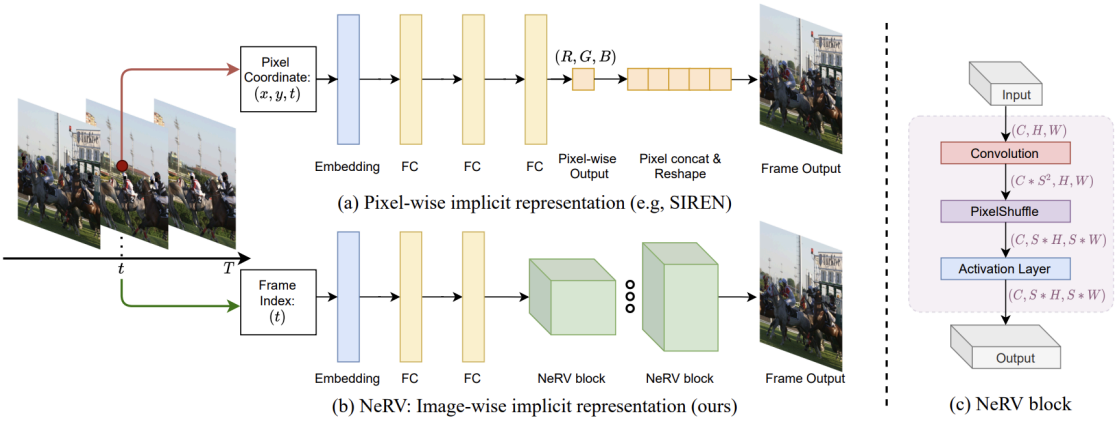
Reproducing NeRV: Neural Representations for Videos



April 2024 - CS4240 Deep Learning

Authors:

Jonathan Dingemanse (4703715)

Pieter-Jan van Dolderen (5088720)

Mahmoud Elaref (5252628)

Luca Goemans (4954645)

Introduction

A new neural representation for videos (NeRV) is proposed which encodes videos in neural networks [1]. Conventional representations interpret videos as a frame sequence, while this method represents videos as a neural network taking frame index as input. This makes it also possible to interpolate between frame indices and upscale to a higher resolution. Besides, it can be used for compression of videos and video denoising. This representation improves the encoding speed by 25x to 70x and the decoding speed by 38x to 132x compared to pixel-wise implicit video representations. In this paper we are going to reproduce and attempt to improve on some of the results of the original paper and try this representation on two videos taken from the Cholec80 dataset.

Background

In the paper and the code two metrics are used to track performance: Peak Signal to Noise Ratio (PSNR) and Multi-Scale Structural Similarity Index Measure (MSSSIM) [4]. The formula for PSNR is given by:

$$PSNR = 10 * log_{10}(MAX_I^2 / MSE)$$

with MAX being the maximum possible pixel value of the image and MSE the Mean Squared Error. The PSNR rate is commonly used to measure reconstruction quality, an important factor for video encoding in a neural network.

The formula for MS-SSIM is given by

$$MSSSIM = (2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2) / (\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)$$

with μ the mean of an image x/y, σ the variance of x/y, σ^2 the covariance and *c* two stabilization variables.

Experiments and results

Effect of learning rate on MS-SSIM and PSNR values

This part of this post investigates the effect of different learning rates on the MS-SSIM and PSNR values. The PSNR and MS-SSIM are the metrics used to evaluate the video. To do that, the prompt of the authors' Github is used [2]. The number of epochs is changed to 1200, since in the paper they stated that they use 1200 epochs to train the network on the "Big Buck Bunny" video [1]. Training a single model requires approximately 5 hours, and due to limited access to Kaggle's GPUs, careful selection of parameter values is essential. Our idea was to search around the value they used in the paper (learning rate of 0.0005). So, we started our search by taking the following values for the learning rate: 0.0001, 0.0005, 0.001, 0.005. After training the models, we already see that 0.0005 gives the highest performance. After that, we investigated the region around 0.0005 and trained the network with learning rates of 0.00025 and 0.00075. In the end to get a more overall picture, the network was trained with a learning rate of 0.003 and 0.01. The results are shown in Table 1.

