HNeRV: A Hybrid Neural Representation for Videos

Anonymous authors
Paper under double-blind review

Abstract

Implicit neural representations store videos as neural networks and have performed well for various vision tasks such as video compression and denoising. With frame index or positional index as input, implicit representations (NeRV, E-NeRV, *etc*.) reconstruct video frames from *fixed* and *content-agnostic* embeddings. Such embedding largely limits the regression capacity and internal generalization for video interpolation. In this paper, we propose a Hybrid Neural Representation for Videos (**HNeRV**), where *learnabl* and *content-adaptive* embeddings act as the decoder input. Besides the input embedding, we introduce a HNeRV block to make model parameters evenly distributed across the entire network, therefore higher layers (layers near the output) can have more capacity to store high-resolution content and video details. With content-adaptive embedding and re-designed architecture, HNeRV outperforms implicit methods (NeRV, E-NeRV) in video regression task for both reconstruction quality and convergence speed, and shows better internal generalization. As a simple and efficient video representation, HNeRV also shows decoding advantages for speed, flexibility, and deployment, compared to traditional codecs (H.264, H.265) and learning-based compression methods. Finally, we explore the effectiveness of HNeRV on downstream tasks such as video compression and video inpainting.

1 Introduction

Given the massive amount of videos generated every day, representing them efficiently is a key task in computer vision and video processing. Even for modern storage systems, the space requirements of raw video data can be overwhelming. Despite storage becoming cheaper, network speeds and I/O processing remain a bottleneck and make transferring and processing videos expensive.

Traditional video codecs, such as H.264 (Wiegand et al., 2003a) and HEVC (Sullivan et al., 2012), manually design an encoder and decoder based on discrete cosine transform (Ahmed et al., 1974). With the success of deep learning, many attempts (Lu et al., 2019; Rippel et al., 2021; Agustsson et al., 2020; Djelouah et al., 2019; Habibian et al., 2019; Liu et al., 2019; 2021; Rippel et al., 2019; Wu et al., 2018) have been made to replace certain components of existing compression pipelines with neural networks. Although these learning-based compression methods show high potential in terms of rate-distortion performance, they suffer from complex pipelines and expensive computation, not just to train, but also to encode and decode.

To address the complex pipelines and heavy computation, implicit neural representations (Rahaman et al., 2019; Sitzmann et al., 2020; Schwarz et al., 2021; Chen & Zhang, 2019; Park et al., 2019) have become popular due to their simplicity, compactness, and efficiency. These methods show great potential for visual data compression, such as COIN (Dupont et al., 2021) for image compression, NeRV (Chen et al., 2021) for video compression. By representing video as neural networks, video compression problems can be converted to model compression problems, and greatly simplify the encoding and decoding pipeline.

Given frame index t as input, as shown in Figure 1(a), NeRV (Chen et al., 2021) uses a fixed position encoding function and a learnable decoder to reconstruct video frames from content-agnostic embedding. E-NeRV (Li et al., 2022) takes a temporal embedding and spatial embedding, both content-agnostic, to reconstruct video frames. In this paper, we propose a hybrid neural representation for videos (HNeRV, Figure 1(b), which takes a learnable, content-adaptive frame embedding

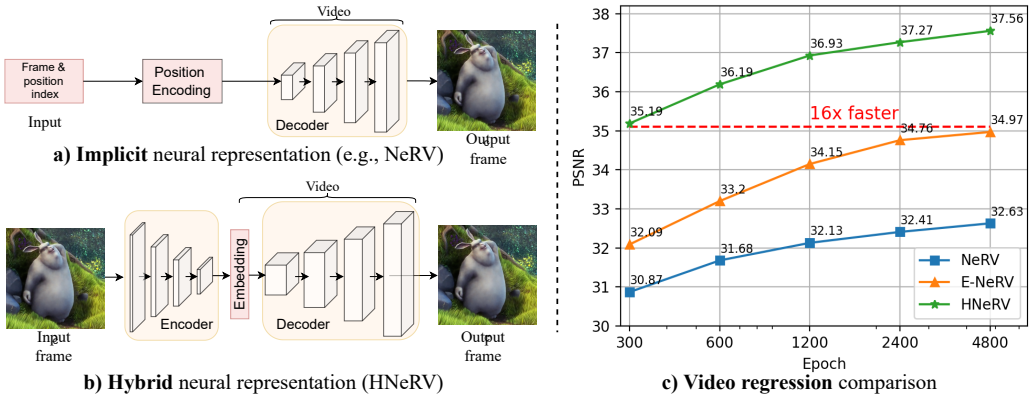

Figure 1: **a) Implicit** neural representation with fixed frame embedding. **b) Hybrid** neural representation with learnable and content-adaptive embedding (ours). **c) Video regression** for hybrid and implicit neural representations.

as input. HNeRV can be viewed as an auto-encoder with a tiny frame embedding. With a tiny embedding and a powerful decoder, HNeRV can reconstruct the video with high fidelity and therefore represent it well. Besides the fixed embedding, NeRV suffers from an imbalance in the distribution of model parameters, where later layers (closer to the output image) have much fewer parameters than earlier layers (closer to the embedding), meaning it cannot effectively reconstruct massive video content at high resolutions. Therefore, we introduce an HNeRV block to make parameters evenly distributed over the entire network, by increasing kernel sizes and channel widths at later stages.

Our proposed neural representation is a hybrid approach since it stores video in two parts: tiny frame embeddings and a learned neural decoder. Together with the re-designed decoder blocks, HNeRV improves the reconstruction quality for video regression and boosts the convergence speed up to $16\times$ compared to implicit methods, shown in Figure 1(c). With content-adaptive embeddings, HNeRV also shows much better internal generalization (ability to encode and decode frames from the video that were not seen during training), and we verify this by frame interpolation results.

As a simple representation, HNeRV shows great decoding advantages in terms of speed, flexibility, and ease of deployment. Traditional codecs, like H.264, have most of the operations hand-crafted and need a special design for their implementation and deployment. Therefore, although H.264 have good hardware and software support (*e.g.*, FFMPEG), newer codecs like H.266 and AV1 still suffer from deployment difficulty due to their complexity and increasing computation. In contrast, learning-based compression methods use deep neural networks for most operations, which simplifies deployment; however, large latency greatly limits their application inspite of their bits-distortion improvements. HNeRV, as a simple and efficient neural representation, only needs a network forward operation for video decoding, which enables both fast decoding speed and easy deployment. Besides, most video compression methods are auto-regressive and there is a high dependency on the sequential order of video frames. To decode one frame in a picture group, they need to decode key frames and all previous frames first. In contrast, there is no dependency on the sequential order of frames for HNeRV, which means it can randomly access frames efficiently to decode frames in parallel. The decoding time decreases linearly when we reduce decoded frames, while other representations still needs to decode most frames. Such simplicity and parallelism make HNeRV a good codec for further speedups, like a special NPU chip, or parallel decoding with huge batches.

