NeRV: Neural Representations for Videos

NeurIPS 2021 Poster

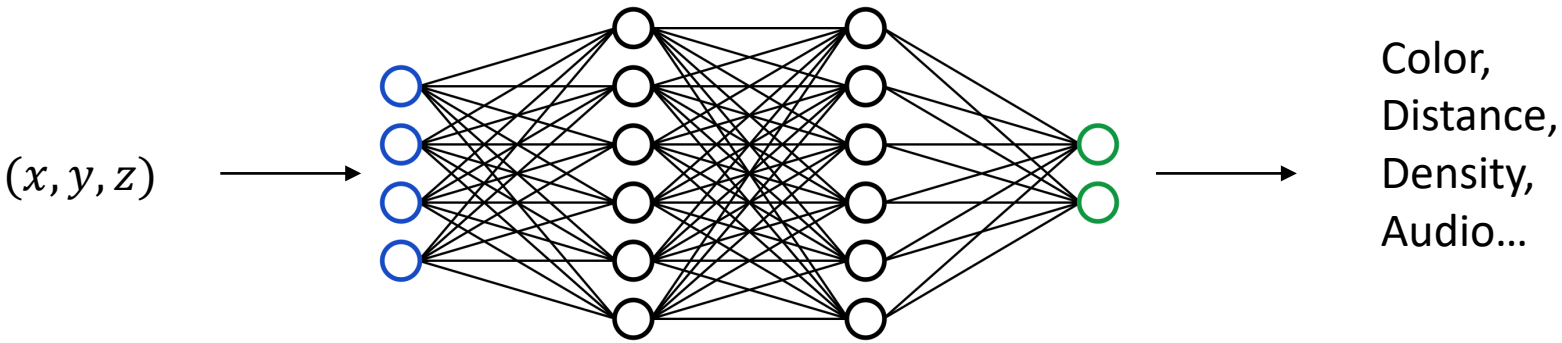
발표자: 석사과정 박여정

https://arxiv.org/pdf/2110.13903.pdf

https://github.com/haochen-rye/NeRV

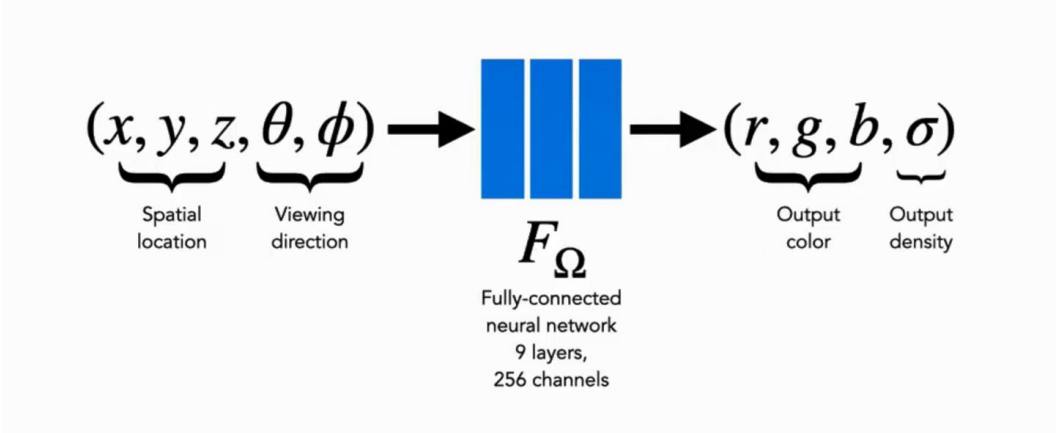
Introduction

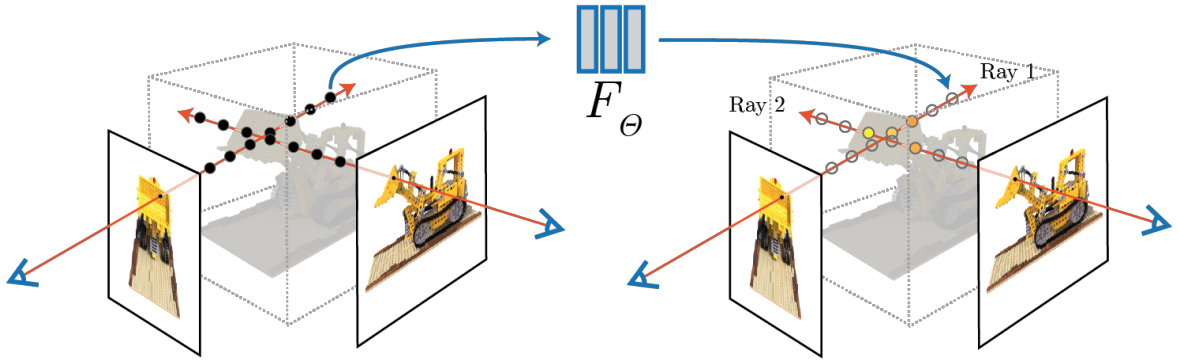
- Implicit Neural Representation
  - Parameterize a signal as a continuous function that maps the domain of the signal to whatever at that coordinate
  - 어떤 정보를 Neural net으로 나타내거나 저장하는 것



