

AUS Repository

Binary-NeRV: Hybrid-Precision Weights Binarization for Efficient Neural Video Representation

Item Type	Article;Peer-Reviewed;Published version
Authors	Shanableh, Tamer
Citation	T. Shanableh, "Binary-NeRV: Hybrid-Precision Weights Binarization for Efficient Neural Video Representation," in IEEE Access, vol. 14, pp. 16143-16157, 2026, doi: 10.1109/ACCESS.2026.3658604.
DOI	10.1109/ACCESS.2026.3658604
Publisher	IEEE
Download date	2026-03-16 06:23:40
Link to Item	https://hdl.handle.net/11073/33172

Received 14 January 2026, accepted 25 January 2026, date of publication 28 January 2026, date of current version 2 February 2026.

Digital Object Identifier 10.1109/ACCESS.2026.3658604

RESEARCH ARTICLE

Binary-NeRV: Hybrid-Precision Weights Binarization for Efficient Neural Video Representation

TAMER SHANABLEH¹, (Senior Member, IEEE)

Department of Computer Science and Engineering, American University of Sharjah, Sharjah, United Arab Emirates

e-mail: tshanableh@aus.edu

This work was supported in part by the Open Access Program of the American University of Sharjah, United Arab Emirates.

ABSTRACT Neural implicit video representations such as NeRV have emerged as a powerful alternative to traditional video codecs. However, the high computational cost and full-precision storage of NeRV limit its practicality for resource-constrained and embedded platforms. In this work, we propose Binary-NeRV, a hybrid-precision extension of NeRV that integrates XNOR-based binary convolutions into the decoding pipeline to significantly reduce model size, bitrate, and computational complexity while reasonably preserving reconstruction fidelity. Inspired by XNOR-Net, convolutional weights are binarized to ± 1 with learned scaling factors, while critical components such as the stem MLP, normalization layers, activations, skip connections, and upsampling operators are selectively retained in FP32 to ensure stable training and high visual quality. We introduce two directional progressive binarization strategies, Left-to-Right (L2R) and Right-to-Left (R2L), to analyze the impact of binarizing layers at different spatial resolutions. A detailed complexity analysis shows that over 75% of NeRV's computational cost is dominated by the final high-resolution convolutional layer, enabling highly effective targeted binarization. Extensive experiments on standard video sequences demonstrate that selectively binarizing only the deepest layer achieves up to 68–89% reduction in equivalent GFLOPs with reasonable degradation in PSNR and MS-SSIM. Temporal consistency analysis shows that selective binarization preserves temporal stability, while aggressive full-component binarization introduces noticeable flickering artifacts. We additionally present an ablation study where all convolutional layers are binarized, both before and after pruning and quantization. These results systematically validate the hybrid-precision design, showing that full binarization yields substantial bitrate reductions but at a notable quality cost, thereby justifying the selective binarization strategy adopted in Binary-NeRV. Compared with state-of-the-art NeRV variants, Binary-NeRV delivers substantial efficiency gains, establishing hybrid binarization as a practical and scalable approach for efficient neural video representation.

INDEX TERMS Neural video representation, binary neural networks, deep video coding, model binarization, deep learning.

I. INTRODUCTION

Neural implicit representations have recently emerged as a powerful paradigm for representing visual data, replacing traditional bitstream storage which is the output of video codecs [1] and [2] with overfitted deep learning models.

The associate editor coordinating the review of this manuscript and approving it for publication was Olarik Surinta¹.

The study of Implicit Neural Representations (INRs) for videos has become a major research direction in neural compression, aiming to efficiently encode video content as neural network parameters rather than explicit pixel data [3] and [4].

Among these, Neural Representations for Videos (NeRV) pioneered an image-wise implicit representation, mapping frame indices to reconstructed RGB frames through multilayer perceptron (MLPs) and convolutional decoders.

Methods such as NeRV [3] and NeRF [4] encode images or videos within neural networks, where frame indices in NeRV or input coordinates NeRF are mapped directly to RGB outputs. The resulting model encapsulates the entire visual content within its weights and parameters, offering a modern deep learning approach to video compression.

However, despite these advantages, NeRV has high model complexity and redundancy, as prior to post training quantization, its convolutional filters and fully connected layers remain in full 32-bit floating-point precision. This design limits its scalability for resource-constrained environments such as mobile and embedded systems. To address this limitation, we propose a binarized variant of NeRV, termed Binary-NeRV, that reduces memory and computational demands. Our approach builds on three core principles:

1. Binarization of convolutional layers, transforming weights to ± 1 values with learned scaling factors to preserve representational capacity.
2. Hybrid precision design, ensuring that critical elements such as normalization, activations, and skip connections remain in FP32 to maintain reconstruction quality.
3. Selective pruning and quantization, applied to the remaining floating-point components for additional compression.

The result is a model that operates at a fraction of NeRV's original bit cost which is potentially suitable for mobile and embedded systems.

II. LITERATURE REVIEW

Subsequent works on Implicit Neural Representations (INRs) for videos have focused on improving NeRV's reconstruction fidelity as measured by PSNR and SSIM and reducing the model size, often reported as bits per pixel (bpp). For instance, to address blurry reconstructions, several enhanced NeRV variants have been proposed. This includes Hybrid-NeRV which introduced content-adaptive input embeddings derived from an image encoder, rather than fixed positional encodings [5]. It redistributes network capacity to higher-resolution layers via HNeRV blocks, leading to improved reconstruction fidelity. HNeRV outperforms both NeRV on video regression tasks, improving internal consistency and high-frequency reconstruction. Another variant is the Pyramidal-NeRV which addressed spatial inconsistency issues in NeRV by introducing pyramidal feature fusion through shortcut connections between high-level and fine-grained layers [6]. It employs Kronecker fully connected layers for efficient upscaling and introduces a Benign Selective Memory (BSM) mechanism to fuse contextual and local features adaptively. Likewise, Frequency-Augmented NeRV tackles NeRV's frequency bias directly by decomposing features into high and low frequency components via wavelet-based separation [7]. The proposed Wavelet Frequency Upgrade Block (WFUB) enhances sharpness and detail recovery, achieving substantial PSNR improvements compared to baseline NeRV models. Additionally, NRVC

(Neural Representation for Video Compression) introduced an Implicit Multiscale Fusion Network (MSRVC) with Normalized Residual Convolution (NoRVC) and Feature Extraction Channel Attention (FECA) modules [8]. NRVC achieved PSNR improvements at comparable bpp to NeRV by enhancing feature extraction and residual refinement. The work in [9] proposed a spectra-preserving neural representation for video. The spectra-preserving mechanism ensures that the model maintains the fidelity of the frequency domain components of the video signal and therefore delivers reconstructions with fewer artifacts and higher perceptual quality, especially regarding intricate textures and fine details.

Other methods focus on decomposing video representation into spatial and temporal components to improve reconstruction consistency. This includes E-NeRV (Expedite NeRV) which explicitly separates spatial and temporal contexts in the network structure [10]. This disentangled design improved both training efficiency and reconstruction quality, reducing redundancy and accelerating convergence in comparison to NeRV. The work in [11] proposed a novel framework for capturing the inherent movement and temporal evolution, referred to as dynamics, present within video sequences. This modeling is accomplished via a specialized difference neural representation approach, which emphasizes the changes between frames rather than modeling absolute frame content independently. Likewise, DS-NeRV (Decomposed Static and Dynamic Codes) decomposes videos into static codes for background and dynamic codes for motion [12]. This separation allows the model to efficiently reuse static features while capturing high-frequency temporal variations. DS-NeRV achieves competitive quality with much fewer parameters than existing solutions. The work in [13] proposed a model that integrates information about the movement of pixels using optical flow across consecutive frames. As such, the frame-wise representation process for each individual frame is made more temporally consistent and accurate, leading to better overall video reconstruction.

On the other hand, efforts to minimize the number of bits per pixel, primarily focused on reducing parameter redundancy through architectural redesigns [10]. Efforts for real-time representation of video appeared in Rabbit-NeRV which optimizes for time-efficiency and encoding speed, combining best-performing modules from NeRV variants [14]. The work analyzed the impact of various components (i.e. positional encoding, stem, and upsample blocks) and proposed configurations based on different target optimizations pertaining to computational time, video quality and size trade-off.

The work in [15] proposed to process video data by dividing frames into smaller patches and applying stylistic learning or transformation to these units. This enhanced the fidelity and robustness of the resulting representation, especially concerning intricate details or complex textures within the video.

In addition to architectural optimization, recent research emphasizes quantization-aware coding and entropy modeling of trained weights to minimize bitrate. For example,

NVRC (Neural Video Representation Compression) is the first fully rate–distortion optimized INR-based video compression framework [16]. It jointly optimizes neural quantization, entropy coding, and reconstruction objectives, achieving a 23% coding gain over existing video coders. Additionally, NeuroQuant (Post-Training Quantization for INRs) introduces variable-rate quantization through network-wise calibration and channel-wise quantization [17]. NeuroQuant adjusts quantization parameters post-training, reducing encoding time up to $8\times$ while supporting coarse quantization with minimal PSNR loss. The work reported in [18], Weight Token Masking (WTM) in HyperNeRV enhances bitrate control for hyper-network-based NeRV variants. By selectively masking predicted weight tokens, WTM encourages compact encoding and enables multi-rate storage without retraining. This strategy improves PSNR and MS-SSIM at low bitrates compared to the baseline NeRV.

