Learning for Video Compression

Zhibo Chen, *Senior Member, IEEE,* Tianyu He, Xin Jin, Feng Wu, *Fellow, IEEE*

***Abstract*—One key challenge to learning-based video compression is that motion predictive coding, a very effective tool for video compression, can hardly be trained into a neural network. In this paper we propose the concept of VoxelCNN which includes motion extension and hybrid prediction networks. VoxelCNN can model spatiotemporal coherence to effectively perform predictive coding inside the learning network. On the basis of VoxelCNN, we further explore a learning based framework for video compression with additional components of iterative analysis/synthesis, binarization, etc. Experiment results demonstrate the effectiveness of the proposed scheme. Although entropy coding and complex configurations are not employed in this paper, we still demonstrate superior performance compared with MPEG-2 and achieve comparable results with H.264 codec. The proposed learning based scheme provides a possible new direction to further improve compression efficiency and functionalities of future video coding.**

***Index Terms*—video coding, learning, VoxelCNN.**

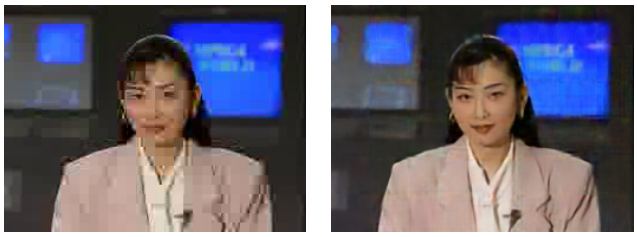
I. Introduction

VIDEO occupies about 75% of the data transmitted on world-wide networks and that percentage has been steadily growing and is projected to continue to grow further [1]. Meanwhile the introduction of ultra-high definition (UHD), high dynamic range (HDR), wide color gamut (WCG), high frame rate (HFR) and future immersive video services have dramatically increased the challenge. Therefore, the need for highly efficient video compression technologies are always pressing and urgent.

Since the proposal of concept of hybrid coding by Habibi in 1974 [2] and hybrid spatial-temporal coding framework by Forchheimer in 1981 [3], this Hybrid Video Coding (HVC) framework has been widely adopted into most popular existing image/video coding standards like JPEG, H.261, MEPG-2, H.264, and H.265, etc. The video coding performance improves around 50% every 10 years under the cost of increased computational complexity and memory. And now it encountered great challenges to further significantly improve the coding efficiency and to deal efficiently with novel sophisticated and intelligent media applications such as face/body recognition, object tracking, image retrieval, etc. So there exist strong requirements to explore new video coding directions and frameworks as potential candidates for future video coding schemes, especially considering the outstanding development of machine learning technologies.

Recently, some learning-based image compression schemes have been proposed [4]–[6] with types of neural networks such as auto-encoder, recurrent network and adversarial network, demonstrating a new direction of image/video compression.

Zhibo Chen, Tianyu He, Xin Jin and Feng Wu are with University of Science and Technology of China, Hefei, Anhui, 230026, China, (e-mail: chenzhibo@ustc.edu.cn)



(a) MPEG-2 (b) Ours

Fig. 1: Visualization of reconstructed video compared with MPEG-2 under the compression ratio of about 575. It is worth noting that entropy coding is not employed for our results, even though it is commonly done in standard video compression codecs, and may bring more gain about $5\% \sim 57\%$ [4] performance improvement.

For example, the first work of learning-based image compression [4], [7] was introduced in 2016 and demonstrates their better performance compared with the first image coding standard JPEG.

However, all learning-based methods proposed so far were developed for still image compression and there is still no published work for video compression. One key bottleneck is that motion compensation, as a very effective tool for video coding, can hardly be trained into a neural network (or would be tremendously more complex than conventional motion estimation) [8]. Therefore, there exist some research work on replacing some modules (e.g., sub-pel interpolation, up-sampling filtering, post-processing, etc.) in HVC framework by learning-based modules [9]–[11]. However, such partial replacements are still under the heuristically optimized HVC framework without capability to successfully deal with aforementioned challenges.

In this paper, we first propose the concept of VoxelCNN by modeling spatiotemporal coherence to effectively deal with the aforementioned bottleneck of motion compensation trained into a neural network, and explore a learning-based framework for video compression. Specifically, we construct a neural network to predict each block of video sequence conditioned on previously reconstructed frame as well as the reconstructed blocks above and to the left of current block. The difference between predicted and original pixels is then analyzed and synthesized iteratively to produce a compact discrete representation. Consequently, the bitstream is obtained that can be used for storage or transmission. To the best of our knowledge, this is the first fully learning-based video compression framework.

II. Related Work

Recently there are two kinds of research work trying to apply machine learning techniques into image/video compression problem, one is Codec-based improvements which

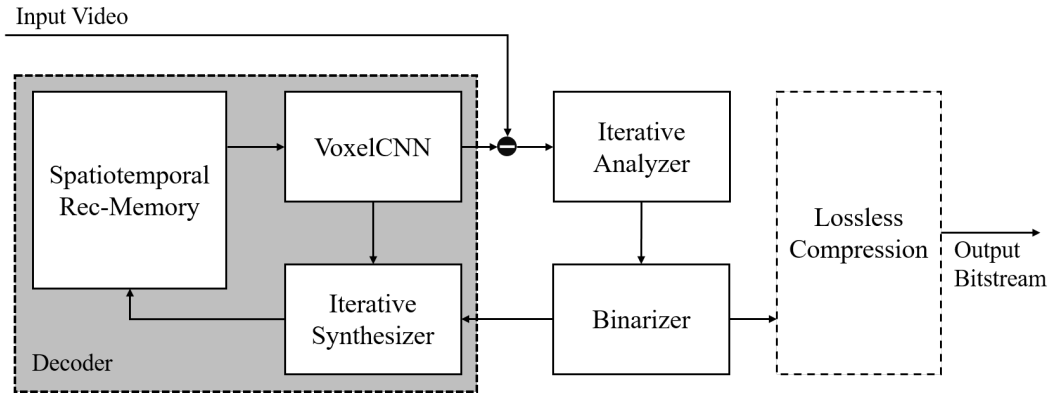


Fig. 2: The pipeline of the proposed learning based encoder. The encoder converts video into a compressed format (bitstream), then the decoder converts bitstream back into a reconstructed video. Note that, at decoding time, we only have access to the reconstructed data (Spatiotemporal Rec-Memory) instead of original data, therefore, the decoder are included in the encoder to produce reconstructed data for sequentially encoding. We does not employ lossless compression (entropy coding) in this paper, it can be complemented as dashed line in the figure. The details of each components are described in Section IV.

introduces learning based optimization modules combined with traditional image/video codecs, another is pure Learning-based compression framework which are mainly focused on learning based image compression schemes in current stage.

