DeepWiVe: Deep-Learning-Aided Wireless Video Transmission

Tze-Yang Tung and Deniz Gündüz
Information Processing and Communications Lab (IPC-Lab),
Imperial College London, UK
{tze-yang.tung14, d.gunduz}@imperial.ac.uk

***Abstract*—We present *DeepWiVe*, the first-ever end-to-end joint source-channel coding (JSCC) video transmission scheme that leverages the power of deep neural networks (DNNs) to directly map video signals to channel symbols, combining video compression, channel coding, and modulation steps into a single neural transform. Our DNN decoder predicts residuals without distortion feedback, which improves the video quality by accounting for occlusion/disocclusion and camera movements. We simultaneously train different bandwidth allocation networks for the frames to allow variable bandwidth transmission. Then, we train a bandwidth allocation network using reinforcement learning (RL) that optimizes the allocation of limited available channel bandwidth among video frames to maximize the overall visual quality. Our results show that *DeepWiVe* can overcome the *cliff-effect*, which is prevalent in conventional separation-based digital communication schemes, and achieve graceful degradation with the mismatch between the estimated and actual channel qualities. *DeepWiVe* outperforms H.264 video compression followed by low-density parity check (LDPC) codes in all channel conditions by up to 0.0485 in terms of the multi-scale structural similarity index measure (MS-SSIM), and H.265 + LDPC by up to 0.0069 on average. We also illustrate the importance of optimizing bandwidth allocation in JSCC video transmission by showing that our optimal bandwidth allocation policy is superior to uniform allocation as well as a heuristic policy benchmark.**

***Index Terms*—Deep learning, joint source-channel coding, video compression, wireless video transmission.**

I. Introduction

Video content contributes to more than 80% of Internet traffic and the percentage is only expected to increase [1]. Video compression is widely used to reduce the bandwidth requirement when transmitting video signals wirelessly. This follows the modular approach employed in almost all wireless video transmission systems, where the end-to-end transmission problem is divided into two: (1) a source encoder that compresses the video into a sequence of bits of the shortest possible length such that a reconstruction of the original video is possible within an allowable distortion; and (2) a channel encoder that introduces redundancies such that the compressed bits are protected against channel errors and interference. A diagram of this modular approach is shown in Fig. 1.

Separate source and channel coding design provides modularity and allows independent optimization of each component. It has been applied successfully in a large variety of applications from on-demand mobile video streaming to video conferencing and digital TV broadcasting. However, the limits of the separation-based designs are beginning to rear with the emergence of more demanding video delivery applications, such as wireless virtual reality (VR) and drone-based surveillance systems. These applications impose low latency requirements, suffer from highly-varying channel conditions, and need to be implemented on energy-limited mobile devices, making the separation-based approach highly suboptimal.

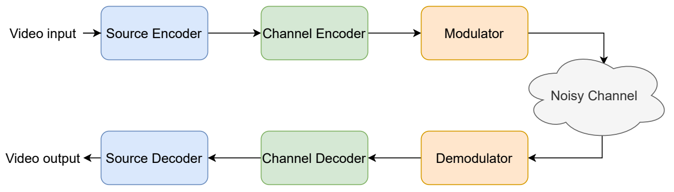


Fig. 1. Diagram of a typical separation-based digital video delivery system employed by almost all communication systems today.

In the context of wireless video transmission, separation-based designs lead to what is known as the *cliff-effect*. That is, when the channel condition deteriorates below the level anticipated by the channel encoder, the source information becomes irrecoverable. This leads to a cliff edge deterioration of the system performance. As a result, most current systems operate at a much more conservative transmission rate than is suggested by the instantaneous channel capacity, and employ additional error correction mechanisms through automatic repeat requests (ARQ).

An alternative to the separation-based architecture is joint source-channel coding (JSCC). It has been shown theoretically that under finite delay constraints, that is, in the finite block length regime, JSCC can achieve lower distortion than separate source and channel coding [2]–[5]. The most straightforward JSCC approach continues to employ separate modules for compression and communication, but jointly optimizes various parameters of these modules in a cross-layer framework. While there have been many such proposals over the years [6]–[13], these techniques typically do not provide sufficient gains to justify the significant increase in system complexity.

A more fundamental approach is to design the transmission system from scratch, without considering any digital interface in between. The best example for such an approach is analog communications, such as AM/FM radio or analog TV, where the information is directly modulated onto the carrier waveform without any compression. By doing so,

This work was supported by the European Research Council (ERC) through project BEACON (No. 677854).

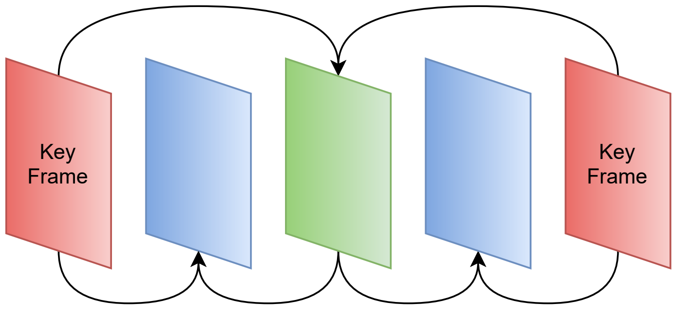


Fig. 2. Diagram of a typical interpolation structure used in video compression algorithm.

analog communication can overcome the cliff-effect problem by showing graceful degradation with the channel parameters, and it is extremely simple, unlike the aforementioned cross-layer designs. From a fundamental information theoretic perspective, when transmitting independent Gaussian samples over an additive white Gaussian noise (AWGN) channel, with one sample per channel use on average, uncoded transmission, where each sample is simply scaled and transmitted, meets the theoretical Shannon bound [4]. With digital transmission, the same performance can only be achieved by vector-quantizing an arbitrarily long sequence of source samples, followed by a capacity achieving channel code. Benefits of analog transmission has also been shown in various multi-user scenarios [14], [15]. However, analog modulation cannot exploit the available bandwidth efficiently, and the optimality of simple uncoded transmission does not generalize to bandwidth-mismatched scenarios. Despite this theoretical suboptimality, analog modulation approaches to image and video transmission have recently gained popularity [16]–[19], mainly due to their low computational complexity and the graceful degradation with channel quality.

Recently, it has been shown in [20], [21] that deep neural networks (DNNs) can break the complexity barrier in designing effective JSCC schemes, focusing particularly on the image transmission problem. The approach there is to train DNN models in an autoencoder architecture with a non-trainable channel layer between the encoder and decoder. The authors showed that this approach not only provides graceful degradation with channel quality, but can also achieve results similar to or superior than state-of-the-art separation-based digital designs. An extension to this work [22] further shows that JSCC is successively refineable; that is, an image can be transmitted in stages, and each additional channel block further refines the quality of the decoded image, with almost no additional cost.

Herein, we propose a DNN-based JSCC solution for wireless video transmission, called *DeepWiVe*, which is trained to optimize the reconstructed video quality in an end-to-end fashion. Although the problem of video compression using deep learning has received significant attention [23]–[31], no prior work has considered deep learning aided wireless video delivery.

The core idea behind video compression algorithms is to exploit the temporal correlations across video frames. In a standard video sequence, motion differences between successive frames is typically very small, and video compression algorithms can be very efficient by identifying intermittent key frames, which are compressed and decoded independently, and conveying only the motion and residual information for the remaining frames, thereby exploiting temporal redundancy. Here, the residual information refers to the difference between the true frame and the motion compensated key frame. Fig. 2 shows a typical interpolation structure in a video compression algorithm. The residual information is available at the encoder as the encoder can simply decode the compressed frames and observe the difference between the source and its reconstruction. On the other hand, for JSCC, the residual depends on the reconstructed key frames and motion information, which in turn depend on the channel condition during transmission. Therefore, the residual is not known at the encoder. To overcome this, we propose to use a DNN to predict the residual, without the need for distortion feedback.

In *DeepWiVe* we directly map the video sequence into the channel vector under a channel bandwidth constraint for the transmission of a group-of-pictures (GoP). Similarly to [22], we employ variable bandwidth transmission and allocate the bandwidth dynamically using reinforcement learning (RL) to train a bandwidth allocation network that optimizes the bandwidth utilization of each frame. Our results show that *DeepWiVe* can meet or beat industry standard video compression codecs, such as H.264, combined with state-of-the-art channel codes, such as low density parity check (LDPC) codes, in almost all channel conditions tested, while achieving graceful degradation of video quality with respect to channel quality, thereby avoiding the *cliff-effect*.

The contributions of this paper are summarized as follows:

1) We propose *DeepWiVe*, a JSCC-based wireless video transmission scheme leveraging DNNs to jointly compress and channel code video frames in an end-to-end manner to maximize the end video quality.
2) We train our DNN decoder to predict residuals without the need for distortion feedback.
3) We optimize bandwidth allocation among video frames by training a bandwidth allocation policy using RL, and show that it achieves superior performance than naïve uniform allocation and a heuristic policy.
4) We show that it is possible to achieve variable bandwidth transmission by simply training different bandwidth allocation networks.
5) Numerical results show that our proposed *DeepWiVe* is superior to industry standard H.264 [32] codec with state-of-the-art LDPC channel codes [33] in all channel conditions tested and can avoid the cliff-effect. It also beats H.265 [34] using the same channel codes when evaluated in terms of the multi-scale structural similarity index measure (MS-SSIM).

II. Related Work

JSCC for video delivery has consistently received attention over the years. The earliest work we could find is [10], which studies the problem of video multicast to heterogeneous receivers. They approached the problem from the receivers' perspective, where the source video is encoded in a hierarchical manner, with each layer of the hierarchy distributed on a separate network channel. Each receiver then adapts to its local channel capacity by adjusting the number of layers it decodes. In a similar line of work, [35] uses scalable video coding (SVC), which encodes the source video into multiple bitstreams, with a base layer that represents the lowest supported quality and a set of enhancement layers representing versions of the video at different qualities. Since the base layer and lower quality layers require more error protection than higher quality layers, unequal error protection (UEP) of packets is used. To determine the optimal channel code rate for each layer such that the average distortion is minimized, they devised a low complexity search algorithm to find the optimal choice of channel code rates among a set of available rates. There is a large body of works that use source coding schemes similarly to SVC and minimize the end-to-end distortion by jointly optimizing various parameters of the source and channel codes [8], [9], [11]–[13], [36]. However, none of the scheme proposed were able to achieve adequate performance gains for the increased complexity they introduce as a result of the optimization of the parameters. Moreover, with the exception of [10] and [12], which model the channel as a physical link with a certain error probability, the above works mainly consider time-varying channels with proposed schemes aimed at adapting to channel variations.

A completely refreshed approach to JSCC video delivery, called SoftCast, utilizing low complexity methods to map videos or images from the pixel domain to channel symbols directly was first introduced in [16]. Their scheme involves a hybrid digital and analog design by leveraging frequency domain sparsity. Since then, various works have improved upon [16] by optimizing different aspects of the hybrid digital and analog design. In [16]–[19], frequency domain sparsity is exploited by utilizing compressed sensing to reduce the bandwidth requirement. In [37], the power allocation problem is addressed by optimizing the division of frequency domain coefficients into chunks. In [38], the proposed scheme separates a base layer (low frequency coefficients) from the enhancement layer (high frequency coefficients) and sends the base layer using a digital separation-based system, while the enhancement layer is sent using a scheme similar to SoftCast. This method also improves power allocation since low frequency coefficients have larger values than high frequency coefficients in natural images so the majority of the signal energy is sent through the digital system. In [39], a similar approach is utilized, however, instead of the high frequency components, they send the gradient of the image as the enhancement information. In [40], the prior used for estimation is modified to obtain better reconstruction quality (SoftCast assumes a Gaussian prior). In [41], the variability of temporal redundancies between frames is addressed by introducing adaptive GoP sizes. Although these methods have been shown to overcome the *cliff-effect*, they are not competitive to separation-based schemes when it comes to video quality and cannot exploit the available bandwidth, or adapt to channel and network conditions dynamically.

TABLE I
DEFINITION OF NOTATIONS.

Parameter	Definition
$\mathbb{E}[\cdot]$	Expectation operator
$\|\| \cdot \|\|_2$	l_2 norm operator
P	Transmitter power constraint
H, W	Height and width of video frame

Following the success of DNNs in various image and video intelligence tasks, there has been a growing interest in employing them for video compression [23]–[31]. Although some recent works have reported competitive or superior performance with respect to state-of-the-art video compression standards H.264/5, none of the works have considered the wireless video delivery problem, taking into account the channel conditions. In principle, the works treat the communication layer simply as a perfect bit pipe. As a result, the shortcomings of separation-based schemes, such as the *cliff-effect*, are inherent to those works if applied in a wireless communication scenario.

