UCL

On-the-Sphere Residual Learning for Neural Compression of 360-degree Videos

Author
Eden CHENG

Student Number
22039170

Supervisor
Dr. TONI

September 4, 2023

Final Project Dissertation
Msc in Integrated Machine Learning System
Dept. of Electrical and Electronic Engineering

Abstract

Video compression is of great importance since a large amount of bandwidth will be saved by transmitting compressed video data. Developing deep learning theories and a series of practical applications have made video compression methods based on neural networks (NNs) possible. Instead of developing handcrafted processing modules that are optimized separately in conventional video compression, researchers have employed convolutional neural networks (CNNs) based auto-encoders as key building blocks of end-to-end optimized video compression systems. 360 degree videos consisting of omnidirectional images are a special type of video providing users a fully immersive viewing experience. Applying existing video compression designed for 2D videos will introduce distortions in 2D convolution steps. Recently, researchers have developed a neural compression framework for single omnidirectional images. However, there seems very little preceding research on building neural compression systems for 360-degree videos. In this project, we propose the spherical CNN to perform on-the-sphere learning for frames of 360-degree videos and incorporate the spherical CNN into an existing end-to-end video compression system. Then we compare the results between the original system and our updated version and analyze the effectiveness of our proposed system when encoding and decoding 360-degree videos. The experimental result shows that our model can reconstruct videos with better visual quality. The code is released at `https://github.com/N1ghtstalker2022/MSc_IMLS_Research_Project`.

Contents

1 Introduction and Problem Statement

In today's digital age, over 80 percent of internet traffic is dominated by video content, and this proportion is only anticipated to rise further [1]. Consequently, the need to establish an effective video compression system, capable of delivering superior quality frames within a given bandwidth has become vital. Furthermore, the performance of many video-related computer vision tasks, including video object detection and tracking, relies heavily on the quality of decompressed videos. As such, improvements in video compression efficiency could have wide-ranging benefits, enhancing outcomes across various computer vision tasks. Compression is a process that reduces the file size of a video by removing redundant or unnecessary data while preserving the overall quality to the extent possible. 360-degree video compression is particularly important because of the following reasons. First, video compression is essential to reduce file sizes for transmission and optimize bandwidth usage. The file size for 360 video is usually higher than 2D videos as its application domain demands a higher video resolution ratio. For example, virtual reality (VR) devices provide an immersive experience of video gaming, but it requires 360-degree video data sent by the internet. By compressing 360-degree videos, the efficient transmission of data becomes possible, mitigating buffering and reducing loading times to ensure a seamless VR experience.

Previously, video compression is conducted by traditional compression methods like video codec H.264/AVC whose modules are well designed but optimized separately [2]. With the development of deep learning, it is possible to design a video compression system that could be end-to-end optimized. It has been proved that the compression system designed by Lu et al. already outperforms H.264 when measured by PSNR [3]e . However, their system is designed for 2D videos, not targeted for 360-degree videos. The key drawback of using the previous compression model on 360-degree videos could be that the features of video frames could not be learned properly by 2D convolution blocks in encoders or decoders. Therefore, a new convolution algorithm should be designed to enable the video neural compression to perform reasonably on 360-degree videos.

Thanks to Hierarchical Equal Area isoLatitude Pixelation (HEALPix) provided by [4], an omnipresent image could be partitioned into regions with the same size, which avoids the distortion introduced by 2D planar projection like equirectangular projection (ERP). A spherical convolution algorithm could be designed based on the HEALPix partitioning. Bidgoli et al. define the tools, including spherical convolution used in deep learning models for omnidirectional images using the properties of HEALPix sampling [5]. They applied their innovative spherical convolution to an end-to-end optimized image compression framework introduced by [6] and found it effective for bit rate saving for 360 image compression.

Therefore, it is worth exploring the effect of replacing 2D CNNs to spherical CNNs of video neural compression methods like deep video compression framework (DVC) proposed in [3]. Particularly, we change the residual compression module in DVC to our sphere-dedicated residual compression network, which could perform on-the-sphere learning for residual information obtained by subtracting two 360-degree frames. The on-the-sphere learning is performed by our implemented Spherical CNN. After completing the training of our model, we first test our model by comparing the quality of reconstructed videos (first compressed, then decompressed) between DVC and our proposed model with on-the-sphere residual learning. Then, we perform more evaluations and detailed analysis of our model. Experimental results show that estimating and compressing motion information and especially compressing residual information by using

our proposed model could achieve decent compression performance.

The main features of our proposed model can be summarized as follows:

- Our proposed model could be end-to-end optimized, with important modules including motion estimation, motion compression, motion compensation, and residual compression.
- On-the-sphere learning is conducted in a residual compression module called Sphere-Dedicated Residual Compression where spherical CNN is involved.
- Our spherical CNN is implemented along with HEALPix sampling, which converts the projected frame to the sphere representation. We need a sufficiently high sampling resolution to allow our model to perform at its full potential.

The rest of the report is organized as follows. Section 2 introduces the theoretical background knowledge for this research project. Section 3 provides a detailed literature review about video neural compression techniques and representation learning of 360-degree images, thereby we point out the research gap. We then present in Section 4 the methodologies that are used in the project, with the emphasis on the innovative part. In Section 5, we present experimental results and analysis of our proposed model.

2 Theory and Background

2.1 Image Compression

Before neural network-based image compression methods proposed, the way of image compression is usually by standard compression standards including JPEG, JPEG2000, and BGP whose building blocks are all handcrafted algorithms [7]. By dividing the original image into 8×8 blocks, JPEG is able to apply DCT (Discrete Cosine Transform) to each block, which converts spatial domain data into the frequency domain, expressing the block as a sum of sinusoidal patterns with varying frequency and amplitude. The amplitude of the frequency components is then rounded off according to a quantization table to achieve further compression. After that, DC coefficient in a block is encoded as the difference from the previous DC component, while all AC coefficient are zig-zag ordered. The resulting data is then run through an entropy encoding stage by Huffman encoding or arithmetic encoding to achieve more compression.

With the prevalence of deep learning, neural network-based image compression has been introduced. In [6], balle et al. designed an image neural compression framework based on nonlinear transforms. Unlike traditional image compression methods like JPEG whose components of transform coding (i.e. transform, quantizer, and entropy code) are separately optimized, the compression system proposed by [6] can be end-to-end optimized. In other words, the parameters of the compression system can be learned instead of manual parameter adjustment. Concretely, it takes advantage of two parametric transforms called analysis transform and synthesis transform. Meanwhile, it also introduces three spaces called data space, code space, and perceptual space, where vectors of image intensities exist in dataspace and vectors obtained after applying analysis or synthesis transform exist in code space. Perceptual space is used to assess distortions D between the original image and the decompressed image. In the first place, an input image is transformed by analysis transform (encoding), which consists of 2D convolution, subsampling, and divisive normalization (GDN). Then, the generated representation is quantized, compressed, and sent to the decoder. During the decompressing stage, the quantized data is reinterpreted into the code space and then transformed to the decompressed image representation by the synthesis transform (decoding) consisting of inverse GDN, upsampling, and 2D convolution. The end-to-end framework will optimize the parametric vectors in analysis and synthesis transforms using a weighted sum of the rate and distortion measures $\lambda D + R$ where R representing the bit rate required for compression is computed by lower-bounding the entropy of the discrete probability distribution of the quantized vector.

2.2 Video Compression

Video compression is the process of reducing the amount of data required to represent a digital video. Concretely, performing video compression is the process of reducing inter-frame and intra-frame redundancies, namely temporal and spatial redundancies.

H.264, also known as MPEG-4 Part 10 Advanced Video Coding (AVC), is a widely used video compression standard. It is one of the most commonly used formats for video compression. Due to its widespread support and compatibility, H.264 has become the de facto standard for video compression in many industries. However, newer video compression standards, such as H.265/HEVC (High-Efficiency Video Coding), have emerged to provide even higher compression efficiency and improved video qual-

ity, particularly for 4K and 8K video content. Traditional video compression algorithms such as H.264 and H.265 could achieve highly efficient performance by their predictive coding architecture. However, one disadvantage of them is that the component of the whole algorithm is designed and optimized separately. In another word, they cannot be end-to-end optimized.

In recent years, neural networks have been utilized in video compression methods. One of the most notable ones is the DVC framework proposed by Lu et al. [3]. The DVC framework follows the overall architecture of the traditional video compression codec like H.264 and H.265. Therefore, it is also of great importance to understand the way traditional video compression codec like H.264 and H.265 compress videos.

2.2.1 Traditional Video Compression

Below is a concise description of one classic traditional video coding methods of H.264/AVC Video Coding.

Generally, a video sequence consists of a sequence of frames. A frame of video can be considered as two interleaved fields and referred to as interlaced frame when arriving timings of fields are different, otherwise it is referred to as a progressive frame. H.264 is agnostic with respect to the arriving timing of fields of a frame, so its coding strategy specifies a representation based primarily on geometric concepts (the geometrical difference between two fields).

According to [2], human visual system is more sensitive to brightness than colour information. Taking advantage of this feature, H.264 (as in prior video compression standards) uses YCbCr colour space (Y stands for luma component, Cb and Cr stand for chroma components) to separate original colour representation and a sampling strategy called 4:2:0 sampling to reduce colour information, which could save the bits required to represent a image or frame of a video.

H.264 adopts the preprocessing step that partitions a picture (frame) into fixed-size macroblocks with each covering an area of 16×16 samples of the luma component and 8×8 samples of each of the two chroma components. In H.264, a frame is devided into slices or even slice groups where one slice contains a sequence of macroblocks. Slices are self-contained, meaning that their syntax elements can be extracted from the bitstream, and the samples within the slice area can be accurately decoded without relying on data from other slices. This is possible as long as the reference pictures used by the encoder and decoder are identical. However, when applying the deblocking filter across slice boundaries, some information from other slices may be required. Slices can be coded with different coding types including I slice, P slice, B slice, SP slice and SI slice. The video input is divided into macroblocks, and the assignment of macroblocks to slice groups and slices is determined. Then, each macroblock within each slice is processed accordingly. Efficient parallel processing of macroblocks becomes feasible when the picture contains multiple slices. Fig.1 demonstrates the compression system of H.264. All luma and chroma components of a macroblock are encoded either spatially or temporally.