A. Codec-based Improvements

Lossy image/video codecs, such as JPEG and High Efficiency Video Coding (HEVC) [12], give a profound impact on image/video compression. Considering the recent success of neural network-based methods in various domains, a stream of CNN-based improvements for these codecs have been proposed to further increase coding efficiency. Most of these works focus on enhancing the performance [10], [13] or reducing the complexity [14], [15] of codec by replacing manually designed function with learning-based approach. Similarly, a series of works adopt CNN in post-processing to reduce the artifacts of compression [16]–[18]. Encouraged by positive results in domain of super-resolution, another line of work encodes the down-sampled content with codec and then up-samples the decoded one by CNN for reconstruction [11], [19].

B. Learning-based Image Compression

End-to-end image compression has surged for almost a year, opening up a new avenue for lossy compression. The majority of them adopt an autoencoder-like scheme.

Ballé *et al.* relaxed discontinuous quantization step with additive uniform noise to alleviate the non-differentiability, and developed an effective non-linear transform coding framework in the context of compression [5]. Similarly, Theis *et al.* replaced quantization with a smooth approximation and optimized a convolutional auto-encoder with an incremental training strategy [6]. Dumas *et al.* incorporates a stochastic hyperparameter to control a competition mechanism between image patches [20]. As a fast-growing architecture in the field of neural network, GANs have also been proved to be effective on image compression in practice [21], [22].

In the works of [4], [7], [23], [24], a discrete and compact representation is obtained by applying a quantization to the bottleneck of auto-encoder. To achieve variable bit rates, the model progressively analyzes and synthesizes residual errors with several auto-encoders. Progressive codes are essential to Rate-Distortion Optimization (RDO), since a higher quality can be attained by adding additional bits. On the basis of these progressive analyzer, Baig *et al.* introduce an inpainting scheme that exploits spatial coherence exhibited by neighboring blocks to reduce redundancy in image [25].

Encouraged by the aforementioned achievements, we take advantage of the successful exploration in learning-based image compression and further explore a learning-based framework for video compression.

III. Spatiotemporal Modeling with VoxelCNN

In a typical scene, there are great spatiotemporal correlations between pixels in a video sequence. One effective approach to de-correlate highly correlated neighboring signal samples is to model the spatiotemporal distribution of pixels in the video. Since videos are generally encoded and decoded in a sequence, the modeling problem can be solved by estimating a product of conditional distributions (conditioned on reconstructed values), instead of modeling a joint distribution of the pixels. Oord *et al.* [26] introduced PixelCNN for generative image modeling, which can be regarded as a kind of intra-frame prediction for de-correlating neighboring pixels inside one frame. To further de-correlate neighboring frames of the same sequence, we here propose VoxelCNN that sequentially predicts the blocks in a video sequence.

A. Model

We consider the circumstance where videos are encoded and decoded frame-by-frame in chronological order, and block-by-block in a raster scan order. We define a video sequence $\{f^1, f^2, ..., f^I\}$ as a collection of m frames that are ordered along the time axis. Each frame comprises n blocks sequentialized in a raster scan order, formulated as $f^i = \{b^i_1, b^i_2, ..., b^i_J\}$.

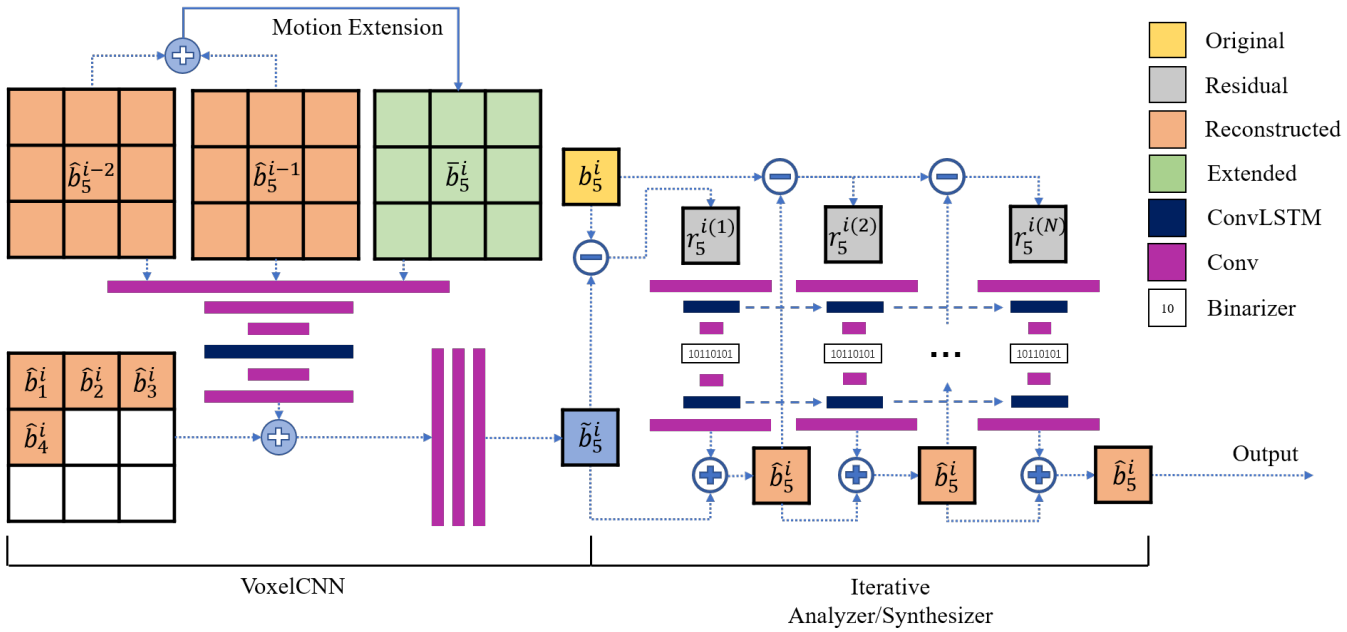


Fig. 3: Detailed architecture of our video compression scheme.

Inspired by PixelCNN, we can factorize the distribution of each frame:

$$p(f^i) = \prod_{j=1}^{J} p(b_j^i \mid f^1, ..., f^{i-1}, b_1^i, ..., b_{j-1}^i) \tag{1}$$

where $p(b_j^i \mid f^1, ..., f^{i-1}, b_1^i, ..., b_{j-1}^i)$ is the probability of the j^{th} block b_j^i in the i^{th} frame, given the previous frames $f^1, ..., f^{i-1}$ as well as the blocks $b_1^i, ..., b_{j-1}^i$ above and to the left of the current block.