The reviewed literature revealed two parallel research trends, either enhancing reconstruction quality or optimizing compression. Despite substantial progress, prior works retain full-precision parameters prior to quantization. Our proposed Binary-NeRV differs by integrating binarization within the NeRV pipeline, offering an end-to-end solution for compact neural video representation. This is achieved by proposing selective binarization of convolutional layers to reduce computational complexity and bitrate while maintaining high reconstruction fidelity.

Unlike NVRC [16] and NeuroQuant [17] which operate purely in the quantization and entropy coding domain while retaining full-precision convolutional operators, Binary-NeRV fundamentally replaces the dominant convolutional computations with XNOR-based binary operators. This is not an extreme form of quantization, but a change in the underlying arithmetic, enabling a different compute–memory trade-off regime that cannot be achieved by aggressive 4–8 bit quantization alone. Additionally, While HNeRV [5] and SNeRV [9] redesign the feature hierarchy and representation capacity, Binary-NeRV is orthogonal and can be applied on top of these architectures, as it targets operator-level efficiency rather than representation design.

III. REVIEW OF NERV

The Neural Representation for Videos (NeRV) framework, introduced by Chen et al. in [3] proposes a novel paradigm for implicit neural video representation. Unlike traditional video codecs that store and reconstruct discrete pixel frames, NeRV learns a continuous neural function that directly maps a frame index to its corresponding RGB image. This eliminates the need for explicit motion estimation, discrete transforms, or hand-crafted compression heuristics.

Formally, NeRV defines a mapping function:

$$f_{\theta} : t \rightarrow I_t, \quad (1)$$

where t is a normalized temporal coordinate (frame index), and $I_t \in \mathbb{R}^{H \times W \times 3}$ is the corresponding decoded frame. The parameters θ of the function are stored as the video

representation, effectively encoding the entire sequence within a single network.

NeRV adopts an encoder-free design built from fully convolutional decoder blocks that progressively upsample spatial resolution. A frame index is first embedded using a sinusoidal or learned positional encoding, passed through a multi-layer perceptron (MLP), and then decoded through a stack of five 2D convolutional blocks with pixel shuffle upsampling. Each block increases spatial resolution while maintaining the temporal coherence of the video.

The general NeRV pipeline can be summarized as follows. The input embedding is a frame index t is encoded into a feature vector using a multi-frequency positional embedding. The stem and feature expansion provides an embedding which is projected into a higher-dimensional latent space via two fully connected (linear) layers. This is followed by progressive convolutional decoding in which the latent feature is decoded by five upsampling convolutional layers to reconstruct the final RGB frame. And the training objective is based on perceptual losses to minimize reconstruction error across all frames.

IV. PROPOSED BINARIZATION APPROACH IN NERV

This section presents the theoretical background of model binarization and presents its proposed implementation into the NeRV model.

A. BACKGROUND AND THEORETICAL FOUNDATION

The binarization scheme in NeRV follows the principles introduced in XNOR-Net [19]. The central goal is to replace floating-point convolutional operations with highly efficient binary operations, reducing both memory and computational cost without drastically compromising accuracy. In this framework, a real-valued convolutional filter W is approximated by a binary tensor $B \in \{+1, -1\}^{c \times w \times h}$ scaled by a real-valued factor α :

$$W \approx \alpha B \quad (2)$$

The optimal binary weights and scaling factors are obtained by minimizing the quantization error:

$$J(B, \alpha) = \|W - \alpha B\|^2 \quad (3)$$

subject to $B \in \{+1, -1\}^n$.

The optimal solutions are given analytically as:

$$B^* = \text{sign}(W), \alpha^* = 1/n \|W\|_1 \quad (4)$$

where α^* represents the mean magnitude of the original weights, serving as a normalization term. This formulation ensures that binary filters maintain an appropriate scale relative to their full-precision counterparts, preventing vanishing gradients and stabilizing the training process.

B. PROPOSED IMPLEMENTATION OF BINARIZATION IN NERV

In NeRV, this binarization concept is integrated through the binarizing the 2D convolution layers in the NeRV model,

hence the name Binary-NeRV. The convolutional weights are binarized as in Equations 2,3 and 4 above. This binary 2D convolution is then used to replace the standard convolution operations, leading to significant efficiency gains in both compute, model size and bitrate.

Thus, a binary-NeRV block combines these binary 2D convolutions with batch normalization, PReLU activations, and skip connections to compensate for information loss due to binarization. Importantly, the full-precision weights are still retained during training for gradient updates, following

XNOR-Net's original strategy of binarizing only in the forward and backward passes. This selective binarization allows gradients to flow through the network using the real-valued parameters, improving convergence stability.

Beyond weight binarization, the proposed binary-NeRV also approximates the dot product between input activations and weights using XNOR and bit count operations. For two real-valued vectors $X, W \in \mathbb{R}^n$, their binary approximation is formulated as:

$$X^T W \approx \beta H^T \alpha B \quad (5)$$

where $H = \text{sign}(X)$, $B = \text{sign}(W)$, and α, β are real-valued scaling factors that preserve the mean magnitude of each term. This enables efficient binary convolutions of the input tensor I computed primarily through XNOR bit count operations:

$$I * W \approx (\text{sign}(I) \odot \text{sign}(W)) \otimes (K\alpha) \quad (6)$$

where \odot denotes elementwise XNOR, and K represents the spatially varying input scaling factors.

In Binary-NeRV, the binarization process is specialized to match the network's video reconstruction objective, where weight binarization is applied per 2D convolutional block. On the other hand, scaling factors (α) are stored in FP32 for numerical stability. Batch Normalization (γ, β) parameters and PReLU slopes are kept in FP32 to maintain gradient precision, and PixelShuffle layers upsample intermediate feature maps without binarization, preserving spatial fidelity in decoded frames.

This hybrid precision design ensures that while the 2D convolutions are binarized (± 1), the model retains enough floating-point expressivity in normalization and upsampling layers to produce high-quality reconstructions.

During training, binary convolution weights are optimized using a straight-through estimator (STE). In the forward pass, real-valued weights are binarized using a sign function, which maps zeros to +1, and the convolution output is rescaled by a per-filter factor α , implemented either as the mean absolute weight magnitude or as a learnable parameter with lazy initialization. During backpropagation, gradients are propagated to the underlying real-valued weights using a straight-through estimator (STE). Following standard XNOR-style training, the derivative of the sign function is approximated by an identity function within a bounded range, while gradients outside this range are suppressed. This stabilizes training while preserving binary computation in the forward pass.

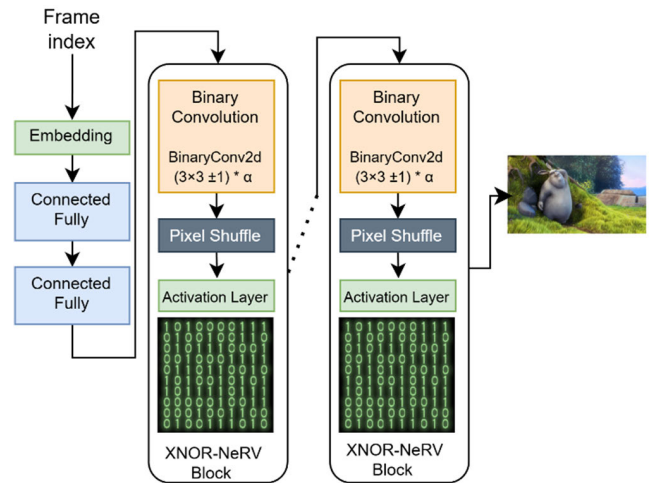


FIGURE 1. Overall architecture of the proposed Binary-NeRV.

V. PROPOSED MODEL ARCHITECTURE

The proposed Binary-NeRV architecture follows the overall design of the original NeRV framework but introduces binarization selectively to reduce model size and computation while preserving reconstruction fidelity. The generator comprises three main components including a fully-connected stem, a stack of five binary residual upsampling blocks (XNOR-NeRV Blocks), and a lightweight convolutional head that outputs RGB frames.

The stem consists of two fully connected layers ($80 \rightarrow 512$ and $512 \rightarrow 3744$) with SiLU activations, responsible for mapping temporal embeddings into a latent space. These layers remain in full precision (FP32) to retain expressive capacity, as early binarization of the input embedding was empirically found to degrade convergence and temporal coherence.

The core body contains five XNOR-NeRV Blocks, each equipped with two binary convolutional submodules (Binary Conv2d) that replace standard convolutions in NeRV. Each binary convolution learns a 1-bit weight tensor constrained to $\{-1, +1\}$, scaled by a learned floating-point coefficient α (one per output channel), following the scaling strategy introduced in XNOR-Net. This ensures stable gradient propagation during training. Surrounding components, Batch Normalization layers, 1×1 convolutional skip projections, and PReLU activations, are maintained in floating-point precision to stabilize optimization and mitigate the information bottleneck introduced by binarization.

The head module is a single 1×1 convolutional layer mapping the final feature map to RGB space. It remains in FP32 to prevent artifacts in the output frames. All upsampling is handled via parameter-free PixelShuffle operations, which are unaffected by binarization.

The architecture of the proposed Binary-NeRV is illustrated in Figure 1.

Table 1 summarizes the parameter composition of the final model. Approximately 1.24M parameters ($\approx 37\%$) are binary

TABLE 1. Parameter type summary of Binary-NeRV.