We also evaluate HNeRV on downstream tasks video compression and inpainting. By converting video compression to model compression, HNeRV shows better bit-distortion performance over implicit methods and achieves comparable performance with state-of-the-art methods. We posit that neural representation can be robust to distortion in pixel space and therefore restore well for video distortion, and verify this observation on the video inpainting task.

In summary, we propose a hybrid neural representation for videos. With content-adaptive embedding and re-designed architecture, HNeRV shows much better video regression performance over implicit methods, in reconstruction quality, convergence speed, and internal generalization. Besides video regression, HNeRV has a simple, fast, and flexible advantage during video decoding, as well

as easy deployment. Finally, HNeRV shows good performance over downstream tasks like video compression and video inpainting.

2 Related Work

Neural Representation. Implicit representations fit to each individual test signal (Mehta et al., 2021) where the model is regressed to a given image, scene, or video. Most implicit neural representations are coordinate-based. These coordinate-based implicit representations are used in image reconstruction (Tancik et al., 2020; Sitzmann et al., 2020), shape regression (Chen & Zhang, 2019; Park et al., 2019), and 3D view synthesis (Mildenhall et al., 2020; Schwarz et al., 2021). NeRV (Chen et al., 2021) instead proposes an image-wise implicit representation, which takes frame indices as inputs and leverages neural representation for fast and accurate video compression. Relying only on index, and not coordinates, speeds up the encoding and decoding process compared to coordinate-based (pixel-wise) methods. Based on NeRV, E-NeRV (Li et al., 2022) further boosts the video regression performance via decoupling frame index and spatial index. Traditional autoencoders would not be considered implicit representations since most information is stored in their large image-specific embeddings. Nevertheless, they are a form of neural representation, and HNeRV borrows the general concept for its encoder from standard U-shaped autoencoders (Ahmed et al., 1974; Vincent et al., 2008; Kingma & Welling, 2014; Pu et al., 2016). The major difference is HNeRV keeps the embedding intentionally tiny and compact, so as to keep most of the representation implicit (stored in the decoder).

Video Compression. Traditional video compression methods such as MPEG (Le Gall, 1991), H.264 (Wiegand et al., 2003b), and H.265 (Sullivan et al., 2012) achieve good reconstruction results with decent decompression speeds. Recently, deep learning techniques have been proposed for video compression. While these approaches focus on replacing the entire compression pipeline, they each borrow principles from the traditional handcrafted approaches. Some have framed the problem primarily as image compression and interpolation (Wu et al., 2018; Djelouah et al., 2019), or attempt to solve this task with image compression via autoencoders (Habibian et al., 2019), or focus purely on interpolation for the sake of compression (Liu et al., 2020). Others essentially reformulate traditional video compression pipelines using deep learning tools (Rippel et al., 2019; Liu et al., 2019; Agustsson et al., 2020), at varying levels of complexity. Recent approaches have focused on tackling the computational inefficiencies of existing art, including by fine-tuning traditional codecs (Khani et al., 2021), and by optimizing pieces of the compression pipeline (Rippel et al., 2021). The approach which inspired much of this work, NeRV, responds to these same inefficiencies by proposing a specialized architecture for video memorization (Chen et al., 2021). Once video is represented as a neural network, the video compression problem can be converted to a model compression problem and achieve good bit-distortion performance. With learnable embeddings and re-designed decoder blocks, HNeRV improves the video regression capacity and convergence speed, while video compression is still viable by model compression.

Model Compression. NeRV formulated video compression as model compression (Chen et al., 2021), which is a diverse area. In this paper we apply only a small subset of possible methods. We use weight pruning (Han et al., 2015b) and weight quantization (Gupta et al., 2015; Jacob et al., 2017; Krishnamoorthi, 2018). We also use entropy encoding for lossless compression after pruning and quantization (Huffman, 1952; Han et al., 2015a). Note that many other model compression methods can be leveraged to further reduce the size and video neural representation can always benefit from developments in the model compression area.

Video Inpainting & Internal Learning. Video inpainting is typically framed as some combination of object removal and attempting to recreate missing regions of images. Whereas some methods rely on priors from training on large datasets (Wang et al., 2021), ours has more in common with a recent zero-shot fully-internal approach (Ouyang et al., 2021). We define "Internal learning" in terms of exploiting recurrence of information within a single domain, like within an image (Shocher et al., 2018)) or within an video (Zhang et al., 2019). It can be thought of as a sort of DIP-for-video, a line of work that was started for images with DIP (Ulyanov et al., 2018) and extended for video by double-DIP (Gandelsman et al., 2019). Other methods have embraced this paradigm partially, learning some priors from large external datasets, before learning video-specific priors via internal learning (Wang et al., 2021).

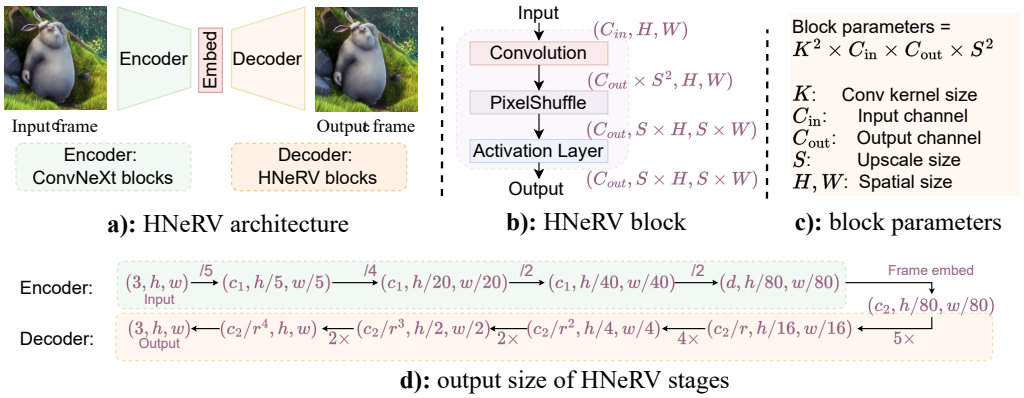

Figure 2: **a**): HNeRV uses ConvNeXt blocks to encode frames as tiny embeddings, which are decoded by HNeRV blocks. **b**): HNeRV block of three layers, with input/output size illustrated. **c**): Parameter number for an HNeRV block. **d**): Output size of each stage with strides 5,4,2,2.

