Image Semantic Communication with Quadtree Partition-based Coding

Yinhuan Huang, *Student Member, IEEE,* Zhijin Qin, *Senior Member, IEEE*

***Abstract*—Deep learning based semantic communication (DeepSC) system has emerged as a promising paradigm for efficient wireless transmission. However, existing image DeepSC methods, frequently encounter challenges in balancing rate-distortion performance and computational complexity, and often exhibit inferior performance compared to traditional schemes, especially on high-resolution datasets. To address these limitations, we propose a novel image DeepSC system, using quadtree partition-based joint semantic-channel coding, named Quad-DeepSC, which maintains low complexity while achieving state-of-the-art transmission performance. Based on maturing learned image compression technologies, we establish a unified DeepSC system design and training pipeline. The proposed Quad-DeepSC integrates quadtree partition-based entropy estimation and feature coding modules with lightweight feature extraction and reconstruction networks to form an end-to-end architecture. During training, all components except the feature coding modules are jointly optimized as a compact learned image codec, Quad-LIC, for source compression tasks. The pretrained Quad-LIC is then embedded into Quad-DeepSC and fine-tuned end-to-end over wireless channels. Extensive experimental results demonstrate that Quad-DeepSC is the first DeepSC system to surpass conventional communication systems, which employ VTM for source coding and adopt the optimal MCS index under 3GPP standards for channel coding and digital modulation, in performance across datasets of varying resolutions. Notably, both Quad-DeepSC and Quad-LIC exhibit minimal latency, rendering them well-suited for deployment in real-time wireless communication systems.**

***Index Terms*—Deep learning, semantic communication, image transmission, image compression.**

I. Introduction

In recent years, the vision of the sixth-generation (6G) mobile communication system has emerged, aiming for a deep integration of artificial intelligence (AI) and communications, with an emphasis on ultra-reliable, low-latency transmission and extended functional capabilities beyond the fifth generation (5G) [1]–[3]. Meanwhile, the explosive growth of digital traffic and the rising demand for ultra-large-scale image and video interactions have imposed stringent requirements on existing communication systems [4], necessitating efficient transmission schemes to lower the network traffic and mitigate channel impairments.

Conventional communication systems primarily emphasize source coding design in an effort to meet the escalating demands. Traditional lossy image compression standards, including better portable graphics (BPG), and versatile video coding (VVC), have achieved remarkable compression performance. Among these, VVC is widely regarded as one of the most advanced manually designed codecs in terms of reconstruction quality, as measured by peak signal-to-noise ratio (PSNR) and multi-scale structural similarity (MS-SSIM) [5]. However, this advancement comes at the expense of increased computational complexity. As the number of partitioning modes and prediction directions increases, the encoding latency for searching the optimal rate-distortion (R-D) configuration grows, and the potential for further optimization becomes limited [6].

Data-driven AI technologies enable two main approaches for image source coding: one is to enhance traditional image codecs [7] and the other is to develop learned image codecs (LICs). The latter has undergone substantial development, as it employ neural networks to optimize the entire semantic coding process, including nonlinear transforms, latent distributions estimation, and quantization. LICs have demonstrated the ability to effectively capture spatial dependencies among image pixels. For instance, ELIC [5] outperforms VVC in still image compression and is regarded as a promising candidate for next-generation standards. However, the performance of existing LICs is still limited by the inherent trade-off between R-D performance and computational complexity. Additionally, source coding often optimize independently of transmission constraints. When LICs are adopted in conventional image communication systems, channel impairments and floating-point arithmetic inaccuracies across heterogeneous hardware [8] can severely degrade reconstruction quality, as LICs rely on probability estimates with extremely high precision for entropy coding, thereby limiting their applicability.

Deep learning based semantic communication (DeepSC) systems have rapidly emerged as a novel paradigm that fully incorporates AI and are capable of merging the physical layer of conventional communication systems [9]. By adopting joint semantic-channel coding (JSCC), DeepSC systems can semantically perceive source information, extract task-relevant features, and apply importance-aware unequal error protection, thereby effectively addressing the limitations of conventional communication systems. The pioneering work [10] proposed the first image DeepSC system. However, it exhibited several limitations, such as fixed output symbol length per model, due to static bandwidth allocation. Building upon this framework, [11] proposed a symbol-layering mechanism for progressive transmission and [12] made the model bit-based, enabling integration into existing digital communication systems. In contrast, [13] adopted a different DeepSC framework by incorporating a LIC into the JSCC architecture for semantic feature extraction, entropy estimation, and reconstruction. [14]

Yinhuan Huang and Zhijin Qin are with the Department of Electronic Engineering, Tsinghua University, Beijing 100084, China, and with the State Key Laboratory of Space Network and Communications, Beijing, 100084, China. (email: huangyh24@mails.tsinghua.edu.cn; qinzhijin@tsinghua.edu.cn).

Our project will be available at https://github.com/hyh-bingo/Quad-LIC_Quad-DeepSC.

enhanced this architecture by integrating a LIC with checkerboard partition-based entropy model [15], achieving superior performance that surpassed the image communication scheme, which employs VTM [16] (state-of-the-art engineered image codec of the VVC standard) for source coding and adopt the optimal modulation and scheme (MCS) index under the 3GPP standards [17] for channel coding and digital modulation, on medium-resolution image datasets. [18] introduced digital modulation into the framework proposed in [13], further boosting performance improvements. Similar to [13], [19] adopted a variational autoencoder-based LIC and outperformed [11] in progressive transmission.

These developments of image DeepSC systems underscore a key insight: DeepSC system employs JSCC, which is built upon a LIC, significantly enhancing the transmission efficiency. However, not all LICs can be seamlessly integrated into a DeepSC framework. For wireless communication deployment, a DeepSC system must achieve high transmission quality and computational efficiency. Existing studies have not yet established effective methods to adapt LICs for DeepSC or to design and train DeepSC models when low-complexity, high-efficiency LICs are unavailable. The degree to which current LIC techniques can be leveraged remains an open issue.

To address these challenges, we revisit the entropy estimation, which fundamentally determines the efficiency of both LIC and its DeepSC extension. In entropy estimation, latent features are hierarchically partitioned into groups, where each group is conditioned on the estimation of previous ones to model the probability distribution of the current group. Over the years, various partitioning strategies have been explored to balance R-D performance and computational complexity. Traditional channel-wise autoregressive partitioning [20] achieves high accuracy but incurs heavy computation. Checkerboard partitioning [15] simplifies spatial dependency into two sequential steps, improving speed but reducing accuracy. Hybrid methods [5] divide features into multiple channel groups with checkerboard refinement, balancing accuracy and speed. More recent approaches, such as quadtree partitioning [6], [21] and quincunx partitioning [22], further exploit joint spatial–channel correlations, achieving superior trade-offs.

Building upon these insights, we propose a unified design and training strategy for the DeepSC framework, emphasizing efficient entropy estimation and feature coding based on quadtree partitioning. In this design, the feature coding module maps latent features to transmitted symbols following the same conditional and hierarchical structure used in entropy estimation. This coherence ensures that symbol lengths align with their estimated probabilities, while inter-symbol dependencies enhance robustness against channel impairments. The proposed modules are integrated with feature extraction and reconstruction networks to form an end-to-end DeepSC system, termed Quad-DeepSC. From this architecture, a corresponding LIC, named Quad-LIC, is derived, consisting of feature extraction, entropy estimation, and reconstruction components. Quad-LIC is first pretrained and subsequently embedded into Quad-DeepSC for end-to-end optimization over fading channels, achieving substantial gains in transmission efficiency. The main contributions of this work are summarized as follows:

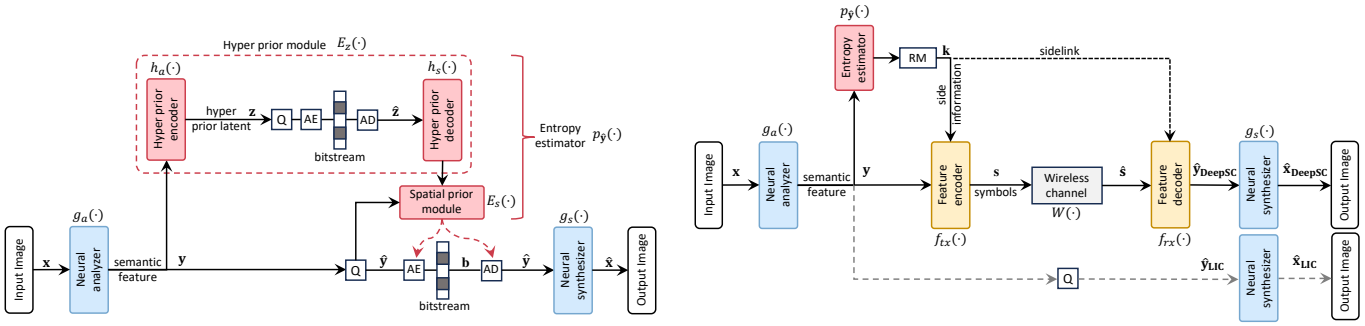
1) A unified design and training strategy is developed to bridge LIC and semantic communication. Rather than relying on pretrained black-box LICs, the method first constructs a DeepSC model based on efficient feature partitioning, from which specific modules are extracted to form a compact LIC. The LIC is trained on source compression tasks and subsequently embedded into DeepSC model for end-to-end transmission optimization, ensuring effective reuse of current LIC techniques and stable convergence under diverse channel conditions.
2) The proposed Quad-DeepSC architecture incorporates a quadtree partition-based entropy estimation and feature coding mechanism, in which hierarchical spatial–channel grouping jointly drives entropy modeling and feature-to-symbol transformation. This end-to-end design enhances coding efficiency and supports adaptive semantic transformation guided by entropy and contextual information.
3) An efficient LIC, Quad-LIC, is derived from the Quad-DeepSC architecture. It inherits the quadtree partition-based entropy estimation and achieves high compression performance with fast coding speed. The unified training pipeline allows Quad-LIC to operate both as a standalone codec and as an embedded module within Quad-DeepSC.
4) Extensive experiments demonstrate that Quad-DeepSC achieves state-of-the-art transmission efficiency and fast coding speed compared with existing DeepSC systems, while Quad-LIC exhibits competitive compression performance and verifies the effectiveness of the proposed hierarchical design strategy.

The rest of this paper is structured as follows. Section II presents the system models of LIC and DeepSC. Section III describes the architecture and implementations of Quad-LIC and Quad-DeepSC. Section IV presents the numerical experimental results. Finally, Section V concludes this paper.

Notation: Scalars, vectors, and matrices are represented by lowercase (e.g., x), bold lowercase (e.g., $\mathbf{x}$), and bold uppercase letters (e.g., $\mathbf{X}$), respectively. Additionally, $\mathbb{C}^{m\times n}$ and $\mathbb{R}^{m\times n}$ represent the space of $m\times n$ complex and real matrices, respectively. p_x is the probability density function for the continuous random variable x, and $P_{\bar{x}}$ is the probability mass function for the discrete random variable $\bar{x}$. $\mathbb{E}[\cdot]$ denotes the expectation operator, $||\cdot||_2$ denotes the euclidean norm and $\odot$ denotes the element-wise multiplication. θ denotes the neural network parameters, and subscripts indicate the corresponding module (e.g., θ_{g_a}). $\mathcal{N}(x|\mu,\sigma^2) = (2\pi\sigma^2)^{-1/2}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$ represents a gaussian distribution, and $\mathcal{U}(y-u, y+u)$ denotes a uniform distribution centered at y with range $[y-u, y+u]$.

II. System Model and Problem Formulation

In this section, we presents the system models of LIC and DeepSC, including problem formulations, variable definitions, loss functions, and detailed descriptions of the processing pipelines.



(a) System model of LIC. (b) System model of proposed DeepSC.

