Exploring Spherical Autoencoder for Spherical Video Content Processing

Jin Zhou
George Mason University
Fairfax, Virginia, USA
jzhou23@gmu.edu

Na Li
Rutgers University
Piscataway, New Jersey, USA
na.li@rutgers.edu

Yao Liu
Rutgers University
Piscataway, New Jersey, USA
yao.liu@rutgers.edu

Shuochao Yao
George Mason University
Fairfax, Virginia, USA
shuochao@gmu.edu

Songqing Chen
George Mason University
Fairfax, Virginia, USA
sqchen@gmu.edu

ABSTRACT

3D spherical content is increasingly presented in various applications (e.g., AR/MR/VR) for better users' immersiveness experience, yet today processing such spherical 3D content still mainly relies on the traditional 2D approaches after projection, leading to the distortion and/or loss of critical information. This study sets to explore methods to process spherical 3D content directly and more effectively. Using 360-degree videos as an example, we propose a novel approach called Spherical Autoencoder (SAE) for spherical video processing. Instead of projecting to a 2D space, SAE represents the 360-degree video content as a spherical object and employs encoding and decoding on the 360-degree video directly. Furthermore, to support the adoption of SAE on pervasive mobile devices that often have resource constraints, we further propose two optimizations on top of SAE. First, since the FoV (Field of View) prediction is widely studied and leveraged to transport only a portion of the content to the mobile device to save bandwidth and battery consumption, we design p-SAE, a SAE scheme with the partial view support that can utilize such FoV prediction. Second, since machine learning models are often compressed when running on mobile devices in order to reduce the processing load, which usually leads to degradation of output (e.g., video quality in SAE), we propose c-SAE by applying the compressive sensing theory into SAE to maintain the video quality when the model is compressed. Our extensive experiments show that directly incorporating and processing spherical signals is promising, and it outperforms the traditional approaches by a large margin. Both p-SAE and c-SAE show their effectiveness in delivering high quality videos (e.g., PSNR results) when used alone or combined together with model compression.

CCS CONCEPTS

• **Computing methodologies → Machine learning**.

KEYWORDS

Spherical Autoencoder, Partial View, Compressive Sensing

ACM Reference Format:
Jin Zhou, Na Li, Yao Liu, Shuochao Yao, and Songqing Chen. 2022. Exploring Spherical Autoencoder for Spherical Video Content Processing. In *Proceedings of the 30th ACM International Conference on Multimedia (MM '22), October 10–14, 2022, Lisboa, Portugal.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3503161.3548364

1 INTRODUCTION

The ever-improving technologies have driven the increasing demand of better "immersive" experiences from end-users for various video content. While traditional video traffic has dominated the Internet backbone for over a decade [11], recent years have seen the increasing portion of 360-degree video traffic. These days many people start to share their video content in the 360-degree format on YouTube or Facebook. Real estates also use 360-degree videos to showcase houses [6].

Compared to the traditional (2D) videos, 360-degree videos, offering 3 Degree of Freedom (3 DoF), can provide better "immersive" experiences to users, and are the basis for the augmented/mixed/virtual (AR/MR/VR) applications that offer 6 DoF. With the wide availability of RGBD cameras and Lidar, many video analytics engines also need to process such content in the domains of autonomous driving and robotics. More broad applications are also been explored, such as remote education, telementoring [27], holoportation [35] from Microsoft, and Starline [12] from Google.

However, currently, processing and transmitting such spherical 3D content still mainly rely on the traditional approaches developed for 2D video content. For example, for a 360-degree video, a common thread of existing approaches is to project the video frames from 3D to 2D and then use the traditional coding schemes, such as MPEG, to compress the video content. After being received by the receiver, the process is reversed and the 360-degree frames are reconstructed.

While this approach can quickly utilize the existing codecs and the transport and processing support, during this process, a lot of critical "immersive" information may get distorted or lost. For example, the projection is often the first step when mapping a 3D frame to a 2D space. The commonly used projection schemes include equirectangular projection [3], cubic projection [8], rectilinear projection [9], equi-angular cube (EAC) [7], etc. However, regardless which projection scheme is used, the distortion of the

MM '22, October 10–14, 2022, Lisboa, Portugal

ACM ISBN 978-1-4503-9203-7/22/10…$15.00
https://doi.org/10.1145/3503161.3548364

polar areas is inevitable. Furthermore, traditional schemes were developed without being aware of the 3D information available. Thus when they are utilized to process the mapped 3D spherical content, such information may not be appropriately processed, leading to degraded "immersiveness" experience.

In this work, we set to explore a new approach that can handle spherical videos directly and more effectively. Instead of following the traditional approach by projecting the 3D content to a 2D space, we consider a 360-degree video as a spherical object in its entirety and propose to build spherical auto-encoder (SAE) to process such 360-degree videos directly. For this purpose, in SAE, icosahedral spherical mesh [17] is used to represent pixels on the spherical surface, which allows flexible "refinement" based on the user demand. Compared to traditional approaches, no sphere-to-2D projection is required. Moreover, we adopt the spherical convolution neural network (CNN) to process the spherical content directly in order to preserve the 3D information as much as possible. To upsample the pixels to the original resolution, we use a novel VertexShuffle [32] operation on the mesh, inspired by the PixelShuffle on 2D images.

Furthermore, to support SAE on the pervasive mobile devices that have resource and battery constraints, we further propose two optimizations on top of SAE. First, since field-of-view (FoV) prediction [16, 41] has been heavily studied to only transport content in FoV in high quality while other parts in low quality or no transmission at all, thus reducing bandwidth and battery consumption, we design p-SAE, namely partial SAE, a SAE scheme with the partial view support that can utilize such FoV prediction.

Second, since machine learning models are often compressed when running on mobile devices in order to reduce the processing load, which usually leads to degradation of output (e.g., video quality in SAE), we propose c-SAE by applying the compressive sensing theory into SAE to maintain the video quality when the model is compressed.

To evaluate the performance of SAE and p-SAE, we conduct experiments and compare to the traditional approach. The results show the SAE approach is promising, and outperforms the traditional approach (e.g., 2D convolutional autoencoder coupled with the equirectangular projection) by a large margin. Moreover, p-SAE and c-SAE also show comparable performance to the full SAE, when used alone or combined together with model compression.

The rest of the paper is organized as follows. Section 2 presents some background information and related work. We present the design of our spherical autoencoder in section 3, the optimization for mobile devices with p-SAE in section 4, and c-SAE with compressive sensing in section 5. Experimental results are discussed in section 6. We make concluding remarks in section 7.

