INVR: Implicit Neural Video Representation

Zakir Singh
zs391@cam.ac.uk

Alexandru-Andrei Iacob
aai30@cam.ac.uk

Abstract

A novel approach for implicit neural image representation leverages high-dimensional positional encodings and MLPs with periodic activation functions to encode images by mapping pixel coordinates to their RGB values. This simple design allows for highly-parallel decoding and can benefit from model compression. We adapt this technique for the video domain by incorporating the time dimension into the positional encoding. To fully leverage the dynamism of the time dimension, we propose two techniques. The first is to decouple the frequency with which time is encoded from the constant spatial dimensions, allowing us to tune the quality of the representation based on the number of frames. The second is to temporally decompose the video, encoding each chunk into an individual MLP, allowing our approach to accomodate any video length with consistent quality. Furthermore, we apply a series of model compression techniques to reduce the memory footprint and evaluate the usefulness of implicit neural representations as a means of video compression.

1 Introduction

Research in image and video processing has historically been focused on classical algorithmic approaches [15]. However, since the emergence of Deep Learning (DL) [8] there has been growing interest in neural methods [7, 6]. While the initial attempts to encode images and video using neural networks relied on proven architectures—such as CNN's [1, 2], RNN's [22, 23], and GAN's [14]—designs based on high-dimensional "positional encodings" and SIREN (Sinusoidal Representations Network) MLPs have recently been proposed [16, 21, 5, 17]. Research has focused on the ability of such "positional encodings" and SIRENs to capture high-frequency functions in situations where standard MLPs have historically been ineffective, such as in the task of learning images [21].

Our approach, INVR, focuses on the natural application of SIRENs[16] to implicit neural video representation, using frames as a "time-axis". We show that not only do the benefits of SIRENs extend to video but also that the similarities between successive frames make them especially well-suited to the task. We combine sinusoidal positional encodings—as presented in [21, 11, 25]—with SIREN models and show that their benefits are additive in the case of video. We further develop the encoding by allowing for varying frequencies between the spatial and temporal dimensions, enabling us to achieve a higher level of precision compared to encoding all dimensions equally.

Having constructed an MLP capable of learning video, we then evaluate its use as a means of compression. By sending the network weights over the wire, the client can reconstruct the video by running inference using the received pretrained weights. We aim to explore whether it is possible for the total bandwidth used in this scenario to be smaller than sending a compressed version of the original video. While prior work, such as COIN [5], has attempted this for image compression, we are the first to propose it for video. While COIN was not competitive with the state-of-the-art, further work [17] has suggested that model compression techniques, such as low precision parameter storage and entropy coding, may be used to further reduce the size of network parameters. In this vein, we utilize the DeepCABAC [24] entropy-coding system to compress the trained network weights, thereby minimising network storage with minimal impact on accuracy.

While not yet competitive with modern compression methods such as H.264, we demonstrate competitiveness with the older MPEG compression standard, and enumerate opportunities for expansion which may yield state-of-the-art results.

2 Background

The use of positional encodings for image processing can be traced back to the concurrently published SIREN model [16] and the high-dimensional positional input encodings used by NeRF for Novel View Synthesis [11]. The intuition behind positional encoding is that by expanding an input coordinate (x, y) into a higher number of dimensions, points become easier to separate. This is similar to the 'kernel trick' in Machine Learning literature. NeRF applies the following sinusoidal encoding on each coordinate:

$$\phi(p) = (sin(2^0\pi p), cos(2^0\pi p), ..., sin(2^{l-1}\pi p), cos(2^{l-1}\pi p)) \tag{1}$$

before feeding them into an MLP with ReLU activations. SIREN instead uses sinusoidal activation functions between MLP layers. This increase in non-linearity allows SIREN to capture more complex patterns, making it potentially better suited to video processing. It is important to note that SIREN MLPs and the input positional encodings used by NeRF are not contradictory, and have been applied concurrently with success [17]. We replicate this combined approach in this paper.

