



Quality Scalable Video Coding Based on Neural Representation

Qian Cao¹, Dongdong Zhang^{1(✉)}, and Chengyu Sun²

¹ Department of Computer Science and Technology, Tongji University, Shanghai, China

2230788@tongji.edu.cn, ddzhang@tongji.edu.cn

² Shanghai Key Laboratory of Urban Renewal and Spatial Optimization Technology, Tongji University, Shanghai, China

cy.sun@tongji.edu.cn

Abstract. Neural Representation for Videos (NeRV) encodes each video into a network, providing a promising solution to video compression. However, existing NeRV methods are limited to representing single-quality videos with fixed-size models. To accommodate varying quality requirements, NeRV methods need multiple separate networks with different sizes, resulting in additional training and storage costs. To address this, we propose a Quality Scalable Video Coding method based on Neural Representation, in which a hierarchical network consisting of a base layer (BL) and several enhancement layers (ELs) represents the same video with coarse-to-fine qualities. As the smallest subnetwork, the BL represents basic content. The larger subnetworks can be formed by gradually adding the ELs which capture residuals between the lower-quality reconstructed frames and original ones. Since the larger subnetworks share the parameters of the smaller ones, our method saves 40% of storage space. In addition, our structural design and training strategy enable each subnetwork to outperform the baseline on average +0.29 PSNR.

Keywords: Implicit Neural Representation · Video compression · Quality Scalable coding

1 Introduction

Traditional video compression approaches such as H.264 [1], HEVC [2] rely on manually-designed modules, such as motion estimation and discrete cosine transform (DCT). With the success of deep learning, learning-based codecs [3,4] replace handcraft modules with neural networks and achieve improved rate-distortion performance. Recently, Implicit Neural Representation (INR) has received increasing attention in the signal compression tasks such as image [5–7] and video [8–16], due to its simpler pipelines and faster decoding speed.

Neural Representation for Videos (NeRV) methods [10–16] train a neural network to represent a video, thus encoding the video into the network weights. The encoding is the process of training a neural network to overfit video frames, and the decoding is a simple forward propagation operation. The network itself

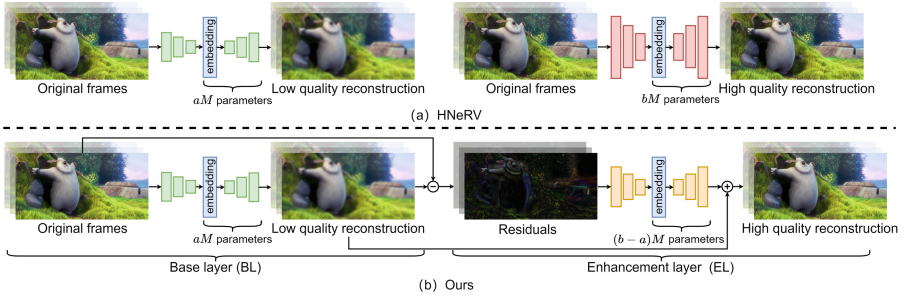


Fig. 1. An example to represent a video at two different quality levels: HNeRV requires two separate networks with different sizes (aM and bM parameters), while our methods achieves this with just one network with totaling bM parameters. The network comprises a BL that can independently decode the lower-quality video and a EL that enhances the quality by adding residual details to the BL.

serves as the video stream, and the model size dictates the trade-off between rate and distortion. Larger networks exhibit greater reconstruction accuracy at the expense of storage space and transmission bandwidth.

The original videos often need to be transmitted in different qualities adapted to terminal device and network conditions. For NeRV methods, this requires multiple sizes of the networks to represent the video at different quality levels. Since these networks of different sizes represent the same video content, there is information redundancy among them. However, existing methods do not consider this, and each network is trained and stored independently, resulting in significant costs. If the larger network can directly leverage the knowledge learned by the smaller ones instead of starting training from scratch, it can clearly converge faster and save storage space by sharing parameters. Scalable Video Coding (SVC) schemes [17, 18] encapsulate videos with varying quality in a hierarchical video stream that can be partially decoded to provide flexible quality. Motivated by this, we propose using a single network that can be separated into executable subnetworks with different sizes to represent a video in flexible quality. The small subnetwork can operate independently or as part of larger ones. Therefore, larger subnetworks can leverage information already learned by smaller ones.

