DNeRV: Modeling Inherent Dynamics via Difference Neural Representation for Videos

Qi Zhao
Nanjing University
qizhao@smail.nju.edu.cn

M. Salman Asif
University of California Riverside
sasif@ucr.edu

Zhan Ma
Nanjing University
mazhan@nju.edu.cn

Abstract

Existing implicit neural representation (INR) methods do not fully exploit spatiotemporal redundancies in videos. Index-based INRs ignore the content-specific spatial features and hybrid INRs ignore the contextual dependency on adjacent frames, leading to poor modeling capability for scenes with large motion or dynamics. We analyze this limitation from the perspective of function fitting and reveal the importance of frame difference. To use explicit motion information, we propose Difference Neural Representation for Videos (DNeRV), which consists of two streams for content and frame difference. We also introduce a collaborative content unit for effective feature fusion. We test DNeRV for video compression, inpainting, and interpolation. DNeRV achieves competitive results against the state-of-the-art neural compression approaches and outperforms existing implicit methods on downstream inpainting and interpolation for 960 × 1920 videos.

1. Introduction

In recent years, implicit neural representations (INR) have gained significant attention due to their strong ability in learning a coordinate-wise mapping of different functions. The main principle behind INR is to learn an implicit continuous mapping \( f \) using a learnable neural network \( g_\theta(\cdot) : \mathbb{R}^m \rightarrow \mathbb{R}^n \). The idea was first proposed for the neural radiance fields (NeRF) [28] and since then has been applied to various applications [4, 8, 58]. INR attempts to approximate the continuous \( f \) by training \( g_\theta \) with \( m \)-dimensional discrete coordinates \( x \in \mathbb{R}^m \) and corresponding quantity of interest \( y \in \mathbb{R}^n \). Once trained, the desired \( f \) can be fully characterized using \( g_\theta \) or the weights \( \theta \), and it would be benefit for the tasks which need to model the intrinsic generalization for given data, such as interpolation or inpainting tasks shown in Fig. 1.

The success of INR can be attributed to the insight that a learnable and powerful operator with a finite set of data samples \( S = \{ x_i, y_i \}_{i=0}^N \) can fit the unknown mapping \( f \). The accuracy of the mapping depends on the number of samples \( N \) and the complexity of the map \( f \). INR for

Figure 1. Results of the proposed DNeRV with 3M parameters for (a) video interpolation on UVG [26] and (b) video inpainting on Davis [30]. The superior performance shows the effectiveness and generalization capability of DNeRV on neural representation for videos.
videos requires a large \( N \), which primarily depends on the size and internal complexity of the video sequence. Furthermore, video representation is complicated due to different sampling or frames-per-second (FPS) rates of videos. Large motion (in terms of direction, speed, rotation, or blur) and transformations of the objects or scene can make adjacent frames quite different. Figure 2 shows examples of such mismatch between consecutive frames, which we attribute to adjacent dynamics.

Adjacent dynamics are the short-term transformations in the spatial structure, which are difficult to represent using existing methods for neural representation of videos (NeRV). Existing NeRV approaches can be broadly divided into two groups: (1) Index-based methods, such as [4] and [21], use positional embedding of the index as input and lack content-specific information for given videos. (2) Hybrid-based methods [3] use frames for index embedding and neglect the temporal correlation between different frames. Therefore, neither index nor frame-based NeRV are effective against adjacent dynamics.

In this work, we propose Difference NeRV (DNeRV) that attempts to approximate a dynamical system by absorbing the difference of adjacent frames, \( y_t^D = y_t - y_{t-1} \) and \( y_{t+1}^D = y_{t+1} - y_t \), as a diff stream input. Further analysis for the importance of diff stream is presented in Section 3. An illustration of DNeRV pipeline is presented in Figure 3. Diff encoder captures short-term contextual correlation in the diff stream, which is then merged with the content stream for spatiotemporal feature fusion. In addition, we propose a novel gated mechanism, collaborative content unit (CCU), which integrates spatial features in the content stream and temporal features in the diff stream to obtain accurate reconstruction for those frames with adjacent dynamics.

The main contribution of this paper are as follows. 