Image Compression: The authors of COIN [5] emphasise the remarkable simplicity and flexibility of encoding images in neural networks. Firstly, parallel inference allows for all pixels in an image to be computed simultaneously. Secondly, operating pixel-by-pixel allows for progressive image decoding. Thirdly, because it operates over a continuous representation space, it is possible to upscale or downscale images by simply changing the size of the canvas. Moreover, the simple MLP architecture allows COIN to obtain a reasonable levels of compression while only employing standard half-precision post-training quantisation.

The authors of [17] attempt a more comprehensive approach than COIN, addressing the entire training pipeline. By constructing a training process comprised of meta-learning for model initialisation, use of quantisation-aware training, L1-regularisation, and entropy coding—among others—they obtain competitive results against standard codecs with varying bits-per-pixel. We show that their techniques are entirely applicable to video compression; however, the computational requirements of meta-learning are prohibitive, and we have not attempted to replicate that aspect in our work.

Model Compression Model Compression [3, 4] refers to methods which decrease the size of a neural network while keeping model performance within bounds set by the user. We make use of the entropy coding and quantisation techniques proposed by DeepCABAC [24] for compressing the weights of our MLPs.

Entropy coding is a process that has been applied with success by traditional video compression systems [9, 20, 19] and operates by assigning a unique code to each symbol S whose length is proportional to $-log_2(P(S))$, such that frequent symbols receive shorter codes and the signal compression ratio is maximised relative to the entropy of the source. *Arithmetic coding* refers to entropy coding techniques which pack multiple symbols into a single integer. The widely-used video compression standard H.264 utilises Context-Based Adaptive Binary Arithmetic Coding (CABAC) [9], an implementation of these methods.

DeepCABAC adapts CABAC for the domain of model compression by exploiting the underlying matrix structure of neural networks. A key limitation of DeepCABAC is that if a model weight grows beyond a set bound, it falls back to storing it in a less efficient encoding. We therefore utilize regularization techniques to ensure weights never exceed this bound, enabling DeepCABAC's arithmetic encoding to run much more efficiently.

Prior to this encoding process, DeepCABAC utilizes quantisation to reduce the entropy of weight values. To avoid degrading video quality, DeepCABAC dynamically chooses quantisation points which minimise distortion of the cost function to be within user-defined bounds, leading to greater compression with minimal impact on network accuracy. Although compressing weights in this manner is still lossy, neural networks are known to be resilient to loss of precision in their weights due to overparameterization [18]. Two hyperparameters concern our compression, the step-size S

which controls the frequency of quantisation points, and λ, which modulates the amount of distortion at a given size.

3 Methods

3.1 Video Encoding

In order to construct the neural network representation, we draw upon the precedent set by [17] of combining high-dimensional positional encodings with SIREN MLPs to obtain superior performance over either method alone. We mention that these aspects of the system have gone through multiple iterations, as such, we will also present the evolution and advantages brought on by every addition.

3.1.1 Positional Input Encoding

Our positional encoding uses a similar structure to eq.1, taking inspiration from the authors of [17] by prepending the original coordinate to the encoding for improved performance. To accommodate the inherent variability of video length, we decouple the space and time dimensions of the encoding from one another. We argue that this addition opens up several optimisation avenues and confers greater flexibility for learning videos. Eq.2 shows how our positional encoding is constructed from applying the encoding functions to components of each individual coordinate $(x, y, z) \in H \times W \times T$ together with the intended length of the encoding (l_x, l_y, l_t).

$$\begin{aligned} &\gamma((x, l_x), (y, l_y), (t, l_t)) = ((x, y, z, \phi(x, l_x), \phi(y, l_y), \phi(t, l_t)) \\ &\phi(p, l) = (sin(\sigma^0 \pi p), cos(\sigma^0 \pi p), ..., sin(\sigma^{l-1} \pi p), cos(\sigma^{l-1} \pi p)) \end{aligned} \tag{2}$$

The work of [25] proves that the ability of a sinusoidal embedding to fit a signal depends on the dimension of the encoding and on the scale of the frequencies utilised. In terms of the scale, we keep a constant $\sigma = 1.4$ throughout, given both the recommendation on [17] and the unsatisfactory results found by attempting to raise its value.