3 METHOD

3.1 HNERV OVERVIEW

HNeRV can be viewed as an auto-encoder with tiny embeddings (Figure 2(a)). We choose ConvNeXt blocks (Liu et al., 2022) to build the encoder (1 block each stage), and propose novel HNeRV blocks (Figure 2(b)) to build the decoder.

Hybrid Neural Representation. Compact video representations can be divided into two parts: explicit methods and implicit methods. Explicit methods use an auto-encoder to encode and decode all videos, and store content *explicitly* as a latent embedding. Given a *video-specific embedding* as input, the decoder can reconstruct the video. Implicit methods use only a learnable decoder to represent the video. Given fixed frame index as input, the *video-specific decoder* can reconstruct the video. With content-adaptive embedding as input, explicit representation shows better generalization and compression performance, while implicit representations have a much simpler encoding/decoding pipeline and a high potential for compression (benefits from model compression techniques) and other downstream tasks (*e.g.*, efficient video dataloader, video denoising, inpaining). In this paper, we propose a hybrid neural representation to combine the advantages of both explicit and implicit methods. Similar to implicit representation, we use a learnable decoder to model video separately and store most content implicitly in the video-specific decoder. To achieve better reconstruction, we use learnable embedding as input and store information explicitly in these frame-specific embeddings, which is similar to explicit methods. Therefore, we can use any powerful encoder to generate tiny content-adaptive embedding to boost the performance of implicit representation. Since such embedding is quite small (*e.g.*, a 128-d vector for a 640×1280 frame), our hybrid neural representation is as compact as implicit methods, but with better reconstruction capacity, convergence speed, and internal generalization, and keeps the full potential for downstream tasks.

Model Architecture. Similar to a NeRV block, it consists of three layers: convolution layer, pixelshuffle layer, and activation layer. Within HNeRV block, only the convolution layer has learnable parameters (Figure 2(b)). Illustrated in Table 1, a NeRV block use fix kernel sizes for all stages $K = 3$, and reduces channel width by 2, $\text{C}_{\text{out}} = C_{\text{in}}/2$. Therefore, for blocks at later stages, the parameters are quite few and may not be strong enough to store video content at high resolution. In contrast, we increase the kernel size and channel width for later HNeRV blocks, where K increase from 1 (stage 1), 3 (stage 2), to K_{max} (5, *etc.* for later stages), and decrease channel width by a reduction factor r (1.2, *etc.*). With kernel size 1, the first block has much fewer parameters; with larger kernel size and wider channels, HNeRV blocks at later stages are much stronger; and we therefore get a more even distribution of model parameters across layers. We list output size of various stages in Figure 2(d), with embedding dimension d, channel reduction r. Each stage has one block, and we use a 1×1 convolution layer to get low-dimension frame embeddings (channel width from c_1 to d), and a 3×3 convolution layer for final image predictions (channel width from c_2/r^4 to 3).

Table 1: **HNeRV block *vs*. NeRV block**. K is kernel size for each stage, C_{out} and C_{in} are output/input channel widths for each block. We decrease parameters via a small $K = 1$ for first block, and increase parameters for later layers with a larger K and wider channels

NeRV blocks	$K = 3, \quad C_{\text{out}} = C_{\text{in}}/2$
HNeRV blocks	$K = 1, 3, ..., K_{\text{max}}, \quad C_{\text{out}} = C_{\text{in}}/r$

Loss Functions. Since HNeRV attempts to reconstruct video with high fidelity, we use the loss objective $L = \text{Loss}(x, p)$, where x is the input frame, p is the HNeRV prediction, and Loss is any reconstruction loss function like L2, L1, or SSIM loss.

Total size. As a hybrid neural representation, we include both frame embedding and decoder parameters to compute the total size of our video representation: TotalSize = EmbedSize + DecoderSize.

3.2 Downstream Tasks

Video Compression. We leverage both model compression and embedding quantization for video compression. Similar to NeRV, we apply global unstrutured pruning, model quantization, and weight entropy encoding for model compression (details can be found in the appendix).

For quantization of an vector μ, we linearly map every element to the closest integer,

$$\mu_i = \text{Round}\left(\frac{\mu_i - \mu_{\text{min}}}{\text{scale}}\right) * \text{scale} + \mu_{\text{min}}, \quad \text{where scale} = \frac{\mu_{\text{max}} - \mu_{\text{min}}}{2^b - 1} \tag{1}$$

μ_i is vector element, 'Round' is a function that rounds to the closest integer, b is the quantization bit length, μ_{max} and μ_{min} are the max and min value of vector μ, and 'scale' is the scaling factor.

Video Inpainting. For partially distorted video, we only compute loss for non-masked pixels,

$$L_{\text{inpainting}} = (1 - M) * \text{Loss}(x, p) \tag{2}$$

where M is the mask matrix where distorted pixels are 1 and other are 0. For inpainting output, following IIVI (Ouyang et al., 2021), we fill the masked region with HNeRV's output.

4 Experiment

4.1 Dataset and Implementation Details

We use the Big Buck Bunny (Bunny) (big), UVG (Mercat et al., 2020) and DAVIS (Wang et al., 2016) datasets. Bunny video has 132 frames with resolution 720×1280, and we center-crop 640×1280 to get tiny spatial size (*e.g.*, 1×2) for embedding. UVG dataset has 7 videos[1] with size 1080×1920 at FPS 120 of 5s or 2.5s, and we center-crop 960×1920. We also take 10 videos[2] from DAVIS validation (1080×1920, 50-200 frames) and center crop the 960×1920. Unless otherwise specified, we use Adam optimizer, beta as (0.9, 0.999), weight decay as 0, learning rate at 0.001 with cosine learning rate decay, batch size as 2, L2 loss as reconstruction loss function. K_{max} is set as 5, reduction r is set as 1.2 in Table 1. We set stride list as (5,4,4,2,2), (5,4,3,2,2), and (5,4,4,3,2) for video resolutions of 640×1280, 480×960, and 960×1920 respectively.