- Existing NeRV methods cannot model content-specific features and contextual correlations simultaneously. We offer an explanation using adjacent dynamics. Furthermore, we reveal the importance of diff stream through heuristic analysis and experiments.
- We propose the Difference NeRV, which can model the content-specific spatial features with short-term temporal dependence more effectively and help network fit \( \ldots \) implicit mapping efficiently. We also propose a collaborative content unit to merge the features from two streams adaptively.
- We present experiments on three datasets (Bunny, UVG, and Davis Dynamic) and various downstream tasks to demonstrate the effectiveness of the proposed method. The superior performance over all other implicit methods shows the efficacy of modeling videos with large motion. As a result, DNeRV can be regarded as a new baseline for INR-based video representation.

2. Related Work

Implicit neural representations (INRs) have been used in various vision tasks in recent years [6, 28]. In 3D vision, [24, 27, 31, 51] aim to use INRs from static and simple to dynamic and complicated visual data. In image analysis, INRs have been used to learn the mapping between 2D spatial coordinates \( x \in \mathbb{R}^2 \) and corresponding RGB value \( y \in \mathbb{R}^3 \) via various positional embedding techniques [9, 39, 43, 55] or meta-learning [40]. In video analysis, INRs learn the mapping from frame index \( t \in \mathbb{R} \) to RGB frame \( y \in \mathbb{R}^{3 \times w \times h} \). In other visual tasks, INRs can encode both spatial coordinates and specific feature vectors [36, 53].

Neural representation for videos (NeRV) methods can be broadly divided into two groups. Index-based NeRV methods use the positional embedding of \( t \) as the input [4, 21, 25]. The follow-up work has improved network structure for acceleration or disentangled feature modeling, but those methods could not capture the content-specific information, causing spatial redundancy. Hybrid-based NeRV [3] provides an insightful view by treating the current frame itself as the index embedding. The method shows much better performance over index-based ones via content-adaptive fashion. The main limitation of hybrid NeRV is that they ignore temporal relationship between frames, resulting in poor performance with adjacent dynamics.

Video compression methods based on INR use the traditional pipeline but change some intermediate components into networks [18, 19, 22, 23, 33]. However, their utility is limited by large number of parameters and computational redundancy, causing long coding time and limited generalized ability. One significance of INRs for video is that the video compression can be regarded as a model compression problem, and the existing model compression and acceleration techniques [11, 12] could be used for INR-based network. Neural compression is expected to break through the limitation of traditional pipeline with the help of INRs, resulting in better performance.

Two stream vision and fusion resemble the idea of adopting multi-stream mechanism for disentangling features learning in video analysis. [35] observe the importance in typical flow, which is widely used in action recogni-
tion [2, 10, 38] or other visual tasks [16, 42]. Separating motion and content as two stream is also widely used in video prediction [44] and image-to-video generation [57]. Furthermore, hidden state and time-dependent input in RNN/LSTM could be thought as different stream [12, 14, 46]. More effective fusion module for different stream information is a fundamental task in sequence learning, generating lots of novel gated mechanism [13, 45, 56]. The two-stream fusion aims to reduce the spatiotemporal redundancy caused by the inherent continuity of video, which motivated us to propose DNeRV based on diff stream. Additionally, inspired by the gated mechanism in RNN, we introduce a fusion unit for adaptive feature fusion.

3. Motivation

In this section, we discuss how the frame differences benefit the adjacent dynamics. A video sequence is a collection of images captured over time; therefore, motion of different objects in video is often spatially continuous and smooth. The main objective of INR is to learn an implicit neural representation for the given video, consisting of embeddings, lightweight CCU, and decoder.

Let us note that although NeRV methods use various positional encoding techniques [39, 43] to map the discrete index into a higher dimension for better performance (we treat the frame itself as an embedding of the index for hybrid-based methods, the network $g_{θ}$ is actually a mapping from a time index in $Φ$ to $R^{3 × H × W}$. Once trained, we can generate frames as

$$g_{θ}(t) = y_t, \quad t \in \{1, 2, \ldots , N\}. \quad (2)$$

Since video is usually continuous along spatial and temporal dimensions, $f$ can also be described as a dynamical system:

$$\dot{f} = A(f(t), t),$$

where $A(·, ·)$ represents a nonlinear function determined by the system. The goal of INR is to use the neural network $g_{θ}$ to fit the $f$ with the training tuples $\{(t, f(t))\}_{t=0}^{N}$. However, in general, the training tuple $(t, f(t))$ is not enough for networks to learn a dynamical system. For example, we consider a simple time-invariant system:

$$\dot{f} = A \cdot f(t), \quad f(0) = y_0,$$

where $A$ is a constant. The solution of the problem can be written as $f(t) = \exp(At)y_0$. If $g_{θ}$ is a general feed-forward ReLU network, then we can achieve $\|g_{θ}(t) − f(t)\| ≤ 0$ for all $t$ under some conditions. In particular, we require $O(\text{log}(ε)^{d})$ non-zero hidden units to train $g_{θ}$ using training tuples $\{(t, f(t))\}$, where $d$ is the dimension of $f(t)$ if $f$ is invertible [34, 49].

Interestingly, if we use $\{(t, \dot{f}(t)), f(t)\}$ as training tuples, then one-layer linear network is sufficient to learn $f(t)$ and

$$\|g_{θ}(t, \dot{f}(t)) − f(t)\| = 0, \quad \forall t \in [1, N].$$

This is because the learning problem simplifies to learning the constant $A$ instead of learning a continuous function $f(·)$. Hence, considering the high order differentials $(\dot{f} \ddot{f} \cdots)$ as the network input can significantly benefit in using a dynamical system.
For hybrid-based INR methods, we introduce the diff stream $y_t^D = y_t - y_{t-1}$ as an auxiliary input with the main content stream $y_t$ (for more details, see the ablation studies). The difference between two adjacent frames could be treated as discrete differential $\nabla f$,

$$\nabla f\bigg|_{\tau=t} = \frac{f(\tau) - f(\tau - \Delta \tau)}{\Delta \tau} \bigg|_{\tau=t}$$ \hfill (3)$$

where $\Delta \tau$ represents the parameter quantity, we reduce the channels in the last two stages of the decoder. Final output is given as follows

Decoder. We adopt NeRV block as the basic block in each decoding stage. For the embeddings from encoder, they would be fused in same shape. We explored different connections for fusion, and the final architecture is shown in Fig. 4. For the fusion module, it could be sum fusion, conv fusion, or other gated mechanisms. Once the features are merged, they pass through other stages and map into pixel domain via channel reduction.

Fusion. To fuse features from diff stream and content stream, conv fusion $s_t = \text{Conv}(b_t^C) + b_t^D$ or concat fusion $s_t = \text{concat}(b_t^C, b_t^D)$ may not be suitable. This is because the features come from different domains as $b_t^D$ is the discretization of differential of $f$, while $b_t^C$ represents the value of $f$. To merge the two streams, inspired by gated mechanism in temporal sequence learning [7, 45, 54], we introduce a collaborative content unit (CCU). Our motivation is that diff stream needs to be fused with content stream collaboratively to obtain the refined content features by adding higher-order information. Specifically, the content stream could be treated as hidden state in an RNN-fashion, which contains time-varying features helping reconstruction. CCU can be represented as

$$z_t^D = \text{GELU}(\text{PS}(\text{W}_z^{1 \times 1} \ast (b_t^D))),$$
$$\tilde{b}_t^C = \text{BLOCK}(\tilde{b}_t^C),$$
$$u_t = \text{tanh}(W_{ub} \ast \tilde{b}_t^C + W_{uz} \ast z_t^D),$$
$$v_t = \text{Sigmoid}(W_{vb} \ast \tilde{b}_t^C + W_{zu} \ast z_t^D),$$
$$s_t = u_t \odot v_t + (1 - v_t) \odot \tilde{b}_t^C,$$

where $\text{PS}$ is PixelShuffle, $*$ is convolution operator and $\odot$ is Hadamard product. $v_t$ could be treated as the update gate in GRU [7], to decide how much information in content feature could be remained. Finally, two streams are merged and the adjacent dynamics collaboratively captured by CCU can help network $g_y$ learn the implicit mapping. To balance the parameter quantity, we reduce the channels in the last two stages of the decoder. Final output is given as follows

$$s_t = \text{FUSION}(b_t^C, b_t^D),$$
$$\tilde{y}_t = \text{Sigmoid}(\text{W}_y^{1 \times 1} \ast (\text{BLOCK}(s_t))),$$

where $\text{FUSION}$ represents CCU in our implementation.