When constructing a neural image representation, the image axes may either be equal or close in scale, making a symmetrical encoding dimension across the x and y directions sufficient. However, the number of frames which are fit by one SIREN MLP may be an order of magnitude smaller than either spatial dimension. By lowering the length of encoding along the time axis, we can increase the encoding size on the spatial dimensions without having to raise the parameter count of the network. It is important to note that this does not give us freedom to keep the time dimension to a bare minimum while continuously increasing the others; since the first layer of the MLP is connected to every input, the proportion $l_t/(l_x+l_y)$ serves a crucial role as an implicit weighing of the importance of time. As a heuristic, we have found that using between $[1/3$ and $1/2]$ of the spatial dimension length for time to work well.

We have also attempted to use a Gaussian encoding as proposed by [25]:

$$\phi(p, l) = (exp(-\frac{||x-0||^2}{2\sigma^2}), ...exp(-\frac{||x-(l-1)||^2}{2\sigma^2})) \tag{3}$$

However, we have found the results to be of significantly lower quality compared to positional encoding (sec.4). We speculate that such an encoding might work better with a different network architecture and activation function combination or in domains where generalisation is more important than in image representation.

3.1.2 Model Design

ReLU MLP: The initial network we constructed followed the design of NeRF [11], in which a positional encoding of the input is fed into a ReLU-based MLP. This structure proved incapable of obtaining an acceptable image, even for low numbers of frames (sec.4). This further confirms the speculations of [16] and [21] on the limits of classical MLP structures in modelling very-high-frequency functions, even with the help of an initial encoding via a periodic function. Specifically, all derivatives of a ReLU MLP are zero except for the first, making the network ill-suited towards modelling signals which depend on information found in higher-order derivatives. Given sufficient depth, it is quite likely that the benefits of the input encoding simply cannot propagate through the layers.

SIREN MLP: Following this attempt we changed the model to the original SIREN [16] using the initialisation methodology proposed by the authors as well as their recommended $w_0 = 30$ weight for the sin activations. In terms of model size, we follow the lead of [17] who found that for image representation tasks the benefits of increased network depth plateau at 3–5 hidden layers when using 64–128 neurons per layer. While the amount of data necessary to represent a video is significantly higher than image, the inherent sequentiality and similarity between the frames means that we can use a network on the higher end of the range $h \times w = 4 \times 128$ to fit a significant number of frames (depending on the resolution) without a large loss in average quality.

Video Chunking: At the highest level, a video is first split into chunks of N frames in order to guarantee consistency in terms of both quality and model size. This allows us to tune the network size and positional encoding dimensions for the first chunk of the video and then handle any arbitrary number of frames which may later arrive, making the system suitable for scenarios where either the video size is initially unknown or too large to fit with one network. We refer to this as constant-size chunking. It is imperfect given that chunks are not identically difficult to learn, however, we find that it produces relatively consistent results in practice (sec.4).

The maximum number of frames that can be learned by one MLP is heavily dependent on the complexity and amount of movement in the video; one MLP can learn 50 frames of a relatively static video, whereas the same MLP will struggle with 20 frames of a video with constant motion. We therefore attempted to devise a progressive chunking algorithm which would expand and contract the size of chunks depending on PSNR changes, however, this would have massively increased training time while heavily limiting the potential for parallel training of chunks, and was therefore deemed impractical.

Model Training: Our training pipeline for one training step first normalises (eq.4) the coordinates to be in the interval $[-1, 1]$, similarly to COIN [5], and memoizes the results for the current chunk. We process one frame at a time by running inference for every coordinate point within the canvas and predicting an RGB value to form a predicted frame. The loss function then applied is a weighted combination of the Mean Squared Error between the prediction and ground truth and the l_1-norm of the model (eq.5).

$$[x, y, z] = 2 * ([x, y, z] \cdot [1/H, 1/W, 1/T] - [0.5, 0.5, 0.5]) \tag{4}$$

$$L(p, f, M) = L_{MSE}(p, f) + \lambda \times ||M|| \tag{5}$$

The inclusion of the l_1-norm was initially implemented by [17] in order to penalise large absolute weight values and to facilitate easier compression later in the pipeline. We weigh the l_1-norm by $\lambda = 10^{-6}$ rather than the $\lambda = 10^{-5}$ they recommend as we find the negative impact on PSNR to be too great for video. Fitting an MLP to a given chunk is then simply a process of training on frames in random order for a given number of iterations or until a pre-set upper limit on PSNR has been reached.