Parameter type	Precision	Count / contribution
Stem linear layers	FP32	~1.9M
BinaryConv2D	1-bit	~1.2M
α per binary filter	FP32	~0.1M
BatchNorm (γ, β)	FP32	~0.002M
PreLU	FP32	~0.1M
Skip 1×1 convs	FP32	~0.14M
Head conv2d	FP32	~0.0003M

TABLE 2. Spatial resolutions of the input feature maps to the convolutional layers considered for progressive binarization.

Layer ID	Spatial Resolution of Input (W×H)
0	9 × 16
1	45 × 80
2	90 × 160
3	180 × 320
4	360 × 640

convolutional weights, while the remaining 2.10M parameters are float-valued, primarily from the stem, normalization, skip connections, and output layers.

Again, the binary weights are from the 2D convolution layers inside each block, everything else (BNs, skip path convs, α scales, PReLU) are float. This matches the standard XNOR-ResNet design philosophy; binarize only the heavy convolutions and keep the support layers as float.

This work explores two progressive binarization strategies applied to the five 2D convolutional layers of NeRV including Left-to-Right (L2R) and Right-to-Left (R2L) progressive binarization. In the L2R approach, binarization begins with the earliest 2D convolutional layer (layer 0) and is gradually extended toward deeper layers, following the sequence: (0) → (0, 1) → (0, 1, 2) → (0, 1, 2, 3) → (0, 1, 2, 3, 4). Conversely, in the R2L approach, binarization starts from the deepest layer (layer 4) and progresses toward the shallower ones in the order: (4) → (3, 4) → (2, 3, 4) → (1, 2, 3, 4) → (0, 1, 2, 3, 4).

These two directional schemes produce distinct effects on reconstruction quality, bitrate efficiency, and computational complexity measured in FLOPs, as analyzed in the experimental results section. The differences primarily arise from the varying spatial resolutions of feature maps processed by each convolutional layer. This is so because the later layers operate on higher-resolution maps, thereby dominating the total computational cost. Table 2 summarizes the spatial dimensions of the inputs to the five convolutional layers targeted for binarization.

VI. QUANTIZATION AND PRUNING IN BINARY-NERV

While binarization reduces parameter precision and model size, additional compression can be achieved through structured pruning and post-training quantization. In the proposed Binary-NeRV, these stages are designed to complement the

binarization pipeline rather than conflict with it. Specifically, pruning targets full-precision weights, and quantization is applied to non-binary parameters, such as scaling factors and normalization statistics, while binary weights remain fixed at one bit. This ensures that compression is achieved without introducing instability into the binary convolutional layers.

A. PRUNING STRATEGY

Similar to NeRV, pruning in Binary-NeRV follows a global unstructured magnitude pruning scheme. After each pruning step, the remaining parameters are re-evaluated, and the model undergoes fine-tuning to recover any potential degradation in PSNR or SSIM. Formally, for a given parameter tensor W , pruning is applied as:

$$W_p = M \odot W \quad (7)$$

where $M \in \{0, 1\}^{|W|}$ is a binary mask computed by thresholding the magnitude of W . For binary layers, M is set to an all-ones tensor to prevent any change. This masking formulation allows gradients to flow through W during fine-tuning, maintaining differentiability while preserving the binarized forward computation.

B. QUANTIZATION OF NON-BINARY LAYERS

Following pruning and fine-tuning, quantization with various bit widths is applied to all remaining non-binary parameters, ranging from 4 to 16 bits. While binary convolutional parameters are excluded from quantization since they already occupy a single bit per weight. However, the scaling factors (α) associated with each binary filter are quantized to align with the rest of the network's compressed representation.

Following the original NeRV architecture, the quantization process also computes an entropy-encoded bit efficiency using Huffman coding to estimate actual storage savings.

VII. EXPERIMENTAL RESULTS

In this section we evaluate the proposed solutions in terms of quality, bitrate and computational complexity in comparison to existing work.

Using the video sequence deployed in the original NeRV paper [3], we evaluate the proposed Binary-NeRV framework on the Bunny video sequence, which contains 132 frames of resolution $1280 \times 720 \times 3$. We then repeat the best results on videos sequences from the UVG sequences [21]. Representative images from each sequence are shown in Figure 7. We adopt the small NeRV model configuration as the baseline, as it produces a network with approximately 3.2 million parameters before compression, making it suitable for video coding while maintaining tractable training complexity. Unless otherwise specified, all models were generated using 300 epochs with a batch size of 1, and pruning refinement was conducted over an additional 100 epochs with a pruning ratio of 40% as reported in NeRV.

A. EXPERIMENTAL SETUP

As described in Sections IV and V, only the 2D convolutional layers are binarized in this work. The reference NeRV architecture consists of five convolutional layers, and we investigate the effects of progressively binarizing the layers with the proposed Binary-NeRV solution. This experimental design enables an analysis of the trade-off between compression (measured by bpp) and video reconstruction quality (measure by PSNR and MS-SSIM). The calculation of the bpp for the proposed Binary-NeRV solution is detailed in Appendix A.

The subsequent experiments present the results of directional progressive binarization of the 2D convolutional layers under both Left-to-Right (L2R) and Right-to-Left (R2L) configurations. The original NeRV model [3] is used as the baseline reference.

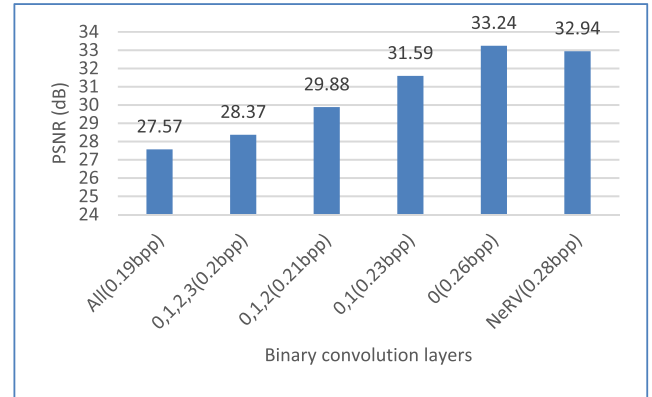
B. RECONSTRUCTION QUALITY PRIOR TO PRUNING AND QUANTIZATION

Figure 2 presents the average reconstruction quality (PSNR) prior to pruning and quantization, with the corresponding average bit-per-pixel (bpp) values shown in parentheses. The figure is divided into two parts: (a) Left-to-Right (L2R) progressive binarization and (b) Right-to-Left (R2L) progressive binarization.

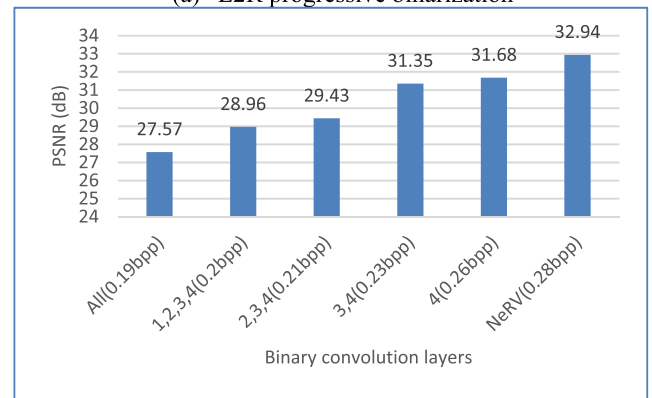
In the L2R direction, where binarization begins from the first convolutional layer (layer 0) and extends toward deeper layers, the results exhibit a predictable trade-off between compression and quality. When only the first layer is binarized, the model achieves higher quality to the baseline NeRV model while reducing the bitrate from 0.28 bpp to 0.26 bpp, again, before pruning and quantization. When only the first layer is binarized, in comparison to NeRV, the model reduces the bitrate from 0.28 bpp to 0.26 bpp, again, before pruning and quantization. Extending binarization to the first two or three layers maintains competitive PSNR values with additional bitrate savings (≈ 0.21 – 0.23 bpp). Fully binarizing all convolutional layers achieves the strongest compression (0.19 bpp) but at a more noticeable quality drop.

Interestingly, binarizing only the first convolutional layer (Layer 0) resulted in a marginal improvement in reconstruction quality (33.24 dB vs 32.94 dB for the full-precision baseline). This can be attributed to the fact that layer 0 suppresses small-magnitude weights and acts like an implicit low-pass filter. That can reduce pixel-level noise in reconstructions. Simultaneously, the model adapts the weights of deeper layers to compensate for the early binarization. Nonetheless, as additional layers are binarized, the expected trend of reduced reconstruction fidelity and improved compression becomes evident.

In contrast, the R2L direction—starting from the last convolutional layer (layer 4) and progressively moving toward shallower layers, exhibits slightly lower PSNR values for the same number of binarized layers. This difference can be attributed to the higher spatial resolution of later layers, where binarization more directly impacts the reconstruction output.



(a) L2R progressive binarization



(b) R2L progressive binarization

FIGURE 2. Reconstruction quality (PSNR) versus the number of binarized convolutional layers in the proposed Binary-NeRV architecture before pruning and quantization.