Either intra-frame prediction or inter-frame prediction is applied to a macroblock in H.264. When using intra-frame prediciton for a macroblock, each block is predicted from spatially neighbouring samples. Also, H.264 involves a constrained intra coding mode that guarantees predictions are only calculated from intra-coded macroblocks. Both luma and chroma components can be intra-frame predicted by either copying or averaging values from prior decoded adjacent blocks. One slight difference is that different sizes of prediction region are applied to luna and chroma components. H.264/AVC establishes

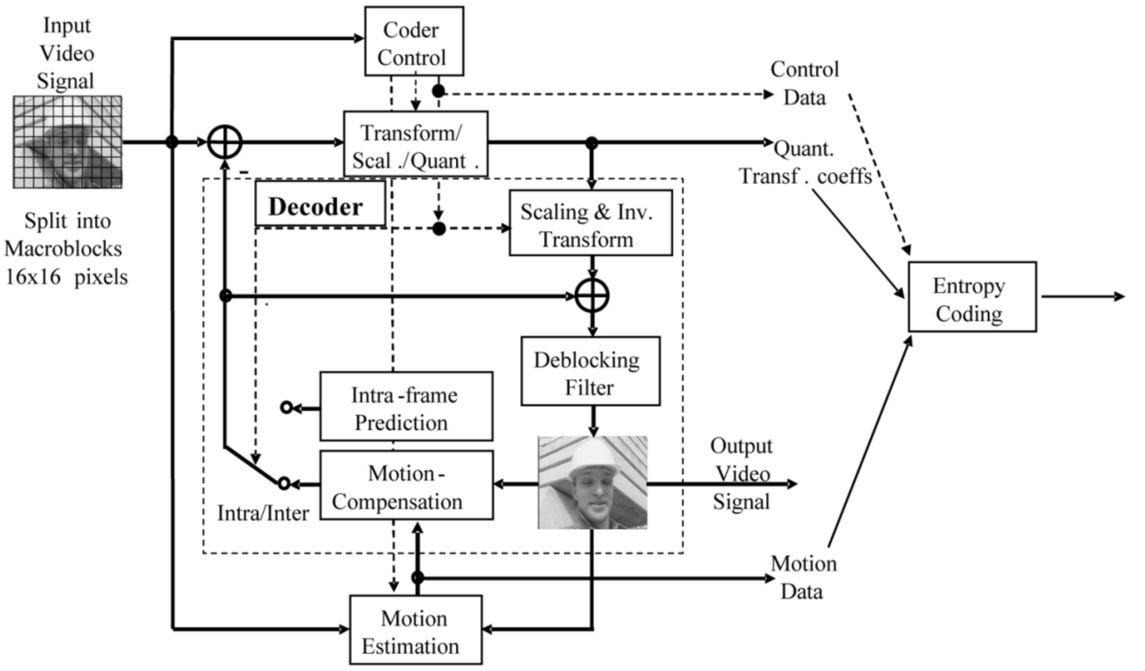


Figure 1: Basic coding structure for H.264/AVC. [2]

a set of segmentation standards for inter-frame prediction. Motion estimation process is performed to estimate the motion between the current frame and the reference frames. It involves comparing corresponding blocks of pixels in the current frame to search areas in the reference frames to find the best match. For each block in the current frame, a motion vector is derived based on the motion estimation results. The motion vector indicates the displacement of the block from its original position in the reference frame. Then The motion vectors and reference frames are used for motion compensation. By applying the motion vectors to the corresponding blocks in the reference frames, a predicted version of the current frame is generated. The prediction is performed by warping the reference frame pixels according to the motion vectors.

The process dealing with residual information at the encoder involves several steps: forward transform, zig-zag scanning, scaling, quantization, and rounding, followed by entropy coding. Conversely, during decoding, the reverse of the encoding process is performed, except for the rounding step.

2.2.2 From Hand-crafted To Neural Video Compression

In the past ten years, there has been a notable rise and rapid growth of deep learning, a set of methods that are being increasingly embraced with the aim of getting closer to achieving artificial intelligence's ultimate objective including the application to video compression. A significant advantage of deep networks is thought to be their ability to handle data at various levels of abstraction and transform it into diverse representations. It's important to note that these representations are not manually crafted; instead, the deep network, along with its processing layers, is learned from extensive data using a general machine learning approach. By doing so, deep learning eliminates the need for manually designed representations, making it particularly valuable for handling inherently unstructured data types like acoustic and visual signals.

Deep neural networks (DNN) like Convolutional Neural Networks (CNN) could be introduced in video compression to remove the redundant content among multiple frames. Recently, trained deep networks have been adopted for optical flow estimation and motion compensation to generate optical flow value and the predicted frame, respectively [8]. Meanwhile, as a specific form of DNN, autoencoders could be used to reduce the bits to be transmitted through the network.

Lu et al. provide a effective and vital reference method for this project [3]. In their work, an end-to-end video compression framework based on neural networks is proposed and extends the core architecture of traditional methods like H.264. Provided sufficient training videos, it has been proved that DNNs could learn the effective way to compress the videos.

Although handcrafted compression algorithm like HEVC performs well on major video forms, it may not be an optimal solution to other video forms such as stereo videos and 360 degree videos. Meanwhile, traditional compression algorithm performs the same operation for any input video sequences, which means it does not make any adjustment to videos with different types (e.g. games or real world). Research conducted in the past suggest that DNN based video compression can learn distinctive features from different video types. The performance of proposed model in [9] even surpass the state of the art video codec H.265.

3 Related Works

3.1 Learning-based Video Compression

Recently, various learning-based video compression algorithms and frameworks have been proposed. Based on a review of the available literature, the first end-to-end deep learning-based video compression (DVC) is proposed by Lu et al.. [3]. They follow the predictive coding architecture of traditional video codec H.264 while changing all key modules to neural networks. All the modules are jointly learned through the rate-distortion loss function, which penalize the bits used for compression and the distortion between the original and the decoded video. However, their model can only learn to reduce inter-frame redundancies. In other words, the model only works for P-frame compression.

Likewise, models proposed in [9, 10, 11, 12, 13, 14, 15, 16, 17] are all learning based P-frame compression frameworks which follow the same model paradigm of the work by [3] but involves modification or improvement of modules of DVC framework. Inspired by the fact that bi-directional coding outperforms sequential coding in conventional video compression algorithms, Yilmaz et al. proposed an end-to-end compression framework using both previous and proceeding frames for motion estimation and compensation [10]. They also redefine the loss function to penalize over groups of pictures (GOP) rather than a single pair of the original and reconstructed frames. In the same vein, Yilmaz et al. formulate a learned hierarchical bi-directional video compression framework [11] based on the work in [10]. In this work, they only use the first and last frame of the group as key frames, which are compressed before conducting bidirectional predictive coding for the rest of the frames in a hierarchical manner. Agustsson et al. argued that using scale-space flow and scale space warping are more effective for compressing videos with disocclusions and fast motions. Their proposed model jointly estimates and encodes the scale-space flow (added scale field), rather than conducting motion estimation and encoding the generated two-channel optical flow [12]. Similarly, Rippel et al. also used scale-space flow in their compression framework, but they made use of the previous reconstructed flow for flow prediction [18]. Cheng et al. revised the traditional digital coding theory and argued that compression framework with high coding efficiency should have good spatial-temporal energy compaction, thereby adding spatial-temporal energy compaction-based penalty into widely used rate distortion loss function [19].

In a follow-up research w.r.t [3] by Hu et al., inter-frame correlations learning was performed in the feature space rather than in pixel space (Concretely, input frames were mapped to feature space by convolutional neural networks before conducting motion estimation), and for frame reconstruction multiple previously reconstructed frames were fused by attention mechanism [14]. Likewise, instead of relying on pixel-space motion, the compression model proposed in End-to-End Learning for Video Frame Compression with Self-Attention could learn deep embeddings of frames (in feature space) and encodes their difference in latent space. At decoder-side, similarly, an attention mechanism is designed to attend to the latent space of frames representing different parts of the previous and current frame so as to form the final predicted current frame [20]. Work in [16] also made use of multiple previously reconstructed frames but they were used for residual refinement before frame reconstruction. It also took advantage of multiple decoded motion vectors for both motion compensation and residual refinement. On the other hand, [15] focused on improving the performance of motion vector (MV) auto-encoder network. They argued the proposed MV encoder network could better handle various

motion patterns by using multi-resolution representations instead of single-resolution representations for both the input flow maps and the output motion features of the motion vector auto-encoder. The compression model proposed by Chen et al. combined the motion compensation network and residual auto-encoder network to one frame reconstruction network. It reduced the number of module needed to perform inter-frame prediction compared with the work in [3, 15, 14]. Using the same GOP training strategy of [10], Lu et.al claimed their proposed compression framework can alleviates error propagation in reconstructed frames and they applied an online encoder updating strategy which could effectively adapt to different video content and achieve better compression performance during inference stage [21]. Sun et al. use GOP as the model input, learn correlations between frames in a group once for all by using Frame-Conv3D in residual encoder and decoder. [22]

There are also plenty of research employing the strategy of combining conventional video compression codec with neural network [23, 24]. He et al. proposed a video compression framework using an overfitted restoration neural network aiming at achieving best result for each video with its own restoration network [23]. However, they did not demonstrate the performance of their compression framework in their paper. Likewise, to enhance interlayer restoration quality in scalable high efficiency video coding (SHVC), He et al. proposed a video compression model combining DNN and SHVC [24]. Chen et al. proposed a pixel-level texure segmentation based AV1 video compression which incorporate a CNN based semantic scene segmentation into conventional codec AV1 by generating pixel-level texture segmentation masks to represent perceptually insignificant regions in a frame and use motion models to reconstruct the texture regions at the decoder to enhance the coding efficiency [17]. The work by Ho et al. used traditional codec VVC as the backbone of their proposed model [13]. They removed colour information and downsample video frames before feeding them into the VVC encoder, and they added DL-based restoration module to upsample and restore colour information of compressed frames.

Except for autoencoder-based neural video compression model and NN with traditional codec hybrid model, generative models including generative adversarial network (GAN) and variational autoencoder (AVE) were proved efficient for video compression in [25, 26, 27]. Besides, by applying intra-frame compression for each frame in the group and modeling the conditional entropy between frames, Liu et al. proposed a simple and efficient video compression framework [28]. Likewise, Li et al. built their deep video compression framework according to the conditional coding paradigm instead of predictive coding [29].

Reducing spatial redundancies (i.e. intra-coding) are also a crucial part of a complete video compression. It is commonly used in keyframes (or I-frames) in video compression schemes like H.264, H.265 (HEVC), and H.266 (VVC) to remove redundant information within the same frame, rather than exploit the similarities between different frames. For example, work in [30] tried extending the HEVC by introducing additional neural network modules to perform intra-prediction more efficiently. However, their model is based on conventional coding schemes and can not be end-to-end optimized.

