entropy

MDPI

Article

U-Shaped Split Federated Learning with Compact Features for Deep Learning-Based Image Coding

Qizheng Sun [1], Caili Guo [1,*], Meiyi Zhu [2] and Yang Yang [1]

[1] Beijing Key Laboratory of Network System Architecture and Convergence, School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China; qizheng_sun@bupt.edu.cn (Q.S.); yangyang01@bupt.edu.cn (Y.Y.)
[2] Department of Engineering, King's College London, London WC2R 2LS, UK; meiyi.1.zhu@kcl.ac.uk
* Correspondence: guocaili@bupt.edu.cn

Abstract

U-shaped Split Federated Learning (U-SFL) is a promising paradigm for distributed image coding, offering parallel training capabilities and privacy preservation while mitigating computational burdens on edge devices. However, the frequent bidirectional transmission of intermediate features between dual-split points incurs substantial communication overhead. To mitigate this issue, we propose a compact-feature U-shaped split federated learning framework (CoF U-SFL), which reduces communication overhead and improves training efficiency while maintaining low image distortion. We introduce a feature entropy estimation network to model the distribution of split-layer features, enabling effective compression during transmission. Furthermore, we formulate a joint optimization objective incorporating entropy constraints to guide the end-to-end training. Experimental results demonstrate that CoF U-SFL reduces communication overhead by 104.6 times while maintaining reconstruction performance.

Keywords: U-shape split federated learning; image coding; distributed learning; entropy estimation; joint source channel coding

1. Introduction

With the exponential growth of image data, a vast amount of visual information is continuously generated at edge devices, necessitating efficient image coding methods to support effective transmission and storage. Traditional approaches typically rely on centralized learning (CL) frameworks, where raw image data from all devices is transmitted to a central server for unified model training. However, this paradigm faces significant uplink communication overhead and poses severe privacy leakage risks, making it challenging to deploy in practical and data-sensitive scenarios [1].

To address privacy concerns, federated learning (FL) [2–4] has been proposed. In FL, clients train the entire model locally and only upload model parameters, thereby avoiding the exposure of raw data. However, FL implicitly assumes that edge devices have the capability to deploy and execute large-scale deep neural networks, especially those used for high-fidelity image encoding and decoding, such as autoencoders or Transformers. This assumption is often invalid in resource-constrained edge devices due to their strict limitations in computational power and memory resources.

To alleviate these limitations, split learning (SL) [5] was introduced by partitioning the model at specific layers: only the initial shallow layers are deployed on the client

Check for updates

Academic Editors: Min Qiu, Xiaowei Wu, Peng Kang and Jinhong Yuan

Received: 6 February 2026
Revised: 12 March 2026
Accepted: 14 March 2026
Published: 16 March 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

side, while the computationally intensive deeper layers are offloaded to the server. The training process involves iteratively exchanging intermediate activations and gradients between the split layers. Although this approach significantly reduces the computational and memory requirements on the client's side, it introduces considerable time delays and significant communication overhead due to the frequent transmission of high-dimensional intermediate data.

Recent studies have introduced split federated learning (SFL) [6–9] to combine the strengths of FL and SL. SFL retains the parallel training capability of FL while employing model partitioning to reduce the computational burden on clients. Similar to SL, SFL also suffers from significant communication overhead due to the frequent exchange of high-dimensional intermediate activations and gradients. Furthermore, SFL faces a privacy challenge in image coding tasks: calculating reconstruction loss requires access to original images, which cannot be transmitted to the server.

To enable collaborative training while preserving data confidentiality in image coding tasks, the U-shaped split federated learning (U-SFL) framework [10–13] was proposed. This framework partitions the global model into a client-side head, a server-side body, and a client-side tail network. Consequently, original images remain local, and optimization relies solely on the bidirectional exchange of intermediate data. U-SFL has demonstrated its effectiveness in preserving data privacy across various applications, including medical imaging [12], vehicular networks [11], and facial privacy protection [13]. However, the dual-split architecture necessitates the transmission of two sets of high-dimensional tensors, namely forward activations and backward gradients, during each training round. This results in significantly higher communication overhead than single-split SL or SFL, creating a bottleneck for training efficiency in bandwidth-constrained edge environments.

To alleviate this burden, various compression techniques have been developed for distributed learning paradigms. In FL, representative approaches include parameter coding via Gaussian mixture models [14], clustering-based dimensionality reduction [15], autoencoder-assisted model update compression and reconstruction [16], and attention-driven adaptive fusion of client updates [17]. Additionally, recent advances explore methods like sparse subnetwork identification through weight freezing and binary mask transmission [18]. In SL, compression targets intermediate representations directly. Notable methods include end-to-end feature compression using autoencoders [19], batch-level activation fusion through cyclic convolutions in C3-SL [20], and channel-wise sparsification based on adaptive importance scoring in SL-ACC [21]. In SFL, further innovations include (i) local loss-driven parallel training to eliminate server-to-client communication [22]; (ii) auxiliary networks enabling client-side weight updates to reduce synchronization frequency [23]; and (iii) quantization-based compression techniques, ranging from system-aware dynamic model splitting for latency reduction [24] to adaptive feature-wise dropout and quantization for bandwidth efficiency [25]. However, these compression approaches often incur distortion in the transmitted features. In contrast, compression research for U-SFL is still in its early stages. Existing efforts [11] primarily rely on simplifying the client head; for example, by fixing its parameters or employing fixed embedding modules. Although semantic autoencoders (SAEs) have been explored to reduce feature dimensionality [26], such projection-based compression often introduces distortions that degrade reconstruction fidelity. Consequently, there remains a need for an efficient, low-distortion compression mechanism specifically designed for the bidirectional high-dimensional data streams in U-SFL.

In this paper, we propose a U-shaped split federated learning framework with compact features (CoF U-SFL), which aims to reduce communication overhead and improve training

efficiency while maintaining low image distortion and strong privacy protection. The main contributions of this paper are summarized as follows:

- We propose a feature entropy estimation network designed to model the statistical distribution of split-layer features. By utilizing a dual-layer Transformer-based hyperprior architecture, this network achieves accurate entropy estimation without accessing raw data, achieving a favorable balance between estimation accuracy and model complexity.
- We introduce CoF U-SFL, a novel U-shaped split federated learning framework that supports compact feature representations. By leveraging an entropy-guided quantization strategy, our method significantly reduces the bit-width of high-dimensional intermediate features while preserving image reconstruction quality and inherently guaranteeing client-side data privacy.
- We establish a robust training pipeline governed by a joint objective function that integrates entropy constraints with image distortion metrics. This formulation enables unified optimization of both the entropy estimation network and the image coding network, facilitating convergence towards optimal coding efficiency.
- We conduct comprehensive experiments across multiple datasets to validate the effectiveness of CoF U-SFL. Results demonstrate that the proposed method reduces communication overhead by approximately 104.6 times, while maintaining comparable image reconstruction performance.

2. Related Work

2.1. Evolution of Collaborative Learning Paradigms

Collaborative learning has evolved through several paradigms to address the tension between data privacy and computational constraints. Initially, FL [2–4] was introduced to keep raw data local. However, the requirement for clients to execute full models imposes substantial computational burdens. To mitigate this, SL [5,27] partitions the model, offloading intensive deep layers to a server. Building on this, SFL [6–9] incorporates parallelism into SL to improve scalability. For tasks specifically requiring end-to-end data confidentiality, such as image coding, U-SFL [10,11] has emerged as the key architecture. Yang et al. [12] applied it to medical image restoration for patient data protection, Zheng et al. [11] utilized it in vehicular perception networks, and other studies have extended it to facial privacy preservation [13]. The evolution of collaborative learning has been driven by the dual need for data privacy and resource efficiency, with communication compression playing a pivotal role in each paradigm [24], particularly for handling the communication-intensive bidirectional transmission characteristic of U-SFL.

2.2. Communication Compression in Federated Learning

In FL, communication bottlenecks arise from the transmission of high-dimensional model updates. To address this, statistical approaches have been proposed, such as encoding gradients via Gaussian mixture models [14] or employing clustering-based dimensionality reduction [15]. Structural optimization methods further reduce overhead through sparse subnetwork identification using binary masks [18] or attention-driven adaptive fusion [17]. Additionally, auxiliary-based techniques utilize autoencoders to compress and reconstruct model updates [16]. However, these methods focus exclusively on compressing static model parameters or gradients derived from full-model local training. They are inapplicable to split architectures, where the communication payload consists of dynamic intermediate features that vary with each input sample.

2.3. Intermediate Feature Compression in Split Learning

In SL, the frequent transmission of high-dimensional intermediate activations constitutes a major bottleneck. Compression techniques have been developed to target these representations directly. End-to-end approaches integrate lightweight autoencoders at the split layer to compress features [19]. Batch-level strategies, such as C3-SL [20], employ cyclic convolutions to fuse activation maps from multiple samples into a compact representation. Furthermore, structural sparsification methods, like SL-ACC [21], utilize adaptive importance scoring to transmit only task-sensitive channels. However, these methods are inherently designed for a single split point. They do not account for the dual-split architecture of U-SFL, where error signals need to propagate through two independent network segments. Compared to standard SL, U-SFL involves more complex bidirectional interactions among the head, body, and tail networks.

