, normalized to (0, 1], is initially mapped to a high-dimensional space using a regular frequency positional encoding function $\mathrm{PE}(\cdot)$ [30], and then processed through a small MLP network M to produce the temporal embedding $\boldsymbol{z}_t$. As depicted in Fig. 2, when the input embedding $\boldsymbol{y}_t$ passes through the proposed SNeRV block, the size of the embedding usually increases step by step. Meanwhile, the temporal embedding $\boldsymbol{z}_t$ modulates the intermediate features in the proposed TAT residual block.

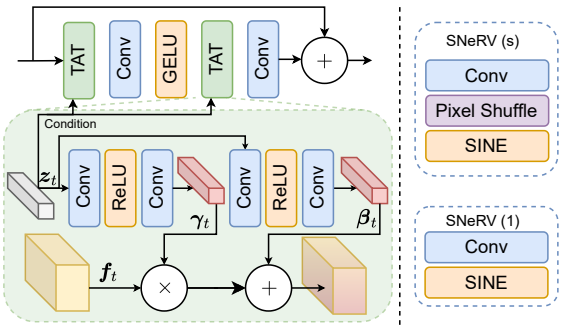


Figure 3. (Left) Illustration of the temporal-aware affine transform layer and residual block. The TAT layer takes the temporal embeddings $\boldsymbol{z}_t$ to produce channel-wise scaling and shifting parameters $\boldsymbol{\gamma}_t$ and $\boldsymbol{\beta}_t$. As a result, the affine transformation is performed to the intermediate features of the previous layer. (Right) The architecture of the sinusoidal NeRV-like block. When the stride s of the convolutional layer is larger than 1, it includes a pixelshuffle layer.

In the end, a header layer transforms the output features of the last stage into the reconstructed frame $\hat{\boldsymbol{x}}_t$. Formally, the overall representation procedure is formulated as

$$\boldsymbol{y}_t = E(\boldsymbol{x}_t;\boldsymbol{\phi}), \boldsymbol{z}_t = M(\mathrm{PE}(t);\boldsymbol{\psi}), \hat{\boldsymbol{x}}_t = F(\boldsymbol{y}_t, \boldsymbol{z}_t;\boldsymbol{\theta}) \tag{1}$$

where $\boldsymbol{\phi}$, $\boldsymbol{\psi}$, and $\boldsymbol{\theta}$ are the learnable parameters of the video-specific encoder, the temporal embedding generator, and the frame reconstruction network, respectively. The positional encoding function $\mathrm{PE}(t)$ is defined as $(\sin(b^0\pi t), \cos(b^0\pi t), \cdots, \sin(b^{l-1}\pi t), \cos(b^{l-1}\pi t))$ with hyperparameters b and l.

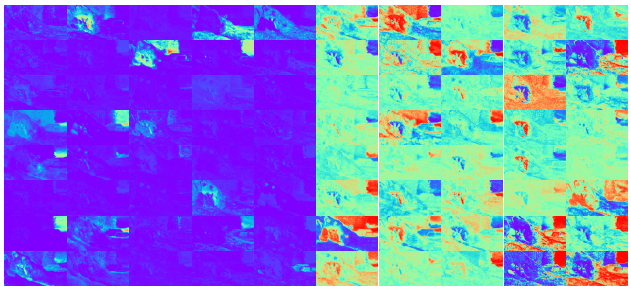
3.2. Temporal-aware Conditional Decoder

Temporal-aware Affine Transform. Given the intermediate features $\boldsymbol{f}_t$ and temporal embedding $\boldsymbol{z}_t$, previous works [1, 14, 23] first adopt a small MLP network to learn the channel-wise mean $\boldsymbol{\mu}_t$ and variance $\boldsymbol{\sigma}_t$ of the target frame, and then use the AdaIN [17] module to change the distribution of intermediate features:

$$\begin{aligned} (\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t) &= \mathrm{MLP}(\boldsymbol{z}_t), \\ \mathrm{AdaIN}(\boldsymbol{f}_t, \boldsymbol{\mu}_t, \boldsymbol{\sigma}_t) &= \boldsymbol{\sigma}_t\left(\frac{\boldsymbol{f}_t - \boldsymbol{\mu}(\boldsymbol{f}_t)}{\boldsymbol{\sigma}(\boldsymbol{f}_t)}\right) + \boldsymbol{\mu}_t, \end{aligned} \tag{2}$$

where $\boldsymbol{\mu}(\boldsymbol{f}_t)$ and $\boldsymbol{\sigma}(\boldsymbol{f}_t)$ are computed across spatial locations. However, the AdaIN module couples normalization and conditional affine transformation. The normalization operation typically serves to prevent the overfitting in neural networks, which conflicts with video INRs that utilize the overfitting to represent the data.

To overcome this limitation, we present a temporal affine transform (TAT) layer without normalization and its associated residual block to unleash the potential of feature alignment. Fig. 3 (left) illustrates the details of our TAT residual block, which is inspired by the network design of [48].



(a) GELU activation (b) SINE activation

Figure 4. Visual comparisons of intermediate features from different activation functions in the HNeRV-Boost model. We select the first 40 channel features from the last NeRV-like block on the first frame generation of the Bunny video.