Table 1: PSNR and MS-SSIM value for different learning rates

Learning rate	PSNR	MS-SSIM
0.0001	31.54	0.9481
0.00025	33.56	0.9697
0.0005	**34.01**	0.9735
0.00075	33.97	**0.9738**
0.001	33.94	**0.9738**
0.003	33.89	0.9737
0.005	31.94	0.9258
0.01	27.31	0.9258

Our reproduction experiments have verified that the optimal learning rate, as originally reported, is 0.0005. However, the region around a learning rate of 0.0005 gives similar results. To give a better overview, the results from the table are shown in figures as well (see figure 2 and 3). In figure 2, every point is specified first by the learning rate and then by the PSNR value. In figure 3, every point is specified first by the learning rate followed by its corresponding MS-SSIM value.

Figure 2: Effect of learning rate on the PSNR value

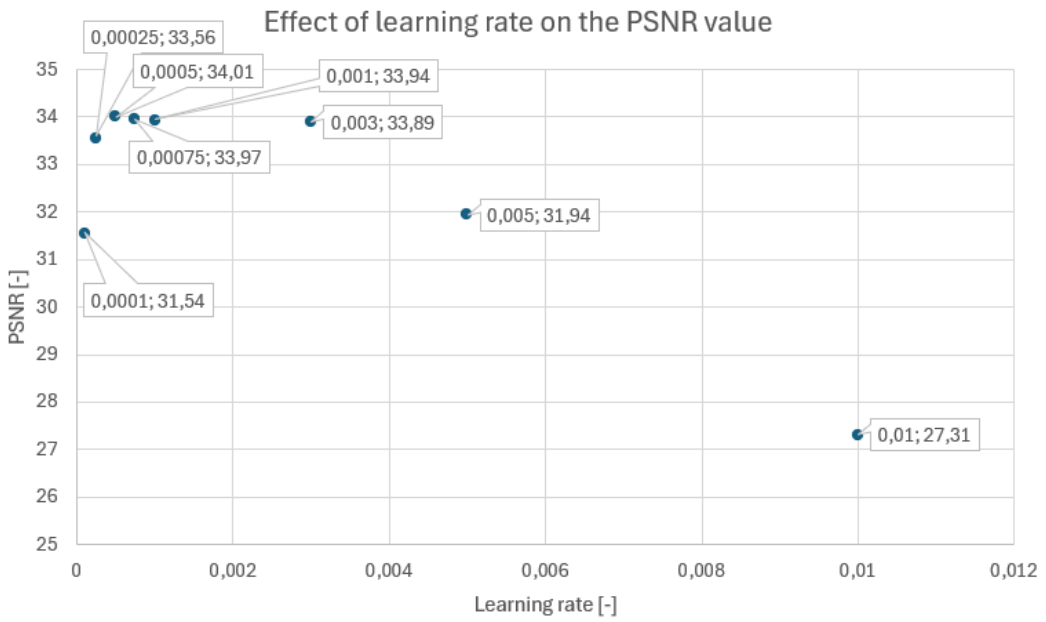
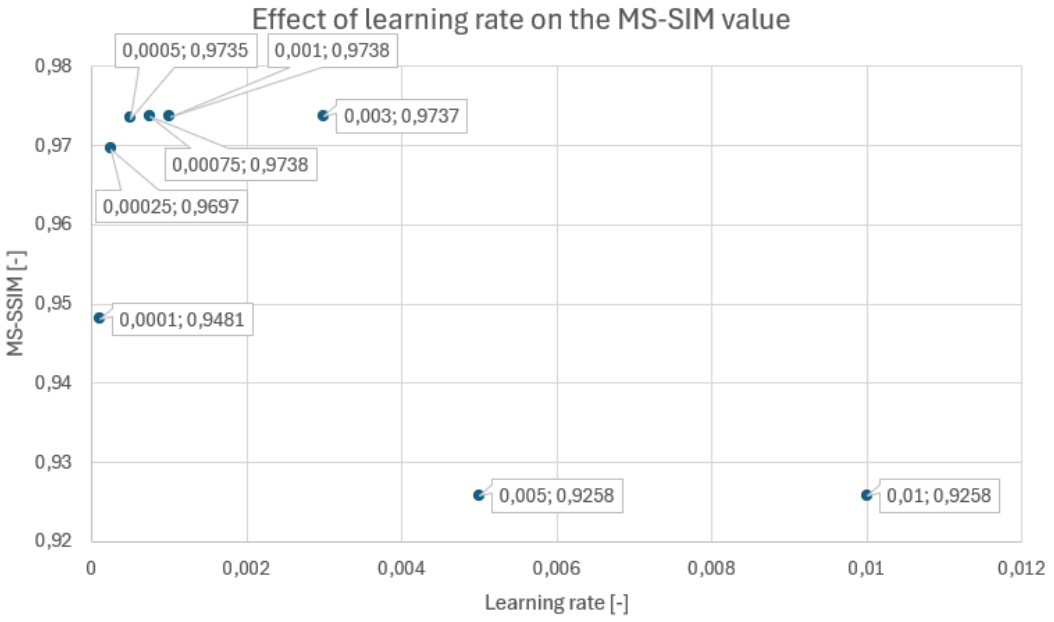