Introduction

- NeRF *Representing Scenes as Neural Radiance Fields for View Synthesis, ECCV 2020*
- represent a scene using a fully-connected (non-convolutional) deep network, whose input is a single continuous 5D coordinate (spatial location (x, y, z) and viewing direction (θ, ϕ))

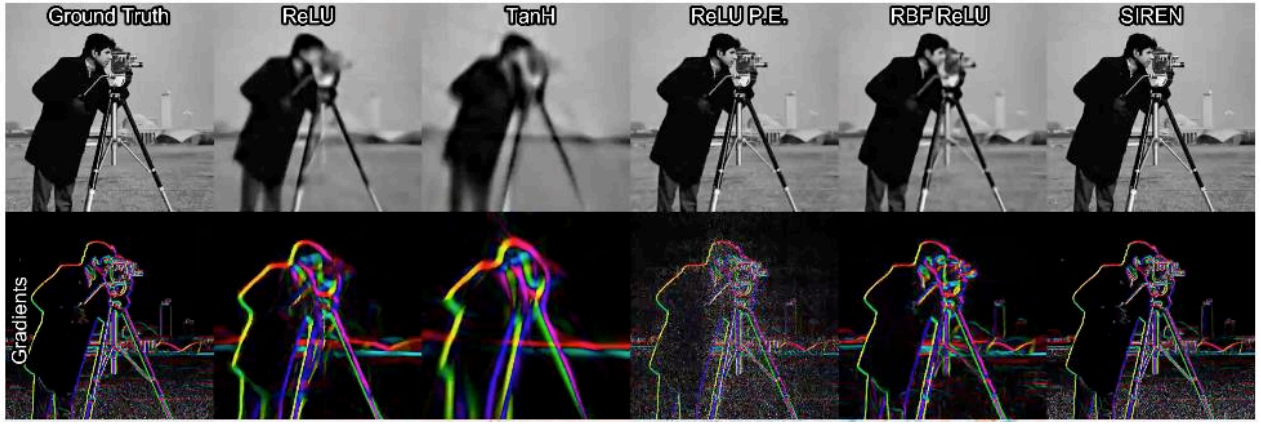


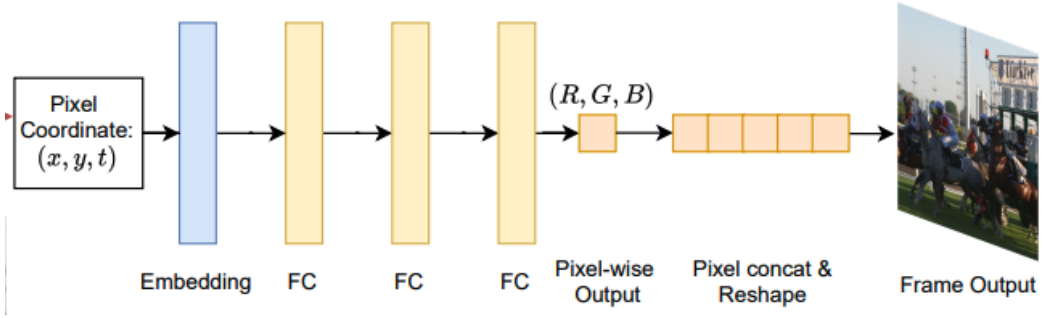


Introduction

https://medium.com/@dc3042/two-papers-from-stanford-on-implicit-neural-representations-22e3fc639b1