Fig. 1: System models of LIC and DeepSC. (a) The architecture of LIC consists of a neural analyzer $g_a(\cdot)$, a neural synthesizer $g_s(\cdot)$, and an entropy estimator $p_{\hat{\mathbf{y}}}(\cdot)$. The entropy estimator includes a hyper prior module $E_z(\cdot)$ and a spatial prior module $E_s(\cdot)$. The hyper prior module contains an encoder $h_a(\cdot)$ and a decoder $h_s(\cdot)$. AE, AD, and Q represent arithmetic encoding, arithmetic decoding, and quantization, respectively. (b) The proposed DeepSC system extends the LIC framework by adding a feature encoder $f_{tx}(\cdot)$, a feature decoder $f_{rx}(\cdot)$, and a fixed, non-trainable channel model $W(\cdot)$. RM indicates the rate matching operation estimating symbol lengths. Procedures shown with dashed-gray lines in (b) are used solely to stabilize the training of DeepSC model. The identical naming indicates that the corresponding modules are the same.

A. System Model of LIC

LIC is a data-driven R-D optimization (RDO) framework. Consider an image vector $\mathbf{x} \in \mathbb{R}^{H\times W\times C}$, where H, W and C denote the height, width and number of color channels, respectively. As illustrated in Fig. 1 (a), the LIC framework consists of a neural analyzer $g_a(\cdot)$, a neural synthesizer $g_s(\cdot)$, and an entropy estimator $p_{\hat{\mathbf{y}}}(\cdot)$. Given $\mathbf{x}$, the $g_a(\cdot)$ maps it to a semantic feature $\mathbf{y}$:

$$\mathbf{y} = g_a(\mathbf{x}; \theta_{g_a}). \tag{1}$$

The $\mathbf{y}$ is then quantized to $\hat{\mathbf{y}} = Q(\mathbf{y})$. Since $\hat{\mathbf{y}}$ is discrete, it can be compressed into a bitstream $\mathbf{b}$ using entropy coding techniques, such as arithmetic coding, and can be losslessly recovered from $\mathbf{b}$:

$$\mathbf{b} = AE(\hat{\mathbf{y}}, p_{\hat{\mathbf{y}}}(\hat{\mathbf{y}}; \theta_{p_{\hat{\mathbf{y}}}})), \tag{2}$$

$$\hat{\mathbf{y}} = AD(\mathbf{b}, p_{\hat{\mathbf{y}}}(\hat{\mathbf{y}}; \theta_{p_{\hat{\mathbf{y}}}})), \tag{3}$$

where $AE(\cdot)$ and $AD(\cdot)$ represent arithmetic encoding and arithmetic decoding. The $g_s(\cdot)$ is used to reconstruct the image $\hat{\mathbf{x}}$ from $\hat{\mathbf{y}}$:

$$\hat{\mathbf{x}} = g_s(\hat{\mathbf{y}}; \theta_{g_s}). \tag{4}$$

Due to the quantization process, reconstruction errors between $\mathbf{x}$ and $\hat{\mathbf{x}}$ are inevitable, which leads to the RDO problem. The loss function is expressed as:

$$L_{LIC} = \lambda \cdot \mathcal{D}(\mathbf{x}, \hat{\mathbf{x}}) + R, \tag{5}$$

where λ is a weighting coefficient, the distortion $\mathcal{D}(\cdot)$ is defined according to the downstream task, such as mean squared error (MSE) or MS-SSIM for image reconstruction, and the rate R represents the length of $\mathbf{b}$, defined by cross entropy:

$$R = \mathbb{E}_{\mathbf{x}\sim p_{\mathbf{x}}}\left[-\log_2 P_{\hat{\mathbf{y}}}\left(Q\left(g_a\left(\mathbf{x}\right)\right)\right)\right]. \tag{6}$$

For training model, [23] proposed using additive uniform noise to replace quantization:

$$\tilde{\mathbf{y}} = \mathbf{y} + \mathbf{u}, \tag{7}$$

where $\mathbf{u}$ denotes independent identically distributed (i.i.d.) samples from the $\mathcal{U}(-\frac{1}{2}, \frac{1}{2})$ for random quantization offsets. It is a technique widely adopted to address the zero gradient problem caused by quantization [24], [25], though quantization is still employed in testing model. The RDO problem can be formally described as a variational autoencoder [26]. $q(\tilde{\mathbf{y}}|\mathbf{x})$ estimated by $p_{\hat{\mathbf{y}}}(\cdot)$ is used to approximate the true posterior $p_{\tilde{\mathbf{y}}|\mathbf{x}}(\tilde{\mathbf{y}}|\mathbf{x})$ by minimizing the Kullback-Leibler (KL) divergence over the data distribution $p_{\mathbf{x}}(\cdot)$:

$$\begin{aligned}\mathbb{E}_{\mathbf{x}\sim p_{\mathbf{x}}}\left[\mathcal{D}_{\mathrm{KL}}\left(q(\tilde{\mathbf{y}}|\mathbf{x})\|p_{\tilde{\mathbf{y}}|\mathbf{x}}(\tilde{\mathbf{y}}|\mathbf{x})\right)\right] &= \mathbb{E}_{\mathbf{x}\sim p_{\mathbf{x}}}\,\mathbb{E}_{\tilde{\mathbf{y}}\sim q(\tilde{\mathbf{y}}|\mathbf{x})} \\ &\Big[\cancel{\log q(\tilde{\mathbf{y}}\mid\mathbf{x})}^{\,c_1} \underbrace{-\log p_{\mathbf{x}|\tilde{\mathbf{y}}}(\mathbf{x}\mid\tilde{\mathbf{y}})}_{\text{weighted distortion}} \underbrace{-\log p_{\tilde{\mathbf{y}}}(\tilde{\mathbf{y}})}_{\text{rate}}\Big] + c_2,\end{aligned} \tag{8}$$

where c_1 and c_2 represent constants. Since the additive uniform noise aligns the center of the $\tilde{\mathbf{y}}$ with $\mathbf{y}$, in (8), the first term is a constant and can be dropped, as can the last term. The second term quantifies the distortion between $\mathbf{x}$ and $\hat{\mathbf{x}}$, which can be replaced by MSE. The third term corresponds to R in (6).

The $p_{\hat{\mathbf{y}}}(\cdot)$ comprises a hyper prior module $E_z(\cdot)$ and a spatial prior module $E_s(\cdot)$. The $E_z(\cdot)$ consists of a hyper prior encoder $h_a(\cdot)$ and a hyper prior decoder $h_s(\cdot)$. The $h_a(\cdot)$ learns the hyper prior latent $\mathbf{z}$, which is then quantized to $\hat{\mathbf{z}} = Q(\mathbf{z})$ and entropy coded with a learned factorized prior. For this, the R in the loss function (5) during training is updated as:

$$R = R_y + R_z = \mathbb{E}_{\mathbf{x}\sim p_{\mathbf{x}}}\left[-\log_2 p_{\tilde{\mathbf{y}}|\tilde{\mathbf{z}}}(\tilde{\mathbf{y}}\mid\tilde{\mathbf{z}}) - \log_2 p_{\tilde{\mathbf{z}}}(\tilde{\mathbf{z}})\right]. \tag{9}$$

At the decoder side, we first use the $h_s(\cdot)$ to obtain the initial estimate parameters, which assists the $E_s(\cdot)$ in capturing the spatial dependencies. Under the joint influence of the $\hat{\mathbf{z}}$ and $g_a(\cdot)$, each element y_i in $\mathbf{y}$ is modeled as an independent gaussian random variable, thus the conditional probability of

$\tilde{\mathbf{y}}$ can be expressed as:

$$p_{\tilde{\mathbf{y}}}(\tilde{\mathbf{y}} \mid \hat{\mathbf{z}}) = \prod_i \left(\mathcal{N}(\mu_i, \sigma_i^2) * \mathcal{U}\left(-\frac{1}{2}, \frac{1}{2}\right) \right)(\tilde{y}_i), \tag{10}$$

where

$$\mu_i, \sigma_i = E_s\left(h_s\left(\hat{\mathbf{z}}; \theta_{h_s}\right), \tilde{y}_{<i}; \theta_{E_s}\right). \tag{11}$$

B. System Model of DeepSC

As for end-to-end wireless image transmission scenario, the DeepSC aims for low-latency transmission using minimal symbol sequences under channel constraints, facilitating efficient execution of downstream tasks [27]. Its architecture partially aligns with that of LIC. At the transmitter side, the neural analyzer $g_a(\cdot)$ extracts semantic features $\mathbf{y}$ from the source image $\mathbf{x}$. These features are subsequently mapped to complex-valued channel symbols $\mathbf{s} \in \mathbb{C}^l$ via a feature encoder $f_{tx}(\cdot)$:

$$\mathbf{s} = f_{tx}\Big(g_a(\mathbf{x}; \theta_{g_a}), p_{\hat{\mathbf{y}}}(g_a(\mathbf{x}; \theta_{g_a}); \theta_{p_{\hat{\mathbf{y}}}}); \theta_{f_{tx}}\Big). \tag{12}$$

At the receiver side, the feature decoder $f_{rx}(\cdot)$ reconstructs semantic features $\hat{\mathbf{y}}_{\mathbf{DeepSC}}$ from the received symbols $\hat{\mathbf{s}}$, which are then fed into the synthesizer $g_s(\cdot)$ to recover the reconstructed image $\hat{\mathbf{x}}_{\mathbf{DeepSC}}$. The proposed DeepSC system is illustrated in Fig. 1 (b). The dashed-gray-line procedures are only utilized during training to stabilize convergence. In this phase, arithmetic coding is omitted, and the quantization $Q(\cdot)$ is approximated by additive uniform noise to ensure differentiability.

The wireless channel is modeled by a transfer function $W(\mathbf{s}) = \mathbf{h} \odot \mathbf{s} + \mathbf{n}$, where $\mathbf{h} \in \mathbb{C}^l$ is the complex channel coefficient sequence and $\mathbf{n} \sim \mathcal{CN}(\mathbf{0}, \sigma^2 \mathbf{I}_l)$ is additive white gaussian noise (AWGN). The signal-to-noise ratio (SNR) is defined as:

$$\begin{aligned} \text{SNR} = 10\log_{10}\left(\frac{\mathbb{E}\left[\|\mathbf{h}\|_2^2\right] P}{\sigma^2}\right) \geq \\ 10\log_{10}\left(\frac{\mathbb{E}\left[\|\mathbf{h}\|_2^2\right] \mathbb{E}_{\mathbf{s}}\left[\|\mathbf{s}\|_2^2\right]}{\mathbb{E}_{\mathbf{n}}\left[\|\mathbf{n}\|_2^2\right]}\right), \end{aligned} \tag{13}$$

where the σ^2 denotes the average noise power and P represents the power constraint on the transmitted symbols $\mathbf{s}$, i.e., $\mathbb{E}_{\mathbf{s}}\left[\|\mathbf{s}\|_2^2\right] \leq P$. When the channel state information (CSI) $\hat{\mathbf{h}}$ is available, the receiver applies zero-forcing (ZF) equalization:

$$\hat{\mathbf{s}} \leftarrow \frac{\hat{\mathbf{h}}^*}{|\hat{\mathbf{h}}|^2} \odot W(\mathbf{s}), \tag{14}$$

where the $\hat{\mathbf{h}}^*$ is the complex conjugate of $\hat{\mathbf{h}}$. In practice, CSI is imperfect. We adopt an additive CSI error model $\hat{\mathbf{h}} = \mathbf{h} + \mathbf{e}$, where $\mathbf{e} \sim \mathcal{CN}\left(0, \sigma_e^2 \mathbf{I}_l\right)$. The estimation quality is measured by the normalized mean-squared error (NMSE) [28]:

$$\text{NMSE}_{\text{dB}} = 10\log_{10}\left(\frac{\mathbb{E}\left[\|\mathbf{e}\|_2^2\right]}{\mathbb{E}\left[\|\mathbf{h}\|_2^2\right]}\right). \tag{15}$$

We consider AWGN and fading channel scenarios. In the AWGN case, the channel coefficient $\mathbf{h}$ remains constant. For rayleigh fading, it is assumed that $\mathbf{h}$ varies independently for each transmission. In particular, $h_k \overset{\text{i.i.d.}}{\sim} \mathcal{CN}(0,1)$ for the k-th channel use. For correlated fading, $\mathbf{h}$ is generated through a first-order gauss–markov process: $h_k = \rho\, h_{k-1} + \alpha w_k$, where $w_k \sim \mathcal{CN}(0,1)$, $\alpha = \sqrt{1-\rho^2}$ and $\rho \in [0,1)$. The coefficient ρ controls the correlation level ($\mathbb{E}[h_k h_{k+\Delta}^*] = \rho^{|\Delta|}$) [29]. To capture burst errors, we let the noise variance follow a two-state Gilbert–Elliott markov process with transition probabilities p (good→bad) and q (bad→good), and in the bad state we additionally erase each symbol with probability ε. The instantaneous variance switches between σ^2 (good) and $\kappa\sigma^2$ (bad) [30]. Random deep fading due to intermittent blockages is modeled by a markov switch on the channel itself: line-of-sight (LOS) segments use a rician model $h_k = \mu + \sigma_s w_k$, where $w_k \sim \mathcal{CN}(0,1)$, $\mu = \sqrt{\frac{K}{K+1}}$ and $\sigma_s = \sqrt{\frac{1}{K+1}}$, while non-line-of-sight (NLOS) segments are rayleigh but attenuated in amplitude by $10^{-A_{\text{blk}}^{(\text{dB})}/20}$ and multiplied by a log-normal shadowing factor $10^{X_k/20}$, $X_k \sim \mathcal{N}(0, \sigma_{\text{sh}}^2)$. Segment durations are drawn from geometric distributions to produce piecewise-constant runs along time [31], [32].