Figure 3: Effect of learning rate on the MS-SSIM value



Effect on the Loss Function

In our exploration of NeRV, we delve into the loss functions to attempt to enhance the quality of the generated videos. Loss functions are crucial metrics because they evaluate the difference between the original and reconstructed images. Our experimentation involved three primary loss functions: **L1**, **L2**, and **SSIM**. Same training conditions were used, 1200 epoch on Kaggle's P100 GPU. There was no significant difference in training time so it was omitted from the table. PSNR and MS-SSIM scores both play important but distinct roles when analyzing results. PSNR emphasizes pixel-level accuracy, measuring noise levels in the reconstructed videos compared to the original. Conversely, MS-SSIM more reflects human perception. Their importance varies depending on the application, however, a good balance between the two ensures both technical precision and perceptual similarity, optimizing the viewing experience.

Firstly, the **L1** loss, also known as the Mean Absolute Error (MAE), measures the absolute differences between corresponding pixel values in the original and generated images. Then, **L2** loss, referred to as the Mean Squared Error (MSE), calculates the average of the squared differences between corresponding pixel values. However, **L2** loss tends to prioritize large errors, making it less suitable for perceptual tasks where small but perceptually significant differences matter.

Lastly, **SSIM**, or Structural Similarity Index, represents a departure from the pixel-wise approach of **L1** and **L2** losses. Instead, **SSIM** is a perceptual metric designed to measure the similarity between two images by considering their luminance, contrast, and structure, capturing perceptual differences that the traditional loss functions may overlook.

Table 2: PSNR and MS-SSIM for varying loss function

Video	PSNR	MS-SSIM
100% L1	**34.31**	0.9684
30% L1 + 70% SSIM	32.92	0.9704
50% L1 + 50% SSIM	33.34	0.9716
70% L1 + 30% SSIM	34.02	**0.9736**
100% SSIM	28.08	0.9663
50% L2 + 50% SSIM	31.88	0.9686

Upon analyzing the results in Table 2 of our experimentation with various loss function combinations, several key observations emerge. Initially, when looking at the perception quality, seen through the MS-SSIM score, hybrid combinations of losses outperform any loss function on its own. Additionally, an experiment with L2 loss hybrid instead of L1 was enough to conclude that in this case, L1 is more effective when combined with SSIM Loss as opposed to L2. Secondly, the L1 loss on its own provided a great PSNR score, but fell behind with the MS-SSIM score.

Moreover, we observe that decreasing the weight of SSIM loss in the hybrid loss functions tends to improve both PSNR and MS-SSIM scores, suggesting that prioritizing L1 loss leads to better overall video quality.

Based on these observations, we learn that using a hybrid of L1 and SSIM while giving a higher weight to the L1 component, the hybrid loss function consisting of 70% L1 loss and 30% SSIM loss emerges as the most effective combination, achieving a balance between technical fidelity and perceptual quality through maximizing both PSNR and MS-SSIM scores.