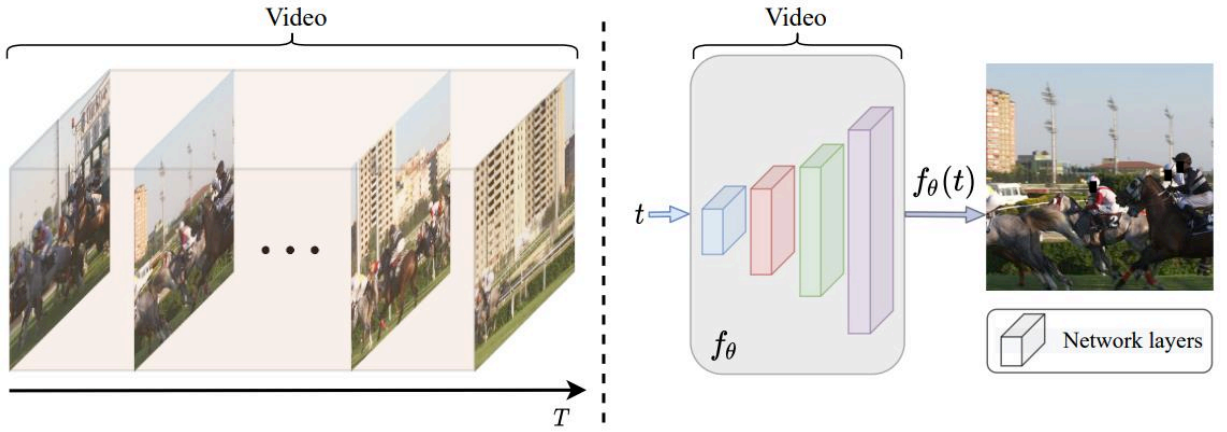
- SIREN *Implicit Neural Representations with Periodic Activation Functions, NIPS 2020*
- Use Sine activation function instead of ReLU





Introduction

- **NeRV**, a novel representation that represents videos as implicit functions and encodes them into neural networks.
- $V = \{v_t\}_{t=1}^{T}, where\ v_t = f_\theta(t)$
- Can reformulate video compression into model compression