The closest prior art to our work are [20]–[22], [42], [43], which explore the JSCC problem, but they focus on image transmission. In the context of video transmission, there are unique challenges that sets it apart from simple image transmission. Namely, exploiting the inter-frame redundancies to improve coding efficiency and to optimize resource allocation across the frames. Therefore, extending the problem from image to video transmission is not a trivial task.

III. SYSTEM MODEL AND PROBLEM FORMULATION

We consider the problem of wireless video transmission in a constrained bandwidth setting. Table I shows the definition of notations used herein. Let $\mathbf{X} = \{\mathbf{X}^n\}_{n=1}^T$ be a video sequence made up of T GoPs, where $\mathbf{X}^n = \{\mathbf{x}_1^n, \ldots, \mathbf{x}_N^n\}$, $\mathbf{x}_i^n \in \mathbb{R}^{H \times W \times 3}$, $\forall i \in [1, N]$, represents the nth GoP of size N frames in the video sequence. Each frame $\mathbf{x}_i^n$ is represented as a 24 bit RGB image. We wish to design an encoding function $E : \mathbb{R}^{TN \times H \times W \times 3} \mapsto \mathbb{C}^{Tk}$, which maps the video sequence $\mathbf{X}$ to a set of complex symbols $\mathbf{z} = E(\mathbf{X}) \in \mathbb{C}^{Tk}$, and a decoding function $D : \mathbb{C}^{Tk} \mapsto \mathbb{R}^{TN \times H \times W \times 3}$, which maps the channel output $\mathbf{y} = \Upsilon(\mathbf{z})$, to an approximate reconstruction of the original video sequence $\hat{\mathbf{X}} = D(\mathbf{y})$. Here, $\Upsilon(\cdot)$ is the channel transfer function.

In this setting, we restrict the number of channel uses to k per GoP, which can be considered as a bandwidth constraint and we define the *bandwidth compression ratio* as

$$\rho = \frac{k}{3HWN}. \tag{1}$$

We consider the following channel model:

$$\mathbf{y} = \Upsilon(\mathbf{z}) = \mathbf{H}\mathbf{z} + \mathbf{n}, \tag{2}$$

where $\mathbf{H} = \text{diag}(h^1\mathbf{I}, \ldots, h^T\mathbf{I}) \in \mathbb{C}^{Tk \times Tk}$ is a block-wise diagonal matrix where each block $h^n\mathbf{I} = \text{diag}(h^n, \ldots, h^n) \in \mathbb{C}^{k \times k}$ represents the channel gain for the nth GoP. The channel noise $\mathbf{n} \sim \text{CN}(0, \sigma^2\mathbf{I})$ is an independent and identically distributed (i.i.d.) additive Gaussian noise with variance σ^2 and $\mathbf{I}$ is the identity matrix.

We impose an average power constraint P at the transmitter:

$$\frac{1}{Tk}\mathbb{E}_{\mathbf{z}}\left[||\mathbf{z}||_2^2\right] \leq P, \tag{3}$$

where the expectation is over the distribution of the encoder output. We consider two scenarios regarding the channel variations and the availability of channel state information (CSI):

1) The channel gain $\mathbf{H}$ is static and both the transmitter and receiver have full knowledge of the channel gain and the noise variance through channel estimation and feedback [44].
2) The channel gain h^n for each GoP is i.i.d. and the transmitter only knows the average SNR, while the receiver knows the phase of the channel gain $\arg(h^n)$, but not the magnitude $|h^n|$, $\forall n$.

We note that in the first scenario, having full knowledge of the CSI reduces the channel model to an additive white Gaussian noise (AWGN) channel for each GoP as the transmitter can perform precoding, such that

$$\mathbf{z} \leftarrow \frac{\mathbf{H}^*}{|\mathbf{H}|}\mathbf{z}, \tag{4}$$

where $\mathbf{H}^*$ is the element-wise complex conjugate of $\mathbf{H}$ and $|\mathbf{H}| = \text{diag}(|h^1|\mathbf{I}, \dots, |h^T|\mathbf{I})$ is the element-wise magnitude of $\mathbf{H}$. Accordingly, the channel signal-to-noise ratio (SNR) for the AWGN channel is defined as

$$\text{SNR}_{\text{AWGN}} = 10\log_{10}\left(\frac{|h^n|^2 P}{\sigma^2}\right) \text{ dB}. \tag{5}$$

In the second scenario, the model is equivalent to a real fading channel with double the bandwidth as only the magnitude of the channel gain changes randomly for each GoP transmission period. This is because the receiver, given the phase of the channel gain, can perform phase equalization, such that

$$\mathbf{y} \leftarrow \exp(-j\arg(\mathbf{H}))\mathbf{y}, \tag{6}$$

where $\arg(\mathbf{H}) = \text{diag}(\arg(h^1)\mathbf{I}, \dots, \arg(h^T)\mathbf{I})$ is the element-wise phase values of the channel gains in $\mathbf{H}$, and $j = \sqrt{-1}$ here exclusively. Accordingly, the average channel signal-to-noise ratio (SNR) is defined as

$$\text{SNR}_{\text{Fading}} = 10\log_{10}\left(\frac{\mathbb{E}[|h^n|^2]P}{\sigma^2}\right) \text{ dB}. \tag{7}$$

We measure the average quality of the reconstructed video using two metrics: peak signal-to-noise ratio (PSNR) and MS-SSIM. They are defined as

$$\text{PSNR}(\mathbf{X}, \hat{\mathbf{X}}) = \frac{1}{TN}\sum_{n=1}^{T}\sum_{i=1}^{N} 10\log_{10}\left(\frac{255^2}{l_{\text{PSNR}}(\mathbf{x}_i^n, \hat{\mathbf{x}}_i^n)}\right) \text{ dB}, \tag{8}$$

and

$$\text{MS-SSIM}(\mathbf{X}, \hat{\mathbf{X}}) = \frac{1}{TN}\sum_{n=1}^{T}\sum_{i=1}^{N} 1 - l_{\text{MS-SSIM}}(\mathbf{x}_i^n, \hat{\mathbf{x}}_i^n), \tag{9}$$

where

$$l_{\text{PSNR}}(\mathbf{x}_i^n, \hat{\mathbf{x}}_i^n) = \frac{1}{3HW}||\mathbf{x}_i^n - \hat{\mathbf{x}}_i^n||_2^2, \tag{10}$$

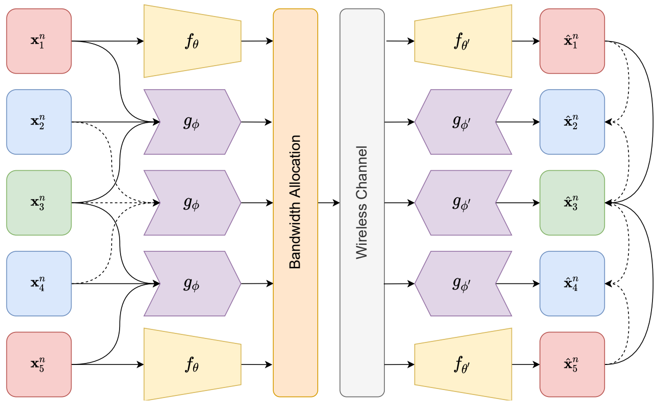


Fig. 3. *DeepWiVe* system overview.

and

$$l_{\text{MS-SSIM}}(\mathbf{x}_i^n, \hat{\mathbf{x}}_i^n) = 1 - \text{MS-SSIM}(\mathbf{x}_i^n, \hat{\mathbf{x}}_i^n). \tag{11}$$

MS-SSIM is defined between two frames as

$$\text{MS-SSIM}(\mathbf{x}_i^n, \hat{\mathbf{x}}_i^n) \tag{12}$$

$$= \left[l_M(\mathbf{x}_i^n, \hat{\mathbf{x}}_i^n)]^{\alpha_M} \prod_{j=1}^{M}[c_j(\mathbf{x}_i^n, \hat{\mathbf{x}}_i^n)]^{\beta_j}[s_j(\mathbf{x}_i^n, \hat{\mathbf{x}}_i^n)\right]^{\gamma_j}, \tag{13}$$

where

$$l_M(\mathbf{x}_i^n, \hat{\mathbf{x}}_i^n) = \frac{2\mu_{\mathbf{x}_i^n}\mu_{\hat{\mathbf{x}}_i^n} + c_1}{\mu_{\mathbf{x}_i^n}^2 + \mu_{\hat{\mathbf{x}}_i^n}^2 + c_1}, \tag{14}$$

$$c_j(\mathbf{x}_i^n, \hat{\mathbf{x}}_i^n) = \frac{2\sigma_{\mathbf{x}_i^n}\sigma_{\hat{\mathbf{x}}_i^n} + c_2}{\sigma_{\mathbf{x}_i^n}^2 + \sigma_{\hat{\mathbf{x}}_i^n}^2 + c_2}, \tag{15}$$

$$s_j(\mathbf{x}_i^n, \hat{\mathbf{x}}_i^n) = \frac{\sigma_{\mathbf{x}_i^n\hat{\mathbf{x}}_i^n} + c_3}{\sigma_{\mathbf{x}_i^n}\sigma_{\hat{\mathbf{x}}_i^n} + c_3}. \tag{16}$$

Here, $\mu_{\mathbf{x}_i^n}$, $\sigma_{\mathbf{x}_i^n}^2$, $\sigma_{\mathbf{x}_i^n\hat{\mathbf{x}}_i^n}^2$ are the mean and variance of $\mathbf{x}_i^n$, and the covariance between $\mathbf{x}_i^n$ and $\hat{\mathbf{x}}_i^n$, respectively. c_1, c_2, and c_3 are coefficients for numeric stability; α_M, β_j, and γ_j are the weights for each of the components. Each $c_j(\cdot)$ and $s_j(\cdot)$ are computed at a different downsampled scale of $(\mathbf{x}_i^n, \hat{\mathbf{x}}_i^n)$. We use the default parameter values of (α_M, β_j, γ_j) provided by the original paper [45]. MS-SSIM has been shown to be as good and better at approximating human visual perception than the more simplistic structural similarity index (SSIM) on different subjective image and video databases.

The goal is to maximize the reconstructed video quality, measured by either Eqn. (8) or (9), between the input video $\mathbf{X}$ and its reconstruction $\hat{\mathbf{X}}$, under the constraints on the bandwidth ratio ρ and average power P.

A. Joint Source-Channel Video Coding

In this section, we present our proposed DNN-based joint source-channel video encoding and decoding scheme. We will deconstruct the design of the encoder (E) and decoder (D) into three parts: the key frame encoder/decoder ($f_{\boldsymbol{\theta}}, f_{\boldsymbol{\theta}'}$), parameterized by ($\boldsymbol{\theta}, \boldsymbol{\theta}'$), the interpolation encoder/decoder ($g_{\boldsymbol{\phi}}, g_{\boldsymbol{\phi}'}$), parameterized by ($\boldsymbol{\phi}, \boldsymbol{\phi}'$), and the bandwidth allocation function $q_{\boldsymbol{\psi}}$, parameterized by $\boldsymbol{\psi}$. We will represent all these functions with DNNs, where the parameters of the functions

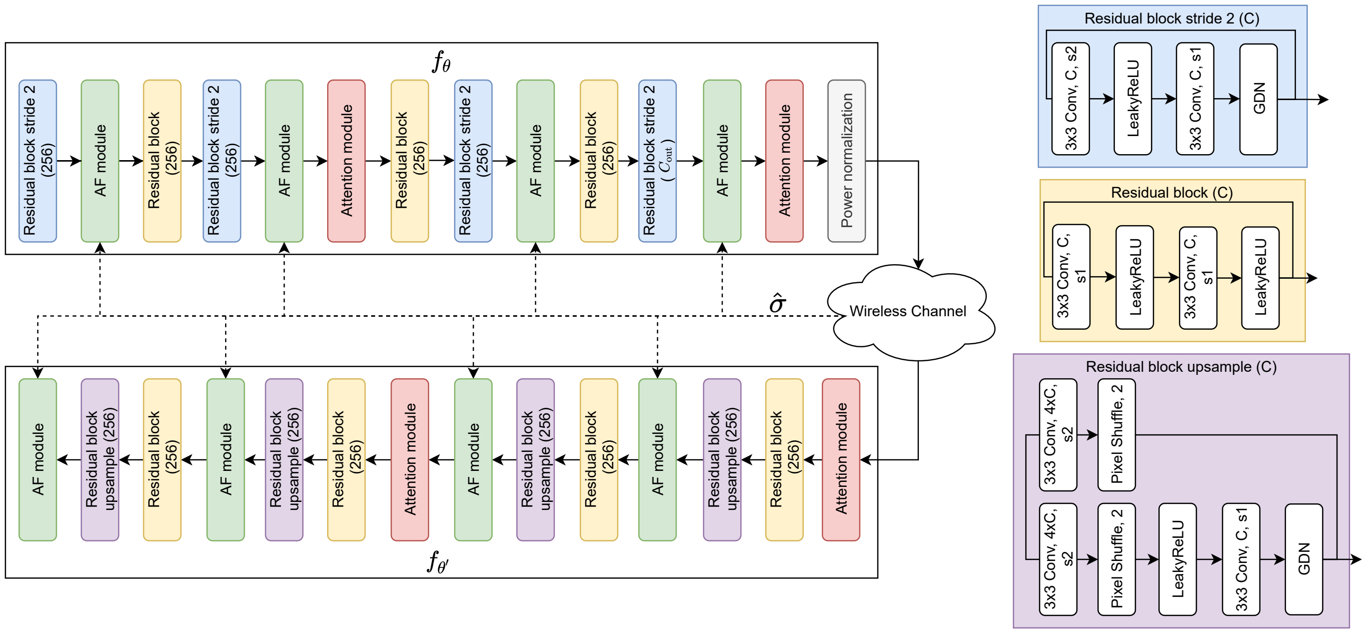


Fig. 4. Key frame encoder/decoder $(f_{\boldsymbol{\theta}}, f_{\boldsymbol{\theta}'})$ network architectures.

correspond to the weights of these DNNs. An overview of our scheme is shown in Fig. 3.