Note that, after the transmission of bitstream, we only have access to the reconstructed data instead of original data at decoding stage. Therefore, as Figure 2 illustrates, we sequentially analyze and synthesize each block, saving all the reconstructed frames $\{\hat{f}^1, \hat{f}^2, ..., \hat{f}^I\}$ and blocks $\{\hat{b}_1^i, \hat{b}_2^i, ..., \hat{b}_J^i\}$ in Spatiotemporal Rec-Memory. Instead of conditioning on the previously generated content as PixelCNN does, VoxelCNN learns to predict the conditional probability distribution $p(b_j^i \mid \hat{f}^1, ..., \hat{f}^{i-1}, \hat{b}_1^i, ..., \hat{b}_{j-1}^i)$ conditioned on previously reconstructed content.

Spatiotemporal modeling is essential to our learning based video compression pipeline, as it offers excellent decorrelation capability for sources. The experiments in Section V demonstrate its effectiveness compared with pure spatial or temporal modeling schemes.

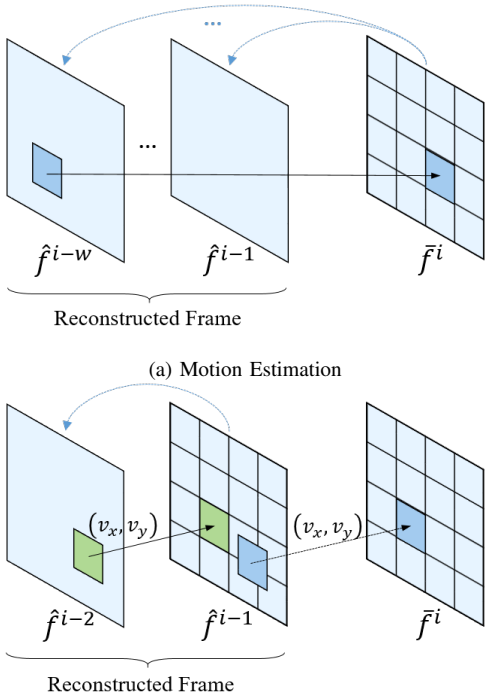
B. Architectural Components

Considering motion information in the temporal direction, we assume that the pixels in neighboring frames are correlated along motion trajectory and there usually exists a linear or non-linear displacement between them. Therefore, as illustrated in Figure 3, we first employ motion extension for approximating linear part of motion displacement, yielding an extended frame $\bar{f}^i$. The non-linear part of motion displacement and the reconstructed blocks (above and to the left of current block) are then jointly modeled with convolutional neural network. This scheme leverages spatiotemporal coherence simultaneously to provide a prediction progressively, and reduces the amount of information required to be transmitted without any other side information (e.g., motion vector).

Motion Extension The objective of motion extension is to extend motion trajectory obtained from previous two reconstructed frames $\hat{f}^{i-2}, \hat{f}^{i-1}$. Let $v_x, v_y, x, y \in \mathbb{Z}$, we first determine a motion vector (v_x, v_y) between $\hat{f}^{i-2}$ and $\hat{f}^{i-1}$ by block matching with 4×4 block size. When building each 4×4 block $\bar{b}^i$ in $\bar{f}^i$, the motion vectors of corresponding 4×4 block in frame $\hat{f}^{i-1}$ are ultilized to locate and copy the data from frame $\hat{f}^{i-1}$, as illustrated in Figure 4. For instance, the values of block $\bar{b}^i$ centered in (x, y) in extended frame $\bar{f}^i$ are copied from $\hat{b}^{i-1}$ centered in $(x - v_x, y - v_y)$. We repeat this operation to obtain complete values of $\bar{f}^i$ as extended frame. It is important to note that there are two intrinsical differences between motion extension and motion estimation [27] used in traditional video coding schemes:

- We employ motion extension as preprocessing to generate an extended input of VoxelCNN which utilizes former reconstructed reference frames to generate current coding block. This is essentially different from motion estimation used in traditional codecs which utilizes current coding block to search matching block in reconstructed reference frames.
- In general, traditional codecs transmit motion vectors as side information since they indicate where the estimation of current coding block is directly from. By contrast, our proposed scheme doesn't need to transmit motion vectors.

Hybrid Prediction In particular, we employ a convolutional neural network that accepts extended frame as its input and outputs an estimation of current block. Previous works [28] have shown that ConvLSTM has the potential to model tem-



(a) Motion Estimation

(b) Motion Extension

Fig. 4: Comparison between motion estimation and motion extension. The query frame (the start point of blue dashed arrow) is divided into blocks, and each of the blocks is compared with the blocks in the former frames (the end point of blue dashed arrow) to form a motion vector (the black arrow). The black dashed arrow in (b) has the same value as the black arrow, which direct where should the values in $\bar{f}^i$ be copied from. Note that, the calculation of motion vectors in (b) only depends on reconstructed content, resulting the omission of transmission, which differs from (a) that transmits motion vectors as side information.

poral correlation while reserving spatial invariance. Moreover, residual learning (Res-Block) [29] is a powerful technique proposed to train very deep convolutional neural network. We here exploit the strength of ConvLSTM and Res-Block to sequentially connect features of $\hat{f}^{i-2}$, $\hat{f}^{i-1}$ and $\bar{f}^{i}$. An estimation of current frame, as well as the blocks above and to the left of current block, is then fed into several Convolution-BatchNorm-ReLU modules [30]. As described in Section III-A, the spatiotemporal coherence is modeled concurrently in this scheme, producing a prediction $\tilde{b}_j^i$ of current block.

In this section, we define the form of VoxelCNN and then describe the detailed architecture of VoxelCNN [1]. In the next section, we give a comprehensive explanation of our framework that employ VoxelCNN as predictive coding.

IV. Learning for Video Compression

Our scheme for video compression can be divided into three components: predictive coding, iterative analysis/synthesis and binarization. Note that entropy coding is not employed in this paper, even though it is commonly done in standard video compression codecs, and may give more performance gain. In this section, we give details about each component in our scheme and introduce various modes used in our framework.

[1]We give all parameters in the Appendix A.

Predictive Coding. We utilize VoxelCNN for predictive coding, to create a prediction of a block $\tilde{b}_j^i$ of the current frame based on previously encoded frames as well as the blocks above and to the left of it. This prediction is subtracted from original value to form a residual r_j^i. A successful prediction may decrease the energy in the residual compared with original frame, and the data can be represented with fewer bits [31]. Our learning objective for the VoxelCNN can be defined as follows:

$$L_{vcnn} = \frac{1}{B \times J} \sum_{i=1}^{B} \sum_{j=1}^{J} (b_j^i - \tilde{b}_j^i)^2, \tag{2}$$

where B is batch size, J is the total number of blocks in each frames, and $\tilde{b}_j^i$ denote the output of VoxelCNN, the superscript and subscript refer to j^{th} block in the i^{th} frame respectively.