In this paper, we propose a quality scalable video coding method based on neural representation (QSVCNR). Building upon HNeRV [15], we design a hierarchical network consisting of a base layer (BL) and one or more enhancement layers (ELs). An example is illustrated in Fig. 1. The hierarchical network enables a coarse-to-fine reconstruction and naturally decomposes the representation into a basic component and progressive levels of details, separately preserved in different layers. As the smallest subnetwork, BL can independently reconstruct a coarser video. Larger subnetworks are constructed by gradually introducing ELs which fit the residuals between the lower-quality reconstructed frames and original ones. We employ per-layer supervision to train the entire network end-to-end. The main contributions are as follows:

- We propose a quality scalable video coding method based on neural representation, in which a hierarchical network consisting of a BL and several ELs represents the same video with coarse-to-fine qualities.
- We introduce additional structures including a context encoder for BL, residual encoders and inter-layer connections for ELs, and design an end-to-end training strategy, to improve the representation capability of our method.
- On the bunny [19] and UVG [20] datasets, Our proposed method saves 40% of storage space, and each subnetwork exhibits an average improvement of +0.29 PSNR compared to the baseline.

2 Related Work

Implicit Neural Representation (INR) [21–23] parameterize the signal as a function approximated by a neural network, which map reference coordinates to their respective signal values. For instance, an image can be represented as function $f(x, y) = (R, G, B)$, where (x, y) are the coordinates of a specific point. This concept has been applied to compression tasks such as image [5–7] and videos [8, 9]. However, these multi-layer perceptron (MLP) based INRs are limited to pixel-wise representation. They predict a single point at a time, resulting in an unacceptably slow coding speed when applied to video representation.

Neural Representation for Videos (NeRV) is specifically designed for video, including two branches: Index-based and Hybrid-based. Index-based methods view video as an implicit function $y = f(t)$ where y is the t^{th} frame of video. NeRV [10] first proposes image-wise representation with convolutional layers, achieving better reconstruction quality and greatly improving efficiency. ENeRV [11] Separate the representation into temporal and spatial branches. NVP [12] uses learnable positional features. D-NeRV [13] introducing temporal reasoning and motion information. However, Index-based methods only focus on location information but ignore the visual information. Hybrid-based method CNeRV [14] proposes a hybrid video neural representation with content-adaptive embeddings. HNeRV [15] proposes that auto-encoder is also a hybrid neural representation. The encoder extracts content-adaptive embedding, while the decoder and embeddings are viewed as neural representations. DNeRV [16] introduces difference between frames as motion information. Whereas, existing methods are limited to representing a single video with a fixed-size neural network, which lacks flexibility in practical applications. We adopt the framework proposed in [15] as the baseline and propose a scalable coding method.

Video Compression algorithms such as H.264 [1], HEVC [2] achieve efficient compression performance. The learning-based codecs [3, 4] use neural networks to replace certain components but still follow the traditional pipeline. These codecs explicitly encodes videos as latent codes of motion and residual. NeRV methods represent videos as neural networks, and then employ model compression techniques like model pruning and weight quantization to reduce the model size further. It provides a novel pipeline for video compression, achieving fast decoding and satisfactory reconstruction quality with a simpler structure.

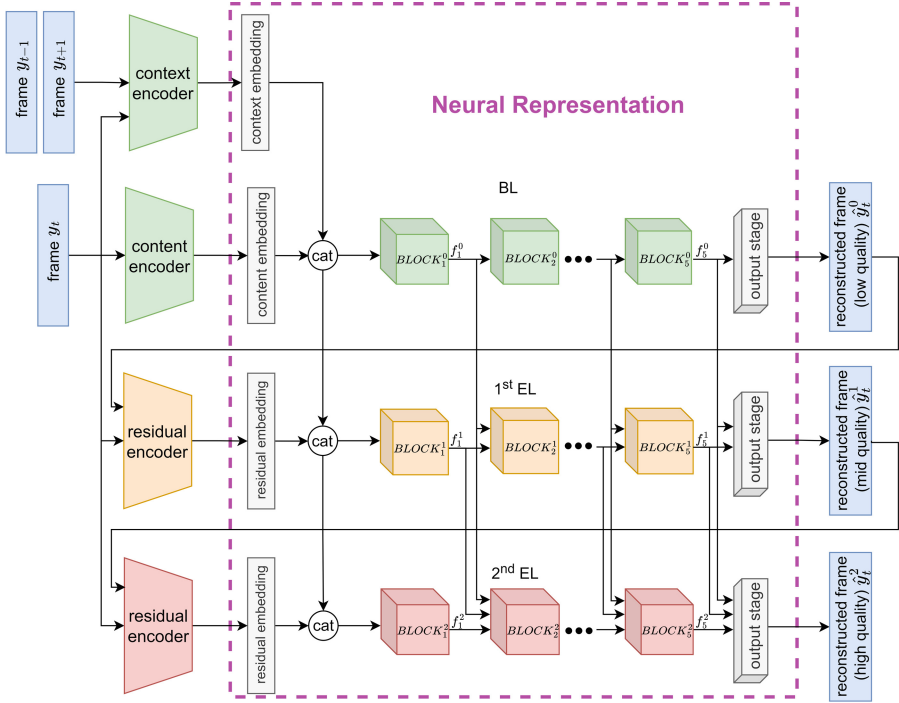


Fig. 2. The pipeline of quality scalable video coding method based on neural representation. Model architecture comprises 3 layers: BL (green part), 1st EL (yellow part) and 2nd EL (red part). The purple part is the implicit neural representation for video. (Color figure online)

Scalable Video Coding (SVC) is an important extension for video compression, e.g., SVC [17] for H.264 and SHVC [18] for HEVC. SVC obtains multi-layer streams (BL and ELs) through one encoding to accommodate different conditions. The BL can be independently decoded, and the ELs can be appended to the BL to enhance the output quality. Typical scalability modes include temporal, spatial, and quality scalability. In the NeRV method, the network itself is the video stream. Inspired by multi-scale implicit neural representation [24, 25], we propose a hierarchical network for implementing quality scalable coding.