Figure 4. Architecture of decoder with CCU as fusion module for 960 × 1920.
Discussion of optical flow. Although optical flow captures adjacent temporal relationship as well as the difference stream, we could not achieve comparable performance when using optical flow. The main reason is that INR-based video representation task is different from semantic video tasks. In the case of NeRV, pixel-level features that directly help decoder reconstruction are more vital. More details can be found in the supplementary materials.

Comparison with NeRV. Now, we look back to the philosophy of NeRV and compare it with DNeRV. For NeRV, we search for an operator \( g_0 \) by solving the following optimization problem:

\[
\arg\min_\vartheta \| g_0(h(t)) - f(t) \|, \tag{7}
\]

where \( h \) represents the embedding of time index \( t \) and \( f(t) = y \) represents the frames in pixel-domain. In the case of hybrid methods [3], we solve the following optimization problem:

\[
\arg\min_\vartheta \| g_0(f(t)) - f(t) \|, \tag{8}
\]

where the embedding is the frame itself. The hybrid method attempts to fit a series of invariant point transformations in function space for every training tuple \((f, y)\). This explains why existing methods only work well on fixed background scene with few dynamics, such as “HoneyBee” or “ShakeANDry” in UVG. In other words, they only take effect when \( y_i \) is within a small neighborhood of training samples. In other words, \( g_0 \) only learns the behavior of \( f \) near the mean of whole training samples, where adjacent dynamics would not be apparent. In the case of DNeRV, we solve the following optimization problem:

\[
\arg\min_\vartheta \| g_0(f, \nabla f, \ldots, \nabla^{(I)} f) - f \|, \tag{9}
\]

where \( i = 1 \) in our realization. DNeRV attempts to learn a dynamical system that represents \( f \) in implicit way.

5. Experiments

Settings. We verify DNeRV on Bunny [18], UVG [26] and DAVIS Dynamic. Bunny owns 132 frames for \( 720 \times 1280 \). UVG has 7 videos at \( 1080 \times 1920 \) with length of 600 or 300. DAVIS Dynamic is a subset of DAVIS16 validation [30] which containing 22 videos\(^1\) in \( 1080 \times 1920 \). Most of the selected videos contain dynamic scenes or moving targets, which are quite difficult for existing methods. Following the settings in [3] for fair comparison, we center-crop the videos into \( 640 \times 1280 \) or \( 960 \times 1920 \) and reshape UVG into \( 480 \times 960 \) for additional comparison.

During training, we adopt Adam [17] as the optimizer with learning rate of \( 5 \times 10^{-4} \) and cosine annealing learning rate schedule [47] and the batch size is set to 1. We use PSNR and SSIM to evaluate the video quality. The stride list, kernel size and reduction rate remain to be same as [3], except for the channels in the last two stages of decoder.

We compare DNeRV with others in video regression and three downstream visual tasks consist of video compression, interpolation and inpainting. In video interpolation we train the model on the sequence of even frames and test it on the odd frames sequence from UVG and DAVIS Dynamics. In video inpainting, we directly use the models trained in regression without any fine-tuning and test them using masked videos in disperse mask or central mask from DAVIS Dynamics. All experiments are conducted in PyTorch with GPU RTX2080ti, with 3M size and 300 epochs unless otherwise clarified.

Discussion of Loss functions. We conduct loss objective ablation between L2 and L1+SSIM, shown in Tab. 1c and Tab. 1d. L1+SSIM is the loss objective in NeRV [4]. \( L(\hat{y}, y) = \alpha \| \hat{y} - y \|_1 + (1 - \alpha)(1 - SSIM(\hat{y}, y)) \), \( \alpha = 0.7 \). Owing to L1 norm is the convex approximation of L0 norm [1], it is better for scenes with complex textures and high-frequency subtle spatial structure but few motion between

---

\(^1\)blacksaw, bmx-bumps, camel, breakdance, car-roundabout, bmx-trees, car-shadow, cows, dance-twirl, dog, car-turn, dog-agility, drift-turn, goat, libby, mallard-fly, mallard-water, parkour, ute, scooter-black, strolle.

Table 1. Video regression results on Bunny and UVG, where DNeRV uses different loss functions for ablation.
frames. While during experiments, L2 is set as default loss function as it is better for low-frequency scenes with large motion.