For evaluation metrics, we use PSNR and MS-SSIM to evaluate reconstruction quality, bits per pixel (bpp) for compression, and pixels per pixel (ppp) for model compactness. We conduct all experiments in Pytorch with GPU RTX2080ti, where it takes around 8s each epoch to train a 130 frame video of 640×1280. We choose HNeRV size to ensure the PSNR lies between 30-40 for fair video reconstruction. We provide more experiment details such as architecture details, qualitative results, quantitative results of some plots, and per-video compression results, in the appendix.

4.2 Main Results

Video regression. We first compare HNeRV with implicit methods NeRV and E-NeRV on Bunny. For fair comparison, we scale channel width to make total size comparable as the original paper did.

[1] Beauty, Bosphorus, HoneyBee, Jockey, ReadySetGo, ShakeNDry, YachtRide

[2] bike-packing, blackswan, bmx-trees, breakdance, camel, car-round, car-shadow, cows, dance-twirl, dog

Table 2: Video regression with different **sizes**

Size	0.35M	0.75M	1.5M	3M	avg.
NeRV	26.99	28.46	30.87	33.21	29.88
E-NeRV	27.84	30.95	32.09	36.72	31.90
HNeRV	**30.15**	**32.81**	**35.19**	**37.43**	**33.90**

Table 3: Video regression with different **epochs**

Epoch	300	600	1200	1800	2400	3600
NeRV	28.46	29.15	29.57	29.73	29.77	29.86
E-NeRV	30.95	32.07	32.79	33.1	33.36	33.67
HNeRV	**32.81**	**33.89**	**34.51**	**34.73**	**34.88**	**35.03**

Table 4: Video regression at resolution **960**×**1920**, PSNR↑ reported

Video	beauty	swan	bmx	bosph	dance	camel	bee	jockey	ready	shake	yach	avg.
NeRV	33.25	28.48	27.86	33.22	26.45	24.81	37.26	31.74	24.84	33.08	28.30	29.94
E-NeRV	33.17	29.38	28.68	33.69	27.88	25.16	37.62	31.63	25.24	34.39	28.42	30.48
HNeRV	**33.58**	**30.35**	**29.98**	**34.73**	**30.45**	**26.71**	**38.96**	**32.04**	**25.74**	**34.57**	**29.26**	**31.49**

Table 5: Video regression at resolution **480**×**960**, PSNR↑ reported

Video	beauty	swan	bmx	bosph	dance	camel	bee	jockey	ready	shake	yach	avg.
NeRV	36.27	29.75	28.81	35.07	29.47	26.75	40.76	32.58	25.81	35.33	30.11	31.88
E-NeRV	36.26	30.27	29.20	36.06	30.83	27.39	**43.26**	32.70	26.19	35.64	30.38	32.56
HNeRV	**36.91**	**31.92**	**31.27**	**36.95**	**33.85**	**28.85**	42.05	**33.33**	**27.07**	**36.97**	**30.96**	**33.65**

a) Ground truth b) NeRV (ppp=0.003)

c) E-NeRV (ppp=0.003) d) HNeRV (ppp=0.003)

Figure 3: **Visualization of video neural representations** at 0.003 ppp, which means the total size is only about 0.3% of the original video size. Red squares are added for the reader's convenience to emphasize areas of greatest discernible difference.

In Table 2, with the same size and 300 epochs, HNeRV outperforms both NeRV and E-NeRV. We also show comparison of different training time in Table 3 with 0.75M size and in Figure 1(Right) with 1.5M size, where HNeRV converges much faster compared to implicit methods. We show such improvements qualitatively as well in Figure 3. As a compact representation, HNeRV reconstructs the video well with only 0.35M parameters, at 0.003 ppp. We also evaluate it on 7 UVG videos and 4 DAVIS videos, where HNeRV shows large improvements at resolution 960×1920 in Table 4, and its resized 480×960 version in Table 5, with size 3M and 300 epochs.

Video decoding. We evaluate video decoding on Bunny with channel reduction r as 1.5, where H.264 and H.265 are tested with 4 CPUs[3], while DCVC (Li et al., 2021) and HNeRV are tested

[3]Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz

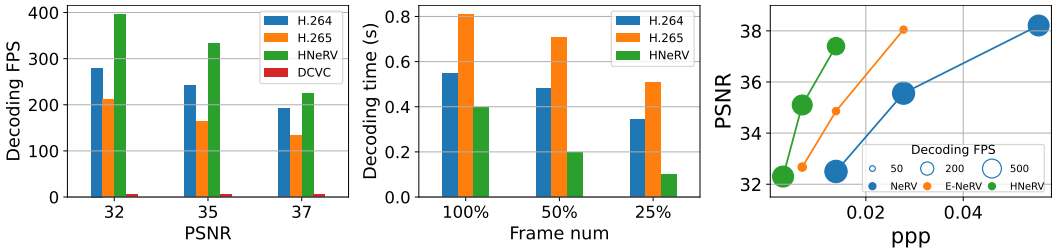

Figure 4: **Video decoding. Left**: HNeRV outperforms traditional video codecs H.264 and H.265, and learning-based compression method DCVC. **Middle**: HNeRV shows much better flexibility when decoding part of video frames where the decoding time decrease linearly for HNeRV while other methods still need to decode most frames. **Right**: As neural representation, HNeRV performs well for compactness (ppp), reconstruction quality (PSNR), and decoding speed (FPS).

Left: NeRV Middle: E-NeRV Right: HNeRV

Figure 5: Visualization of **Embedding interpolation**.

Table 6: **Internal generalization** results. NeRV, E-NeRV, and HNeRV use interpolated embedding as input, HNeRV† uses held-out frames as input. With content-adaptive embedding as input, HNeRV shows much better reconstruction on held-out frames