2.4. Compression Mechanisms in Split Federated Learning

SFL combines the parallelism of FL with the model partitioning of SL, necessitating strategies that simultaneously handle aggregation and feature transmission. Communication path elimination has been proposed to enable local loss-driven parallel training, effectively removing server-to-client gradient transmission [22]. To reduce synchronization frequency, auxiliary networks have been introduced to approximate server-side gradients locally [23]. Recent works have also explored quantization-based techniques, ranging from system-aware dynamic model splitting [24] to adaptive feature-wise dropout combined with quantization [25]. Despite these advances, quantization and dropout-based methods inevitably introduce information distortion in the transmitted features. Furthermore, methods relying on local losses to eliminate the backward path [22] are structurally incompatible with U-SFL. Since U-SFL computes the loss at the client-side tail, a complete backward communication link is strictly required to propagate supervision signals back to the body and head networks.

2.5. Compression Challenges and Research Gap in U-SFL

U-SFL achieves end-to-end privacy preservation by placing both the encoder head and decoder tail on the client, effectively keeping raw inputs and reconstruction targets local. However, this architectural benefit comes at the cost of doubled communication overhead compared to standard SL, as it requires the bidirectional exchange of high-dimensional intermediate features. Currently, compression research tailored for U-SFL remains nascent. Existing implementations, such as those by Zheng et al. [11], primarily simplify the client head by fixing its parameters or utilizing non-trainable embedding modules, thereby avoiding explicit feature compression at the expense of model flexibility. While recent attempts have employed Semantic Autoencoders (SAEs) to reduce feature dimensionality [26], such projection-based methods often introduce uncontrollable distortion. In summary, the current literature lacks a unified framework that simultaneously addresses the bidirectional communication bottleneck and the high-fidelity reconstruction requirement of U-SFL.

3. System Model

This section establishes the system model for the proposed CoF U-SFL scheme. As illustrated in Figure 1, the system primarily consists of N distributed clients (denoted by the set $\mathcal{N} = \{1, \ldots, N\}$), an edge server, and a Federated (Fed) server. During distributed training, the edge server and the clients collaboratively execute the forward and backward propagation processes. Specifically, the edge server hosts the intermediate body network, while the clients manage their respective head and tail networks. Crucially, to prevent recon-

struction attacks arising from simultaneous access to models and activations [28], the Fed server operates independently from the edge server to perform periodic model aggregation.

In the end-to-end deep joint source channel coding (JSCC) pipeline, let $\boldsymbol{x}_i$ denote the raw input data of client i. To account for the stochastic nature of wireless transmission environments within the U-SFL architecture, the feature exchange process between the client and the edge server is modeled as an additive white Gaussian noise channel (AWGN) at the two split points involving the head–body interface and the body–tail interface. The received feature $\boldsymbol{a}_i'$ can be expressed as

$$\boldsymbol{a}_i' = \boldsymbol{a}_i + \boldsymbol{n},$$

where $\boldsymbol{a}_i$ represents the original feature representation transmitted at the corresponding split point, and $\boldsymbol{n} \sim \mathcal{N}(\boldsymbol{0}, \sigma^2 \boldsymbol{I})$ denotes the noise term introduced by the AWGN channel. The communication quality of this link is quantified by the signal-to-noise ratio (SNR) predefined during training.

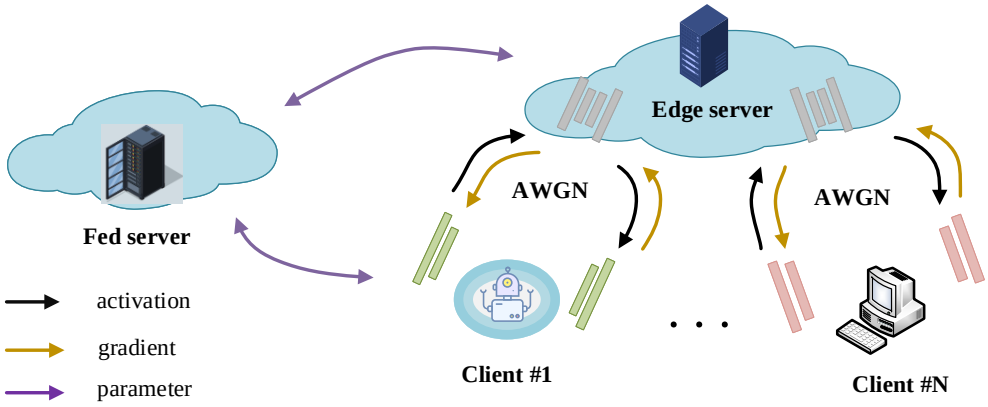


Figure 1. The system model of the proposed CoF U-SFL method.

During the model optimization phase, the AWGN channel is directly embedded into the end-to-end computational graph, effectively serving as a non-trainable stochastic linear operator. In the forward pass, the system performs noise sampling and overlays it onto the features at each split point, ensuring that subsequent networks operate on the received features $\boldsymbol{a}_i'$ to compute the training loss. Under this mechanism, the channel noise participates in the closed-loop gradient conduction via the chain rule, thereby imposing the stochastic constraints of the communication environment directly onto the parameter updates of each sub-network. This channel-aware training mechanism [29] drives the model to extract channel-robust features, implicitly incorporating error tolerance into the training process. Furthermore, following the classical assumption in deep JSCC [29], this paper assumes error-free gradient transmission during back-propagation to ensure the convergence stability of the distributed model training.

Since U-SFL requires frequent exchange of features $\boldsymbol{a}_i$ at both split points during training, the communication overhead becomes a core bottleneck. Therefore, it is necessary to investigate methods for reducing the feature data volume during training while maintaining image quality.

4. The Proposed U-Shaped Split Federated Learning Framework with Compact Features

In this section, we elaborate on the proposed U-shaped split federated learning framework with compact features (CoF U-SFL). Specifically, this framework aims to reduce communication overhead and improve efficiency during the collaborative training process,

while maintaining low image distortion and strong privacy protection. We first present the overall system architecture and the bidirectional data flow in Section 4.1. Subsequently, we detail the core component of our method, the Transformer-based feature entropy estimation module, which captures the statistical distribution of intermediate features for accurate bit-rate estimation in Section 4.2. Then, the joint optimization objective, which balances the communication rate of intermediate features and image reconstruction fidelity, is formulated in Section 4.3. Finally, we describe the end-to-end collaborative training procedure in Section 4.4.

4.1. Framework Overview

As depicted in Figure 2, each client i processes its local dataset $\mathcal{D}_i$ using the resident head network $\boldsymbol{w}_{ch,i}$ and tail network $\boldsymbol{w}_{ct,i}$. The heavy computational load is offloaded to the edge server, which executes the body network $\boldsymbol{w}_{sb,i} = [\boldsymbol{\theta}_{sb}; \boldsymbol{\theta}_{m,i}]$. To handle client heterogeneity, this body network comprises a shared part $\boldsymbol{\theta}_{sb}$, synchronized per round, and a personalized part $\boldsymbol{\theta}_{m,i}$, which is aggregated every I rounds.