Experiments on the Cholec80 Dataset

For this set of experiments the network has been trained using two videos from a different data source, the Cholec80 [3] dataset. This dataset contains 80 videos of

cholecystectomy surgeries. The interesting thing about the videos in this dataset is that the lighting conditions can vary quite a bit, leading to different reflections and exposure levels from frame to frame. The surgeons also exhibit complex motion patterns that could make it harder to correctly reproduce the video frames.

To carry out the experiment we selected two videos: video_50 and video_70. Video_50 was selected as a general baseline. Video_70 was selected because it had varying lighting conditions and an artificial material presence. We wanted to see if this would influence the network's capability to reproduce the frames. From the selected videos we extracted ~5 second clips and converted those to frames, leading to ~125 frames each (as the videos in the dataset were shot at 25 fps).

The network was then trained on both videos separately using Kaggle's P100 gpu. The results of this training can be found in Table 3.

Table 3: PSNR Rate and MS-SSIM for Cholec80 videos

Video	**PSNR**	**MS-SSIM**
Video_50	37.68	0.9753
Video_70	39.46	0.9763

If we look at individual frame results we see some interesting things. For video_50 we mainly notice a loss of detail. It can be seen that there is much less detail in the blood vessels for example. Another thing that stands out is that the reflections are not represented as well as in the original frame. You can see two examples of frames in Figure 4 and Figure 5.

Figure 4: Ground truth (left) and predicted (right) frame for video_50

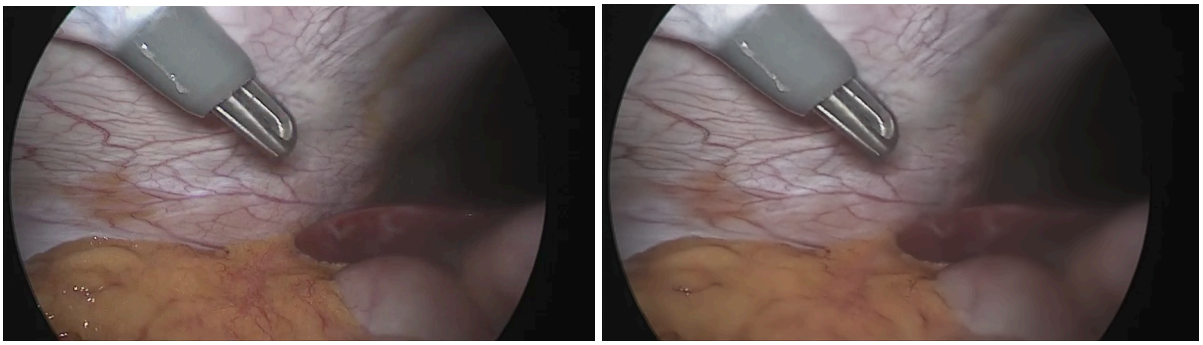
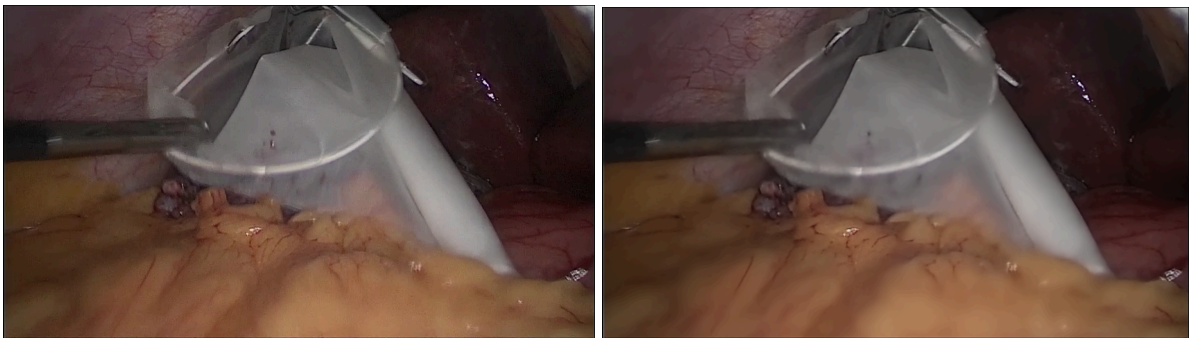


Figure 5: Ground truth (left) and predicted (right) frames for video_70.