While COIN [5] trains in full precision before applying post-training quantisation, we have found that applying mixed-precision [10] training improved speed without a noticeable negative impact on PSNR. This allows us to execute more iterations and ultimately improve representation quality. For quantisation we use the publicaly available DeepCABAC implementation with λ set to the default value $\lambda = 0$. We manipulated the size of the final compression by varying the step-size S until we settled upon $S = 3^{-\frac{13}{2}}$ offering a good compression ratio without greatly reducing quality, which is in keeping with the authors [24] finding that step-size is the primary determiner of compression performance.

3.2 Video Decoding

Video decoding is simply the process of applying inference to every pixel from every frame sequentially. Although decoding speed has not been a challenge, low-latency future applications have multiple avenues of parallelism at their disposal given the per-pixel design.

3.3 Experimental Design

Two 128x128 pixel videos have been used for testing, $firework$ and $lego$. Both videos contain 500 frames and are processed in chunks of 15 frames by the neural models, which are each trained for

25,000 iterations. Unless mentioned otherwise, all sinusoidal positional encodings use the following embedding dimensions $[l_x, l_y, l_t] = [64, 64, 32]$. While this may seem large given the video and network dimensions, the positional encodings themselves are a means of information extraction whose effectiveness scales with the number of dimensions. In fact [21, fig.8] shows that a higher quality encoding can improve both model convergence and performance. Consequently, the encoding should be seen as closer to a deterministic layer with a disproportionate impact for a given parameter count.

Neural Video Reconstruction: In terms of image reconstruction quality, we are concerned with comparing our full uncompressed INVR-U system against the original picture, as well as reconstructions made using the same model with an even encoding size $[l_x, l_y, l_t] = [52, 52, 52]$ and similar total encoding size (INVR-Even), no encoding (INVR-None), a Gaussian encoding [25] of size $[l_x, l_y, l_t] = [64, 64, 32]$ and $\sigma = 0.003$ (INVR-Gaussian) and a ReLU model with the same sinusoidal encoding as INVR-U $[l_x, l_y, l_t] = [64, 64, 32]$ (ReLU-SiN). Given the space constraint of this work, we reserve direct visual comparisons for very different models and baseline compression comparisons. All others will be realised by comparing PSNR levels at a given iteration or by the probability density function of PSNR values in the final result.

Compression: This set of experiments is interested in establishing a clear comparison between the DeepCABAC-compressed INVR-C against MPEG and H.264 as the baseline algorithmic approaches to compression. Furthemore, we investigate how well DeepCABAC can maintain model performance in the face of a large compression ratio by comparing the image quality of INVR-U and INVR-C. Given that video compression is not a task of uniform complexity, different chunks may suffer disproportionately from having to handle the higher entropy cause by movement or rapid changes in video content. As such, we compare the PSNR of the different compression methods over the entire 500-frame video to better understand their response to difficult sections. Additionally, we plot the probability density function of PSNR for the entire video.

System Details: An AWS EC2 instance equipped with an Nvidia T4 GPU with 16GB of VRAM has been utilized for all model training and testing. All systems have been written in PyTorch [13], with full experiment code available on GitHub.[1]

Image Quality Metrics: To evaluate image quality, the Peak Signal-to-Noise ratios of the reconstructed frames are averaged for either the entire video, a given chunk, or simply reported for a single frame.

Compression Metrics: The final compressed size of the video is measured as the size of the network weights of INVR. This is compared to MPEG and H.264 compressed versions at similar bitrates (killobytes per second).