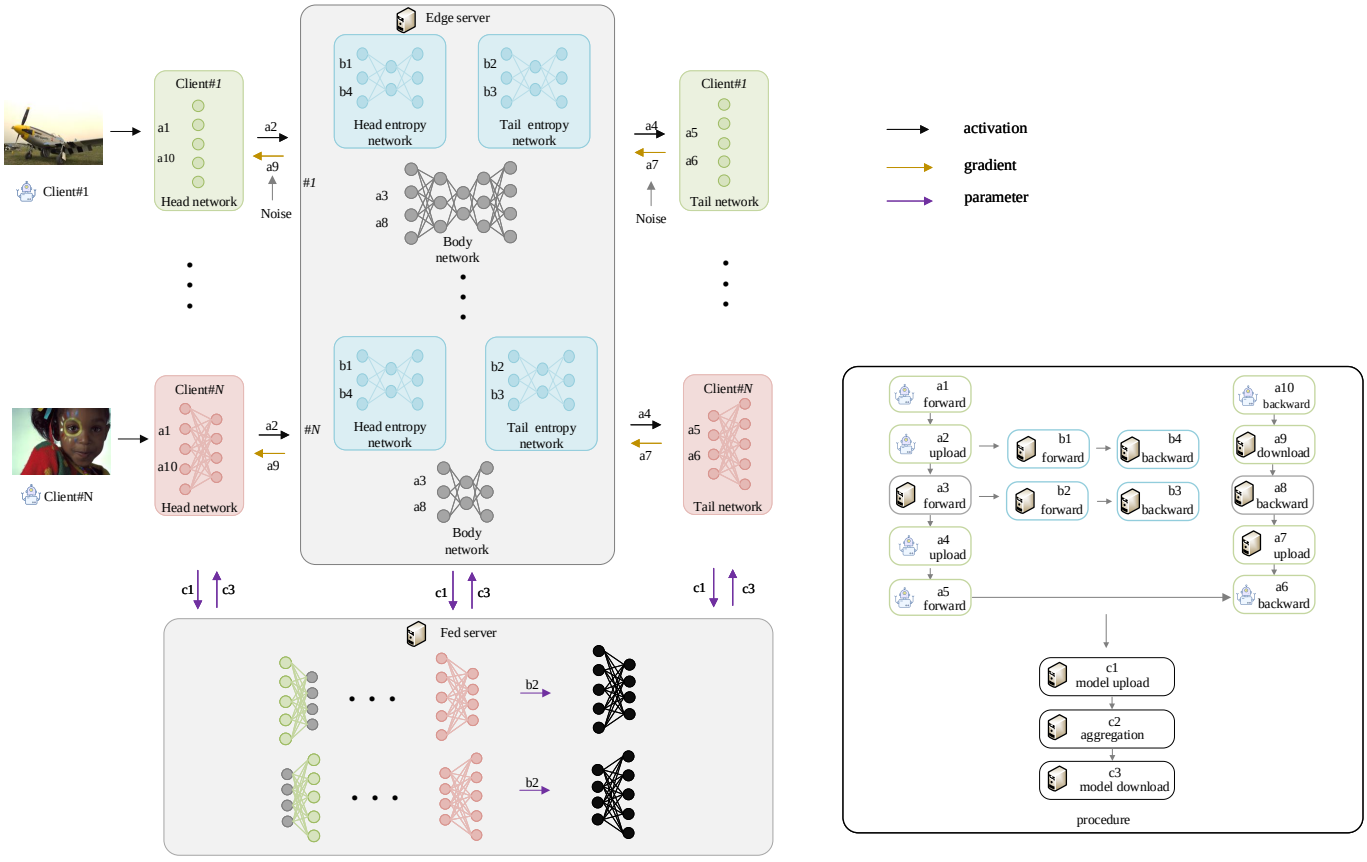


Figure 2. The detailed workflow of the proposed CoF U-SFL framework: (a) Forward and backward propagation of the primary image coding model ($a1$–$a5$ for forward propagation, $a6$–$a10$ for backward propagation); (b) Forward and backward propagation of the entropy estimation model for feature compression; (c) Global parameter aggregation via the Fed server.

This architecture establishes an end-to-end JSCC pipeline [30,31] between clients and the edge server. To minimize communication overhead, Transformer-based entropy estimation modules are embedded at the split points to guide efficient feature compression (detailed in Section 4.2). Furthermore, the entire system is driven by a joint optimization objective that balances reconstruction fidelity and transmission rate (formulated in Section 4.3). In Figure 2, data flows are color-coded, where black, yellow, and purple lines represent the transmission of features, gradients, and parameters, respectively.

4.2. Transformer-Based Feature Entropy Estimation Mechanism

In this subsection, we propose a specialized feature entropy estimation mechanism to capture the spatial redundancy within the split-layer features.

As illustrated in Figure 3, the mechanism comprises two distinct modules: the head–body entropy estimation network and the body–tail entropy estimation network. Both modules incorporate a dual-layer Transformer-based hyperprior architecture, utilizing self-attention mechanisms to effectively model the global dependencies of the feature distribution. Regarding the architectural depth, we specifically adopt a dual-layer configuration to balance performance and overhead. A single-layer structure often lacks the representational capacity to accurately estimate the distribution parameters, while a deeper network inevitably increases computational latency. Therefore, the proposed dual-layer design aims to ensure sufficient estimation fidelity while maintaining high computational efficiency. This module incorporates hyperprior analysis and synthesis networks (comprising patch merging and transformer blocks) to estimate the spatial mean (μ) and scale (σ) of the intermediate features. This estimated distribution is subsequently utilized to calculate the feature entropy and integrated into the final loss function, guiding the end-to-end rate-distortion optimization.

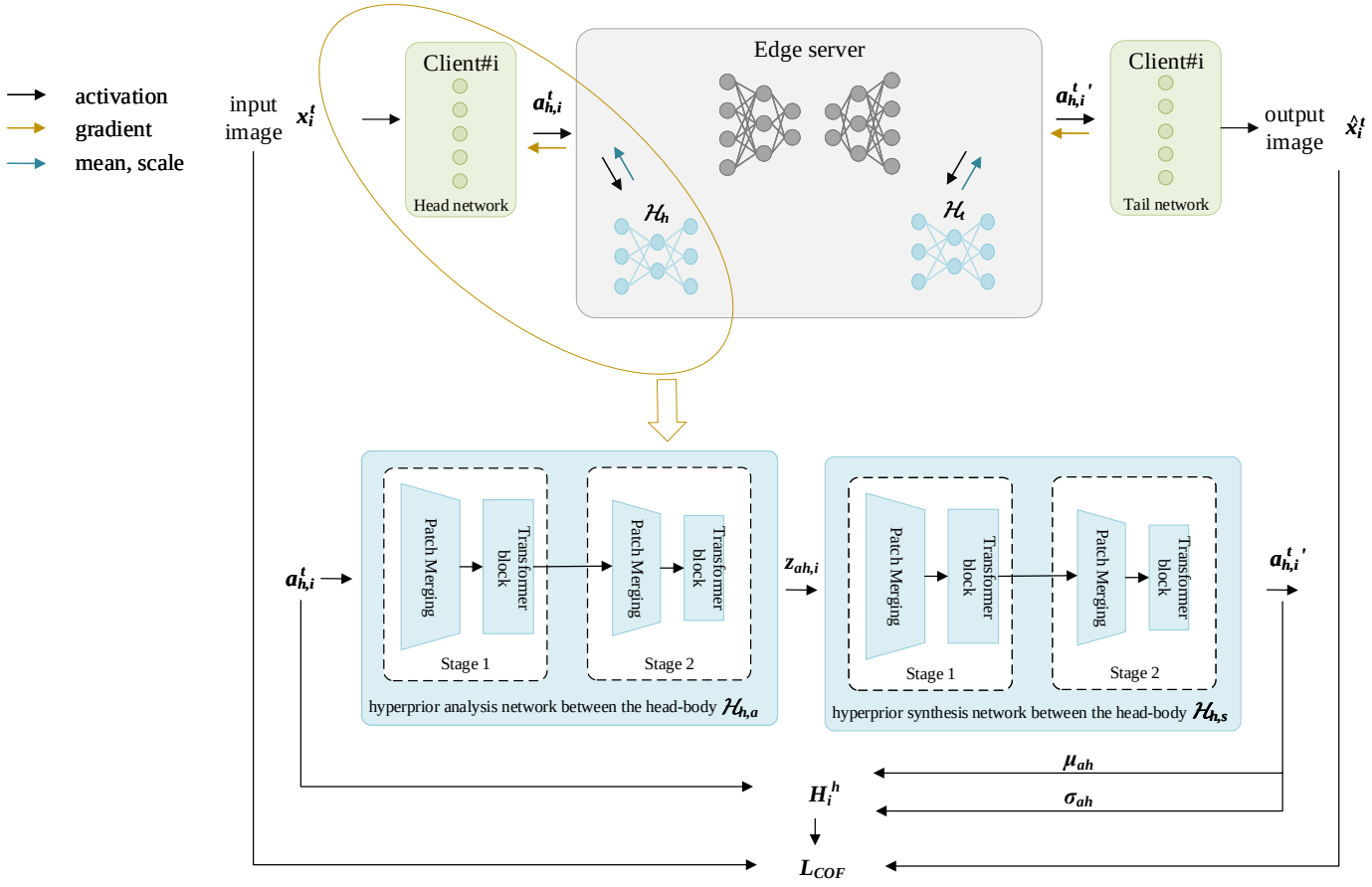


Figure 3. Detailed illustration of the feature entropy estimation process at the split point. The figure depicts the internal structures of the hyperprior networks, demonstrating how these modules estimate the spatial mean and scale parameters to formulate the training loss. Specifically, the black, yellow, and blue arrows indicate the forward activations, backward gradients, and the estimated parameters (μ_{ah} and σ_{ah}), respectively.

The detailed workflow begins at the head–body interface. First, the raw image is processed by the head network to generate intermediate features $\boldsymbol{a}_{h,i}^{t}$:

$$\boldsymbol{a}_{h,i}^{t} = \mathcal{H}\left(\boldsymbol{x}_{i}^{t}; \boldsymbol{w}_{ch,i}^{t-1}\right), \tag{1}$$

where $\mathcal{H}(\cdot)$ denotes the mapping function of the head network, and $\boldsymbol{w}_{ch,i}^{t-1}$ represents its learnable parameters. To estimate the bit-rate, we employ a hyperprior network consisting of an analysis network and a synthesis network. Specifically, $\boldsymbol{a}_{h,i}^{t}$ is compressed by the hyperprior analysis network to obtain the latent representation $\boldsymbol{z}_{ah,i}^{t}$:

$$\boldsymbol{z}_{ah,i}^{t} = \mathcal{H}_{h,a}\left(\boldsymbol{a}_{h,i}^{t}; \boldsymbol{ha}_{sh,i}^{t-1}\right), \tag{2}$$

where $\mathcal{H}_{h,a}(\cdot)$ consists of two stacked Transformer layers capturing global context, and $\boldsymbol{ha}_{sh,i}^{t-1}$ denotes the parameters of the hyperprior analysis network. Subsequently, $\boldsymbol{z}_{ah,i}^{t}$ is decoded by the hyperprior synthesis network to predict the parameters of the underlying Gaussian distribution, yielding the mean $\boldsymbol{\mu}_{ah}$ and variance $\boldsymbol{\sigma}_{ah}$:

$$\boldsymbol{\mu}_{ah}, \boldsymbol{\sigma}_{ah} = \mathcal{H}_{h,s}\left(\boldsymbol{z}_{ah,i}^{t}; \boldsymbol{hs}_{sh,i}^{t-1}\right), \tag{3}$$

where $\mathcal{H}_{h,s}(\cdot)$ denotes the mapping function of hyperprior synthesis network, and $\boldsymbol{hs}_{sh,i}^{t-1}$ represents the parameters of the synthesis network.

