Deep-Learning-Aided Wireless Video Transmission

Tze-Yang Tung* and Deniz Gündüz*†
* Information Processing and Communications Lab (IPC-Lab), Imperial College London, UK
† Enzo Ferrari Department of Engineering, University of Modena and Reggio Emilia, Italy
{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 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 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 on average in terms of the multi-scale structural similarity index measure (MS-SSIM).**

***Index Terms*—Joint source-channel coding, wireless video transmission, deep learning, reinforcement learning.**

I. Introduction

Video content contributes to more than 80% of Internet traffic and this number is expected to increase [1]. Video compression is widely used to reduce the bandwidth requirement when transmitting video signals wirelessly. This follows the separation principle, where the end-to-end transmission problem is divided into source coding and channel coding. The former removes redundancy within the video data, such that a prescribed reconstruction quality is achieved, while the latter introduces structured redundancy to allow reliable decoding under the presence of channel noises.

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

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

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. An alternative to the separation-based architecture is joint source-channel coding (JSCC). It has been shown theoretically that for finite delay, JSCC achieves lower distortion for a given code length than separate source and channel coding [2]. Nevertheless, JSCC schemes have found limited use in practice due to the difficulty in designing such codes in the digital domain. An alternative is to design the transmission system without considering any digital interfaces. In [3] it was shown that deep neural networks (DNNs) can break the complexity barrier in designing effective JSCC schemes, focusing particularly on the image transmission problem. The authors showed that DNN based JSCC schemes not only provide graceful degradation with channel quality, but also achieve results superior to state-of-the-art separation-based digital designs.

Herein, we propose an end-to-end optimized deep learning-based JSCC solution for wireless video transmission, called *DeepWiVe*. *DeepWiVe* directly maps each group-of-pictures (GoP) of the video sequence to a limited channel bandwidth. Our results show that *DeepWiVe* can meet or beat industry standard video compression codecs, such as H.264, combined with low density parity check (LDPC) codes, in all the 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:

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 optimize the bandwidth allocation among video frames using RL.
3) Numerical results show that *DeepWiVe* is superior to industry standard H.264 [4] codec combined with state-of-the-art LDPC channel codes [5] in all the channel conditions considered, and can avoid the cliff-effect. This is particularly attractive for highly mobile scenarios, in which accurate channel estimation is challenging.

II. Related Work

JSCC for video delivery has consistently received attention over the years. The earliest work we could find is [6],

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. In a similar line of work, [7] 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. However, these types of schemes typically do not achieve adequate performance gains for the increased complexity they introduce.

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 [8]. SoftCast involves a hybrid digital and analog design by leveraging frequency domain sparsity. Since then, various works have improved upon [8] by optimizing different aspects of the hybrid digital and analog design [9]–[11]. 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 efficiently, or adapt to channel and network conditions dynamically.

The closest prior art to our work are [3], [12]–[14], which explore the JSCC image transmission problem. In the context of video transmission, there are unique challenges, such as exploiting the inter-frame redundancies to improve coding efficiency and optimizing resource allocation across frames. Therefore, extending the problem from image to video transmission is not a trivial task.

III. Problem Formulation

We consider the problem of wireless video transmission in a constrained bandwidth setting. 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. Here, H and W represent the height and width of the video frames. Each frame $\mathbf{x}_i^n$ is represented as a 24bit 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 additive white Gaussian noise (AWGN) channel output $\mathbf{y} = \mathbf{z} + \mathbf{n}$, to an approximate reconstruction of the original video sequence $\hat{\mathbf{X}} = D(\mathbf{y})$. Here, $\mathbf{n} \sim \text{CN}(0, \sigma^2\mathbf{I})$ follows a complex Gaussian noise distribution with zero mean and covariance $\sigma^2\mathbf{I}$ ($\mathbf{I}$ is the identity matrix). We will assume that the transmitter and receiver are able to estimate the noise power σ^2 using, for example, pilot symbols [15].

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 $\rho = \frac{k}{3HWN}$. We impose an average power constraint P per channel use at the transmitter, such that $\mathbb{E}_{\mathbf{z}}\left[||\mathbf{z}||_2^2\right] \leq TkP$, where the expectation is taken over the distribution of the encoder output. Accordingly, the channel signal-to-noise ratio (SNR) is defined as $\text{SNR} = 10\log_{10}(P/\sigma^2)$ dB.