Overall, the results in Figure 2 demonstrate that L2R progressive binarization provides a more favorable rate–distortion balance. Partial binarization of early convolutional layers retains most of NeRV’s reconstruction fidelity while achieving up to 30% reduction in bitrate before any pruning or quantization is applied. However, since the input spatial resolution to early convolution layers is small, the overall effect on reducing computational complexity is low in comparison to R2L progressive binarization as detailed later in the experimental results.

It is worth noting that, with the proposed weight binarization and selective pruning and quantization, a number of the proposed progressive binarization schemes will result in a reconstruction quality that surpasses the baseline NeRV as illustrated in the experiments to follow.

C. RATE-DISTORTION (RD) PERFORMANCE WITH PRUNING AND QUANTIZATION

To further enhance compression efficiency, we apply the joint pruning and quantization pipeline described in Section VI. Specifically, a 40% global pruning ratio is used, and the remaining floating-point weights are quantized to integer precisions ranging from 4 to 16 bits.

Table 3 summarizes the average PSNR and bits-per-pixel (bpp) for the two binarization orders; Left-to-Right (L2R)

and Right-to-Left (R2L), where layers closer to the input or output are progressively binarized. Results are averaged across quantization levels $q = 4, 6, 8, 12,$ and 16 . The pruning ratio applied is 40% . The corresponding PSNR average over all quantization bit widths for the NeRV baseline is 28.82 dB with an average of 0.049 bpp.

The listed values in Table 3, reveal a consistent trend in which Left-to-Right binarization yields slightly higher average PSNR (29.08 dB vs. 28.76 dB) and lower average bpp (0.0398 vs. 0.0403) compared to Right-to-Left binarization. The corresponding PSNR average over all quantization bit widths for the NeRV baseline is 32.1 dB with an average of 0.0398 bpp.

The listed values in Table 3, reveal a consistent trend in which Left-to-Right binarization yields slightly higher average PSNR (29.21 versus 28.71 dB) and slightly higher average bpp (0.0435 vs. 0.0425) compared to Right-to-Left binarization.

Additionally, when only one layer is binarized (layer 0 vs. layer 4), the performance gap is even larger: $+1.9$ dB in favor of binarizing the first convolutional stage. As binarization extends to deeper layers, the degradation in PSNR is more severe for the Left-to-Right case, compared to Right-to-Left. This behavior indicates that binarizing early (low-level) layers, which process the embedded input before high-resolution spatial upsampling, retains better representational power than binarizing late (high-resolution) layers responsible for fine detail synthesis. The deeper layers of NeRV appear more sensitive to binarization due to their role in reconstructing texture and high-frequency content.

Overall, the results suggest that progressive binarization from early to deeper layers (L2R) provides a more favorable trade-off between compression and quality, likely because low-level filters can be coarsely quantized without significantly impairing spatial detail reconstruction downstream.

For a detailed comparison, Figure 3 illustrates the Rate-Distortion (RD) curves of all progressive binarization configurations for both L2R and R2L orders. Each curve corresponds to a specific set of binarized convolutional layers, while the quantization bit-width varies from 4 to 16 .

In the figure, the L2R curves are dashed and the R2L curves are connected. The results clearly show that progressive L2R binarization (starting from the first convolutional layer) consistently yields higher PSNR at a given bitrate than the R2L strategy (starting from the last layer). Specifically, binarization of the early layers (e.g., layers 0 and 1) achieves a favorable trade-off, maintaining high reconstruction quality while reducing the average bitrate by roughly $25\text{--}30\%$ compared to the fully float baseline of NeRV.

As more layers are binarized, the performance gap between the two binarization strategies narrows, and when all layers are binarized, both approaches converge to the same RD curve since they share identical binary configurations. Nevertheless, the L2R order demonstrates smoother degradation and better resilience to quantization, suggesting that early-stage binarization is less detrimental to spatial detail reconstruction

TABLE 3. PSNR and bpp averaged over quantization bit widths of 4,6,8,12 and 16 for the two proposed progressive binarization solutions (L2R) and (R2L). NeRV baseline readings are 31.6 dB and 0.0456 bpp.

Binary layers L2R		
Binary layers	Avg. PSNR(dB)	Avg bpp.
0,	31.60	0.0456
0,1	30.43	0.0374
0,1,2	28.34	0.0429
0,1,2,3	26.67	0.0407
Avg.	29.21	0.0435

(a) L2R progressive binarization

Binary layers R2L		
Binary layers	Avg. PSNR(dB)	Avg bpp.
4,	29.66	0.0423
3,4	29.17	0.0419
2,3,4	28.96	0.0355
1,2,3,4	27.25	0.0394
Avg.	28.78	0.0398

(b) R2L progressive binarization

than binarizing later, high-resolution layers responsible for texture synthesis.

As listed in Table 2, the input resolutions ($H \times W$) for convolutional layers 0 through 4 are $9 \times 16, 45 \times 80, 90 \times 160, 180 \times 320,$ and $360 \times 640,$ respectively. Consequently, the performance differences observed between the L2R and R2L binarization schemes can be attributed to the varying spatial resolutions of these layers. In general, higher-resolution layers contribute more to the total computational cost but are also more sensitive to binarization, leading to a larger degradation in PSNR when binarized.

The RD curves in Figure 4 highlight the clear advantage of the proposed Binary-NeRV architecture over the baseline NeRV model across all bitrates. At comparable bits-per-pixel (bpp) levels, the proposed solution consistently achieves higher PSNR values, by up to $+2.0$ dB on average.

Overall, these results confirm that binary convolution in early layers can bitrate compared to traditional full-precision NeRV architectures, even before applying pruning or additional quantization.

Overall, these results confirm that binary convolution in early layers can substantially improve rate-distortion efficiency compared to traditional full-precision NeRV architectures, even before applying pruning or additional quantization.

Figure 4 also includes the RD curve corresponding to the configuration where convolutional layers 0 and 1 are binarized. As expected, the overall RD performance is slightly lower than the case of binarizing only layer 0.

To systematically validate the design choice of retaining specific components in full precision, we conduct an ablation

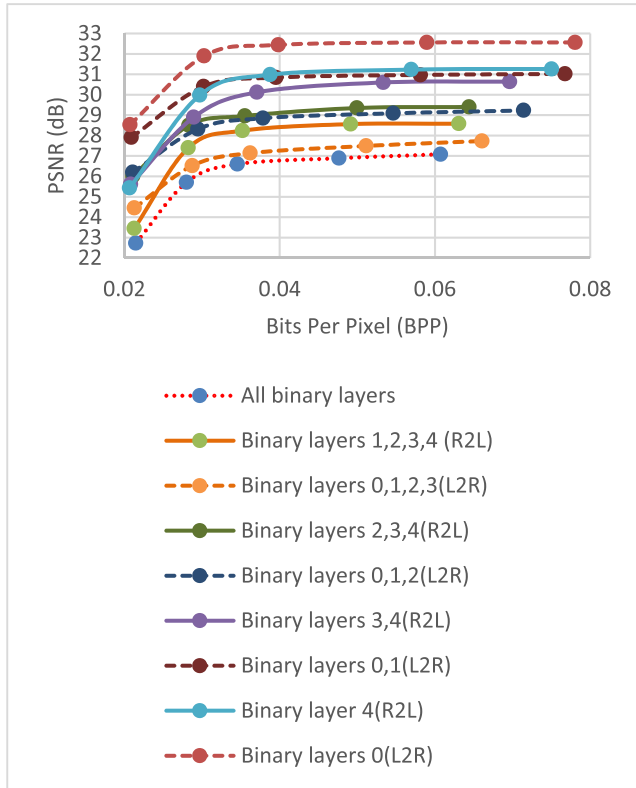


FIGURE 3. Rate-Distortion (RD) curves comparing Left-to-Right (L2R) and Right-to-Left (R2L) progressive binarization across different quantization bit-widths.

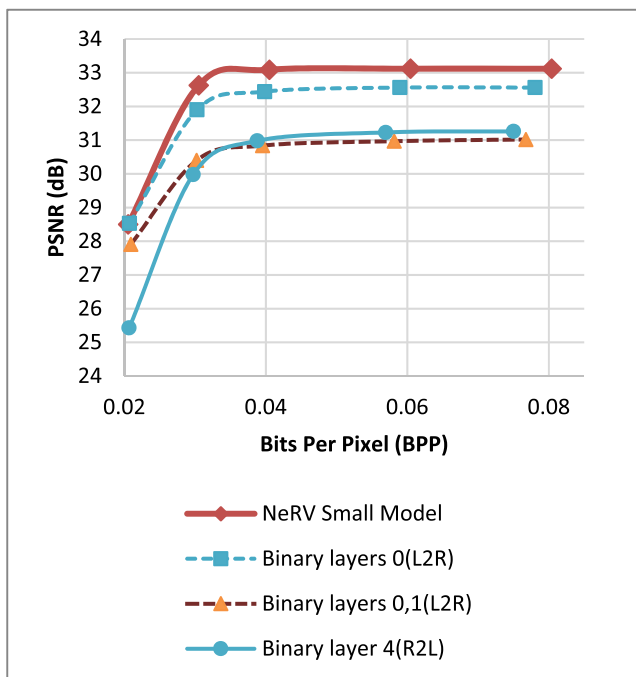


FIGURE 4. Rate-Distortion (RD) comparison between the proposed Binary-NeRV (binarized layer 0 and layers 0,1) and the baseline small NeRV model across multiple quantization levels.

study comparing the standard Binary-NeRV configuration (binarizing decoder layers 0–4 while keeping the stem in

TABLE 4. Ablation study on full-component binarization showing a comparison between partial (layers 0–4) and fully binarized Binary-NeRV configurations.