Through this process, the distribution of the features at the head–body interface is estimated, enabling the computation of the Shannon cross-entropy [32] as follows:

$$H_{i}^{h} = \mathbb{E}_{\boldsymbol{x_i} \sim p_{\boldsymbol{x_i}}}\left[-\log_2 \; p_{\boldsymbol{a}_{h,i}^{t}}\left(\boldsymbol{a}_{h,i}^{t} | \boldsymbol{\mu}_{ah}, \boldsymbol{\sigma}_{ah}\right)\right], \tag{4}$$

where $p_{\boldsymbol{a}_{h,i}^{t}}$ denotes the probability model predicted by the hyperprior network at the head–body interface, and $\mathbb{E}_{\boldsymbol{x_i} \sim p_{\boldsymbol{x_i}}}[\cdot]$ represents the expectation over the input data distribution.

Similarly, at the body–tail interface, the entropy estimation mechanism with an identical architecture is applied. Specifically, the features at the body–tail interface $\boldsymbol{a}_{t,i}^{t}$ are compressed into a latent representation $\boldsymbol{z}_{at,i}^{t}$, and subsequently estimate their distribution parameters. The corresponding formulations are given by

$$\boldsymbol{z}_{at,i}^{t} = \mathcal{H}_{t,a}\left(\boldsymbol{a}_{t,i}^{t}; \boldsymbol{ha}_{st,i}^{t-1}\right), \tag{5}$$

$$\boldsymbol{\mu}_{at}, \boldsymbol{\sigma}_{at} = \mathcal{H}_{t,s}\left(\boldsymbol{z}_{at,i}^{t}; \boldsymbol{hs}_{st,i}^{t-1}\right), \tag{6}$$

where $\mathcal{H}_{t,a}(\cdot)$ and $\mathcal{H}_{t,s}(\cdot)$ denote the hyperprior analysis and synthesis network parameterized by $\boldsymbol{ha}_{st,i}^{t-1}$ and $\boldsymbol{hs}_{st,i}^{t-1}$, respectively. Consequently, the entropy at the body–tail interface is derived as

$$H_{i}^{t} = \mathbb{E}_{\boldsymbol{x_i} \sim p_{\boldsymbol{x_i}}}\left[-\log_2 \; p_{\boldsymbol{a}_{t,i}^{t}}\left(\boldsymbol{a}_{t,i}^{t} | \boldsymbol{\mu}_{at}, \boldsymbol{\sigma}_{at}\right)\right], \tag{7}$$

where $p_{\boldsymbol{a}_{t,i}^{t}}$ represents the probability model predicted by the hyperprior network at the body–tail interface.

4.3. Joint Optimization Objective

To achieve high-performance distributed learning under bandwidth constraints, the CoF U-SFL framework is trained using a joint rate-distortion optimization strategy. The objective function is designed to minimize the system's end-to-end distortion while strictly constraining the entropy of the intermediate features transmitted over the wireless channel.

Formally, the total loss function of proposed CoF U-SFL $\mathcal{L}_{\text{CoF}}$ is formulated as a weighted sum of the distortion loss and the entropy estimates:

$$\mathcal{L}_{\text{CoF}} = \underbrace{d(\boldsymbol{x}, \boldsymbol{x}')}_{\text{Distortion}} + \underbrace{\lambda_h H_i^h + \lambda_t H_i^t}_{\text{Rate}}, \tag{8}$$

where $d(\boldsymbol{x}, \boldsymbol{x}')$ denotes the mean squared error (MSE) quantifying the reconstruction loss between the input $\boldsymbol{x}$ and the output $\boldsymbol{x}'$. The terms H_i^h and H_i^t represent the cross-entropy of the feature distributions at the head–body and body–tail interfaces, respectively. The coefficients λ_h and λ_t serve as Lagrange multipliers that regulate the trade-off between reconstruction fidelity and communication efficiency.

This formulation effectively compels the U-SFL networks to learn compact feature representations with low spatial redundancy, thereby minimizing the transmission latency without compromising the overall model performance.

It is worth noting that directly optimizing the joint loss function $\mathcal{L}_{\text{CoF}}$ from the beginning of training may lead to instability, as the network struggles to balance reconstruction quality and entropy constraints simultaneously. To mitigate this, we adopt a two-stage training strategy. In the warm-up phase, we set $\lambda_h = \lambda_t = 0$ and optimize the network solely using the distortion loss $d(\boldsymbol{x}, \boldsymbol{x}')$. This allows the head, body, and tail networks to establish a stable feature representation and achieve a baseline reconstruction quality. Once the distortion loss stabilizes, we transition to the second phase by activating the entropy constraints by setting $\lambda_h, \lambda_t > 0$ to optimize the full objective $\mathcal{L}_{\text{CoF}}$. This strategy ensures robust convergence and prevents the model from collapsing into low-quality solutions driven by excessive rate penalties.

4.4. Training Procedure

The training procedure of the proposed CoF U-SFL framework is organized into three distinct phases. These phases are (a) *image coding network training*, which handles the end-to-end forward and backward propagation for the head, body, and tail networks; (b) *entropy network training*, which involves the forward and backward propagation for the head and tail entropy networks, responsible for estimating the feature entropy at the head–body and body–tail interfaces, respectively; and (c) *global model aggregation*, performed periodically every I rounds to synchronize the head network, tail network, and the personalized part of the body network. Let t denote the current training round. The detailed workflow is described as follows.

(a) Image Coding Network Training: In this phase, the client and edge server collaboratively update the image coding parameters ($\boldsymbol{w}_{ch}, \boldsymbol{w}_{sb}, \boldsymbol{w}_{ct}$) through a sequence of forward propagation (FP) and backward propagation (BP), corresponding to steps a1–a5 and a6–a10 in Figure 2, respectively. Specifically, in training round t, client i employs the local head network to extract the feature map $\boldsymbol{a}_{h,i}^t$ at the head–body interface:

$$\boldsymbol{a}_{h,i}^t = \mathcal{H}\left(\boldsymbol{x}_i^t; \boldsymbol{w}_{ch,i}^{t-1}\right), \tag{9}$$

where $\mathcal{H}(\cdot)$ represents the head network mapping function, and $\boldsymbol{w}_{ch,i}^{t-1}$ denotes the corresponding model parameters from the previous round. Subsequently, the extracted feature map $\boldsymbol{a}_{h,i}^t$ is transmitted to the edge server. Modeling the wireless channel as an AWGN channel, the received signal at the server is expressed as

$$\boldsymbol{a}_{h,i}^{t\ \prime} = \boldsymbol{a}_{h,i}^t + \boldsymbol{n}, \tag{10}$$

where $\boldsymbol{n}$ represents the noise of the channel, with its elements following a distribution $\mathcal{N}(0, \sigma^2)$. Upon reception, the edge server executes the body network FP to extract the feature map $\boldsymbol{a}_{t,i}^t$ at the body–tail split point:

$$\boldsymbol{a}_{t,i}^t = \mathcal{B}\left(\boldsymbol{a}_{h,i}^{t\,\prime}; \boldsymbol{w}_{sb,i}^{t-1}\right), \tag{11}$$

where $\mathcal{B}(\cdot)$ represents the body network mapping, and $\boldsymbol{w}_{sb,i}^{t-1}$ denotes the corresponding body network parameters from the previous round. The server then transmits $\boldsymbol{a}_{t,i}^t$ back to the client over the AWGN channel, resulting in the received tail input $\boldsymbol{a}_{t,i}^{t\,\prime} = \boldsymbol{a}_{t,i}^t + n$. Finally, the client reconstructs the image via the FP of the local tail network:

$$\hat{\boldsymbol{x}}_i^t = \mathcal{T}\left(\boldsymbol{a}_{t,i}^{t\,\prime}; \boldsymbol{w}_{ct,i}^{t-1}\right), \tag{12}$$

where $\mathcal{T}(\cdot)$ denotes the tail network mapping, $\boldsymbol{w}_{ct,i}^{t-1}$ denotes the corresponding tail network parameter, and $\hat{\boldsymbol{x}}_i^t$ represents the reconstructed image.