Iterative Analysis/Synthesis. Several works put efforts on directly encoding pixel values [4], [5]. By contrast we encode the difference between the predicted and the original pixel values. We adopt the model of Toderici *et al.* [4], which is composed of several LSTM-based auto-encoders with connections between adjacent stages. The residuals between reconstruction and target are analyzed and synthesized iteratively to provide a variable-rate compression. Each stage n produces a compact representation required to be transmitted of input residual $r_j^{i(n)}$. We can represent the loss function of iterative analysis/synthesis as follows:

$$L_{res} = \frac{1}{B \times J} \sum_{i=1}^{B} \sum_{j=1}^{J} (r_j^{i(1)} - \sum_{m=1}^{S} \hat{r}_j^{i(m)})^2, \tag{3}$$

where

$$r_j^{i(1)} = b_j^i - \tilde{b}_j^i, \tag{4}$$

$$r_j^{i(n)} = r_j^{i(1)} - \sum_{m=1}^{n-1} \hat{r}_j^{i(m)}, \tag{5}$$

and $\hat{r}_j^{i(n)}$ indicates the output of n^{th} stage, S is the total number of stages (8 in this paper).

Binarization. Binarization is actually where significant amount of data reduction can be attained, since such a many-to-one mapping reduce the number of possible signal values at the cost of introducing some numerical errors. Unfortunately, binarization is an inherently non-differentiable operation that cannot to be optimized with gradient-based techniques. However, some researchers have tackled this problem by mathematical approximation [4]–[6]. Following Raiko *et al.* [32] and Toderici *et al.* [7], we add a probabilistic quantization noise ϵ for the forward pass and keep the gradients unchanged for the backward pass:

$$\epsilon = \begin{cases} 1 - c_{in}, & \text{with probability } \frac{1+c_{in}}{2} \\ -c_{in} - 1, & \text{otherwise,} \end{cases} \tag{6}$$

where $c_{in} \in [-1, 1]$ represents the input of binarizer. The output of binarizer can thus be formulated as $c_{out} = c_{in} + \epsilon$. Note that, the binarizer takes no parameter in our scheme.

At test phase, in order to generate extended frame, we need to encode the first two frames directly (without VoxelCNN). Similarly, we encode the first row and the first column of

(a) Akiyo (b) BasketballDrill (c) Claire (d) Drawing

(e) Foreman (f) FourPeople (g) Pairs (h) Silent

Fig. 5: Thumbnail of test sequences.

blocks in each frame only conditioned on previous frames $\{\hat{f}^1, ..., \hat{f}^{i-1}\}$ since they have no spatial neighborhood to be used for predication.

A. Objective Function

Although there exist potential to use various metrics in this framework for gradient-based optimization, we here employ MSE as the metric for simplicity. During the training phase, VoxelCNN, iterative analyzer/synthesizer and binarizer are jointly optimized to learn a compact representation of input video sequence. The overall objective can be formulated as:

$$L_{total} = L_{vcnn} + L_{res}, \tag{7}$$

where L_{vcnn} and L_{res} represent the learning objective of VoxelCNN and iterative analysis/synthesis respectively.

B. Spatially Progressive Coding

We design a spatially progressive coding scheme in the test phase, by performing various number of iterations determined for each block by a quality metric (e.g. PSNR). This spatially progressive coding scheme enables the functionality of adaptively allocating different bits to different blocks, which can be applied to further improving coding performance similar to rate control in traditional video coding framework. In this paper, we perform this spatially progressive coding scheme in the simplest way, that is to continue to progressively encode residual when the MSE between reconstructed block and the original block is lower than a threshold.

C. Temporally Progressive Coding

In addition to spatially progressive coding, we also employ a temporally progressive coding scheme that exploits invariance between adjacent frames to progressively determine coding methods of blocks in each frame. We can achieve this by optionally encoding each block according to a specific metric. Also, we implement this scheme in a simplest way in this paper, similar to skip mode defined in traditional video coding framework [33]. In particular, for each block, we first determine whether the block should be encoded or not by calculating MSE between b_j^i and b_j^{i-1}. The encoding process is ignored if the MSE is lower than a threshold and a flag (a bit) is transmitted to decoder for indicating the selected mode. The flag is encoded by arithmetic coding as the only overhead ($< 1\%$ of bitstream) in our framework. Note that, the encoding of flag have no effect on the training of entire model since it is a out-loop operation.

V. Experimental Results

In this section, we first evaluate the performance of VoxelCNN, then we compare our video compression scheme with traditional video codecs.

Dataset. In general, neighboring frames of the same sequence is highly correlated, resulting in a limited diversity. Hence, we first collect a image dataset for pre-training and then fine-tune on video sequences. The image dataset contains $530,000$ color images collected from Flickr. Each image is down-sampled to 256x256 to enhance the complexity of texture. During training time, we further perform data augmentation including random rotation and color perturbation. The video dataset contains $10,000$ sequences sampled from UCF-101 [34]. For test set, we collect 8 representative sequences from MPEG/VCEG common test sequences [35] as demonstrated in Figure 6, including various content categories (e.g. global motion, local motion, different motion amplitude, different texture complexity, etc.) In line with the image dataset, all sequences are resized to 256x192 according to 4:3 aspect ratio. Note that, our method used in this paper take 256x192 as a coding unit, and can be easily extended to a high-resolution sequence.

Implementation Details. We adopt a 32x32 block size for VoxelCNN and iterative analyzer/synthesizer in our paper. Although variable block size coding typically demonstrates higher performance than fixed block size in traditional codec [36], we just verify the effectiveness of our method with fixed block size for simplicity. We first pretrain VoxelCNN on aforementioned video dataset using Adam optimizer [37] with 10^{-3} learning rate and trained with 20 epochs. The learning rate is decreased by 10^{-1} every 5 epochs. All parameters are initialized by Kaiming initialization [38]. Our iterative