We measure the average quality of the reconstructed video using two metrics: peak signal-to-noise ratio (PSNR) and MS-SSIM [16]. They are defined as

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

and

$$\text{MS-SSIM}(\mathbf{X}, \hat{\mathbf{X}}) = \frac{1}{TN}\sum_{n=1}^{T}\sum_{i=1}^{N} 1 - l_{\text{MS-SSIM}}(\mathbf{x}_i^n, \hat{\mathbf{x}}_i^n), \tag{2}$$

where $l_{\text{PSNR}}(\mathbf{x}_i^n, \hat{\mathbf{x}}_i^n) = \frac{1}{3HW}||\mathbf{x}_i^n - \hat{\mathbf{x}}_i^n||_2^2$, $l_{\text{MS-SSIM}}(\mathbf{x}_i^n, \hat{\mathbf{x}}_i^n) = 1 - \text{MS-SSIM}(\mathbf{x}_i^n, \hat{\mathbf{x}}_i^n)$. The goal is to maximize the video quality, measured by either Eqn. (1) or (2), between the input video $\mathbf{X}$ and its reconstruction $\hat{\mathbf{X}}$, under the given constraints on the available bandwidth ratio ρ and the 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 correspond to the weights of these DNNs.

Consider the nth GoP, $\mathbf{X}^n$. The last ($\mathbf{x}_N^n$) frame is called the key frame and is compressed and transmitted using the key frame encoder $f_{\boldsymbol{\theta}} : \mathbb{R}^{H\times W\times 3} \mapsto \mathbb{C}^k$, $\mathbf{z}_i^n = f_{\boldsymbol{\theta}}(\mathbf{x}_i^n, \hat{\sigma}^2),\ i = N$, where $\hat{\sigma}^2$ is the estimated channel noise power at the transmitter. Each element of $\mathbf{z}_i^n$, denoted by $z_{i,j}^n$, is first normalized according to

$$\hat{z}_{i,j}^n = \sqrt{kP}\frac{z_{i,j}^n}{\sqrt{(\mathbf{z}_i^n)^H\mathbf{z}_i^n}},\ j = 1, \ldots, k, \tag{3}$$

where H refers to the Hermitian transpose. These values are then directly sent through the channel as, $\hat{\mathbf{y}}_i^n = \hat{\mathbf{z}}_i^n + \mathbf{n}$. 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 $\hat{\mathbf{x}}_i^n = f_{\boldsymbol{\theta}'}(\hat{\mathbf{y}}_i^n, \hat{\sigma}^2),\ i = N$. The loss between the original frame $\mathbf{x}_i^n$ and the reconstructed frame $\hat{\mathbf{x}}_i^n$ is computed using l_{PSNR} or $l_{\text{MS-SSIM}}$ 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. 1. The attention feature (AF) module, proposed by [12], allows the network to learn to assign different weights to different features for a given SNR, similar to resource assignment strategies in traditional JSCC schemes [17]. By randomizing the channel SNR during training, the AF module allows us to obtain a single model that works over a range of SNRs. The Attention layer refers to the simplified

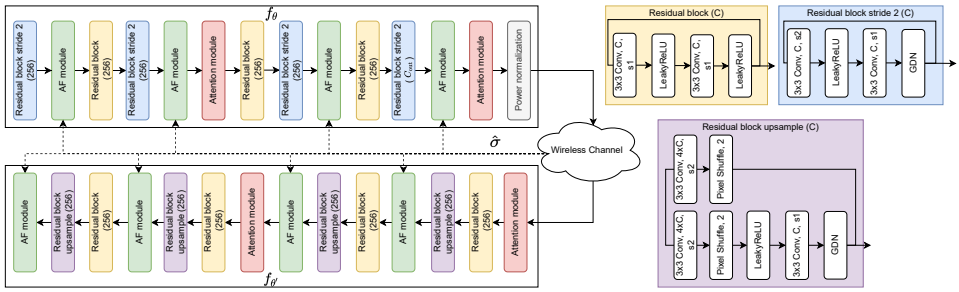


Fig. 1. Key frame encoder/decoder $(f_{\boldsymbol{\theta}}, f_{\boldsymbol{\theta}'})$ network architectures.

attention module proposed in [18], which has been shown to improve the compression efficiency by focusing the neural network on regions in the image that require higher bit rate.