There are also proposed neural compression frameworks focusing on certain types of videos. Work conducted in [31] and [32] focused on improving compressing efficiency for videos with dynamic textures. [33] focused on talking face video compression. [34] and [35] proposed neural compression framework for stereo and multiview video. [36] proposed learning-based compression for multi-modality videos.

3.2 Representation Learning for Spheres

360-degree videos are a type of video recording where the field of view encompasses a full circle, unlike traditional videos, which have a limited 2D view. The frames of 360-degree videos are omnidirectional images (360-degree images). There are already various proposed solutions to carry out representation learning for omnidirectional images [5, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51]. Some of them used the method to first map omnidirectional to planar representation and then apply existing 2D CNNs, while others developed new convolution filters for on-the-sphere learning.

Before diving into previous representative work of representation learning for omnidirectional images, it is of paramount importance to know the key properties for convolutional filters (i) rotation equivariance (meaning that filtering and rotation commute), (ii) expressive filter (the filter can have any size and any impulse response), and (iii) computational efficiency (complexity grows linearly with the number of pixels, and computation is local) [5].

Works in [37, 38, 39, 40, 41, 42] performed 2D convolutions on planar representations of spherical images after mapping onto 2D planar grid. In their work, Su and Grauman applied to omnidirectional images the popular equirectangular projection (ERP) which samples the sphere with fixed longitude and latitude steps [37]. The distribution of pixels after ERP projection is highly heterogeneous, which means simply performing 2D convolutions on the projected image can not keep the same area over the surface nor describes a constant inter-pixel correlation. The author introduced a variable-size filter whose size is correlated to the latitude of the omnidirectional image, thereby alleviating the issue of heterogeneous pixel distribution. Work conducted in [38] and [39] used the cube map projection to project the omnidirectional image to six planar faces of a cube, and each planar grid was performed 2D CNN. Likewise, [40, 41, 42] tried to obtain a closer distribution to uniform distribution by increasing the number of faces of a polyhedron on which the sphere is projected. However, it was argued by Bidgoli et al. [5] that mapping-based solutions introduce various drawbacks including non-uniform sampling, discontinuities (for [38] and [39]) and local orientation loss.

On the other hand, works in [43, 44, 45] took advantage of graph-based representation learning to perform representation learning for omnidirectional images. For instance, Khasanova and Frossard proposed a graph-based convolution for equirectangular projected images, by modeling the sphere as a graph (i.e. each pixel is a node) and using the inverse of inter-pixel distance as the edge weights between nodes [43]. In contrast to Khasanova and Frossard, the work by [44] and [45] applied quasi-uniform sampling of the sphere and graph-based convolution was defined and performed directly on the sphere. Concretely, Equal Area isoLatitude Pixelation (HEALPix) is used for sampling in [44], whereas Geodesic ICOsahedral Pixelation (GICOPix) is used for sampling before applying the designed convolution kernel approximated by Chebyshev polynomial [45]. Despite good consistency and adaptivity to the topology, Bidgoli et al. argued that the learned filters by the aforementioned graph-based learning lack of expressiveness [5].

Additionally, there are also multi-graph representation learning which uses multiple directed graphs to ensure decent filter expressiveness [51] and spectral Learning [46, 47, 48, 49, 50] which could transform convolution on the sphere to product operation in frequency domain thanks to Fourier transform. However, the complexity of those learning methods are significantly higher than 2D convolution. The spherical CNN proposed in [5] realized high filter consistency, expressiveness, and computational efficiency altogether, and was proved efficient in 360-degree image compression task.

While there has been considerable investigation into feature learning for 360-degree

images and spherical CNN has been successfully applied to 360-degree image compression [5], the application of representation learning on the sphere, particularly, **spherical convolution to 360-degree video compression remains largely unexplored.**

4 Methods

In the previous section, we introduced various video compression models, include but not limited to the conventional codec, NN-based model, and conventional hybrid model. One can benefit from the learning abilities of the neural compression model since there are emerging video types that are not planar. The DL-based video compression frameworks like DVC [3], however, are theoretically not suitable for 360-degree video since involved operations are designed for 2D frames. Just as we treat a 2D planar image as a matrix of pixels, we need to define a suitable transformation method for a 360 degree image, in order to apply spherical convolution. Thanks to the previous work of [4], we could utilize the sphere representation algorithm called HEALPix (the Hierarchical Equal Area isoLatitude Pixelisation of a sphere) to generate a reasonable representation for 360-degree frames. Concretely, we follow the neural compression framework of DVC and modify the 2D CNNs in the residual compression module of DVC [3] to spherical CNNs, along with 2D-sphere and sphere-2D transformations enabled by HEALPix sampling.

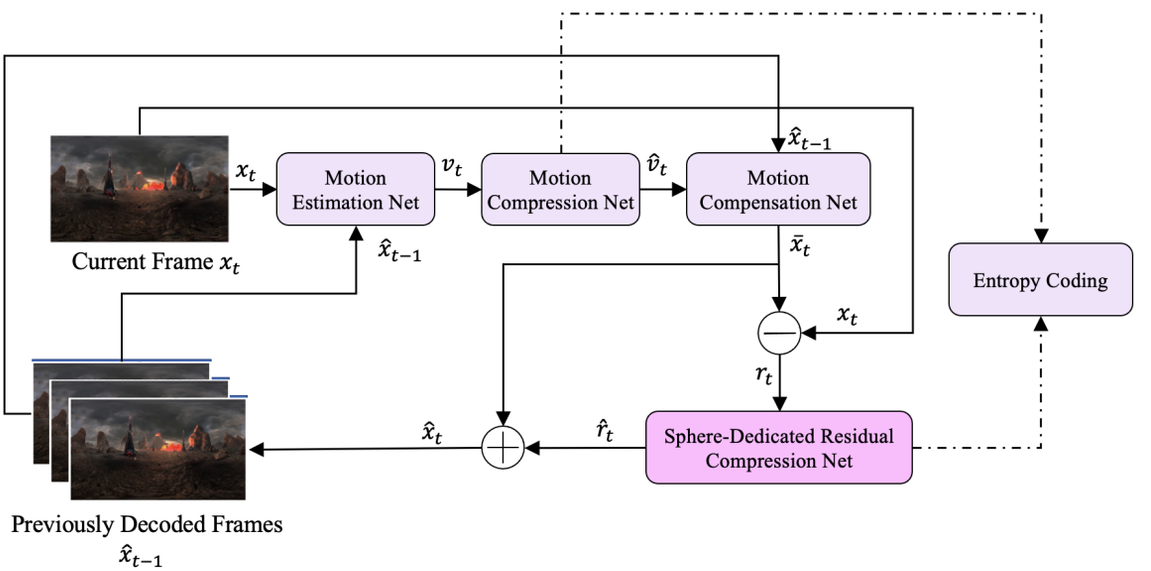


Figure 2: The overall architecture of our proposed model. We used the proposed motion estimation net, motion compression net, motion compression net and entropy coding module in DVC, while changing the original residual auto-encoder to our Sphere-Dedicated Residual Compression Net. On-the-Sphere learning is carried out in the innovated Sphere-Dedicated Residual Compression Net.

Before diving deep into our proposed model, we will first introduce it in a high-level overview. During the compression process, the open flow is computed by the Motion Estimation Net with the current frame and preceding decoded reference frame as the input. Then, an auto-encoder network is used to encode, quantize and decode the optical flow value. Next, to obtain the compensated frame, a pixel-wise motion compensation net is used to compute the compensated frame. Then, the residual information is calculated by the current frame and the predicted frame is transformed into a spherical representation. The latent vectors in the spherical representation are then compressed by the residual encoder net with spherical convolution. Meanwhile, the quantized motion vector and quantized residual information are sent to the decoder side. When decompressing, the

compressed motion vector is decoded and combined with the previous reconstructed frame to serve as the input of the motion compensation net. Then, the generated compensated frame is added up with the residual information recovered by the residual decoder net to the newly reconstructed frame. The overview architecture of our proposed framework is demonstrated in Figure 2.

4.1 Proposed Pipeline

Motion Estimation: Pyramid network [52] is utilized to estimate the motion between the current frame and the previous reconstructed frame (the detail of the Motion Estimation Network is provided in Extra Material). Optical flows (Motion Vectors) could be estimated at different pyramidal scales. Meanwhile, large motions could be better handled with the large receptive field because of the pyramid architecture [53]. We used the settings described in [52] of the 5-level pyramid network where for each level there are five convolutional layers with the filter number 32, 64, 32, 16, 2 and a kernal size of 7×7. The estimated motion vectors v_t are then passed to the Motion Compression network.

Motion Compression: An autoencoder network introduced in [6] is used to compress the estimated motion v_t . The motion compression network consists of a CNN-style encoder, an entropy bottleneck, and a CNN-style decoder (as shown in the Extra Material). Concretely, when encoding, the estimated motion vector v_t is passed to the encoder network and quantized by the entropy bottleneck. When decoding, the quantized representation $\hat{m}_t$ is first dequantized by the entropy bottleneck and then decoded by the decoder network. The reconstructed motion information is denoted in $\hat{v}_t$. For the motion compression network, the filter dimensions are configured to be 3×3, with 128 filters utilized for all layers except the final one in the decoder. The last layer in the decoder uses only 2 filters to rebuild the 2-channel motion vector.

Motion Compensation: The motion compensation network is aimed at compensating previously reconstructed frame $\hat{x}_{t-1}$ with the reconstructed motion vector $\hat{v}_t$ so that the compensated result $\bar{x}_t$ is as equivalent as the current frame. It is argued by Lu et al. that by using a CNN-based pixel-wise motion compensation approach could avoid blockness artifact in the traditional block-based motion compensation method [3]. In our motion compensation network, the previously reconstructed frame $\hat{x}_{t-1}$ and $\hat{v}_t$ is first warped to remove artifacts. Then, the warped frame is concatenated with $\hat{x}_{t-1}$ and $\hat{v}_t$ as the input of the CNN, which learns to match the warped frame with the original frame x_t.

Residual Compression: In this step, the residual information r_t is transformed into a spherical representation and then encoded by spherical CNNs. The recovered residual information $\hat{r}_t$ is obtained by the decoder with spherical CNNs. The whole module is named as Sphere-Dedicated Residual Compression which will be illustrated in detail in **Section 4.2**

Entropy Coding: Compressed motion vectors and residual embeddings exist in the form of bit stream. Since the bit rate is part of the optimization function of our model, we need to measure it correctly. By using Entropy Bottleneck proposed by [6], we can estimate the probability distribution of motion vectors and residual embeddings in the latent space, and then calculate the entropy which approximates the bit rate.