Method	beauty	swan	bmx	bosph	dance	camel	bee	jockey	ready	shake	yach	avg.
NeRV	28.05	17.94	15.55	30.04	16.99	14.83	36.99	20.00	17.02	29.15	24.50	22.82
E-NeRV	27.35	19.4	15.12	28.95	17.16	17.97	38.24	19.39	16.74	30.23	22.45	23.00
HNeRV	30.97	21.44	17.35	34.38	20.2	19.93	38.83	23.67	20.90	**32.69**	**27.30**	26.15
HNeRV†	**31.10**	**21.97**	**18.29**	**34.38**	**20.29**	**20.64**	**38.83**	**23.82**	**20.99**	32.61	27.24	**26.38**

with 1 GPU (RTX2080ti). We only measure the forward time for DCVC and HNeRV. We compare video decoding at various reconstruction qualities (PSNR at 32, 35, and 37) in Figure 4(Left), where HNeRV outperforms traditional codecs (H.264 and H.265) and learning-based DCVC. Note that although many prior learning-based compression methods show bit-distortion improvements, their decoding speeds *lag far behind* traditional codecs and neural representation. Besides, most compression methods encode and decode frames in an auto-regressive way and can not access frames randomly. Compared to these methods, the decoding of HNeRV is much simpler and can be deployed to any platforms easily. We compare decoding time in Figure 4(Middle) (PSNR at 35) where $100\%, 50\%$, and 25% frames (evenly sampled) are decoded. Since there is no dependency among video frames, HNeRV can decode them in parallel and decoding time decrease linearly with frame number. In contrast, H.264 and H.265 still need to decode most frames, even though only some of them are needed. Finally, we compare with implicit methods in Figure 4(Right), where HNeRV is slightly slower than NeRV since the computation of later layers is more expensive due to large K and channel width. As a hybrid neural representation, HNeRV achieves much better trade-offs in regards of compactness (ppp metric), reconstruction quality (PNSR), and decoding speed (FPS).

Internal generalization. Since HNeRV leverages content-adaptive embeddings, we also evaluate it for the video interpolation task. Holding out every other frame as a test set, NeRV, E-NeRV, and

Table 7: Video **inpainting** results. With 5 fix box masks on input videos, we evaluate the output with PSNR ↑. 'Input' is the baseline of mask video and ground truth

Video	bike	b-swan	bmx	b-dance	camel	c-round	c-shadow	cows	dance-twirl	dog	avg.
Input	23.14	20.24	19.99	21.36	17.3	20.47	18.92	19.37	20.45	18.39	19.96
NeRV	30.94	33.43	32.07	27.82	31.99	29.09	31.63	30.08	30.45	33.85	31.14
IIVI	**31.87**	**36.02**	**34.36**	27.63	**35.11**	**32.61**	**33.69**	**31.26**	**31.44**	**35.7**	**32.97**
HNeRV	31.27	34.24	33.95	**27.94**	32.21	30.88	33.07	30.82	31.21	34.7	32.03

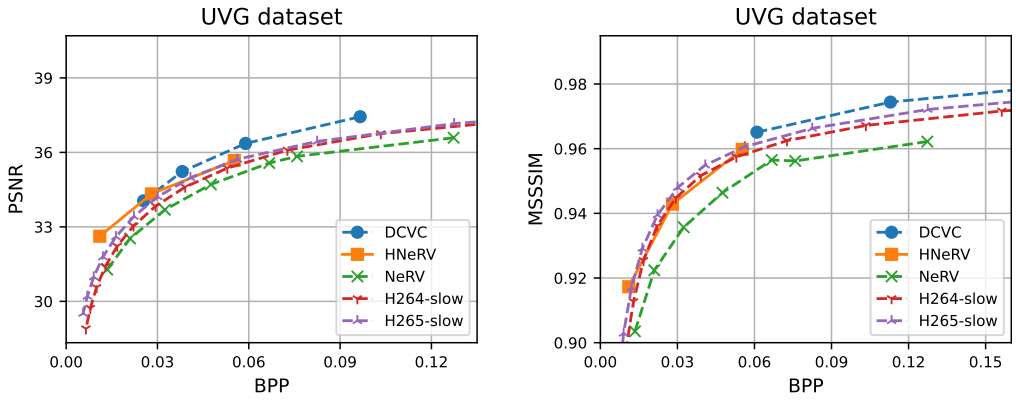

Figure 6: **Compression** results on UVG dataset.

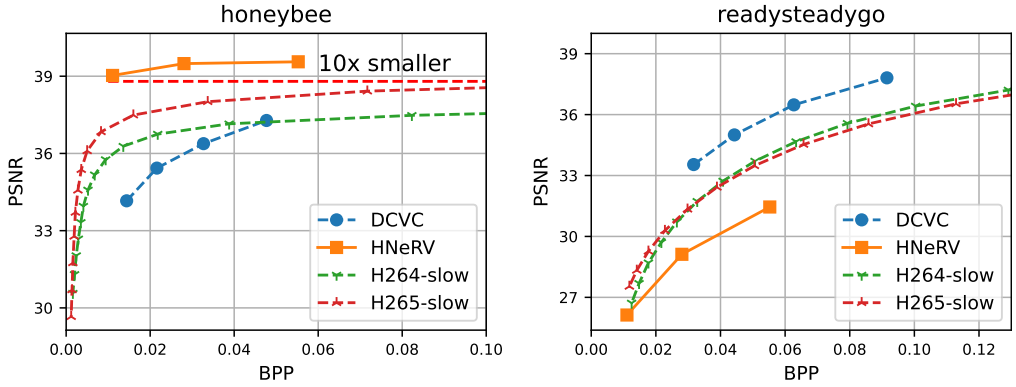

Figure 7: Compression results of **best/worst cases** from UVG dataset. HNeRV achieves good performance especially for videos caputured by still cameras, like 'honeybee' video.

HNeRV use interpolated embedding as input, while HNeRV† uses the test frame as input. With learnable and content-adaptive embedding, our HNeRV shows much better generalization, quantitatively in Table 6. and qualitatively in Figure 5.

4.3 Downstream Tasks

Video compression. With model pruning (10% pruned), embedding quantization (8 bits), model quantization (8 bits), and model entropy encoding (8% saved), we show video compression results on UVG in Figure 6. HNeRV outperforms the implicit method, NeRV, and traditional video codecs H.264 and H.265. We also show the best and worst cases of compression for 'honeybee' and 'readysteadygo' videos respectively in Figure 7, where HNeRV achieves outstanding performance when the camera is not moving, like the 'honeybee' video (10× smaller). Given the limited performance on videos of highly dynamic scenes, we propose finding a good size and network architecture for such videos as future work.

Video inpainting. We also explore video inpainting with fixed and object masks. For fixed masks, we use 5 boxes of width 50 (Figure 8 top) and show quantitative results in Table 7 where HNeRV

Left: Input Right: HNeRV output

Figure 8: **Inpainting** results of fix masks and object masks.

improves inpainting performance over implicit methods NeRV. Although we do not have any specific design for inpainting task, HNeRV achieves comparable performance with an SOTA inpainting method IIVI (Ouyang et al., 2021). We show qualitative results in Figure 8 and the appendix.

4.4 Ablation Study

We show the effectiveness of even-distributed parameters in Table 8 and Table 9 by increasing kernel size and channel width of later layers. For the NeRV block, it uses fixed $K = 3$, and channel reduction factor $r = 2$. We also show an embeddings ablation study, for spatial size ($h \times w$) in Table 10 and embedding dimensions (d) in Table 11.