3 Methodology

3.1 Overview

Figure 2 illustrates the overview of our Quality Scalable Video Coding method based on Neural Representation. We follow the hybrid philosophy proposed in [15] that treats the embedding and decoder of the auto-encoder as the INR

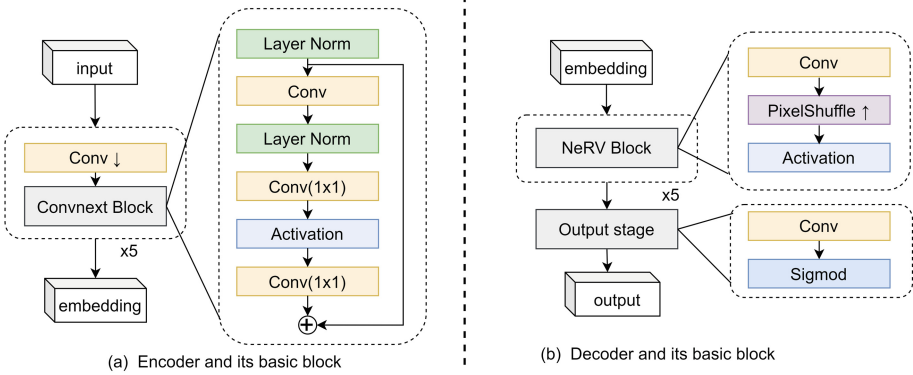


Fig. 3. Structure of encoder and decoder

for videos. Our proposed network is hierarchical and consists of a BL and several ELs. BL learns basic low-frequency information from current frame content and neighboring contextual information. ELs learn high-frequency detailed information from residue between the reconstructed frames and original ones. The modular structure allows for the flexible selection of a BL and any number of ELs to form subnetworks with different sizes, thus achieving quality scalability.

3.2 The Base Layer

The base layer (BL) is the initial layer and can function as an independent neural representation for coarse reconstruction.

The encoder comprises two components: the content encoder and the context encoder. The content encoder extracts the essential content-adaptive embedding $e^{content}$ from the current frame y_t . However, the baseline methods [15] overlook the contextual dependency on adjacent frames, which plays a significant role in video tasks. We draw inspiration from [16] and introduce a context encoder. The context encoder captures short-term context by considering neighboring frames y_{t-1}, y_{t+1} , allowing for an explicit utilization of the temporal correlations.

$$\begin{aligned}
 e^{content} &= CONTENT_ENC(y_t) \\
 e^{context} &= CONTEXT_ENC(y_t, y_{t-1}, y_{t+1})
 \end{aligned} \tag{1}$$

Each encoder consists of five stages, including a downsampling convolutional layer and a ConvNext block [26], as illustrated in Fig. 3(a). All encoders share the same structure, with variations only in the inputs. This enables all encoders generate the embeddings of the same size, which can be merged through direct concatenation. The dimensions of the embeddings are flexible and determined by the downsampling stride and embedding channels specified in the hyperparameters. We maintain consistency with the baseline approach. For 1920×960 video, the downsampling strides are set as [5, 4, 4, 3, 2], providing the embedding in size of $16 \times 4 \times 2$.

The content embedding $e^{content}$ and context embedding $e^{context}$ are combined as inputs and pass through the decoder. Finally, the output stage maps the final output features into pixel domain and gets the reconstructed frame \hat{y}_t^0 as follows:

$$\begin{aligned} f_1^0 &= BLOCK_j^0(e^{content}, e^{context}) \\ f_j^0 &= BLOCK_j^0(f_{j-1}^0) \\ \hat{y}_t^0 &= Sigmoid(Conv(f_5^0)) \end{aligned} \quad (2)$$

The decoder includes five blocks, denoted as $BLOCK_1^0$ through $BLOCK_5^0$, and f_j^0 represents the output intermediate feature of the j^{th} block. We utilize NeRV block [10] as the fundamental unit of decoder, as shown in Fig. 3(b). Among these, only the convolutional layer contains trainable parameters, and the pixel shuffle layer serves as upsampling method. To ensure a consistent size between the reconstructed frames and original ones, the upsampling strides are the same as the downsampling strides applied in the encoder.