Based on the external temporal embedding $\boldsymbol{z}_t$, the TAT layer learns to generate a set of channel-wise affine parameters $(\boldsymbol{\gamma}_t, \boldsymbol{\beta}_t)$ for the intermediate features $\boldsymbol{f}_t$. Within this layer, the feature transformation is expressed as:

$$\mathrm{TAT}(\boldsymbol{f}_t|\boldsymbol{\gamma}_t, \boldsymbol{\beta}_t) = \boldsymbol{\gamma}_t \boldsymbol{f}_t + \boldsymbol{\beta}_t, \tag{3}$$

By inserting the TAT residual block into existing video INRs, these aligned intermediate features can significantly enhance the models' overfitting ability.

Sinusoidal NeRV-like Block. Previous upsampling blocks [7, 8, 21, 23] commonly use GELU as their default activation function. However, our analysis of feature maps, as visualized in Fig. 4, reveals a limitation: the GELU layer tends to activate only a limited number of feature maps, whereas those activated by the SINE layer are more diverse and focus on different regions. This motivates us to introduce the sinusoidal NeRV-like (SNeRV) block. As shown in Fig. 3 (right), our SNeRV block has two types, where the one with a pixelshuffle layer serves for upsampling features.

Besides, we notice that the HNeRV blocks in the final three stages use a 5×5 kernel size, resulting in about 2.7× more parameters than a 3×3 kernel. However, reducing the kernel size directly adversely affects the regression performance. To resolve this difficulty, we substitute a single HNeRV block with a 5×5 kernel for two SNeRV blocks with a 3×3 kernel, setting the stride of the second SNeRV block to 1. This alteration enables us to maintain a similar level of video reconstruction quality with fewer parameters.

Loss Function. The objective of video INRs is to reduce the distortion between the original frame $\boldsymbol{x}_t$ and reconstructed frame $\hat{\boldsymbol{x}}_t$. Although the L1 loss is adept at preserving brightness as well as color, it falls short in maintaining high-frequency details. Therefore, we integrate a combination of the MS-SSIM and frequency domain losses into the L1 loss, ensuring a more comprehensive capture of high-frequency regions. For the frequency domain loss, we apply the fast Fourier transform (FFT) to both $\boldsymbol{x}_t$ and $\hat{\boldsymbol{x}}_t$, and then compute their L1 loss. The complete distortion

loss function is formulated as follows:

$$\mathcal{L}_d = \mathcal{L}_1(\mathrm{FFT}(\boldsymbol{x}_t), \mathrm{FFT}(\hat{\boldsymbol{x}}_t)) + \lambda\alpha\mathcal{L}_1(\boldsymbol{x}_t, \hat{\boldsymbol{x}}_t) + \lambda(1-\alpha)(1-\mathcal{L}_{\text{MS-SSIM}}(\boldsymbol{x}_t, \hat{\boldsymbol{x}}_t)) \tag{4}$$

Here, λ and α are hyperparameters used to balance the weight of each loss component.

3.3. Consistent Entropy Minimization

After overfitting the video, we propose a consistent entropy minimization technique to refine the compression pipeline in [14, 28]. As indicated in Table 1, Gomes *et al.* [14] and Maiya *et al.* [28] primarily concentrate on the quantization of model weights, overlooking the significance of embedding compression in improving the RD performance for hybrid-based video INRs. Furthermore, these methods use different entropy models in training and inference stages. Specifically, a small neural network is employed as a surrogate entropy model during training to estimate the bitrate, while in the inference phase, it is replaced with either CABAC or an arithmetic coder using a fixed statistical frequency table. This strategy aims to minimize transmission overhead from numerous small proxy networks. However, the discrepancy between the estimated bitrate by the surrogate model and the actual bitrate may mislead network optimization, resulting in sub-optimal coding efficiency. To overcome these shortcomings, our study introduces two key modifications to the entropy minimization pipeline: (i) applying a symmetric/asymmetric quantization scheme to model weights/embeddings, and (ii) introducing a network-free Gaussian entropy model with tiny metadata overhead to ensure the consistency during training and inference.

Quantization. Since the model weights are empirically observed to be distributed symmetrically around zero [6, 12], a symmetric scalar quantization scheme with a trainable scale parameter ς is used for weights:

$$Q(x) = \lfloor \frac{x}{\varsigma} \rceil,\ Q^{-1}(x) = x \times \varsigma \tag{5}$$

Conversely, for embeddings with skewed distributions, we adopt an asymmetric activation quantization scheme where both the scale parameter ς and the offset parameter η are learned during training:

$$Q(x) = \lfloor \frac{x-\eta}{\varsigma} \rceil,\ Q^{-1}(x) = x \times \varsigma + \eta \tag{6}$$

Given the non-differentiable nature of the quantization operation during training, we leverages a mixed quantizer outlined in [14, 28] to allow end-to-end optimization. Specifically, the rounding operation is substituted with uniform noise $\mathcal{U}(-\frac{1}{2}, \frac{1}{2})$ for entropy calculation, while the straight-through estimator (STE) is applied for distortion calculation when computing the gradient for the rounding operation.

Table 1. Comparisons between different entropy minimization techniques in INR compression.