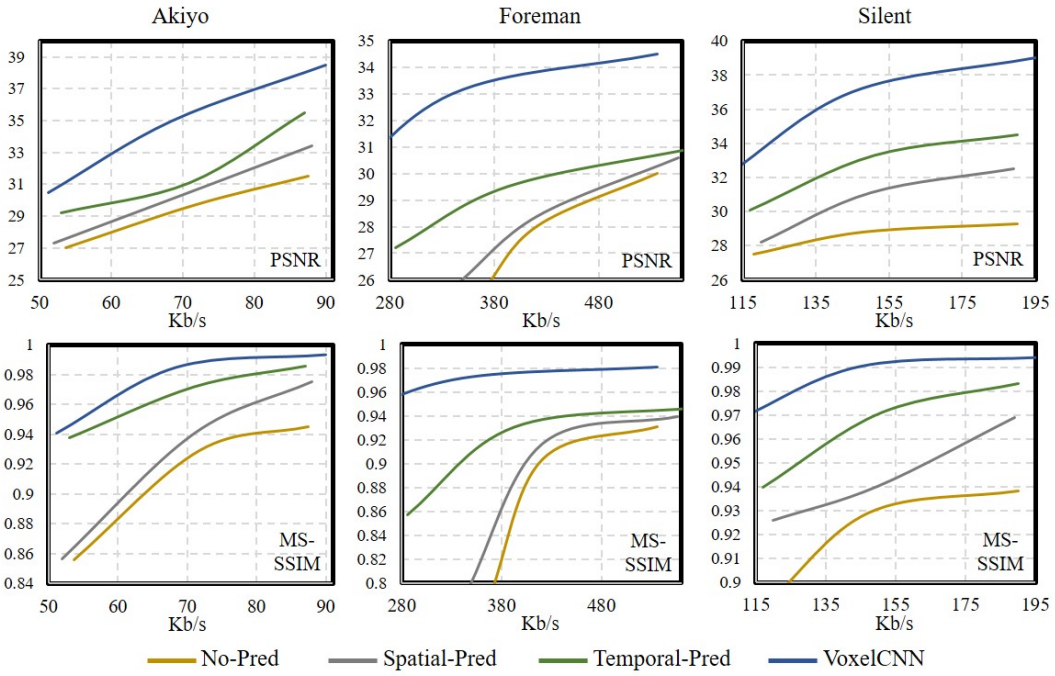


Fig. 6: Quantitative analysis of VoxelCNN. Each column represents the PSNR/MS-SSIM performance on test sequence. We can observe that VoxelCNN leverages spatiotemporal dependencies, and surpass the performance of other prediction schemes.

analyzer/synthesizer model is pretrained on 32x32 image blocks randomly cropped from collected image dataset using Adam optimizer with 10^{-3} learning rate. We train iterative analyzer/synthesizer with 40 epochs, dropping learning rate by 10^{-1} every 10 epochs. The pretrained VoxelCNN and pretrained iterative analyzer/synthesizer are then merged and tuned on video dataset with 10^{-4} learning rate and trained with 10 epochs.

Baseline. As the first work of learning-based video compression, we compare our scheme with two representative HVC codecs: MPEG-2 (v1.2) [39] and H.264 (JM 19.0) [40] in our experiments. Both of them take YUV 4:2:0 video format as input and output. We encode the first frame as Intra-frame mode and Predicted-frame mode for the remaining with fixed QP. Rate control is disabled for both codecs.

Evaluation Metric. To assess the performance of our model, we report PSNR between the original videos and the reconstructed ones. Following Toderici *et al.* [4], we also adopt Multi-Scale Structural Similarity (MS-SSIM) [41] as a perceptual metric. Note that, these metrics are applied both on RGB channels and the reported results in this paper are averaged on each test sequence.

TABLE I: Equivalent bit-rate savings (based on PSNR) of different learning based prediction modes with respect to No-Pred mode.

Sequences	Spatial-Pred	Temporal-Pred	VoxelCNN
Akiyo	-9.57%	-42.56%	-66.31%
Foreman	-3.82%	-18.47%	-56.60%
Silent	-12.12%	-53.23%	-72.84%
Average	-6.78%	-30.06%	-57.10%

TABLE II: Equivalent bit-rate savings (based on PSNR) of Our Scheme with respect to modern codecs.

Sequences	MPEG-2	H.264
Akiyo	-52.57%	-6.12%
BasketballDrill	-28.31%	+31.99%
Claire	-58.41%	-40.25%
Drawing	-30.41%	+56.71%
Foreman	-39.73%	+56.67%
FourPeople	-80.01%	-11.71%
Pairs	-41.87%	+8.85%
Silent	-56.01%	-30.74%
Average	-48.415%	+8.175%

A. Quantitative Analysis of VoxelCNN

Our proposed VoxelCNN model leverages spatiotemporal coherence and provide a hybrid prediction $\tilde{b}_j^i$ conditioned on previously reconstructed frames $\hat{f}^1, ..., \hat{f}^{i-1}$ and blocks $\hat{b}_1^i, ..., \hat{b}_{j-1}^i$. To evaluate the impact of each condition, we train our VoxelCNN conditioned on individual dependency respectively. We refer 'Spatial-Pred' as the model trained only conditioned on blocks $\hat{b}_1^i, ..., \hat{b}_{j-1}^i$, 'Temporal-Pred' as the model trained only conditioned on frames $\hat{f}^1, ..., \hat{f}^{i-1}$, 'No-Pred' as the model trained on none of these dependencies. We compare each case with VoxelCNN on the first 30 frames of three representive sequences including local motion (Akiyo), global motion (Foreman) and different motion amplitude (Silent).