Following the loss calculation, gradients are back-propagated to update the sub-networks in reverse order. First, the client i independently updates the tail network parameters $\boldsymbol{w}_{ct,i}$ via gradient descent:

$$\boldsymbol{w}_{ct,i}^t \leftarrow \boldsymbol{w}_{ct,i}^{t-1} - \gamma \nabla_{\boldsymbol{w}_{ct}} F_i\left(\boldsymbol{w}_{ct,i}^{t-1}; \xi_i^t\right), \tag{13}$$

where γ is the learning rate, and $\nabla_{\boldsymbol{w}} F(\cdot)$ denotes the parameter gradient. $\xi^t \triangleq \left[\xi_i^\tau\right]_{i\in\{1,2,\ldots,N\},\tau\in\{1,\ldots,t\}}$ represents all the randomness. The gradients of the activations are then transmitted back to the server. Next, the server updates the body network. As defined in Section 4.1, the body network $\boldsymbol{w}_{sb}$ comprises a shared part $\boldsymbol{\theta}_{sb}$ and a personalized part $\boldsymbol{\theta}_{m,i}$. The *shared part* aggregates gradients from all N clients to emulate centralized training:

$$\boldsymbol{\theta}_{sb}^t = \frac{1}{N}\sum_{i=1}^{N}\left(\boldsymbol{\theta}_{sb,i}^{t-1} - \gamma \nabla_{\boldsymbol{\theta}_{sb}} F_i\left(\boldsymbol{\theta}_{sb,i}^{t-1}; \xi_i^t\right)\right), \tag{14}$$

where $\nabla_{\boldsymbol{\theta}_{sb}} F_i\left(\boldsymbol{\theta}_{sb,i}^{t-1}; \xi_i^t\right)$ represents the gradients of the *shared part* of the body network. The *personalized part*, which handles client-specific heterogeneity, is updated independently for each client i:

$$\boldsymbol{\theta}_{m,i}^t \leftarrow \boldsymbol{\theta}_{m,i}^{t-1} - \gamma \nabla_{\boldsymbol{\theta}_m} F_i\left(\boldsymbol{\theta}_{m,i}^{t-1}; \xi_i^t\right), \tag{15}$$

where $\nabla_{\boldsymbol{\theta}_m} F_i(\cdot)$ represents the gradient with respect to the *personalized part* of body network parameters for client i. Finally, the edge server returns the gradients to the clients, enabling the final update of the client-side head network:

$$\boldsymbol{w}_{ch,i}^t \leftarrow \boldsymbol{w}_{ch,i}^{t-1} - \gamma \nabla_{\boldsymbol{w}_{ch}} F_i\left(\boldsymbol{w}_{ch,i}^{t-1}; \xi_i^t\right). \tag{16}$$

(b) *Entropy network training*: This phase focuses on optimizing the hyperprior networks to accurately estimate the probability distributions of the features. First, the server performs the forward propagation of the entropy estimation network to calculate the head-side entropy H_i^h, as defined in Equations (2)–(4). Subsequently, at the head–body interface, the server updates the parameters of the hyperprior analysis and synthesis networks, denoted as $\boldsymbol{ha}_{sh,i}^{t-1}$ and $\boldsymbol{hs}_{sh,i}^{t-1}$, to minimize the total loss $\mathcal{L}_{\text{CoF}}$. This ensures that the probability model $p_{\boldsymbol{a}_{h,i}^t}$ closely fits the distribution of the features at the head–body interface. The update rule is formulated as

$$\boldsymbol{ha}_{sh,i}^t \leftarrow \boldsymbol{ha}_{sh,i}^{t-1} - \eta \nabla_{\boldsymbol{ha}_{sh,i}} \mathcal{L}_{\text{CoF}}, \tag{17}$$

$$\boldsymbol{hs}_{sh,i}^{t} \leftarrow \boldsymbol{hs}_{sh,i}^{t-1} - \eta \nabla_{\boldsymbol{hs}_{sh,i}} \mathcal{L}_{\mathrm{CoF}}, \tag{18}$$

where η denotes the learning rate for the entropy estimation networks.

Similarly, at the body–tail interface, the server first performs the forward propagation to estimate the entropy H_i^t according to Equations (5)–(7). Then, the parameters $\boldsymbol{ha}_{st,i}^{t-1}$ and $\boldsymbol{hs}_{st,i}^{t-1}$ are optimized to minimize the total loss $\mathcal{L}_{\mathrm{CoF}}$, ensuring accurate modeling of the feature distribution:

$$\boldsymbol{ha}_{st,i}^{t} \leftarrow \boldsymbol{ha}_{st,i}^{t-1} - \eta \nabla_{\boldsymbol{ha}_{st,i}} \mathcal{L}_{\mathrm{CoF}}, \tag{19}$$

$$\boldsymbol{hs}_{st,i}^{t} \leftarrow \boldsymbol{hs}_{st,i}^{t-1} - \eta \nabla_{\boldsymbol{hs}_{st,i}} \mathcal{L}_{\mathrm{CoF}}. \tag{20}$$

By continuously updating these parameters, the entropy estimation mechanism adapts to the dynamic feature representations driven by the joint optimization objective.

This phase constitutes the core of the optimization process, where the entropy constraints are fully activated (i.e., $\lambda_h, \lambda_t > 0$). Unlike the warm-up phase which focuses solely on distortion, this phase employs a joint optimization strategy to simultaneously minimize both the reconstruction distortion and the feature transmission rate. In this stage, the gradients derived from the total loss $\mathcal{L}_{\mathrm{CoF}}$ are back-propagated through the entire framework.

(c) Global model aggregation: To leverage collaborative knowledge, the global model aggregation phase is executed at the Fed server every I training rounds, corresponding to steps c1–c3 in Figure 2. First, the clients and the edge server upload their respective sub-models, including the head network $\boldsymbol{w}_{ch}$, the tail network $\boldsymbol{w}_{ct}$, and the server-side personalized body network $\boldsymbol{\theta}_m$. Subsequently, the Fed server performs uniform averaging to synchronize these components:

$$\boldsymbol{w}_{ch}^{t} = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{w}_{ch,i}^{t}, \tag{21}$$

$$\boldsymbol{w}_{ct}^{t} = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{w}_{ct,i}^{t}, \tag{22}$$

$$\boldsymbol{\theta}_{m}^{t} = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{\theta}_{m,i}^{t}. \tag{23}$$

After aggregation, the synchronized parameters are distributed back to the participants for the next round of training.

We adopt a collaborative client-server strategy to train the CoF U-SFL framework. Specifically, clients are responsible for executing the head and tail networks utilizing their local private data. Conversely, the server orchestrates the operations of the body network and the entropy estimation modules, while also managing the global model aggregation. The comprehensive workflow is outlined in Algorithm 1.

Algorithm 1 The training process of CoF U-SFL

Require: b, γ, E, $\mathcal{N}$, and $\mathcal{D}$.
Ensure: w^*.
Initialization: w^0, $w_i^0 \leftarrow w^0$, $I^0 \leftarrow 1$, $\tau \leftarrow 0$ and $\rho \leftarrow 0$.
for $t = \rho + 1, \rho + 2, \ldots, \rho + I^\tau$ **do**
  / Runs on clients **/**
  for each client in parallel **do**
    $\boldsymbol{a}_{h,i}^t \leftarrow \mathcal{H}\left(\boldsymbol{x}_i^t; \boldsymbol{w}_{ch,i}^{t-1}\right)$
    Send $\boldsymbol{a}_{h,i}^t$ to the edge server.
  end for
  / Runs on edge server **/**
  $\boldsymbol{a}_{t,i}^t \leftarrow \mathcal{B}\left(\boldsymbol{a}_{h,i}^t; \boldsymbol{w}_{sb,i}^{t-1}\right)$
  Send $\boldsymbol{a}_{t,i}^t$ to corresponding clients.
  FP of entropy estimation network.
  Obtain the feature entropy H_i^h and H_i^t using Equations (4) and (7).
  / Runs on clients **/**
  for each client in parallel **do**
    $\hat{\boldsymbol{x}}_i^t \leftarrow \mathcal{T}\left(\boldsymbol{a}_{t,i}^t; \boldsymbol{w}_{ct,i}^{t-1}\right)$
    Calculate loss function value:
    $\mathcal{L}_{\text{CoF}} = d(\boldsymbol{x}, \boldsymbol{x}') + \lambda_h H_i^h + \lambda_t H_i^t$
    BP and send gradients to the edge server.
  end for
  / Runs on edge server **/**
  $\boldsymbol{\theta}_{sb,i}^t \leftarrow \boldsymbol{\theta}_{sb,i}^{t-1} - \gamma \nabla_{\boldsymbol{\theta}_{sb}} F_i\left(\boldsymbol{\theta}_{sb,i}^{t-1}; \xi_i^t\right)$
  for each client in parallel **do**
    $\boldsymbol{\theta}_{m,i}^t \leftarrow \boldsymbol{\theta}_{m,i}^{t-1} - \gamma \nabla_{\boldsymbol{\theta}_m} F_i\left(\boldsymbol{\theta}_{m,i}^{t-1}; \xi_i^t\right)$
    Send gradients to corresponding clients.
    BP of entropy estimation network.
  end for
  / Runs on clients **/**
  for each client in parallel **do**
    $\boldsymbol{w}_{ch,i}^t \leftarrow \boldsymbol{w}_{ch,i}^{t-1} - \gamma \nabla_{\boldsymbol{w}_{ch}} F_i\left(\boldsymbol{w}_{ch,i}^{t-1}; \xi_i^t\right)$
  end for
end for
Runs on the Fed server:
if $(t - \rho) \mod I^\tau = 0$ **then**
  $\boldsymbol{w}_{ch}^t \leftarrow \frac{1}{N} \sum_{i=1}^N \boldsymbol{w}_{ch,i}^t$
  $\boldsymbol{w}_{ct}^t \leftarrow \frac{1}{N} \sum_{i=1}^N \boldsymbol{w}_{ct,i}^t$
  $\boldsymbol{\theta}_m^t \leftarrow \frac{1}{N} \sum_{i=1}^N \boldsymbol{\theta}_{m,i}^t$
  $\boldsymbol{w}_{ch,i}^t \leftarrow \boldsymbol{w}_{ch}^t$
  $\boldsymbol{w}_{ct,i}^t \leftarrow \boldsymbol{w}_{ct}^t$
  $\boldsymbol{\theta}_{m,i}^t \leftarrow \boldsymbol{\theta}_m^t$
  Update $\rho \leftarrow \rho + I^\tau$.
  Update $\tau \leftarrow \tau + 1$.
end if