One of the drawbacks of prior works was the need to train multiple networks, one for each channel condition. To address this issue, [42] proposed an attention feature (AF) module, motivated by resource assignment strategies in traditional JSCC schemes [46]–[48], which allows the network to learn to assign different weights to different features for a given SNR. That is, given the SNR and the input tensor of the channel, the AF module produces an attention mask for each channel of the input tensor and scales each channel based on the attention weights. By doing so, the network learns to assign different importance levels to each feature channel, similarly to unequal power allocation. To learn the attention mask for each SNR, we deliberately randomize the channel SNR during training, and provide the AF modules with the SNR values. The results in [42] show that a single model whose parameters are adjusted to the channel SNR with the help of the AF modules perform at least as well as the models trained for each SNR individually. We adopt the AF module in *DeepWive* to obtain a single model that can work over a range of SNRs.

The encoding and decoding procedures are described herein. Consider the nth GoP, $\mathbf{X}^n = \{\mathbf{x}_1^n, ..., \mathbf{x}_N^n\}$. The last $(\mathbf{x}_N^n)$ frame is called the key frame and is compressed and transmitted using the key frame encoder $f_{\boldsymbol{\theta}} : \mathbb{R}^{H\times W\times 3} \mapsto \mathbb{C}^k$,

$$\mathbf{z}_i^n = f_{\boldsymbol{\theta}}(\mathbf{x}_i^n, \hat{\sigma}^2), \;\; i = N, \tag{17}$$

where $\hat{\sigma}^2$ is the estimated channel noise power at the transmitter. Here, each element of $\mathbf{z}_i^n$, denoted by $z_{i,j}^n$, represents the in-phase (I) and quadrature (Q) components of a complex channel symbol. We note that, while the channel input/output values are complex, we employ real-valued DNN architecture. The mapping between real network outputs and complex channel inputs (or vice versa) is achieved by pairing consecutive real values at the output of the encoder DNN.

The values in the complex latent vector are first normalized according to

$$\hat{z}_{i,j}^n = \sqrt{kP}\frac{z_{i,j}^n}{\sqrt{(\mathbf{z}_i^n)^H\mathbf{z}_i^n}}, \;\; j = 1, \ldots, k, \tag{18}$$

where P is the power constraint, k is the bandwidth constraint, and H refers to the Hermitian transpose.

These values are then directly sent through the channel as,

$$\hat{\mathbf{y}}_i^n = \Upsilon(\hat{\mathbf{z}}_i^n). \tag{19}$$

Consequently, the key frame decoder $f_{\boldsymbol{\theta}'} : \mathbb{C}^k \mapsto \mathbb{R}^{H\times W\times 3}$ that maps the channel output $\hat{\mathbf{y}}_i^n \in \mathbb{C}^k$ observed at the receiver to a reconstructed frame $\hat{\mathbf{x}}_i^n \in \mathbb{R}^{H\times W\times 3}$ is defined as:

$$\hat{\mathbf{x}}_i^n = f_{\boldsymbol{\theta}'}(\hat{\mathbf{y}}_i^n, \hat{\sigma}^2), \;\; i = N. \tag{20}$$

The loss between the original frame $\mathbf{x}_i^n$ and the reconstructed frame $\hat{\mathbf{x}}_i^n$ is computed using Eqn. (10) or (11) depending on which performance measure is being used. The network weights $(\boldsymbol{\theta}, \boldsymbol{\theta}')$ are then updated via backpropagation with respect to the gradient of the loss.

The network architectures of the key frame encoder and decoder are shown in Fig. 4. The Pixel Shuffle module, first proposed in [49], increases the height and width of the input tensor dimensions efficiently by exchanging channel dimensions for height and width dimensions. The GDN layer refers to generalized divisive normalization, initially proposed in [50], and has been shown to be effective in density modeling and compression of images. The Attention layer refers to the simplified attention module proposed in [51], which reduces the computation cost of the attention module originally proposed in [52]. The attention mechanism has been used in both [51] and [52] to improve the compression efficiency by focusing the neural network on regions in the image that require higher bit rate. The network in Fig. 4 is fully

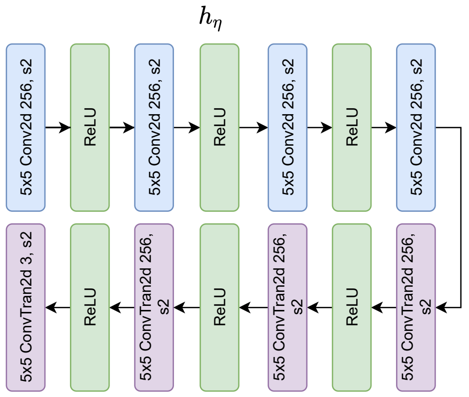


Fig. 5. Architecture of the SSF estimator network $h_{\boldsymbol{\eta}}$.

convolutional, therefore it can accept input of any height (H) and width (W), making it versatile to any video resolution.

For the remaining frames, i.e., $\mathbf{x}_i^n$, $i = 1, \ldots, N-1$, we use the interpolation encoder $g_{\boldsymbol{\phi}}(\cdot)$ to encode the motion information ($\boldsymbol{\delta}_{i-t}^n$, $\boldsymbol{\delta}_{i+t}^n$) and residual information ($\mathbf{r}_{i-t}^n$, $\mathbf{r}_{i+t}^n$) of $\mathbf{x}_i^n$ with respect to two reference frames ($\bar{\mathbf{x}}_{i-t}^n$, $\bar{\mathbf{x}}_{i+t}^n$) that are t frames away from the current one. These reference frames are what the encoder expects the corresponding reconstructed frames $\hat{\mathbf{x}}_{i-t}^n$, $\hat{\mathbf{x}}_{i+t}^n$ to be. This is done via channel emulation and decoding at the transmitter to obtain an approximation of what the transmitter expects the receiver to reconstruct.

We follow the same interpolation structure as the one presented in Fig. 2 for a GoP size $N = 4$. That is, for $i = 2$, $t = 2$, while for $i = 1, 3$, $t = 1$. We define $\bar{\mathbf{x}}_0^n = \bar{\mathbf{x}}_N^{n-1}$, and assume that the GoPs are encoded and decoded sequentially, such that the frames from the previous decoded GoP are available as reference for the current GoP. To interpolate $\mathbf{x}_i^n$ from $\bar{\mathbf{x}}_{i-t}^n$ and $\bar{\mathbf{x}}_{i+t}^n$, motion information, such as optical flow [53], is usually used to warp a reference image by translating the pixels in the reference image according to the optical flow vectors. These optical flow vectors describe the horizontal and vertical translations of each pixel in a reference image in order to transform it into the target image. The difference between the optical flow transformed image and the true target image is called the *residual*, which is used to capture information that cannot be described by optical flow, such as occlusion/disocclusion and camera movements.

To estimate the motion information ($\boldsymbol{\delta}_{i-t}^n$, $\boldsymbol{\delta}_{i+t}^n$), instead of using optical flow, we use scaled space flow (SSF), which was first proposed by [29] as a more general description of pixel warping than optical flow. The idea is to blur regions of the frame where the motion is difficult to model using traditional pixel warping and instead compensate those regions using the residual. To that end, in scale-space warping (SSW), a frame is first transformed into a fixed-resolution volume $\bar{\mathbf{X}}_{i+t}^n = [\bar{\mathbf{x}}_{i+t}^n, \bar{\mathbf{x}}_{i+t}^n \otimes G(\sigma_0), \bar{\mathbf{x}}_{i+t}^n \otimes G(2\sigma_0), \ldots, \bar{\mathbf{x}}_{i+t}^n \otimes G(2^{V-1}\sigma_0)]$, where $\bar{\mathbf{x}}_{i+t}^n \otimes G(\sigma_0)$ denotes Gaussian blurring of the frame $\bar{\mathbf{x}}_{i+t}^n$ by convolving $\bar{\mathbf{x}}_{i+t}^n$ with a Gaussian kernel $G(\sigma_0)$ with standard deviation σ_0 and $\otimes$ is the convolution operation. V is the number of levels in the volume. $\bar{\mathbf{X}}_{i+t}^n \in \mathbb{R}^{H \times W \times (V+1)}$ represents a progressively blurred version of $\mathbf{x}_{i+t}^n$, which can be sampled at continuous points via trilinear interpolation. The scaled space flow $\boldsymbol{\delta}_{i+t}^n \in \mathbb{R}^{H \times W \times 3}$ that warps frame $\bar{\mathbf{x}}_{i+t}^n$ to an approximation of $\mathbf{x}_i^n$ denoted by $\tilde{\mathbf{x}}_{i+t}^n$ is then defined as

$$\begin{aligned} &\tilde{\mathbf{x}}_{i+t}^n = \text{SSW}(\bar{\mathbf{x}}_{i+t}^n, \boldsymbol{\delta}_{i+t}^n). \\ &\text{s.t. } \tilde{\mathbf{x}}_{i+t}^n[x, y] \\ &\quad = \bar{\mathbf{X}}_{i+t}^n[x + \boldsymbol{\delta}_{i+t}^n[x,y,1], y + \boldsymbol{\delta}_{i+t}^n[x,y,2], \boldsymbol{\delta}_{i+t}^n[x,y,3]]. \end{aligned} \tag{21}$$

To estimate the scaled space flow $\boldsymbol{\delta}_{i+t}^n$, we use the network architecture $h_{\boldsymbol{\eta}} : \mathbb{R}^{H \times W \times 6} \mapsto \mathbb{R}^{H \times W \times 3}$ proposed in [29],

$$\boldsymbol{\delta}_{i+t}^n = h_{\boldsymbol{\eta}}(\mathbf{x}_i^n, \bar{\mathbf{x}}_{i+t}^n). \tag{22}$$

Fig. 5 shows the architecture of the SSF estimator network $h_{\boldsymbol{\eta}}$. The architecture progressively downsamples the input using convolutional layers before upsampling it back to the frame dimensions. The channel dimension of the output SSF $\boldsymbol{\delta}_{i+t}^n$, instead of representing the three color channels, represent the (x, y, z) sampling points in the SSF volume $\bar{\mathbf{X}}_{i+t}^n$. This architecture is similar to U-Net [54], which has been shown to perform well for image segmentation problems. The intuition here is that the network can also be used to estimate objects or segments moving in the same direction in a scene, which helps to estimate the SSF.

Given the above definition of SSF, the residual $\mathbf{r}_{i+t}^n$ is defined as

$$\mathbf{r}_{i+t}^n = \mathbf{x}_i^n - \tilde{\mathbf{x}}_{i+t}^n. \tag{23}$$

The interpolation encoder $g_{\boldsymbol{\phi}} : \mathbb{R}^{H \times W \times 21} \mapsto \mathbb{C}^k$ defines the mapping

$$\begin{aligned} &\mathbf{z}_i^n = g_{\boldsymbol{\phi}}(\mathbf{x}_i^n, \bar{\mathbf{x}}_{i-t}^n, \bar{\mathbf{x}}_{i+t}^n, \mathbf{r}_{i-t}^n, \mathbf{r}_{i+t}^n, \boldsymbol{\delta}_{i-t}^n, \boldsymbol{\delta}_{i+t}^n, \hat{\sigma}^2), \\ &i = 1, 2, \ldots, N-1. \end{aligned} \tag{24}$$

The vector $\mathbf{z}_i^n \in \mathbb{C}^k$ is power normalized according to Eqn. (18) and sent across the channel according to Eqn. (19).