Method	Quantization		Entropy Model	
	Weight	Embedding	Training	Inference
Gomes *et al.*[14]	Asymmetric	-	Neural network	CABAC
Maiya *et al.*[28]	Symmetric	-	Neural network	Fixed frequency table
CEM (ours)	Symmetric	Asymmetric	Network-free Gaussian entropy model	

Network-free Gaussian Entropy Model. We model the probability of the quantized embedding $\hat{\boldsymbol{y}}_t$ with a Gaussian distribution:

$$p(\hat{\boldsymbol{y}}_t) = \prod_i \left(\mathcal{N}(\mu_{\boldsymbol{y}_t}, \sigma^2_{\boldsymbol{y}_t}) * \mathcal{U}(-\frac{1}{2}, \frac{1}{2})\right)(\hat{y}^i_t) \tag{7}$$

where $\mu_{\boldsymbol{y}_t}$ and $\sigma^2_{\boldsymbol{y}_t}$ are the mean and variance of the embedding $\boldsymbol{y}_t$. $*$ denotes convolution. Similarly, we independently compute the mean and variance of each weight as the entropy model parameters for weight compression. This novel entropy model brings two main advantages. On the one hand, when compared with the neural entropy model of [14, 28] that only captures the relationship between elements of each kernel, our model can capture the global relationships within all elements in the weight, which facilitates the accurate estimation of the probability distribution. On the other hand, thanks to only transmitting two scalar values for each weight/embedding, our entropy model can get rid of the surrogate position and directly provide the probability distribution for arithmetic coding in the inference stage.

Optimization Objective. The goal of INR compression is to achieve high reconstruction quality with minimal bitrate consumption. To this end, we incorporate an entropy regularization term $\mathcal{L}_r$ to encourage smaller compressed models. In order to control the whole compression ratio, we introduce R_{target} into the regularization term. Once R_{target} is satisfied, $\mathcal{L}_r$ diminishes to zero:

$$\mathcal{L} = \mathcal{L}_d + \kappa\mathcal{L}_r = \mathcal{L}_d + \kappa\mathrm{ReLU}(R - R_{target}) \tag{8}$$

Here, κ is a hyperparameter balancing the compression rate and distortion. R is calculated as $\frac{\sum_{t=1}^{T} R(\hat{\boldsymbol{y}}_t) + R(\hat{\boldsymbol{\theta}}) + R(\hat{\boldsymbol{\psi}})}{T \times H \times W}$, where $R(\hat{\boldsymbol{y}}_t)$ denotes the estimated bitrate of the quantized embedding $\hat{\boldsymbol{y}}_t$ and $R(\hat{\boldsymbol{\theta}})/R(\hat{\boldsymbol{\psi}})$ represents the estimated bitrate of the quantized weights $\hat{\boldsymbol{\theta}}/\hat{\boldsymbol{\psi}}$. For R_{target}, we define it as $B_{avg}\frac{\sum_{t=1}^{T} \mathrm{Numel}(\boldsymbol{y}_t) + \mathrm{Numel}(\boldsymbol{\theta}) + \mathrm{Numel}(\boldsymbol{\psi})}{T \times H \times W}$, in which B_{avg} indicates the average bit-width of the compressed INR and $\mathrm{Numel}(\cdot)$ means the amount of parameters.

4. Experiments

4.1. Experimental Setup

Dataset. We evaluate the effectiveness of our framework on multiple benchmarks with various kinds of video contents, including Bunny [19] (720×1280 with 132 frames), UVG [29] (7 videos at 1080×1920 with length of 600 or 300), and DAVIS validation [35] (20 videos at 1080×1920).

Table 2. Average PSNR on UVG with various model sizes.

Size	3M	5M	10M	15M	Avg.
NeRV [7]	31.11	32.44	34.20	35.18	33.23
NeRV-Boost	**32.76**	**33.76**	**35.28**	**35.94**	**34.43**
E-NeRV [23]	31.51	33.99	34.42	35.32	33.81
E-NeRV-Boost	**33.40**	**34.31**	**35.58**	**36.27**	**34.89**
HNeRV [9]	32.47	33.40	34.70	35.10	33.92
HNeRV-Boost	**33.89**	**35.06**	**36.49**	**37.29**	**35.68**

Table 3. Detailed PSNR of each video on the UVG dataset with 3M model size.

Video	Beauty	Bosph.	Honey.	Jockey	Ready.	Shake.	Yacht.	Avg.
NeRV [7]	33.14	32.74	37.18	30.99	23.97	33.06	26.72	31.11
NeRV-Boost	**33.55**	**34.51**	**39.04**	**32.82**	**26.08**	**34.54**	**28.76**	**32.76**
E-NeRV [23]	33.29	33.87	38.88	28.73	23.98	34.45	27.38	31.51
E-NeRV-Boost	**33.75**	**35.62**	**39.61**	**32.39**	**27.75**	**35.48**	**29.23**	**33.40**
HNeRV [9]	33.36	33.62	39.17	32.31	25.60	34.90	28.33	32.47
HNeRV-Boost	**33.80**	**36.11**	**39.65**	**34.28**	**28.19**	**35.88**	**29.33**	**33.89**

Table 4. PSNR on the Bosphorus video with different epochs.

Epoch	300	600	1200	1800	2400
NeRV [7]	32.74	33.00	33.20	33.27	33.32
NeRV-Boost	**34.51**	**34.73**	**34.89**	**34.97**	**35.02**
E-NeRV [23]	33.87	34.19	34.40	34.50	34.56
E-NeRV-Boost	**35.62**	**35.92**	**36.16**	**36.27**	**36.32**
HNeRV [9]	33.62	34.15	34.35	34.41	34.46
HNeRV-Boost	**36.11**	**36.33**	**36.52**	**36.59**	**36.64**

Evaluation Metrics. Two popular image quality assessment metrics, namely, PSNR and MS-SSIM, are used to evaluate the distortion between the reconstructed and original frames. We use bits per pixel (bpp) to measure the bitrate of video compression.