5. Performance Evaluation

We evaluate the proposed CoF U-SFL framework across two dimensions: communication efficiency and reconstruction fidelity. Communication efficiency is quantified by the total volume of feature data transmitted between the client and server during training, reflecting the reduction in overhead achieved through feature compression. Reconstruction

fidelity is assessed using the peak signal-to-noise ratio (PSNR) on test datasets, ensuring that reduced communication does not compromise image quality.

5.1. Implementation Details

We adopt the SOTA Wireless Image Transmission Transformer (WITT) [33] as the backbone. Specifically, this backbone has a total model size of approximately 107.43 MB (with 53.72 MB for the encoder and 53.71 MB for the decoder). To align with the U-SFL topology, we partitioned the architecture into the head, body, and tail networks.

For the feature entropy estimation module, we construct a dual-layer Transformer-based hyperprior network to capture global dependencies in the feature distribution. This module introduces an additional parameter overhead of 58.69 MB.

Furthermore, the intermediate feature sizes and model allocations vary depending on the selected split points within the U-shaped architecture. Specifically, partitioning at the first and second split points yields intermediate feature dimensions of [1, 16,384, 128] (8 MB data volume) and $[1, 4096, 192]$ (3 MB data volume), corresponding to client-side head network sizes of 3.05 MB and 10.61 MB, respectively.

All models were implemented in Python 3.7 using PyTorch 1.9.1 and trained on an NVIDIA GeForce RTX 4090 GPU. We used the Adam optimizer with an initial learning rate of 1×10^{-4} and a mini-batch size of 16. To ensure stable convergence, we employed a two-stage training strategy. In the initial warm-up phase, we set the entropy weights $\lambda_h = \lambda_t = 0$ to optimize solely for distortion. Once stabilized, we activated the entropy constraints. Since the estimated entropy magnitude is significantly larger than the distortion loss, we empirically set λ_h and λ_t to the order of 10^{-5} to balance the gradient scales. The training process spanned 70 epochs, with uplink and downlink transmission emulated within a JSCC framework at a fixed SNR of 15 dB.

We used the DIV2K dataset [34] for training and the Kodak [35] and CLIC2021 [36] test sets for evaluation. For training, random cropping was applied to generate 256×256 patches.

5.2. Performance Evaluation and Analysis

We compare the proposed CoF U-SFL framework against several representative baselines as follows:

- **CL:** This method represents the ideal scenario where all client data are collected and trained on a central server using a unified WITT model. It establishes the performance upper bound for reconstruction quality, although it fundamentally violates data privacy constraints.
- **FL:** We employ the standard FedAvg algorithm, where clients train complete WITT models locally and the server aggregates model weights every 6 local epochs. This represents the conventional privacy-preserving approach that relies on full client-side computation.
- **U-SFL (WITT):** This baseline implements the standard U-SFL framework with the SOTA WITT (built on the Swin Transformer architecture) [33]. For a fair comparison, it maintains strictly consistent experimental settings with our CoF U-SFL, including the identical WITT-based network architecture, specific split points, and global aggregation frequency. This baseline provides a direct comparison to isolate and quantify the performance gains attributed to our proposed method.
- **U-SFL-FCA:** In this baseline, we implement the feature compression via an autoencoder (FCA) mechanism [19,26] within the U-SFL framework. This method compresses features at the split point through dimensionality reduction. While this approach effec-

tively reduces communication overhead, the inevitable feature distortion introduced during the compression process results in irreversible information loss.

It is worth noting that in the aforementioned U-SFL and CoF U-SFL, the transmission over physical channels inherently provides an implicit regularization effect, and the channel noise acts as a stochastic constraint that prevents the network from overfitting to ideal communication links. Consequently, during back-propagation, the gradients derived from the noise-corrupted features force the model to discard noise-sensitive details, naturally driving the extraction of channel-robust feature representations.

As summarized in Table 1, CL naturally establishes the performance upper bound. The communication overhead of FL is highly sensitive to aggregation frequency. Aggregating every epoch results in a volume of 7.5 GB, whereas extending the interval to 6 epochs reduces this to 1.18 GB with a marginal PSNR drop of approximately 0.1 dB. However, FL imposes a high computational load on clients, requiring full forward and backward propagation of the WITT model. In contrast, standard U-SFL offloads the majority of computational burdens to the server, requiring clients to execute only the lightweight head and tail components. Specifically, when the split point is set at the first layer, the client-side model size is drastically reduced from 107.43 MB (in FL) to merely 6 MB. While this successfully redistributes computational pressure, the fundamental limitation of this framework lies in the transmission mechanism: the iterative bidirectional transfer of high-dimensional features and gradients generates excessive communication overhead. With feature sizes reaching up to 8 MB, the communication volume for features can escalate to 13.67 GB, and the total communication volume is 28.52 GB, creating a critical bottleneck that hinders practical deployment in bandwidth-limited scenarios.

Table 1. Comparison of performance and communication overhead across different distributed learning methods on the CLIC 2021 test set.

Method	Aggregation	PSNR (dB)	Feature Vol. (GB)	Total Comm. (GB)	Entropy of Feature
CL	–	31.0	–	–	–
FL (Every Round)	Every Round	30.6	–	7.50	–
FL (6 Rounds)	6 Rounds	30.5	–	1.18	–
U-SFL (WITT)	6 Rounds	30.1	13.67	28.52	1360
U-SFL-FCA	6 Rounds	29.3	**6.84**	21.69	–
CoF U-SFL (Ours)	6 Rounds	**30.1**	**0.12**	**14.97**	**13**

As shown in Table 1, the proposed CoF U-SFL framework introduces a compact feature learning mechanism that maintains nearly identical rate-distortion performance to the vanilla U-SFL while significantly improving communication efficiency. Specifically, feature communication volume drops from 13.67 GB to 0.12 GB, a 104.6× reduction. Correspondingly, total communication decreases from 28.52 GB to 14.97 GB. This approach mitigates the communication bottleneck while preserving the low client-side computational load and privacy benefits of U-SFL, making it a viable solution for resource-constrained edge environments.

Additionally, U-SFL-FCA employs an autoencoder to compress the intermediate features. Although this strategy reduces the feature communication volume to 6.84 GB, the experimental results exhibit a trade-off in representation capability. Unlike our entropy-guided approach, the fixed compression mechanism in U-SFL-FCA lacks the adaptability to distinguish between redundancy and task-critical information. Consequently, U-SFL-FCA achieves a PSNR of only 29.3 dB compared to 30.1 dB for our method.

To investigate the mechanism behind this communication efficiency, we analyzed the entropy of features at the split point. Feature entropy serves as a fundamental metric for

information redundancy, providing a theoretical basis for the efficiency gains. As shown in Table 1, by systematically comparing the feature entropy values of different methods, we reveal the essential reason behind the communication optimization achieved by the proposed CoF U-SFL. Experimental results indicate that CoF U-SFL reduces feature entropy by two orders of magnitude, from approximately 1360 in vanilla U-SFL to about 13. This proves that our proposed compact feature learning mechanism effectively eliminates information redundancy in the feature representation, laying a theoretical foundation for the improvement in communication efficiency. The entropy estimation network effectively guides the model to eliminate redundancy and learn a feature space with higher information density.

As shown in Table 2, we comprehensively evaluated the proposed CoF U-SFL against the vanilla U-SFL baseline across three datasets with varying resolutions: CLIC 2021 (high), MS-COCO (medium), and CIFAR-10 (extremely small). The experimental results clearly demonstrate that our method is highly effective across different spatial scales, successfully achieving massive reductions in both feature volume and feature entropy while maintaining comparable image reconstruction quality (PSNR). Specifically, for the CLIC 2021 and MS-COCO datasets, since the input images are uniformly processed into 256×256 patches in our experimental setup, the intermediate feature volumes generated by the vanilla U-SFL are inherently identical (13.67 GB). By applying our feature entropy estimation mechanism, CoF U-SFL drastically compresses these volumes down to 0.12 GB and 0.22 GB, respectively. Furthermore, for extremely small-scale images like CIFAR-10 (32×32), the intrinsic data dimension is much smaller than the standard patch size, leading to a naturally smaller baseline feature volume (1.71 GB for U-SFL). Because such low-resolution images inherently contain less spatial redundancy, the theoretical room for compression is relatively restricted compared to high-resolution datasets. However, the results confirm that even under these constrained conditions, CoF U-SFL still effectively compresses the feature volume to 0.057 GB and drops the feature entropy from 1340 to 45.