3.3 The Enhancement Layers

The Enhancement Layers (ELs) builds on the BL to capture progressive levels of details, enabling higher-quality reconstruction.

It is inadequate for ELs to share a single fixed-size embedding with the BL. The ELs primarily learn high-frequency details, but the low-frequency part of embedding leads to poor fitting of the high-frequency information. It is important for each layer’s embedding to align with its representation content, and the embedding size should adapt to the changing subnetwork sizes. The high-frequency details that previous layers fail to reconstruct are exactly the residuals. Therefore, we introduce a residual encoder for each EL to extract the residual embedding r^i from the reconstructed frame of the previous layer \hat{y}_t^{i-1} and the original frame y_t . The structure of the residual encoder is the same as the encoders of BL described in the previous section.

$$r^i = RESIDUAL_ENC(\hat{y}_t^{i-1}, y_t) \quad (3)$$

Regarding the decoder, a straightforward approach is to apply five blocks, similar to the base layer (BL), and aggregate the final features from different layers only in the output stage. In addition, we introduce Inter-layer connections at each block to effectively utilize the information preserved in lower layers, as shown in Fig. 2. Specifically, embeddings and intermediate features from previous layers are included as additional input to each block of EL as follows:

$$\begin{aligned} f_1^i &= Block_j^i(e^{content}, e^{context}, r^1, \dots, r^i) \\ f_j^i &= Block_j^i(f_{j-1}^0, \dots, f_{j-1}^i) \\ \hat{y}_t^i &= Sigmoid(Conv(f_5^0, \dots, f_5^i)) \end{aligned} \quad (4)$$

where, $Block_j^i$ is the j^{th} block of the i^{th} EL, and f_j^i denotes its output intermediate feature. The reconstructed frame, \hat{y}_t^i is the output of the subnetwork consisting of the BL and i ELs.

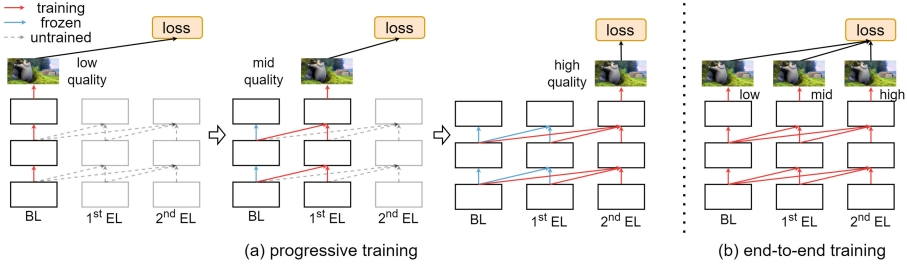


Fig. 4. Training strategy

While the Inter-layer connections require BL to accommodate the representation tasks of higher layers, thereby impacting its performance, ELs can effectively leverage the knowledge acquired from previous layers through this structure to better capture high-frequency residuals. Notably, connections are unidirectional, maintaining the independence of lower layers. Consequently, even without higher ELs, the preceding layers can still serve as a standalone subnetwork representing lower-quality frames. This modular structure allows the BL and ELs to form subnetworks with different sizes, thus providing flexible quality scalability.

3.4 End-to-End Multi-task Training Strategy

There are two training strategies: progressive training [24,27] and end-to-end training. In progressive training, BL is trained first, and then ELs are trained layer by layer with the weights of previous layers frozen, as illustrated in Fig. 4(a). This approach maintains the performance of the previous layers as the training progresses. However, it only considers the current layer, ignoring that each layer is part of larger subnetworks. As a result, this strategy may fail to achieve the global optimal between multiple subnetworks.

Each subnetwork shares specific layers and has similar objective to reconstruct the same frames at different quality. We adopt a multi-task learning strategy that enables adaptive balancing of the objectives across subnetworks. Unlike training each layer sequentially, our strategy trains the entire network end-to-end, as depicted in Fig. 4(b). Each layer is supervised with an individual loss, and these losses are weighted and summed to obtain the total loss. Higher weights are assigned to ELs to emphasize the capture of high-frequency details. The total loss function is expressed in Eq. 5.

$$Loss = \sum_{i=1}^l \lambda_i MSE(\hat{y}_t^i, y_t) \tag{5}$$

where l is the total number of layers, MSE is Mean squared error, \hat{y}_t^i represents the reconstructed frame of the i^{th} layer, and y_t represents the original frame, λ_i denotes the weight for the loss of the i^{th} layer.

4 Experiments

4.1 Datasets and Settings

Datasets. Experiments are conducting on the bunny [19] and UVG [20] datasets. The bunny is a video consisting of 132 frames with a 1280×720 resolution. The UVG dataset contains seven videos, in a total of 3900 frames, with a 1920×1080 resolution. To ensure consistency with previous approaches, we apply center-cropping to the videos to match the size of the embeddings. Specifically, we center crop the bunny to 1280×640 and UVG videos to 1920×960 .