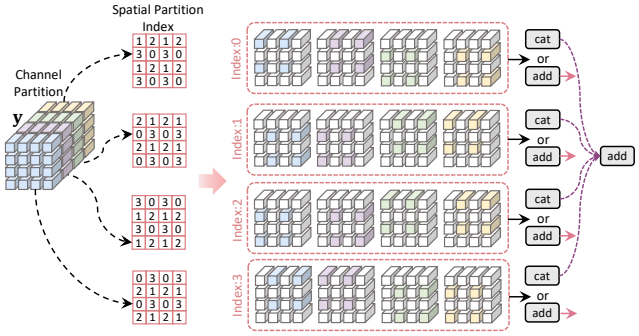


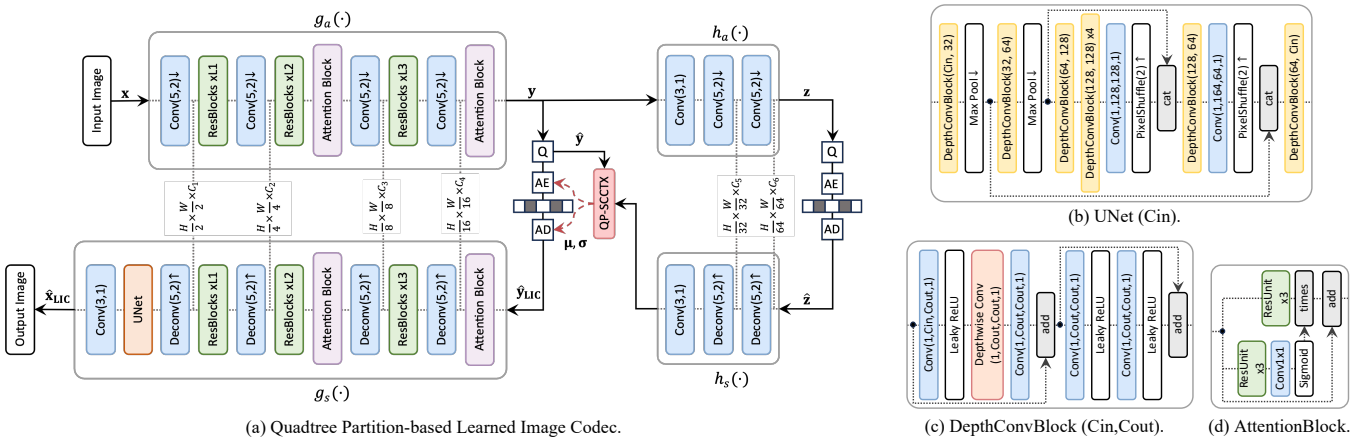
Fig. 2: The detailed procedure of the quadtree partition, the inverse process is called quadtree fusion.

Following [10], we denote the source bandwidth by $n = HWC$ and the channel bandwidth by l. The channel bandwidth ratio (CBR) is defined as CBR $\triangleq \frac{l}{n} = \frac{l}{HWC}$, measuring bandwidth efficiency as the number of channel uses per source pixel. The optimization of proposed DeepSC system is formulated as a RDO problem [13], analogous to (5), and can be expressed as:

$$L_{DeepSC} = \lambda \cdot \left[\mathcal{D}\left(\mathbf{x}, \hat{\mathbf{x}}_{\mathbf{LIC}}\right), +\mathcal{D}\left(\mathbf{x}, \hat{\mathbf{x}}_{\mathbf{DeepSC}}\right)\right] + K, \tag{16}$$

where λ is a weighting coefficient, $\mathcal{D}(\cdot)$ denotes the distortion metric, and K represents the rate term. The first distortion term, corresponding to the reconstructed image $\hat{\mathbf{x}}_{\mathbf{LIC}}$ from the LIC, serves to stabilize the training process, shown in the dashed-gray line in Fig. 1 (b). The second term measures the distortion of the DeepSC output. The rate term K quantifies the bandwidth cost and is estimated via LIC-based entropy modeling:

$$\begin{aligned} K = \eta \cdot \Big(\mathbb{E}_{\mathbf{x}\sim p_{\mathbf{x}}} \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}} \Big[-\log_2 p_{\mathbf{y}|\tilde{\mathbf{z}}}(\mathbf{y}|\tilde{\mathbf{z}}) \Big] + \mathbb{E}_{\mathbf{k}}\left[b(\mathbf{k})\right] \\ \underbrace{+ \mathbb{E}_{\mathbf{x}\sim p_{\mathbf{x}}} \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}} \Big[-\log_2 p_{\tilde{\mathbf{z}}}(\tilde{\mathbf{z}}) \Big]}_{\text{not actually transmitted}} \Big), \end{aligned} \tag{17}$$



(a) Quadtree Partition-based Learned Image Codec. (b) UNet (Cin). (c) DepthConvBlock (Cin,Cout). (d) AttentionBlock.

Fig. 3: (a) Overview of proposed quadtree partition-based learned image codec (Quad-LIC). AE, AD, and Q denote the arithmetic encoding, arithmetic decoding, and quantization, respectively. QP-SCCTX is the proposed quadtree partition-based space-channel context model, illustrated in Fig. 5. (b) The network structure of the UNet. (c) The details of DepthConvBlock. (d) The details of AttentionBlock.

where η is the scaling hyperparameter that controls the spectral efficiency of transmission and $b(\mathbf{k})$ denotes the number of bits of the side information $\mathbf{k}$ after lossless compression. The $\mathbf{k}$ assists f_{rx} in reconstructing the dimension corresponding to each symbol:

$$\hat{\mathbf{x}}_{\mathbf{DeepSC}} = g_s\Big(f_{rx}(\hat{\mathbf{s}}, \mathbf{k}; \theta_{f_{rx}}); \theta_{g_s}\Big). \tag{18}$$

In (17), the second item is applied only during the testing phase, as it does not contribute gradients in training, while the final term is included solely to stabilize training and is not transmitted.

The semantic feature $\mathbf{y} \in \mathbb{R}^{H_y \times W_y \times C_y}$ consists of $H_y \times W_y$ spatial units $\mathbf{y}_{sp}^{(i,j)}$, each represented by a C_y-dimensional vector and corresponds to a symbol $\mathbf{s}_{i,j} \in \mathbb{C}^{\lceil k_{i,j}/2 \rceil}$, where the symbol length factor $k_{i,j}$ is computed as the entropy accumulation of $\mathbf{y}_{sp}^{(i,j)}$:

$$k_{i,j} = Q(\sum_{k=1}^{C_y} -\eta \log_2 p_{y_{i,j,k}|\tilde{\mathbf{z}}}(y_{i,j,k}|\tilde{\mathbf{z}})). \tag{19}$$

In practice, $k_{i,j}$ is quantized to the nearest value within a predefined discrete range and contributes to the overall side information vector $\mathbf{k} \in \mathbb{R}^{H_y \times W_y}$. This process is referred to as rate matching. The selection of this range balances the trade-off among search complexity, side information rate $\eta \cdot b(\mathbf{k})$, and reconstruction quality.

III. Architecture and Implementations

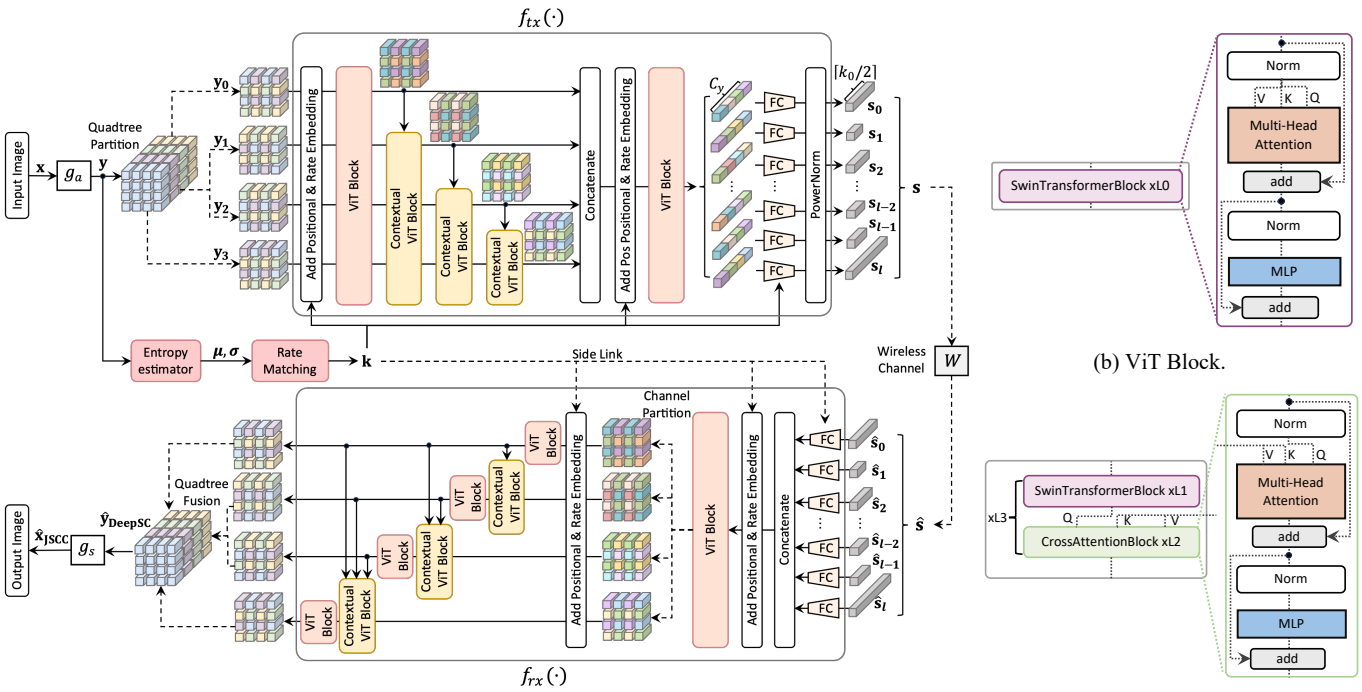
This section presents the architecture and implementation details of the proposed Quad-DeepSC. First, the integrated LIC-DeepSC strategy, which forms the foundation of the design, is introduced. Then, the model architecture of Quad-LIC and Quad-DeepSC are described. The quadtree partition-based coding scheme, which improves image transmission efficiency, is then detailed. Subsequently, the variable symbol length mapping technique and the transmission of associated side information are discussed. Finally, the training strategy used to optimize the entire framework is outlined.