Given the noisy $\hat{\mathbf{y}}_i^n$, the decoder first estimates the SSF, the residual, and a mask. That is, the interpolation decoder $g_{\boldsymbol{\phi}'} : \mathbb{C}^k \mapsto \mathbb{R}^{H \times W \times 12}$ defines the mapping

$$(\hat{\boldsymbol{\delta}}_{i-t}^n, \hat{\boldsymbol{\delta}}_{i+t}^n, \hat{\mathbf{r}}_i^n, \mathbf{m}_i^n) = g_{\boldsymbol{\phi}'}(\hat{\mathbf{y}}_i^n, \hat{\sigma}^2), \tag{25}$$

where $\hat{\boldsymbol{\delta}}_{i\pm t}^n \in \mathbb{R}^{H \times W \times 3}$, $\hat{\mathbf{r}}_i^n \in \mathbb{R}^{H \times W \times 3}$, and $\mathbf{m}_i^n \in \mathbb{R}^{H \times W \times 3}$. $\mathbf{m}_{i,c}^n \in \mathbb{R}^{H \times W}, c = 1, 2, 3$, a 2D matrix in the third dimension of $\mathbf{m}_i^n$, satisfies:

$$\sum_{c=1}^{3} \mathbf{m}_{i,c}^n = \mathbf{1}_{H \times W}. \tag{26}$$

That is, for each H and W index of the mask $\mathbf{m}_i^n$, the sum of values along the channel dimension is equal to 1, which is achieved by using the softmax activation. The reconstructed frame is then defined as:

$$\begin{aligned} \hat{\mathbf{x}}_i^n = &(\mathbf{m}_i^n)_1 * \text{SSW}(\hat{\mathbf{x}}_{i-t}^n, \hat{\boldsymbol{\delta}}_{i-t}^n) + \\ &(\mathbf{m}_i^n)_2 * \text{SSW}(\hat{\mathbf{x}}_{i+t}^n, \hat{\boldsymbol{\delta}}_{i+t}^n) + (\mathbf{m}_i^n)_3 * \hat{\mathbf{r}}_i^n, \end{aligned} \tag{27}$$

where $*$ refers to element-wise multiplication. The predicted mask $\mathbf{m}_i^n$ acts as a convex set of weights to sum the two motion compensated predictions and the predicted residual, such that the resultant prediction $\hat{\mathbf{x}}_i^n$ remains within

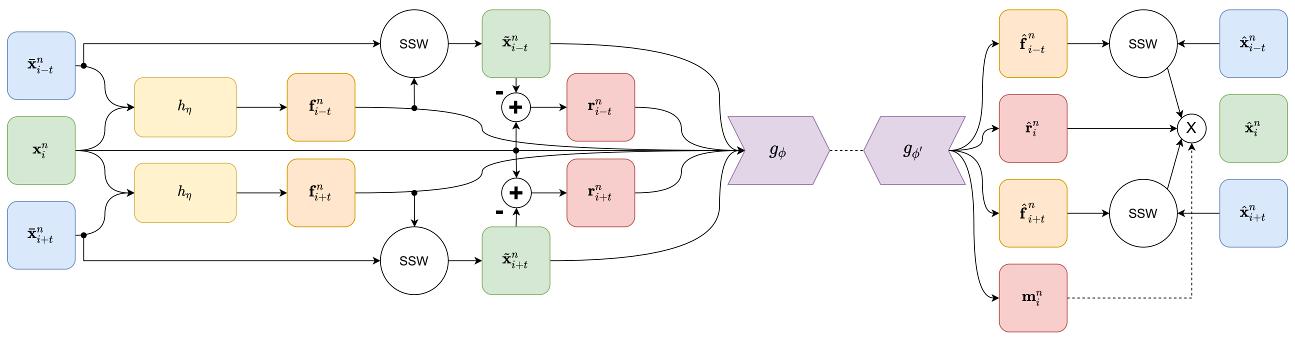


Fig. 6. Information flow over the interpolation network.

$[0, 255]$. A diagram of the interpolation structure described herein can be seen in Fig. 6. The architectures of $g_{\boldsymbol{\phi}}$ and $g_{\boldsymbol{\phi}'}$ are functionally the same as $f_{\boldsymbol{\theta}}$ and $f_{\boldsymbol{\theta}'}$, except the size of the input tensor, which is the concatenation of $(\mathbf{x}_i^n, \bar{\mathbf{x}}_{i-t}^n, \bar{\mathbf{x}}_{i+t}^n, \mathbf{r}_{i-t}^n, \mathbf{r}_{i+t}^n, \boldsymbol{\delta}_{i-t}^n, \boldsymbol{\delta}_{i+t}^n)$ along the channel dimension. As in the key frame, the network weights $(\boldsymbol{\phi}, \boldsymbol{\phi}', \boldsymbol{\eta})$ are updated via backpropagation with respect to the gradient of the loss between the original and the reconstructed frame.

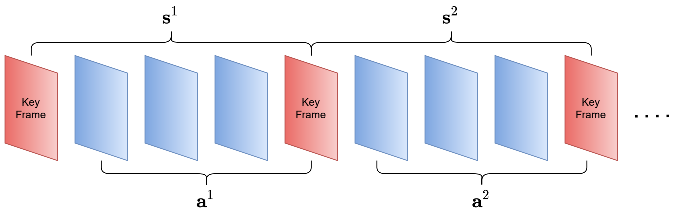


Fig. 7. Illustration of the bandwidth allocation problem formulated as a MDP.

B. Bandwidth Allocation

In the previous section, we have assumed that each frame utilizes the full bandwidth of k channel uses allowed for each GoP. In order to satisfy the bandwidth constraint defined in Section III, the encoder must decide how to allocate k channel uses to the N frames in a GoP. Intuitively, if the frame in consideration $\mathbf{x}_i^n$ is exactly the same with respect to the reference frames $(\bar{\mathbf{x}}_{i-t}^n, \bar{\mathbf{x}}_{i+t}^n)$ that it is interpolated from, then no information needs to be transmitted. On the other hand, if there is significant differences with respect to the reference frames, then more information needs to be sent in order to accurately interpolate the frame. Since the last frame of a previous GoP becomes the reference frame of the next GoP $(\bar{\mathbf{x}}_N^n = \bar{\mathbf{x}}_0^{n+1})$, we formulate the problem of allocating available bandwidth in each GoP as a Markov decision process (MDP) and solve the optimal bandwidth allocation policy using reinforcement learning.

An MDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the action set, $\mathcal{P}$ is the probability transition kernel that defines the probability of one state transitioning to another state given an action, and $r : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is the reward function. At each time step n, an agent observes state $\mathbf{s}^n \in \mathcal{S}$ and takes an action $\mathbf{a}^n \in \mathcal{A}$ based on its policy $\pi : \mathcal{S} \mapsto \mathcal{A}$. The state then transitions to $\mathbf{s}^{n+1}$ according to the probability $\mathcal{P}(\mathbf{s}^{n+1}|\mathbf{s}^n, \mathbf{a}^n)$, and the agent receives a reward $r^n(\mathbf{s}^n, \mathbf{a}^n)$. The objective is to maximize the expected sum of rewards $J(\pi) = \mathbb{E}_{\mathbf{s}^1 \sim \omega_{\mathbf{s}^1}, \pi}[\sum_{i=1}^{\infty} \gamma^{(i-1)} r^i(\mathbf{s}^i, \mathbf{a}^i)]$, where $\omega_{\mathbf{s}^1}$ is the initial state distribution and $\gamma \in (0, 1)$ is the reward discount factor to ensure convergence.

In the dynamic bandwidth allocation problem, we define each GoP as one time step, where the state at time step n is $\mathbf{s}^n = \{\mathbf{M}^n, \mathbf{R}^n, \mathbf{F}^n, \hat{\sigma}^2\}$, with

$$\mathbf{M}^n = \{\bar{\mathbf{x}}_i^n\}_{i=0}^{N}, \tag{28}$$

$$\mathbf{R}^n = \{\mathbf{r}_{i\pm t}^n\}_{i=1}^{N-1}, \tag{29}$$

$$\mathbf{F}^n = \{\boldsymbol{\delta}_{i\pm t}^n\}_{i=1}^{N-1}, \tag{30}$$

and $\bar{\mathbf{x}}_0^n = \bar{\mathbf{x}}_N^{n-1}$. The action set $\mathcal{A}$ is the set of all the different ways the available bandwidth k can be allocated to each frame in the GoP. In order for the decoder functions $f_{\boldsymbol{\theta}'}$ and $g_{\boldsymbol{\phi}'}$ to be able to decode each frame that has been given different amounts of bandwidth, we use a result in [22], which showed that joint source-channel encoded images can be successively refined by sending increasingly more information. This is achieved by dividing the latent vectors $\mathbf{z}_i^n$ into U equal sized blocks (i.e., $\mathbf{z}_i^n = \{\mathbf{z}_{i,1}^n, \ldots, \mathbf{z}_{i,U}^n\}$, $\mathbf{z}_{i,u}^n \in \mathbb{C}^{\frac{k}{U}}, u = 1, ..., U$), while randomly varying the number of blocks u_i^n of the latent code transmitted in each batch $\mathbf{z}_i^n(u_i^n) = \{\mathbf{z}_{i,1}^n, \ldots, \mathbf{z}_{i,u_i^n}^n\}$, $u_i^n \leq U$. This training process leads to the descending ordering of information from $\mathbf{z}_{i,1}^n$ to $\mathbf{z}_{i,U}^n$. As such, each action represents $\mathbf{a}^n = [u_1^n, ..., u_N^n]$. We implement this training process in the algorithm described in Section III-A by zeroing out the blocks in the latent vector not transmitted. As such, the action set is all the ways to assign U blocks to the N frames in the GoP; that is, $\sum_{i=1}^{N} u_i^n = k$. Consequently, it can be shown that the number of ways to assign U blocks to N sets without replacement is

$$|\mathcal{A}| = \frac{(U + N - 1)!}{U!(N - 1)!}. \tag{31}$$

Since we are concerned with maximizing the visual quality of the final video, we define the reward function r^n as

$$r^n = -\log_{10}\Big(l(\mathbf{X}^n, \hat{\mathbf{X}}^n)\Big), \tag{32}$$

where $l(\cdot, \cdot)$ is either l_{PSNR} or $l_{\text{MS-SSIM}}$ depending on the video quality metric used. Note that in the previous section, we said that the transmitter performs channel emulation in order to obtain the reference frames $(\bar{\mathbf{x}}_{i-t}^n, \bar{\mathbf{x}}_{i+t}^n)$. Since the precise estimate of the reference frames is bootstrapped to the

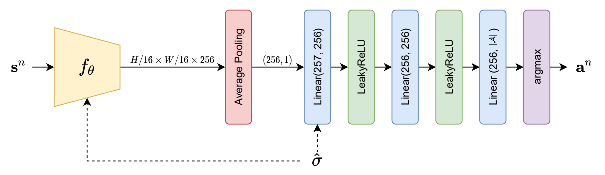


Fig. 8. Architecture of the bandwidth allocation network $q_{\boldsymbol{\psi}}$. The convolutional part of the network for feature extraction is functionally the same as the key encoder network ($f_{\boldsymbol{\theta}}$) but with $21(N-1)+6$ input dimensions to account for all the tensors in state $\mathbf{s}^n$.

amount of bandwidth allocated, we initially assume a uniform bandwidth allocation (i.e., $u_i^n = k/U$) when computing the SSF and residuals; but once the allocation has been done, the reference frame in the next state (i.e., $\bar{\mathbf{x}}_0^{n+1} \in \mathbf{M}^{n+1}$) is estimated based on the bandwidth allocated.