Settings. The stride list for encoder and decoder is set as [5, 4, 3, 2, 2] for the UVG and [5, 4, 2, 2, 2] for the bunny. We vary the number of channels of the NeRV blocks to obtain representation models of specific sizes. During training, we use the Adam optimizer [28] with cosine learning rate decay. The maximum learning rate is $1e-3$, with 10% of epochs dedicated to warm-up. The batch size is 1. The reconstruction quality is assessed using the peak signal-to-noise ratio (PSNR), while video compression performance is evaluated by measuring the bits per pixel (Bpp) required for encoding the video. All experiments are performed in the PyTorch framework on an RTX 3090 GPU.

Table 1. Video reconstruction on bunny and UVG.

Model	bunny	UVG							
		beauty	bosph	bee	jockey	ready	shake	yach	avg.
NeRV 0.5M	25.77	30.53	28.90	32.05	26.48	20.71	28.41	24.89	27.42
NeRV 1.5M	29.20	32.00	31.09	36.28	28.95	22.79	31.57	26.35	29.86
NeRV 3M	32.67	32.88	33.22	38.44	31.03	24.73	33.52	27.73	31.65
NeRV avg	29.21	31.80	31.07	35.59	28.82	22.74	31.16	26.32	29.64
ENeRV 0.5M	27.07	31.16	29.68	36.10	25.84	20.56	30.99	25.30	28.51
ENeRV 1.5M	31.01	33.25	31.11	37.68	27.59	22.36	33.37	26.00	30.19
ENeRV 3M	35.41	34.06	33.94	38.59	29.52	24.34	35.30	27.74	31.92
ENeRV avg	31.16	32.82	31.57	37.45	27.65	22.42	33.22	26.34	30.21
HNeRV 0.5M	31.98	31.69	30.49	36.79	26.83	21.02	32.44	25.94	29.31
HNeRV 1.5M	35.57	33.06	33.06	38.65	29.79	23.66	34.06	27.85	31.44
HNeRV 3M	37.43	33.58	34.73	38.96	32.04	25.74	34.57	29.26	32.69
HNeRV avg	34.99	32.77	32.76	38.13	29.55	23.47	33.69	27.68	31.14
QSVCNR(2/3 training) 0.5M	32.07	31.70	30.50	37.10	27.65	21.47	32.30	25.96	29.53
QSVCNR(2/3 training) 1.5M	35.97	32.90	32.83	38.54	30.28	24.02	33.99	27.83	31.48
QSVCNR(2/3 training) 3M	38.02	33.46	34.17	39.12	32.04	25.76	34.97	28.98	32.64
QSVCNR(2/3 training) avg	35.35	32.69	32.50	38.25	29.99	23.75	33.75	27.59	31.22
QSVCNR(full training) 0.5M	32.38	32.02	30.60	37.21	27.90	21.68	32.44	26.05	29.70
QSVCNR(full training) 1.5M	36.25	33.20	32.99	38.69	30.59	24.30	34.14	28.00	31.70
QSVCNR(full training) 3M	38.29	33.71	34.41	39.23	32.45	26.04	35.21	29.26	32.90
QSVCNR(full training) avg	35.64	32.98	32.67	38.38	30.31	24.01	33.93	27.77	31.43

4.2 Main Results

We compare our method QSVCLR with baseline method HNeRV [15] and other implicit methods NeRV [10], ENeRV [11]. Our three-layer network consists of the BL with 0.5M parameters, the first EL with 1M parameters, and the second EL with 1.5M parameters. The network is trained end-to-end for 900 epochs. For other methods [10, 11, 15], we scale channel width to get three models with sizes of 0.5M, 1.5M and 3M. Each is trained for 300 epochs, totaling 900 epochs. The reconstruction performance is compared on the bunny and UVG datasets, as shown in Table 1. Remarkably, our proposed method, QSVCLR, achieved comparable or even superior performance within just two-thirds of the total epochs. After completing the full training, an average improvement of 0.29 PSNR on UVG dataset is achieved, specifically, +0.39 PSNR for 0.5M model, +0.26 PSNR for 1.5M model, and +0.21 PSNR for 3M model.

Notably, our method only requires storing a single 3M model, while other methods require storing three models, totaling 5M parameters. This saves 40% of storage space. The information represented in models with different sizes is similar, but training three models separately fails to utilize their redundancy and correlation. In our approach, each subnetwork shares lower layers and avoids learning the low-frequency information repeatedly. Hence, within the same total number of training epochs, each layer undergoes more extensive training. Additionally, the hierarchical coarse-to-fine framework and residual embedding enhance the ability of the ELs to capture intricate high-frequency details.