Implementation Details. We select three typical INR methods (*i.e.*, NeRV [7], E-NeRV [23], and HNeRV [9]) as our baselines. Then we enhance them with our proposed framework. To fit 720p and 1080p videos, we set the stride list as (5,2,2,2,2) and (5,3,2,2,2), respectively. We use $b = 1.25$ and $l = 80$ as our default setting in the position encoding. For the distortion loss in Equation 4, λ and α are set as 60 and 0.7, receptively. During overfitting, we set the batch size as 1 and adopt Adan [51] as the optimizer with cosine learning rate decay [9], in which the number of warm-up epochs is 10% of total fitting epochs. The learning rates for the boosted E-NeRV and NeRV/HNeRV are $1.5e^{-3}$ and $3e^{-3}$, respectively. Baseline models are implemented using open-source codes, and experiments are conducted on one NVIDIA GTX 1080Ti GPU using PyTorch, with 3M model size and 300 epochs unless otherwise denoted. For more details about different tasks, please refer to the supplementary material.

4.2. Video Regression

Table 2 show the regression performance of various methods on the UVG dataset at different scales. It is evident that our boosted methods achieve superior reconstruction quality over the corresponding baselines. As detailed in Table 3, the improvements are consistent across all test videos in the UVG dataset. For instance, compared to the NeRV, E-NeRV, and HNeRV baselines on the ReadySetGo video, our boosted versions exhibit considerable improvements of about 2.11dB, 3.77dB, and 2.59dB, respectively. In Table 4, we offer a comparison of regression performance between the boosted versions and baselines on the Bosphorus video across different fitting epochs. Notably, our boosted versions at 300 epochs outperform the baselines at 2400 epochs. Furthermore, as shown in Fig. 1 (top), our boosted versions at minimal training time surpass the baselines at their maximum training time by a significant margin on the Bunny video, which indicates the superiority of our method in accelerating convergence speed and improving the representation capabilities.

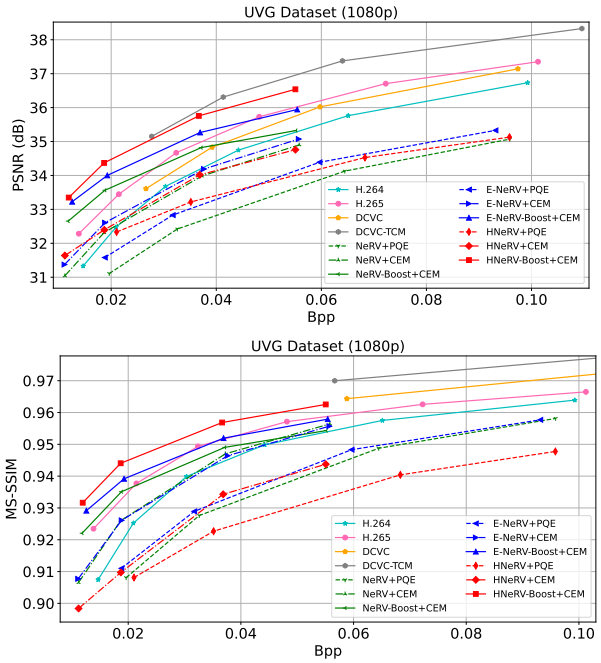


Figure 5. Rate-distortion curves of our boosted approaches and different baselines on the UVG dataset in PSNR and MS-SSIM. PQE denotes the three-step compression pipeline of NeRV.

Table 5. Complexity comparison at resolution 1920×1080. The decoding latency is evaluated by an NVIDIA V100 GPU.

Method	Params ↓	Decoding time ↓	FPS ↑
DCVC [22]	35.2M	35590ms	0.028
DCVC-TCM [39]	40.9M	470ms	2.12
NeRV [7]	3.04M	7ms	135.64
NeRV-Boost	3.06M	23ms	43.54
E-NeRV [23]	3.01M	18ms	54.75
E-NeRV-Boost	3.03M	53ms	18.74
HNeRV [9]	3.05M	41ms	24.22
HNeRV-Boost	3.06M	76ms	13.15

4.3. Video Compression

To assess video compression performance, we follow [9] to train a model for each video, rather than encoding all videos together using a single network as in [1, 7]. After model overfitting, we fine-tune these models using our en-

Table 6. Video inpainting results on the DAVIS validation dataset in PSNR. Mask-S and Mask-C refers to disperse and central mask scenarios, respectively.