2 BACKGROUND AND RELATED WORK

In this section, we present some background and prior work on autoencoders and spherical convolutional neural networks, which motivates our new design presented in the next section.

2.1 Image Compression and Autoencoders

Traditional 2D image/video compression has been extensively and continuously researched. Traditional approaches for still image or motion picture compression have often focused on using JPEG [43] and MPEG based schemes, such as JPEG, MPEG2 [2], JPEG 2000 [1], H.264 [4] and H.265 [5].

The recent deep neural network research has also motivated a number of studies to use machine/deep learning techniques for image/video compression [13, 21–23, 30, 38, 42, 44]. While some studies aimed to improve based on existing frameworks, e.g., Chen et al. [21] proposed a learning based framework to effectively perform predictive coding inside the learning network for video compression with iterative analysis/synthesis and binarization, a lot of studies have turned attention to autoencoders [13, 22, 23, 30, 42], because compared to the traditional approaches, autoencoders are more flexible and adaptive to different media formats and resource requirements. For example, Theis et al. [42] proposed a deep autoencoders framework and achieved competitive performance to JPEG2000. For video compression, Habibian et al. [30] studied rate distortion and proposed a deep generative model for lossy video compression that outperforms the learned video compression networks based on motion compensation or interpolation.

However, most of existing works on autoencoders focused on the traditional 2D image or video compression. The burgeoning spherical objects, such as 360-degree videos, received little attention. In this work, we aim to explore the construction of autoencoders for spherical video processing.

2.2 Convolution Neural Network (CNN) and Spherical CNN

Convolution neural networks (CNN) has been widely used in deep learning for processing traditional images. However, for images from emerging applications like omnidirectional vision for drones, robots, and autonomous cars, and planetary signals in scientific domains like global weather and climate modelling, the images have to be projected to the plantar space before being processed, which inevitably introduces information distortion and loss.

To properly process spherical images, spherical convolution neural networks (S-CNN) have been proposed [24, 28, 31]. Cohen et al. [24] proposed the building blocks of Spherical CNN, formed the theory of spherical CNNs and verified its properties, and showed that it can be utilized for rotation invariant classification and regression problems. More recently, Jiang et al. [31] optimized the implementation on unstructured grid with UGSCNN using parameterized differential operators. UGSCNN is shown to be extremely efficient and it can match or outperform state-of-the-art network architectures in terms of performance but with a significantly lower number of network parameters. While there is no work on spherical autoencoders yet, in our design, we will leverage the building blocks of UGSCNN to implement SAE.

3 SPHERICAL AUTOENCODER (SAE)

In this section, we present our design of SAE, after a brief introduction of the icosahedral mesh that we utilize to represent the spherical objects.

3.1 Icosahedral Mesh Representation

In SAE design, similar to UGSCNN [31], we represent spherical objects with icosahedral mesh [17]. We consider each spherical video

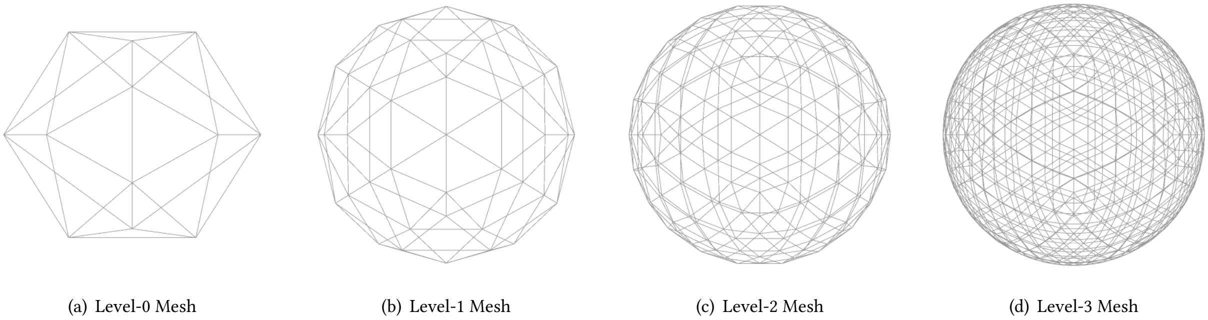

(a) Level-0 Mesh (b) Level-1 Mesh (c) Level-2 Mesh (d) Level-3 Mesh

Figure 1: The icosahedral mesh representation in Level 0 to Level 3.

consisting of spherical frames. To represent pixels on each spherical frame, we discretize the sphere using refined icosahedral mesh. Pixels of the video frame are then mapped to their corresponding vertices of the refined mesh. The icosahedral mesh representation starts with a regular icosahedron and re-projects all its 12 vertices to a unit sphere (Figure 1(a)). To refine the mesh, we can divide each of the 20 triangular faces into 4 smaller faces by creating new vertices at mid-points of the edges, creating new edges among these new vertices, and normalizing the new vertices to the unit sphere.

We call the regular icosahedron re-projected to the unit sphere the Level-0 mesh, and we obtain Level-N mesh by repeating the refining process N times, each time dividing a triangular into 4 triangles, by selecting the midpoints of each of the three edges and connecting these midpoints to form three new edges. Figures 1(b), (c), and (d) show the icosahedral mesh obtained after refining the Level-0 mesh 1, 2, and 3 times. Each time the mesh is refined, the number of vertices is approximately multiplied by 4. That is, the more times the mesh is refined, the more spherical pixels can be represented, and thus the higher resolution of the spherical frame can be obtained. For example, with this discretization, a Level-9 mesh contains 2,621,442 vertices. The pixel density of this mesh representation around the sphere's equator area is roughly equivalent to a spherical frame represented in the 2D equirectangular projection in 2880x1440 resolution.

3.2 Spherical Autoencoder (SAE) with Full Icosahedral Mesh

Figure 2 shows the proposed spherical autoencoder (SAE) architecture. To process an input video, we first load each frame into the icosahedral mesh. The encoder of SAE loads the RGB values of pixels on the spherical video frame as values of vertices on the icosahedral mesh, e.g., a Level-9 mesh as shown in the figure. This results in a $3 \times N_{v,9}$ tensor, where $N_{v,9}$ represents the number of vertices of a full Level-9 icosahedral mesh. It then goes through a MeshConv layer with batch normalization and ReLU function.