Figure 6 demonstrates efficiency of the proposed VoxelCNN

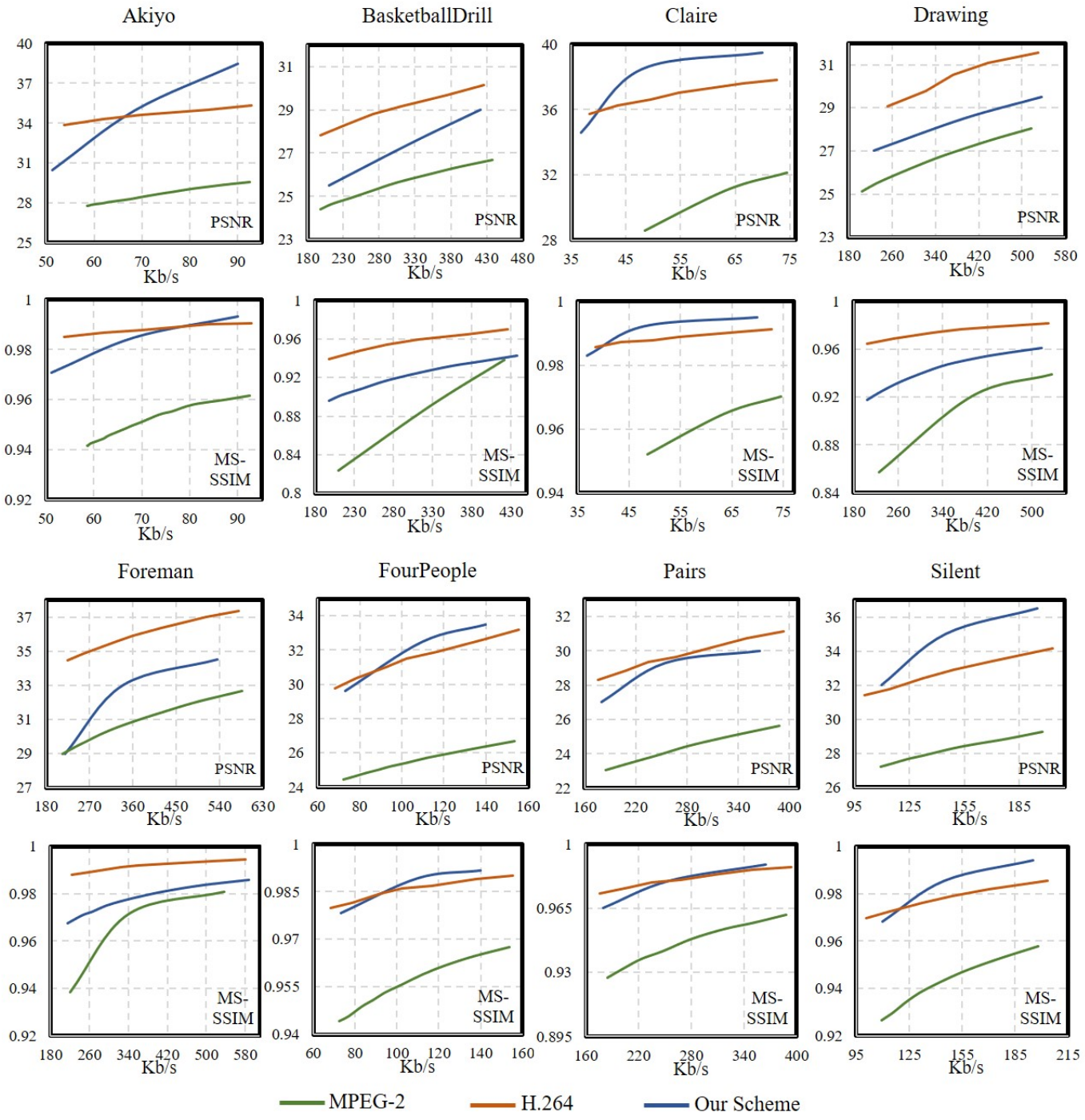


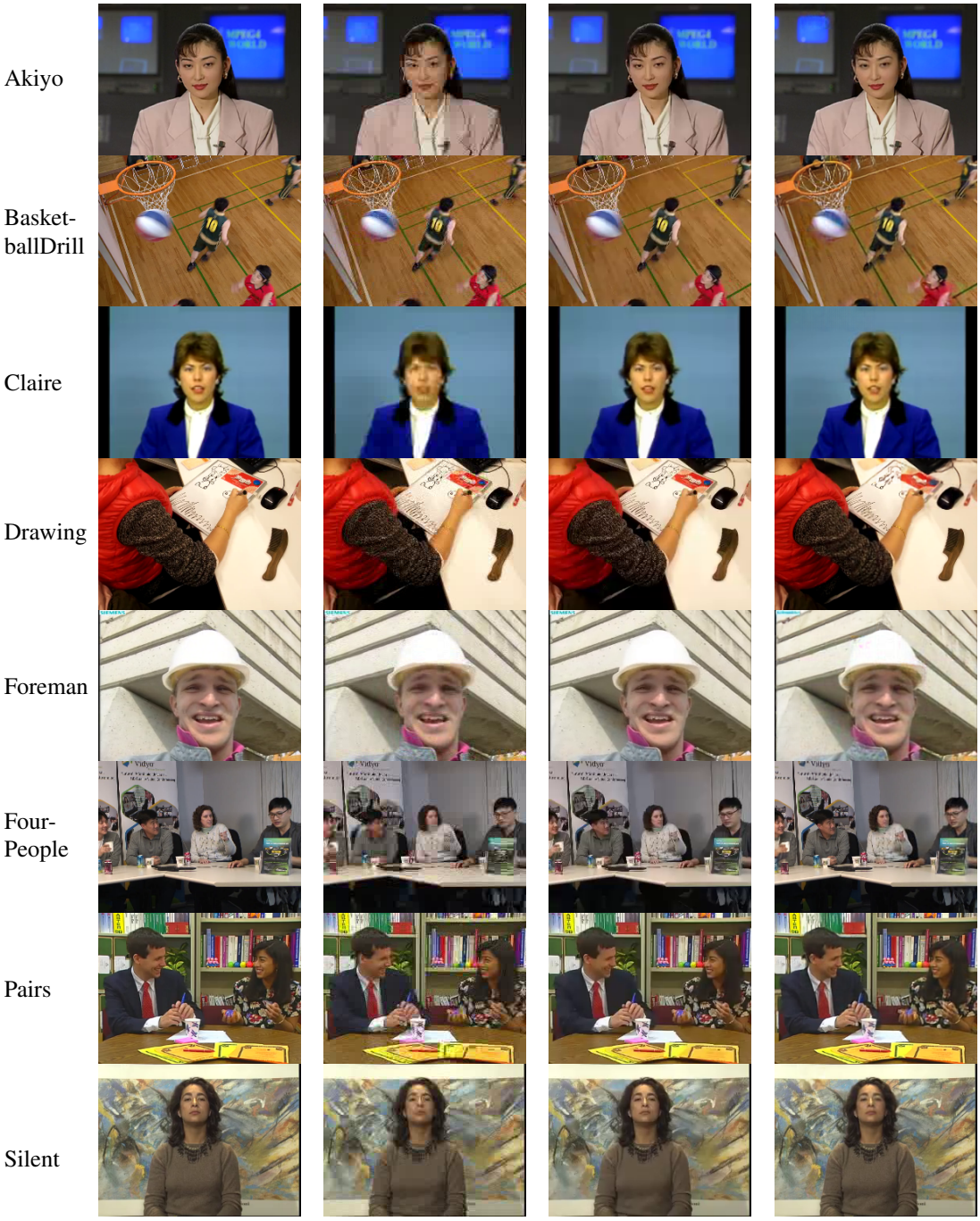
Fig. 7: Quantitative analysis of our learning-based video compression framework. It is worth noting that entropy coding is not employed for our results, even though it is commonly done in standard video compression codecs, and may bring more gain about $5\% \sim 57\%$ [4] performance improvement.

framework, the one simultaneously conditioned on spatial and temporal dependencies ('VoxelCNN') outperforms the other two patterns that conditioned on individual dependency ('Temporal-Pred' and 'Spatial-Pred') or none of these dependencies ('No-Pred'). It is natural because VoxelCNN modeled on a stronger prior knowledge, while 'Temporal-Pred' and 'Spatial-Pred' only model the temporal motion trajectory or spatial content relevance respectively.

We further refer BD-rate [42] (bit-rate savings) to calculate equivalent bit-rate savings between two compression schemes. As Table I illustrates, compared with individual dependency ('Spatial-Pred' and 'Temporal-Pred'), VoxelCNN effectively exploits spatiotemporal coherence and outperforms 'Spatial-Pred' and 'Temporal-Pred' significantly.