Video	Mask-S						Mask-C					
	NeRV	NeRV-Boost	E-NeRV	E-NeRV-Boost	HNeRV	HNeRV-Boost	NeRV	NeRV-Boost	E-NeRV	E-NeRV-Boost	HNeRV	HNeRV-Boost
Blackswan	27.06	**30.46**	29.53	**31.34**	30.20	**34.10**	24.11	**26.89**	26.38	**27.88**	26.45	**29.18**
Bmx-trees	26.77	**30.16**	27.75	**30.86**	29.05	**32.99**	22.43	**25.14**	23.79	**26.66**	22.28	**22.28**
Breakdance	25.48	**28.46**	26.97	**30.57**	26.34	**33.10**	20.16	**22.28**	22.15	**22.15**	20.23	**20.24**
Camel	23.70	**26.09**	25.70	**27.56**	26.13	**31.08**	21.21	**23.16**	22.62	**23.55**	17.74	**19.81**
Car-roundabout	23.92	**28.25**	26.32	**29.43**	28.64	**31.90**	21.24	**23.53**	22.73	**24.51**	21.71	**22.36**
Car-shadow	26.58	**32.40**	30.63	**33.00**	31.01	**35.85**	23.07	**24.13**	23.21	**24.10**	21.05	**23.65**
Cows	22.17	**24.77**	23.92	**26.41**	24.68	**28.30**	20.48	**22.39**	21.88	**23.13**	21.82	**24.14**
Dance-twirl	25.29	**28.49**	27.42	**29.38**	28.74	**30.79**	21.17	**23.14**	22.40	**23.34**	21.06	**21.77**
Dog	29.29	**31.97**	31.72	**32.79**	28.80	**33.87**	25.37	**27.02**	27.07	**28.25**	24.16	**24.66**
Drift-chicane	34.09	**39.94**	39.26	**41.60**	38.52	**43.32**	27.52	**28.01**	29.81	**31.52**	23.40	**27.44**
Drift-straight	26.78	**32.26**	29.53	**33.19**	30.81	**36.16**	22.76	**26.00**	24.69	**27.12**	18.88	**21.49**
Goat	24.04	**26.30**	25.34	**27.21**	26.91	**30.59**	22.03	**23.90**	23.43	**24.56**	23.06	**25.10**
Horsejump-high	25.74	**30.39**	29.27	**31.26**	29.31	**30.86**	21.54	**23.46**	23.06	**23.93**	20.72	**23.16**
Kite-surf	29.34	**34.18**	32.87	**35.16**	33.49	**37.08**	23.92	**27.22**	26.71	**28.87**	24.73	**27.49**
Libby	29.81	**34.24**	31.39	**34.95**	28.66	**37.35**	25.71	**28.14**	26.91	**28.95**	23.39	**26.96**
Motocross-jump	29.82	**37.36**	34.15	**36.92**	28.27	**36.42**	26.19	**29.65**	28.75	**29.30**	22.36	**26.25**
Paragliding-launch	29.03	**31.40**	30.62	**32.28**	30.99	**33.64**	25.95	**26.97**	26.65	**27.41**	26.00	**28.07**
Parkour	24.74	**27.19**	25.62	**27.54**	26.34	**28.79**	22.32	**24.48**	22.99	**24.43**	19.06	**20.55**
Scooter-black	23.35	**27.75**	26.46	**29.07**	28.41	**30.42**	19.24	**21.77**	20.99	**22.14**	18.94	**19.86**
Soapbox	27.20	**30.56**	28.83	**31.44**	30.30	**32.95**	22.29	**25.00**	23.82	**25.51**	17.98	**19.20**
Average	26.71	**30.63**	29.17	**31.60**	29.28	**33.48**	22.94	**25.11**	24.50	**25.87**	21.75	**23.68**

tropy minimization method for 100 epochs with the initial learning rate $5e{-}4$ and cosine learning rate decay. We empirically choose B_{avg} as 4 bits to optimize the RD trade-off. For the baselines, both the three-step compression pipeline of NeRV and our CEM technique are applied for comparison. Besides, we compare our boosted models with traditional codecs (H.264 [50], H.265 [45]) and state-of-the-art (SOTA) learning-based codecs (DCVC [22], DCVC-TCM [39]), where H.264 and H.265 are tested using FFmpeg with the *veryslow* preset and enabling B frames.

Fig. 5 presents the RD curves of these methods on the UVG dataset. Our boosted models offer remarkable improvements over their corresponding baselines, indicating the generalization of our framework. Notably, the original HNeRV shows limited robustness in compression, which constrains the efficacy of content-relevant embeddings, especially at higher bitrates. On the contrary, with our modifications, the boosted HNeRV consistently surpasses DCVC, H.265, H.264, and other INR methods across all bitrates in terms of PSNR, which underscores the ability of our framework in amplifying the advantages of the INR model itself. Additionally, baselines with CEM outperform those using the three-step compression, highlighting the effectiveness of our compression technique in enhancing RD performance. The complexity comparison of video decoding with two SOTA neural codecs is shown in Table 5.

4.4. Video Inpainting

In this section, we evaluate video inpainting on the DAVIS validation dataset using both disperse [9] and central masks [55]. For the disperse mask scenario, each frame is overlaid with five uniformly distributed square masks of width 50. In the central mask scenario, a single rectangular mask, spanning one quarter of the frame's width and height, is centrally placed. Consistent with [9], the distortion loss during training is calculated only for the non-masked pixels. During inference, the masked regions are reconstructed using the output from video INRs.

Table 7. Video interpolation results on the UVG dataset in PSNR.