Table 8: **Kernel size** (K_{min}, K_{max}) ablation

K	PSNR	MS-SSIM
1,3	35.02	0.9752
1,5	**35.19**	**0.9773**
1,7	35.07	0.9757
3,3	33.09	0.9587

Table 9: **Channel reduction** r ablation

r	PSNR	MS-SSIM
1	34.96	0.9745
1.2	**35.19**	**0.9773**
1.5	34.98	0.9762
2	34.32	0.9715

Table 10: **Embedding spatial size** ablation

$h \times w$	PSNR	MS-SSIM
1×2	34.79	0.9735
2×4	**35.19**	**0.9773**
4×8	35.12	0.9761

Table 11: **Embedding dimension** ablation

d	PSNR	MS-SSIM
8	35.13	0.9770
16	**35.19**	**0.9773**
32	35.08	0.9758

5 Conclusion

In this paper, we propose a hybrid neural representation for videos (HNeRV). With content-adaptive embedding and evenly-distributed parameters, HNeRV improves video regression performance compared to implicit methods in reconstruction quality, convergence speed, and internal generalization. As a video representation, HNeRV is also simple, fast, and flexible for video decoding, and shows good performance for video compression and inpainting.

There are many limitations of HNeRV as well. Firstly, as a neural representation, HNeRV stores each video as a neural network. Given a new video, HNeRV still needs time to train to fit the video. Secondly, although HNeRV can represent a video well, finding a best-fit embedding size, model size, and network architecture (K_{max}, r, *etc.*) remains an open problem. Finally, although increasing kernel sizes and channel widths at later layers largely improves the regression performance, it slightly slows down the network (Figure 4(Right) and Table 14).

Reproduction: we provide architecture details in the appendix for reproduction and will release code upon acceptance.

REFERENCES

Big buck bunny, sunflower version. http://bbb3d.renderfarming.net/download.html. Accessed: 2010-09-30.

Eirikur Agustsson, David Minnen, Nick Johnston, Johannes Balle, Sung Jin Hwang, and George Toderici. Scale-space flow for end-to-end optimized video compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

N. Ahmed, T. Natarajan, and K.R. Rao. Discrete cosine transform. *IEEE Transactions on Computers*, C-23(1):90–93, 1974. doi: 10.1109/T-C.1974.223784.

Hao Chen, Bo He, Hanyu Wang, Yixuan Ren, Ser-Nam Lim, and Abhinav Shrivastava. Nerv: Neural representations for videos, 2021.

Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Abdelaziz Djelouah, Joaquim Campos, Simone Schaub-Meyer, and Christopher Schroers. Neural inter-frame compression for video coding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

Emilien Dupont, Adam Goliński, Milad Alizadeh, Yee Whye Teh, and Arnaud Doucet. Coin: Compression with implicit neural representations. *arXiv preprint arXiv:2103.03123*, 2021.

Yosef Gandelsman, Assaf Shocher, and Michal Irani. "double-dip": Unsupervised image decomposition via coupled deep-image-priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. *CoRR*, abs/1502.02551, 2015. URL http://arxiv.org/abs/1502.02551.

Amirhossein Habibian, Ties van Rozendaal, Jakub M. Tomczak, and Taco S. Cohen. Video compression with rate-distortion autoencoders. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015a.

Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. *CoRR*, abs/1506.02626, 2015b. URL http://arxiv.org/abs/1506.02626.

David A Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew G. Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. *CoRR*, abs/1712.05877, 2017. URL http://arxiv.org/abs/1712.05877.

Mehrdad Khani, Vibhaalakshmi Sivaraman, and Mohammad Alizadeh. Efficient video compression via content-adaptive super-resolution. *arXiv preprint arXiv:2104.02322*, 2021.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2014.

Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018.

Didier Le Gall. Mpeg: A video compression standard for multimedia applications. *Commun. ACM*, 34(4):46–58, April 1991. ISSN 0001-0782. doi: 10.1145/103085.103090. URL https://doi.org/10.1145/103085.103090.

Jiahao Li, Bin Li, and Yan Lu. Deep contextual video compression. *Advances in Neural Information Processing Systems*, 34, 2021.

Zizhang Li, Mengmeng Wang, Huaijin Pi, Kechun Xu, Jianbiao Mei, and Yong Liu. E-nerv: Expedite neural video representation with disentangled spatial-temporal context. *ECCV*, 2022.

Haojie Liu, Tong Chen, Ming Lu, Qiu Shen, and Zhan Ma. Neural video compression using spatio-temporal priors, 2019.

Haojie Liu, Ming Lu, Zhan Ma, Fan Wang, Zhihuang Xie, Xun Cao, and Yao Wang. Neural video coding using multiscale motion compensation and spatiotemporal context model. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(8):3182–3196, 2021. doi: 10.1109/TCSVT.2020.3035680.

Jerry Liu, Shenlong Wang, Wei-Chiu Ma, Meet Shah, Rui Hu, Pranaab Dhawan, and Raquel Urtasun. Conditional entropy coding for efficient video compression. In *ECCV*, 2020.

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*, pp. 11976–11986, 2022.

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*, pp. 11006–11015, 2019.

Ishit Mehta, Michaël Gharbi, Connelly Barnes, Eli Shechtman, Ravi Ramamoorthi, and Manmohan Chandraker. Modulated periodic activations for generalizable local functional representations, 2021.

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*, pp. 297–302, 2020.

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, 2020.

Hao Ouyang, Tengfei Wang, and Qifeng Chen. Internal video inpainting by implicit long-range propagation. In *International Conference on Computer Vision (ICCV)*, 2021.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Yunchen Pu, Zhe Gan, Ricardo Henao, Xin Yuan, Chunyuan Li, Andrew Stevens, and Lawrence Carin. Variational autoencoder for deep learning of images, labels and captions. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/eb86d510361fc23b59f18c1bc9802cc6-Paper.pdf.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5301–5310. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/rahaman19a.html.

Oren Rippel, Sanjay Nair, Carissa Lew, Steve Branson, Alexander G. Anderson, and Lubomir Bourdev. Learned video compression. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

Oren Rippel, Alexander G. Anderson, Kedar Tatwawadi, Sanjay Nair, Craig Lytle, and Lubomir Bourdev. Elf-vc: Efficient learned flexible-rate video coding, 2021.

Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis, 2021.

Assaf Shocher, Nadav Cohen, and Michal Irani. "zero-shot" super-resolution using deep internal learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions, 2020.