4.2 Sphere-Dedicated Residual Compression

Residual compression is a crucial step for frame reconstruction. The residual information r_t is computed by subtracting compensated frame $\bar{x}_t$ from original frame x_t. In the previous 2D video compression framework [3, 21, 36], residual information is directly compressed by an autoencoder network with 2D convolution. This autoencoder automatically learn to better compress the residue so that less bit is needed and the distortion between the original frame x_t and the restored residue $\hat{r}_t$ plus compensated frame $\bar{x}_t$ is as small as possible. However, if the residue is computed from 360-degree frames (which is our case), performing 2D convolution directly on a projected 360-degree image will induce distortions [5]. In other words, using 2D convolution on 360-degree image can not guarantee the kernel keeps the same surface. Therefore, to build an efficient residual compression towards 360-degree videos, we introduce our Sphere-Dedicated Residual Compression Neural Network with spherical learning. The diagram of the Sphere-Dedicated Residual Compression net is demonstrated in Figure 3. Being transformed to sphere representation, we could conduct residual learning for r_t on the sphere by our designed spherical CNN. Then, the decoder will dequantize the latent representation of residual information and recover it to the decompressed one. Finally, we need to transform the obtained residual map to the 2D plane so that it can be added with compensated frame $\bar{x}_t$ to reconstruct the compressed frame.

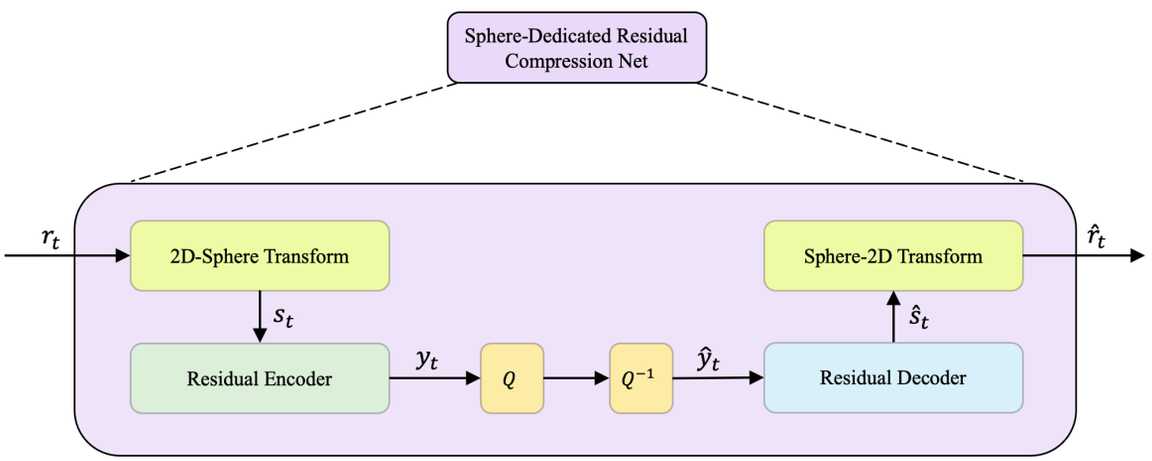
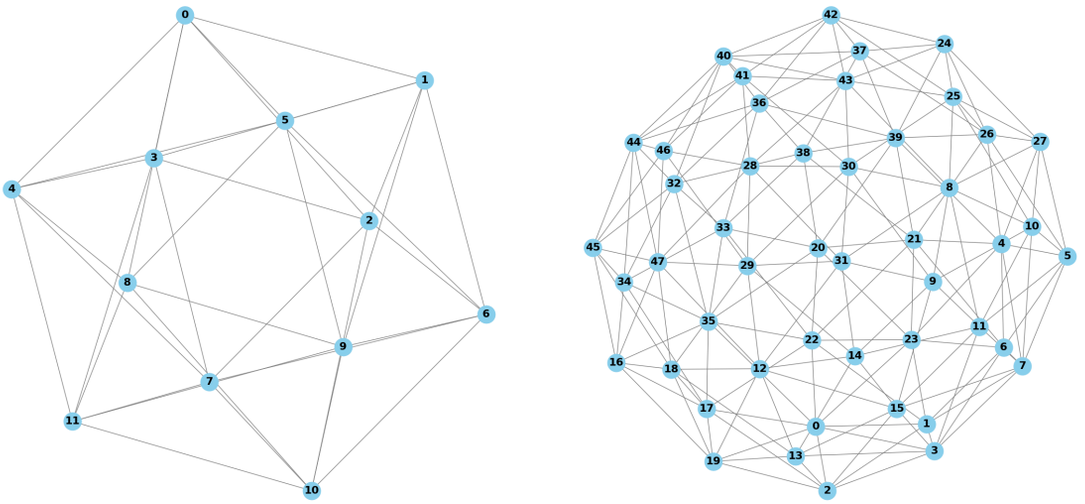


Figure 3: The Sphere-Dedicated Residual Compression Network.

4.2.1 Transform to Sphere Representation

Before conducting the spherical learning process, the equirectangular frames are sampled according to a sampling approach named HEALPix [4]. In HEALPix, the tessellation process first divides the spherical surface into 12 equal-area regions and could further divide each region into 2 by 2 sub-regions recursively. After the representation construction, each pixel has eight neighbours except for 24 pixels with seven neighbours (for any resolution higher than the base resolution). For high-resolution images, the implication of exceptional points could be negligible. Therefore, it ensures that for almost every pixel, there is the same number of neighbours. With the fact that in HEALPix, the orientation of neighboring points with the central point is almost fixed all over the sphere, one could say the representation meets the prerequisite to be applied by an expressive filter. Meanwhile, the distance between a specific neighbor and the center pixel

is almost constant all over the sphere, and the relative orientation of a specific neighbor concerning the center pixel is also constant over the sphere. It guarantees the consistency of the shape of the filter. A proper resolution of HEALPix sampling is needed to generate effective representations on the sphere. Fig. 4 shows the data structure of a sampled frame represented by $12 \times 2 \times 2$ pixels. With HEALPix sampling, the residue could be transformed from 2D representation to on-the-sphere format.



(a) Base sampling resolution *(b) Sampling resolution one*

Figure 4: A demonstration of the result data structure after HEALPix Sampling. The Healpix tessellation is parametrized by resolution, and the resulting total number of pixels is equal to $12 \times 2^{(resolution)} \times 2^{(resolution)}$*. Note that this is not the on-the-sphere numbering for pixels. This figure is used to show inter-pixel relations, i.e. neighbourhood structures, which can be treated as an undirected graph.*

4.2.2 Spherical CNN

With the effective and reasonable representation for the 360-degree frame or video (since one video could be seen as a sequence of frames), we could design an efficient spherical convolution filter to conduct representation learning.

Convolution

For 2D images, convolution operation on one pixel requires values of its neighbours whose range is determined by the size of 2d convolution filters. Likewise, for 360-degree images, pixel values of the pixel itself and neighbours are required to perform spherical convolution. By HEALPix sampling, we can obtain neighborhood structures for all pixels on the sphere, which makes the implementation of spherical convolution a feasible work.

For each vertex i, we use $N_i(k)$ to denote the index of k-th neighbour ($k = 1, 2, ..., 8$ representing eight directions). we define x_i the input feature at vertex i and θ_k the learnable weight of the filter corresponding to the k-th neighbour. The spherical convolution filter could be formularized as:

$$o_i = \theta_0 \cdot x_i + \sum_{k=1}^{8} \theta_k \cdot x_{N_i(k)} + b \tag{1}$$

where θ_0 denotes the weight of the pixel itself, b denotes the bias term. Moreover, to handle the 24 exceptions of HEALPix pixels that do not have 8 neighbors, we introduce a mask parameter w for each weight:

$$w_{N_i(k)} = \begin{cases} 0, & \text{if the neighbor } N_i(k) \text{ is missing} \\ 1, & \text{otherwise} \end{cases} \tag{2}$$

Hence, the expression for convolution filter becomes:

$$o_i = \theta_0 \cdot x_i + \sum_{k=1}^{8} \theta_k \cdot x_{N_i(k)} \cdot w_{N_i(k)} + b \tag{3}$$

The convolution operation is demonstrated in Fig.5.

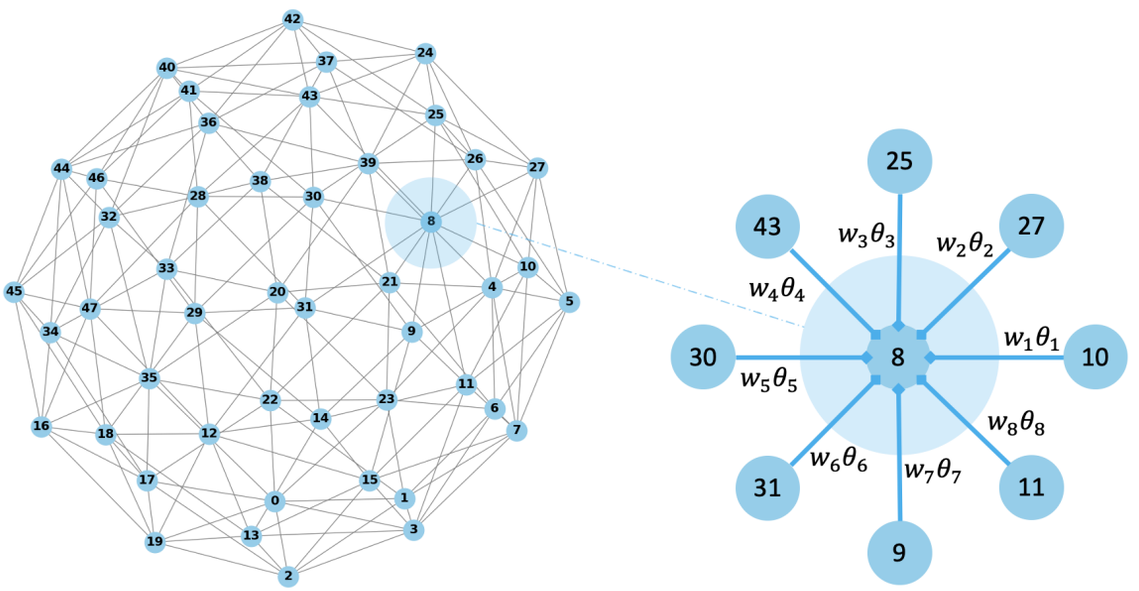


Figure 5: The conceptual diagram of spherical convolution operation. The convolution operation will be performed for every node in the graph shown on the left. For instance, the diagram on the right demonstrates the convolution for node with index 8.

Stride

Stride is an indispensable part in a complete convolution step, since how the convolution kernels move over the input is controlled by the stride. It can be used to perform downsample operation along with convolution. Stride operation in 2D convolution can not be used in spherical convolution, since the pixels of Healpix sampled image is not arranged in simple 2d arrays. We need to redefine stride operation that is align with the Healpix sampled image.