	Binary layers 0-4	All binary configuration
PSNR	27.7	23.38
MS-SSIM	0.8496	0.6881
BPP	0.56499	0.53215

(a) Before pruning and quantization

	Binary layers 0-4	All binary configuration
PSNR	26.60	21.93
MS-SSIM	0.8218	0.6261
BPP	0.03456	0.014673

(b) After pruning and quantization (8 bits)

FP32) against a fully binarized model in which all components, including the stem and head, are binarized.

As shown in Table 4, full binarization results in a substantial degradation in reconstruction quality, with PSNR dropping from 27.7 dB to 23.38 dB and MS-SSIM decreasing from 0.8496 to 0.6881 before pruning and quantization. After applying 40% pruning and 8-bit quantization, the quality gap further widens (26.60 dB vs. 21.93 dB PSNR; 0.8218 vs. 0.6261 MS-SSIM), despite the fully binarized model achieving lower bitrate. These results demonstrate that while full-component binarization yields additional bitrate reduction, it comes at a large cost in visual quality, confirming that the stem and early feature extraction layers are particularly sensitive to aggressive precision reduction. This empirically validates our design choice to retain selected components in full precision, achieving a more favorable trade-off between compression efficiency and reconstruction quality.

D. TEMPORAL FLICKER ANALYSIS OF BINARY-NERV

To address concerns regarding temporal artifacts introduced by aggressive binarization, we evaluate temporal consistency using temporal PSNR (tPSNR), temporal SSIM (tSSIM), and LPIPS-based temporal perceptual distance. Table 5 parts (a) and (b) report results for both Left-to-Right (L2R) and Right-to-Left (R2L) binarization strategies at different depths.

For the NeRV baseline, high temporal stability is observed (tPSNR = 37.22 dB, tSSIM = 0.955), and the LPIPS ratio between reconstructed and ground truth temporal changes is below 1 (0.841), indicating that the reconstructed sequence exhibits less perceptual temporal variation than the ground truth, i.e., no flickering amplification.

Under aggressive L2R binarization of layers 0–3, temporal quality degrades noticeably (tPSNR = 31.60 dB, tSSIM = 0.913), and the LPIPS recon/GT ratio increases to 2.357, indicating amplification of perceptual temporal changes, which is consistent with visible flicker. As fewer layers are binarized, temporal stability progressively improves: for L2R

TABLE 5. Temporal flicker analysis of Binary-NeRV.

	NeRV baseline	L2R layers 0-3	L2R layers 0,1,2	L2R layers 0,1	L2R layers 0
tPSNR	37.22	31.60	33.37	35.22	36.70
tSSIM	0.955	0.913	0.930	0.944	0.954
LPIPS(recon)	0.019	0.054	0.038	0.027	0.021
LPIPS(GT)	0.023	0.023	0.023	0.023	0.023
Ratio recon/GT	0.841	2.357	1.651	1.173	0.925
(a) L2R binarization					
	NeRV baseline	R2L layers 1-4	R2L layers 2,3,4	R2L layers 3,4	R2L layers 4
tPSNR	37.22	32.60	33.75	34.25	35.05
tSSIM	0.955	0.923	0.931	0.936	0.939
LPIPS(reco)	0.019	0.040	0.033	0.030	0.027
LPIPS(GT)	0.023	0.023	0.023	0.023	0.023
Ratio recon/GT	0.841	1.774	1.450	1.299	1.185
(b) R2L binarization					

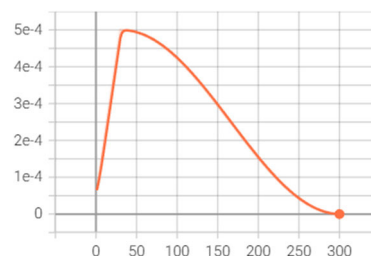
layers 0–1, tPSNR increases to 35.22 dB, tSSIM to 0.944, and the LPIPS ratio drops to 1.173. When only layer 0 is binarized, temporal metrics are very close to baseline (tPSNR = 36.70 dB, tSSIM = 0.954), and the LPIPS ratio falls below 1 (0.925), indicating no perceptual flicker amplification.

A similar trend is observed for the R2L strategy. Full R2L binarization of layers 1–4 leads to moderate temporal degradation (tPSNR = 32.60 dB, tSSIM = 0.923, LPIPS ratio = 1.774). However, as binarization is restricted to deeper layers (layers 3–4 and layer 4 only), temporal stability improves steadily. With only layer 4 binarized, tPSNR reaches 35.05 dB, tSSIM 0.939, and the LPIPS ratio reduces to 1.185, approaching baseline behavior.

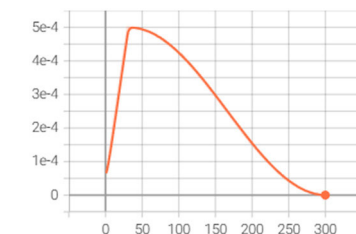
Overall, these results demonstrate that temporal incoherence is correlated with the depth and ordering of binarized layers. Early-layer binarization (L2R) is more prone to introducing flicker due to disruption of low-level spatial and temporal feature representations, whereas deeper-layer binarization (R2L) preserves temporal stability more effectively. Importantly, for moderate binarization configurations (e.g., L2R layer 0 only or R2L layer 4 only), Binary-NeRV maintains temporal consistency comparable to the NeRV baseline while achieving significant computational and bitrate reductions.

E. OPTIMIZATION STABILITY AND CONVERGENCE BEHAVIOR

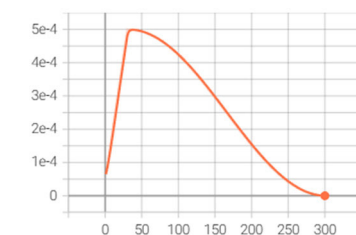
To analyze the impact of binarization on optimization stability, we report training loss and validation PSNR convergence curves for the NeRV baseline, full binarization, and selective



(a) Train loss versus epochs, (NeRV baseline)



(b) Train loss versus epochs, (Binarize all layers)



(c) Train loss versus epochs (Binarize layer 4)

FIGURE 5. Training loss convergence under different binarization configurations.

binarization configurations. The plots are shown in Figures 5 and 6 below.

Figure 5 and Figure 6 report the training loss and validation PSNR convergence under different binarization configurations, including the NeRV baseline, full binarization of all decoder layers, and selective binarization of layer 4 only. The baseline NeRV model exhibits smooth and stable convergence, with monotonic loss reduction and steady PSNR improvement.

In contrast, full binarization of all decoder layers converges more slowly and shows increased variance in both loss and PSNR, reflecting the increased optimization difficulty commonly associated with binary networks. Importantly, no divergence or training collapse was observed in any configuration. When binarization is restricted to deeper layers (e.g., layer 4 only), convergence behavior closely matches that of the baseline, with stable loss decay and smooth PSNR progression. These results demonstrate that while aggressive binarization increases optimization sensitivity, Binary-NeRV remains stable and trainable, and selective binarization provides a favorable trade-off between efficiency gains and optimization robustness.

F. RESULTS ON ADDITIONAL VIDEO SEQUENCES

In addition to the Bunny sequence, which was used in the original NeRV paper, we test the proposed solution on 3 more

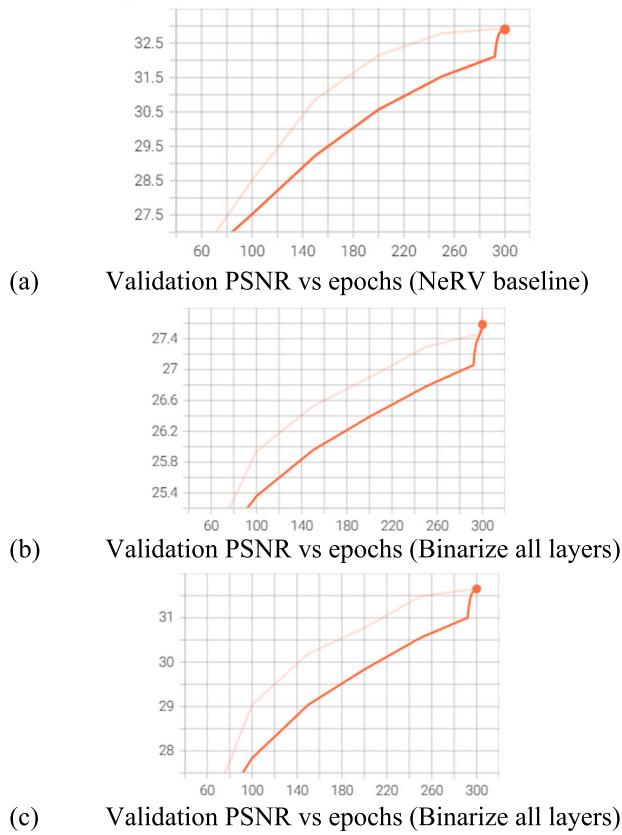


FIGURE 6. Validation PSNR convergence under different binarization configurations.



FIGURE 7. First video frame of each video sequence.

sequences from the UVG dataset [21], specifically, we use the Jockey, Yacht Ride and Honeybee sequences. The first frame of each sequence is shown in Figure 7.