For the remaining frames, i.e., $\mathbf{x}_i^n$, $i = 1, 2, \ldots, N-1$, 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 frame. The reference frames $\bar{\mathbf{x}}_{i-t}^n$, $\bar{\mathbf{x}}_{i+t}^n$ 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 side 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 $\bar{\mathbf{x}}_{i+t}^n$, we use scaled space flow (SSF), which was first proposed by [19] as a more general description of pixel warping than optical flow [20], used in traditional video codecs. The idea is to blur regions of the frame where the motion is difficult to model 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 $\tilde{\mathbf{x}}_{i+t}^n = \text{SSW}(\bar{\mathbf{x}}_{i+t}^n, \boldsymbol{\delta}_{i+t}^n) = \bar{\mathbf{X}}_{i+t}^n[x + \boldsymbol{\delta}_{i+t}^n[x, y, 1], y + \boldsymbol{\delta}_{i+t}^n[x, y, 2], \boldsymbol{\delta}_{i+t}^n[x, y, 3]]$. 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 [19], to obtain $\boldsymbol{\delta}_{i+t}^n = h_{\boldsymbol{\eta}}(\mathbf{x}_i^n, \bar{\mathbf{x}}_{i+t}^n)$. Given the above definition of SSF, the residual $\mathbf{r}_{i+t}^n$ is defined as $\mathbf{r}_{i+t}^n = \mathbf{x}_i^n - \tilde{\mathbf{x}}_{i+t}^n$. The interpolation encoder $g_{\boldsymbol{\phi}} : \mathbb{R}^{H \times W \times 21} \mapsto \mathbb{C}^k$ defines the mapping $\mathbf{z}_i^n = g_{\boldsymbol{\phi}}(\mathbf{x}_i^n, \bar{\mathbf{x}}_{i-t}^n, \bar{\mathbf{x}}_{i+t}^n, \mathbf{r}_{i-t}^n, \mathbf{r}_{i+t}^n, \boldsymbol{\delta}_{i-t}^n, \boldsymbol{\delta}_{i+t}^n, \hat{\sigma}^2)$, $i = 1, 2, \ldots, N-1$. The vector $\mathbf{z}_i^n \in \mathbb{C}^k$ is power normalized according to Eqn. (3) and sent across the channel as $\hat{\mathbf{y}}_i^n = \hat{\mathbf{z}}_i^n + \mathbf{n}$.

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 $(\hat{\boldsymbol{\delta}}_{i-t}^n, \hat{\boldsymbol{\delta}}_{i+t}^n, \hat{\mathbf{r}}_i^n, \mathbf{m}_i^n) = g_{\boldsymbol{\phi}'}(\hat{\mathbf{y}}_i^n, \hat{\sigma}^2)$, 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 $\sum_{c=1}^3 \mathbf{m}_{i,c}^n = \mathbf{1}_{H \times W}$. 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 $\hat{\mathbf{x}}_i^n = (\mathbf{m}_i^n)_1 * \text{SSW}(\hat{\mathbf{x}}_{i-t}^n, \hat{\boldsymbol{\delta}}_{i-t}^n) + (\mathbf{m}_i^n)_2 * \text{SSW}(\hat{\mathbf{x}}_{i+t}^n, \hat{\boldsymbol{\delta}}_{i+t}^n) + (\mathbf{m}_i^n)_3 * \hat{\mathbf{r}}_i^n$, where $*$ refers to element-wise multiplication. 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.

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. 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.

Herein, we define each GoP as one time step with the nth state $\mathbf{s}^n = \{\{\bar{\mathbf{x}}_i^n\}_{i=0}^N, \{\mathbf{r}_{i\pm t}^n\}_{i=1}^{N-1}, \{\boldsymbol{\delta}_{i\pm t}^n\}_{i=1}^{N-1}, \hat{\sigma}^2\}$, where $\bar{\mathbf{x}}_0^n = \bar{\mathbf{x}}_N^{n-1}$. The action set $\mathcal{A}$ is the set of all the different

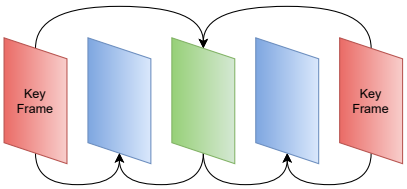


Fig. 2. Diagram of a typical interpolation structure used in video compression algorithm.