Apart from its graph-based neighborhood structure, the nested number scheme of the HEALPix is another important feature that is closely related to our proposed stride definition [4]. The relation of pixels and their children pixels is represented in a tree structure, where each node has four children pixels since every father pixel node is split into 2×2 children pixels to obtain finer pixelization. The numbering of 12 pixels in the base resolution is by decimal number $0, 1, 2, \cdots, 11$, while the numbering of a child pixel is computed by concatenating the parent index (in binary representation) with two additional bits. For example, pixel index 1 (01, in binary) in the base resolution has four

children pixels with indices 4,5,6,7 (0100,0101,0110,0111 in binary notation) in the first resolution. Children pixels in the higher resolution representation occupies the position of the father pixel in the lower resolution representation.

With the support of HEALPix nested numbering scheme demonstrated in Fig. 6, we define a stride of $n \times n$ on the sphere (where $n = 2^r, r = 0,1,2,\ldots$). n muse be a power of 2 since in HEALPix subdivision scheme each parent is divided into 4 children.

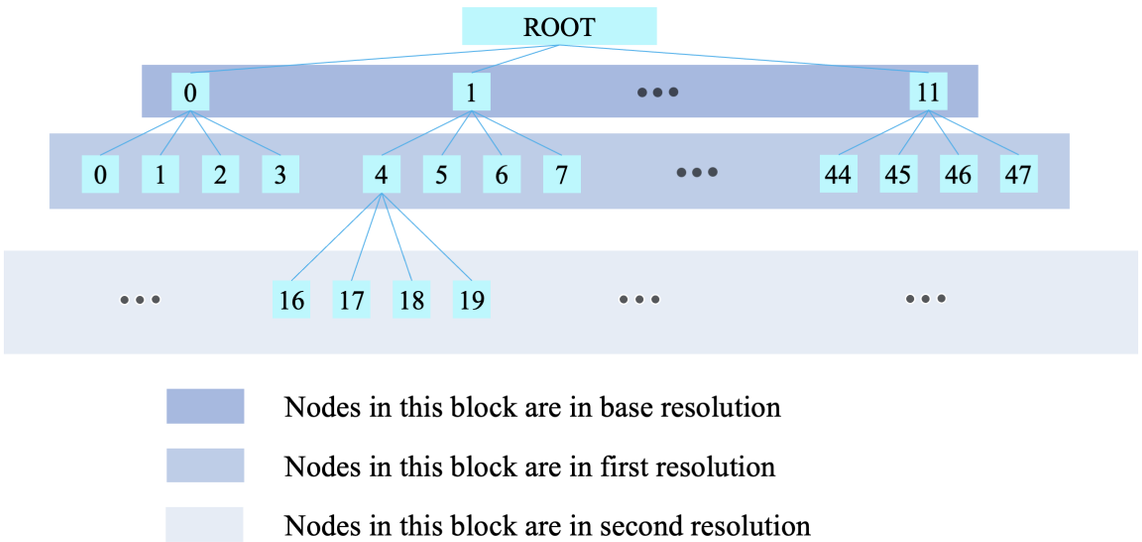


Figure 6: The demonstration of HEALPix nested numbering scheme. In base resolution setting, there are 12 *nodes in total. In another word, the omnipresent image (frame) is sampled to* 12 *pixels in this case. In the first resolution setting ($r = 1$), the number of pixels is $12 \times 2^1 \times 2^1 = 48$. In the second resolution setting ($r = 2$), the number of pixels is $12 \times 2^2 \times 2^2 = 192$. The pixels with indices* 4,5,6,7 *in the first resolution setting is the children of the pixel with index* 1 *in the base resolution. The pixels with indices* 16,17,18,19 *in the second resolution setting is the children of the pixel with index* 4 *in the first resolution, etc..*

4.2.3 Residual Encoder and Decoder Network

With the well-designed spherical convolution, we could implement a residual compression model 360 degree videos. Concretely, we will reuse the residual compression architecture in [53]. However, the neural networks involving 2D convolution operations will be replace by spherical substitutions to conduct representation learning more effectively. Also the GDN is updated to perform correctly for on-the-sphere learning. The architecture of residual encode and decoder network is shown in Figure 7. The residual embedding s_t is fed into a series of convolutions and nonlinear transforms. Concretely, the encoder consists of four spherical convolution layers with $\times 2$ downsampling, and first three layers also perform GDN to introduce non-linearity. The structure of the decoder is similar to the encoder, except that convolutional layers of the decoder performs $\times 2$ upsampling. The latent residual representation y_t is quantized to $\hat{y}_t$, and when decoding $\hat{y}_t$ will be dequantized and passed to the residual decoder. The reconstructed residual embedding $\hat{s}_t$ is the output of the residual decoder. $\hat{y}_t$ will also used by entropy coding module to approximate the bit rate introduced by residual compression.

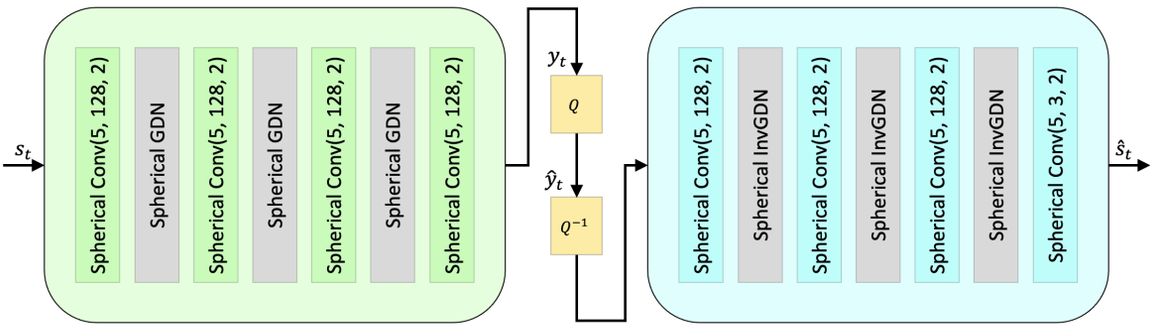


Figure 7: The Residual Encoder and Decoder Network. Conv(5,128,2) represents the convoluation operation with the kernel size of 5×5*, the output channel of 128 and the stride of 2.*

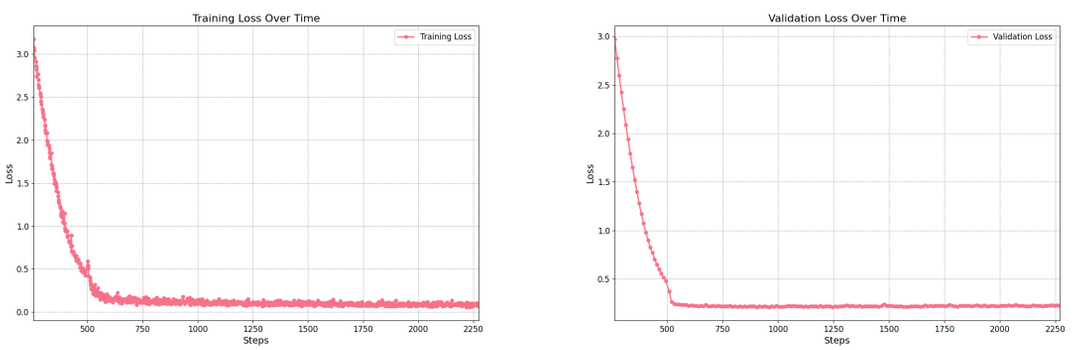
5 Experimental Results and Analysis

5.1 Training Setup

This section first introduces the video dataset used for training and testing. Then, a detailed implementation of the model and evaluation metric are explained. After that, the result of model training and evaluation is demonstrated.

Dataset: Our proposed 360-degree video compression model is trained and tested on the VR-HM48 dataset [54] which contains 48 panoramic videos (the most common way 360-degree videos are stored). We segment original videos into 7-frame video sequences and use 10,000 segmented videos as our dataset. We further split the whole dataset into training, validation, and testing set with a ratio of $8:1:1$. Before training, we downscale the video frames to the size of 64×128 to save CPU memory usage and speed up data loading procedure.

Model Implementation: The batch size for training and validation is set to 16 to better take advantage of parallel computing of GPUs. The initial learning rate is set to 10^{-4} with Adam optimizer [55]. The learning rate is divided by 10 until 10^{-6} when the loss becomes stable. We train four models with $\lambda = 256, 512, 1024, 2048$, respectively, where λ is used to balance the penalties of bit rate and distortion. Each model is trained 50 epochs using the same dataset. The whole system is implemented based on PyTorch Lightning. Training of the model is on one Nvidia Tesla V100 GPU with 32Gb memory. The learning curve with $\lambda = 2048$ is shown in Figure 8.



(a) Reconstructed Frame *(b) Current Frame*

Figure 8: The training and validation loss of proposed model with $\lambda = 2048$

Training Details: The proposed model is trained in a progressive manner. First, we train the motion estimation net. Since only the motion estimation net is trained, we reconstruct the frame by warping the last reconstructed frame with the estimated motion vector v_t. The loss function is formularized as

$$\mathcal{L}_{ME} = D(x_t, W(\hat{x}_{t-1}, v_t)), \tag{4}$$

where D denotes the distortion (mean square error) between original frame x_t and reconstructed frame, W denotes the warping operation. After the convergence of the motion estimation net, and motion compression net is included in the training. In this stage, the frame is reconstructed by warping the last reconstructed frame with the decoded estimated motion vector $\hat{v}_t$. The loss function becomes

$$\mathcal{L}_{MCP} = D(x_t, W(\hat{x}_{t-1}, \hat{v}_t)) + R(\hat{m}_t), \tag{5}$$

where R stands for the estimated bit rate by entropy coding module in our framework. Next, the motion compensation net is trained. In this stage, we use the compensated frame as the reconstructed frame. The motion compensation net is trained by

$$\mathcal{L}_{MCO} = D(x_t, \bar{x}_t) + R(\hat{m}_t). \tag{6}$$

After $\mathcal{L}_{MCO}$ is converged, our sphere-dedicated residual compression net is included for training, using the loss defined by

$$\mathcal{L} = D(x_t, \hat{x}_t) + R(\hat{m}_t) + R(\hat{y}_t), \tag{7}$$

where the bit rate for entropy coded residual is included in the loss $\mathcal{L}$. The objective of our video compression is to minimize the number of bits used for encoded video while keeping the distortion between the original frame and the reconstructed frame as small as possible. Only our Sphere-Dedicated Residual Compression network is trained using a group of pictures (GOP) size 7. The motion estimation, motion compression, and motion compensation networks are trained using GOP size 2. It is noteworthy that quantization does not really happen in training stage since quantization is not a differential operation. To make the whole model trainable in an end-to-end manner, the quantization operation is replaced by adding uniform noise.