5.1. Video Regression

Bunny. The comparison between different implicit methods trained in 300 epochs on Bunny is shown in Table 1a. DNeRV outperforms others. Also, we compare various implicit methods in same 0.75M but different training epochs reported in Table 1b. DNeRV surpasses other methods relying on the reasonable structure with no additional gradient propagation difficulties.

UVG. The PSNR results on UVG are given in Table 1c and Table 1d. DNeRV shows large improvements at resolution 960×1920. The excellent results are attributed to the high-resolution diff stream, containing a great deal of content-specific spatial information. The adjacent dynamic hidden among frames could be captured in a more reasonable way, helping the network converge faster.

DAVIS Dynamic. The results of regression on DAVIS Dynamic are shown together with inpainting results in Table 2. DNeRV achieves an impressive performance when processing those videos with complicated scene transformation or object movements. Another difficulty of DAVIS Dynamic videos is that the number of frames is quite smaller, e.g., 25 for “dog-agility” and 43 for “Scooter-black”. Fewer frames and ubiquitous adjacent dynamics present extreme difficulty for implicit methods to fit the latent mapping, indicating the effectiveness of DNeRV.

5.2. Video Compression

We show the compression results in PSNR and SSIM on UVG dataset in Figure 5. Without pruning and only 8 bits quantization with entropy encoding adopted, DNeRV outperforms other NeRV methods especially on PSNR. DNeRV optimizes the network structure under the condition of parameter amount, thus reducing the redundancy in weights. Although it couldn’t maintain performance in 40% pruning like [4,21], the 10% pruned DNeRV is still competitive compared with other implicit methods. We also report VMAF [32] results on UVG in the appendix.

Table 3. Ablation study for fusion module and diff embedding size, training on Bunny in 300 epochs. A indicates sum fusion, C is conv fusion and U is the CCU. The final version of DNeRV consists of diff embedding in shape of 2×40×80, 3rd stage where merging and CCU as the fusion module. The first two rows which are marked as N/A represent HNeRV baseline, where the size of diff embedding is not available.

Table 4. Ablation study of various difference on Bunny in 0.35M and 300 epochs, where ∆ is first order and ∇ is second order difference.

Table 5. Complexity comparison.

M-LVC [22], DCVC [19] and ELF-VC [33]. Without any specific modification, DNeRV is better than traditional video codecs H.264 or H.265 in both PSNR and SSIM, and it is also competitive with the state-of-the-art deep compression methods. We will explore the potential of DNeRV in video compression in the follow-up studies. The complexity comparison of video decoding with two rapid neural compression methods [20,37] in model size, decoding time and FPS is shown in Table 5.

5.3. Video Interpolation

Since the INR-based video representation is over-fitting to the given video, we evaluate the generalization of different methods by video interpolation task. We follow the setting in [3] and demonstrate the quantitative results on UVG in Table 6 and qualitative results in Figure 6. Thanks to the consideration of adjacent dynamics, DNeRV outperforms implicit methods especially on the videos owning dy-
2. Table. Video regression and inpainting results on 960×120 Davis Dynamic in PSNR/SSIM, larger is better.

