We study how to represent a video with implicit neural representations (INRs). Classical INRs methods generally utilize MLPs to map input coordinates to output pixels. While some recent works have tried to directly reconstruct the whole image with CNNs. However, we argue that both the above pixel-wise and image-wise strategies are not favorable to video data. Instead, we propose a patch-wise solution, PS-NeRV, which represents videos as a function of patches and the corresponding patch coordinate. It naturally inherits the advantages of image-wise methods, and achieves excellent reconstruction performance with fast decoding speed. The whole method includes conventional modules, like positional embedding, MLPs and CNNs. We also introduce AdaIN to enhance intermediate features. Extensive experiments have demonstrated its effectiveness in several video-related tasks, such as video compression and video inpainting.

Index Terms— Implicit neural representation, video representation, video compression, video inpainting

1. INTRODUCTION

With the fast development of streaming media, numerous video data have been widely filled in our daily life. However, the large file sizes, especially for high-resolution (1080p-4k) videos, are becoming heavy burdens to storage and transmission. Traditional video representations, which explicitly represent videos as frame sequences, are not efficient enough to meet this challenge. Recently, Implicit Neural Representation [1, 2, 3, 4] (INR) has gained increasing attention. As a novel and effective representation method, INR is able to produce high-fidelity results for various data types, such as images [5], 3D shapes [6], and scenes [2].

For the implicit video representation, pixel-wise methods (like SIREN [5]) output the RGB value for each pixel according to the spatio-temporal coordinates (x, y, t). In contrast, NeRV [7] is proposed as an image-wise representation method, which represents a video as a function of time. However, neither pixel-wise nor image-wise representation is the most suitable strategy for video data. They could increase the network burden in different ways, resulting in unsatisfactory reconstruction results. Specifically, the pixel-wise method requires a large number of samples for each frame, which is inefficient for both encoding and decoding. The image-wise method could struggle to represent complex signals, such as high-resolution videos. It requires a much larger network to overfit the content and details of the whole frame, bringing additional computational cost. Therefore, we need a more appropriate and effective way to implicitly represent video data.

Inspired by the similarity of local adjacent pixels (widely exists in real-world signals [8]), we propose a patch-wise implicit representation method of videos. As the features in adjacent patches have great similarities, they can be easily represented by a single network. In our implementation, each frame of the video is divided into split patches, so that each patch has a corresponding spatial coordinate. We take the patch coordinate and timestamp as the input of the network. Then the network outputs the corresponding image patch by convolutional network. Our method not only enjoys the fast encoding and decoding speed as the image-wise method, but also reconstructs vivid high-frequency details.

Another point worth noting is that the normalization layer widely used in convolution neural networks could reduce the fitting ability of the network, which has also been found in NeRV [7]. This is because the common normalization operation is used to prevent overfitting, which will make the distribution of features unable to shift in the direction related to the target frame accurately. As an alternative, can we improve the fitting ability by aligning the mean and variance of features with the target frame? Based on the above consideration, we further introduce the Adaptive Instance Normalization layer (AdaIN) [9] to modulate features. This simple strategy can significantly improve the fitting ability of the network, especially for details. Finally, we name our whole method as Patch-wise Stylized Neural Representations for Videos (PS-NeRV).

We also explore some applications of our method, such as video compression and video inpainting tasks. Compared with NeRV, our method shows better compression potential. When the input video is masked, our method can generate high-quality inpainting output and even outperforms state-of-the-art video inpainting methods. To summarize, the main contributions of this work are as follows:

• We design a novel video implicit representation method PS-NeRV, which represents the video as image patches, and verify the effectiveness of the proposed method in representing the high-resolution details.

• We find that introducing the AdaIN layer to the ConvNets-based INR can significantly improve the fitting effect of the model.

• Our method shows excellent performance in several video-related applications, including video compression and video inpainting.