The best results obtained thus far are obtained from the binarization of the 4th convolution layer followed by the binarization of the 3rd and 4th convolution layers, thus, we repeat the RD curves for the UVG sequences using these solutions.

In Figure 8 and 9, the results are obtained after resizing the UVG sequences to the same dimensions of the NeRV sequence using the same number of video frames. Figure 8

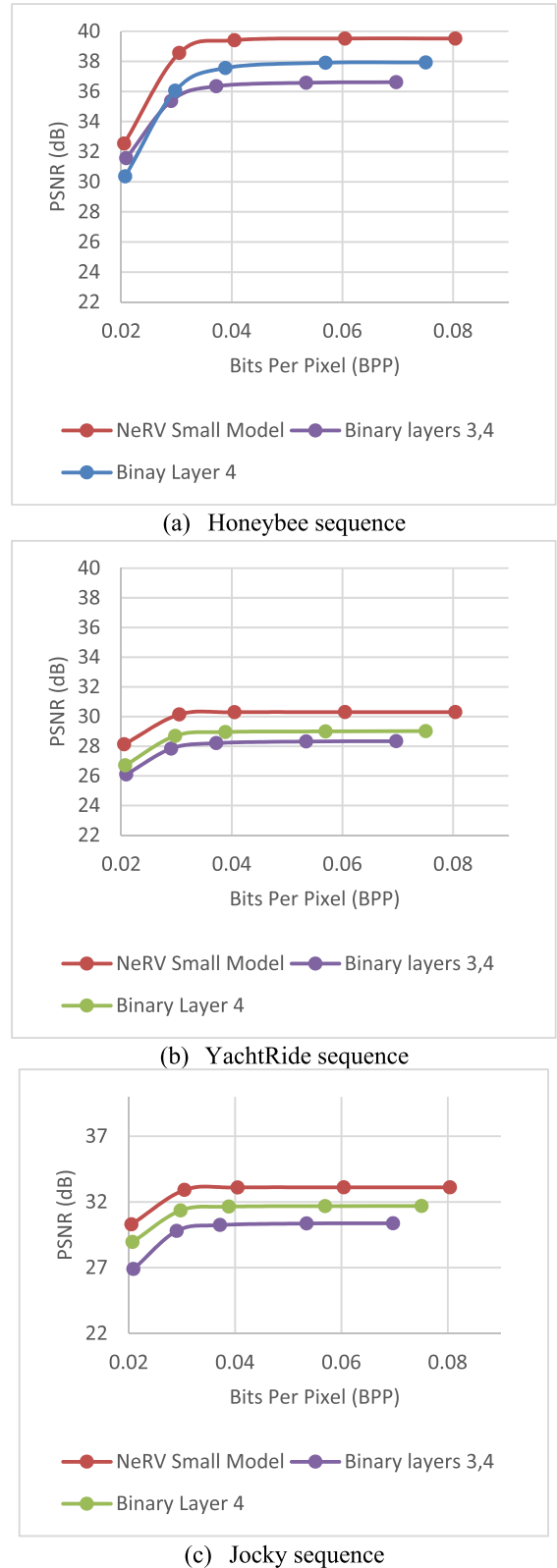


FIGURE 8. PSNR-based rate-distortion curves for the proposed solution with binarizing layers 3 and or 4 using three additional video sequences.

plots the RDs using PSNR and Figure 9 plots the same results using MS-SSIM.

TABLE 6. Average decrease in PSNR and reduction percentage in bitrate for the four video sequences.

	Decrease in PSNR (dB)	Reduction in Bitrate (%)
Honeybee	-1.94	4.84%
Yacht Ride	-1.35	4.84%
Jockey	-1.44	4.84%
Bunny	-2.6	4.84%
Avg	-1.83	4.84%

The same results are repeated in Figure 9 but displays the bpp versus the MS-SSIM results. A comparison between PSNR-based and MS-SSIM-based RD curves reveals that perceptual quality, as measured by MS-SSIM, degrades a bit more gracefully than PSNR for natural video sequences. In particular, Honeybee, Jockey, and Yacht Ride exhibit relatively smaller MS-SSIM losses compared to their PSNR drops, indicating that binarization primarily affects high-frequency distortions that are less perceptually salient. In contrast, the cartoon-style Bunny sequence shows a larger degradation under both metrics due to its reliance on sharp edges and uniform regions.

The four evaluation sequences used in this section exhibit different visual characteristics, which helps explain the observed variation in rate–distortion behavior under progressive binarization. Among them, Bunny is the only cartoon-style sequence, while HoneyBee, Jockey, and YachtRide are natural video sequences with increasing levels of texture complexity and motion.

The Bunny sequence is dominated by large smooth regions, sharp edges, and piecewise-constant textures. In contrast, the HoneyBee sequence contains fine-grained textures and high-frequency spatial detail, making it sensitive to binarization. The Jockey sequence introduces complex motion, articulated objects, and sharp luminance transitions, placing higher demands on temporal and spatial fidelity. Finally, YachtRide exhibits large smooth regions combined with structured edges and camera motion, resulting in intermediate sensitivity to binarization.

The PSNR results of the RD curves are summarized in Table 6 below.

On average, the proposed binarization strategy results in a PSNR decrease of less than 2 dB while achieving a consistent bitrate reduction of 4.84% across all sequences. The identical bitrate reduction is expected, as all sequences share the same spatial resolution and frame count.

Overall, these results demonstrate that selective, resolution-aware binarization consistently reduces bitrate across diverse video content. While texture-rich natural sequences may incur a perceptual quality penalty that must be traded against compression efficiency, the proposed strategy maintains favorable perceptual fidelity under MS-SSIM while simultaneously achieving substantial reductions in

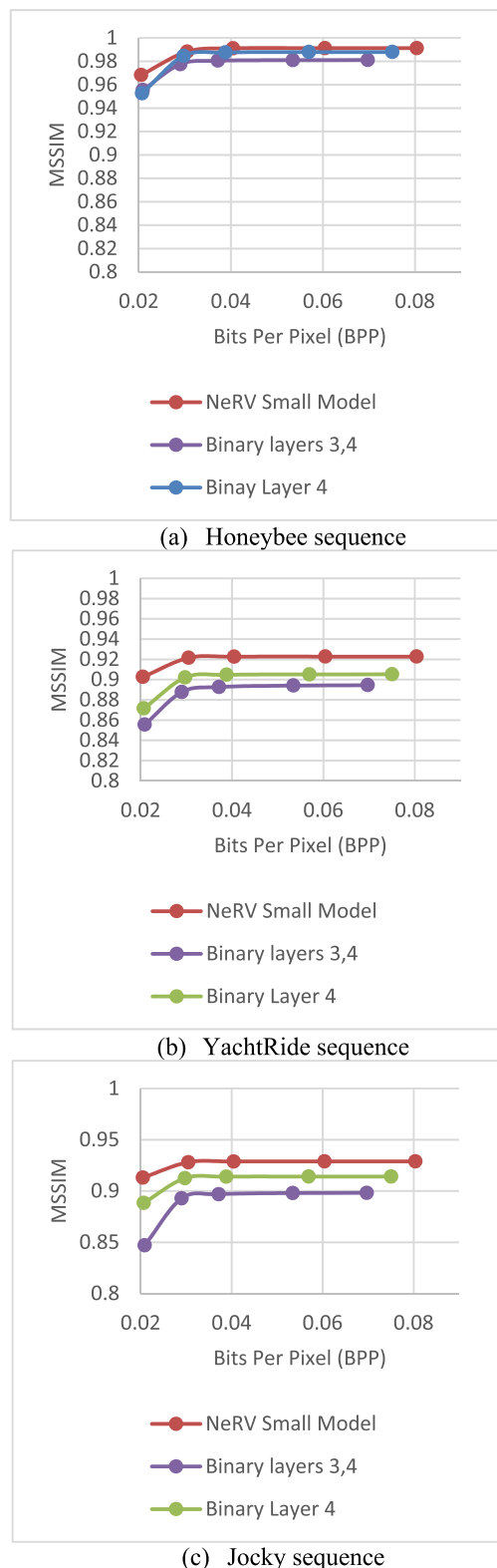


FIGURE 9. MS-SSIM-based rate-distortion curves for the proposed solution with binarizing layers 3 and 4 using three additional video sequences.

computational complexity, as detailed in the next section. Rate–distortion trade-offs remain content-dependent and are not uniform across all sequences.

TABLE 7. Computational complexity analysis of the baseline NeRV model.

Layer	Type	Input Shape	GFLOPs
stem-linear 0	Linear	80	0.00008
stem-linear 1	Linear	512	0.0038
Conv-L0	Conv2d (26,650,k=3)	(26, 9, 16)	0.044
Conv-L1	Conv2d (26,384,k=3)	(26, 45, 80)	0.646
Conv-L2	Conv2d (96,384,k=3)	(96, 90, 160)	9.556
Conv-L3	Conv2d (96,384,k=3)	(96, 180, 320)	38.22
Conv-L4	Conv2d (96,384,k=3)	(96, 360, 640)	152.882
head	Conv2d (96,3,k=1)	(96, 720, 1280)	0.53
Total			201.88

G. COMPUTATIONAL COMPLEXITY ANALYSIS

To quantify the computational complexity of the proposed solutions, we start by analyzing the baseline NeRV architecture used. We calculate the MACs and FLOPs of each layer as listed in Table 7.