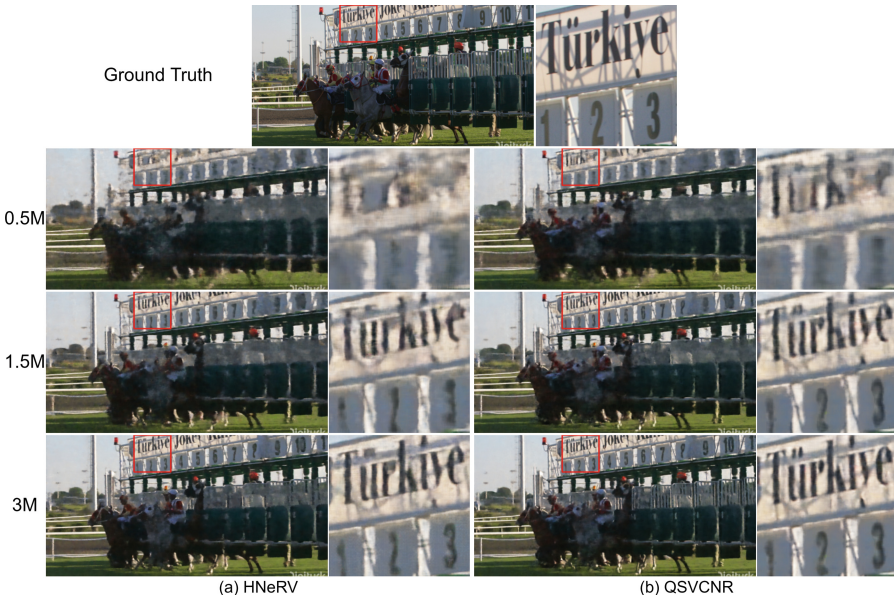


Fig. 5. Video reconstruction visualization. With the same parameters, ours reconstructs videos with better details.

For example, ‘Ready’ in Fig. 5 illustrates the visualization result of the video reconstruction. At the same memory budget, the improved performance in reconstructing details is evident. Specifically, the numbers and letters in the images exhibit enhanced visual quality.

We apply 8-bit quantization to both the model and the embedding for video compression. Finally, lossless entropy coding is applied. To assess the compression performance, we compare our method QSVCNR with other NeRV methods [10, 11, 15], as well as traditional methods like H.264 and HEVC [1, 2], along with the state-of-the-art SVC method, SHVC [18]. As shown in Fig. 6, QSVCNR exhibits superior performance compared to HNeRV [10](+over 0.3 PSNR) and HEVC [2]. It is worth noting that our experiments are all conducted at small bpp, showing its potential in high-rate video compression.

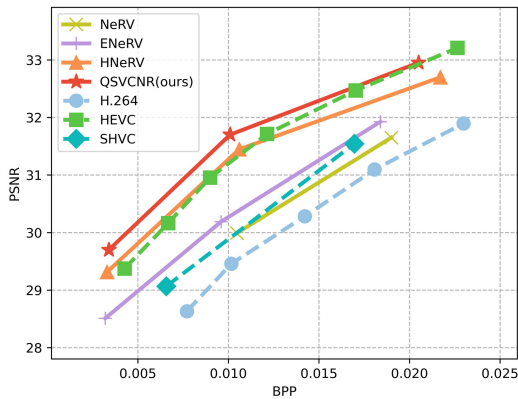


Fig. 6. PSNR vs. BPP on UVG dataset.

4.3 Ablation

Context Encoder. Despite the challenge of modeling long-time contexts with auto-encoder framework, the inclusion of neighbouring frames as short-term context can enhance the reconstruction performance. To demonstrate this, we integrate a context encoder into the baseline method and present the results in Table 2. The incorporation of the context encoder leads to improvements of 0.16 PSNR for the 1.5M model and 0.18 PSNR for the 3M model.

Table 2. Context Encoder ablation on UVG

Model	beauty	bosph	bee	jockey	ready	shake	yach	avg.
HNeRV 1.5M	33.06	33.06	38.65	29.79	23.66	34.06	27.85	31.44
HNeRV 3M	33.58	34.73	38.96	32.04	25.74	34.57	29.26	32.69
HNeRV 1.5M +context	33.11	33.14	38.68	30.24	24.12	34.10	27.96	31.62
HNeRV 3M +context	33.68	34.74	39.23	32.13	25.94	35.01	29.25	32.85

Residual Encoder. The residual encoder makes the embedding of each EL consistent with its representation content and helps ELs better fit high-frequency detailed information. To evaluate its effectiveness, we conduct an experiment where we remove the residual encoder from the ELs and allow all layers to share the content and context embedding as input. To ensure a fair comparison, we increased the channels of the content embedding to three times, maintaining the total embedding size unchanged. The results are presented in Table 3. The model without the residual encoder shows an evident decrease over 0.5 PSNR. Each layer represents content of a different frequency, but sharing a common embedding mixes all frequency components together, resulting in poor reconstruction. It is necessary to introduce residual encoders and embeddings for ELs.