The MeshConv [31] operation shows in the figure can be represented as: $\text{MeshConv}(F;\ \theta) = \theta_0 IF + \theta_1 \nabla_x F + \theta_2 \nabla_y F + \theta_3 \nabla^2 F$. Here, I represents the identity function, ∇_x and ∇_y represent the first order differential operator on the mesh in x and y, two orthogonal dimensions, ∇^2 represents the 2nd order differential operator on the mesh, and θ_0, θ_1, θ_2, and θ_3 are learned parameters.

The output then goes through two ResBlocks [31] to both coarsen the mesh (i.e., coarsen the mesh from Level-9 mesh to Level-8 and Level-7 meshes, respectively.) and increase the channel dimension. Finally, we use another MeshConv layer to change the channel dimension to 3. In this way, the output tensor of the SAE encoder is a $3 \times N_{v,7}$ tensor, a low-dimensional representation of the spherical pixels, achieving a 16x compression ratio compared to the input $3 \times N_{v,9}$ tensor.

The decoder of SAE takes the low-dimensional $3 \times N_{v,7}$ tensor compressed input. Instead of using deconvolution operations to reconstruct the original data, we use a novel VertexShuffle operation proposed in our prior work [32]. The VertexShuffle operation on icosahedral meshes is inspired by the PixelShuffle operation on 2D images [40]. We denote the VertexShuffle operation as:

$$M_{i+1} = \text{VertexShuffle}(M_i),$$

where $M_i \in \mathbb{R}^{C \times N_{v,i}}$ represents features of the Level-i mesh where C is the feature dimension, $M_{i+1} \in \mathbb{R}^{C/4 \times N_{v,i+1}}$ represents features of the Level-$i+1$ mesh with $N_{v,i+1}$ vertices, and the feature dimension is reduced to $C/4$. Similar to the PixelShuffle operation, the VertexShuffle operation on the mesh also does not require any parameters. Unlike the PixelShuffle operation, feature maps are not simply shuffled in VertexShuffle. Instead, given that the spherical mesh is refined by progressively creating new vertices at edge midpoints and sub-dividing each face into four equal triangles, we split M_i into four parts $\{M_{i0}, M_{i1}, M_{i2}, M_{i3}\}$ (thus the feature dimension of M_{i+1} becomes $C/4$) and use $\{M_{i1}, M_{i2}, M_{i3}\}$ for constructing midpoints on three edges of triangles as follows:

$$N'_{i0} = (M_{i1}(v_0) + M_{i1}(v_1))/2$$
$$N'_{i1} = (M_{i2}(v_1) + M_{i2}(v_2))/2$$
$$N'_{i2} = (M_{i3}(v_2) + M_{i3}(v_0))/2$$

Here, v_0, v_1, and v_2 represent vertices of a triangle face. Due to shared edges among triangular faces on the mesh, we deduplicate the new midpoints: $N_i = unique(N'_i)$. Finally, we can obtain the output of the VertexShuffle operation: $\text{VertexShuffle}(M_i) = concat(M_{i0}, N_i)$.

The decoder passes its input tensor through one MeshConv layer and two ResBlocks to increase the channel dimensions. It then passes the output through two VertexShuffle operations to increase the number of vertices, e.g., from a Level-7 mesh to a Level-9 mesh.

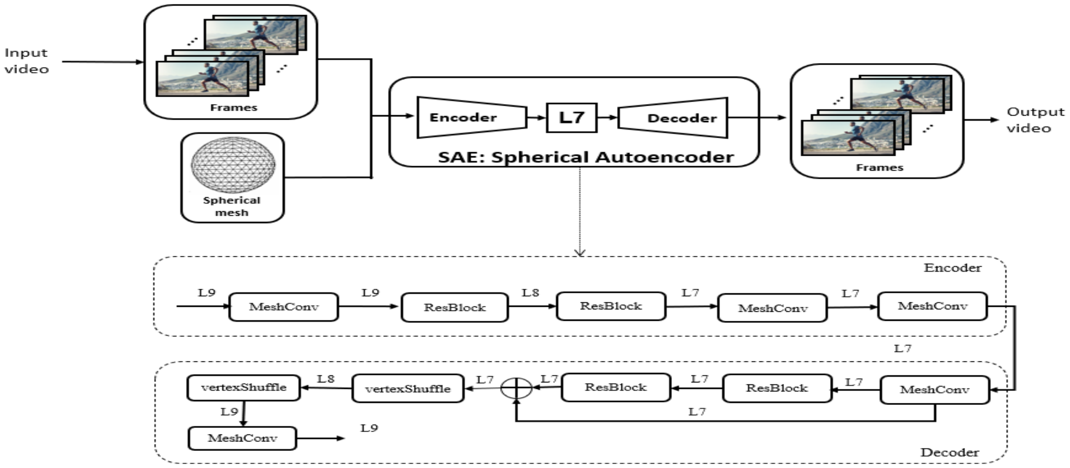


Figure 2: The SAE architecture. As an example, the input to the SAE encoder in this figure is a Level-9 mesh with each vertex representing the RGB values of a pixel.

A final MeshConv layer in the decoder produces a $3 \times N_{v,9}$ tensor, in the same dimension as the original input to the SAE encoder.

To minimize the reconstruction error, we compare the input to the encoder with the output of the decoder of SAE and use a customized negative PSNR loss, i.e., $10 \times \log_{10}(\text{MSELoss})$.

While Figure 2 shows an example of compressing spherical pixels in Level-9 mesh input to Level-7 mesh, achieving 16x compression, it is possible to adapt the model to allow inputs and create low-dimension representations of different mesh granularity, e.g., Level-8 mesh input and Level-6 encoder output if the original video is in lower resolution.

4 PARTIAL SAE (p-SAE) WITH PARTIAL ICOSAHEDRAL MESH

In parallel to the increase of spherical videos, another trend seen in the recent years is that today more and more users tend to use mobile devices in sharing and watching videos. Thus, it is necessary and desirable to optimize SAE for mobile devices, given that mobile devices often have resource constraints, particularly the limited battery power supply. For this purpose, we further present two optimizations, p-SAE and c-SAE, that we design for mobile devices in this and next section, respectively.

p-SAE is designed to utilize the field-of-view (FoV) prediction [16, 41]. FoV prediction has been extensively studied to only transport content in FoV in high quality while other parts in low quality or no transmission at all, thus reducing bandwidth and battery consumption. A spherical video contains information about every direction surrounding the camera. However, usually only a small portion of the spherical content may be of interest to the viewers at a time. Thus, for bandwidth-efficient spherical content transportation, only spherical content expected to be viewed can be transmitted and decoded.