4 Evaluation

4.1 Neural Video Reconstruction

Fur qualitative analysis of a single frame from $fireworks$, visible in Fig. 1, showcases the ability of INVR-U and INVR-C to accurately reconstruct an image in a manner comparable to both the original and classical encodings. To interpret this result, it is important to note is how positional encodings behave when transforming from parameter space to representation space. As previously hinted at in Section 2 and by [21, 25], exceptionally high encoding dimensions, such as those of INVR, lead to a large distancing between coordinates in representation space and thus a "grainy" look in the reconstructed image. This is clearly observable in the fact that INVR-U retains greater detail than H.264 yet captures less of the smooth areas of the original. On the opposite end, despite a PSNR ≥ 20 and similar encoding dimension, INVR-Gaussian utterly deforms the image in an exceptionally smooth fashion. Given that we have used the exact same frequency scale as [25, fig.10], it is likely that the exponential decay (eq.3) of the Gaussian encoding results in projections that get squished too close together as they pass through the sinusoidal activations—unlike the ReLU activations from [25].

[1] `https://github.com/ZakSingh/neural-positional-compression`

Given that SIRENs [16] are tuned to align the periodic sinusoidal activations, it is quite possible that an entirely different activation function would be needed for a Gaussian encoding to provide good results.

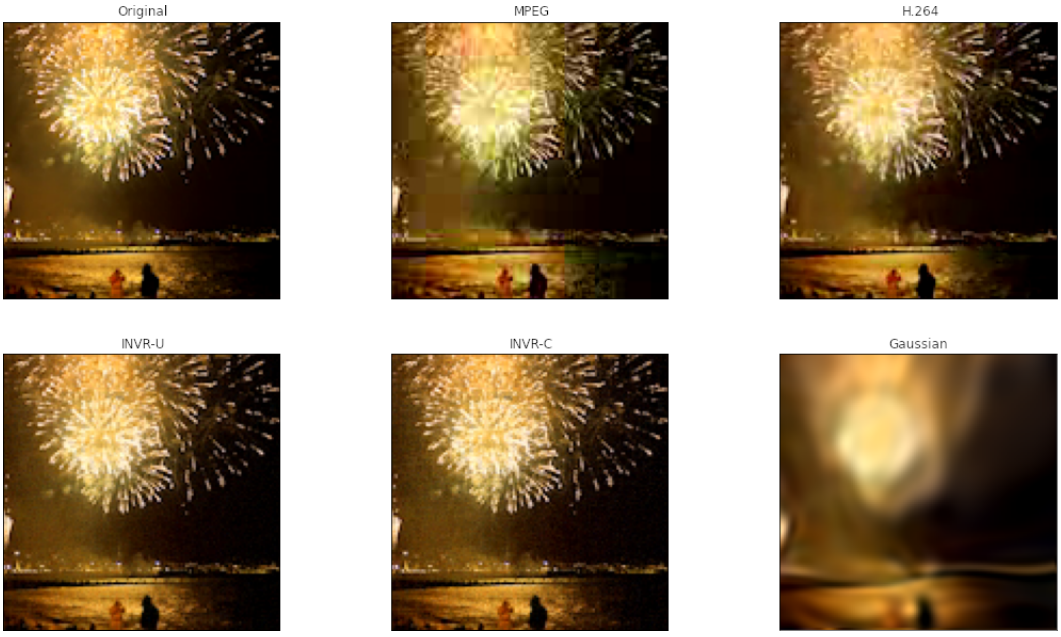


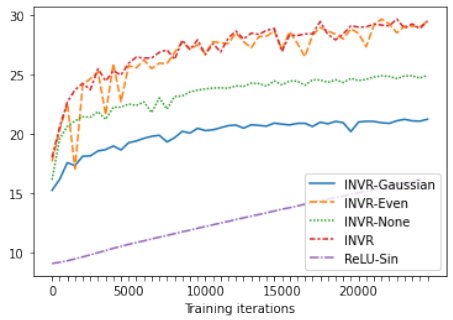
Figure 1: Qualitative analysis of the same frame with different encoding methods. INVR-U, INVR-C and H.264 resemble the original with H.264 over-emphasising smoothness for the water while INVR-U and INVR-C induce a grain-filter like effect to the smoke which is not present in the original. INVR-C greatly outperforms MPEG compression. The most "alien" result out of all the neural representations is produced by INVR-Gaussian.