B. Comparison with Traditional Video Codecs

As the first work of learning-based video compression, we quantitatively analyze the performance of our framework and



(a) Raw (b) MPEG-2 (c) H.264 (d) Our Scheme

Fig. 8: Subjective comparison between various codecs under the same bit-rate.

compare it with modern video codecs.

We provide quantitative comparison with traditional video codecs in Table II and Figure 7, as well as subjective quality comparison in Figure 8. The experimental results illustrate that our proposed scheme outperforms MPEG-2 significantly while demonstrates a comparable results with H.264 codec. We achieve this despite the facts:

- We do not perform entropy coding in our experiments, while traditional codecs integrate entropy coding module for further improving coding efficiency. As reported in [4], entropy coding may bring more gain about $5\% \sim 57\%$ performance improvement.
- We do not perform complex prediction modes selection or adaptive transformation schemes as developed for decades in traditional video coding schemes.

Although affected by the above factors and our learning based video compression framework is in its infancy stage to compete with latest H.265/HEVC video coding technologies [12], it still shows great improvement over the first successful video codec MPEG-2 and has enormous potential in the following aspects:

- We provide a possible new direction to solve the limitation of heuristics in HVC by learning the parameters of encoder/decoder.
- The gradient-based optimization used in our framework can be seamlessly integrated with various metrics (loss function) including perceptual fidelity and semantic fidelity, which is infeasible to HVC. For instance, a neural network-based object tracking algorithm can be employed as a semantic metric for surveillance video compression.
- We have less side information required to be transmitted than HVC. The only overhead in our scheme is the flag ($< 1\%$ of bitstream) used for temporal progressive coding. By contrast, HVC require considerable side information (e.g., motion vector) to indicate sophisticated coding modes.

We also observe that, our approach shows unstable performance on various test sequences (especially in the case of global motion). This is reasonable since the coding modes we adopted in our algorithm are still very simple and unbalanced. However, we successfully demonstrate the potential of this framework and provide a potential new direction for video compression.

VI. Conclusion

We propose the concept of VoxelCNN by modeling spatiotemporal coherence to effectively perform predictive coding and explore a learning-based framework for video compression. Although lack of entropy coding, this scheme still achieve a promising result for video compression, demonstrating a new possible direction of video compression. In future, we expect more optimization since there are lots of aspects can be further improved in this framework, including entropy coding, variable block size coding, enhanced prediction and integration of various metrics (e.g., perceptual/semantic metric), etc.

Appendix A
Detailed architecture for VoxelCNN

We provide all parameters in VoxelCNN in the Table III. The notation is consistent with paper. In addition, 'BN' denotes Batch Normalization. 'DeConv' denotes deconvolution layer. 'Concat' denotes concatenate feature maps along the last dimension. 'DilConv' denotes dilated convolution layer.

TABLE III: Detailed architecture for VoxelCNN.

Layer	Type	Input	Size / Stride	Dilation	BN	Activation	Output	Output Size
1	Motion Extension	$\hat{f}^{i-2}$, $\hat{f}^{i-1}$	-	-	-	-	$\bar{f}^i$	256×192×3
2	Conv	$\hat{f}^{i-2}$, $\hat{f}^{i-1}$, $\bar{f}^i$	4×4 / 2	-	Y	ReLU	conv2	128×96×96
3	ResBlock×4	conv2	-	-	-	-	rb3	128×96×96
4	Conv	rb3	4×4 / 2	-	Y	ReLU	conv4	64×48×192
5	ResBlock×8	conv4	-	-	-	-	rb5	64×48×192
6	Conv	rb5	4×4 / 2	-	Y	ReLU	conv6	32×24×192
7	ResBlock×12	conv6	-	-	-	-	rb7	32×24×192
8	Conv	rb7	4×4 / 2	-	Y	ReLU	conv8	16×12×96
9	ConvLSTM	conv8	3×3 / 1	-	N	-	convlstm9	16×12×32
10	DeConv	convlstm9	5×5 / 2	-	Y	ReLU	deconv10	32×24×32
11	Pooling	$\bar{f}^i$	5×5 / 8	-	N	-	pooling11	32×24×32
12	Conv	pooling11	4×4 / 1	-	Y	ReLU	conv12	32×24×32
13	Concat	deconv10, conv12	-	-	-	-	concat13	32×24×64
14	ResBlock×12	concat13	-	-	-	-	rb14	32×24×64
15	DeConv	rb14	5×5 / 2	-	Y	ReLU	deconv15	64×48×32
16	Pooling	$\bar{f}^i$	5×5 / 4	-	N	-	pooling16	64×48×32
17	Conv	pooling16	4×4 / 1	-	Y	ReLU	conv17	64×48×32
18	Concat	deconv15, conv17	-	-	-	-	concat18	64×48×64
19	ResBlock×8	concat18	-	-	-	-	rb19	64×48×64
20	DeConv	rb19	5×5 / 2	-	Y	ReLU	deconv20	128×96×16
21	Pooling	$\bar{f}^i$	5×5 / 2	-	N	-	pooling21	128×96×16
22	Conv	pooling21	4×4 / 1	-	Y	ReLU	conv22	128×96×16
23	Concat	deconv20, conv22	-	-	-	-	concat23	128×96×32
24	ResBlock×4	concat23	-	-	-	-	rb24	128×96×32
25	DeConv	rb24	5×5 / 2	-	Y	Tanh	deconv25	256×192×3
26	Conv	$\bar{f}^i$, deconv25	4×4 / 1	-	N	Tanh	conv26	256×192×3
27	ConvBlock×8	$\hat{b}^i_{j-9}$, $\hat{b}^i_{j-8}$, $\hat{b}^i_{j-1}$, conv26	-	-	-	Tanh	cb27	64×64×3
28	Crop	cb27	-	-	-	-	output	32×32×3
			ResBlock					
1	Conv	input	3×3 / 1	-	Y	ReLU	conv1	-
2	Conv	conv1	3×3 / 1	-	N	ReLU	conv2	-
3	Add	input, conv2	-	-	-	-	output	-
			ConvBlock					
1	DilConv	input	3×3 / 1	1	Y	ReLU	conv1	-
2	DilConv	input	3×3 / 1	2	Y	ReLU	conv2	-
3	DilConv	input	3×3 / 1	4	Y	ReLU	conv3	-
4	DilConv	input	3×3 / 1	8	Y	ReLU	conv4	-
5	Concat	conv1, conv2, conv3, conv4	-	-	N	-	output	-

REFERENCES

[1] C. Systems, "Cisco visual networking index: Forecast and methodology, 2016–2021," *CISCO Systems White paper*, 2017.

[2] A. Habibi, "Hybrid coding of pictorial data," *IEEE Transactions on Communications*, vol. 22, no. 5, pp. 614–624, 1974.