Table 2. Comparison of PSNR, feature volume, and feature entropy across datasets of varying resolutions.

Dataset	Method	PSNR (dB)	Feature Vol. (GB)	Entropy of Feature
CLIC 2021	U-SFL (WITT)	30.1	13.67	1360
	CoF U-SFL (ours)	30.1	0.12	13
MS-COCO	U-SFL (WITT)	31.2	13.67	1340
	CoF U-SFL (ours)	31.1	0.22	22
CIFAR-10	U-SFL (WITT)	29.8	1.71	1340
	CoF U-SFL (ours)	29.9	0.0574	45

As illustrated in Figure 4, part (a) displays the original input image. The feature maps generated by the vanilla U-SFL (b,d) preserve distinct spatial structures and perceptible contours, indicating strong spatial correlation at both split interfaces. In contrast, CoF U-SFL features (c,e) manifest as highly abstract patterns where geometric details are significantly obfuscated. This suppression of spatial distinctness provides intuitive evidence that our method successfully reduces non-task-critical visual redundancy. Consequently, it encodes the data into a compact, information-dense latent representation, thereby effectively minimizing the feature entropy required for transmission.

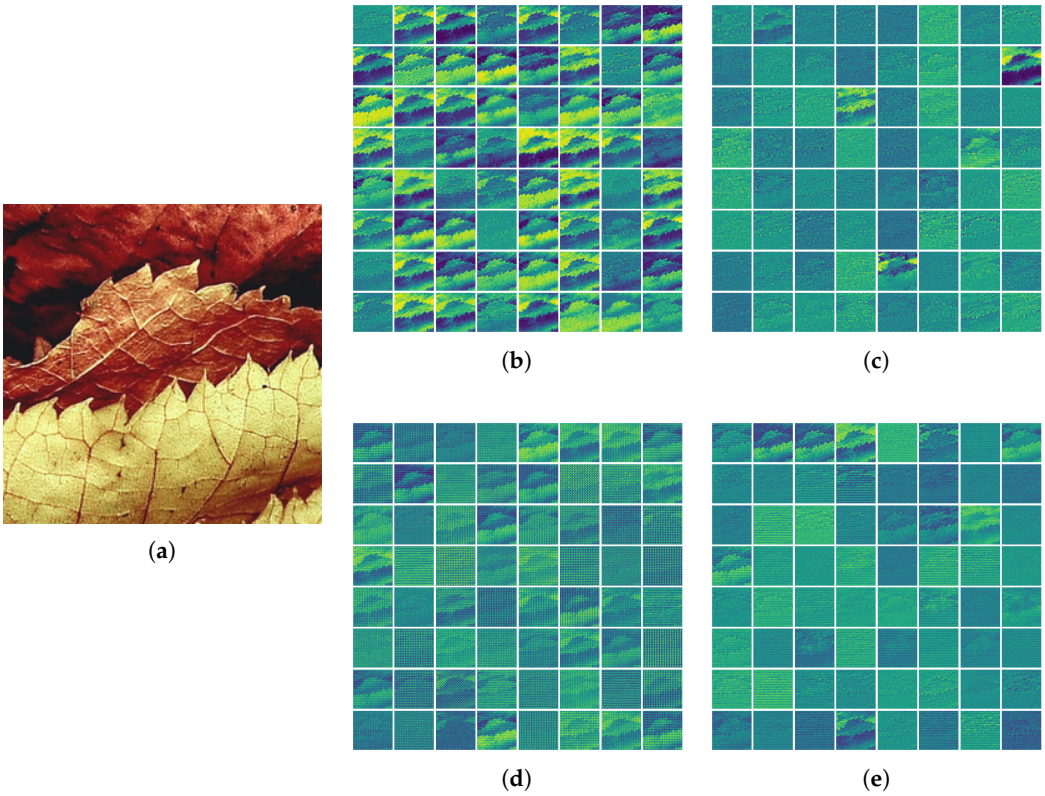


Figure 4. Visual comparison of feature maps. (**a**) The original input image. (**b**) Feature map of U-SFL (WITT) at the head–body interface. (**c**) Feature map of CoF U-SFL at the head–body interface. (**d**) Feature map of U-SFL (WITT) at the body–tail interface. (**e**) Feature map of CoF U-SFL at the body–tail interface. Compared to the contour-heavy U-SFL (WITT) features (**b**,**d**), the CoF U-SFL features (**c**,**e**) exhibit highly abstract patterns, indicating reduced spatial redundancy.

To further evaluate the robustness and adaptability of the proposed CoF U-SFL framework, we further investigated the impact of split-point selection. Specifically, we conducted comparative experiments using two representative split points defined within the WITT architecture:

- **Split Point 1 (Shallow Split):** Positioned after the first Transformer encoder block. As detailed in IV.A, while the client-side model is lightweight, the feature maps at this shallow stage preserve rich low-level spatial details and high dimensionality. Consequently, this results in greater communication overhead.
- **Split Point 2 (Mid Split):** Positioned after the second Transformer encoder block. The features have undergone deeper transformations, capturing more abstract information with reduced spatial resolution. This significantly alleviates the communication overhead, albeit at the cost of deploying a larger head network on the edge client.

Table 3 presents a comparative analysis of performance and communication efficiency across different split strategies on the Kodak dataset. To ensure a fair comparison and isolate the impact of the split point, the server aggregation interval was fixed at every 6 rounds for all experiments. First, regarding the vanilla U-SFL, the communication overhead is strictly dictated by the split depth. The shallower split point 1 outputs high-dimensional feature maps, inevitably resulting in a much heavier transmission load. As the network deepens to split point 2, the intrinsic feature entropy naturally decreases from 1360 to 463, causing the total communication volume to drop from 28.52 GB to 12.04 GB.

Table 3. Performance and efficiency comparison under different model split points.

Split Point	Method	PSNR (dB)	Communication (GB)	Feature Vol. (GB)	Feature Entropy	Compression Ratio of Feature
Split Point 1	U-SFL (WITT)	30.1	28.52	13.67	1360	1.0x
	CoF U-SFL (Ours)	**30.2**	**14.97**	**0.12**	**13**	**104.6x**
Split Point 2	U-SFL (WITT)	30.3	12.04	5.12	463	1.0x
	CoF U-SFL (Ours)	**30.3**	**7.02**	**0.10**	**9**	**51.4x**

Crucially, CoF U-SFL exhibits a distinct trend where higher initial redundancy correlates with greater compression gains. At split point 1, where intermediate features possess high dimensionality and redundancy, our method achieves its most significant efficiency improvement. Specifically, by reducing feature entropy from 1360 to 13, the feature communication volume drops from 13.67 GB to 0.12 GB, achieving a 104.6× compression ratio. Similarly, at split point 2, where the vanilla feature volume decreases to 5.12 GB, our mechanism further compresses the data to 0.10 GB. This comparison demonstrates the generality of our approach. Regardless of the initial feature dimension at the split point, CoF U-SFL effectively guides the model to learn a compact, information-dense feature representation. In particular, the superior compression capability at shallow network layers makes our method especially suitable for resource-constrained edge devices facing prohibitive transmission costs.

Furthermore, regarding model performance, CoF U-SFL matches the PSNR values of vanilla U-SFL at both split points. This confirms that our communication optimization is achieved by eliminating informational redundancy within the features rather than at the expense of accuracy. Notably, the reconstruction performance at split point 1 is marginally higher than at split point 2, which is attributed to the server undertaking a heavier computational load. This finding highlights the inherent trade-off between client-side computational cost and model performance, allowing for flexible selection based on available resources. Consequently, these results provide strong evidence that the CoF U-SFL framework is not a specialized optimization tailored to a specific model structure, but rather a general solution for enhancing communication efficiency. It can flexibly adapt to diverse system design requirements, whether the priority lies in alleviating client-side computational burden or minimizing communication overhead.

6. Conclusions

We proposed CoF U-SFL, a collaborative image coding framework designed for U-shaped split federated learning. By integrating a dual-layer Transformer-based hyperprior network, CoF U-SFL models the statistical distribution of split-layer features to estimate entropy without accessing raw data. We established a stable end-to-end training pipeline through a joint optimization objective that balances entropy constraints with distortion metrics. Experimental results indicate that CoF U-SFL reduces feature communication overhead by 104.6× while maintaining comparable image reconstruction quality. This approach effectively balances privacy preservation, communication efficiency, and image fidelity within distributed learning environments. Future research will investigate defenses against reconstruction attacks and extend the framework to video coding applications in high-concurrency edge systems.

Author Contributions: Conceptualization, Q.S.; Methodology, Q.S.; Software, Q.S.; Validation, Q.S.; Formal analysis, Q.S.; Investigation, Q.S.; Data curation, Q.S.; Writing—original draft, Q.S.; Writing—review, editing, Q.S.; Visualization, Q.S.; Supervision, C.G., M.Z. and Y.Y.; Project adminis-

tration, C.G. and Y.Y.; Funding acquisition, C.G. and Y.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Natural Science Foundation of China (No. 62371070), in part by the Beijing Natural Science Foundation (No. L222043), and in part by Beijing University of Posts and Telecommunications Excellent Ph.D. Students Foundation (CX2021101).