A. Integrated LIC-DeepSC Strategy

The DeepSC system, built on LIC, can achieve excellent transmission performance. As illustrated in Fig. 1, the neural analyzer $g_a(\cdot)$ and neural synthesizer $g_s(\cdot)$ in LIC learn an effective transformation between images and semantic features. The entropy estimator $p_{\hat{\mathbf{y}}}(\cdot)$ provides accurate probabilistic modeling, enabling efficient source compression. Based on the efficient LIC framework, DeepSC introduces a feature encoder $f_{tx}(\cdot)$ and a feature decoder $f_{rx}(\cdot)$ to learn a robust mapping between features and transmitted symbols, while leveraging $p_{\hat{\mathbf{y}}}(\cdot)$ for reliable symbol-length estimation. This joint design allows efficient conversion between images and variable-length symbol sequences.

However, this does not imply that all LIC are universally applicable. Although some LICs exhibited efficient source compression, their computational complexity restrict their applicability in real-time image transmission. Furthermore, when designing the DeepSC system based on LIC, both the $f_{tx}(\cdot)$ and $f_{rx}(\cdot)$ need to align with the $p_{\hat{\mathbf{y}}}(\cdot)$. Specifically, the reference content at each step of $p_{\hat{\mathbf{y}}}(\cdot)$ needs to correspond to that in $f_{tx}(\cdot)$ and $f_{rx}(\cdot)$. This underscores that LICs based on autoregressive entropy models involve numerous steps within their entropy estimation, making them suboptimal for this specific application. A mismatch between the symbol-length estimation and symbol-mapping processes can lead to performance degradation, as confirmed in Section IV-C.

To enhance the design of the DeepSC system by leveraging advancements in LIC, we propose an integrated strategy. First, select a feature partitioning methods. In this paper, we adopt the quadtree partition [6], [21], as illustrated in Fig. 2. This partitioning method divides the feature $\mathbf{y}$ into four parts along the channel dimension, with each part segmented step-by-step according to its corresponding spatial partition index. For instance, in step 0, we extract elements with index 0 from each part and then combine them through concatenation or addition to form the partitioned feature $\mathbf{y}_0$, where concatenation is employed for entropy estimation, as it requires maintaining



(a) Quadtree Partition-based Image DeepSC System.

(b) ViT Block.

(c) Contextual ViT Block.

Fig. 4: (a) Overview of proposed quadtree partition-based image DeepSC system (Quad-DeepSC). The side information $\mathbf{k}$ is derived from the estimated parameters $(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ and transmitted via the side link to facilitate feature decoding, including symbol dimension mapping, positional and rate embedding selection, as detailed in Section III-D. (b) The network structure of the vision transformer (ViT) Block. (c) The network structure of contextual ViT Block. The module with the same name as in Fig. 3 represents the identical component.

spatially-related positions, whereas addition is used for feature coding. The advantage of this approach lies in its reduced step count and its ability to effectively guide $f_{tx}(\cdot)$ and $p_{\hat{\mathbf{y}}}(\cdot)$ in removing spatial correlation from each element of $\mathbf{y}$ while maintaining parallelism. Next, we design the feature codec and entropy model, where the input at each step i corresponds to the partitioned feature $\mathbf{y}_i$, and the $\mathbf{y}_j$ from a previous step j $(j < i)$ serves as a reference, enabling mutual adaptation. Finally, we develop a lightweight, fast-inference, high-performance neural analyzer $g_a(\cdot)$ and neural synthesizer $g_s(\cdot)$. The training methodology for this model will be presented in the following Section III-E.

B. Model Architecture

The detailed design of the proposed Quad-LIC is illustrated in Fig. 3(a). Notably, the Quad-LIC is implemented based on the ELIC framework [5]. In contrast to ELIC, the proposed approach discards its renowned entropy model and instead leverages a quadtree partition to construct a lightweight model, termed the quadtree partition-based space-channel context model (QP-SCCTX) that performs entropy estimation in merely four steps, thereby enabling fast coding speed. Despite its tendency to increase coding latency, the AttentionBlock in ELIC is retained to enhance the performance of the neural analyzer $g_a(\cdot)$ and neural synthesizer $g_s(\cdot)$. The corresponding structure is shown in Fig. 3(d). Additionally, a UNet, as illustrated in Fig. 3(b), is incorporated at the tail of $g_s(\cdot)$ to refine the reconstructed features. This enhancement improves the quality of source decoding and enhances robustness to channel impairments when the pre-trained Quad-LIC is embedded within the DeepSC system. For constructing both the QP-SCCTX and the UNet in Quad-LIC, we adopt techniques from DCVC-DC [6], utilizing the DepthConvBlock illustrated in Fig. 3(c) as the basic block. In the the DepthConvBlock, except for the deep convolutional layers employing a 3×3 kernel, all regular convolutional layers utilize a 1×1 kernel to minimize computational complexity.

Building upon Quad-LIC, the proposed Quad-DeepSC introduces a feature encoder $f_{tx}(\cdot)$, a feature decoder $f_{rx}(\cdot)$, and a channel transfer function $W(\cdot)$. As shown in Fig. 4, we built the vision transformer (ViT) block as the basic block based on Swin-Transformer [33], owing to its demonstrated efficacy in generating noise-resilient channel symbol vectors $\mathbf{s}$ [19]. Furthermore, the relative position and rate configuration information can be effectively conveyed to the ViT Block by adding embedding to its inputs. To ensure consistency with entropy estimation, a cross-attention mechanism is introduced to construct the contextual ViT block [14], which models contextual dependencies among DeepSC output codewords. Its structure is detailed in Fig. 4(c). The transformation of symbol dimensions is achieved using fully connected (FC) layers. The output symbols from the FC layers are power-normalized to satisfy the power constraint prior to wireless channel transmission.

C. Quadtree Partition-Based Coding

We propose a quadtree-partitioned coding scheme that jointly designs an entropy estimator and feature coding. The entropy estimator comprises a hyper prior encoder $h_a(\cdot)$, a hyper prior decoder $h_s(\cdot)$ and QP-SCCTX $E_s(\cdot)$. The detailed architecture of proposed QP-SCCTX is illustrated in Fig. 5. Compared with the two-step checkerboard entropy model [15], QP-SCCTX provides finer granularity and more effectively exploits spatial contexts. Along the channel dimension, the semantic feature $\mathbf{y}$ is divided into four groups. Each element within a group is assigned an integer from 0 to 3 based on the spatial partition index defined by quadtree partition, which determines the sequential order of entropy estimation. Accordingly, at the $i_{>0}$th step, the context for estimating the current feature $\mathbf{y}^{(i)}$ is constructed from the previously estimated features $\hat{\mathbf{y}}_{ctx}^{(i)}$, thereby capturing both spatial and channel-wise dependencies:

$$(\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i^2) = g_{sp}^{(i)}\left(\hat{\mathbf{y}}_{ctx}^{(i)}, (\boldsymbol{\mu}_0, \boldsymbol{\sigma}_0^2); \theta_{g_{sp}^{(i)}}\right), \tag{20}$$

where

$$(\boldsymbol{\mu}_0, \boldsymbol{\sigma}_0^2) = h_s(\hat{\mathbf{z}}; \theta_{h_s}). \tag{21}$$

Quantization is subsequently performed using the predicted entropy parameters:

$$\hat{\mathbf{y}}^{(i)} = Q(\mathbf{y}^{(i)} - \boldsymbol{\mu}_i) + \boldsymbol{\mu}_i. \tag{22}$$

Notably, spatial positions associated with index i are non-overlapping within each group. Hence, at each step, one-quarter of the channels are estimated at each spatial location, and the previously estimated elements are concatenated to form the context for the next step. Across four steps, each element is progressively estimated with 0, 4, 4, and 8 neighboring references, respectively. In contrast, the checkerboard model [15] uses 0 and 4 neighbors in the two steps. On average, QP-SCCTX leverages twice as many neighboring references, thereby enhancing inter-channel correlation modeling and enabling more accurate conditional probability estimation.

The semantic feature $\mathbf{y}$ contains redundancy along both the spatial and channel axes. To address this, we introduce quadtree partition-based feature coding scheme, as illustrated in Fig. 4(a). The estimation results of QP-SCCTX effectively guide feature coding to eliminate these redundancies while linking the output codewords of DeepSC to minimize CBR. $\mathbf{y}$ is partitioned into four groups: $[\mathbf{y_0}, \mathbf{y_1}, \mathbf{y_2}, \mathbf{y_3}]$ through quadtree partition. The probability distribution parameters of $\mathbf{y}_i$ is characterized by $(\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i^2)$, with its coding referencing the coding result of $\mathbf{y}_{<i}$ as context. This context corresponds to the reference $\hat{\mathbf{y}}_{ctx}^{(i)}$ for estimating $(\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i^2)$, allowing the feature coding to match the QP-SCCTX and exploit more granular and diverse contexts, thereby maximizing the utilization of correlations in both the spatial and channel dimensions. Furthermore, to provide $f_{tx}(\cdot)$ and $f_{rx}(\cdot)$ with position and probability distribution estimates related to $\mathbf{y}_i$, we compute the corresponding symbol length factor vector $\mathbf{k}_i$ for each element of $\mathbf{y}_i$, based on $(\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i^2)$. This $\mathbf{k}_i$ is then used to select the positional and rate embedding to be added to the inputs of $f_{tx}(\cdot)$ and $f_{rx}(\cdot)$.

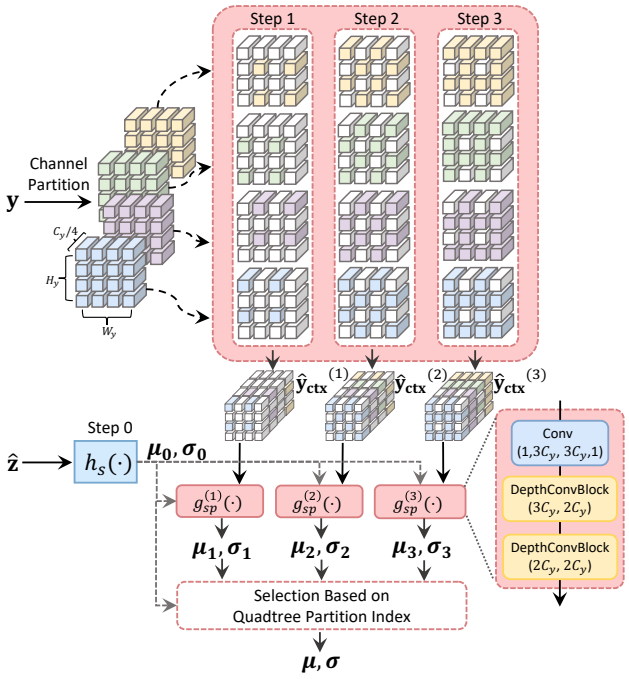


Fig. 5: Diagram of the quadtree partition-based space-channel context model (QP-SCCTX).

D. Variable Symbol Length Mapping

Quad-LIC leverages arithmetic coding to inherently perform variable-length coding according to the estimated probability distribution. However, gradients are difficult to propagate through arithmetic codecs, and both CNN and Transformer architectures enforce fixed input-output channel dimensions, which limits the DeepSC systems from flexibly generating variable-length symbol sequences. A more computationally efficient approach is to utilize a set of FC layers to realize symbol mappings with variable lengths. Nonetheless, this introduces overhead in conveying to the receiver which FC layer was used for each symbol to restore dimensionality, thus incurring side information transmission costs. As shown in Fig. 3(a), the semantic feature $\mathbf{y} \in \mathbb{R}^{H_y \times W_y \times C_y}$, has a fixed channel dimension C_y, while the spatial dimensions $H_y = H/16$ and $W_y = W/16$ vary with input resolution (H, W). Therefore, we choose the spatial unit $\mathbf{y}_{sp}^{(i,j)} \in \mathbb{R}^{C_y}$ as the fundamental entity for symbol mapping. On the one hand, the C_y elements in each unit can interact during the mapping process. On the other hand, only $H_y \times W_y$ FC-layer indices need to be transmitted, reducing the overhead of side information.