2. PROPOSED METHOD

2.1. Represent videos as images patches

For each video \( V = \{v^t\}_{t=1}^T \in \mathbb{R}^{T \times H \times W \times 3} \), we split each frame into \( N \times N \) patches and get \( \{v^t_p\}_{p=1}^{N^2} \in \mathbb{R}^{N^2 \times H/N \times W/N \times 3} \). Then,
2.2. Time-Coordinate Embedding

When the coordinates are used as the input of the neural network, it has been found [10, 11] that mapping them to a high embedding space can effectively improve the fitting effect of the network. In addition to the patch coordinate $i$, there is another input – timestamp $t$. We encode both inputs into embedding using Positional Encoding [11, 12, 13]

$$\Gamma(t, i) = (p(b^0 \pi t), p(b^1 \pi t), \ldots, p(b^{t-1} \pi t); p(b^0 \pi i), p(b^1 \pi i), \ldots, p(b^{t-1} \pi i)), \quad (1)$$

where $p(\cdot) = (\sin(\cdot), \cos(\cdot))$ maps $t$ and $i$ to their Fourier features. $b$ and $l$ are hyper-parameters of the networks. According to the length of the video and the number of patches, timestamp $t$ and coordinate $i$ are normalized to $(0, 1]$. Then, their embeddings will be concatenated together as the input of the network.

2.3. Network Architecture

The input of the time-coordinate embedding is sent to the subsequent layers of MLPs to get a suitable size for the later block. The latter patch-wise stylized block (PSB) then gradually recovers it to an image patch. Our PSB consists of convolution and up-sampling layers. The AdaIN module is followed after each up-sampling layer. With the joint effect of the two conditions – time and coordinate, the network outputs patches instead of the whole image. Such a practice greatly reduces the burden of the network. Experiments also show that our model is easier to fit the details than image-wise method, i.e., NeRV.

2.4. Time-Coordinate Stylization

The training process is to overfit a video, which requires moving the distribution of network activations towards the target video features. NeRV has found that the normalization layer widely used in convolution neural networks could reduce the fitting ability of ConvNets-based INR. This is because the common normalization operation is used to prevent overfitting, which will make the distribution of features unable to shift in the direction related to the target frame accurately. On the contrary, we find that aligning the mean and variance of features with the corresponding target frame can speed up the fitting process and obtain higher quality results.

We use Adaptive Instance Normalization (AdaIN) [9] to modulate the features of the later convolution layers according to the time-coordinate condition:

$$\text{AdaIN}(x_i, \sigma^*, \mu^*) = \sigma^* x_i - \mu^* x_i + \mu^* \sigma^* x_i, \quad (2)$$

where $\mu(x_i)$ and $\sigma(x_i)$ represent the $i$th feature map’s mean and variance, respectively. We use another MLP network to learn $\sigma^*$ and $\mu^*$ required by the later AdaIN:

$$\sigma^*, \mu^* = \text{MLP}_G(\Gamma(t, i)). \quad (3)$$

2.5. Objective Function

For PS-NeRV, we adopt a similar loss function as [7], which combines L1 and SSIM loss for network optimization. This function calculates the loss between the output patch and the ground-truth patch. In order to accelerate the speed of network fitting video, we add an additional total variation regularization $L_{tv}$ on image level to the reconstructed frame. When the video fitting is finished, there are no artifacts around patch boundaries. The final loss function is as follows:

$$L = \frac{1}{T \times N^2} \sum_{t=1}^{T} \sum_{i=1}^{N^2} (\alpha \| f_0(t, i) - v_p \|^2_1 + (1 - \alpha)(1 - \text{SSIM}(f_0(t, i), v_p))) + L_{tv}, \quad (4)$$

where $T$ is the frame number, $N^2$ is the number of patches in each frame, $f_0(t, i) \in \mathbb{R}^{H/N \times W/N \times 3}$ is the PS-NeRV prediction, $v_p \in \mathbb{R}^{H/N \times W/N \times 3}$ is the ground-truth, $\alpha$ is hyper-parameter to balance the weight for each loss component properly.