Table 3. Residual Encoder ablation on UVG

Model	beauty	bosph	bee	jockey	ready	shake	yach	avg.
QSVCNR 0.5M	32.02	30.60	37.21	27.90	21.68	32.44	26.05	29.70
QSVCNR 1.5M	33.20	32.99	38.69	30.59	24.30	34.14	28.00	31.70
QSVCNR 3M	33.71	34.41	39.23	32.45	26.04	35.21	29.26	32.90
QSVCNR avg	32.98	32.67	38.38	30.31	24.01	33.93	27.77	31.43
-residual 0.5M	31.63	30.11	36.54	27.27	21.00	32.23	25.67	29.20
-residual 1.5M	32.85	32.46	38.35	29.64	23.39	33.99	27.30	31.15
-residual 3M	33.45	33.80	38.94	31.30	25.02	34.92	28.60	32.29
-residual avg	32.64	32.12	37.94	29.40	23.13	33.71	27.21	30.88

Inter-layer connection in Decoder. We conduct experiments to remove Inter-layer connections and only aggregate the final feature of layers at output stage. We scale the channels of each block, ensuring that every subnetwork maintains the same total of parameters as before. The results are presented in Table 4. By removing Inter-layer sharing, the BL can focus on representing its specific part, resulting in improved performance. However, the ELs tend to exhibit lower performance. Overall, Inter-layer sharing enables ELs to learn higher-frequency details by leveraging the knowledge gained from the lower layers, leading to better overall performance.

Table 4. Inter-layer connection ablation on UVG

Model	beauty	bosph	bee	jockey	ready	shake	yach	avg.
QSVCNR 0.5M	32.02	30.60	37.21	27.90	21.68	32.44	26.05	29.70
QSVCNR 1.5M	33.20	32.99	38.69	30.59	24.30	34.14	28.00	31.70
QSVCNR 3M	33.71	34.41	39.23	32.45	26.04	35.21	29.26	32.90
QSVCNR avg	32.98	32.67	38.38	30.31	24.01	33.93	27.77	31.43
-interlayer 0.5M	32.04	30.63	37.25	27.94	21.73	32.48	26.15	29.74
-interlayer 1.5M	33.17	32.95	38.71	30.52	24.01	34.11	28.01	31.64
-interlayer 3M	33.67	34.39	39.22	32.30	25.90	35.18	29.28	32.84
-interlayer avg	32.95	32.65	38.39	30.25	23.88	33.92	27.81	31.40

Training Strategies. In progressive learning, each layer is individually trained for 300 epochs, and then its weights are frozen before training the next layer. We compare the performance of a 3-layer model trained using progressive training to our end-to-end trained model. The results presented in Table 5 indicate that freezing a portion of the weights harms the representational capability. The end-to-end training of all layers with multi-objective performs better.

Table 5. Training Strategies ablation on UVG

Model	beauty	bosph	bee	jockey	ready	shake	yach	avg.
QSVCNR 0.5M	32.02	30.60	37.21	27.90	21.68	32.44	26.05	29.70
QSVCNR 1.5M	33.20	32.99	38.69	30.59	24.30	34.14	28.00	31.70
QSVCNR 3M	33.71	34.41	39.23	32.45	26.04	35.21	29.26	32.90
QSVCNR avg	32.98	32.67	38.38	30.31	24.01	33.93	27.77	31.43
progress 0.5M	31.78	30.56	36.69	27.35	21.49	32.62	25.68	29.45
progress 1.5M	32.73	32.25	38.10	29.15	23.01	33.90	27.06	30.88
progress 3M	33.22	33.40	38.66	30.49	24.16	34.65	27.94	31.78
progress avg	32.57	32.07	37.81	28.99	22.88	33.72	26.89	30.70

The experiment confirms that the end-to-end training is preferable. However, in some specific cases where a new EL needs to be added to an already trained network, the progressive training method can serve as an alternative that does not require retraining the entire network.

5 Conclusion

This paper proposes a quality scalable video coding method based on neural representation, in which a hierarchical network comprising a BL and several ELs represents the same video with different qualities simultaneously. Without retraining multiple individual networks, these layers can be gradually combined to form subnetworks with varying sizes. The larger subnetworks share the parameters of small ones to save storage space. In addition, the context encoder improves performance by exploiting temporal redundancy, while the residual encoders and Inter-layer connections effectively enhance the ELs' capability to learn high-frequency details. The entire network can be trained end-to-end, allowing for the dynamic balancing of optimization objectives across different layers. As a result, these subnetworks outperform individually trained networks of the same size.