<table>
<thead>
<tr>
<th>video</th>
<th>NeRV</th>
<th>E-NeRV</th>
<th>HNeRV</th>
<th>DNeRV</th>
<th>NeRV-S</th>
<th>HNeRV-S</th>
<th>DNeRV-S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blackswan</td>
<td>29.48/0.812</td>
<td>29.38/0.867</td>
<td>30.35/0.891</td>
<td>30.92/0.913</td>
<td>26.51/0.825</td>
<td>29.01/0.896</td>
<td>24.64/0.783</td>
</tr>
<tr>
<td>Bmx-bumps</td>
<td>29.42/0.864</td>
<td>28.90/0.851</td>
<td>29.98/0.872</td>
<td>30.59/0.890</td>
<td>23.16/0.726</td>
<td>27.50/0.819</td>
<td>20.39/0.665</td>
</tr>
<tr>
<td>Bmx-trees</td>
<td>26.24/0.789</td>
<td>27.26/0.876</td>
<td>28.76/0.861</td>
<td>29.63/0.882</td>
<td>22.93/0.720</td>
<td>26.57/0.841</td>
<td>20.26/0.653</td>
</tr>
<tr>
<td>Breakdance</td>
<td>26.45/0.915</td>
<td>28.33/0.941</td>
<td>30.45/0.961</td>
<td>30.88/0.968</td>
<td>27.36/0.945</td>
<td>29.16/0.961</td>
<td>23.84/0.907</td>
</tr>
<tr>
<td>Camel</td>
<td>24.81/0.781</td>
<td>25.85/0.804</td>
<td>26.71/0.844</td>
<td>27.38/0.887</td>
<td>20.94/0.661</td>
<td>24.71/0.832</td>
<td>21.85/0.733</td>
</tr>
<tr>
<td>Car-round</td>
<td>24.60/0.857</td>
<td>26.01/0.912</td>
<td>27.75/0.912</td>
<td>29.35/0.937</td>
<td>23.96/0.752</td>
<td>27.63/0.854</td>
<td>22.06/0.810</td>
</tr>
<tr>
<td>Car-shadow</td>
<td>26.41/0.871</td>
<td>30.41/0.922</td>
<td>31.32/0.916</td>
<td>31.95/0.944</td>
<td>25.87/0.875</td>
<td>27.90/0.914</td>
<td>28.67/0.908</td>
</tr>
<tr>
<td>Car-turn</td>
<td>27.45/0.811</td>
<td>29.02/0.888</td>
<td>29.65/0.879</td>
<td>30.25/0.892</td>
<td>23.96/0.752</td>
<td>27.63/0.854</td>
<td>24.43/0.773</td>
</tr>
<tr>
<td>Cows</td>
<td>22.55/0.702</td>
<td>23.74/0.819</td>
<td>24.11/0.792</td>
<td>24.88/0.827</td>
<td>21.37/0.682</td>
<td>22.91/0.770</td>
<td>20.81/0.668</td>
</tr>
<tr>
<td>Dance-twirl</td>
<td>25.79/0.797</td>
<td>27.07/0.864</td>
<td>28.19/0.845</td>
<td>29.13/0.870</td>
<td>23.05/0.743</td>
<td>26.13/0.830</td>
<td>21.10/0.704</td>
</tr>
<tr>
<td>Dog</td>
<td>28.17/0.795</td>
<td>30.40/0.882</td>
<td>30.96/0.898</td>
<td>31.32/0.905</td>
<td>25.34/0.739</td>
<td>27.43/0.824</td>
<td>23.09/0.677</td>
</tr>
<tr>
<td>Dog-ag</td>
<td>29.08/0.821</td>
<td>29.30/0.905</td>
<td>28.75/0.893</td>
<td>29.94/0.923</td>
<td>27.70/0.884</td>
<td>28.28/0.913</td>
<td>25.12/0.856</td>
</tr>
<tr>
<td>Drift-straight</td>
<td>26.65/0.860</td>
<td>29.10/0.941</td>
<td>30.80/0.932</td>
<td>31.50/0.940</td>
<td>24.79/0.833</td>
<td>27.00/0.892</td>
<td>20.15/0.725</td>
</tr>
<tr>
<td>Drift-turn</td>
<td>26.70/0.812</td>
<td>27.94/0.875</td>
<td>29.72/0.834</td>
<td>30.37/0.862</td>
<td>22.27/0.677</td>
<td>26.20/0.816</td>
<td>19.95/0.636</td>
</tr>
<tr>
<td>Goat</td>
<td>23.90/0.746</td>
<td>25.25/0.855</td>
<td>26.62/0.858</td>
<td>27.79/0.887</td>
<td>21.11/0.675</td>
<td>22.65/0.755</td>
<td>20.21/0.639</td>
</tr>
<tr>
<td>Libby</td>
<td>29.08/0.821</td>
<td>31.43/0.890</td>
<td>32.69/0.917</td>
<td>33.43/0.927</td>
<td>27.59/0.825</td>
<td>29.51/0.875</td>
<td>24.33/0.752</td>
</tr>
<tr>
<td>Mallard-fly</td>
<td>26.83/0.757</td>
<td>28.84/0.847</td>
<td>29.22/0.848</td>
<td>28.77/0.833</td>
<td>23.81/0.709</td>
<td>25.83/0.774</td>
<td>20.83/0.618</td>
</tr>
<tr>
<td>Mallard-water</td>
<td>25.20/0.824</td>
<td>27.28/0.896</td>
<td>29.08/0.908</td>
<td>29.60/0.922</td>
<td>23.55/0.803</td>
<td>24.11/0.845</td>
<td>21.18/0.743</td>
</tr>
<tr>
<td>Parkour</td>
<td>25.14/0.794</td>
<td>25.31/0.845</td>
<td>26.56/0.851</td>
<td>25.75/0.827</td>
<td>21.32/0.685</td>
<td>24.51/0.799</td>
<td>19.97/0.650</td>
</tr>
<tr>
<td>Rollerblade</td>
<td>29.28/0.898</td>
<td>33.32/0.964</td>
<td>32.19/0.935</td>
<td>32.49/0.940</td>
<td>29.32/0.911</td>
<td>30.41/0.931</td>
<td>27.31/0.901</td>
</tr>
<tr>
<td>Scooter-black</td>
<td>22.73/0.835</td>
<td>23.79/0.927</td>
<td>27.38/0.923</td>
<td>28.53/0.940</td>
<td>21.05/0.794</td>
<td>24.27/0.897</td>
<td>19.76/0.789</td>
</tr>
<tr>
<td>Stroller</td>
<td>29.89/0.859</td>
<td>30.37/0.914</td>
<td>31.31/0.894</td>
<td>32.73/0.928</td>
<td>25.90/0.796</td>
<td>28.46/0.876</td>
<td>22.86/0.734</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td>26.56/0.819</td>
<td>28.19/0.887</td>
<td>29.21/0.886</td>
<td>29.86/0.902</td>
<td>24.09/0.774</td>
<td>26.53/0.854</td>
<td>22.40/0.742</td>
</tr>
</tbody>
</table>