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 [13], 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. This training process leads to the descending ordering of information from $\mathbf{z}_{i,1}^n$ to $\mathbf{z}_{i,U}^n$. 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. We define the reward function as $r^n = -\log_{10}(l(\mathbf{X}^n, \hat{\mathbf{X}}^n))$, where $l(\cdot,\cdot)$ is either l_{PSNR} or $l_{\text{MS-SSIM}}$ depending on the 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 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 learn an optimal allocation policy, we use deep Q-learning [21], 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,

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

As is typical in DQN methods, we use *replay buffer*, *target network*, and ϵ-*greedy* to aid the learning of the Q-function. We use target parameters $\boldsymbol{\psi}^-$, which are copies of $\boldsymbol{\psi}$, to compute the DQN loss function:

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

The parameters $\boldsymbol{\psi}$ are then updated via gradient descent according to the gradient $\nabla_{\boldsymbol{\psi}} L_{\text{DQN}}(\boldsymbol{\psi})$, while the target network parameters are updated via $\boldsymbol{\psi}^- \leftarrow \tau\boldsymbol{\psi} + (1-\tau)\boldsymbol{\psi}^-$, $0 \le \tau \le 1$. To promote exploration, we use ϵ-greedy, which chooses a uniformly random action with probability ϵ at each GoP. A diagram of the architecture used for $q_{\boldsymbol{\psi}}$ is shown in Fig. 3.

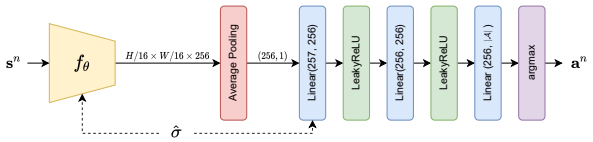


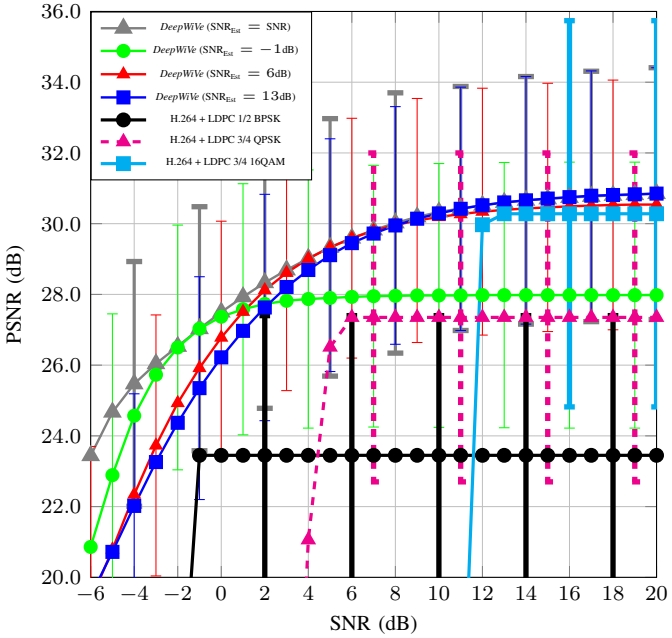
Fig. 3. Architecture of the bandwidth allocation network $q_{\boldsymbol{\psi}}$. The convolutional part of the network for feature extraction is functionally the same as the key encoder network ($f_{\boldsymbol{\theta}}$) but with $21(N-1)+6$ channels to account for all the tensors in the state $\mathbf{s}^n$.

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.