[3] R. Forchheimer, "Differential transform codinga new hybrid coding scheme," in *Proc. Picture Coding Symp.(PCS-81), Montreal, Canada*, 1981, pp. 15–16.

[4] G. Toderici, D. Vincent, N. Johnston, S. Jin Hwang, D. Minnen, J. Shor, and M. Covell, "Full resolution image compression with recurrent neural networks," in *Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[5] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," in *International Conference on Learning Representations (ICLR)*, 2017.

[6] L. Theis, W. Shi, A. Cunningham, and F. Huszár, "Lossy image compression with compressive autoencoders," in *International Conference on Learning Representations (ICLR)*, 2017.

[7] G. Toderici, S. M. O'Malley, S. J. Hwang, D. Vincent, D. Minnen, S. Baluja, M. Covell, and R. Sukthankar, "Variable rate image compression with recurrent neural networks," in *International Conference on Learning Representations (ICLR)*, 2016.

[8] J. Ohm and M. Wien, "Future video coding coding tools and developments beyond hevc," in *Tutorial in International Conference on Image Processing (ICIP)*, Sept 2017.

[9] A. Prakash, N. Moran, S. Garber, A. DiLillo, and J. Storer, "Semantic perceptual image compression using deep convolution networks," in *Data Compression Conference (DCC)*. IEEE, 2017, pp. 250–259.

[10] N. Yan, D. Liu, H. Li, and F. Wu, "A convolutional neural network approach for half-pel interpolation in video coding," in *International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2017.

[11] F. Jiang, W. Tao, S. Liu, J. Ren, X. Guo, and D. Zhao, "An end-to-end compression framework based on convolutional neural networks," *IEEE Transactions on Circuits and Systems for Video Technology*, 2017.

[12] G. J. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

[13] R. Song, D. Liu, H. Li, and F. Wu, "Neural network-based arithmetic coding of intra prediction modes in hevc," in *International Conference on Visual Communications and Image Processing (VCIP)*. IEEE, 2017.

[14] T. Li, M. Xu, and X. Deng, "A deep convolutional neural network approach for complexity reduction on intra-mode hevc," in *International Conference on Multimedia and Expo (ICME)*. IEEE, 2017, pp. 1255–1260.

[15] X. Yu, Z. Liu, J. Liu, Y. Gao, and D. Wang, "Vlsi friendly fast cu/pu mode decision for hevc intra encoding: Leveraging convolution neural network," in *International Conference on Image Processing (ICIP)*. IEEE, 2015, pp. 1285–1289.

[16] C. Dong, Y. Deng, C. Change Loy, and X. Tang, "Compression artifacts reduction by a deep convolutional network," in *International Conference on Computer Vision (ICCV)*, 2015, pp. 576–584.

[17] T. Wang, M. Chen, and H. Chao, "A novel deep learning-based method of improving coding efficiency from the decoder-end for hevc," in *Data Compression Conference (DCC)*. IEEE, 2017, pp. 410–419.

[18] Y. Dai, D. Liu, and F. Wu, "A convolutional neural network approach for post-processing in hevc intra coding," in *International Conference on Multimedia Modeling (ICMM)*. Springer, 2017, pp. 28–39.

[19] Y. Li, D. Liu, H. Li, L. Li, F. Wu, H. Zhang, and H. Yang, "Convolutional neural network-based block up-sampling for intra frame coding," *IEEE Transactions on Circuits and Systems for Video Technology*, 2017.

[20] T. Dumas, A. Roumy, and C. Guillemot, "Image compression with stochastic winner-take-all auto-encoder," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP 2017)*, 2017.

[21] S. Santurkar, D. Budden, and N. Shavit, "Generative compression," *arXiv preprint arXiv:1703.01467*, 2017.

[22] O. Rippel and L. Bourdev, "Real-time adaptive image compression," in *International Conference on Machine Learning (ICML)*, 2017.

[23] K. Gregor, F. Besse, D. J. Rezende, I. Danihelka, and D. Wierstra, "Towards conceptual compression," in *Advances In Neural Information Processing Systems (NIPS)*, 2016, pp. 3549–3557.

[24] N. Johnston, D. Vincent, D. Minnen, M. Covell, S. Singh, T. Chinen, S. J. Hwang, J. Shor, and G. Toderici, "Improved lossy image compression with priming and spatially adaptive bit rates for recurrent networks," *arXiv preprint arXiv:1703.10114*, 2017.

[25] M. H. Baig, V. Koltun, and L. Torresani, "Learning to inpaint for image compression," in *Advances In Neural Information Processing Systems (NIPS)*, 2017.

[26] A. v. d. Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks," in *International Conference on Machine Learning (ICML)*, 2017.

[27] Z. Chen, J. Xu, Y. He, and J. Zheng, "Fast integer-pel and fractional-pel motion estimation for h. 264/avc," *Journal of Visual Communication and Image Representation*, vol. 17, no. 2, pp. 264–290, 2006.

[28] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 802–810.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[30] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning (ICML)*, 2015, pp. 448–456.

[31] J. O'neal, "Predictive quantizing systems (differential pulse code modulation) for the transmission of television signals," *Bell Labs Technical Journal*, vol. 45, no. 5, pp. 689–721, 1966.

[32] T. Raiko, M. Berglund, G. Alain, and L. Dinh, "Techniques for learning binary stochastic feedforward neural networks," in *International Conference on Learning Representations (ICLR)*, 2015.

[33] G. J. Sullivan and T. Wiegand, "Video compression-from concepts to the h. 264/avc standard," *Proceedings of the IEEE*, vol. 93, no. 1, pp. 18–31, 2005.

[34] K. Soomro, A. R. Zamir, and M. Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild," *CoRR*, vol. abs/1212.0402, 2012.

[35] H. Common, "Test conditions and software reference configurations, jctvc-l1100," ITU-T/ISO/IEC Joint Collaborative Team on Video Coding (JCT-VC), Tech. Rep., 2013.

[36] D. Vaisey and A. Gersho, "Variable block-size image coding," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 12. IEEE, 1987, pp. 1051–1054.

[37] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015.

[38] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *International Conference on Computer Vision (ICCV)*, 2015, pp. 1026–1034.

[39] ITU-T and I. J. 1, "Generic coding of moving pictures and associated audio information-part 2: video," 1994.

[40] 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.

[41] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multiscale structural similarity for image quality assessment," in *Conference Record of the Thirty-Seventh Asilomar Conference on Signals, Systems and Computers*, vol. 2. IEEE, 2003, pp. 1398–1402.

[42] G. Bjontegaard, "Calcuation of average psnr differences between rd-curves," *Doc. VCEG-M33 ITU-T Q6/16, Austin, TX, USA, 2-4 April 2001*, 2001.