Table 6. Video regression and inpainting results on 960×120 UVG in PSNR.

5.5. Ablation Study

We provide various ablation studies on Bunny with P-SSNR and MS-SSIM as the metrics, reported in Tab 3 and Tab 4.

Diff stream. We compare the backward difference, forward difference, central difference and second order difference as input and merge one of them with content stream in 3rd stage. The results are shown in Tab 4, indicating that higher-order difference may not be helpful for fitting. We concatenate both backward and forward difference as our default diff stream.

Param quantity. To verify the effectiveness of diff stream, we compare the hybrid ones without diff stream in varying model size (by changing the channels in decoder) with DNeRV. More parameters may not attributed to better performance. The params, converge speed and generalization ability should be balanced.

mbedding. The smaller shape of diff embedding
would reduce both the model size and reconstruction quality. The shape in $2 \times 40 \times 80$ is adopted as default.

**Fusion module.** The ablation results reported in Tab 3 and Tab. 4 verify the effectiveness of proposed CCU. Compared with conv fusion or sum fusion, CCU improves PSNR by adaptively merging two-stream features.

**Fusion stages.** It is vital for DNeRV to select the stage where two streams are merged, we do ablations on every possible connection. Decoder should balance the params and the difficulty of gradient descent. DNeRV adopts merging in 3rd stage without sum fusion. Further we upgrade conv fusion to CCU.

### 6. Conclusion

In this paper, we propose the difference neural representation for videos (DNeRV) for modeling the inherent dynamics of contextual frames. Relying on the diff strea collaborative content unit, DNeRV retains its advantages in reconstruction quality for video regression, compression, inpainting and interpolation. Consequently, the experimental results show that the proposed DNeRV could achieve effective and robust representation for videos by achieving a better approximation to the implicit mapping than existing NeRV methods.

**Future directions.** DNeRV shows its potential on various visual tasks. The improved task-specific approach based on DNeRV is promising to challenge the state-of-the-art methods. Also, rigorous theoretical analysis needs to be improved for INR-based networks $g_{\theta}$ fitting the continuous $f$ on finite training tuple via gradient descent.

**Acknowledgements.** We thank Xingyu Xie for comments that greatly improved the manuscript. The work was supported in part by National Key Research and Development Program of China (2022YFF0902402).
References


Xihua Sheng, Jiahao Liu Li, Bin Li, Li Li, Dong Liu, and Yan Lu. Temporal context mining for learned video compression. arXiv:2111.13850, 2022. 6


Shaofeng Zhi, Tristan Laidlow, Stefan Leutenegger, and Andrew J. Davison. In-place scene labelling and understanding with implicit scene representation. In ICCV, 2021. 1