Video	Beauty	Bosph.	Honey.	Jockey	Ready.	Shake.	Yacht.	Avg.
NeRV [7]	**31.26**	32.21	36.84	**22.24**	**20.05**	32.09	26.09	28.68
NeRV-Boost	31.06	**34.28**	**38.83**	21.74	19.88	**32.58**	**27.07**	**29.35**
E-NeRV [23]	31.25	33.36	38.62	**22.35**	20.08	**32.82**	26.74	29.32
E-NeRV-Boost	**31.35**	**35.01**	**39.24**	21.96	**20.45**	32.75	**27.79**	**29.79**
HNeRV [9]	31.42	34.00	39.07	23.02	20.71	32.58	26.74	29.65
HNeRV-Boost	**31.61**	**36.16**	**39.38**	**23.14**	**21.61**	**32.94**	**28.01**	**30.41**

As shown in Table 6, our boosted versions significantly improve the inpainting performance of the original baselines. Under the disperse mask case, we observe an average improvement of 3.92dB, 2.43dB, and 4.2dB for NeRV, E-NeRV, and HNeRV, respectively. Even in the more challenging central mask scenario, our enhanced versions still achieve improvements of 2.17dB, 1.37dB, and 1.93dB.

4.5. Video Interpolation

We evaluate the video interpolation performance of our boosted models on the UVG dataset. In this experiment, we use the odd-numbered frames of each video as the training set and the even-numbered frames as the test set. Following the approach in [23], we adjust the frequency value b to 1.05 for achieving better interpolation results while maintaining robust regression performance on the training set. The quantitative outcomes are detailed in Table 7, These results indicate that our boosted models outperform the baselines in terms of overall interpolated quality.

4.6. Ablation Study

Feature Modulation. We conduct a sets of ablation studies on Bunny to evaluate the effectiveness of our TAT module. In the first variant (V1), the TAT residual blocks are removed from our boosted INR models. As Table 8 shows,

Table 8. Ablation studies for different boosting components on the Bunny video over 300 epochs, with results presented in PSNR.

Variant	NeRV-Boost	E-NeRV-Boost	HNeRV-Boost
Ours	**37.25**	**40.07**	**41.09**
(V1) w/o TAT	34.63	35.75	39.12
(V2) w/ AdaIN	35.59	39.51	38.03
(v3) w/ SAF	34.28	39.62	40.93
(V4) w/ GELU	34.85	38.34	41.00
(V5) w/ L2	35.32	38.55	40.19
(V6) w/ L1+SSIM	36.28	39.34	41.00
(V7) w/ L1	34.75	37.76	40.37
(V8) w/ L1+MS-SSIM	36.12	38.66	40.49
(V9) w/ L1+freq.	37.12	39.60	41.08
(V10) w/ L1+SSIM+freq.	36.99	40.05	41.01

Table 9. Ablation studies for various upsampling blocks in the HNeRV-Boost framework on the Bunny video. GELU and SINE represent the activation function employed in different blocks. STD refers to the standard deviation of the model parameters' distribution, where a lower STD value signifies a more uniform distribution of model parameters.

Block	NeRV [7]	E-NeRV [23]	FFNeRV [21]	HNeRV [9]	SNeRV
GELU	39.61	39.26	39.33	40.77	**41.00**
SINE	40.35	39.99	40.06	40.93	**41.09**
STD	0.225	0.208	0.176	0.047	0.045

this variant experiences an average drop of 2.97dB, implying the significance of using temporal embeddings to modulate intermediate features for accurate frame generation. In the second variant (V2), the TAT block is replaced with an AdaIN [17] module, resulting in an average decrease in PSNR by 1.76dB and 1.19dB. It suggests that integrating normalization into the conditional affine transformation limits the overfitting capabilities of INR models to some degree. Furthermore, we compare the spatially-adaptive fusion (SAF) [15] block. The results in variant V3 imply the superior temporal alignment capability of our TAT module.

NeRV-like Blocks. Table Table 8 investigates the effect of activation layers in SNeRV blocks. By replacing the SINE layer with a common GELU layer (Variant V4), we find that the periodic inductive bias of the SINE layer is beneficial to the reconstruction quality. Subsequently, we explore different convolution configurations. As shown in Table 9 and Fig. 6, integrating SINE in the NeRV, E-NeRV, and FFNeRV blocks results in a significant performance improvement of about 0.7dB. However, only modest improvements are observed in the HNeRV and SNeRV modules. This is attributed to the smaller STD values and more evenly distributed parameters within the HNeRV and SNeRV modules, helping the layers near the output to have sufficient capacity to store high-resolution video content and details. Thus, these modules efficiently fit videos without requiring elaborate feature extraction.

Loss Function. Different loss functions are compared in Table 8. These results demonstrate that integrating the MS-SSIM and frequency domain losses into L1 loss effectively enhances the quality of video reconstruction.

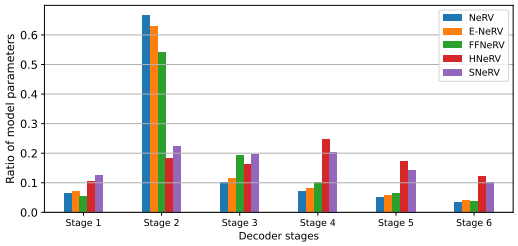


Figure 6. Distribution of model parameters across various decoder blocks in our HNeRV-Boost framework. See Table 9 for PSNR results under these five configurations.