In our proposed solution, we progressively binarize the 2D convolution layers, and this is where the computational saving are achieved. An interesting observation is that the fourth layer of the 2D convolution counts for 75.7% of the total GFLOPs of the model. Likewise, layers 3 and 4 of the 2D convolution counts for 94.6% of the total GFLOPs of the model. Binarizing such layers results in massive drop in complexity. On the other hand, binarizing layers 0 and 1 of the 2D convolution does not result in a noticeable reduction in complexity as they count for around 3% of the GFLOPs.

As for our proposed Binary-NeRV solution, Table 8 summarizes the progressive reduction in GFLOPs as binary 2D convolution layers are introduced into the NeRV architecture. The Table also presents the binary operation involved in 2D convolution and their equivalent FLOPs.

All results are computed for 720p video frames, with equivalent FLOPs reflecting binary operations (BOPs) converted to FP32 FLOP cost using the standard XNOR-popcount (population count) efficiency factor (1 BOP \approx 1/64 FLOP). Hence the total equivalent GFLOPs are equal to GFLOPs (of float layers) + GBOPs (of binary layers)/64.

For the R2L scheme, binarizing only the final convolution layer (L4) reduces the total computation by about 67%, while progressively binarizing layers 4, 3 and 2, leads to an 89% FLOP reduction. When all five convolutional stages are binarized, the equivalent computation decreases from 224 GFLOPs to 24 GFLOPs, corresponding to nearly a 9 \times improvement in efficiency.

TABLE 8. Computational complexity of the proposed solutions measured in FLOPs and BOPs.

Model	GFLOPs	GBOPs	(GFLOPs & GBOPs) Equivalent GFLOPs
NeRV baseline[3]	201.9	0	201.9
Binary layers 4	71.4	76.44	72.6
Binary layers 3,4	33.1	95.55	34.6
Binary layers 2-4	23.6	100.33	25.2
Binary layers 1-4	22.9	100.65	24.5
Binary layers 0-4	22.9	100.67	24.5

(a) R2L progressive binarization

Model	GFLOPs	GBOPs	(GFLOPs & GBOPs) Equivalent GFLOPs
NeRV baseline[3]	201.9	0	202
Binary layers 0	201.8	0.02	201.8
Binary layers 0,1	201.2	0.34	201.3
Binary layers 0-2	192.6	5.12	192.7
Binary layers 0-3	158.2	24.23	158.6
Binary layers 0-4	22.9	100.67	24.5

(b) L2R progressive binarization

On the other hand, the L2R scheme shows a much slower initial reduction, since early layers operate at much smaller spatial resolutions and contribute less to total compute. Binarizing up to layer L2 (i.e. layers 0,1 and 2) yields only a 4.5 % decrease, whereas binarizing layers L0–L3 results in a 21 % reduction, and full binarization again reaches the 89% reduction limit.

Overall, these results confirm that binarizing deeper layers first (R2L) provides a more favorable compute–accuracy trade-off. This is because late decoder stages dominate FLOPs due to their large spatial dimensions after upsampling. Hence, a selective binarization policy that targets the last one or two convolutional blocks captures most of the computational savings while preserving the rate–distortion performance observed in Figures 3 and 4.

Recall from Figure 4 above, that binarizing convolution layer 0 resulted in higher PSNR at slightly lower bitrate compared to the NeRV baseline model, yet the reduction in FLOPs is insignificant as listed in Table 8. However, binarizing convolution layer 4 resulted in lower bitrate and slightly higher PSNR in comparison to NeRV. This comes with the advantage of a 68% reduction in FLOPs are listed in Table 8 above.

In comparison to prior research, Table 9 summarizes the reported per-frame computational cost (in GFLOPs) for representative NeRV-based architectures, as published in the

Algorithm 1 Binary-NeRV Pipeline with Pruning ($p = 0.4$) and 8-bit Quantization

Input:

- Video frames $\{I_t\}$ for $t = 1 \dots T$ at resolution $H \times W$
- NeRV generator G_θ with K decoder stages
- Binarization plan $B \subseteq \{1 \dots K\}$
- Training epochs E , learning rate η
- Pruning ratio $p = 0.4$
- Quantization bitwidth $q = 8$
- Entropy coder C

Output:

- Compressed bitstream \mathcal{B}
- Reconstructed frames $\{\hat{I}_t\}$

Procedure

1. Data Preparation
 - 1.1. For $t = 1$ to T :
 - Extract frame I_t , normalize to $[0, 1]$, resize to $H \times W$
 - 1.2. Build dataset $D = \{(t, I_t)\}$
2. Model Initialization
 - 2.1. Initialize NeRV generator G_θ in FP32
3. Hybrid-Precision Training with Binarization
 - 3.1. For epoch = 1 to E :
 - 3.1.1. For each minibatch $\{(t_i, I_i)\} \subset D$:
 - a) Compute positional encoding $x_i \leftarrow PE(t_i)$
 - b) For $s = 1$ to K (decoder stages):
 - If $s \in B$:
 - Apply weight binarization: $W_s^b \leftarrow \text{sign}(W_s) \cdot \alpha_s$
 - Else:
 - Use FP32 weights W_s
 - c) Reconstruct frame $\hat{I}_i \leftarrow G_\theta(x_i)$
 - d) Compute reconstruction loss:

$$L = \|\hat{I}_i - I_i\|_1 + (1 - \text{SSIM}(\hat{I}_i, I_i))$$
 - e) Backpropagate L using straight-through estimator
 - f) Update θ using learning rate η
4. Structured Pruning ($p = 0.4$)
 - 4.1. For each weight tensor W in G_θ :
 - Compute importance scores (e.g., magnitude)
 - Prune lowest 40% of weights by setting them to zero
5. Fine-Tuning of Pruned Model (Optional)
 - 5.1. For epoch = 1 to E_{ft} :
 - Repeat Step 3 using remaining non-zero weights only
6. Post-Training Quantization (8-bit)
 - 6.1. For each remaining FP32 tensor (scaling factors, biases, BatchNorm parameters):
 - Quantize tensor to 8-bit using uniform quantizer $Q(\cdot)$
7. Bitstream Construction
 - 7.1. Serialize binary weight tensors (1 bit per non-zero weight)
 - 7.2. Serialize pruning masks / indices
 - 7.3. Serialize 8-bit quantized parameters
 - 7.4. Pack all symbols into stream S
8. Entropy Coding
 - 8.1. $\mathcal{B} \leftarrow C(S)$
9. Decoding and Reconstruction
 - 9.1. Decode \mathcal{B} to recover binary weights, pruning masks, and quantized parameters
 - 9.2. Rebuild hybrid-precision model \hat{G}
 - 9.3. For $t = 1$ to T :
 - $\hat{I}_t \leftarrow \hat{G}(PE(t))$
10. Output
 - 10.1. Return reconstructed frames $\{\hat{I}_t\}$ and compressed bitstream \mathcal{B}

literature for the Bunny input videos with a spatial resolution of 720×1280 .

TABLE 9. Reported per-frame computational cost (in GFLOPs) of existing NeRV variants.

Existing work	GFLOPs
NeRV (baseline) [3]	201.9
E-NeRV [10]	208
HNeRV [5]	188
FFNeRV [13]	204
SNeRV-B [9]	363.6
SNeRV-T [9]	206.6
HiNeRV [20]	190
DNeRV [11]	181
Proposed (Binary-NeRV, Layer 4)	72.6

When compared to Table 8, and considering the higher 720p input resolution used in our analysis, the Binary-NeRV configurations, particularly those with binarized layer 4 on its own or layers 3-4, exhibit highly competitive computational efficiency relative to these state-of-the-art NeRV variants. Namely, as reported in Table 8 the GFLOPs of the proposed solution of binarizing layer 4 is 72.6 and that of layers 3-4 is 34.6. Our reported 72.6 GFLOPs is achieved while maintaining reconstruction quality comparable to that of the original NeRV baseline.

The following is a summary of experimental findings; across all evaluated sequences (Bunny, Honeybee, Jockey, and YachtRide), Binary-NeRV consistently achieves substantial reductions in computational complexity and bitrate while maintaining competitive reconstruction quality. Full binarization of all decoder layers provides the highest efficiency gains but introduces increased optimization sensitivity and perceptual degradation, particularly on high-motion and texture-rich content. In contrast, selective and deep-layer binarization offers a favorable trade-off, preserving PSNR, MS-SSIM, and temporal stability close to the NeRV baseline while still delivering significant complexity and bitrate reductions. Temporal analysis using tPSNR, tSSIM, and LPIPS-Temporal confirms that early-layer binarization is more prone to flicker, whereas later-layer binarization maintains temporal coherence. Convergence analysis further shows that Binary-NeRV remains stable and trainable under all configurations, with no divergence observed. Overall, the results demonstrate that hybrid-precision binarization is an effective and practical strategy for efficient neural video representation.

VIII. CONCLUSION

This work introduced Binary-NeRV, a hybrid-precision extension of NeRV that combines binarization, pruning, and quantization to realize efficient neural video representation framework. By selectively binarizing convolutional layers, the proposed model achieves an effective balance between computational efficiency and reconstruction fidelity.