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. doi: 10.1109/TCSVT.2012.2221191.

Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains, 2020.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9446–9454, 2018.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. ICML '08, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605582054.

Haiqiang Wang, Weihao Gan, Sudeng Hu, Joe Yuchieh Lin, Lina Jin, Longguang Song, Ping Wang, Ioannis Katsavounidis, Anne Aaron, and C-C Jay Kuo. Mcl-jcv: a jnd-based h. 264/avc video quality assessment dataset. In *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 1509–1513. IEEE, 2016.

Tengfei Wang, Hao Ouyang, and Qifeng Chen. Image inpainting with external-internal learning and monochromic bottleneck. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5120–5129, 2021.

T. Wiegand, G.J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the h.264/avc video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):560–576, 2003a. doi: 10.1109/TCSVT.2003.815165.

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, 2003b.

Chao-Yuan Wu, Nayan Singhal, and Philipp Krahenbuhl. Video compression through image interpolation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

Haotian Zhang, Long Mai, Ning Xu, Zhaowen Wang, John Collomosse, and Hailin Jin. An internal learning approach to video inpainting. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2720–2729, 2019.

A APPENDIX

A.1 VIDEO DECODING

We firstly show command to evaluate decoding speed of H.264 and H.265:
ffmpeg -threads ThreadsNum -i Video -preset medium -f null -benchmark -

And we also show quantitative decoding results in Table 12, 13, and Table 14. In Table 14, we can further increase video decoding speed with a smaller channel width (*i.e.*, a big reduction factor $r = 2$).

Table 12: Decoding FPS ↑

PSNR	32	35	37
H.264	279.7	240.9	192.7
H.265	211.9	163.2	132.5
DCVC	4.7	4.6	4.5
HNeRV	**395.9**	**332.7**	**224.8**

Table 13: Decoding time (s) ↓

# Frames	100%	50%	25%
H.264	0.548	0.480	0.343
H.265	0.809	0.708	0.506
DCVC	27.913	24.424	17.446
HNeRV	**0.397**	**0.198**	**0.099**

Table 14: HNeRV Decoding FPS

PSNR	32	35	37
r=1.5	395.9	332.7	224.8
r=1.75	397.4	373.8	320.7
r=2	**405.5**	**383.3**	**350.5**

A.2 VIDEO COMPRESSION

Then we show the details for downstream tasks of video compression, which can be divided into three steps: global unstructure pruning, quantization, and entropy encoding.

1) Model Pruning. Given a pre-trained model, we use global unstructured pruning to reduce the model size, where parameters below a threshold are pruned and set as zero. For a model parameter θ_i, $\theta_i = \begin{cases} \theta_i, & \text{if } \theta_i \geq \theta_q \\ 0, & \text{otherwise,} \end{cases}$ where θ_q is the q percentile value for all model parameters θ. As a normal practice, we fine-tune the model to regain the representation after pruning.

2) Model and embedding quantization. Model quantization and embedding quantization follow the same scheme. Given an vector μ, we linearly map every element to the closest integer,

$$\mu_i = \text{Round}\left(\frac{\mu_i - \mu_{\text{min}}}{\text{scale}}\right) * \text{scale} + \mu_{\text{min}}, \quad \text{where scale} = \frac{\mu_{\text{max}} - \mu_{\text{min}}}{2^{\text{b}} - 1} \tag{3}$$

μ_i is one vector element, 'Round' is a function that rounds to the closest integer, 'b' is the bit length for quantization, μ_{max} and μ_{min} are the max and min value of vector μ, and 'scale' is the scaling factor. For scaling factor and zero points at this step, we can also try other methods instead of current min-max one, like choosing 2^b evenly-distributed values to minimum the mean square error.

3) Entropy encoding. Finally, we use entropy encoding to further reduce the size. Specifically, we leverage Huffman coding (Huffman, 1952) for quantized weights and get lossless compression.

A.3 MORE VISUALIZATIONS

We show more visualizations for video regression (Figure 9), video interpolation (Figure 10), and video inpainting (Figure 11.

A.4 HNERV ARCHITECTURE DETAILS

We also provide architecture detials for HNeRV models in various tasks and datasets in Table 15, with total size, strides list, encoder dimension $c1$, embedding dimension $c2$, channel width of decoder input $c2$, channel reduction r, lowest channel width Ch_{min}, min and max kernel size Ch_{min}, Ch_{max}

A.5 PER-VIDEO COMPRESSION RESULTS

We also show video compression results for each video in Figure 12.

Figure 9: **Video regression** results. **Left**: ground truth. **Middle**: NeRV output. **Right**: HNeRV output.

Figure 10: **Interpolation** results.

Table 15: HNeRV architecture details

Video resolution	size	strides	c1	d	c2	r	Ch_{min}	K_{min}, K_{max}
640×1280	0.35	5,4,4,2,2	64	16	32	1.2	12	1,5
640×1280	0.75	5,4,4,2,2	64	16	48	1.2	12	1,5
640×1280	1.5	5,4,4,2,2	64	16	68	1.2	12	1,5
640×1280	3	5,4,4,2,2	64	16	97	1.2	12	1,5
480×960	3	5,4,3,2,2	64	16	110	1.2	12	1,5
960×1920	3	5,4,4,3,2	64	16	92	1.2	12	1,5

Figure 11: **Inpainting** results.

A.6 VIDEO DATASET

We show video frames for 'bunny' and UVG dataset in Figure 13. Note that for video of huge dynamics like Jockey and Setgo, HNeRV is a little struggle compared to other compression methods.