In practice, we compute the length factor $k_{i,j}$ in (19) for each corresponding symbol $\mathbf{s}_{i,j}$ by summing the entropy of the C_y elements in $\mathbf{y}_{sp}^{(i,j)}$. To map $\mathbf{y}_{sp}^{(i,j)}$ from C_y to $k_{i,j}$ dimensions, we use the shared FC layer $\mathrm{fc}_{k_{i,j}}(\cdot)$ with weight shape $C_y \times C_y$, followed by a binary mask $\mathbf{m}_{k_{i,j}}$ to perform dimensionality reduction:

$$\mathbf{r}_{i,j} = \mathrm{fc}_{k_{i,j}}(\mathbf{y}_{sp}^{(i,j)}; \theta_{\mathrm{fc}_{k_{i,j}}}) \odot \mathbf{m}_{k_{i,j}}, \tag{23}$$

where

$$\mathbf{m}_{k_{i,j}} = [\underbrace{1, ..., 1}_{k_{i,j}}, 0, ..., 0] \in \mathbb{R}^{C_y}. \tag{24}$$

The elements in $\mathbf{r}_{i,j}$ are paired to form the complex symbol $\mathbf{s}_{i,j}$. Similarly, $k_{i,j}$ is also used to select the positional and rate embedding. Consequently, $k_{i,j}$ constitutes a side information tensor $\mathbf{k} \in \mathbb{R}^{H_y \times W_y}$, whose transmission cost is given by:

$$\text{CBR}_{\text{side info}} = \frac{b(\mathbf{k})/C_{\mathbf{k}}}{H \times W \times 3} = \frac{H_y \times W_y \times \lceil \log_2(C_y) \rceil}{H \times W \times 3 \times C_{\mathbf{k}}}, \tag{25}$$

where $C_{\mathbf{k}}$ denotes the spectral efficiency of the side link to transmit $\mathbf{k}$. There is potential to optimize the side information cost in (25). Notably, $\mathbf{k}$ correlates with the semantic content and can be represented as a rate map on the $H_y \times W_y$ grid. By encoding this map using lossless portable network graphics (PNG), we can further reduce $b(\mathbf{k})$. Additionally, we restrict the range of $k_{i,j}$ using a quantization function $Q_k(\cdot)$ that maps each $k_{i,j}$ to its nearest value in a rate predefined set $[k_0, \ldots, k_{n-1}]$, and transmit only the corresponding indices:

$$\text{CBR}_{\text{side info}} = \frac{b\Big(\text{PNG}(Q_k(\mathbf{k}))\Big)}{H \times W \times 3 \times C_{\mathbf{k}}}, \tag{26}$$

where $Q_k(\cdot) \in [0, .., n-1]$ (e.g., $Q_k(k_i) = i$). This not only reduces $b\Big(\text{PNG}(Q_k(\mathbf{k}))\Big)$, but also limits the search space for FC layers and positional and rate embedding, thereby reducing computational complexity. The rate matching module in Fig. 4 (a) corresponds to the computation of $Q_k(\mathbf{k})$ based on the parameters estimated by the entropy estimator.

E. Training Methodology

The training procedure of Quad-DeepSC is presented in Algorithm 1. Owing to the complexity of multiple distinct modules, direct end-to-end training may impede convergence. Therefore, the training for Quad-DeepSC is divided into four stages. Initially, the parameters $\theta_{f_{tx}}$ and $\theta_{f_{rx}}$ related to feature coding are frozen, and the Quad-LIC-related modules are optimized for source compression. The Quad-LIC training is divided into two stages. In the first stage, additive uniform noise is used to replace quantization and accelerate model convergence. In the second stage, actual quantization is applied with training conducted using the straight-through estimator (STE) [34]. After these two stages, the Quad-LIC performance is evaluated. The performance of Quad-LIC is considered indicative of the overall DeepSC system performance. If Quad-LIC demonstrates slight size, fast inference and satisfactory performance, it implies that the designs of $g_s(\cdot)$, $g_a(\cdot)$, $E_a(\cdot)$, and $E_s(\cdot)$ are well-structured. Otherwise, additional ablation studies are required to eliminate redundant or ineffective components. Subsequently, we unfreeze $\theta_{f_{tx}}$ and $\theta_{f_{rx}}$, integrate the pre-trained Quad-LIC from the second stage into Quad-DeepSC, and conduct end-to-end training in two additional stages. Continuous quantization relaxation via additive uniform noise is preferable to discrete rounding for DeepSC: it preserves differentiability and yields smoother, more stable gradients over the stochastic channel, improving end-to-end optimization [24], [25]. Accordingly, we use the continuous surrogate during DeepSC training and reserve STE only for LIC validation. In the third stage, the quantization function $Q_k(\cdot)$ allows unrestricted selection of $k_{i,j}$ over the initial rate set $[0, \ldots, C_y]$, with $C_y + 1$ FC layers and positional and rate embedding. In the fourth stage, the FC layers and embedding are distilled, restricting $k_{i,j}$ to a reduced rate set $[k_0, \ldots, k_{n-1}]$. This reduces the side information cost,

Algorithm 1 Training the Quad-DeepSC

Input:
Training data $\mathbf{x}$, training steps for four stages N_1, N_2, N_3, and N_4, scaling hyperparameter η, lagrange multiplier λ, learning rate l_r.
Output:
Parameters $(\theta^*_{g_a}, \theta^*_{g_s}, \theta^*_{h_a}, \theta^*_{h_s}, \theta^*_{g_{sp}}, \theta^*_{f_{tx}}, \theta^*_{f_{rx}})$
Stage 1: Train the Learned Image Codec
Randomly initialize LIC parameters and fix feature encoder and decoder parameters, $(\theta^*_{f_{tx}}, \theta^*_{f_{rx}})$.
Approximate quantization with additive uniform noise.
for $i \leftarrow 1$ **to** N_1 **do**
  Sample $\mathbf{x} \sim p_{\mathbf{x}}$
  Compute the loss function (5): $L_{LIC} = \lambda \cdot \mathcal{D}(\mathbf{x}, \hat{\mathbf{x}}_{\mathbf{LIC}}) + R$
  Update the parameters: $(\theta^*_{g_a}, \theta^*_{g_s}, \theta^*_{h_a}, \theta^*_{h_s}, \theta^*_{g_{sp}})$
end for

Stage 2: Finetune the Learned Image Codec
Load parameters from Stage 1 and fix the feature encoder and decoder parameters.
Apply actual quantization with gradient copying (STE).
Lower the learning rate l_r.
for $i \leftarrow 1$ **to** N_2 **do**
  Repeat steps 5 to 7 from Stage 1.
end for

Stage 3: Train the DeepSC System
Load LIC parameters from Stage 2.
Approximate quantization with additive uniform noise.
Apply quantization function $Q_k(\cdot)$ with a initial rate predefined set $[0, 1, ..., C_y]$.
for $i \leftarrow 1$ **to** N_3 **do**
  Sample $\mathbf{x} \sim p_{\mathbf{x}}$
  Compute the loss function (16): $L = \lambda \cdot [\mathcal{D}(\mathbf{x}, \hat{\mathbf{x}}_{\mathbf{LIC}}) + \mathcal{D}(\mathbf{x}, \hat{\mathbf{x}}_{\mathbf{DeepSC}})] + K$
  Update the parameters: $(\theta^*_{g_a}, \theta^*_{g_s}, \theta^*_{h_a}, \theta^*_{h_s}, \theta^*_{g_{sp}}, \theta^*_{f_{tx}}, \theta^*_{f_{rx}})$
end for

Stage 4: Finetune the DeepSC System
Load parameters from Stage 3.
Distill the rate predefined set $[0, 1, ..., C_y]$ to $[k_0, ..., k_{n-1}]$.
Lower the learning rate l_r.
for $i \leftarrow 1$ **to** N_4 **do**
  Repeat steps 17 to 19 from Stage 3.
end for

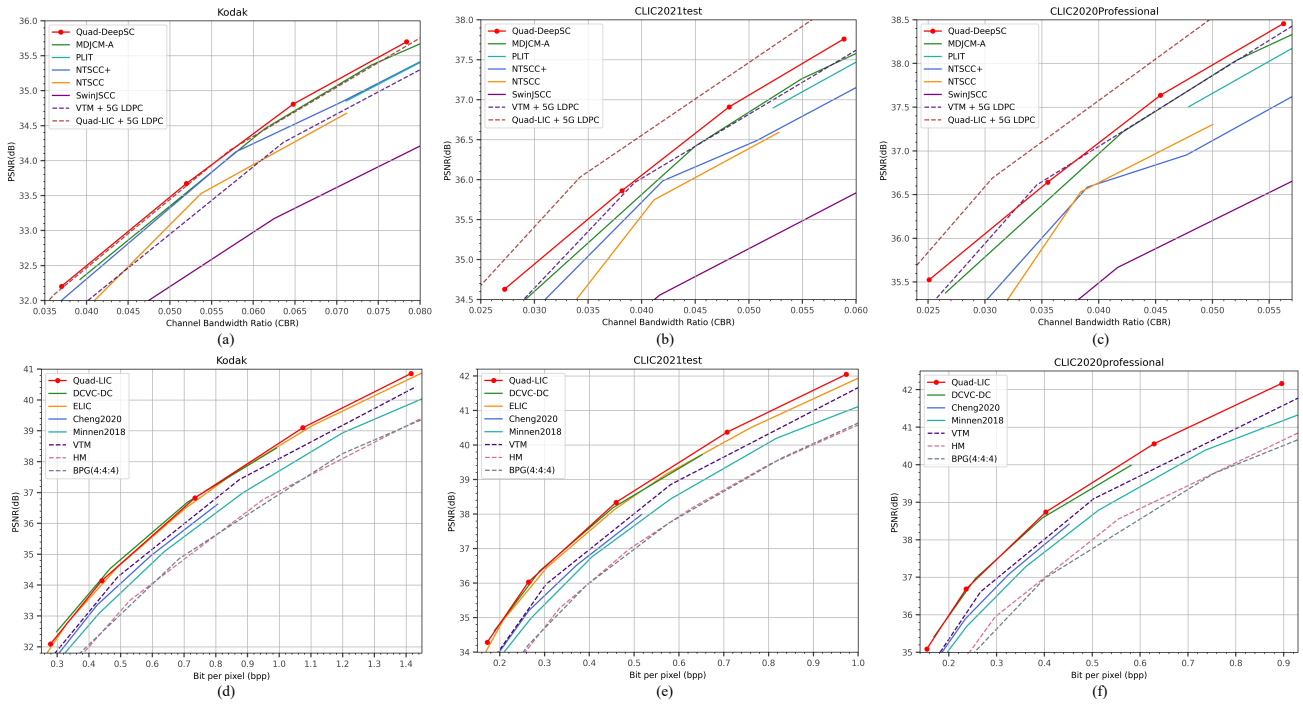


Fig. 6: (a)-(c) PSNR performance of Quad-DeepSC for image transmission over the AWGN channel at SNR = 10 dB. (d)-(f) PSNR performance of Quad-LIC for image compression. All models are optimized to minimize MSE.

computational space complexity, and the search complexity. Furthermore, this distillation enables the selection of an optimal rate set tailored to each CBR range.

Notably, arithmetic codecs are only used during Quad-LIC evaluation. During training, the quantized semantic features $\hat{\mathbf{y}}_{\mathbf{LIC}}$ are treated as arithmetic-decoding features. For training the Quad-DeepSC, additive uniform noise replaces quantization to alleviate the RDO problem. Additionally, channel impairments and the small batch size introduces gradient instability during training. To stabilize convergence, $\hat{\mathbf{y}}_{\mathbf{LIC}}$ is directly passed to $g_s(\cdot)$, forming $\mathcal{D}(\mathbf{x}, \hat{\mathbf{x}}_{\mathbf{LIC}})$ in the loss function (16), thereby anchoring entropy modeling and feature representation in Quad-LIC-related modules.

IV. Experimental Results

This section evaluates the performance of the proposed Quad-LIC and Quad-DeepSC through numerical experiments. We begin by detailing the experimental setup, followed by a presentation of the results using both visualizations and quantitative metrics.