To support the utilization of FoV, we use the partial icosahedral mesh [32]. The partial icosahedral mesh is created by selecting one triangular face from the full Level-1 mesh (that is, only 1 out of the 80 faces) and only refine triangles within this face. As a result, the refined face is about 1/80 of the sphere and contains roughly 1/80 of the vertices in a full mesh.

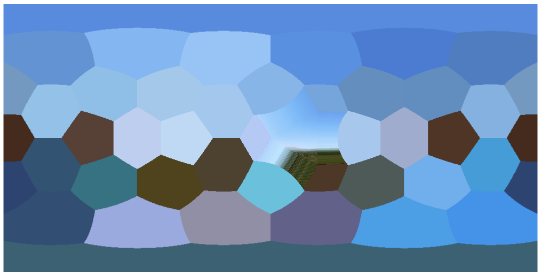


Figure 3: Example image after using partial mesh

For example, if we load a spherical frame to the partial icosahedral mesh, then re-project the partial mesh to 2D planar using the equirectangular projection, we may obtain an image as shown in Figure 3. In this image, only one triangular face roughly in the middle of the image contains detailed information about the image.

To perform encoding and decoding operations for spherical content belonging to any of the rest 79 faces, vertices in these 79 faces (in a full mesh) are rotated to vertices of the selected refined face using a calculated rotation matrix (79 matrices in total). In this way, we can use SAE that operates on the partial icosahedral mesh to selectively encode and decode the spherical content, and we call this partial SAE (p-SAE).

With p-SAE, if reliable prediction can be made that only N out of 80 full Level-1 mesh faces are required to be transmitted, then the autoencoder calculation only needs to be performed on these faces, thereby saving both network transmission and encoder/decoder computation costs.

Using the partial icosahedral mesh also brings another benefit of disk and memory storage space savings. The storage size of a partial mesh is only about 1/80 compared to the full mesh at the same level.

5 COMPRESSIVE SAE (c-SAE)

p-SAE can reduce the resource consumption on mobile devices by transmitting less data. On the other hand, today when machine learning models are deployed on mobile devices, a common practice is to compress the model in order to reduce its processing load and fit better with the resource constraints on mobile devices. However, this often comes with the degradation of the model output, which, for SAE, is the video quality. To maintain the video quality while the model is compressed, we propose to integrate the compressive sensing theory in SAE.

5.1 Compressive Sensing

Compressive sensing has been widely used to recover an unknown data vector, $x \in \mathbb{R}^n$, from a few linear measurements, $y \in \mathbb{R}^m$.

$$y = Ex, \tag{1}$$

where $E \in \mathbb{R}^{m \times n}$ is the measurement matrix. Since the number of measurements is far less than the number of data points, *i.e.*, $m \ll n$, determining x by solving the equation (1) is an ill-posed inverse problem with no unique solution. As a result, we must add prior knowledge to the data vector x and measurement matrix E. In classical compressive sensing theory, x is commonly assumed to be a sparse vector in a set of basis Φ and E to satisfy the Restricted Isometry Property (RIP) or the Restricted Eigenvalue Condition (REC) [20], and thus we can guarantee that minimizing the recovery error,

$$\hat{x} = \arg\min_{x} \|y - Ex\|^2 \quad s.t. \;\; \|\Phi x\|_1 \leq \beta, \tag{2}$$

leads to accurate reconstruction with a high probability. The constrained minimization problem (2) is usually solved by iterative gradient projection algorithms [18, 29]. However, conventional compressive sensing has two fundamental limitations that make it inappropriate for domain-specific encoder-decoder construction. On the one hand, while sparsity priors have been shown to be effective, the sparsity property of spherical 3D image data is unclear, and more complicated models with more structure have recently been proposed with superior reconstruction performance [19, 45, 46]. On the other hand, the slow iterative gradient projection algorithms are used to solve the reconstruction problem (2), which significantly slows down the decoding time.

5.2 Compressive SAE (c-SAE)

To address the aforementioned two drawbacks, compressive SAE (c-SAE) links the proposed SAE (spherical encoder-decoder structure) with compressive sensing theory. Instead of relying on pre-defined sparsity, the spherical decoder acts as an implicit prior constraint for decoding compressed spherical images, allowing us to reconstruct the encoded data with a single decoder run and avoid the costly iterative methods used in traditional compressive sensing techniques. The spherical encoder component, on the other hand, is viewed as a learnable yet lightweight module. It can also automatically learn the appropriate transformation for compressing spherical video content for offloading with the least computational cost on local devices. According to the recent deep compressive offloading theory [45], we should impose the Restricted Isometry Property (RIP) (with orthogonal regularization) and Lipschitz continuity (with spectral normalization) on the spherical encoder and decoder, respectively, to provide recovery guarantees for the data encoding-decoding process based on compressive sensing theory.

Orthogonal Regularization If we want to maintain the compression ratio and the quality of recovery, we are required to make our encoder to be isometric. The general idea to achieve this goal is to add the orthogonal regularization with the spherical convolution kernel in the encoder. Orthogonal regularization uses weights to be orthogonal by pushing them towards the nearest orthogonal manifold. Hence, we apply the orthogonal regularization on the spherical convolution layers which are fully-connected. In this model, We can consider the spherical convolution kernel as a 3D kernel, where the convolution part within encoder and decoder takes the value of vertices on Level-n mesh. For this purpose, first, we convert the kernel $K \in \mathbb{R}^{h \times w \times d \times c_i \times c_j \times c_o}$ to $K' \in \mathbb{R}^{h \cdot w \cdot d \times c_i \times c_j \times c_o}$, to make these convolution to be considered as matrix multiplications. Then, to ensure that the convolutions are isometry for a constrained feature space, we can apply orthogonal regularization to the kernel. A linear transformation with a semi-orthogonal matrix is used to guarantee the preservation of the isometric property. We can thus add the orthogonal regularization to the convolution kernel K' during training as follows,

$$argmin \left\| {K'}^T K' - I \right\|,$$

where I is the identity matrix.