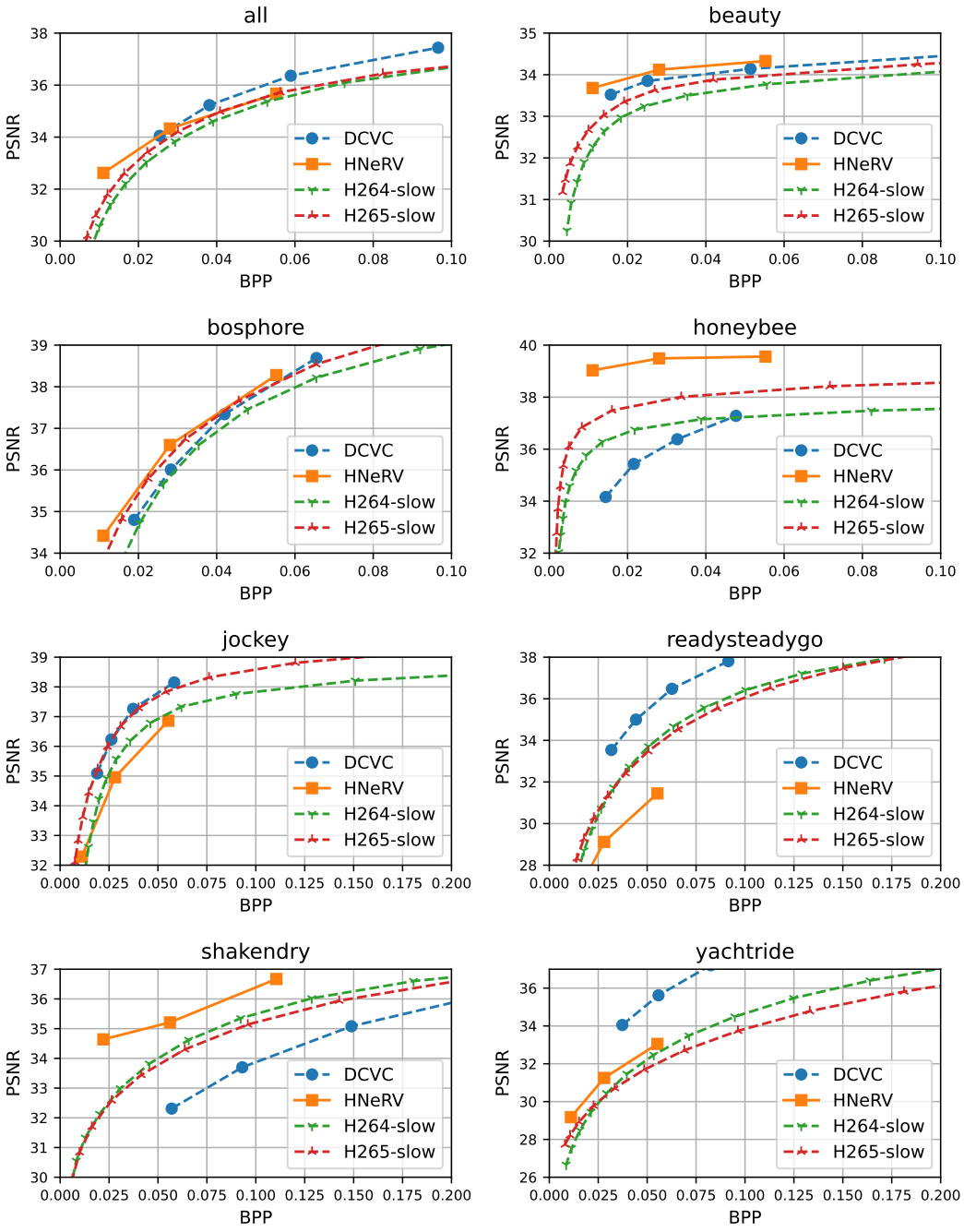

Figure 12: **Compression** results for all videos

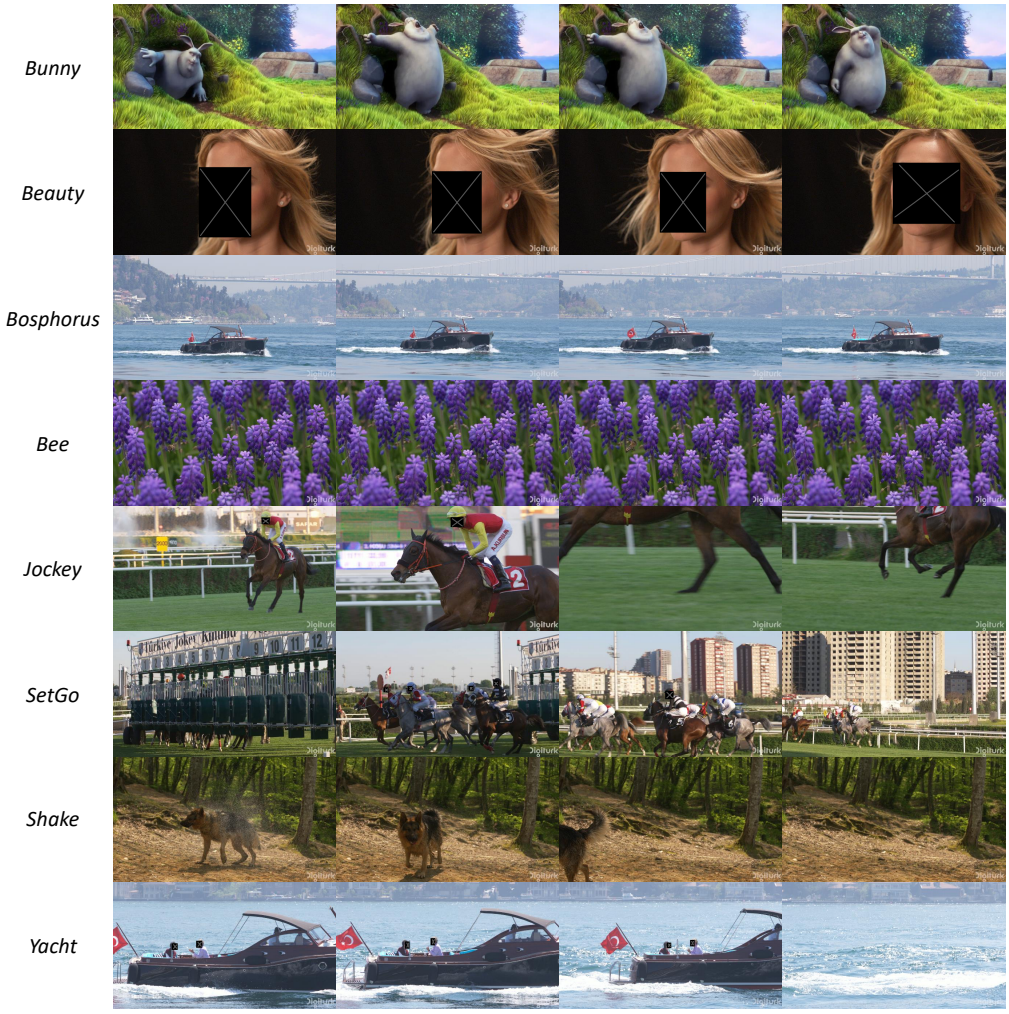

Figure 13: Video frames of **bunny and UVG dataset**. We mask pepple faces to avoid legal and copyright issues.