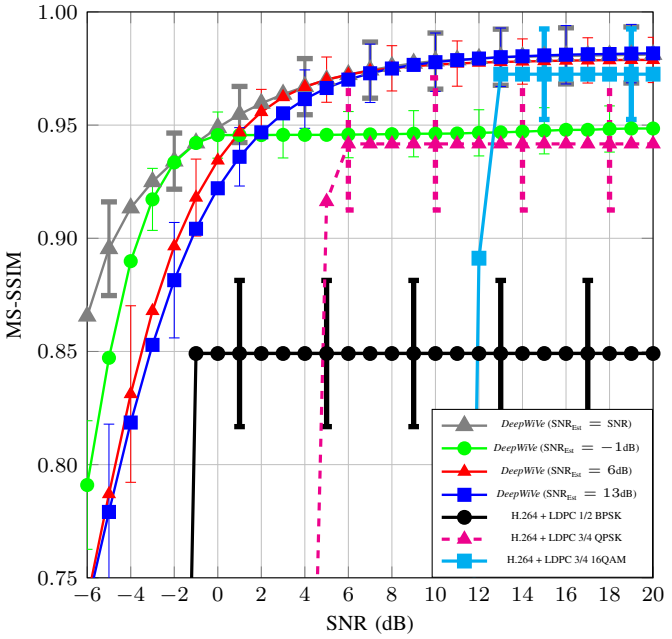
IV. Numerical Results

We train our models on the UCF101 dataset [22] using Pytorch [23], with the Adam optimizer [24] at learning rate $1e^{-4}$. We then test the model using the BVI-DVC dataset [25]. We train the JSCC ($f_{\boldsymbol{\theta}}, f_{\boldsymbol{\theta}'}, g_{\boldsymbol{\phi}}, g_{\boldsymbol{\phi}'}, h_{\boldsymbol{\eta}}$) networks first until convergence, before we train the bandwidth allocation network $q_{\boldsymbol{\psi}}$ to find the optimal bandwidth allocation policy. We define the SNR estimated by the transmitter and receiver to be $\text{SNR}_{\text{Est}} = 10\log_{10}(P/\hat{\sigma}^2)$. 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 $\epsilon = \epsilon_{\text{end}} + (\epsilon_0 - \epsilon_{\text{end}})\exp(-\text{episode}/\lambda)$, where λ controls the decay rate of ϵ. We choose $\epsilon_0 = 0.9$, $\epsilon_{\text{end}} = 0.05$, and $\lambda = 1000$. 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. We let $P = 1$ and compute the necessary σ^2 to achieve the desired SNR.

We compare the performance of our model with that of the conventional separation-based schemes. In particular, we use the H.264 [4] codec for source coding, LDPC codes [5] for channel coding, and QAM modulation. 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. 4, we show the effect of channel estimation error on the performance of *DeepWiVe* in an AWGN channel by fixing SNR_{Est}. It is clear that *DeepWiVe* is able to overcome the *cliff-effect*, as video quality gracefully degrades as the SNR decreases for a given SNR_{Est}. This is in contrast to the cliff edge drop off that separation-based designs suffer from. When given accurate estimate of the channel SNR (i.e., $\text{SNR}_{\text{Est}} = \text{SNR}$), we see that *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, demonstrating the theoretical superiority of JSCC in finite block length regimes, as shown in [2]. On average, for the AWGN channel and $\rho = 0.031$, *DeepWiVe* is 0.46dB better in PSNR and 0.0081 better in MS-SSIM than H.264 for SNR $\in [13, 20]$dB, 3.07dB better in PSNR and 0.0485



(a) PSNR



(b) MS-SSIM

Fig. 4. Comparison of *DeepWiVe* to H.264 paired with LDPC codes ($\rho = 0.031$).

better in MS-SSIM for SNR $\in [3, 6]$dB. For more results, including ablation studies of different bandwidth allocation schemes, please refer to the journal version [26].

V. Conclusion

We presented the first ever DNN-aided joint source-channel wireless video transmission scheme in the literature, called *DeepWiVe*. *DeepWiVe* is capable of dynamic bandwidth allocation and residual estimation without the need for distortion feedback. Our results show that *DeepWiVe* overcomes the *cliff-effect* that all separation-based schemes suffer from, and achieves a graceful degradation with channel quality. *DeepWiVe* also outperforms the separation-based scheme using industry standard H.264 codec and LDPC channel codes in all the channel conditions considered.

References

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

[2] 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.

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

[4] 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.

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

[6] 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.

[7] 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.

[8] S. Jakubczak and D. Katabi, "SoftCast: One-size-fits-all wireless video," in *Proceedings of the ACM SIGCOMM 2010 Conference*, Sept. 2010.

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

[10] 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.

[11] 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.

[12] J. Xu, B. Ai, W. Chen, A. Yang, P. Sun, and M. Rodrigues, "Wireless image transmission using deep source channel coding with attention modules," *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1–1, 2021.

[13] D. B. Kurka and D. Gündüz, "Bandwidth-agile image transmission with deep joint source-channel coding," *IEEE Transactions on Wireless Communications*, pp. 1–1, 2021.

[14] 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.

[15] 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.

[16] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multiscale structural similarity for image quality assessment," in *The Thrity-Seventh Asilomar Conference on Signals, Systems Computers, 2003*, vol. 2, Nov. 2003.

[17] 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.

[18] 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.

[19] 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.

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

[21] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.

[22] 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.
[23] A. Paszke *et al.*, "Automatic differentiation in PyTorch," Oct. 2017.
[24] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv:1412.6980 [cs]*, Jan. 2017. arXiv: 1412.6980.
[25] D. Ma, F. Zhang, and D. Bull, "Bvi-dvc: A training database for deep video compression," *IEEE Transactions on Multimedia*, 2021.
[26] T.-Y. Tung and D. Gündüz, "DeepWiVe: Deep-Learning-Aided Wireless Video Transmission," *arXiv:2111.13034 [cs, eess]*, Nov. 2021.