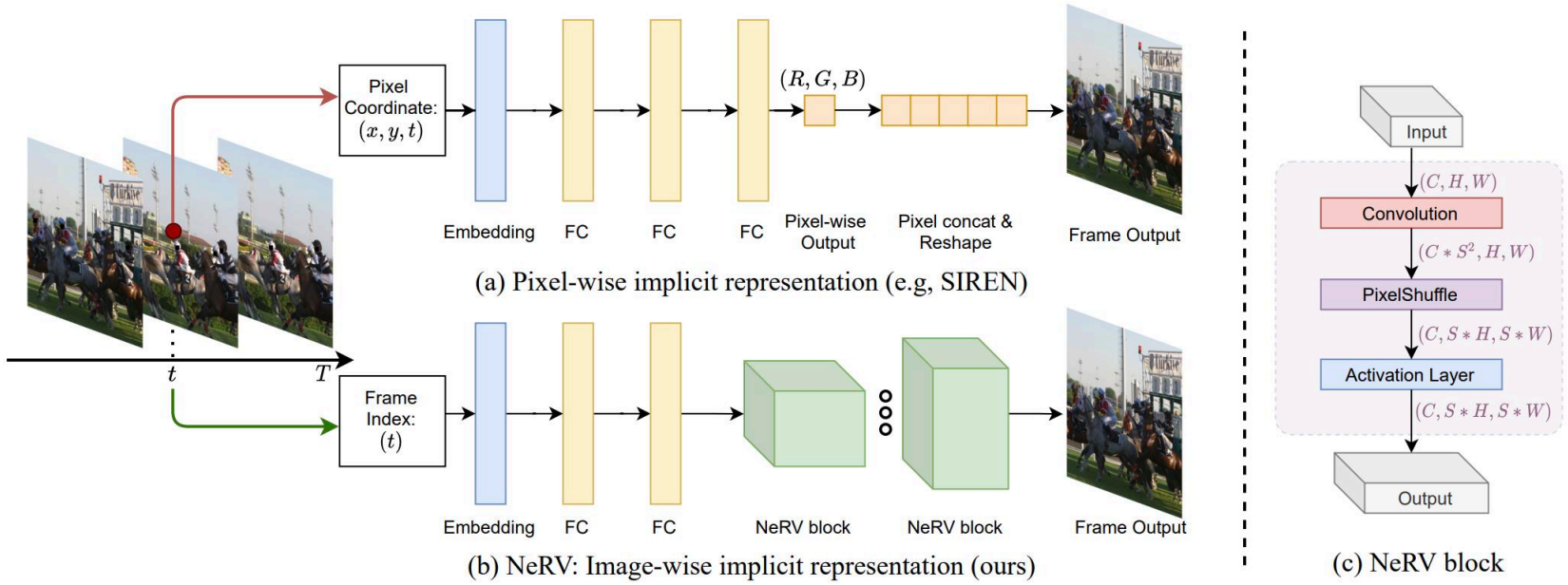


(a) Explicit representations for videos (e.g., HEVC) (b) Neural implicit representations for videos (e.g., NeRV)

	Explicit (frame-based)		Implicit (unified)	
	Hand-crafted (*e.g.*, HEVC [2])	Learning-based (*e.g.*, DVC [3])	Pixel-wise (*e.g.*, NeRF [4]	Image-wise (Ours)
Encoding speed	**Fast**	Medium	Very slow	Slow
Decoding speed	Medium	Slow	Very slow	**Fast**
Compression ratio	Medium	**High**	Low	Medium

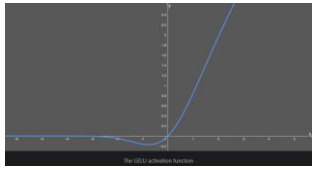
Architecture

- $V = \{v_t\}_{t=1}^{T}, where\ v_t = f_\theta(t)$
- Video encoding by fitting a NN to a given video
- Time embedding for learning high frequency variations