Our quantitative comparison between several Neural Video Representations (see Fig. 2) validates the superiority of combining SIREN with positional encodings [17] in INVR over using INVR-None (no positional encoding) or ReLU-Sin (ReLU with positional encoding) in isolation. ReLU-Sin cannot extract sufficient high-frequency information from the initial encoding, resulting in the worst absolute performance across all models (Fig. 2a, c). INVR-None on the other hand begins with a significantly lower Average PSNR than either INVR or INVR-Even (Fig. 2a)—as it must rely on a purely learnt encoding in its first hidden layer—and never reaches the same levels of overall quality (Fig. 2a, c).

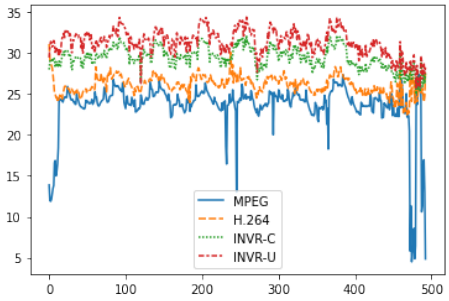
A novel finding which emerges from INVR is the benefit provided by decoupling the spatial and temporal dimensions of the input encoding. INVR has a significantly smoother convergence process (Fig. 2a) compared to INRV-Even (which has an equal number of frequencies per dimension), and its average PSNR never drops close to the baseline set by INVR-None, unlike INVR-Even. Furthermore, despite reaching similar average PSNRs for the chunk by the end of training, INVR has a larger proportion of high-PSNR frames (Fig. 2c) indicating better peak quality and long-term potential. We speculate that the cause of low performance in INVR-Even is that since the number of encoding dimensions on the time axis (54) too far exceeds the number of frames being fit (15), the resulting high differentiation between frames makes the network unable to exploit similarity between successive images. Our proposed solution, INVR, avoids this problem by reducing the number of frequencies along the time axis, while reallocating the remainder to the obtain a higher number of frequencies in the spatial dimensions which benefit more from them. Overall, INVR is shown to retain higher image quality than all other tested methods when compressed to the same bitrate.

4.2 Compression

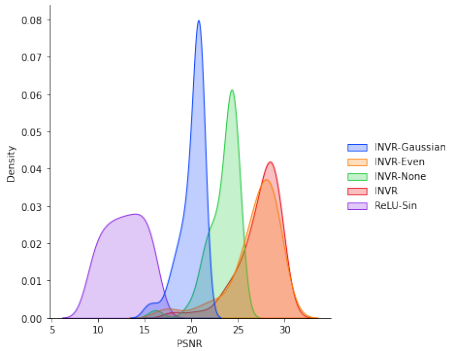
Qualitatively comparing the INVR-U and INVR-C reconstructions of the frame in Fig. 1 makes it clear that the the chosen step-size for DeepCABAC [24] is sufficiently low to allow the network to maintain most of its quality, although the INVR-C image is still noticeably noisier. When compared



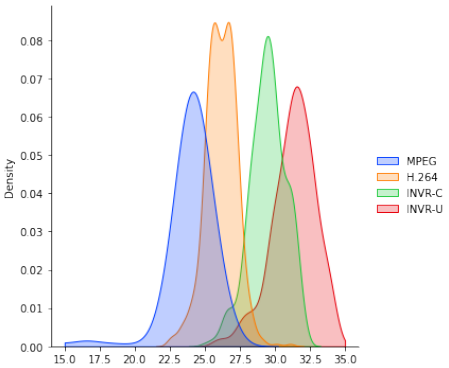
(a) Neural Video Reconstruction Average PSNR vs Iteration for One Chunk



(b) Compressed Video PSNR per Frame



(c) Probability Density Function of Frame PSNRs for One Chunk



(d) Probability Density Function of Compressed Video Frame PSNRs

Figure 2: Quantitative analysis of image quality.

against the algorithmic baselines, INVR-C is able to construct a much finer-grained image than MPEG, and is competitive with H.264.