For video_70 we expected the network to do worse, due to the varying lighting conditions, more complex movements and the presence of artificial material. Apart from the occasional loss of detail however we have not noticed a real difference. As can be seen in the images, the frames look quite identical, except for some sharpness in the droplet of blood on the white plastic. The PSNR rates for both videos are higher than the one achieved on the bunny dataset. A possible explanation for this might be that the subsequent individual frames for the Cholec80 dataset differ less from one another than those of the bunny video.

Upscaling steps and blocks

In another experiment, we tried out several changes to the layout of the upscaling strides and the number of blocks per upscaling stride, to see the effect on the performance of the model. Nine models with different layouts were trained and evaluated. As before, the small version of the model was used (NeRV-S), which was trained for 1200 epochs on the bunny video. The results can be found in table 4 and are visualized in figure 6.

The '5 2 2 2 2' layout as used in the paper is the most 'split out' layout. We tried to take larger and lesser upscaling strides. This resulted in slightly lower PSNR and MS-SSIM scores and slightly shorter computation times (on a P100 GPU). Overall it can be concluded that the model is quite robust to using different stride layouts.
It looks like the last step of the upscaling has the largest impact. The '10 4 2' and '5 4 4' layouts both have three upscaling steps, but the '10 4 2' layout performs much better than the '5 4 4' layout. In fact, the '10 4 2' layout performs even better than the '5 4 2 2' layout which has more steps and also ends with a stride of 2. This is

probably because the number of trainable parameters is kept approximately constant, so that the model becomes deeper when less upscaling steps are used.

For four upscaling layouts, we trained a model with a single NeRV block per upscaling step and a model with two NeRV blocks per upscaling step. This extra NeRV block is added as a block with upscaling factor of 1 after the upscaling block, so the '5 4 4' layout with 2 blocks per upscaling step can also be seen as a '5 1 4 1 4 1' layout. The extra block turned out to have a large impact on the performance. Both the PSNR and MS-SSIM scores are significantly higher for the model with two blocks compared to the model with one block. Next to that, the computation time for two blocks is about 30% of what it is for one block. Therefore, it seems very advantageous to use two blocks per upscaling step. More research could be done into using even more blocks per upscaling step.