To solve the MDP described herein and to learn the optimal allocation policy, we use deep Q-learning [55], where the network $q_{\boldsymbol{\psi}}$ seeks to approximate the Q-function $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. The purpose of the Q-function is to map each state and action pair to a Q value, which represents the total discounted reward from step n given the state and action pair $(\mathbf{s}^n, \mathbf{a}^n)$. That is,

$$Q(\mathbf{s}^n, \mathbf{a}^n) = E\left[\sum_{i=n}^{\infty} \gamma^{i-n} r^i \middle| \mathbf{s}^n, \mathbf{a}^n\right], \ \forall (\mathbf{s}^n, \mathbf{a}^n) \in \mathcal{S} \times \mathcal{A}. \tag{33}$$

As is typical in DQN methods, we use *replay buffer*, *target network*, and *ϵ-greedy* to aid the learning of the Q-function. The replay buffer $\mathcal{R}$ stores experiences $(\mathbf{s}^n, \mathbf{a}^n, r^n, \mathbf{s}^{n+1})$ and are sampled uniformly to update the parameters $\boldsymbol{\psi}$. This prevents the states from being correlated, which would break the assumption in most optimization algorithms that the samples are independent. We use target parameters $\boldsymbol{\psi}^-$, which are copies of $\boldsymbol{\psi}$, to compute the DQN loss function:

$$L_{\text{DQN}}(\boldsymbol{\psi}) = \left(r^n + \gamma \max_{\mathbf{a}} \left\{q_{\boldsymbol{\psi}^-}(\mathbf{s}^{n+1}, \mathbf{a})\right\} - q_{\boldsymbol{\psi}}(\mathbf{s}^n, \mathbf{a}^n)\right)^2. \tag{34}$$

The parameters $\boldsymbol{\psi}$ are then updated via gradient descent according to the gradient $\nabla_{\boldsymbol{\psi}} L_{\text{DQN}}(\boldsymbol{\psi})$. In practice, the DQN loss is approximated with a batch of samples from the replay buffer $\mathcal{B} \subset \mathcal{R}$. The target network parameters are updated via

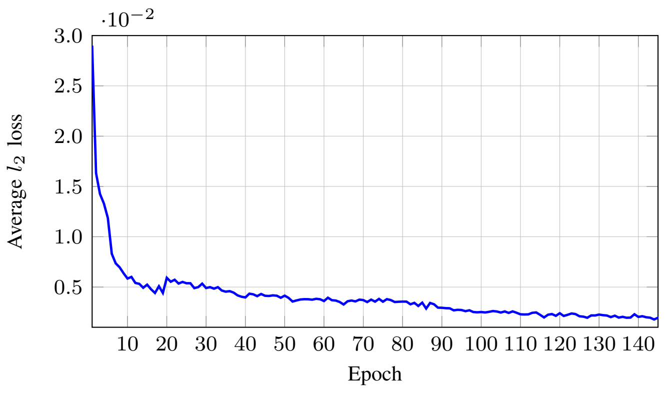
$$\boldsymbol{\psi}^- \leftarrow \tau \boldsymbol{\psi} + (1-\tau)\boldsymbol{\psi}^-, \tag{35}$$

where $0 \le \tau \le 1$. The target networks here stabilize the updates. Due to Q-learning being bootstrapped, if the same $q_{\boldsymbol{\psi}}$ is used to estimate the state-action value of GoP number n and $n+1$, both values would move at the same time, which may lead to the updates to never converge. By introducing the target networks, this effect is reduced due to the much slower updates of the target network, as shown in Eqn. (35).

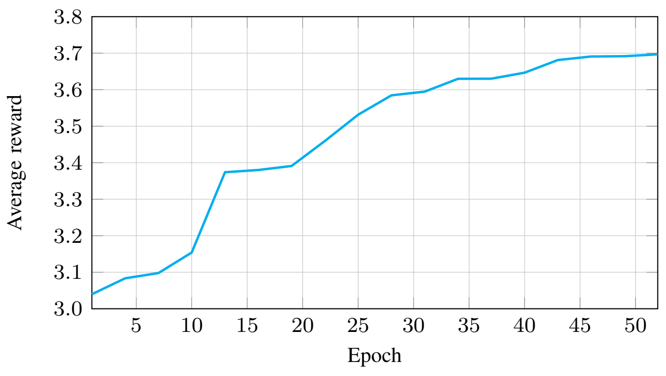
To promote exploration, we use ϵ-greedy, which chooses a random action with probability ϵ at each GoP. That is,

$$\mathbf{a}_n = \begin{cases} \arg\max_{\mathbf{a}} q_{\boldsymbol{\psi}}(\mathbf{s}_n, \mathbf{a}), & \text{w.p. } 1-\epsilon, \\ \mathbf{a} \sim \text{Uniform}(\mathcal{A}), & \text{w.p. } \epsilon, \end{cases} \tag{36}$$

where $\mathbf{a} \sim \text{Uniform}(\mathcal{A})$ denotes an action that is sampled uniformly from the action set $\mathcal{A}$. A diagram of the architecture used for $q_{\boldsymbol{\psi}}$ is shown in Fig. 8.



(a) JSCC networks ($f_{\boldsymbol{\theta}}, f_{\boldsymbol{\theta}'}, g_{\boldsymbol{\phi}}, g_{\boldsymbol{\phi}'}, h_{\boldsymbol{\eta}}$) training loss.



(b) Bandwidth allocation network $q_{\boldsymbol{\psi}}$ average training reward per epoch.

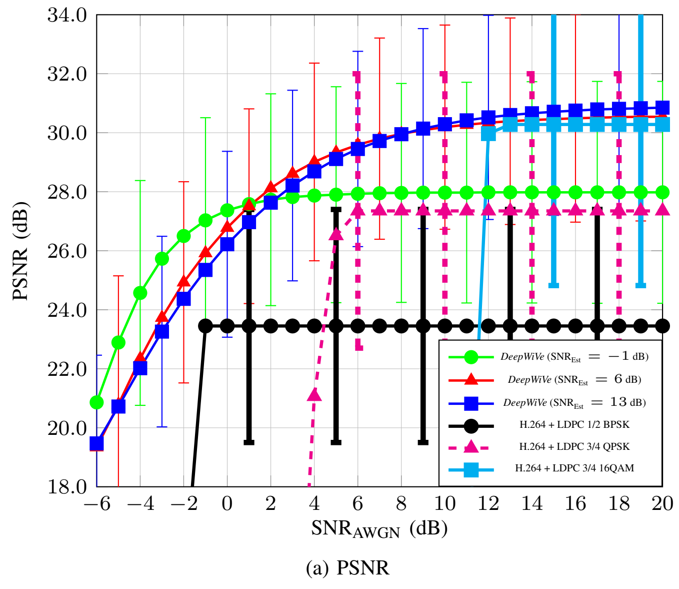
Fig. 9. Convergence of training *DeepWiVe* for $\rho = 0.031$ optimized for the PSNR metric.

Upon initialization, we send the first frame $\mathbf{x}_1^1$ using full bandwidth k. The first frame can be considered as a GoP on its own. For all subsequent GoPs, we perform optimal bandwidth allocation as described in this section. The bandwidth allocation problem is illustrated in Fig. 7.

IV. Numerical Results

A. Training Details

We train our models on the UCF101 dataset [56] using Pytorch [57], with the Adam optimizer [58] at learning rate $1e^{-4}$. We split the dataset with $8:2$ ratio between training and validation. We then test the model using the BVI-DVC dataset [59]. For the first CSI acquisition scenario, defined in Sec. III, we will consider a constant channel gain magnitude $|h^n| = 1, \ \forall n$, while in the second scenario, we consider $h^n \sim CN(0,1), \ \forall n$. We train the JSCC ($f_{\boldsymbol{\theta}}, f_{\boldsymbol{\theta}'}, g_{\boldsymbol{\phi}}, g_{\boldsymbol{\phi}'}, h_{\boldsymbol{\eta}}$) networks first, randomizing the latent vector block sizes as described in Section III-B, until convergence, before we train the bandwidth allocation network $q_{\boldsymbol{\psi}}$ to find the optimal bandwidth allocation policy. We also assume that during this phase, the transmitter and receiver can estimate the channel SNR accurately $\hat{\sigma}^2 = \sigma^2$. We use a training batch size of 4 when training the JSCC networks and a batch size of 8 when training the bandwidth allocation network. The batch sizes are relatively small due to memory restrictions of the available GPU (11GB Nvidia RTX 2080Ti). We use early stopping based on the validation error with a patience of 8



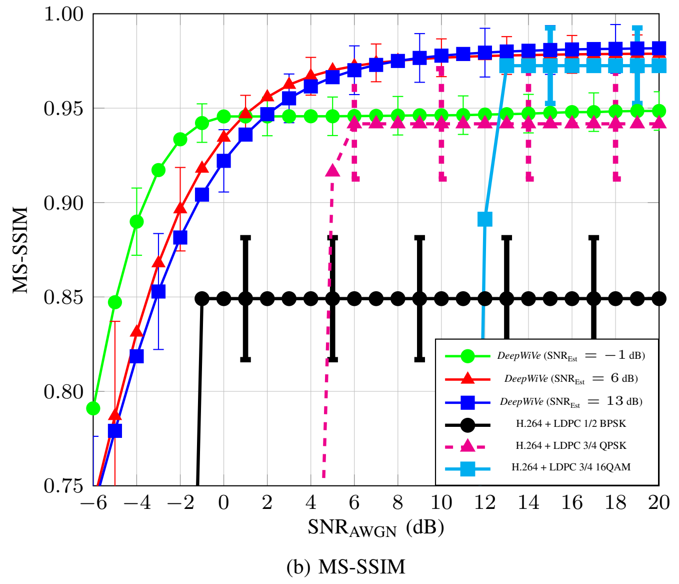
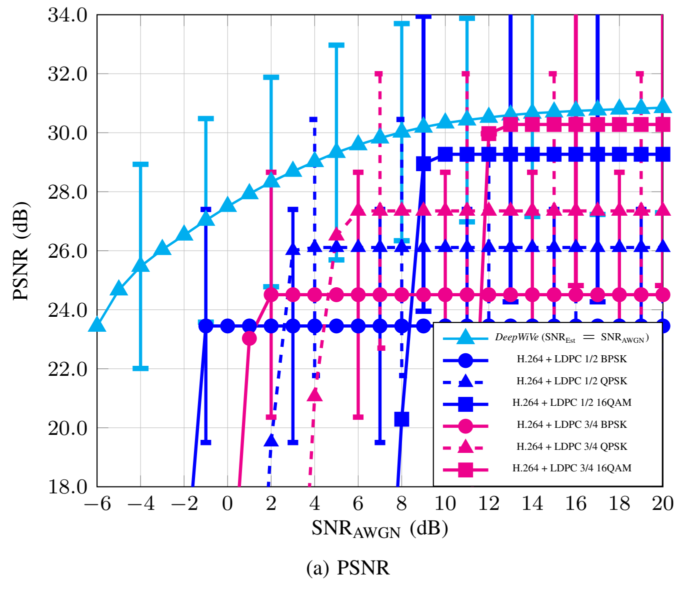


Fig. 10. Comparison of DeepWiVe using different SNR$_{\text{Est}}$ to H.264 paired with LDPC codes in the AWGN channel case ($\rho = 0.031$).



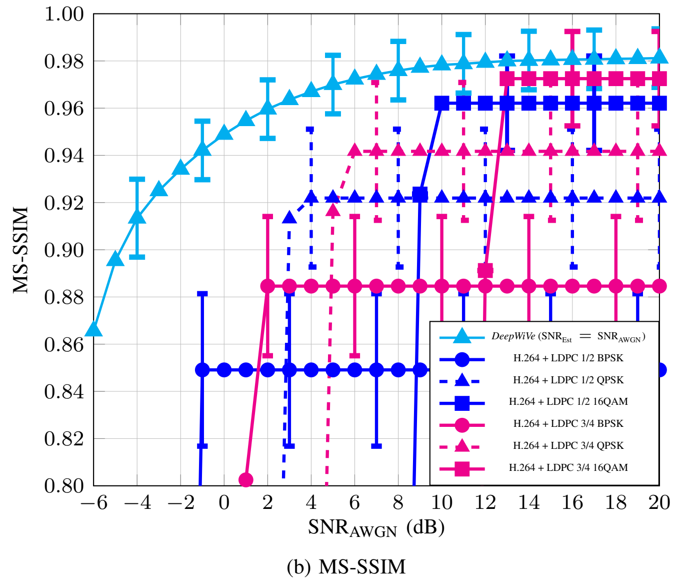


Fig. 11. Performance comparison of DeepWiVe to H.264 paired with LDPC codes in the AWGN channel case ($\rho = 0.031$).

epochs. We adjust the learning rate based on the number of bad epochs: if the validation error does not improve for 4 epochs in a row, the learning rate is multiplied by 0.8. The result of the training is shown in Fig. 9.