A. Experimental Settings

a) Training Details: For training the Quad-LIC, we use the largest 8000 images picked from ImageNet [35] dataset, after applying a noising-downsampling preprocessing [5], [15], [26]. For each architecture, models are trained with various λ values for different quality presets. We use $\lambda = [6, 13, 35, 80, 150] \times 10^{-3}$ for each MSE-optimized model and $\lambda = [13, 35, 80, 150] \times 10^{-3}$ for each MS-SSIM-optimized model. The number of channels is set to $C_y = 320$ and $C_z = 192$ for all models. The Adam optimizer is employed with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and images are randomly cropped to 256×256 patches. The model is initially trained with $\lambda = 0.013$ to obtain an anchor model. The initial learning rate is set to $1e^{-4}$, with a batch size of 16, and uniform noise is sampled to estimate $\hat{\mathbf{y}}$ for the QP-SCCTX and synthesizer $g_s(\cdot)$. The anchor model is trained for 3800 epochs, followed by a learning rate decay to $1e^{-5}$ for an additional 200 epochs with $\hat{\mathbf{y}} = \text{STE}(\mathbf{y} - \boldsymbol{\mu}) + \boldsymbol{\mu}$ for the QP-SCCTX and $g_s(\cdot)$. The anchor model is then fine-tuned using other λ values, with a learning rate of $1e^{-5}$ and trained for 500 epochs. For training the Quad-DeepSC, since ImageNet is in JPEG format (a lossy version of the original images), we use the DIV2K [36] dataset, containing 800 high-resolution images. The trained Quad-LIC parameters are then used to initialize the Quad-DeepSC. The initial rate is set to $[0, 1, ..., C_y]$ and $\lambda = [40, 120, 240, 450] \times 10^{-3}$ for PSNR optimization, and $\lambda = [50, 240, 800, 2000] \times 10^{-3}$ for MS-SSIM optimization. The same optimizer is used, with a batch size of 8 and an initial learning rate of $1e^{-4}$. Each model is trained for 2000 epochs, followed by a learning rate decay to $1e^{-5}$ for another 1000 epochs.The rate set is then distilled to $[k_0, k_1, ..., k_{n-1}]$ and the model is fine-tuned with the same learning rate and batch size for 500 epochs. All experiments are conducted using PyTorch 2.4.1, Intel(R) Core(TM) i7-14700K @ 3.40GHz, and NVIDIA GeForce RTX 4090 GPU.

b) Evaluation Settings: The performance of the proposed Quad-LIC and Quad-DeepSC models is evaluated based on PSNR and MS-SSIM. The test datasets include various resolutions. The Kodak [37] dataset, containing 24 images with resolutions of 512×768 or 768×512, the CLIC2020 profes-

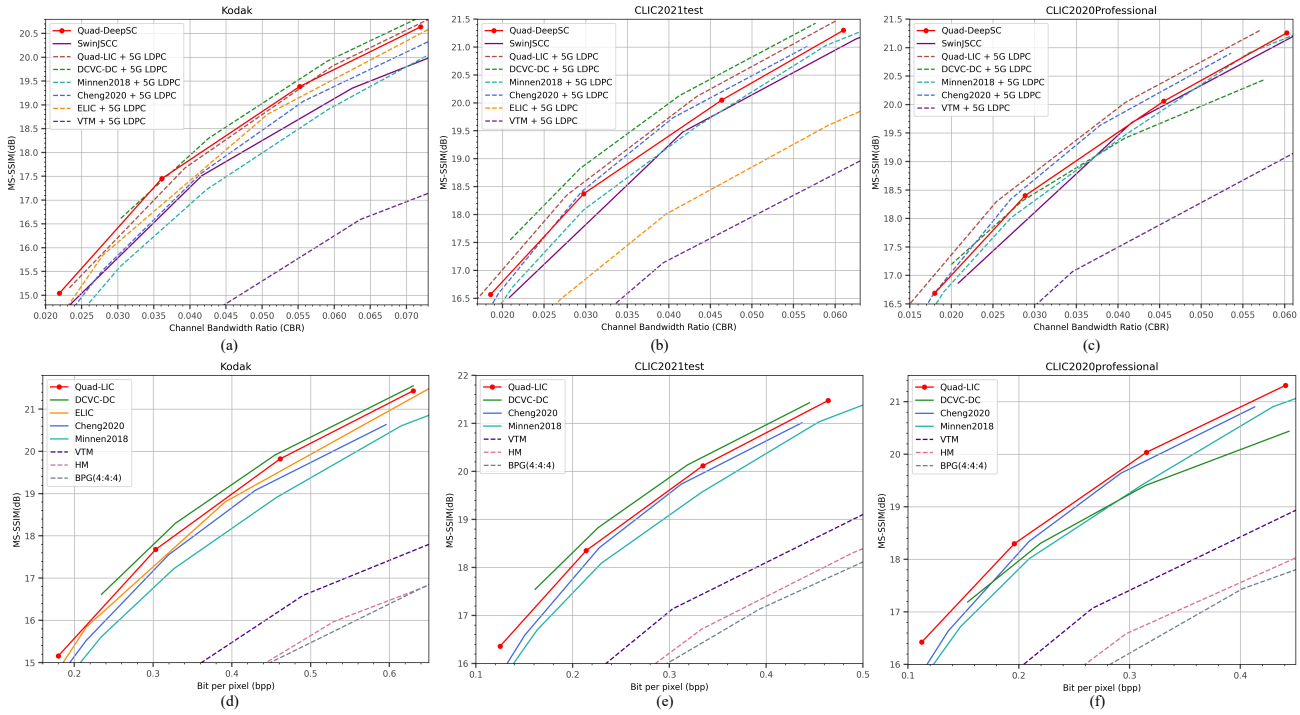


Fig. 7: (a)-(c) MS-SSIM performance of Quad-DeepSC for image transmission over the AWGN channel at SNR = 10 dB. (d)-(f) MS-SSIM performance of Quad-LIC for image compression. All models are optimized to maximize MS-SSIM.

sional testset [38], which includes 250 images with resolutions up to 2048×1365, and the CLIC2021 testset [39], which consists of 60 images with resolutions up to 2048×1890. All images are padded to multiples of 64 for evaluation. For benchmarks, we use the traditional image codecs such as BPG [40], HEVC HM (still image codec) [41], and VVC VTM (still image codec) [16], with the same test settings used in compressai [42]. The emerging LICs include Minnen2018 [20], Cheng2020 [43], ELIC [5], and DCVC-DC (still image codec) [6]. The end-to-end image DeepSC systems include NTSCC [13], NTSCC+ [14], MDJCM [18], SwinJSCC [44], and PLIT [19]. We compare the traditional image codecs and LICs with 5G LDPC channel coding [45], with a length of 6144 for error correction and digital modulation, following the 3GPP standard. The optimal modulation order and channel code rate are chosen based on the MCS Table 5.1.3.1-1 for the PDSCH in [17]. For each channel condition, we sweep the MCS index and adopt the setting that produces the lowest CBR. if decoding fails and the image cannot be reconstructed, we allow up to nine retransmissions. The metrics are computed on the first successful reconstruction. After testing all options, we report the results for the best configuration. Simulations are conducted using Sionna [46]. Source and channel coding schemes are concatenated with the "+" symbol. We losslessly compress $Q_k(\mathbf{k}) \in \mathbb{R}^{H_y \times W_y}$ with PNG. To ensure a fair comparison across DeepSC systems, we transmit the side information using the same scheme as the traditional baselines and report the resulting transmission cost.

B. Results Analysis

a) PSNR and MS-SSIM Performance: Fig. 6 illustrates the PSNR performance of Quad-LIC and Quad-DeepSC. Specifically, Fig. 6 (a)-(c) compare Quad-DeepSC with various transmission schemes under AWGN channel conditions at an SNR of 10 dB, while Fig. 6 (d)-(f) depict the performance comparison between Quad-LIC and several existing image compression methods. The experimental results indicate that the proposed Quad-DeepSC significantly outperforms current DeepSC systems. Furthermore, the corresponding Quad-LIC achieves competitive results relative to existing LIC methods and significantly outperforms traditional image compression schemes.

To the best of our knowledge, this study is the first to propose an end-to-end transmission framework that surpasses the "VTM + 5G LDPC" benchmark on the CLIC2021 testset for 2K-resolution images, attaining a 2.53% bandwidth reduction while maintaining comparable PSNR reconstruction fidelity. On the Kodak dataset, comprising medium-resolution images, Quad-DeepSC consistently surpasses the "VTM + 5G LDPC" benchmark across all considered CBR levels. Specifically, it achieves an 8.8% reduction in bandwidth while preserving comparable reconstructed PSNR at the receiver. Quad-DeepSC also outperforms "Quad-LIC + 5G LDPC" scheme, highlighting the promise of DeepSC. However, on the CLIC2020 Professional testset, which contains many images above 2K resolution, all DeepSC methods underperform relative to "Quad-LIC + 5G LDPC" scheme. In that setting, Quad-DeepSC achieves a 0.8% bandwidth reduction compared to the "VTM + 5G LDPC" scheme.

TABLE I: Rate-distortion (R-D) performance of various LICs on the CLIC2021 test dataset and computational complexity on the Kodak dataset, evaluated on an NVIDIA RTX 4090 GPU. Lower BD-Rate values indicate better R-D performance. The arithmetic codecs were run using two official GitHub repositories: one from [42] and the other from [6]. The anchor is VTM. The best result is shown in **bold**.

LICs	Arithmetic Codec	BD-Rate (%)	Enc time (ms)	Dec time (ms)	MACs (G)	Params (M)
VTM	-	0.0	-	-	-	-
DCVC-DC [6]	Cpp Rans Codec [6]	-11.8	15.7	22.0	217.34	31.00
Quad-LIC (proposed)		**-12.2**	**15.3**	**8.4**	401.91	31.13
Minnen2018 [20]	Compressai Rans Codec [42]	10.2	1057.1	2343.2	**176.85**	**25.5**
Cheng2020 [43]		5.0	1077.2	2343.4	404.05	29.63
ELIC [5]		-9.3	95.2	59.1	327.88	33.79
Quad-LIC (proposed)		-12.2	40.5	36.8	401.91	31.13

TABLE II: Transmission performance of various DeepSC systems on the CLIC2021 test dataset and computational complexity on the Kodak dataset, evaluated on an NVIDIA RTX 4090 GPU. Lower BD-CBR values indicate better transmission performance. The anchor is "VTM + 5G LPDC" over the AWGN channel at SNR = 10 dB. The best result is shown in **bold**.

DeepSC Systems	BD-CBR (%)	Enc time (ms)	Dec time (ms)	MACs (G)	Params (M)
VTM + 5G LPDC	0.0	-	-	-	-
SwinJSCC [44]	34.8	22.2	**3.0**	**203.05**	**33.07**
PLIT [19]	5.7	98.6	4.7	208.20	113.47
NTSCC [13]	17.9	20.3	3.3	262.80	31.36
NTSCC+ [14]	7.0	29.2	7.9	326.00	58.59
MDJCM-A [18]	2.9	20.8	17.4	412.91	32.93
Quad-DeepSC (proposed)	**-2.5**	**12.1**	16.2	422.64	50.58

For better visualization, the MS-SSIM is converted into dB using the following formula: $-10\log_{10}(1-\frac{\sum_{i=1}^{N}\text{MS-SSIM}_i}{N})$. Fig. 7 displays the MS-SSIM performance of Quad-LIC and Quad-DeepSC, where Fig. 7 (a)-(c) illustrate the comparison of various transmission schemes under the AWGN channel with 10 dB SNR, and Fig. 7 (d)-(f) present the performance of different image compression techniques. The overall trend aligns with that observed in Fig. 6. Our Quad-LIC shows competitive performance compared to existing LIC schemes, significantly outperforming traditional image compression methods, while Quad-DeepSC exhibits competitive performance compared to existing "LICs + 5G LDPC" scheme. On the Kodak dataset, Quad-DeepSC achieves superior transmission performance over the "Quad-LIC + 5G LDPC" scheme, saving 3.3% of bandwidth while maintaining the same MS-SSIM quality at the receiver.

b) Computational Complexity: Table I summarizes the Bjøntegaard Delta Rate (BD-Rate. he average bitrate savings between two schemes over a range of qualities, measured at equivalent distortion. Lower is better) [47] results for PSNR, coding speed, and computational complexity of Quad-LIC, in comparison with existing LICs. Results for traditional image compression methods are excluded, as they are generally not optimized for GPU execution. For R-D evaluation, VTM is adopted as the baseline and tested on the 2K-resolution CLIC2021 dataset. Quad-LIC achieves an 12.2% bitrate reduction without compromising PSNR quality, demonstrating its superior performance. The coding speed and computational complexity are evaluated on the Kodak dataset. When employing the arithmetic codec in Compressai (v1.1.5) [42], the encoding and decoding delays, along with computational space complexity during inference, are all lower than those of ELIC, achieving stronger performance owing to its lightweight entropy estimator and enhanced neural synthesizer with the added UNet module. Moreover, under the same arithmetic codec settings as DCVC-DC, the encoding delay of our Quad-LIC is reduced to 15.3 ms, and the decoding delay to 8.4 ms, significantly outperforming DCVC-DC in coding speed, while preserving better PSNR performance. This improvement stems from a more compact entropy estimator and a stronger feature transform network. Notably, by eliminating the computationally intensive RDO search process, Quad-LIC facilitates efficient GPU-based coding, offering substantial advantages for high-quality, real-time visual data transmission.