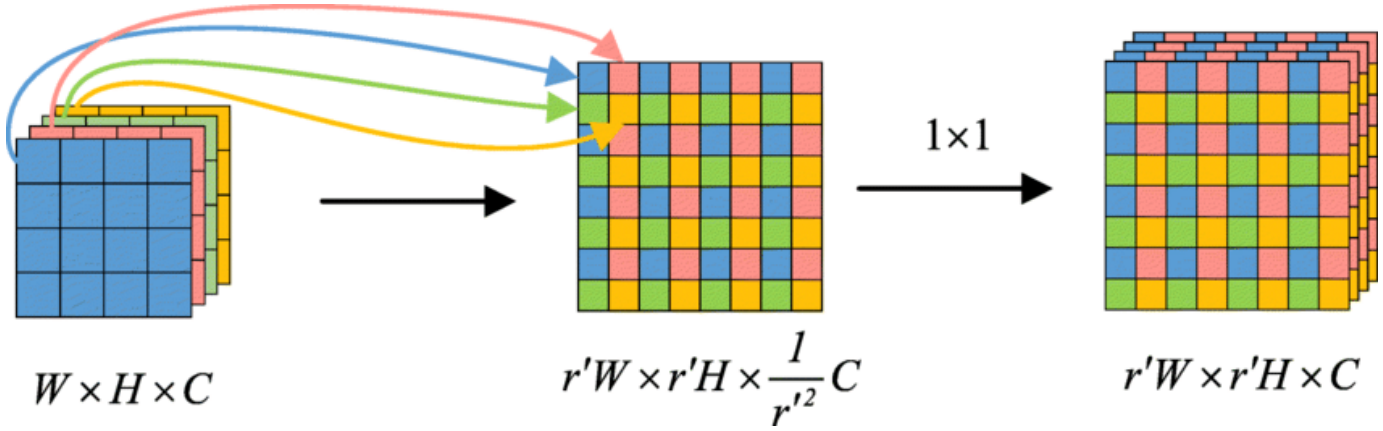


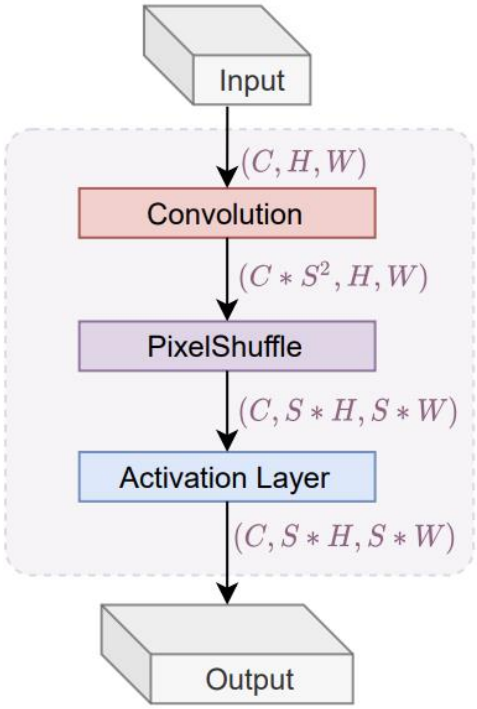
$$\Gamma(t) = \left(\sin\left(b^0\pi t\right), \cos\left(b^0\pi t\right), \ldots, \sin\left(b^{l-1}\pi t\right), \cos\left(b^{l-1}\pi t\right)\right)$$

Architecture



- Stack NeRV Blocks so that pixels at different locations can share conv kernels.
- PixelShuffle for upscaling method
- Up-scale (5,3,2,2,2) for 1080p videos
- Activation : GELU





(c) NeRV block

Architecture

- Loss objective
  - L1
  - SSIM loss: calculated on various windows of an image

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

$$L = \frac{1}{T}\sum_{t=1}^{T} \alpha \left\| f_\theta(t) - v_t \right\|_1 + (1-\alpha)(1 - \text{SSIM}(f_\theta(t), v_t))$$

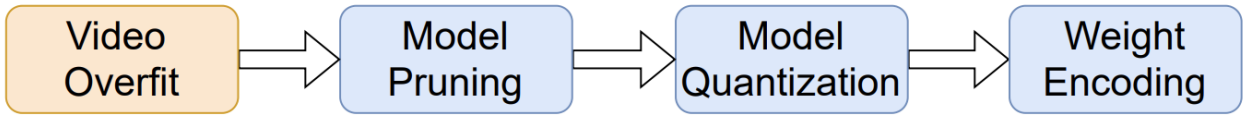
Iteration 32000 (Perceptual)

Iteration 32000 (Perceptual + SSIM)

Model compression

- Model Pruning
  - Global unstructured pruning

$$\theta_i = \begin{cases} \theta_i, & \text{if } \theta_i \geq \theta_q \\ 0, & \text{otherwise,} \end{cases}$$



- Model Quantization
  - Post-hoc

$$\mu_i = \text{round}\left(\frac{\mu_i - \mu_{\min}}{2^{\text{bit}}}\right) * \text{scale} + \mu_{\min}, \quad \text{scale} = \frac{\mu_{\max} - \mu_{\min}}{2^{\text{bit}}}$$

- Entropy Encoding
  - Huffman Coding

Experiments

Table 2: Compare with pixel-wise implicit representations. Training speed means time/epoch, while encoding time is the total training time.

Methods	Parameters	Training Speed ↑	Encoding Time ↓	PSNR ↑	Decoding FPS ↑
SIREN [5]	3.2M	1×	2.5×	31.39	1.4
NeRF [4]	3.2M	1×	2.5×	33.31	1.4
NeRV-S (ours)	3.2M	**25**×	**1**×	**34.21**	**54.5**
SIREN [5]	6.4M	1×	5×	31.37	0.8
NeRF [4]	6.4M	1×	5×	35.17	0.8
NeRV-M (ours)	6.3M	**50**×	**1**×	**38.14**	**53.8**
SIREN [5]	12.7M	1×	7×	25.06	0.4
NeRF [4]	12.7M	1×	7×	37.94	0.4
NeRV-L (ours)	12.5M	**70**×	**1**×	**41.29**	**52.9**

SIREN (Pixel-wise)

NeRV: (Image-wise)

Experiments

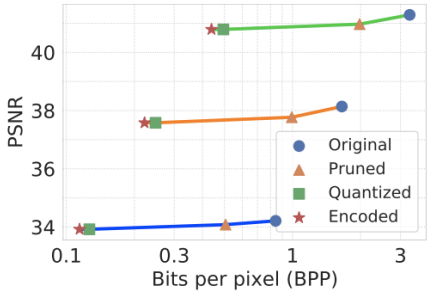


Figure 6: Compression **pipeline** to show how much each step contribute to compression ratio.