To quantitatively evaluate the compression results we need to take into account the quality of the video encoding as well as the final-file compression ratio. A summary of this data for both $fireworks$ and $lego$ is available in Table 1.

The quality results indicate that while INVR-U provides the best average PSNR across both videos, INVR-C still outperforms both baseline methods for $fireworks$ and is only outdone by H.264 for $lego$. Interestingly, the standard deviation of INVR-C is lower than INVR-U, potentially because quantised weights lead to less noise in the activation and more homogeneous results. Overall, INVR-C and INVR in general perform the best when assessed on encoding quality.

In terms of file size, INVR-C approaches MPEG while maintaining a much higher PSNR. We must mention that DeepCABAC entropy coding and quantisation provide a remarkable near $7.5x$ reduction in file-size from INVR-U on $fireworks$ and a $8.8x$ reduction on $lego$ with only a minor loss in PSNR. Following attempts at further increasing the compression ration of INVR-C, we do not believe that large improvements can be derived merely by tuning DeepCABAC parameters – a change in network architecture is required.

Looking deeper into the specific PSNR distribution of each encoding sheds insight into the chunking process. Fig. 2b shows that the problem of variable video complexity is universal, as all techniques show the same peaks and valleys in PSNR. It is clear that fixed-length estimations are incapable of approximating such variance, and a means of automatically detecting which segments are likely to be disproportionately difficult would greatly enhance the consistency of the technique. That being said, INVR-C adapts equally well as the other compression algorithms to this inconsistent difficulty.

		Fireworks	Lego			Fireworks	Lego
PSNR	INVR-C	29.47 ± 1.28	33.77 ± 1.82	File Size (MB)	INVR-C	1.61	0.62
	INVR-U	**31.32** ± 1.57	**35.97** ± 2.08		INVR-U	12.06	5.48
	MPEG	23.48 ± 3.27	30.08 ± 3.77		MPEG	1.23	0.56
	H.264	26.13 ± 1.10	34.52 ± 1.93		H.264	**0.22**	**0.09**
					Original	5.19	1.80

Table 1: Compressed video data. Image quality averaged over all frames, with standard deviation provided, and the file size of the compressed format.

5 Conclusion

5.1 Limitations

Positional Encoding Fragility. While the exceptionally high-dimensional space-focused positional encoding provides a great boost to image reconstruction quality, it sacrifices one of the benefits mentioned by the COIN [5] authors. While a normal SIREN MLP can construct videos on different canvas sizes automatically because of its continuous representation, INVR's encoding is so sensitive that any change to the canvas size—even by one pixel—makes the output unrecognisable as it modifies the normalisation factor of all pixels. This change is then massively amplified in representation space. Such a limitation calls into question the usage of exceptionally high-dimensional encodings for certain application which could benefit from the ability to reconstruct the video at varying resolutions.

Scalability. It is unclear if the MLP-based approach can scale up to high resolution videos, especially given the plateau observed by [17] with regards to the benefits of additional hidden layers. If this limitation is not overcome, better meta-learned initialisation, entropy coding or harsher quantisation will not be sufficient to allow implicit neural representations to be successful for video compression. Outside of a fundamentally different architecture, a solution could be to decompose the video spatially (in addition to the current temporal approach) and construct many small MLPs to learn these smaller space-and-time chunks. However, this may drastically increase the compressed filesize as the weights of more MLPs will need to be transferred.

5.2 Future Work

Fully-Fused MLPs. Muller et al. [12] have proposed TinyCUDA, an MLP implementation that runs entirely on the GPU, taking full advantage of on-chip memory to ensure there is minimal access to slow global VRAM. Such a fully-fused MLP would be an excellent fit for both training and inference, as our model fits well within the memory constraints of a single GPU. It is likely a 5-10x improvement in both training and inference speed could be obtained, ensuring that playback could happen in real-time, with inference being performed on a frame-by-frame basis as necessary. The reduction in training time would significantly improve practicality of INVR, as traditional video compression techniques such as H.264 can currently operate with speeds one or more orders of magnitude higher than our approach.