Table II presents the BD-CBR results for PSNR, coding speed, and computational complexity of Quad-DeepSC, compared to existing learned image transmission methods. For evaluating image transmission performance, the "VTM + 5G LDPC" scheme is adopted as the baseline and assessed on the 2K-resolution CLIC2021 test set. Quad-DeepSC achieves a noticeable reduction in bandwidth usage, while other DeepSC systems fail to yield any bandwidth savings. In terms of latency, Quad-DeepSC attains an encoding delay of 12.1 ms, making it the fastest among current DeepSC systems. The corresponding decoding delay is 16.2 ms, which is nearly symmetrical to the encoding latency. When transmission latency is considered, the convergence of these delays facilitates smooth and efficient wireless delivery of real-time visual data.

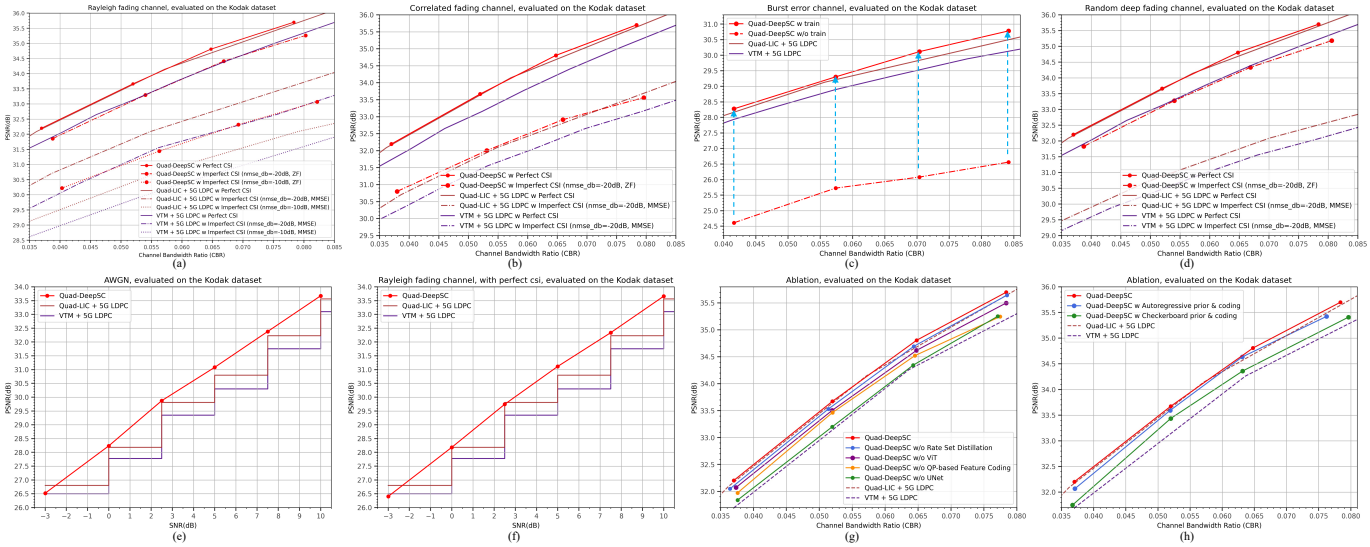


Fig. 8: PSNR of Quad-DeepSC under diverse channels at SNR $= 10$ dB. (a)-(d) involve imperfect CSI with $\mathrm{NMSE}_{\mathrm{dB}} \in \{-20, -10\}$ dB. (e)-(f) use perfect CSI. (a) Rayleigh fading channel. (b) Correlated fading channel with coefficient $\rho = 0.97$. (c) Burst-error channel with $p = 0.05$ (good→bad), $q = 0.2$ (bad→good), $\varepsilon = 0.01$ (erase), and $\kappa = 25$. (d) Random deep fading channel (LOS/NLOS blockages) with $p_{\mathrm{block}} = 0.02$, $L_m = 50$, $K = 10$ dB, $A_{\mathrm{blk}} = 20$ dB, and $\sigma_{\mathrm{sh}} = 2$. (e)-(f) PSNR vs. SNR over AWGN and rayleigh fading channel at average CBR $= 0.0515$. (g)-(h) Ablation studies on Quad-DeepSC over AWGN channel. All results are evaluated on the Kodak dataset.

This property constitute a distinctive advantage of the JSCC-based DeepSC paradigm, as traditional image codecs (e.g., VVC) require considerably longer encoding times, and channel codecs (e.g., 5G LDPC) introduce higher decoding delays.

c) Robustness Performance: All results in Fig. 8 are evaluated on the Kodak dataset with PSNR as the metric. Models are pre-trained on AWGN at 10 dB SNR and fine-tuned for 1,000 epochs with a learning rate of 1×10^{-5} under the respective channel before evaluation. Fig. 8 (a)–(d) report results at a fixed SNR of 10 dB. Fig. 8 (e) and (f) sweep the SNR. When equalization is required, Quad-DeepSC uses ZF, while baselines use minimum MSE (MMSE).

Fig. 8 (a) considers rayleigh fading. Under both perfect and imperfect CSI, Quad-DeepSC substantially outperforms the traditional schemes. With $\mathrm{NMSE}_{\mathrm{dB}} = -20$ dB, Quad-DeepSC matches the performance of "VTM + 5G LDPC" scheme under perfect CSI. With $\mathrm{NMSE}_{\mathrm{dB}} = -10$ dB, it remains comparable to "VTM + 5G LDPC" scheme at $\mathrm{NMSE}_{\mathrm{dB}} = -20$ dB, evidencing robustness to CSI degradation. Fig. 8 (b) evaluates correlated fading with coefficient $\rho = 0.97$, inducing moderately strong temporal correlation. Quad-DeepSC again exceeds the traditional schemes for both CSI settings and performs on par with "Quad-LIC + 5G LDPC" scheme. Fig. 8 (c) studies a burst-error channel modeled by a two-state markov process with $p = 0.05$ (good→bad), $q = 0.2$ (bad→good), erase probability $\varepsilon = 0.01$, and bad state noise inflation $\kappa = 25$. Quad-DeepSC markedly surpasses traditional schemes. Moreover, targeted training further improves its performance under this impairment. Fig. 8 (d) examines random deep fading via a LOS/NLOS process: LOS segments follow a rician model with $K = 10$ dB. NLOS segments are rayleigh with additional amplitude attenuation $A_{\mathrm{blk}} = 20$ dB and log-normal shadowing standard deviation $\sigma_{\mathrm{sh}} = 2$. Blockage entry probability is $p_{\mathrm{block}} = 0.02$ with mean blocked length $L_m = 50$ symbols. Quad-DeepSC again outperforms the traditional schemes, with an even larger margin under imperfect CSI. Fig. 8 (e) and (f) plot performance versus SNR from -3 to 10 dB over AWGN and rayleigh fading, respectively, at an average CBR of 0.0515 with perfect CSI. Across this range, Quad-DeepSC consistently exceeds the "VTM + 5G LDPC" benchmark. Overall, Quad-DeepSC adapts to diverse channel conditions and delivers robust end-to-end image transmission.

C. Ablation Studies

We first evaluate Quad-LIC in isolation for R-D performance, coding speed, and computational complexity. This serves as an ablation proxy for Quad-DeepSC, validating neural analyzer $g_a(\cdot)$, entropy estimator $p_{\hat{\mathbf{y}}}(\cdot)$, and neural synthesizer $g_s(\cdot)$. We then ablate Quad-DeepSC on Kodak dataset by removing or replacing key components, with emphasis on feature coding modules $f_{tx}(\cdot)$ and $f_{rx}(\cdot)$. Fig. 8 (g) tests removal of rate set distillation (RSD), the UNet in $g_s(\cdot)$, ViT blocks in $f_{tx}(\cdot)$ and $f_{rx}(\cdot)$, and quadtree partition–based feature coding in $f_{tx}(\cdot)$ and $f_{rx}(\cdot)$. Removal is denoted "w/o" (e.g., "w/o UNet"). Fig. 8 (h) compares partition strategies for entropy modeling and feature coding. Keeping the backbone fixed, we replace quadtree partitioning with checkerboard or channel-wise autoregressive alternatives in $p_{\hat{\mathbf{y}}}(\cdot)$, $f_{tx}(\cdot)$ and $f_{rx}(\cdot)$. Table III summarizes the corresponding quantitative results.

Experimental results show that RSD improves performance, reduces memory consumption, and slightly accelerates the encoding process. Gains arise from a smaller search space

TABLE III: Ablation studies on Quad-DeepSC, evaluated on the Kodak dataset with NVIDIA GeForce RTX 4090. The best and second-best results are shown in **bold** and underlined, respectively.

	Enc time (ms)	**Dec time (ms)**	$g_a(\cdot)$+$p_{\hat{\mathbf{y}}}(\cdot)$ **(ms)**	$g_s(\cdot)$ **(ms)**	$f_{tx}(\cdot)$ **(ms)**	$f_{rx}(\cdot)$ **(ms)**	$Q_k(\mathbf{k})$ **PNG enc (ms)**	$Q_k(\mathbf{k})$ **PNG dec (ms)**	**BD-CBR (%)**	**MACs (G)**	**Params (M)**
VTM + 5G LPDC	–	–	–	–	–	–	–	–	0.0	–	–
Quad-DeepSC w/o UNet for $g_s(\cdot)$	12.1	15.2	3.8	**1.4**	8.3	13.8	**0.218**	0.231	0.8	**382.74**	50.58
Quad-DeepSC w/o Rate set distillation	13.0	16.6	3.8	2.4	9.2	14.2	0.240	0.241	-7.0	422.64	68.15
Quad-DeepSC w/o Quadtree partition-based feature coding	**8.1**	**5.8**	3.8	2.4	**4.3**	**3.4**	0.222	0.222	-3.1	422.47	54.23
Quad-DeepSC w/o ViT	8.3	9.4	3.8	4.1	4.5	5.3	0.226	0.236	-4.9	421.71	**50.52**
Quad-DeepSC w/ Checkerboard prior & coding	10.7	8.1	**3.6**	2.4	7.1	5.7	0.226	**0.211**	-2.1	422.79	73.82
Quad-DeepSC w/ Autoregressive prior & coding	18.4	14.8	4.1	2.4	14.3	12.4	0.243	0.222	-6.7	420.78	65.88
Quad-DeepSC	12.1	16.2	3.8	2.4	8.3	13.8	0.221	0.233	**-8.8**	422.64	50.58

for side information $\mathbf{k}$ and removal of redundant rate embeddings and FC layers. Removing the UNet in $g_s(\cdot)$ degrades reconstruction and robustness to channel impairments while reducing floating-point operations. We benchmark quadtree partition–based coding against checkerboard and autoregressive variants. Following [15], checkerboard variant splits feature into two groups. The checkerboard spatial indices (i_0, i_1) alternate every $C_y/4$ along channels (group 1: $[i_0, i_1, i_0, i_1]$, group 2: $[i_1, i_0, i_1, i_0]$). Autoregressive variant splits features into five groups as in [5] along channel dimension, with $C_y = 320$ partitioned into $[16, 16, 32, 64, 192]$. Each partition scheme uses matching $p_{\hat{\mathbf{y}}}(\cdot)$, $f_{tx}(\cdot)$ and $f_{rx}(\cdot)$. From Fig. 8 (h) and Table III, quadtree partition–based coding delivers the largest gains and the lowest overall complexity. Autoregressive variant is the next best. Checkerboard variant performs weakest. For average speed, checkerboard variant is fastest and autoregressive variant slowest. Within the quadtree partition–based coding, replacing its feature coder with a uniform 4-layer ViT blocks shows entropy-adapted feature coding improves performance at a cost in speed. Conversely, swapping ViT blocks in $f_{tx}(\cdot)$ and $f_{rx}(\cdot)$ for DepthConvBlocks (all-CNN architecture) yields only a modest performance drop, indicating quadtree partition–based coding contributes more than ViT blocks. In addition, Table III shows that the coding latency of the side information $\mathbf{k}$ is on the order of 0.1 ms, which is negligible compared with other modules. The quantized index $Q_k(\mathbf{k})$ is generated immediately after the entropy estimation of $p_{\hat{\mathbf{y}}}(\cdot)$. It can be transmitted concurrently with the operation of $f_{tx}(\cdot)$, thereby introducing no additional latency to the overall DeepSC system.