References

1. Wiegand, T., Sullivan, G.J., Bjontegaard, G., Luthra, A.: Overview of the h. 264/avc video coding standard. *IEEE Trans. Circuits Syst. Video Technol.* **13**(7), 560–576 (2003)

2. Sullivan, G.J., Ohm, J.R., Han, W.J., Wiegand, T.: Overview of the high efficiency video coding (HEVC) standard. *IEEE Trans. Circuits Syst. Video Technol.* **22**(12), 1649–1668 (2012)
3. Lu, G., Ouyang, W., Xu, D., Zhang, X., Cai, C., Gao, Z.: DVC: an end-to-end deep video compression framework. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11006–11015 (2019)
4. Li, J., Li, B., Lu, Y.: Deep contextual video compression. *Adv. Neural. Inf. Process. Syst.* **34**, 18114–18125 (2021)
5. Dupont, E., Goliński, A., Alizadeh, M., Teh, Y.W., Doucet, A.: COIN: compression with implicit neural representations. *arXiv preprint arXiv:2103.03123* (2021)
6. Dupont, E., Loya, H., Alizadeh, M., Goliński, A., Teh, Y.W., Doucet, A.: COIN++: neural compression across modalities. *arXiv preprint arXiv:2201.12904* (2022)
7. Strümpfer, Y., Postels, J., Yang, R., Gool, L.V., Tombari, F.: Implicit neural representations for image compression. In: *European Conference on Computer Vision*, pp. 74–91. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-19809-0_5
8. Zhang, Y., van Rozendaal, T., Brehmer, J., Nagel, M., Cohen, T.: Implicit neural video compression. *arXiv preprint arXiv:2112.11312* (2021)
9. Rho, D., Cho, J., Ko, J.H., Park, E.: Neural residual flow fields for efficient video representations. In: *Proceedings of the Asian Conference on Computer Vision*, pp. 3447–3463 (2022)
10. Chen, H., He, B., Wang, H., Ren, Y., Lim, S.N., Shrivastava, A.: NeRV: neural representations for videos. *Adv. Neural. Inf. Process. Syst.* **34**, 21557–21568 (2021)
11. Li, Z., Wang, M., Pi, H., Xu, K., Mei, J., Liu, Y.: E-NeRV: expedite neural video representation with disentangled spatial-temporal context. In: *European Conference on Computer Vision*, pp. 267–284. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-19833-5_16
12. Kim, S., Yu, S., Lee, J., Shin, J.: Scalable neural video representations with learnable positional features. *Adv. Neural. Inf. Process. Syst.* **35**, 12718–12731 (2022)
13. He, B., et al.: Towards scalable neural representation for diverse videos. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6132–6142 (2023)
14. Chen, H., Gwilliam, M., He, B., Lim, S.N., Shrivastava, A.: CNeRV: content-adaptive neural representation for visual data. *arXiv preprint arXiv:2211.10421* (2022)
15. Chen, H., Gwilliam, M., Lim, S.N., Shrivastava, A.: HNeRV: a hybrid neural representation for videos. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10270–10279 (2023)
16. Zhao, Q., Asif, M.S., Ma, Z.: DNeRV: modeling inherent dynamics via difference neural representation for videos. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2031–2040 (2023)
17. Schwarz, H., Marpe, D., Wiegand, T.: Overview of the scalable video coding extension of the h. 264/avc standard. *IEEE Trans. Circuits Syst. Video Technol.* **17**(9), 1103–1120 (2007)
18. Boyce, J.M., Ye, Y., Chen, J., Ramasubramonian, A.K.: Overview of SHVC: scalable extensions of the high efficiency video coding standard. *IEEE Trans. Circuits Syst. Video Technol.* **26**(1), 20–34 (2015)
19. Big buck bunny. <http://bbb3d.renderfarming.net/download.html>
20. Mercat, A., Viitanen, M., Vanne, J.: UVG dataset: 50/120fps 4k sequences for video codec analysis and development. In: *Proceedings of the 11th ACM Multimedia Systems Conference*, pp. 297–302 (2020)

21. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: NeRF: representing scenes as neural radiance fields for view synthesis. *Commun. ACM* **65**(1), 99–106 (2021)
22. Sitzmann, V., Martel, J., Bergman, A., Lindell, D., Wetzstein, G.: Implicit neural representations with periodic activation functions. *Adv. Neural. Inf. Process. Syst.* **33**, 7462–7473 (2020)
23. Chen, Y., Liu, S., Wang, X.: Learning continuous image representation with local implicit image function. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8628–8638 (2021)
24. Cho, J., Nam, S., Rho, D., Ko, J.H., Park, E.: Streamable neural fields. In: *European Conference on Computer Vision*, pp. 595–612. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-20044-1_34
25. Landgraf, Z., Hornung, A.S., Cabral, R.S.: PINs: progressive implicit networks for multi-scale neural representations. *arXiv preprint arXiv:2202.04713* (2022)
26. Liu, Z., Mao, H., Wu, C.Y., Feichtenhofer, C., Darrell, T., Xie, S.: A convnet for the 2020s. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11976–11986 (2022)
27. Rusu, A.A., et al.: Progressive neural networks. *arXiv preprint arXiv:1606.04671* (2016)
28. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014)