3. EXPERIMENTS

3.1. Datasets and Implementation Details

We use the “Big Buck Bunny” sequence and the UVG dataset as training data. We use Adam [15] optimizer to train the whole network. The learning rate is set to 5e-4. There are five PSB blocks in
functions and positional embedding, respectively. For NeRV [7], we also use its default settings. We obtain these models with different parameters by adjusting hidden dimension. We also change the convolution filter width of our framework to build PS-NeRV model of comparable sizes to the above models, named as PS-NeRV-S, PS-NeRV-M, and PS-NeRV-L. The PSNR is served as the metric to evaluate reconstructed video quality. Table 1 shows the comparison results. Compared with pixel-wise and image-wise methods, our patch-wise representation significantly improves the quality. Due to the addition of AdaIN and one more layer of MLP, the decoding speed will decrease a little, but remain on the same order as NeRV. We also verify the effectiveness of our framework on other videos in the UVG dataset, where each video is fitted separately. The comparison results with NeRV are shown in Table 2. Our patch-wise representation method has better performance than the image-wise method on various videos.

3.3. Video Compression

Once the video fitting is completed, the purpose of video compression can be achieved through the model compression. For the fairness of comparison, we use the same model compression method as NeRV to achieve the purpose of video compression. We then compare with state-of-the-arts methods on UVG dataset. Figure 3 shows the rate-distortion curves. We compare with H.264 [16], HEVC [17], STAT-SSF-SP [18], HLVC [19], Scale-space [20], and Wu et al. [21]. H.264 and HEVC are performed with medium preset mode. Our method is better than image-wise method in all cases. When the BPP
Table 2. Quantitative comparison (PSNR(dB)) results between NeRV and PS-NeRV. We sample 200 frames from each video in UVG dataset. These networks are trained on different sequence for 300 epochs. Our patch-wise representation method has better performance than the image-wise method on various videos.

| Resolutions      | \(N^2 = 1\) | \(N^2 = 4\) | \(N^2 = 9\) | \(N^2 = 16\) | \(N^2 = 25\) | \(N^2 = 36\) | \(N^2 = 49\) | \(N^2 = 64\) |
|------------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| 480p (854 × 480) | 39.63       | 39.87       | 40.42       | 40.98       | 40.51       | 40.17       | 39.88       | 39.62       |
| 720p (1280 × 720) | 36.45       | 37.06       | 37.55       | 37.94       | 37.71       | 37.43       | 37.32       | 36.93       |
| 1080p (1920 × 1080)| 34.65       | 35.18       | 35.79       | 36.14       | 35.94       | 35.83       | 35.42       | 35.08       |

Table 3. PSNR(dB) vs. number of N. The network parameters used in these experiments are all the size as PS-NeRV-L.

![Inpainting visualization](image)

**Fig. 5.** Inpainting visualization. The inpainted area is marked by a red rectangle. When a large area in the video is masked, our method can also get a good completion effect, while other learning-based method and NeRV can only get blurry results.

is small, our method even exceeds the traditional video compression technology and other learning-based video compression methods. Figure 4 shows visualization for decoding frames. At a similar setting, PS-NeRV reconstructs more accurate details.

3.4. Video Inpainting

Video inpainting is a task that aims at filling missing regions in video frames with plausible contents by fusing spatio-temporal information. In each frame of the incomplete video, regions at different positions are randomly masked. Since our framework constructs a mapping of spatiotemporal coordinates to image patches, this design enables our framework to perform a more accurate reconstruction of local areas. Therefore, our method can achieve an excellent video inpainting effect by simply fitting the incomplete video. During the training process, the patches containing missing regions are not sampled. When the training is completed, we can input the coordinates of these missing regions to the network to obtain the corresponding image patch. We compare our results with the state-of-the-art transformer-based method ViF [22]. As shown in Figure 5, in the case of large missing areas, ViF [22] has difficulty generating meaningful content and can only fill in very blurry results. In order to verify the superiority of introducing spatial information into our framework for the inpainting task, we also use the original NeRV to fit this masked video. The masked areas do not participate in the back propagation. The result of NeRV is also blurry, because only the time prior is used here. In contrast, by establishing an accurate mapping between frame patch and spatiotemporal coordinates, our method yields clear results that are almost indistinguishable from ground truth.

3.5. Ablation Studies

When using our framework to fit videos, how to select a suitable number of patches is a key problem. Therefore, we study the impact of the number of patches on the results. We design different upscale factors for different patch numbers, and change the filter width to get the same size model. The experiment is conducted on the UVG dataset and “Big Buck Bunny” sequence. We fit each video separately, while each video is adjusted to several common resolutions (480p, 720p and 1080p). The average metrics of these video results are shown in Table 3. When the number of patches increases, the PSNR will decrease. This is because when the patch number increases, our method will become closer to a pixel-wise representation, resulting in the decline of fitting efficiency. For these common resolutions, our patch-wise method always performs better than the image-wise \((N^2 = 1)\) and with AdaIN) method. The performance is the best when the number of patches is set to 16.