Evaluation Metrics: Bits per pixel (Bpp) is used to measure the required bits for encoding each pixel in the current frame. The distortion of the reconstructed frames is measured by Peak Signal-to-Noise Ratio (PSNR) and Multi-Scale Structural Similarity (MS-SSIM).

5.2 Experimental Results and Analysis

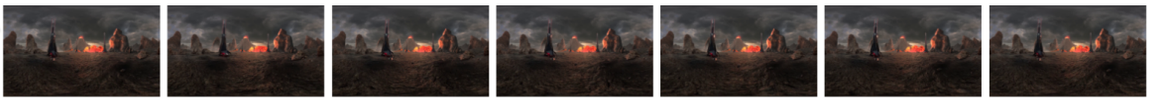
In this section, we first show the performance of our proposed 360-degree video compression framework both visually and qualitatively. Next, we analyze some key building blocks of our model. Last but not least, we discuss the advantages of our proposed model over DVC when conducting 360-degree video compression. We test and evaluate our proposed model with videos of which the resolution is 256×448.

5.2.1 Overall Model Performance

The overall performance of our proposed 360-degree video compression framework is shown in Figure 9, compared with the performance of DVC model by [3]. The quality of reconstructed frames using original DVC without HEALPix sampling or spherical CNN is visually slightly lower than those using our proposed compression framework. As we can see from subfigure (d),(e), and (f), more distortion is introduced by DVC, especially in high-latitude regions. Since the difference between our model and DVC is that we replace 2D CNN learning on the projected ERP map to spherical CNN learning on the sphere, we can intuitively conclude that by learning on the sphere the features of the residual map of 360-degree frames could be learned more efficiently than by learning on the projected plane with existing 2D CNN.

5.2.2 Analysis

Detailed analysis of key model components is carried out in this section, especially for our innovative residual compression network. All the images used under this section are generated when evaluating the model trained by hyperparameter $\lambda = 2048$.



(a) Original Video Frames

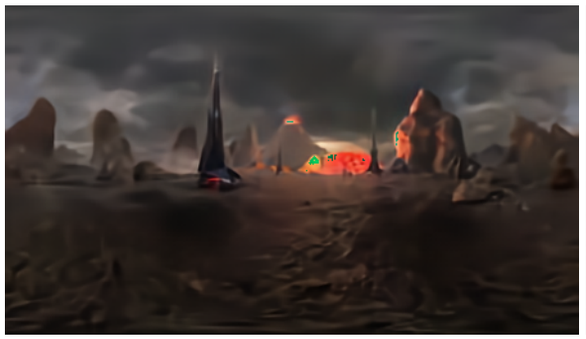
(b) Reconstructed Video Frames By our Proposed Model

(c) Reconstructed Video Frames by original DVC Model

(d) *(e)* *(f)*

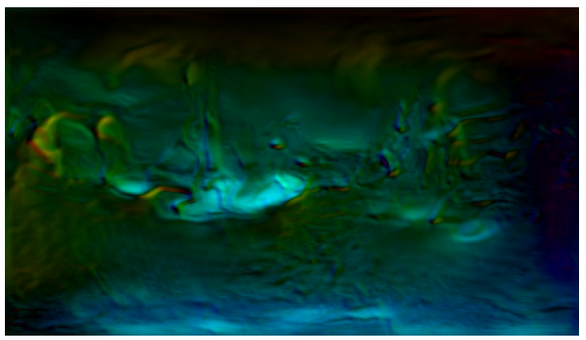
Figure 9: In this figure, (a) represents original frames of one 7-frame video clip. (b) represents the reconstructed video frames after compression and decompression of (a) using our proposed model with spherical CNN. (c) represents the reconstructed video frames after compression and decompression of (a) using DVC in [3]. (d), (e), (f) represents the second frame of original frames, the second frame of Reconstructed Video Frames by our Model, and the second frame of Reconstructed Video Frames by DVC, respectively.

Motion Estimation and Compression In Figure 10, we visualize the optical flow estimated by motion estimation net and reconstructed optical flow after motion compression net. It could be observed that our trained motion estimation net is able to generate decent motion vectors. Intuitively, we can also see from the optical flow map that the motion compression net can successfully reconstruct the optical flow but the reconstructed optical flow is not as detailed as the original one. As it is demonstrated in the probability distribution graph in Figure 10 (e) and (f), the optical flow and reconstructed one share a very similar magnitude distribution, which means they share similar overall motion characteristics. In our video compression framework, motion vectors can be encoded to save bits and later decoded to participate in motion compensation. Although the encoding and decoding process is lossy, important motion information can still be recovered without introducing substantial errors.

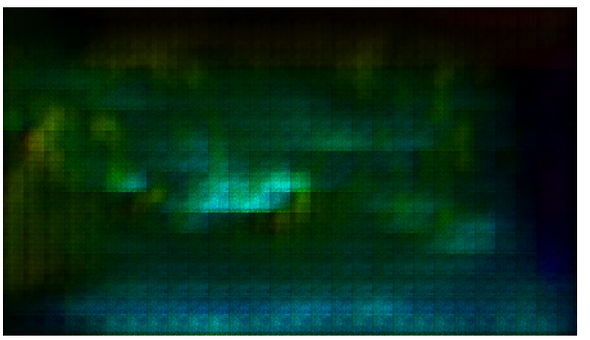


(a) Reconstructed I-Frame

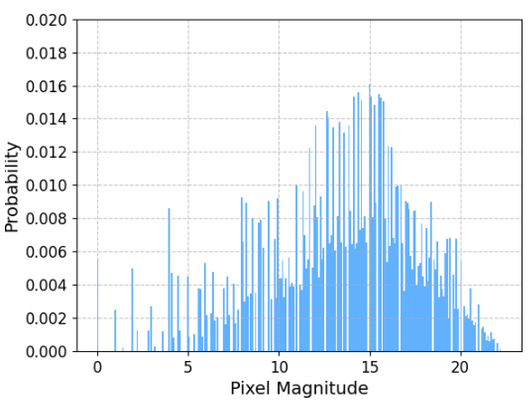
(b) Current Frame



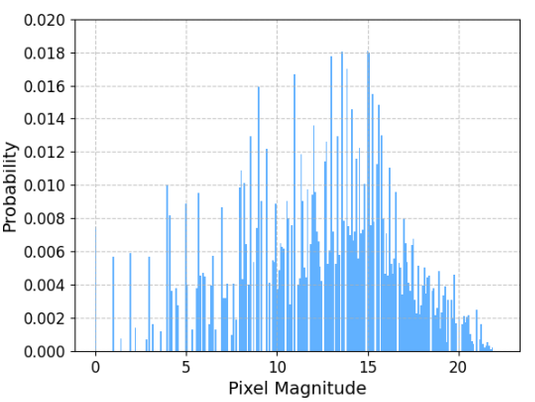
(c) Optical Flow



(d) Reconstructed Optical Flow



(e) Magnitude Distribution of the Optical Flow

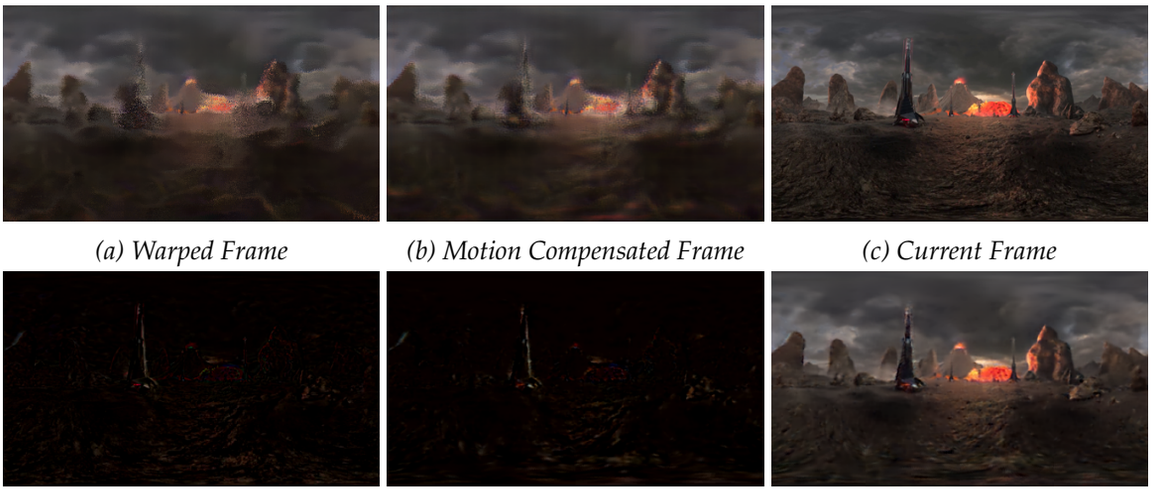


(f) Magnitude Distribution of the Reconstructed Optical Flow

Figure 10: Flow Visualization and Statistical Analysis.

Motion Compensation: As it is shown in Figure 11, the reconstructed frame (b) by applying motion compensation network is visually more similar to the original frame, compared with the frame reconstructed by plain image warping (a). It indicates that using CNN-based motion compensation network can alleviate the blockness artifact introduced by dense image warping and privode more accurate temporal motion information for the subsequent learning steps.

Residual Compression: The most noteworthy part is the Sphere-Dedicated Residual Compression, since it is where corresponding operation is conducted for 360 degree videos (frames). We provide data visualization for our residual compression network trained by hyperparameter $\lambda = 2048$ and HEALPix sampling resolution $r = 7$. As it is shown in Figure 11 (d) and (e), the recovered residual map is visually close to the origi-



(a) Warped Frame *(b) Motion Compensated Frame* *(c) Current Frame*

(d) Residual Information *(e) Reconstructed Residual Information* *(f) Reconstructed Frame*

Figure 11: A comparison between the result of using plain image warping and using motion compensation network is made by images (a), (b), and (c) in the first line. Image (d) and (e) shows the original residual and reconstructed residual by our residual compression network, respectively. (f) is the reconstructed frame by adding motion compensated frame and reconstructed residual map.

nal residual map. Meanwhile, the quality of reconstructed frame after residual learning shown in (f) is much better than motion compensated frame (b) as shown in Figure 11. The corresponding PSNR of reconstructed frame (f) (32.98dB) is significantly higher than motion compensated frame (b) (28.01dB), indicating a substantial improvement in the quality of the reconstructed image.