Figure 6: PSNR, MS-SSIM and computation times for different model layouts

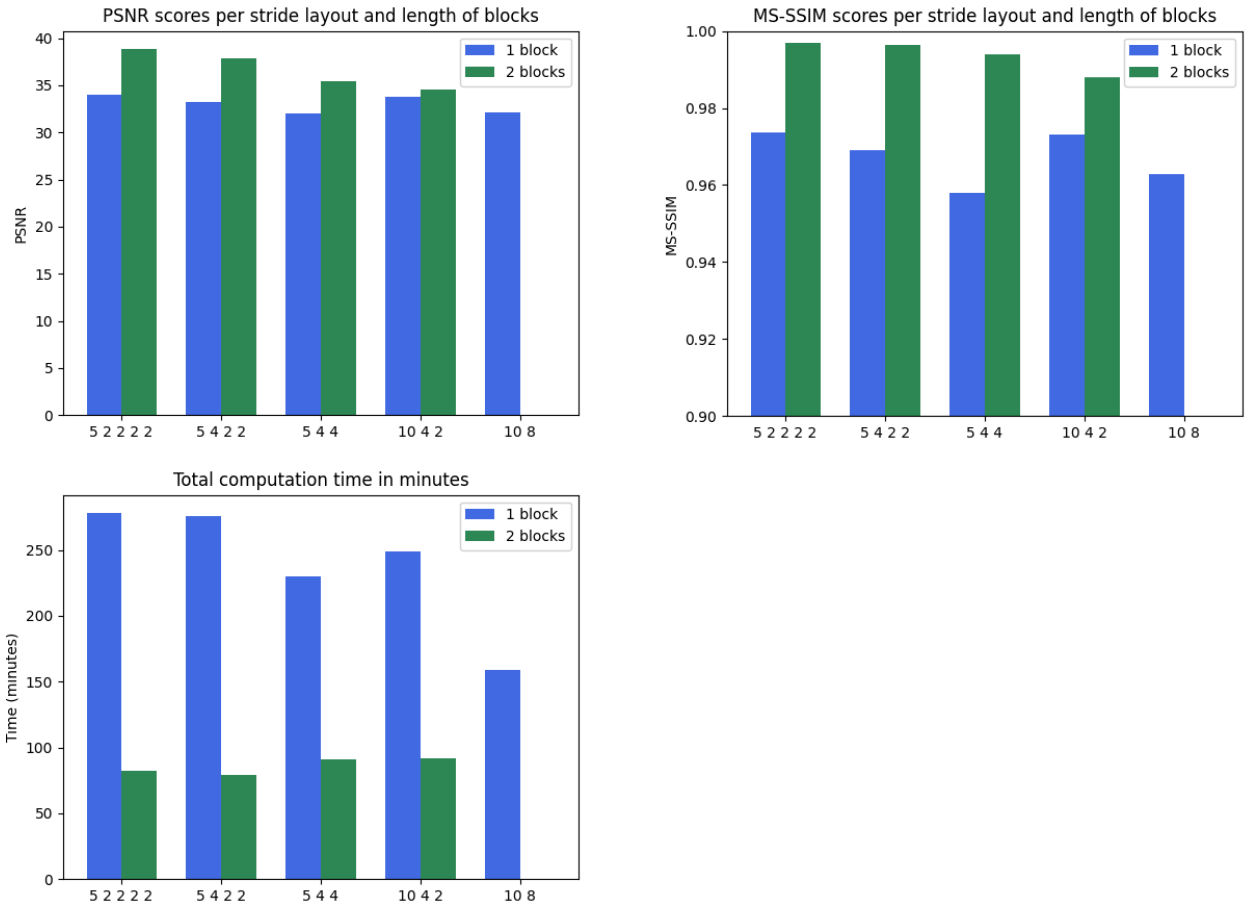


Table 4: PSNR, MS-SSIM and computation times for different model layouts

Strides	Blocks	PSNR	MS-SSIM	Comp. time on P100 (min)
5 2 2 2 2	**1**	34.01	0.9736	278
5 2 2 2 2	**2**	38.83	0.997	82
5 4 2 2	**1**	33.27	0.9691	276
5 4 2 2	**2**	37.9	0.9965	79
5 4 4	**1**	31.99	0.958	230
5 4 4	**2**	35.45	0.99400	91
10 4 2	**1**	33.74	0.9731	249
10 4 2	**2**	34.55	0.9879	92
10 8	**1**	32.10	0.9627	159

Contribution

Luca set up the project on Kaggle and reproduced the results of the original paper. Besides, Luca trained and evaluated the NeRV model on the Cholec80 dataset. Mahmoud was responsible for experimenting with different loss functions. Jonathan worked on the upscaling steps and tried different strides and numbers of blocks. Pieter-Jan investigated how changing the learning rate affects the performance.

References

[1] Hao Chen, Bo He, Hanyu Wang, Yixuan Ren, Ser-Nam Lim, & Abhinav Shrivastava (2021). NeRV: Neural Representations for Videos. In NeurIPS.

[2] Haochen-Rye. (z.d.). *GitHub - haochen-rye/NeRV: Official Pytorch implementation for video neural representation (NeRV)*. GitHub. https://github.com/haochen-rye/NeRV

[3] A.P. Twinanda, S. Shehata, D. Mutter, J. Marescaux, M. de Mathelin, N. Padoy, EndoNet: A Deep Architecture for Recognition Tasks on Laparoscopic Videos, IEEE Transactions on Medical Imaging (TMI), arXiv preprint, 2017

[4] Wang, Z., Simoncelli, E. P., & Bovik, A. C. (2003, November). Multiscale structural similarity for image quality assessment. In The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003 (Vol. 2, pp. 1398-1402). Ieee.