Spectral Normalization To achieve the data recovery assurances offered by compressive offloading, the decoder must be an L-Lipschitz function (where L is the Lipschitz constant). Assume e_1 and e_2 are two encoded data samples, and D represents the spherical decoder. Given that D is an L-Lipschitz function,

$$\|D(e_1) - D(e_2)\| \leq L \, \|e_1 - e_2\|$$

Thanks to the recent advances in compressive sensing theory with generative neural networks as the implicit constraint [19], we can attain a similar data recovery guarantee as to the conventional sparsity constraints when the generative neural network (i.e., the spherical decoder in our paper) is an L-Lipschitz function. So the remaining question is how to apply this Lipschitz constant on the decoder efficiently. Here, we use spectral normalization [37], a technique that has been widely adopted in generative neural network models such as Wasserstein GAN [14].

Neural networks are layered structures. If we can constrain the Lipschitz constant of each layer to be smaller than one, the whole neural network becomes a 1-Lipschitz function. The neural network operation in each layer may be thought of as an affine transformation followed by an activation function. The Lipschitz constant of all widely adopted activation functions, including ReLU and Sigmoid, is less than 1. The biggest singular value of the weight matrix, on the other hand, controls the Lipschitz constant of an affine transformation. We can normalize the weight matrix against the biggest singular value of the weight matrix to maintain the Lipschitz constant of each layer smaller than 1, which is precisely what spectral normalization does.

With the orthogonal regularization and spectral normalization, the final structure of our autoencoder is depicted in Figure 4. According to deep compressive offloading theory [45], with orthogonal regularization on the encoder and spectral normalization on the

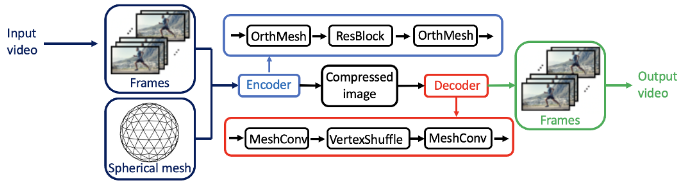


Figure 4: Architecture of Compressive Spherical Autoencoder

decoder, we can ensure that the spherical decoder can reconstruct the offloading data given by the spherical encoder with almost no loss at a high probability.

6 EVALUATION

We present the evaluation results after an introduction of the data sets and our baseline model.

6.1 Experiment Setup

Setup Our models run on a machine with Tesla P100-PCIE GPU with 16G memory, and 2.3 GHz Intel Xeon CPU. Since PSNR is commonly used to evaluate the quality of videos, during training, we use PSNR as the loss function. We also use the model size to evaluate the scalability and flexibility of the SAE. For example, if the model size is very large, it has to be implemented on the server side to ensure its performance. During the training, when the model needs to load Level-9 full mesh, the batch size was set to 2, because the Level-9 full mesh cost about 2 GB memory. If there is no constraint on the memory size, a large batch size can be used. On the other hand, when the video quality requirement is not high, we can use Level-8 or even lower level mesh to reduce the memory demand. In these cases, the batch size can be set to 16 or 32.

Dataset In our experiments, we use a few different 360-degree videos that we downloaded from the Internet. We classify them into four categories and only present the result of one representative video from each category. [1] The first is *highly dynamic* videos such as sports. We choose *Football* for the experiments. The next is *dynamic*, including amusement videos and performance videos. We choose *Roller Coaster* for our experiments. The third is *low motion* videos, which include observation videos recorded indoor, for example, a classroom with students and teacher with few movements. We choose *Indoor* in our experiments. The last includes the relatively *static* videos. These videos contain daily monitoring of the city and urban planning and construction. We choose *City*. These videos have different resolutions: Indoor is 1920 × 960 , City is 2056 × 1080, Roller-Coaster is 1920 × 1080 and Football is 2048 × 4096.

Baseline Model We compare our approaches with the baseline approach – the convolutional autoencoder, denoted as CAE [10]. In the experiments, we use the equirectangular projection to first project each frame into a rectangular image according to its length and width. Then we divide this rectangular picture into n patches, and run 2D convolutional autoencoder on each patch. The compression ratio of CAE is 12x.

6.2 Evaluation Results

PSNR results Table 1 reports the PSNR results of our SAE model and its variants. In these models, the Level-9 mesh is used. In this table, *CAE* represents the results from the traditional 2D convolutional autoencoder, and *co-CAE* represents the compressed CAE. *SAE* (Spherical AutoEncoder) uses the full sphere mesh. *p-SAE* uses partial mesh in SAE instead of full mesh. *c-SAE* applies compressive sensing in SAE, and *c-p-SAE* applies compressive sensing in p-SAE. These models are all compressed with different compression rates as indicated in the table (shown as "model compression ratio").

As shown in Table 1, the second column shows the original CAE (top half) and compressed CAE (bottom half) results. The original CAE delivers spherical videos with a PSNR below 25 in general, while the compressed one can only deliver the video with a PSNR at 20 or lower. We did not include more compression results as these are already too low for users. Compared to CAE, the results of SAE models are significantly better (the PSNR is above 30 in all cases), especially for high resolution videos. This is because for the high resolution video, especially for 360 videos, the resolution is not like 2D. These video frames have to be projected and divided into several patches to process. This will casue some extraction loss for CAE to process high resolution 360 video frames but not for SAE models. Overall, we observe SAE models always outperform CAE by a large margin. This shows the benefit of operating on the spherical pixels directly compared to first projecting the pixels to the 2D planar image which results in distortions.

Comparing p-SAE with SAE, we find that the PSNR results of p-SAE are degrading in general, but are still above 30 dB, indicating acceptable results. These results indicate that p-SAE is promising when mobile devices are used to watch the video and a careful trade-off should be explored when deploying. On the other hand, when integrating compressive sensing with model compression, we observe very interesting results. Compressive sensing techniques are integrated in order to maintain the video quality after decoding, i.e., data recovery. Comparing SAE with c-SAE results in Table 1, we can see that with compressive sensing, the PSNR results are improved. The reason is that through orthogonal regularization and spectral normalization, c-SAE provides better guarantees for data decoding. We can observe similar trends when comparing p-SAE with c-p-SAE. These results indicate that after adopting compressive sensing in these models when they are compressed for mobile devices, c-SAE models can effectively help with the mobile device in accessing videos without decreasing the video quality.