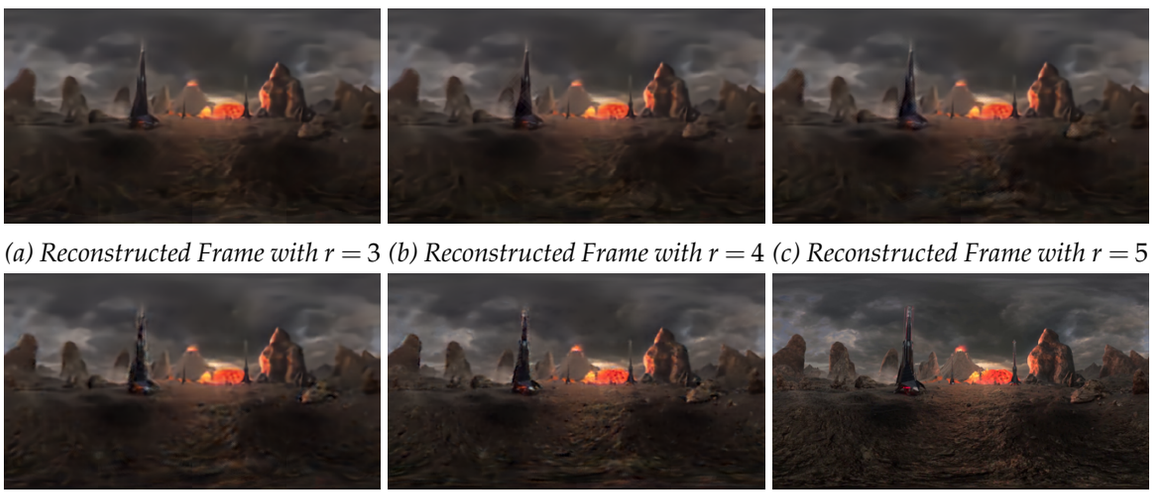
Table 1: Reconstruction performance with different HEALPix sampling resolution settings measured by PSNR, MS-SSMI, and Bpp. NPix represents the number of pixels on-the-sphere.

Resolution	NPix	PSNR	MS-SSIM	Bpp
3	768	27.5639	0.8193	0.0195
4	3,072	28.0133	0.8319	0.0217
5	12,288	29.0025	0.8756	0.0311
6	49,152	30.6967	0.9235	0.0894
7	196,608	32.6750	0.9599	0.3110

To investigate the influence of the HEALPix sampling resolution r on the performance of the trained model (with $\lambda = 2048$), we conducted a comparative experiment. Specifically, we varied the value of r while keeping all other components the same, and evaluated the performance of our proposed model using metrics PSNR, MS-SSIM, and Bpp. As it is indicated from Table 1, the higher the sampling resolution, the higher Bpp, which means more bits are required to store the compressed motion and residual information (theoretically, only the Bpp of residual information will vary as r changes). Meanwhile, PSNR and MS-SSIM becomes lower when r rises from 3 to 7, representing the drop of distortion between original video frames and reconstructed ones.

To visualize the quality of reconstructed videos with different resolution settings, we choose the second frame from reconstructed frames. As it is shown in Figure 12, the

quality of reconstructed frame drops dramatically between (d) and (e). As the number of pixels for one original frame is $256 \times 448 = 114{,}688$ which is much higher than the number of pixel (49152)after sampling with $r = 6$, a considerable amount of residual information is lost during HEALPix sampling. Despite the effectiveness of our spherical CNN to learn on the sphere, the lost residual information caused by 2D-Sphere transform (sampling) can not be recovered in the rest process of residual learning. On the other hand, however, the bits required to store the residual information surge when increasing the resolution from 6 to 7, as measured with Bpp (0.0894 for $r = 6$ and 0.3110 for $r = 7$, respectively).



(a) Reconstructed Frame with $r = 3$ *(b) Reconstructed Frame with $r = 4$* *(c) Reconstructed Frame with $r = 5$*

(d) Reconstructed Frame with $r = 6$ *(e) Reconstructed Frame with $r = 7$* *(f) Original Frame*

Figure 12: Reconstruction results with different resolution settings ($r = 3,4,5,6,7$), compared with the original frame.

From it we can conclude that it is important to choose a suitable sampling resolution to balance the bit rate and quality of reconstructed video when using our model for 360 degree video compression.

6 Conclusion and Future Work

This project proposes and implements a neural compression framework for 360-degree videos. Instead of performing residual learning on a 2d projection map, we transform the residual information from the 2D plane to the sphere representation by HEALPix sampling. We designed and implemented spherical CNN, enabling residual learning on-the-sphere. The experiments show that our compression framework featured by sphere-dedicated residual compression could carry out representation learning more effectively for 360 videos. This work provides a promising track for applying a deep neural network for 360 video compression.

In this work, however, only CNNs in residual learning are replaced by our designed spherical CNN. Since the representation learning in former steps, including motion estimation, motion compression, and motion compensation is still conducted with 2D CNN, distortions that are not negligible might have been introduced before residual learning. Therefore, this work could be improved by replacing all 2D CNNs with spherical ones, which would likely improve the quality of reconstructed frames while saving more bits used for video storing or transmission. Moreover, we can use area-weighted spherical PSNR (AW-SPSNR) [56] instead of PSNR to more accurately evaluate the quality of reconstructed videos. Last but not least, in this work, 8,000 7-frame video clips are used to train the model, and each frame is resized to 64×128. The computational resources available for this study were limited, constraining the volume and resolution of video data that could be used for training the model. With more computational resources, future work could aim to train the model on a more extensive dataset with higher-resolution videos. This is likely to improve the model's robustness and accuracy, allowing it to generalize better to a wider range of 360-degree video content.

References

[1] C. V. Networking, "Cisco global cloud index: Forecast and methodology, 2015-2020. white paper," *Cisco Public, San Jose*, p. 2016, 2016.

[2] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the h. 264/avc video coding standard," *IEEE Transactions on circuits and systems for video technology*, vol. 13, no. 7, pp. 560–576, 2003.

[3] G. Lu, W. Ouyang, D. Xu, X. Zhang, C. Cai, and Z. Gao, "Dvc: An end-to-end deep video compression framework," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 006–11 015.

[4] K. M. Gorski, E. Hivon, A. J. Banday, B. D. Wandelt, F. K. Hansen, M. Reinecke, and M. Bartelmann, "Healpix: A framework for high-resolution discretization and fast analysis of data distributed on the sphere," *The Astrophysical Journal*, vol. 622, no. 2, p. 759, 2005.

[5] N. M. Bidgoli, R. G. d. A. Azevedo, T. Maugey, A. Roumy, and P. Frossard, "Oslo: On-the-sphere learning for omnidirectional images and its application to 360-degree image compression," *IEEE Transactions on Image Processing*, vol. 31, pp. 5813–5827, 2022.

[6] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," *arXiv preprint arXiv:1611.01704*, 2016.

[7] G. K. Wallace, "The jpeg still picture compression standard," *IEEE transactions on consumer electronics*, vol. 38, no. 1, pp. xviii–xxxiv, 1992.

[8] D. Liu, Y. Li, J. Lin, H. Li, and F. Wu, "Deep learning-based video coding: A review and a case study," *ACM Computing Surveys (CSUR)*, vol. 53, no. 1, pp. 1–35, 2020.

[9] H. Liu, M. Lu, Z. Chen, X. Cao, Z. Ma, and Y. Wang, "End-to-end neural video coding using a compound spatiotemporal representation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 8, pp. 5650–5662, 2022.

[10] M. A. Yilmaz and A. M. Tekalp, "End-to-end rate-distortion optimization for bi-directional learned video compression," in *2020 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2020, pp. 1311–1315.

[11] M. A. Yılmaz and A. M. Tekalp, "End-to-end rate-distortion optimized learned hierarchical bi-directional video compression," *IEEE Transactions on Image Processing*, vol. 31, pp. 974–983, 2021.

[12] E. Agustsson, D. Minnen, N. Johnston, J. Balle, S. J. Hwang, and G. Toderici, "Scale-space flow for end-to-end optimized video compression," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 8503–8512.

[13] M. M. Ho, J. Zhou, G. He, M. Li, and L. Li, "Sr-cl-dmc: P-frame coding with super-resolution, color learning, and deep motion compensation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2020, pp. 124–125.

[14] Z. Hu, G. Lu, and D. Xu, "Fvc: A new framework towards deep video compression in feature space," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 1502–1511.

[15] Z. Hu, Z. Chen, D. Xu, G. Lu, W. Ouyang, and S. Gu, "Improving deep video compression by resolution-adaptive flow coding," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*. Springer, 2020, pp. 193–209.

[16] J. Lin, D. Liu, H. Li, and F. Wu, "M-lvc: Multiple frames prediction for learned video compression," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3546–3554.

[17] M. Chen, A. Patney, and A. C. Bovik, "Movi-codec: Deep video compression without motion," in *2021 Picture Coding Symposium (PCS)*. IEEE, 2021, pp. 1–5.

[18] O. Rippel, A. G. Anderson, K. Tatwawadi, S. Nair, C. Lytle, and L. Bourdev, "Elf-vc: Efficient learned flexible-rate video coding. 2021 ieee," in *CVF International Conference on Computer Vision (ICCV)*, vol. 4, 2021.

[19] Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, "Learning image and video compression through spatial-temporal energy compaction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 071–10 080.

[20] N. Zou, H. Zhang, F. Cricri, H. R. Tavakoli, J. Lainema, E. Aksu, M. Hannuksela, and E. Rahtu, "End-to-end learning for video frame compression with self-attention," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 142–143.

[21] G. Lu, C. Cai, X. Zhang, L. Chen, W. Ouyang, D. Xu, and Z. Gao, "Content adaptive and error propagation aware deep video compression," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*. Springer, 2020, pp. 456–472.

[22] W. Sun, C. Tang, W. Li, Z. Yuan, H. Yang, and Y. Liu, "High-quality single-model deep video compression with frame-conv3d and multi-frame differential modulation," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXX 16*. Springer, 2020, pp. 239–254.

[23] G. He, C. Wu, L. Li, J. Zhou, X. Wang, Y. Zheng, B. Yu, and W. Xie, "A video compression framework using an overfitted restoration neural network," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 148–149.

[24] G. He, L. Xu, J. Lei, W. Xie, Y. Li, Y. Fan, and J. Zhou, "Interlayer restoration deep neural network for scalable high efficiency video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 5, pp. 3217–3234, 2021.

[25] B. Liu, Y. Chen, S. Liu, and H.-S. Kim, "Deep learning in latent space for video prediction and compression," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 701–710.