We define the SNR estimated by the transmitter and receiver to be

$$\text{SNR}_{\text{Est}} = 10\log_{10}\left(\frac{P}{\hat{\sigma}^2}\right). \tag{37}$$

For training the bandwidth allocation network, we choose DQN hyper-parameters $\gamma = 0.99$, $\tau = 0.005$, and a replay buffer size $|\mathcal{R}| = 1000$. The function used for ϵ-greedy exploration is

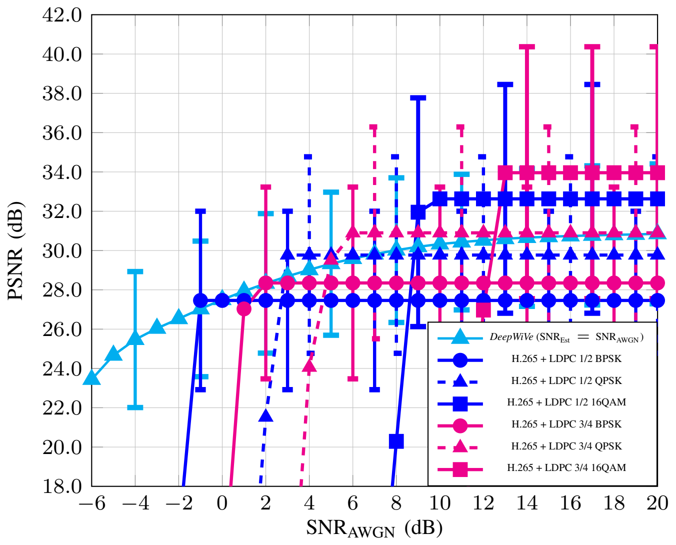
$$\epsilon = \epsilon_{\text{end}} + (\epsilon_0 - \epsilon_{\text{end}})\exp\left(-\frac{\text{episode}}{\lambda}\right), \tag{38}$$

where λ controls the decay rate of ϵ, and in each episode the bandwidth allocation network allocates the bandwidth resource within one video sequence. We choose $\epsilon_0 = 0.9$, $\epsilon_{\text{end}} = 0.05$, and $\lambda = 1000$ for these parameters.

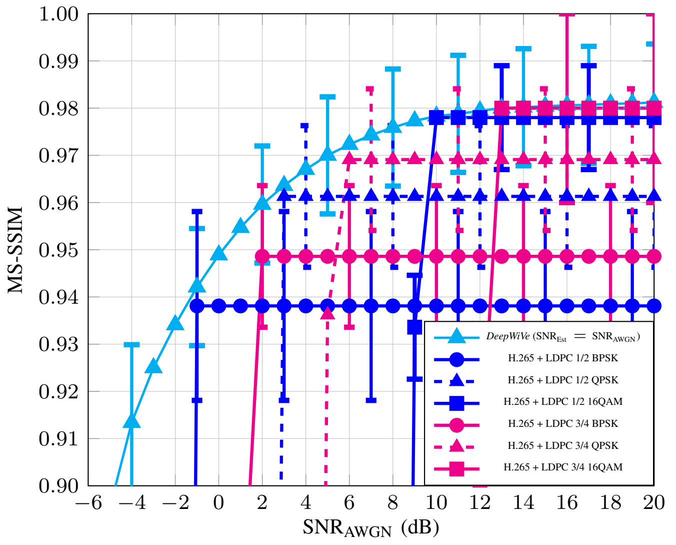
We train our model at different channel SNRs and evaluate each model at the same range of SNRs. In each batch, the training SNR is sampled uniformly from the range $[-5, 20]$ dB. We chose $N = 4$, $V = 5$ and $U = 20$ to train our models. Note that although we can choose $U = k$ to achieve very fine grained bandwidth control, this would lead to a very large action space, which makes Q-learning difficult. We let $P = 1$ and compute the necessary σ^2 to achieve the desired SNR.

B. Simulation Results

We compare the performance of our model with that of the conventional separation-based schemes. In particular, we use the H.264 [32] and H.265 [34] video compression codecs for source coding, LDPC codes [33] for channel coding, and QAM



(a) PSNR



(b) MS-SSIM

Fig. 12. Performance comparison of DeepWiVe to H.265 paired with LDPC codes in the AWGN channel case ($\rho = 0.031$).

modulation. We use the FFMPEG [60] library to perform both H.264 and H.265 encoding. We use two-pass encoding to ensure that the bit rate of the video is equal to the number of channel uses given the modulation order and channel code rate[1]. For example, if the number of channel uses per GoP is 1024, modulation order 16QAM and channel code rate 0.5, then the number of bits that can be used to transmit each GoP of video is $1024 \times 4 \times 0.5 = 2048$ bits. For the LDPC codes, we use Gallagher codes [33] with block length 960 bits for rate $1/2$ code and block length 1440 bits for rate $3/4$ code. We plot the average video quality across the test dataset using each of the schemes considered herein and error bars representing the standard deviation of the video qualities.

In Fig. 10, we show the effect of channel estimation error on the performance of *DeepWiVe* in the AWGN channel case. We specifically compare models using SNR_{Est} where the H.264 codec paired with a specific LDPC code rate and modulation order experiences the *cliff-effect*. It is clear that *DeepWiVe* is able to overcome the *cliff-effect*, with the video quality degrading gracefully as the SNR decreases even as SNR_{Est} remains the same. This is in contrast to the cliff edge drop off that separation-based designs suffer from. We can also see that the variations in the video quality using *DeepWiVe* is lower than those produced by the separation scheme as indicated by the smaller error bars. The error bars represent the standard deviation of the video quality at the receiver side. This is likely due to the fact that the H.264 codec does not have a continuous range of compression rates available but rather a set of discrete levels it can compress. Depending on the complexity of the video, a given target distortion may lead to a larger rate than is allowed by the instantaneous channel condition and it must reduce the target distortion level. Since the allowed target distortion levels can be far apart, this may mean the video is compressed more conservatively than is suggested by the channel in order to meet the channel condition, leading to a large variation in the resultant video quality. *DeepWiVe*, on the other hand, does not have this issue as we do not define a set of possible compression rates. Instead, the weights are adjusted by the AF modules based on the current channel condition to meet the rate-distortion curve as closely as possible.

We present the comparison of *DeepWiVe* using the accurate estimate of the channel SNR (i.e., $\text{SNR}_{\text{Est}} = \text{SNR}_{\text{AWGN}}$) with separation employing H.264 and H.265 in Figs. 11 and 12, respectively. In Fig. 11, we see that at $\rho = 0.031$, *DeepWiVe* is superior to the separation-based scheme using H.264 in all the SNRs tested. This shows that *DeepWiVe* can indeed learn an end-to-end optimized JSCC scheme that achieves lower distortion for a given rate than separation-based schemes, validating the theoretical superiority of JSCC in finite block length regimes [2]–[5]. We see in Fig. 12a that H.265 outperforms *DeepWiVe* in terms of the PSNR metric. However, when compared with the more perceptually aligned MS-SSIM metric in Fig. 12b, we see that *DeepWiVe* can outperform separation-based transmission with H.265. We also highlight that, in the very low SNR regime (i.e., $\text{SNR}_{\text{AWGN}} < -1$ dB), H.265 was unable to meet the compression rate required, and therefore did not produce results in that range. *DeepWiVe* on the other hand, did not have this problem. We believe that further optimization of the network architecture can bring *DeepWiVe* on par or surpass H.265 evaluated using the PSNR metric for higher SNR values as well. Specifically, a more flexible GoP interpolation structure and longer skip connections in the architecture design to improve gradient flow, may improve the coding efficiency of *DeepWiVe*.

Fig. 13 shows the visual results of the plots shown in Fig. 10 for a specific video. At $\text{SNR}_{\text{AWGN}} = 13$ dB, the visual qualities

[1]The parameters used for H.264 are `ffmpeg -y -i <input video> -c:v libx264 -b:v <file size (kbits)> -pass 1 -an -f null /dev/null && ffmpeg -i <input video> -c:v libx264 -b:v <file size (kbits)> -pass 2 -c:a -an <output file name>`.
For H.265, they are `ffmpeg -y -i <input video> -c:v libx265 -b:v <file size (kbits)> -x265-params -pass 1 -an -f null /dev/null && ffmpeg -i <input video> -c:v libx265 -b:v <file size (kbits)> -x265-params -pass 2 -c:a -an <output file name>`.

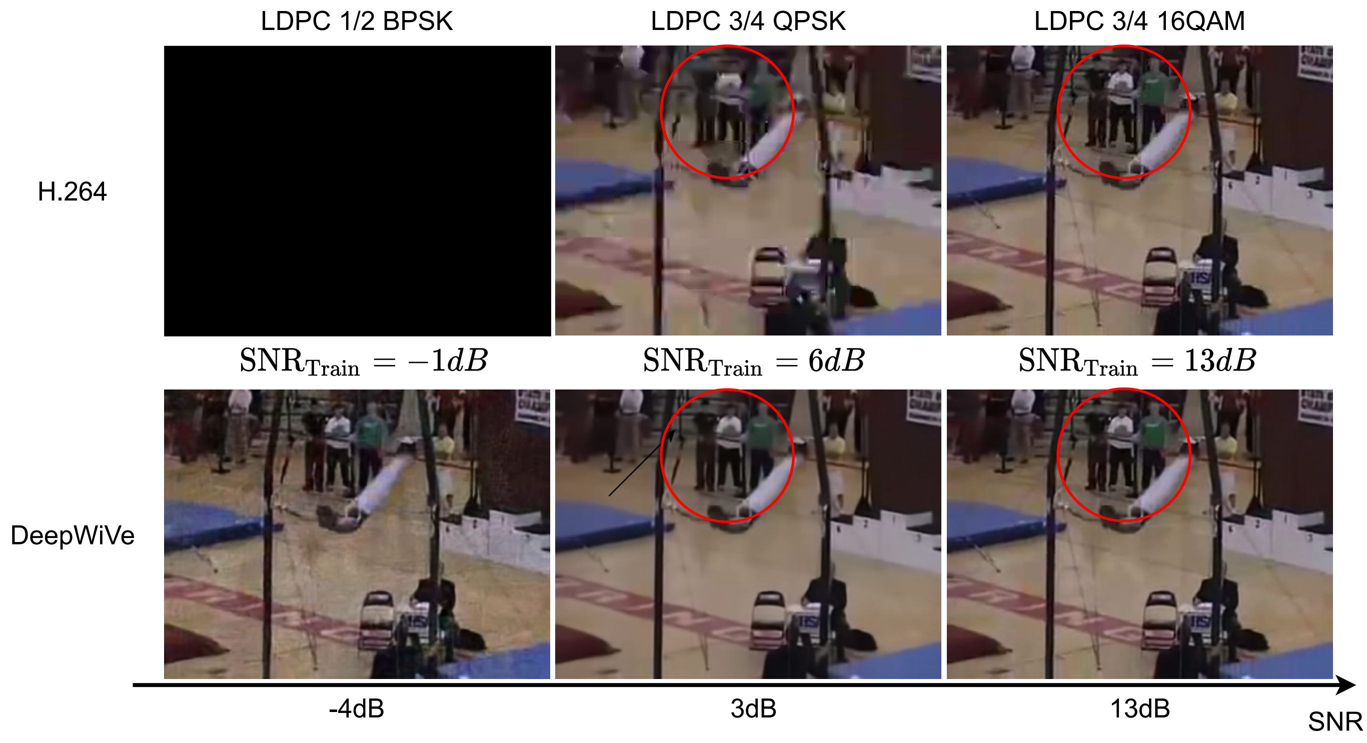

LDPC 1/2 BPSK
LDPC 3/4 QPSK
LDPC 3/4 16QAM
H.264
$\text{SNR}_{\text{Train}} = -1dB$
$\text{SNR}_{\text{Train}} = 6dB$
$\text{SNR}_{\text{Train}} = 13dB$
DeepWiVe
-4dB
3dB
13dB
SNR

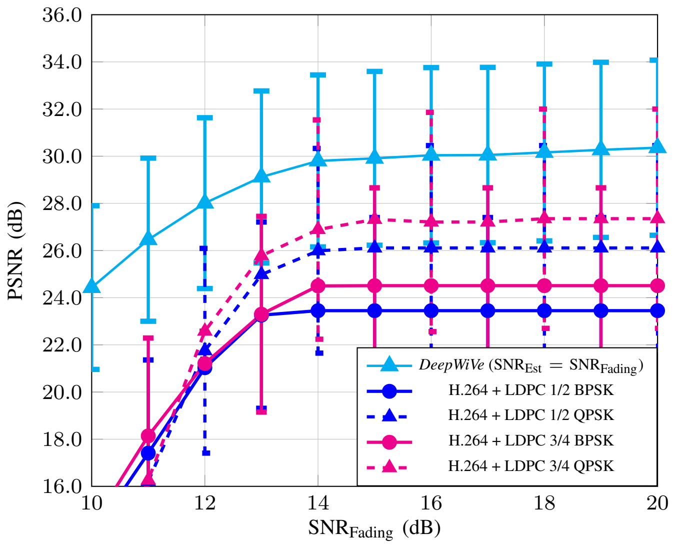
Fig. 13. Visual examples of *DeepWiVe* v.s. H.264. Difference in video quality can be seen most clearly in the 3 people standing at the back of the scene (encircled).

of the videos produced by H.264 and *DeepWiVe* are similar. However, at $\text{SNR}_{\text{AWGN}} = 3$ dB, the video produced by H.264 starts to look very pixelated, while *DeepWiVe* is still able to retain a smooth looking frame. At $\text{SNR}_{\text{AWGN}} = -4$ dB, the capacity of the channel is too low for H.264 to compress the video sufficiently, therefore the output is simply black, while *DeepWiVe* is still able to achieve a reasonable video quality despite the very low channel SNR. On average, in the AWGN case and $\rho = 0.031$, *DeepWiVe* outperforms H.264 by 0.46 dB in PSNR and by 0.0081 in MS-SSIM for $\text{SNR}_{\text{AWGN}} \in [13, 20]$ dB, by 3.07 dB in PSNR and by 0.0485 in MS-SSIM for $\text{SNR}_{\text{AWGN}} \in [3, 6]$ dB. *DeepWiVe* falls short of H.265 by 3.22 dB in PSNR, but outperforms it by 0.0006 in MS-SSIM for $\text{SNR}_{\text{AWGN}} \in [13, 20]$ dB. Similarly, it is 0.61 dB worse than H.265 in PSNR but outperforms it by 0.0069 in MS-SSIM for $\text{SNR}_{\text{AWGN}} \in [3, 6]$ dB. With respect to complexity, we use the NVIDIA TensorRT framework to optimize the inference time of our models and found that the average inference time of *DeepWiVe* is approximately 26 ms. On the other hand, only the encoding time of H.264 took on average 24 ms, using hardware acceleration on the Intel i9-9900K CPU. H.265 is even slower, at 92 ms. Therefore, *DeepWiVe* can be extremely efficient in practice using optimized hardware and library, more so than separation-based methods.