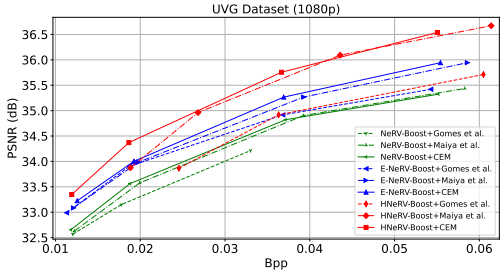


Figure 7. Rate-distortion comparisons between different entropy minimization techniques on the UVG dataset in PSNR.

Entropy Minimization. To highlight the contribution of our proposed CEM scheme, Fig. 7 displays the RD curves comparing various entropy minimization techniques on our boosted INR models. Notably, for content-dependent embedding compression in the boosted HNeRV, all methods employ asymmetric quantization. Since Gomes *et al.* [14] use the asymmetric quantization for model weights, it tends to shift the quantized weights away from their original distribution, causing inferior RD results. Compared with Maiya *et al.* [28], our CEM method achieves more bitrate savings by maintaining consistency in the entropy model between training and inference. These results verify the superiority of the CEM technique in INR compression.

5. Conclusion

In this paper, we develop a universal framework to boost implicit video representations, achieving substantial improvements in key tasks like regression, compression, inpainting, and interpolation. These advancements are primarily due to the integration of several novel developments, including the temporal-aware affine transform, sinusoidal NeRV-like block design, improved reconstruction loss, and consistent entropy minimization. Through comprehensive evaluations against multiple implicit video models, our boosted models demonstrate superior performance, setting a new benchmark in the field of implicit video representation. The contribution of each component is validated through extensive ablation studies.

Acknowledgments. This work was supported by the General Research Fund (Project No. 16209622) from the Hong Kong Research Grants Council.

References

[1] Yunpeng Bai, Chao Dong, Cairong Wang, and Chun Yuan. Ps-nerv: Patch-wise stylized neural representations for videos. In *2023 IEEE International Conference on Image Processing (ICIP)*, pages 41–45. IEEE, 2023. 2, 3, 4, 6

[2] Johannes Ballé, Valero Laparra, and Eero P Simoncelli. End-to-end optimized image compression. In *5th International Conference on Learning Representations, ICLR 2017*, 2017. 3

[3] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. In *International Conference on Learning Representations*, 2018. 3

[4] Robert Bamler. Understanding entropy coding with asymmetric numeral systems (ans): a statistician's perspective. *arXiv preprint arXiv:2201.01741*, 2022. 12

[5] Jean Bégaint, Fabien Racapé, Simon Feltman, and Akshay Pushparaja. Compressai: a pytorch library and evaluation platform for end-to-end compression research. *arXiv preprint arXiv:2011.03029*, 2020. 12

[6] Yash Bhalgat, Jinwon Lee, Markus Nagel, Tijmen Blankevoort, and Nojun Kwak. Lsq+: Improving low-bit quantization through learnable offsets and better initialization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 696–697, 2020. 5

[7] 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. 2, 3, 4, 6, 7, 8, 14

[8] Hao Chen, Matt Gwilliam, Bo He, Ser-Nam Lim, and Abhinav Shrivastava. Cnerv: Content-adaptive neural representation for visual data. In *British Machine Vision Conference*, 2022. 2, 3, 4

[9] 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*, pages 10270–10279, 2023. 2, 3, 6, 7, 8, 14

[10] Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto. Learned image compression with discretized gaussian mixture likelihoods and attention modules. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7939–7948, 2020. 12

[11] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. In *International Conference on Learning Representations*, 2016. 3

[12] Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. In *International Conference on Learning Representations*, 2019. 5

[13] Golnaz Ghiasi, Honglak Lee, Manjunath Kudlur, Vincent Dumoulin, and Jonathon Shlens. Exploring the structure of a real-time, arbitrary neural artistic stylization network. In *British Machine Vision Conference*, 2017. 3

[14] 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*, pages 18497–18506, 2023. 2, 3, 4, 5, 8

[15] Bo He, Xitong Yang, Hanyu Wang, Zuxuan Wu, Hao Chen, Shuaiyi Huang, Yixuan Ren, Ser-Nam Lim, and Abhinav Shrivastava. Towards scalable neural representation for diverse videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6132–6142, 2023. 8, 12, 14

[16] Yanting Hu, Jie Li, Yuanfei Huang, and Xinbo Gao. Channel-wise and spatial feature modulation network for single image super-resolution. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(11):3911–3927, 2019. 3

[17] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE international conference on computer vision*, pages 1501–1510, 2017. 2, 3, 4, 8

[18] Yoonsik Kim, Jae Woong Soh, Gu Yong Park, and Nam Ik Cho. Transfer learning from synthetic to real-noise denoising with adaptive instance normalization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3482–3492, 2020. 3

[19] Janus B. Kristensen. Big buck bunny. 2010. 5

[20] Ho Man Kwan, Ge Gao, Fan Zhang, Andrew Gower, and David Bull. Hinerv: Video compression with hierarchical encoding based neural representation. *arXiv preprint arXiv:2306.09818*, 2023. 3

[21] Joo Chan Lee, Daniel Rho, Jong Hwan Ko, and Eunbyung Park. Ffnerv: Flow-guided frame-wise neural representations for videos. In *Proceedings of the ACM International Conference on Multimedia*, 2023. 2, 3, 4, 8

[22] Jiahao Li, Bin Li, and Yan Lu. Deep contextual video compression. *Advances in Neural Information Processing Systems*, 34:18114–18125, 2021. 2, 6, 7, 12