[26] S. Lombardo, J. Han, C. Schroers, and S. Mandt, "Deep generative video compression," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[27] R. Yang, Y. Yang, J. Marino, and S. Mandt, "Hierarchical autoregressive modeling for neural video compression," *arXiv preprint arXiv:2010.10258*, 2020.

[28] J. Liu, S. Wang, W.-C. Ma, M. Shah, R. Hu, P. Dhawan, and R. Urtasun, "Conditional entropy coding for efficient video compression," in *European Conference on Computer Vision*. Springer, 2020, pp. 453–468.

[29] J. Li, B. Li, and Y. Lu, "Deep contextual video compression," *Advances in Neural Information Processing Systems*, vol. 34, pp. 18 114–18 125, 2021.

[30] C. Liu, H. Sun, Z. Cheng, M. Takeuchi, J. Katto, X. Zeng, Y. Fan *et al.*, "Dual learning-based video coding with inception dense blocks," in *2019 Picture Coding Symposium (PCS)*. IEEE, 2019, pp. 1–5.

[31] K. Yang, D. Liu, Z. Chen, F. Wu, and W. Li, "Spatiotemporal generative adversarial network-based dynamic texture synthesis for surveillance video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 1, pp. 359–373, 2021.

[32] S. Wang, C. Jia, X. Zhang, S. Wang, S. Ma, and W. Gao, "A pixel-level segmentation-synthesis framework for dynamic texture video compression," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 10, pp. 7077–7091, 2022.

[33] B. Chen, Z. Wang, B. Li, R. Lin, S. Wang, and Y. Ye, "Beyond keypoint coding: Temporal evolution inference with compact feature representation for talking face video compression," in *2022 Data Compression Conference (DCC)*. IEEE, 2022, pp. 13–22.

[34] Z. Chen, G. Lu, Z. Hu, S. Liu, W. Jiang, and D. Xu, "Lsvc: A learning-based stereo video compression framework. in 2022 ieee," in *CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 6063–6072.

[35] J. Lei, Z. Zhang, Z. Pan, D. Liu, X. Liu, Y. Chen, and N. Ling, "Disparity-aware reference frame generation network for multiview video coding," *IEEE Transactions on Image Processing*, vol. 31, pp. 4515–4526, 2022.

[36] G. Lu, T. Zhong, J. Geng, Q. Hu, and D. Xu, "Learning based multi-modality image and video compression," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 6083–6092.

[37] Y.-C. Su and K. Grauman, "Learning spherical convolution for fast features from 360 imagery," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[38] R. Monroy, S. Lutz, T. Chalasani, and A. Smolic, "Salnet360: Saliency maps for omni-directional images with cnn," *Signal Processing: Image Communication*, vol. 69, pp. 26–34, 2018.

[39] M. Ruder, A. Dosovitskiy, and T. Brox, "Artistic style transfer for videos and spherical images," *International Journal of Computer Vision*, vol. 126, no. 11, pp. 1199–1219, 2018.

[40] Y. Lee, J. Jeong, J. Yun, W. Cho, and K.-J. Yoon, "Spherephd: Applying cnns on a spherical polyhedron representation of 360deg images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9181–9189.

[41] C. Zhang, S. Liwicki, W. Smith, and R. Cipolla, "Orientation-aware semantic segmentation on icosahedron spheres," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 3533–3541.

[42] T. Cohen, M. Weiler, B. Kicanaoglu, and M. Welling, "Gauge equivariant convolutional networks and the icosahedral cnn," in *International conference on Machine learning*. PMLR, 2019, pp. 1321–1330.

[43] R. Khasanova and P. Frossard, "Graph-based classification of omnidirectional images," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2017, pp. 869–878.

[44] N. Perraudin, M. Defferrard, T. Kacprzak, and R. Sgier, "Deepsphere: Efficient spherical convolutional neural network with healpix sampling for cosmological applications," *Astronomy and Computing*, vol. 27, pp. 130–146, 2019.

[45] Q. Yang, C. Li, W. Dai, J. Zou, G.-J. Qi, and H. Xiong, "Rotation equivariant graph convolutional network for spherical image classification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 4303–4312.

[46] C. Esteves, C. Allen-Blanchette, A. Makadia, and K. Daniilidis, "Learning so (3) equivariant representations with spherical cnns," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 52–68.

[47] T. S. Cohen, M. Geiger, J. Köhler, and M. Welling, "Spherical cnns," *arXiv preprint arXiv:1801.10130*, 2018.

[48] C. Esteves, A. Makadia, and K. Daniilidis, "Spin-weighted spherical cnns," *Advances in Neural Information Processing Systems*, vol. 33, pp. 8614–8625, 2020.

[49] P. J. Roddy and J. D. McEwen, "Sifting convolution on the sphere," *IEEE Signal Processing Letters*, vol. 28, pp. 304–308, 2021.

[50] I. Tosic, P. Frossard, and P. Vandergheynst, "Progressive coding of 3-d objects based on overcomplete decompositions," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 11, pp. 1338–1349, 2006.

[51] R. Khasanova and P. Frossard, "Geometry aware convolutional filters for omnidirectional images representation," in *International Conference on Machine Learning*. PMLR, 2019, pp. 3351–3359.

[52] A. Ranjan and M. J. Black, "Optical flow estimation using a spatial pyramid network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4161–4170.

[53] R. Yang, L. Van Gool, and R. Timofte, "Opendvc: An open source implementation of the dvc video compression method," *arXiv preprint arXiv:2006.15862*, 2020.

[54] M. Xu, C. Li, Y. Liu, X. Deng, and J. Lu, "A subjective visual quality assessment method of panoramic videos," in *2017 IEEE International Conference on Multimedia and Expo (ICME)*, 2017, pp. 517–522.

[55] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[56] X. Xiu, Y. He, Y. Ye, and B. Vishwanath, "An evaluation framework for 360-degree video compression," in *2017 IEEE Visual Communications and Image Processing (VCIP)*, 2017, pp. 1–4.

A Extra Material

Motion Compression Network used in our model

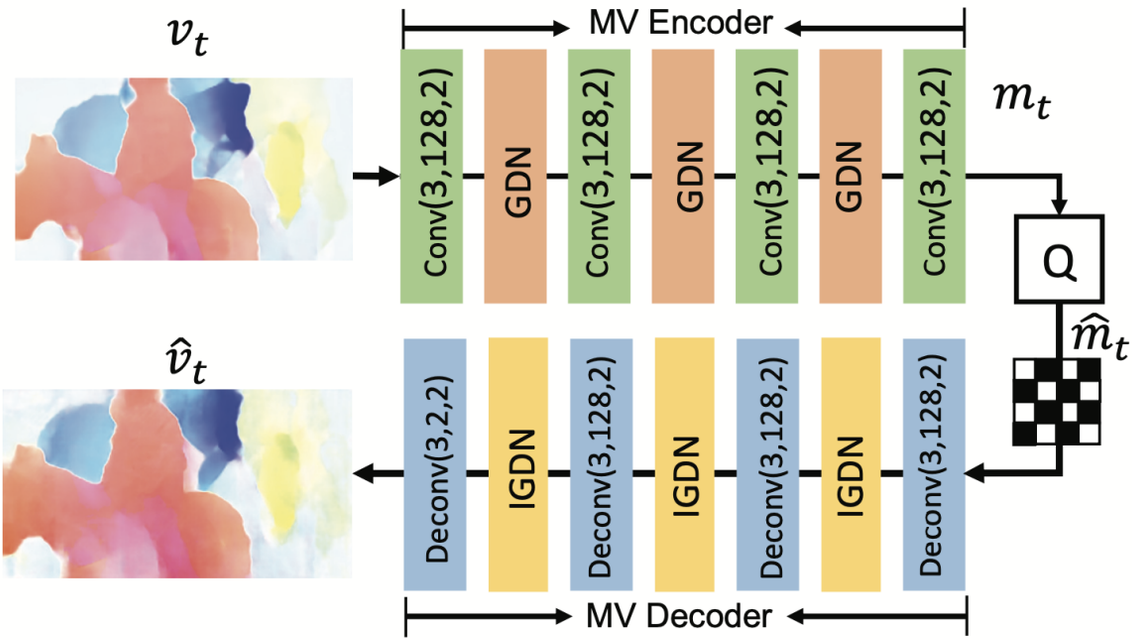
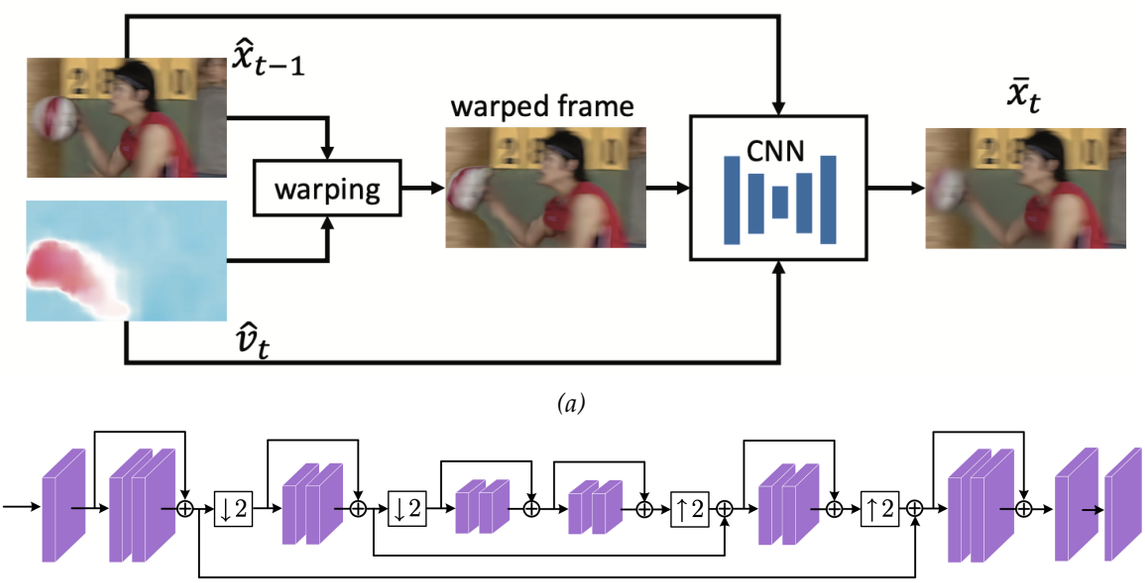


Figure 13: Conv(3,128,2) represents the convoluation operation with the kernel size of 3x3, the output channel of 128 and the stride of 2. GDN and IGDN is the nonlinear transform function.

Motion Compensation Network used in our model



(a)

(b) Network structure of CNN denoted in (a)

Figure 14: Motion compensation network