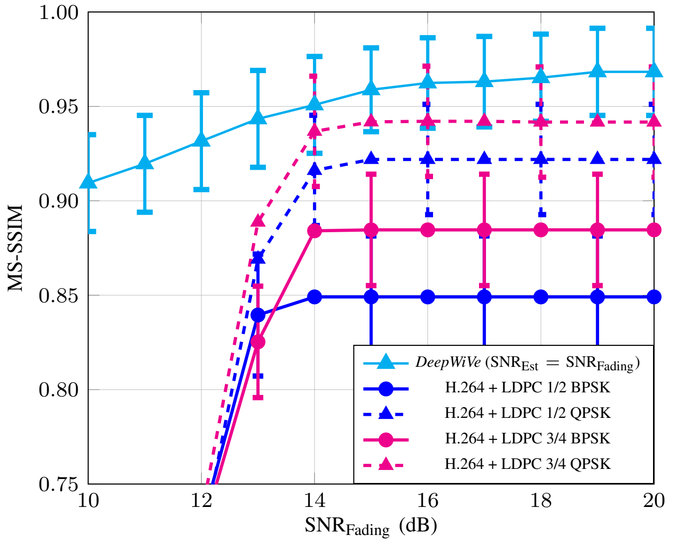
Next, we investigate the performance of *DeepWiVe* in the fading channel, where the transmitter only knows the average SNR and the receiver knows the average SNR as well as the phase of the channel gain $\arg(h^n)$ but not the magnitude $|h^n|$, $\forall n$. It is worth noting that, in this scenario, the capacity of this channel in the Shannon sense is zero, since no positive rate can be guaranteed for reliable transmission. Fig. 14 shows the performance of *DeepWiVe* trained for fading channel compared to H.264 paired with LDPC codes under the same CSI assumptions. It can be seen that the superiority of *DeepWiVe* over H.264 using LDPC codes, as seen in Fig. 11, is replicated in this case. Moreover, it can be seen that the performance of separation deteriorates at a much higher average SNR than in the full CSI knowledge case, showing the impact of not knowing the channel gain magnitude $|h^n|$ has on the channel codes. On average, in the fading channel case and $\rho = 0.031$, *DeepWiVe* outperforms H.264 by 0.64 dB in PSNR and by 0.0166 in MS-SSIM for $\text{SNR}_{\text{Fading}} \in [14, 20]$ dB.

In Fig. 15, we investigate the variable bandwidth transmission capability of *DeepWiVe* by decreasing the bandwidth compression ratio ρ, thereby increasing the compression of the video. To change ρ, we do not require the retraining of the autoencoder networks $(f_{\boldsymbol{\theta}}, f_{\boldsymbol{\theta}'}, g_{\boldsymbol{\phi}}, g_{\boldsymbol{\phi}'}, h_{\boldsymbol{\eta}})$; we only need to retrain the bandwidth allocator $q_{\boldsymbol{\psi}}$ with a different action set. As shown in Fig. 15, we see that *DeepWiVe* beats H.264 with LDPC coding for all the bandwidth compression ratios tested in terms of both the PSNR and MS-SSIM metrics. It also beats H.265 using the MS-SSIM metric as shown in Fig. 15b, although again, it falls short of H.265 in terms of the PSNR metric (Fig. 15a). This shows that *DeepWiVe* can achieve variable bandwidth transmission using RL to allocate an arbitrary number of blocks to meet the desired transmission bandwidth, as outlined in Section III-B.

Lastly, as an ablation study, we evaluate the performance of our models with and without optimal bandwidth allocation. We compare the results obtained by using the allocation network

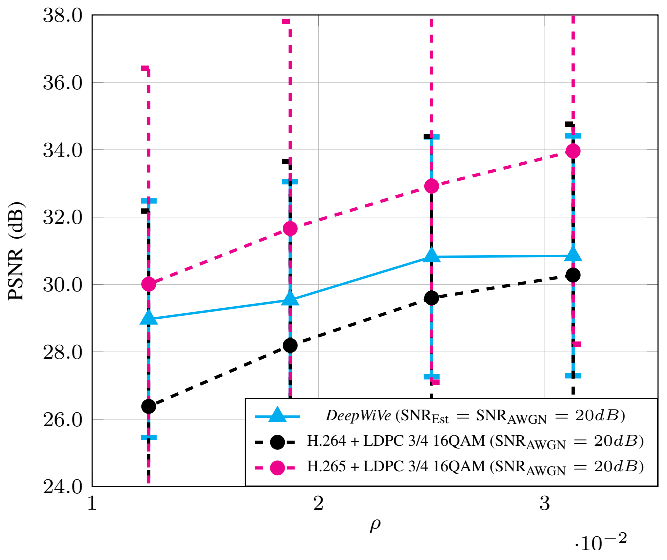


(a) PSNR

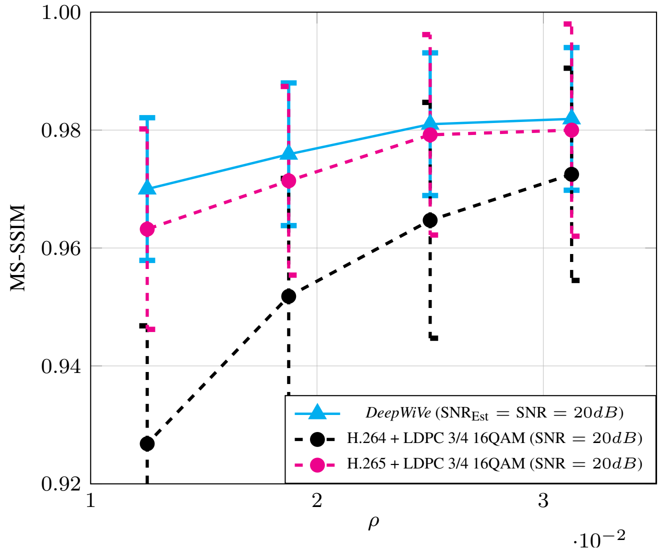


(b) MS-SSIM

Fig. 14. Performance comparison of DeepWiVe to H.264 paired with LDPC codes in the fading channel case ($\rho = 0.031$).



(a) PSNR



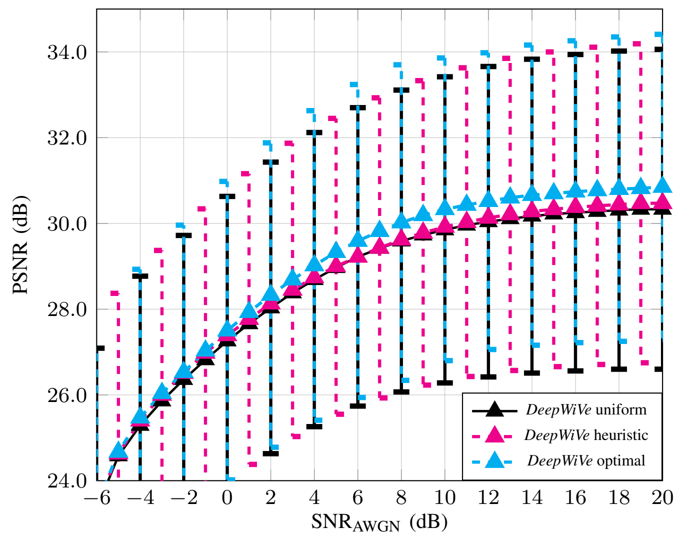
(b) MS-SSIM

Fig. 15. Performance comparison of DeepWiVe to H.264 and H.265 paired with LDPC codes as a function of bandwidth compression ratio ρ in the AWGN channel case (SNR$_{\text{AWGN}} = 20dB$).

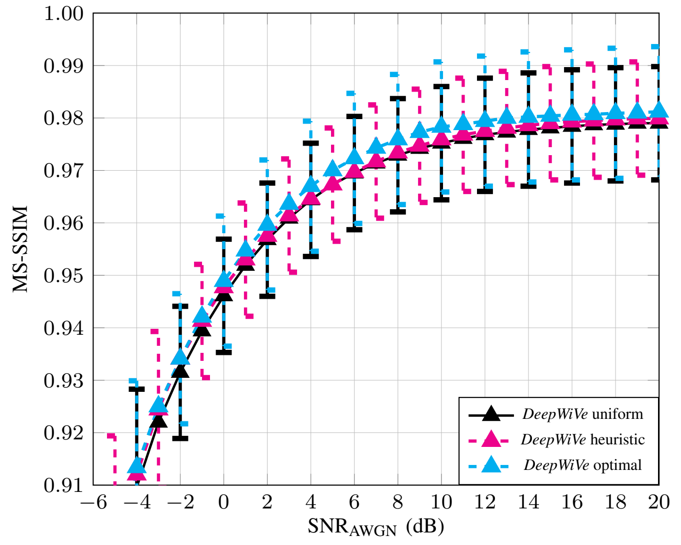
q_ψ with that of uniform allocation (i.e., $u_i^n = 5,\ \forall i, n$ for $\rho = 0.031$) and a heuristic bandwidth allocation policy. For the latter, we choose to allocate 50% of the bandwidth to the key frame and then allocate the remaining 50% to interpolated frames based on the magnitude of their SSF with respect to the reference frames. That is, let $m_i^n = ||\boldsymbol{\delta}_{i+t}^n||_2 + ||\boldsymbol{\delta}_{i-t}^n||_2$ be the sum of the magnitudes of the two SSFs of the ith frame in the nth GoP with respect to the two reference frames $\bar{\mathbf{x}}_{i-t}^n$, $\bar{\mathbf{x}}_{i+t}^n$. Then the bandwidth allocation for the ith frame in the nth GoP is calculated as

$$u_{i,heuristic}^n = \frac{Ue^{-m_i^n}}{\sum_{i=1}^3 e^{-m_i^n}} \tag{39}$$

rounded to the nearest integer. The intuition behind this heuristic policy is that the greater the magnitude of the SSF, the more pixel warping is needed to interpolate the frame from its reference frames. Therefore, more bandwidth should be allocated to such frames, and since the reconstruction quality of the key frame affects the reconstruction quality of the remaining frames in the GoP, we allocate half of the bandwidth to it. In Fig. 16, it can be seen that there is a clear and significant improvement in performance over both uniform and heuristic allocation when using our allocation network q_ψ. It can also be seen that the heuristic allocation policy improves upon the uniform allocation policy, emphasizing the importance of the key frame reconstruction quality for the performance. Overall, our bandwidth allocation network improves upon the uniform allocation policy by 0.35 dB in PSNR and by 0.0025 in MS-SSIM, for $\rho = 0.031$. It also improves upon the heuristic allocation policy by 0.25 dB in PSNR and by 0.0015 in MS-SSIM, for the same ρ.



(a) PSNR



(b) MS-SSIM

Fig. 16. Comparison of uniform, heuristic and optimal bandwidth allocation via auxiliary bandwidth allocation network q_ψ in the AWGN channel case ($\rho = 0.031$).