Model Size Table 2 further shows the model size and GPU usage of these models under different compression rates. For CAE, the total number of parameters is 2,241,859. The model size is 8.6 MB. While for SAE, the total number of parameters with full mesh is 51,050, which means the model size of SAE is about 268 KB. Thus, the CAE model is over 30 times larger than that of SAE. This indicates that SAE not only delivers better quality, but also has better portability, particularly when the quality demand is high. The total number of parameters of p-SAE model for partial mesh implementation is 47,604, which translates into the model size of P-SAE around 250 KB. Note that when training the model with Level-9 full mesh, we will have to load about 2 GB Level-9 spherical mesh that requires a lot of GPU memory. This will

[1] These videos are from the "The Psychology of 360-Video" repository [36], available at https://github.com/vhilab/psych-360

Table 1: PSNR result of CAE, SAE, p-SAE, c-SAE, c-p-SAE with different compression rates

Model	CAE	SAE				p-SAE			
Model Compression Ratio	none	none	26.01%	41.54%	58.30%	none	36.03%	45.97%	55.11%
Indoor	24.7384	39.2887	41.2183	39.0652	38.3835	37.8229	37.7869	36.4909	35.9083
City	17.1524	39.7354	38.9868	38.5078	37.6546	33.4587	33.6031	32.7565	31.8372
Roller-Coaster	17.8414	34.1936	32.8375	32.0299	31.5185	32.5046	32.1746	31.5830	30.3547
Football	20.6538	36.2050	36.3093	36.2165	36.1787	34.4323	33.9375	33.7339	33.2091
Model	co-CAE	c-SAE				c-p-SAE			
Model Compression Ratio	37.21%	none	26.01%	41.54%	58.30%	none	36.03%	45.97%	55.11%
Indoor	20.1124	40.5602	41.3595	39.7724	38.7182	38.6803	38.5517	37.6399	37.5349
City	15.0681	40.3514	39.4804	39.1991	38.9915	34.5360	36.4247	33.6042	33.0823
Roller-Coaster	13.5168	35.5432	33.8656	33.4480	32.6312	33.2020	32.2679	32.1994	31.0052
Football	14.8910	35.5992	36.8822	36.4333	36.0842	36.6811	36.8463	36.4243	36.1195

Table 2: Model size and GPU usage of different models

Model	CAE	SAE				p-SAE			
Model Compression Ratio	none	none	26.01%	41.54%	58.30%	none	36.03%	45.97%	55.11%
Model Size (KB)	8600	268	195	154	110	250	180	152	126
GPU Usage	4702	13261	13053	12371	11647	2973	2619	2578	2477
Model	co-CAE	c-SAE				c-p-SAE			
Model Compression Ratio	37.21%	none	26.01%	41.54%	58.30%	none	36.03%	45.97%	55.11%
Model Size (KB)	5400	330	207	174	135	321	202	187	131
GPU Usage	3514	15479	15291	14695	14027	1692	1447	1397	1375