4. DISCUSSION

**Limitations and Future Work.** There are some limitations with the proposed PS-NeRV. First of all, our patch-wise representation will increase the demand for video memory in the training process. In addition, we also train the whole network from scratch. Therefore, in order to ensure the quality of video reconstruction, the encoding time still needs to be improved compared with the traditional video compression methods. In the future work, we may improve the efficiency of network training, but this is not the goal of this work.

**Acknowledgment.** This work was supported by the National Key R&D Program of China (2022YFB4701400/4701402), SZSTC Grant (JCYJ20190809172201639, WDZC20200820200655001), Shenzhen Key Laboratory (ZDSYS20210623092001004), and Beijing Key Lab of Networked Multimedia.
5. REFERENCES

[1] Julien N. P. Martel, David B. Lindell, Connor Z. Lin, Eric R. Chan, Marco Monteiro, and Gordon Wetzstein, “Acorn: adaptive coordinate networks for neural scene representation,” ACM Trans. Graph., 2021.

[2] 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,” in ECCV, 2020, Springer.

[3] Matthew Tancik, Ben Mildenhall, Terrance Wang, Divi Schmidt, Pratul P. Srinivasan, Jonathan T. Barron, and Ren Ng, “Learned initializations for optimizing coordinate-based neural representations,” in CVPR, 2021.

[4] 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,” in NeurIPS, 2020.

[5] Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein, “Implicit neural representations with periodic activation functions,” in NeurIPS, 2020.

[6] Jeong Joon Park, Peter Florence, Julian Straub, Richard A. Newcombe, and Steven Lovegrove, “Deepsdf: Learning continuous signed distance functions for shape representation,” in CVPR, 2019.

[7] Hao Chen, Bo He, Hanyu Wang, Yixuan Ren, Ser Nam Lim, and Abhinav Shirvastava, “Nerv: Neural representations for videos,” in NeurIPS, 2021.

[8] Sam T Roweis and Lawrence K Saul, “Nonlinear dimensionality reduction by locally linear embedding,” science, 2000.

[9] Xun Huang and Serge J. Belongie, “Arbitrary style transfer in real-time with adaptive instance normalization,” in ICCV, 2017.

[10] 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 International Conference on Machine Learning, PMLR, 2019, pp. 5301–5310.

[11] 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,” in European Conference on Computer Vision, Springer, 2020, pp. 405–421.

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin, “Attention is all you need,” arXiv preprint arXiv:1706.03762, 2017.

[13] 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,” NeurIPS, 2020.

[14] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein, “Implicit neural representations with periodic activation functions,” Advances in Neural Information Processing Systems, vol. 33, 2020.

[15] Diederik P. Kingma and Jimmy Ba, “Adam: A method for stochastic optimization,” in ICLR, 2015.

[16] 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, vol. 13, no. 7, pp. 560–576, 2003.

[17] 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, vol. 22, no. 12, pp. 1649–1668, 2012.

[18] Ruihan Yang, Yibo Yang, Joseph Marino, and Stephan Mandt, “Hierarchical autoregressive modeling for neural video compression,” arXiv preprint arXiv.2010.10258, 2020.

[19] Ren Yang, Fabian Mentzer, Luc Van Gool, and Radu Timofte, “Learning for video compression with hierarchical quality and recurrent enhancement,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 6628–6637.

[20] 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, 2020, pp. 8503–8512.

[21] Chao-Yuan Wu, Nayan Singhal, and Philipp Krahenbuhl, “Video compression through image interpolation,” in Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 416–431.

[22] Rui Liu, Hanming Deng, Yangyi Huang, Xiaoyu Shi, Lewei Lu, Wenzhu Sun, Xiaoang Wang, Jifeng Dai, and Hongsheng Li, “Fuseformer: Fusing fine-grained information in transformers for video inpainting,” in ICCV, 2021, pp. 14040–14049.