[23] 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 *European Conference on Computer Vision*, pages 267–284. Springer, 2022. 2, 3, 4, 6, 7, 8, 14

[24] Zhemin Li, Hongxia Wang, and Deyu Meng. Regularize implicit neural representation by itself. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10280–10288, 2023. 1

[25] Jianping Lin, Dong Liu, Houqiang Li, and Feng Wu. M-lvc: Multiple frames prediction for learned video compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3546–3554, 2020. 2

[26] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11976–11986, 2022. 3

[27] Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Chunlei Cai, and Zhiyong Gao. Dvc: An end-to-end deep video compression framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11006–11015, 2019. 2

[28] Shishira R Maiya, Sharath Girish, Max Ehrlich, Hanyu Wang, Kwot Sin Lee, Patrick Poirson, Pengxiang Wu, Chen Wang, and Abhinav Shrivastava. Nirvana: Neural implicit representations of videos with adaptive networks and autoregressive patch-wise modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14378–14387, 2023. 2, 3, 5, 8

[29] Alexandre Mercat, Marko Viitanen, and Jarno Vanne. Uvg dataset: 50/120fps 4k sequences for video codec analysis and development. In *Proceedings of the 11th ACM Multimedia Systems Conference*, pages 297–302, 2020. 5

[30] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 1, 3

[31] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022. 1

[32] Deniz Oktay, Johannes Ballé, Saurabh Singh, and Abhinav Shrivastava. Scalable model compression by entropy penalized reparameterization. In *International Conference on Learning Representations*, 2019. 3

[33] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5865–5874, 2021. 1

[34] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2337–2346, 2019. 3

[35] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 724–732, 2016. 5

[36] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, 2018. 3

[37] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10318–10327, 2021. 1

[38] Vishwanath Saragadam, Daniel LeJeune, Jasper Tan, Guha Balakrishnan, Ashok Veeraraghavan, and Richard G Baraniuk. Wire: Wavelet implicit neural representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18507–18516, 2023. 1

[39] Xihua Sheng, Jiahao Li, Bin Li, Li Li, Dong Liu, and Yan Lu. Temporal context mining for learned video compression. *IEEE Transactions on Multimedia*, 2022. 2, 6, 7, 12

[40] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems*, 33:7462–7473, 2020. 1

[41] Ivan Skorokhodov, Savva Ignatyev, and Mohamed Elhoseiny. Adversarial generation of continuous images. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10753–10764, 2021. 1

[42] Ivan Skorokhodov, Sergey Tulyakov, and Mohamed Elhoseiny. Stylegan-v: A continuous video generator with the price, image quality and perks of stylegan2. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3626–3636, 2022. 1

[43] Myungseo Song, Jinyoung Choi, and Bohyung Han. Variable-rate deep image compression through spatially-adaptive feature transform. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2380–2389, 2021. 3

[44] Yannick Strümpler, Janis Postels, Ren Yang, Luc Van Gool, and Federico Tombari. Implicit neural representations for image compression. In *European Conference on Computer Vision*, pages 74–91. Springer, 2022. 1

[45] Gary J Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on circuits and systems for video technology*, 22(12):1649–1668, 2012. 1, 7

[46] Filip Szatkowski, Karol J Piczak, Przemysław Spurek, Jacek Tabor, and Tomasz Trzciński. Hypersound: Generating implicit neural representations of audio signals with hypernetworks. *arXiv preprint arXiv:2211.01839*, 2022. 1

[47] Lv Tang, Xinfeng Zhang, Gai Zhang, and Xiaoqi Ma. Scene matters: Model-based deep video compression. In *Proceedings of the IEEE international conference on computer vision*, 2023. 2

[48] Xintao Wang, Ke Yu, Chao Dong, and Chen Change Loy. Recovering realistic texture in image super-resolution by deep spatial feature transform. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 606–615, 2018. 3, 4

[49] Xintao Wang, Yu Li, Honglun Zhang, and Ying Shan. Towards real-world blind face restoration with generative facial prior. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9168–9178, 2021. 3

[50] Thomas Wiegand, Gary J Sullivan, Gisle Bjontegaard, and Ajay Luthra. Overview of the h. 264/avc video coding standard. *IEEE Transactions on circuits and systems for video technology*, 13(7):560–576, 2003. 1, 7

[51] Xingyu Xie, Pan Zhou, Huan Li, Zhouchen Lin, and YAN Shuicheng. Adan: Adaptive nesterov momentum algorithm for faster optimizing deep models. In *Has it Trained Yet? NeurIPS 2022 Workshop*, 2022. 6

[52] Wentian Xu and Jianbo Jiao. Revisiting implicit neural representations in low-level vision. 2023. 1

[53] Runzhao Yang, Tingxiong Xiao, Yuxiao Cheng, Qianni Cao, Jinyuan Qu, Jinli Suo, and Qionghai Dai. Sci: A spectrum concentrated implicit neural compression for biomedi-

cal data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 4774–4782, 2023. 1

[54] Yunfan Zhang, Ties van Rozendaal, Johann Brehmer, Markus Nagel, and Taco Cohen. Implicit neural video compression. In *ICLR Workshop on Deep Generative Models for Highly Structured Data*, 2022. 1

[55] 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*, pages 2031–2040, 2023. 2, 7, 12, 14