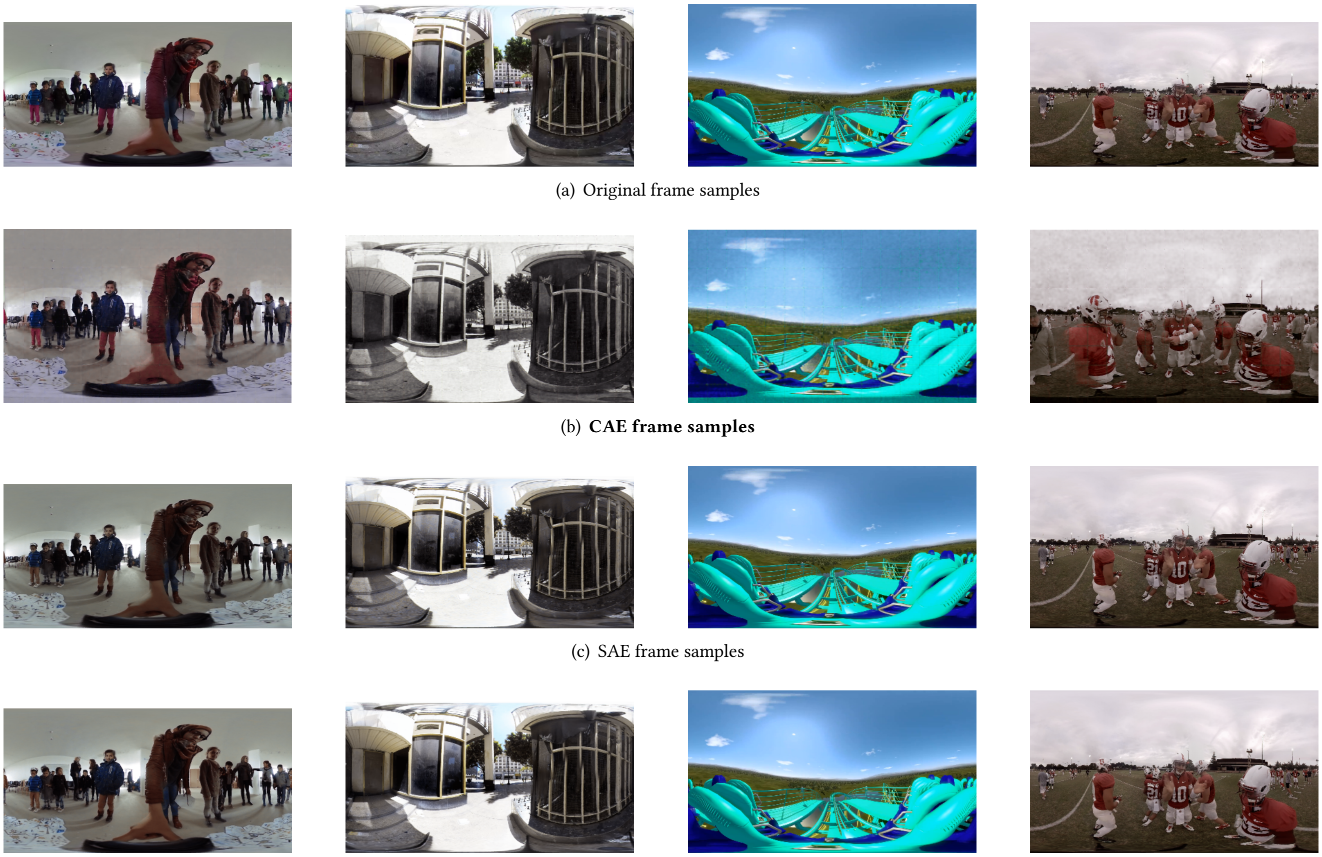
Table 3: VI-VMAF scores

Model	CAE	SAE				p-SAE			
Model Compression Ratio	none	none	26.01%	41.54%	58.30%	none	36.03%	45.97%	55.11%
Indoor	34.6512	73.4501	73.2218	72.6442	72.6351	70.4519	70.1362	69.8044	69.2314
City	34.0079	72.7804	72.6419	71.9773	71.0025	70.7577	70.5893	69.7488	69.6832
Roller-Coaster	28.5093	68.1255	68.1238	67.9836	67.4821	66.2091	65.9343	65.4290	65.1028
Football	29.8409	68.8549	68.6027	68.4981	68.0195	67.5121	67.4982	67.0034	66.2105
Model	co-CAE	c-SAE				c-p-SAE			
Model Compression Ratio	37.21%	none	26.01%	41.54%	58.30%	none	36.03%	45.97%	55.11%
Indoor	32.8001	75.8872	75.4571	74.7633	73.8041	72.6344	72.1874	71.0801	70.3342
City	31.5367	75.0090	74.6623	74.1342	73.3103	71.9523	71.6345	70.7638	70.1020
Roller-Coaster	25.0956	71.8376	71.0186	71.0123	70.8327	69.7491	68.4566	68.1483	67.3970
Football	26.7573	72.9037	71.9642	71.6107	70.9362	69.9907	69.6016	69.2106	68.6433

limit the speed of training. Instead, when using p-SAE with partial mesh, this will speed up significantly. It does come with a cost of slightly decreased PSNR result. On the other hand, comparing c-SAE models to SAE models, we also find that the model size slightly increases. This is because of the orthogonal regularizer and spectral normalization used in compressive sensing, the cost for the improved PSNR results. Thus, there is a clear trade-off here that should be taken into consideration when choosing different models.

Voronoi VMAF In addition to PSNR, we also evaluate the result using the recently developed Video Multimethod Assessment Fusion (VMAF) metric [34]. The VMAF metric is a support vector machine (SVM) regressor which assigns weights to each elementary metric. The final metric could preserve all the strengths of the individual metrics and deliver a more accurate final score. The elementary metrics include visual information fidelity (VIF) [39], detail loss metric (DLM) [33], and mean co-located pixel difference.

VMAF was originally developed for traditional 2D content. To properly use VMAF in our evaluation, we further adopt the Voronoi objective metric [26]-based VMAF that is developed recently to evaluate the quality of experience for spherical videos. In this evaluation, a spherical video is divided into M patches using the spherical Voronoi diagram [15] of M evenly distributed points on the sphere [25]. Table 3 shows the Voronoi VMAF (VI-VMAF) scores. It


(a) Original frame samples

(b) **CAE frame samples**

(c) SAE frame samples

(d) c-SAE frame samples

Figure 5: Sample frames from four videos

is clear from the table that all SAE models outperform CAE significantly, even with a high compression rates. Overall, these results are consistent with the PSNR results. Compared to the PSNR based results, the score difference between different model outputs is more pronounced with VI-VMAF.

Visual Comparisons Figure 5 shows some examples of frames extracted from these videos. We can observe that, for CAE, since it is trained by first projecting a spherical surface onto a rectangular plane and then dividing that into patches, the boundary and details of the image are relatively still clear, but the image clarity and colors are affected. For SAE, the quality of the original frame is well maintained, the borders are clearer and the clarity of the picture is high. We did not include p-SAE as it only shows a partial view like Figure 3. These results indicate that spherical processing of 360-degree videos is more effective than the projection plus the conventional 2D processing approach, potentially preserving more critical spherical information during the processing.

7 CONCLUSION

Spherical video content is getting more and more popular in various applications. Compared to the traditional 2D video, spherical video content not only demands more bandwidth to transmit, but also more efficient techniques for content processing, e.g., for video analytics engines. In this work, we explore a new approach to effectively process spherical content. Compared to the traditional approach where a spherical frame is mapped to a 2D space, we have investigated processing the spherical content directly using a spherical autoencoder (SAE). Motivated by the fact that mobile devices are widely used for video accesses, we further propose two optimizations to make SAE better fit for resource constrained mobile devices while maintaining the video quality. Our experimental results show that our proposed approaches can significantly outperform the traditional approach and both the partial view supported SAE, i.e., p-SAE, and compressive sensing integrated SAE, i.e., c-SAE, are effective in delivering high quality videos.

ACKNOWLEDGMENTS

The authors thank the anonymous reviewers for their constructive comments. This work was supported in part by NSF grants CNS-1943250, CNS-2007153, CNS-2200042, IIS-2107200, CPS-2038658, and by the Commonwealth Cyber Initiative, an investment in the advancement of cyber R&D, innovation, and workforce development.

REFERENCES

[1] 1998. JPEG 2000. https://jpeg.org/jpeg2000/

[2] 1998. MPEG. https://www.mpegstandards.org/

[3] 1999. Equirectangular Projection. http://mathworld.wolfram.com/EquirectangularProjection.html.

[4] 2008. H.264. https://www.itu.int/rec/T-REC-H.264

[5] 2008. H.265. https://www.itu.int/rec/T-REC-H.265

[6] 2011. Matterport. https://matterport.com/industries/real-estate

[7] 2017. Bringing pixels front and center in VR video. https://blog.google/products/google-vr/bringing-pixels-front-and-center-vr-video/.

[8] 2019. Cubic Projection. http://wiki.panotools.org/Cubic_Projection.

[9] 2019. Rectilinear Projection. https://wiki.panotools.org/Rectilinear_Projection.

[10] 2021. CAE. https://github.com/alexandru-dinu/cae

[11] 2021. Global - 2021 Forecast Highlights. https://www.cisco.com/c/dam/m/en_us/solutions/service-provider/vni-forecast-highlights/pdf/Global_2021_Forecast_Highlights.pdf

[12] 2021. Starline. https://blog.google/technology/research/project-starline/

[13] Pinar Akyazi and Touradj Ebrahimi. 2019. Learning-Based Image Compression using Convolutional Autoencoder and Wavelet Decomposition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

[14] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein GAN. arXiv:1701.07875 [stat.ML]

[15] Franz Aurenhammer. 1991. Voronoi Diagrams—a Survey of a Fundamental Geometric Data Structure. *ACM Comput. Surv.* 23, 3 (Sept. 1991), 345–405. https://doi.org/10.1145/116873.116880

[16] Yanan Bao, Huasen Wu, Tianxiao Zhang, Albara Ah Ramli, and Xin Liu. 2016. Shooting a moving target: Motion-prediction-based transmission for 360-degree videos. In *2016 IEEE International Conference on Big Data (Big Data)*. IEEE, 1161–1170.