Table 3: PSNR *vs.* epochs. Since video encoding of NeRV is an overfit process, the reconstructed video quality keeps increasing with more training epochs. NeRV-S/M/L mean models with different sizes.

Epoch	NeRV-S	NeRV-M	NeRV-L
300	32.21	36.05	39.75
600	33.56	37.47	40.84
1.2k	34.21	38.14	41.29
1.8k	34.33	38.32	41.68
2.4k	**34.86**	**38.7**	**41.99**

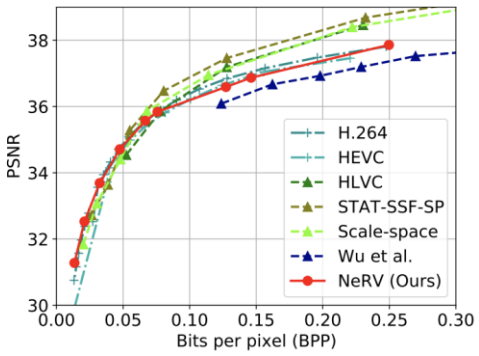


Figure 7: PSNR *vs.* BPP on UVG dataset.

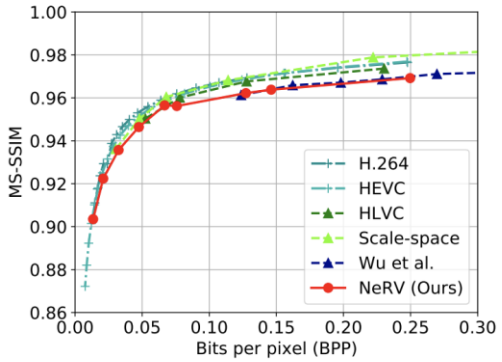


Figure 8: MS-SSIM *vs.* BPP on UVG dataset.

Table 4: **Decoding speed** with BPP 0.2 for 1080p videos

Methods	FPS ↑
Habibian et al. [14]	$10^{-3.7}$
Wu et al. [24]	10^{-3}
Rippel et al. [57]	1
DVC [3]	1.8
Liu et al [58]	3
H.264 [8]	9.2
NeRV (FP32)	5.6
NeRV (FP16)	**12.5**

(a) Ground Truth Frame (b) NeRV Decoded Frame (0.077 BPP) (c) HEVC Decoded Frame (0.075 BPP)

Figure 9: Video compression visualization. At similar BPP, NeRV reconstructs videos with better details.

Experiments

Table 5: PSNR results for **video denoising**. "baseline" refers to the noisy frames before any denoising

noise	white ↑	black ↑	salt & pepper ↑	random ↑	Average ↑
Baseline	27.85	28.29	27.95	30.95	28.74
Gaussian	30.27	30.14	30.23	30.99	30.41
Uniform	29.11	29.06	29.10	29.63	29.22
Median	**33.89**	33.84	33.87	33.89	33.87
Minimum	20.55	16.60	18.09	18.20	18.36
Maximum	16.16	20.26	17.69	17.83	17.99
NeRV	33.31	**34.20**	**34.17**	**34.80**	**34.12**

(a) Ground Truth Frame

(b) Noisy Frame

(c) DIP Output (30.13 PSNR)-Iteration 1800

(d) NeRV Output (39.27 PSNR)

(e) DIP Output (32.30 PSNR)-Iteration 3600

Discussion

- To achieve the comparable PSNR/MS-SSIM, the training time is longer than the encoding time of traditional video compression
- Architecture is not optimal yet
- NeRV is a novel way to represent videos as a function of time, which might be used in many video-related tasks.