V. Conclusion

This paper proposed a deep learning based semantic communication system for images, inspired by quadtree partition as a feature segmentation method, named Quad-DeepSC. It is designed to balance rate-distortion performance and computational complexity, while being well suited for real-time wireless transmission of images with various resolutions. The core innovations of Quad-DeepSC are as follows. First, it employs quadtree partitioning for efficient feature coding and entropy modeling. Second, it introduces an optimized learned image codec (LIC), named quadtree partition-based LIC (Quad-LIC), which achieves excellent source compression performance with minimal coding latency. Third, it embeds Quad-LIC into the overall framework through an end-to-end training strategy, further enhancing system performance. Experimental results demonstrate that Quad-DeepSC surpasses the traditional "VTM + 5G LDPC" scheme in transmission efficiency, while maintaining low latency and strong robustness under diverse channel conditions. The Quad-LIC related components are CNN-based, whereas the feature coding module adopts a transformer structure. Considering that recent LIC increasingly utilize transformer or mamba architectures, future work will extend DeepSC toward these models to incorporate long-range dependency learning and dynamic state modeling, thereby improving coding efficiency. In conclusion, Quad-DeepSC provides an ideal solution for efficient image transmission in future real-time communication scenarios. It has significant theoretical and practical value, effectively facilitating the integration of AI with communication systems and promoting the application of semantic communication.

References

[1] Z. Qin, L. Liang, Z. Wang, S. Jin, X. Tao, W. Tong, and G. Y. Li, "AI empowered wireless communications: From bits to semantics," *Proc. IEEE*, vol. 112, no. 7, pp. 621–652, Jul. 2024.

[2] Z. Qin, J. Ying, D. Yang, H. Wang, and X. Tao, "Computing networks enabled semantic communications," *IEEE Netw.*, vol. 38, no. 2, pp. 122–131, Mar. 2024.

[3] C.-X. Wang, X. You, X. Gao, X. Zhu, Z. Li, C. Zhang, H. Wang, Y. Huang, Y. Chen, H. Haas *et al.*, "On the road to 6g: Visions, requirements, key technologies, and testbeds," *IEEE Commun. Surv. Tutor.*, vol. 25, no. 2, pp. 905–974, 2023.

[4] N. Ling, C.-C. J. Kuo, G. J. Sullivan, D. Xu, S. Liu, H.-M. Hang, W.-H. Peng, J. Liu *et al.*, "The future of video coding," *APSIPA Trans. Signal Inf. Process.*, vol. 11, no. 1, Jan. 2022.

[5] D. He, Z. Yang, W. Peng, R. Ma, H. Qin, and Y. Wang, "ELIC: Efficient learned image compression with unevenly grouped space-channel contextual adaptive coding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2022, pp. 5718–5727.

[6] J. Li, B. Li, and Y. Lu, "Neural video compression with diverse contexts," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2023, pp. 22 616–22 626.

[7] J. Zhao, Y. Wang, and Q. Zhang, "Adaptive CU split decision based on deep learning and multifeature fusion for H. 266/VVC," *Sci. Program.*, vol. 2020, no. 1, p. 8883214, Aug. 2020.

[8] Z. Jia, B. Li, J. Li, W. Xie, L. Qi, H. Li, and Y. Lu, "Towards practical real-time neural video compression," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, Nashville, TN, USA, Jun. 2025, pp. 11–25.

[9] H. Xie, Z. Qin, G. Y. Li, and B.-H. Juang, "Deep learning enabled semantic communication systems," *IEEE Trans. Signal Process.*, vol. 69, pp. 2663–2675, Apr. 2021.

[10] E. Bourtsoulatze, D. B. Kurka, and D. Gündüz, "Deep joint source-channel coding for wireless image transmission," *IEEE Trans. Cognit. Comm. Netw.*, vol. 5, no. 3, pp. 567–579, May. 2019.

[11] D. B. Kurka and D. Gündüz, "Bandwidth-agile image transmission with deep joint source-channel coding," *IEEE Trans. Wirel. Comm.*, vol. 20, no. 12, pp. 8081–8095, Dec. 2021.

[12] Q. Fu, H. Xie, Z. Qin, G. Slabaugh, and X. Tao, "Vector quantized semantic communication system," *IEEE Wirel. Commun. Lett.*, vol. 12, no. 6, pp. 982–986, 2023.

[13] J. Dai, S. Wang, K. Tan, Z. Si, X. Qin, K. Niu, and P. Zhang, "Nonlinear transform source-channel coding for semantic communications," *IEEE J. Select. Areas Commun.*, vol. 40, no. 8, pp. 2300–2316, Aug. 2022.

[14] S. Wang, J. Dai, X. Qin, Z. Si, K. Niu, and P. Zhang, "Improved nonlinear transform source-channel coding to catalyze semantic communications," *IEEE J. Sel. Topics Signal Process.*, vol. 17, no. 5, pp. 1022–1037, Sept. 2023.

[15] D. He, Y. Zheng, B. Sun, Y. Wang, and H. Qin, "Checkerboard context model for efficient learned image compression," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2021, pp. 14 771–14 780.

[16] "VVC VTM reference software," https://vcgit.hhi.fraunhofer.de/jvet/VVCSoftware_VTM, accessed: 2025-04-13.

[17] 3GPP, "3GPP TS 38.214 version 16.2.0 Release 16: 5G; NR; Physical layer procedures for data," https://www.etsi.org/deliver/etsi_ts/138200_138299/138214/16.02.00_60/ts_138214v160200p.pdf, 2020, accessed: 2025-04-13.

[18] G. Zhang, P. Yang, Y. Cai, Q. Hu, and G. Yu, "From analog to digital: Multi-order digital joint coding-modulation for semantic communication," *IEEE Trans. Commun.*, 2024, accepted to appear.

[19] G. Zhang, H. Li, Y. Cai, Q. Hu, G. Yu, and Z. Qin, "Progressive learned image transmission for semantic communication using hierarchical vae," *IEEE Trans. Cognit. Comm. Netw.*, 2025, accepted to appear.

[20] D. Minnen, J. Ballé, and G. D. Toderici, "Joint autoregressive and hierarchical priors for learned image compression," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 31, 2018.

[21] M. Lu, F. Chen, S. Pu, and Z. Ma, "High-efficiency lossy image coding through adaptive neighborhood information aggregation," *arXiv preprint arXiv:2204.11448*, Apr. 2022.

[22] F. Lin, H. Sun, J. Liu, and J. Katto, "Multistage spatial context models for learned image compression," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2023, pp. 1–5.

[23] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Toulon, France, Apr. 2017.

[24] Z. Guo, Z. Zhang, R. Feng, and Z. Chen, "Soft then hard: Rethinking the quantization in neural image compression," in *Proc. Int. Conf. Mach. Learn. (ICML)*, vol. 139, Jul. 2021, pp. 3920–3929.

[25] S. Pan, C. Finlay, C. Besenbruch, and W. Knottenbelt, "Three gaps for quantisation in learned image compression," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. Workshops (CVPRW)*, Jun. 2021, pp. 720–726.

[26] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018.

[27] H. Xie, Z. Qin, X. Tao, and K. B. Letaief, "Task-oriented multi-user semantic communications," *IEEE J. Select. Areas Commun.*, vol. 40, no. 9, pp. 2584–2597, Sept. 2022.

[28] E. Björnson, J. Hoydis, and L. Sanguinetti, "Massive MIMO networks: Spectral, energy, and hardware efficiency," *Found. Trends Signal Process.*, vol. 11, no. 3-4, pp. 154–655, 2017.

[29] S. Akin and M. C. Gursoy, "Training optimization for gauss-markov rayleigh fading channels," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2007, pp. 5999–6004.

[30] E. N. Gilbert, "Capacity of a burst-noise channel," *Bell Syst. Tech. J.*, vol. 39, no. 5, pp. 1253–1265, 1960.

[31] T. Bai and R. W. Heath, "Coverage and rate analysis for millimeter-wave cellular networks," *IEEE Trans. Wireless Commun.*, vol. 14, no. 2, pp. 1100–1114, 2015.

[32] M. Gapeyenko, A. Samuylov, M. Gerasimenko, D. Moltchanov, S. Singh, M. R. Akdeniz, E. Aryafar, N. Himayat, S. Andreev, and Y. Koucheryavy, "On the temporal effects of mobile blockers in urban millimeter-wave cellular scenarios," *IEEE Trans. Veh. Technol.*, vol. 66, no. 11, pp. 10 124–10 138, 2017.

[33] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2021, pp. 10 012–10 022.

[34] Q. Hu, G. Zhang, Z. Qin, Y. Cai, G. Yu, and G. Y. Li, "Robust semantic communications with masked VQ-VAE enabled codebook," *IEEE Trans. Wirel. Commun.*, vol. 22, no. 12, pp. 8707–8722, Dec. 2023.

[35] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. Ieee, 2009, pp. 248–255.

[36] E. Agustsson and R. Timofte, "Ntire 2017 challenge on single image super-resolution: Dataset and study," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. Workshops (CVPRW)*, 2017, pp. 126–135.

[37] "Kodak photocd dataset," http://r0k.us/graphics/kodak/, 1993, accessed: 2025-04-13.

[38] "Challenge on learned image compression," http://compression.cc/, 2020, accessed: 2025-04-13.

[39] "Challenge on learned image compression," http://compression.cc/, 2021, accessed: 2025-04-13.

[40] F. Bellard, "BPG image format," https://bellard.org/bpg, accessed: 2025-04-13.

[41] "HEVC HM reference software," https://hevc.hhi.fraunhofer.de, accessed: 2025-04-13.

[42] J. Bégaint, F. Racapé, S. Feltman, and A. Pushparaja, "Compressai: a pytorch library and evaluation platform for end-to-end compression research," *arXiv preprint arXiv:2011.03029*, Nov. 2020.

[43] Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, "Learned image compression with discretized gaussian mixture likelihoods and attention modules," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2020, pp. 7939–7948.

[44] K. Yang, S. Wang, J. Dai, X. Qin, K. Niu, and P. Zhang, "SWINJSCC: Taming swin transformer for deep joint source-channel coding," *IEEE Trans. Cognit. Comm. Netw.*, vol. 11, no. 1, pp. 90–104, Feb. 2024.

[45] T. Richardson and S. Kudekar, "Design of low-density parity check codes for 5g new radio," *IEEE Commun. Mag.*, vol. 56, no. 3, pp. 28–34, Mar. 2018.

[46] J. Hoydis, S. Cammerer, F. Ait Aoudia, M. Nimier-David, L. Maggi, G. Marcus, A. Vem, and A. Keller, "Sionna," 2022, https://nvlabs.github.io/sionna/.

[47] G. Bjontegaard, "Calculation of average psnr differences between rd-curves," *ITU SG16 Doc. VCEG-M33*, 2001.