A comprehensive study of directional progressive binarization revealed complementary advantages for the two examined schemes: Left-to-Right (L2R) binarization which starts from early convolutional layers, offers superior

rate–distortion performance. While Right-to-Left (R2L) binarization which starts from high-resolution layers, maximizes computational savings. Across configurations, Binary-NeRV reduces the computational cost by 68%–89% in GFLOPs relative to the full-precision NeRV baseline. Compared with state-of-the-art NeRV-based architectures, the proposed Binary-NeRV delivers complexity reductions that tend to outperform existing work with 73 GFLOP in comparison to the lowest reported GFLOPs of 181.

While this work focuses on the NeRV architecture, the proposed binarization strategy operates at the convolutional operator level and is therefore, in principle, applicable to other INR-based video models such as HyperNeRV, DS-NeRV, and related decoders. However, generalization has not yet been experimentally validated and remains a direction for future work.

We acknowledge several limitations of the proposed solution. Performance degradation is more pronounced for sequences with high motion or rich texture content, such as Jockey and Honeybee, where aggressive binarization can amplify reconstruction error and temporal instability. In addition, early-layer binarization is more susceptible to introducing perceptual artifacts and flicker, as shown in the temporal consistency analysis. Moreover, the current study focuses on the standard small NeRV configuration; extending the proposed binarization strategy to larger and more expressive NeRV variants remains a direction for future research. Finally, while analytical complexity reductions strongly indicate suitability for mobile and embedded deployment, real-time hardware benchmarks and energy measurements are left for future work.

APPENDIX A

A. BITS-PER-PIXEL (BPP) COMPUTATION FOR BINARY-NeRV

In this work, the final bitrate of the neural video representation is reported in terms of bits-per-pixel (BPP), following the same principle as the original NeRV formulation, i.e., the total number of stored model bits normalized by the total number of pixels in the video sequence. However, unlike NeRV, Binary-NeRV contains heterogeneous parameter types, including binary weights, per-filter scaling factors, and remaining full-precision parameters. We therefore explicitly account for each category.

Formally, the total number of bits required to store the model is computed as:

$$B_{\text{total}} = B_{\text{bin}} + B_{\alpha} + B_{\text{float}} + B_{\text{other}} \quad (8)$$

where B_{bin} is the contribution of binary convolutional weights, B_{α} is the contribution of the per-filter scaling factors α , B_{float} is the contribution of remaining full-precision weights after pruning and quantization and B_{other} accounts for auxiliary parameters such as batch normalization parameters and biases.

Each term is computed as $B_{\text{bin}} = N_{\text{bin}} \times 1$, $B_{\alpha} = N_{\alpha} \times b_{\alpha}$, $B_{\text{float}} = N_{\text{float}} \times b_q$ and $B_{\text{other}} = N_{\text{other}} \times b_{\text{other}}$. Where

N_{bin} is the number of binary convolutional weights (stored using 1 bit each), N_{α} is the number of scaling factors α (one per output channel in binary convolution layers), b_{α} is the bitwidth used to store each scaling factor (e.g., 16 or 32 bits), N_{float} is the number of remaining trainable weights after pruning (e.g., stem layers, skip connections, head layers), b_q is the quantization bitwidth applied to these weights, N_{other} is the number of auxiliary parameters (batch normalization and biases) and b_{other} is the bitwidth used to store these auxiliary parameters.

Pruning is applied prior to quantization, and only the non-zero parameters are counted in N_{float} . Binary weights are excluded from pruning and quantization, as they are already stored in 1-bit form. The final bits-per-pixel (BPP) is then computed as:

$$\text{BPP} = \frac{B_{\text{total}}}{H \times W \times T} \quad (9)$$

where H is the frame height, W is the frame width and T is the number of frames in the video sequence. This formulation generalizes the original NeRV bitrate computation:

$$\text{BPP}_{\text{NeRV}} = \frac{N_{\text{params}} \times (1 - s) \times b_q}{H \times W \times T} \quad (10)$$

where N_{params} is the total number of parameters, s is the model sparsity, and b_q is the quantization bitwidth. In the special case where all parameters are uniformly quantized and no binarization is used, our formulation reduces exactly to the NeRV expression. For entropy coding, Huffman coding is applied following the same protocol as in NeRV.

APPENDIX B

B. BINARY-NeRV PIPELINE PSEUDOCODE

See Algorithm 1.

ACKNOWLEDGMENT

This article represents the opinions of the author does not mean to represent the position or opinions of the American University of Sharjah.

REFERENCES

- [1] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003, doi: 10.1109/TCSVT.2003.815167.
- [2] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012, doi: 10.1109/TCSVT.2012.2221191.
- [3] H. Chen, B. He, H. Wang, Y. Ren, S. N. Lim, and A. Shrivastava, "NeRV: Neural representations for videos," in *Proc. NeurIPS*, 2021, pp. 21557–21568.
- [4] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing scenes as neural radiance fields for view synthesis," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2020, pp. 405–421.
- [5] H. Chen, M. Gwilliam, S.-N. Lim, and A. Shrivastava, "HNeRV: A hybrid neural representation for videos," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 10270–10279.
- [6] Q. Zhao, M. S. Asif, and Z. Ma, "PNeRV: Enhancing spatial consistency via pyramidal neural representation for videos," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2024, pp. 19103–19112.

- [7] L. Yu, Z. Li, C. Yao, J. Xiao, and M. Gabbouj, "FANeRV: Frequency separation and augmentation based neural representation for video," *Expert Syst. Appl.*, vol. 299, Mar. 2026, Art. no. 129935, doi: 10.1016/j.eswa.2025.129935.
- [8] S. Liu, P. Cao, Y. Feng, Y. Ji, J. Chen, X. Xie, and L. Wu, "NRVC: Neural representation for video compression with implicit multiscale fusion network," *Entropy*, vol. 25, no. 8, p. 1167, Aug. 2023.
- [9] J. Kim, J. Lee, and J.-W. Kang, "SNeRV: Spectra-preserving neural representation for video," in *Proc. 18th Eur. Conf. Comput. Vis. (ECCV)*, 2024, pp. 332–348.
- [10] Z. Li, M. Wang, H. Pi, K. Xu, J. Mei, and Y. Liu, "E-NeRV: Expedite neural video representation with disentangled spatial-temporal context," in *Proc. ECCV*, 2022, pp. 267–284.
- [11] Q. Zhao, M. S. Asif, and Z. Ma, "DNeRV: Modeling inherent dynamics via difference neural representation for videos," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 2031–2040.
- [12] H. Yan, Z. Ke, X. Zhou, T. Qiu, X. Shi, and D. Jiang, "DS-NeRV: Implicit neural video representation with decomposed static and dynamic codes," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2024, pp. 23019–23029.
- [13] J. C. Lee, D. Rho, J. H. Ko, and E. Park, "FFNeRV: Flow-guided frame-wise neural representations for videos," 2023, pp. 7859–7870.
- [14] H. Chen, S. N. Lim, and A. Shrivastava, "RNeRV: How to design and train your implicit neural representation for video compression," 2024, *arXiv:2506.24127*.
- [15] Y. Bai, C. Dong, C. Wang, and C. Yuan, "PS-NeRV: Patch-wise stylized neural representations for videos," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2023, pp. 41–45.
- [16] H. M. Kwan, G. Gao, F. Zhang, A. Gower, and D. Bull, "NVRC: Neural video representation compression," in *Proc. NeurIPS*, 2024, pp. 132440–132462.
- [17] J. Shi, Z. Chen, H. Li, Q. Zhao, M. Lu, T. Chen, and Z. Ma, "On quantizing neural representation for variable-rate video coding," in *Proc. 13th Int. Conf. Learn. Represent.*, 2025, pp. 98407–98430.
- [18] M. Gwilliam, R. Zhang, N. Padmanabhan, H. Du, and A. Shrivastava, "How to design and train your implicit neural representation for video compression," Jun. 2025, *arXiv:2506.24127*.
- [19] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," in *Proc. Eur. Conf. Comput. Vis.*, vol. 9908, Cham, Switzerland: Springer, 2016, pp. 525–542, doi: 10.1007/978-3-319-46493-0_32.
- [20] H. M. Kwan, G. Gao, F. Zhang, A. Gower, and D. Bull, "HiNeRV: Video compression with hierarchical encoding-based neural representation," in *Proc. 37th Int. Conf. Neural Inf. Process. Syst.*, Aug. 2023, pp. 72692–72704.
- [21] A. Mercat, M. Viitanen, and J. Vanne, "UVG dataset: 50/120 fps 4K sequences for video codec analysis and development," in *Proc. 11th ACM Multimedia Syst. Conf.*, May 2020, pp. 297–302.



TAMER SHANABLEH (Senior Member, IEEE) was born in Scotland, U.K. He received the M.Sc. degree in software engineering and the Ph.D. degree in electronic systems engineering from the University of Essex, in 1998 and 2002, respectively. He was a Senior Research Officer with the University of Essex for three years, during which he collaborated with BTextact on inventing video transcoders. He is a Professional Engineer. He then joined Motorola U.K. Research Laboratories and contributed to establishing a new profile within the ISO/IEC MPEG-4 known as the Error Resilient Simple Scalable Profile. He joined the American University of Sharjah, in 2002. He is currently a Professor of computer science. During the summer breaks, he was a Visiting Professor with Motorola Laboratories for five different years. He spent his sabbatical leave as a Visiting Academic with the Multimedia and Computer Vision and Laboratory at Queen Mary, University of London, U.K. He has six patents and authored more than 100 publications, including 10 IEEE TRANSACTIONS papers. His research interests include video coding and processing, and deep learning.

• • •