[17] John R. Baumgardner and Paul O. Frederickson. 1985. Icosahedral Discretization of the Two-Sphere. *SIAM J. Numer. Anal.* 22, 6 (Dec. 1985), 1107–1115. https://doi.org/10.1137/0722066

[18] Amir Beck and Marc Teboulle. 2009. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences* 2, 1 (2009), 183–202.

[19] Ashish Bora, Ajil Jalal, Eric Price, and Alexandros G Dimakis. 2017. Compressed sensing using generative models. In *International Conference on Machine Learning*. PMLR, 537–546.

[20] Emmanuel J Candès, Justin Romberg, and Terence Tao. 2006. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on information theory* 52, 2 (2006), 489–509.

[21] Zhibo Chen, Tianyu He, Xin Jin, and Feng Wu. 2018. Learning for Video Compression. *CoRR* abs/1804.09869 (2018). arXiv:1804.09869 http://arxiv.org/abs/1804.09869

[22] Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto. 2018. Deep Convolutional AutoEncoder-based Lossy Image Compression. In *2018 Picture Coding Symposium (PCS)*. 253–257. https://doi.org/10.1109/PCS.2018.8456308

[23] Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto. 2018. Performance Comparison of Convolutional AutoEncoders, Generative Adversarial Networks and Super- Resolution for Image Compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

[24] Taco S. Cohen, Mario Geiger, Jonas Köhler, and Max Welling. 2018. Spherical CNNs. *CoRR* abs/1801.10130 (2018). arXiv:1801.10130 http://arxiv.org/abs/1801.10130

[25] Simone Croci, Sebastian Knorr, Lutz Goldmann, and Aljosa Smolic. 2017. A framework for quality control in cinematic VR based on Voronoi patches and saliency. In *2017 International Conference on 3D Immersion (IC3D)*. 1–8. https://doi.org/10.1109/IC3D.2017.8251907

[26] Simone Croci, Cagri Ozcinar, Emin Zerman, Julián Cabrera, and Aljosa Smolic. 2019. Voronoi-based Objective Quality Metrics for Omnidirectional Video. In *2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX)*. 1–6. https://doi.org/10.1109/QoMEX.2019.8743345

[27] Simon Erridge, Derek KT Yeung, Hitendra RH Patel, and Sanjay Purkayastha. 2019. Telementoring of surgeons: a systematic review. *Surgical innovation* 26, 1 (2019), 95–111.

[28] Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia, and Kostas Daniilidis. 2018. Learning SO(3) Equivariant Representations with Spherical CNNs. arXiv:1711.06721 [cs.CV]

[29] Mário AT Figueiredo, Robert D Nowak, and Stephen J Wright. 2007. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *IEEE Journal of selected topics in signal processing* 1, 4 (2007), 586–597.

[30] Amirhossein Habibian, Ties van Rozendaal, Jakub M Tomczak, and Taco S Cohen. 2019. Video compression with rate-distortion autoencoders. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 7033–7042.

[31] Chiyu Max Jiang, Jingwei Huang, Karthik Kashinath, Prabhat, Philip Marcus, and Matthias Nießner. 2019. Spherical CNNs on Unstructured Grids. *CoRR* abs/1901.02039 (2019). arXiv:1901.02039 http://arxiv.org/abs/1901.02039

[32] Na Li and Yao Liu. 2022. Applying VertexShuffle Toward 360-Degree Video Super-Resolution. In *Proceedings of the 32nd ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*.

[33] Songnan Li, Fan Zhang, Lin Ma, and King Ngi Ngan. 2011. Image Quality Assessment by Separately Evaluating Detail Losses and Additive Impairments. *IEEE Transactions on Multimedia* 13, 5 (2011), 935–949. https://doi.org/10.1109/TMM.2011.2152382

[34] Z. Li, A. Aaron, I. Katsavounidis, A. Moorthy, and M. Manohara. [n.d.]. Toward a practical perceptual video quality metric. https://netflixtechblog.com/toward-a-practical-perceptual-video-quality-metric-653f208b9652.

[35] Microsoft. 2016. Holoportation. https://www.microsoft.com/en-us/research/project/holoportation-3/ (Access:2022-02).

[36] Mark Roman Miller, Fernanda Herrera, Hanseul Jun, James A Landay, and Jeremy N Bailenson. 2020. Personal identifiability of user tracking data during observation of 360-degree VR video. *Scientific Reports* 10, 1 (2020), 1–10.

[37] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. 2018. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957* (2018).

[38] Oren Rippel, Sanjay Nair, Carissa Lew, Steve Branson, Alexander G. Anderson, and Lubomir Bourdev. 2018. Learned Video Compression. arXiv:1811.06981 [eess.IV]

[39] H.R. Sheikh and A.C. Bovik. 2006. Image information and visual quality. *IEEE Transactions on Image Processing* 15, 2 (2006), 430–444. https://doi.org/10.1109/TIP.2005.859378

[40] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. 2016. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1874–1883.

[41] Liyang Sun, Fanyi Duanmu, Yong Liu, Yao Wang, Yinghua Ye, Hang Shi, and David Dai. 2018. Multi-path multi-tier 360-degree video streaming in 5G networks. In *Proceedings of the 9th ACM Multimedia Systems Conference*. 162–173.

[42] Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. 2017. Lossy Image Compression with Compressive Autoencoders. arXiv:1703.00395 [stat.ML]

[43] Gregory K. Wallace. 1991. The JPEG Still Picture Compression Standard. *Commun. ACM* 34, 4 (April 1991), 30–44. https://doi.org/10.1145/103085.103089

[44] Chao-Yuan Wu, Nayan Singhal, and Philipp Krähenbühl. 2018. Video Compression through Image Interpolation. arXiv:1804.06919 [cs.CV]

[45] Shuochao Yao, Jinyang Li, Dongxin Liu, Tianshi Wang, Shengzhong Liu, Huajie Shao, and Tarek Abdelzaher. 2020. *Deep Compressive Offloading: Speeding up Neural Network Inference by Trading Edge Computation for Network Latency*. Association for Computing Machinery, New York, NY, USA, 476–488. https://doi.org/10.1145/3384419.3430898

[46] Shuochao Yao, Jinyang Li, Dongxin Liu, Tianshi Wang, Shengzhong Liu, Huajie Shao, and Tarek Abdelzaher. 2021. Deep Compressive Offloading: Speeding Up Edge Offloading for AI Services. *GetMobile: Mobile Computing and Communications* 25, 1 (2021), 39–42.