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Hu, S.; Fang, Z.; Deng, Y.; Chen, X.; Fang, Y. Collaborative perception for connected and autonomous driving: Challenges, possible solutions and opportunities. *IEEE Wirel. Commun.* **2025**, *32*, 228–234. [CrossRef]
2. Gafni, T.; Shlezinger, N.; Cohen, K.; Eldar, Y.C.; Poor, H.V. Federated learning: A signal processing perspective. *IEEE Signal Process. Mag.* **2022**, *39*, 14–41. [CrossRef]
3. Feng, C.-M.; Yan, Y.; Wang, S.; Xu, Y.; Shao, L.; Fu, H. Specificity-preserving federated learning for MR image reconstruction. *IEEE Trans. Med. Imaging* **2022**, *42*, 2010–2021. [CrossRef]
4. Zhang, H.; Hanzo, L. Federated learning assisted multi-UAV networks. *IEEE Trans. Veh. Technol.* **2020**, *69*, 14104–14109. [CrossRef]
5. Vepakomma, P.; Gupta, O.; Swedish, T.; Raskar, R. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv* **2018**, arXiv:1812.00564. [CrossRef]
6. Thapa, C.; Arachchige, P.C.M.; Camtepe, S.; Sun, L. SplitFed: When federated learning meets split learning. *Proc. AAAI Conf. Artif. Intell. (AAAI)* **2022**, *36*, 8485–8493. [CrossRef]
7. Wu, M.; Cheng, G.; Ye, D.; Kang, J.; Yu, R.; Wu, Y.; Pan, M. Federated split learning with data and label privacy preservation in vehicular networks. *IEEE Trans. Veh. Technol.* **2023**, *73*, 1223–1238. [CrossRef]
8. Zheng, J.; Chen, Y.; Lai, Q. PPSFL: Privacy-preserving split federated learning for heterogeneous data in edge-based internet of things. *Future Gener. Comput. Syst.* **2024**, *156*, 231–241. [CrossRef]
9. Lin, Z.; Qu, G.; Wei, W.; Chen, X.; Leung, K.K. AdaptSFL: Adaptive split federated learning in resource-constrained edge networks. *arXiv* **2024**, arXiv:2403.13101. [CrossRef]
10. Nan, G.; Liu, X.; Lyu, X.; Cui, Q.; Xu, X.; Zhang, P. UDSem: A unified distributed learning framework for semantic communications over wireless networks. *IEEE Netw.* **2023**, *38*, 296–303. [CrossRef]
11. Zheng, G.; Ni, Q.; Navaie, K.; Pervaiz, H.; Min, G.; Kaushik, A.; Zarakovitis, C. Mobility-aware split-federated with transfer learning for vehicular semantic communication networks. *IEEE Internet Things J.* **2024**, *11*, 17237–17248. [CrossRef]
12. Yang, Z.; Chen, Y.; Huangfu, H.; Ran, M.; Wang, H.; Li, X.; Zhang, Y. Dynamic corrected split federated learning with homomorphic encryption for u-shaped medical image networks. *IEEE J. Biomed. Health Inform.* **2023**, *27*, 5946–5957. [CrossRef]
13. Wang, Z.; Yang, G.; Dai, H.; Rong, C. Privacy-preserving split learning for large-scaled vision pre-training. *IEEE Trans. Inf. Forensics Secur.* **2023**, *18*, 1539–1553. [CrossRef]
14. Kathariya, B.; Li, Z.; Chen, J.; Van der Auwera, G. Gradient compression with a variational coding scheme for federated learning. In *2021 International Conference on Visual Communications and Image Processing (VCIP)*; IEEE: Piscataway, NJ, USA, 2021; pp. 1–5.
15. Cui, L.; Su, X.; Zhou, Y.; Zhang, L. Clustergrad: Adaptive gradient compression by clustering in federated learning. In *GLOBECOM 2020—2020 IEEE Global Communications Conference*; IEEE: Piscataway, NJ, USA, 2020; pp. 1–7.
16. Nguyen, M.-D.; Lee, S.-M.; Pham, Q.-V.; Hoang, D.T.; Nguyen, D.N.; Hwang, W.-J. Hcfl: A high compression approach for communication-efficient federated learning in very large scale iot networks. *IEEE Trans. Mob. Comput.* **2022**, *22*, 6495–6507. [CrossRef]
17. Chen, S.; Jin, T.; Xia, Y.; Li, X. Metadata and image features co-aware semi-supervised vertical federated learning with attention mechanism. *IEEE Trans. Veh. Technol.* **2023**, *73*, 2520–2532. [CrossRef]
18. Isik, B.; Pase, F.; Gunduz, D.; Weissman, T.; Zorzi, M. Sparse random networks for communication-efficient federated learning. *arXiv* **2022**, arXiv:2209.15328.
19. Ayad, A.; Renner, M.; Schmeink, A. Improving the communication and computation efficiency of split learning for iot applications. In *2021 IEEE Global Communications Conference (GLOBECOM)*; IEEE: Piscataway, NJ, USA, 2021; pp. 01–06.
20. Hsieh, C.-Y.; Chuang, Y.-C.; Wu, A.-Y. C3-sl: Circular convolution-based batch-wise compression for communication-efficient split learning. In *2022 IEEE 32nd International Workshop on Machine Learning for Signal Processing (MLSP)*; IEEE: Piscataway, NJ, USA, 2022; pp. 1–6.

21. Lin, Z.; Lin, Z.; Yang, M.; Huang, J.; Zhang, Y.; Fang, Z.; Du, X.; Chen, Z.; Zhu, S.; Ni, W. Sl-acc: A communication-efficient split learning framework with adaptive channel-wise compression. *arXiv* **2025**, arXiv:2508.12984.
22. Han, D.-J.; Bhatti, H.I.; Lee, J.; Moon, J. Accelerating federated learning with split learning on locally generated losses. *arXiv* **2021**, arXiv:2104.14380.
23. Mu, Y.; Shen, C. Federated split learning with improved communication and storage efficiency. *IEEE Trans. Mob. Comput.* **2025**, *25*, 272–283. [CrossRef]
24. Gupta, A.; Chaurasiya, V.K. Adaptive low-latency split federated learning with dynamic model partitioning in resource-constrained healthcare iot. *IEEE Trans. Green Commun. Netw.* **2025**, *10*, 1308–1321. [CrossRef]
25. Oh, Y.; Lee, J.; Brinton, C.G.; Jeon, Y.-S. Communication-efficient split learning via adaptive feature-wise compression. *IEEE Trans. Neural Netw. Learn. Syst.* **2025**, *36*, 10844–10858. [CrossRef]
26. Yu, L.; Chang, Z.; Jia, Y.; Min, G. Model partition and resource allocation for split learning in vehicular edge networks. *IEEE Trans. Intell. Transp. Syst.* **2025**, *26*, 17851–17865. [CrossRef]
27. Matsubara, Y.; Levorato, M.; Restuccia, F. Split computing and early exiting for deep learning applications: Survey and research challenges. *ACM Comput. Surv.* **2022**, *55*, 90. [CrossRef]
28. Pasquini, D.; Ateniese, G.; Bernaschi, M. Unleashing the tiger: Inference attacks on split learning. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*; Association for Computing Machinery: New York, NY, USA, 2021; pp. 2113–2129.
29. Bourtsoulatze, E.; Kurka, D.B.; Gündüz, D. Deep joint source-channel coding for wireless image transmission. *IEEE Trans. Cogn. Commun. Netw.* **2019**, *5*, 567–579. [CrossRef]
30. Theis, L.; Shi, W.; Cunningham, A.; Huszár, F. Lossy image compression with compressive autoencoders. *arXiv* **2017**, arXiv:1703.00395. [CrossRef]
31. Sun, Q.; Guo, C.; Yang, Y.; Tang, R.; Liu, C. Deep joint source-channel coding based on semantics of pixels for wireless image transmission. In *2023 IEEE 34th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*; IEEE: Piscataway, NJ, USA, 2023.
32. Balle, J.; Minnen, D.; Singh, S.; Hwang, S.J.; Johnston, N. Variational image compression with a scale hyperprior. *arXiv* **2018**, arXiv:1802.01436. [CrossRef]
33. Yang, K.; Wang, S.; Dai, J.; Tan, K.; Niu, K.; Zhang, P. WITT: A wireless image transmission transformer for semantic communications. In *ICASSP 2023—2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*; IEEE: Piscataway, NJ, USA, 2023; pp. 1–5.
34. Agustsson, E.; Timofte, R. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*; IEEE: Piscataway, NJ, USA, 2017; pp. 126–135.
35. Kodak. Kodak Photocd Dataset. 1993. Available online: http://r0k.us/graphics/kodak/ (accessed on 13 March 2026).
36. CLIC 2021: Challenge on Learned Image Compression. 2021. Available online: http://compression.cc (accessed on 13 March 2026).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.