Better Model Structure. We would like to replace the hand-crafted network design, parameters and encoding dimensions of INVR with a system based on relations between the resolution, length and entropy of the video itself. This could involve applying Neural Architecture Search along with hyperparameter optimisation techniques. However, performing these searches per-video would drastically inflate compression time.

5.3 Summary

We have presented INVR, an application of implicit neural representations and positional encodings to video. We utilise non-uniform encoding between the spatial and temporal dimensions alongside a time-based chunking approach to fit every (x, y, t) coordinate of a video to its RGB value. We have shown that NN-specific entropy coding and quantisation allow INVR to be used as a form of video compression, although this is not yet competitive with the state-of-the-art. Although our proposed system has limitations, it effectively demonstrates the potential of implicit neural representations for video formats.

References

[1] Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. End-to-end optimized image compression, 2017.

[2] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior, 2018.

[3] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A survey of model compression and acceleration for deep neural networks. *ArXiv*, abs/1710.09282, 2017.

[4] Lei Deng, Guoqi Li, Song Han, Luping Shi, and Yuan Xie. Model compression and hardware acceleration for neural networks: A comprehensive survey. *Proceedings of the IEEE*, 108(4):485–532, 2020.

[5] Emilien Dupont, Adam Goliński, Milad Alizadeh, Yee Whye Teh, and Arnaud Doucet. Coin: Compression with implicit neural representations, 2021.

[6] Licheng Jiao and Jin Zhao. A survey on the new generation of deep learning in image processing. *IEEE Access*, 7:172231–172263, 2019.

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[8] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[9] Detlev Marpe, Heiko Schwarz, and Thomas Wiegand. Context-based adaptive binary arithmetic coding in the h. 264/avc video compression standard. *IEEE Transactions on circuits and systems for video technology*, 13(7):620–636, 2003.

[10] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017.

[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, 2020.

[12] Thomas Müller. Tiny CUDA neural network framework, 2021. https://github.com/nvlabs/tiny-cuda-nn.

[13] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[14] Oren Rippel and Lubomir Bourdev. Real-time adaptive image compression, 2017.

[15] Yun Q. Shi and Huifang Sun. *Image and Video Compression for Multimedia Engineering: Fundamentals, Algorithms, and Standards*. CRC Press, Inc., USA, 2nd edition, 2008.

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

[17] Yannick Strümpler, Janis Postels, Ren Yang, Luc van Gool, and Federico Tombari. Implicit neural representations for image compression, 2021.

[18] Wonyong Sung, Sungho Shin, and Kyuyeon Hwang. Resiliency of deep neural networks under quantization, 2016.

[19] Vivienne Sze and Madhukar Budagavi. High throughput cabac entropy coding in hevc. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1778–1791, 2012.

[20] Vivienne Sze, Madhukar Budagavi, and Gary J Sullivan. High efficiency video coding (hevc). In *Integrated circuit and systems, algorithms and architectures*, volume 39, page 40. Springer, 2014.

[21] 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.

[22] George Toderici, Sean M. O'Malley, Sung Jin Hwang, Damien Vincent, David Minnen, Shumeet Baluja, Michele Covell, and Rahul Sukthankar. Variable rate image compression with recurrent neural networks, 2016.

[23] George Toderici, Damien Vincent, Nick Johnston, Sung Jin Hwang, David Minnen, Joel Shor, and Michele Covell. Full resolution image compression with recurrent neural networks, 2017.

[24] Simon Wiedemann, Heiner Kirchoffer, Stefan Matlage, Paul Haase, Arturo Marban, Talmaj Marinc, David Neumann, Tung Nguyen, Ahmed Osman, Detlev Marpe, Heiko Schwarz, Thomas Wiegand, and Wojciech Samek. Deepcabac: A universal compression algorithm for deep neural networks, 2019.

[25] Jianqiao Zheng, Sameera Ramasinghe, and Simon Lucey. Rethinking positional encoding, 2021.