V. Conclusion

We presented the first ever DNN-aided joint source-channel wireless video transmission scheme in the literature. Our novel architecture, called *DeepWiVe*, is capable of dynamic bandwidth allocation and residual estimation without the need for distortion feedback. Additionally, it utilizes RL to learn a bandwidth allocation network that optimizes the allocation of available bandwidth within a given GoP in a dynamic fashion with the goal of maximizing the visual quality of the video under the given bandwidth constraint. Our results show that *DeepWiVe* overcomes the *cliff-effect* that all separation-based schemes suffer from, and achieves a graceful degradation with channel quality. In highly bandwidth constrained scenarios, *DeepWiVe* produces far superior video quality compared to both H.264 and H.265. We also show that our bandwidth allocation strategy is effective, improving upon the naïve uniform allocation by up to 0.35 dB in PSNR and the heuristic policy by 0.25 dB. Our overall results show that *DeepWiVe* is better than the separation-based schemes using industry standard H.264 codec and LDPC channel codes in all the channel conditions considered. Although, H.265 performs better than *DeepWiVe* in terms of the PSNR metric, *DeepWiVe* outperforms H.265 when compared in terms of MS-SSIM, which is widely accepted as a performance measure that better represents human perceptual quality.

As part of future work, the coding efficiency of *DeepWiVe* may be improved by considering variable GoP sizes. In most video compression codecs, variable GoP sizes are used as the entropy of different scenes changes depending on the amount of motion. The performance of the models should also be evaluated in practical channels, using, for example, software defined radios (SDRs).

References

[1] "Cisco visual networking index: forecast and methodology 2016-2021.," 2017.

[2] R. G. Gallager, *Information Theory and Reliable Communication*. USA: John Wiley & Sons, Inc., 1968.

[3] V. Kostina and S. Verdú, "Lossy joint source-channel coding in the finite blocklength regime," *IEEE Transactions on Information Theory*, vol. 59, pp. 2545–2575, May 2013.

[4] T. Goblick, "A coding theorem for time-discrete analog data sources," *IEEE Transactions on Information Theory*, vol. 15, pp. 401–407, May 1969.

[5] M. Gastpar, B. Rimoldi, and M. Vetterli, "To code, or not to code: Lossy source-channel communication revisited," *IEEE Transactions on Information Theory*, vol. 49, pp. 1147–1158, May 2003.

[6] G. Cheung and A. Zakhor, "Bit allocation for joint source/channel coding of scalable video," *IEEE Transactions on Image Processing*, vol. 9, pp. 340–356, Mar. 2000.

[7] I. Kozintsev and K. Ramchandran, "Robust image transmission over energy-constrained time-varying channels using multiresolution joint source-channel coding," *IEEE Transactions on Signal Processing*, vol. 46, pp. 1012–1026, Apr. 1998.

[8] L. Kondi and A. Katsaggelos, "Joint source-channel coding for scalable video using models of rate-distortion functions," in *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings*, vol. 3, pp. 1377–1380 vol.3, May 2001.

[9] W. Ji, Z. Li, and Y. Chen, "Joint source-channel coding and optimization for layered video broadcasting to heterogeneous devices," *IEEE Transactions on Multimedia*, vol. 14, pp. 443–455, Apr. 2012.

[10] G. Cheung and A. Zakhor, "Joint source/channel coding of scalable video over noisy channels," *AIP Conference Proceedings*, vol. 387, pp. 957–962, Jan. 1997.

[11] L. Kondi, F. Ishtiaq, and A. Katsaggelos, "Joint source-channel coding for motion-compensated DCT-based SNR scalable video," *IEEE Transactions on Image Processing*, vol. 11, pp. 1043–1052, Sept. 2002.

[12] T. P.-c. Chen and T. Chen, "Adaptive joint source-channel coding using rate shaping," in *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 2, pp. II–1985–II–1988, May 2002.

[13] J. Wu, Y. Shang, J. Huang, X. Zhang, B. Cheng, and J. Chen, "Joint source-channel coding and optimization for mobile video streaming in heterogeneous wireless networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2013, p. 283, Dec. 2013.

[14] I. E. Aguerri and D. Gündüz, "Joint source-channel coding with time-varying channel and side-information," *IEEE Transactions on Information Theory*, vol. 62, pp. 736–753, Feb. 2016.

[15] A. Lapidoth and S. Tinguely, "Sending a bivariate Gaussian source over a Gaussian MAC with feedback," vol. 56, pp. 2246–2250, July 2007.

[16] S. Jakubczak and D. Katabi, "SoftCast: One-size-fits-all wireless video," in *Proceedings of the ACM SIGCOMM 2010 Conference*, SIGCOMM '10, pp. 449–450, 2010.

[17] T. Tung and D. Gündüz, "SparseCast: Hybrid digital-analog wireless image transmission exploiting frequency domain sparsity," *IEEE Communications Letters*, pp. 1–1, 2018.

[18] W. Yin, X. Fan, Y. Shi, R. Xiong, and D. Zhao, "Compressive sensing based soft video broadcast using spatial and temporal sparsity," *Mobile Networks and Applications*, vol. 21, pp. 1002–1012, Dec. 2016.

[19] A. Wang, B. Zeng, and H. Chen, "Wireless multicasting of video signals based on distributed compressed sensing," *Signal Processing: Image Communication*, vol. 29, pp. 599–606, May 2014.

[20] E. Bourtsoulatze, D. B. Kurka, and D. Gündüz, "Deep joint source-channel coding for wireless image transmission," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4774–4778, May 2019.

[21] E. Bourtsoulatze, D. Burth Kurka, and D. Gündüz, "Deep joint source-channel coding for wireless image transmission," *IEEE Trans. on Cognitive Comms. and Networking*, vol. 5, no. 3, pp. 567–579, 2019.

[22] D. B. Kurka and D. Gündüz, "Bandwidth-agile image transmission with deep joint source-channel coding," *IEEE Transactions on Wireless Communications*, vol. 20, no. 12, pp. 8081–8095, 2021.

[23] C.-Y. Wu, N. Singhal, and P. Krahenbuhl, "Video Compression through Image Interpolation," in *2018 European Conference on Computer Vision (ECCV)*, pp. 416–431, Sept. 2018.

[24] G. Lu, W. Ouyang, D. Xu, X. Zhang, C. Cai, and Z. Gao, "DVC: an end-to-end deep video compression framework," in *IEEE Conf. on Computer Vision and Pattern Recog. (CVPR)*, pp. 11006–11015, Sept. 2019.

[25] O. Rippel, S. Nair, C. Lew, S. Branson, A. G. Anderson, and L. Bourdev, "Learned Video Compression," pp. 3454–3463, Oct. 2019.

[26] A. Djelouah, J. Campos, S. Schaub-Meyer, and C. Schroers, "Neural inter-frame compression for video coding," in *2019 International Conference on Computer Vision (ICCV)*, pp. 6420–6428, Oct. 2019.

[27] A. Habibian, T. v. Rozendaal, J. M. Tomczak, and T. S. Cohen, "Video compression with rate-distortion autoencoders," in *2019 International Conference on Computer Vision (ICCV)*, pp. 7033–7042, Oct. 2019.

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

[29] 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 *2020 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8500–8509, June 2020.

[30] B. Liu, Y. Chen, S. Liu, and H.-S. Kim, "Deep learning in latent space for video prediction and compression," in *2021 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 701–710, 2021.

[31] Z. Hu, G. Lu, and D. Xu, "FVC: A new framework towards deep video compression in feature space," in *2021 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1502–1511, 2021.

[32] T. Wiegand, G. 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, pp. 560–576, July 2003.

[33] R. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, vol. 8, pp. 21–28, Jan. 1962.

[34] J.-R. Ohm and G. J. Sullivan, "High efficiency video coding: The next frontier in video compression [Standards in a Nutshell]," *IEEE Signal Processing Magazine*, vol. 30, pp. 152–158, Jan. 2013.

[35] M. Stoufs, A. Munteanu, J. Cornelis, and P. Schelkens, "Scalable joint source-channel coding for the scalable extension of H.264/AVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, pp. 1657–1670, Dec. 2008.

[36] H. Y. Shutoy, D. Gunduz, E. Erkip, and Y. Wang, "Cooperative source and channel coding for wireless multimedia communications," *IEEE Journal of Sel'd Topics in Signal Proc.*, vol. 1, no. 2, pp. 295–307, 2007.

[37] R. Xiong, F. Wu, X. Fan, C. Luo, S. Ma, and W. Gao, "Power-distortion optimization for wireless image/video SoftCast by transform coefficients energy modeling with adaptive chunk division," in *Visual Comms. and Image Proc. (VCIP)*, pp. 1–6, Nov. 2013.

[38] Z. Song, R. Xiong, S. Ma, and W. Gao, "Hybridcast: A wireless image/video SoftCast scheme using layered representation and hybrid digital-analog modulation," in *2014 IEEE International Conference on Image Processing (ICIP)*, pp. 6001–6005, Oct. 2014.

[39] H. Liu, R. Xiong, X. Fan, D. Zhao, Y. Zhang, and W. Gao, "CG-Cast: Scalable wireless image SoftCast using compressive gradient," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, pp. 1832–1843, June 2019.

[40] R. Xiong, J. Zhang, F. Wu, and W. Gao, "High quality image reconstruction via non-local collaborative estimation for wireless image/video softcast," in *2014 IEEE International Conference on Image Processing (ICIP)*, pp. 2542–2546, Oct. 2014.

[41] A. Trioux, F.-X. Coudoux, P. Corlay, and M. Gharbi, "Performance assessment of the adaptive GoP-size extension of the wireless SoftCast video scheme," in *2020 10th International Symposium on Signal, Image, Video and Communications (ISIVC)*, pp. 1–6, Apr. 2021.

[42] J. Xu, B. Ai, W. Chen, A. Yang, and P. Sun, "Wireless image transmission using deep source channel coding with attention modules," *IEEE Trans. on Circuits and Sys. for Video Tech.*, pp. 1–1, 2021.

[43] D. B. Kurka and D. Gündüz, "DeepJSCC-f: Deep joint source-channel coding of images with feedback," *IEEE Journal on Selected Areas in Information Theory*, vol. 1, pp. 178–193, May 2020.

[44] M. Morelli and U. Mengali, "A comparison of pilot-aided channel estimation methods for OFDM systems," *IEEE Transactions on Signal Processing*, vol. 49, pp. 3065–3073, Dec. 2001.

[45] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multiscale structural similarity for image quality assessment," in *Asilomar Conf. on Signals, Systems Computers*, vol. 2, pp. 1398–1402 Vol.2, Nov. 2003.

[46] K. Sayood, H. H. Otu, and N. Demir, "Joint source/channel coding for variable length codes," *IEEE Transactions on Communications*, vol. 48, pp. 787–794, May 2000.

[47] M. Loiacono, J. Johnson, J. Rosca, and W. Trappe, "Cross-layer link adaptation for wireless video," in *2010 IEEE International Conference on Communications*, pp. 1–6, May 2010.

[48] P. Zhao, Y. Liu, J. Liu, A. Argyriou, and S. Ci, "SSIM-based error-resilient cross-layer optimization for wireless video streaming," *Image Communication*, vol. 40, pp. 36–51, Jan. 2016.

[49] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1874–1883, June 2016. ISSN: 1063-6919.

[50] J. Ballé, V. Laparra, and E. P. Simoncelli, "Density Modeling of Images using a Generalized Normalization Transformation," *arXiv:1511.06281 [cs]*, Feb. 2016. arXiv: 1511.06281.

[51] Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, "Learned image compression with discretized Gaussian mixture likelihoods and attention modules," in *2020 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[52] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *2018 Conference on Computer Vision and Pattern Recognition (CVRP)*, pp. 7794–7803, June 2018.

[53] S. S. Beauchemin and J. L. Barron, "The computation of optical flow," *ACM Computing Surveys*, vol. 27, pp. 433–466, Sept. 1995.

[54] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3431–3440, June 2015.

[55] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.

[56] K. Soomro, A. R. Zamir, and M. Shah, "UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild," *arXiv:1212.0402 [cs]*, Dec. 2012. arXiv: 1212.0402.

[57] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," Oct. 2017.

[58] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv:1412.6980 [cs]*, Jan. 2017. arXiv: 1412.6980.

[59] D. Ma, F. Zhang, and D. Bull, "BVI-DVC: A training database for deep video compression," *IEEE Transactions on Multimedia*, 2021.

[60] S. Tomar, "Converting video formats with ffmpeg," *Linux Journal*, vol. 2006, no. 146, p. 10, 2006.