Error-resilient Hierarchical Autoencoders for High Dimensional Data Compression in Wireless Networks

Kang Gao

Electrical and Computer Engineering, Michigan State University, East Lansing, United State, gaokang@msu.edu

Amit Kumar Bhuyan

Electrical and Computer Engineering, Michigan State University, East Lansing, United State, bhuyanam@msu.edu

Subir Biswas

Electrical and Computer Engineering, Michigan State University, East Lansing, United State, sbiswas@egr.msu.edu

This paper presents a hierarchical autoencoder framework for multi-stream sensor data compression in resource constrained wireless sensor networks (WSNs). The proposed framework aims to manage computational consumption, and bandwidth usage, while ensuring acceptable task performance despite channel errors. The proposed compression process is divided into two phases: intra-stream and inter-stream compression. In the first phase, a neural encoder compresses each individual data stream separately to eliminate intra-stream temporal redundancies, producing their compact individual representations. In the second phase, those representations of individual streams are combined and further compressed by another neural encoder to eliminate redundancies across streams and produce a single compressed latent representation for all the streams, which is sent to the receiver through a nosy channel. The neural decoder at the receiver uses a similar hierarchy for reconstructing individual original streams, and any subsequent task execution on those streams. To address the vulnerability of compressed data to channel errors, the framework integrates an error-resilient and learning-based transmission coding scheme. The proposed approach is demonstrated using a greenhouse micro-climate monitoring task involving multiple environmental sensors. Extensive simulations on real-world data validate the framework's efficacy in balancing transmission cost, energy consumption, and task accuracy, marking it a significant advancement over existing methods.

CCS CONCEPTS • Computer systems organization → Embedded and cyber-physical systems → Sensor networks• Computing methodologies → Machine learning → Machine learning approach → Neural Networks • Information System → Data compression • Information System → Data encoding and canonicalization

Additional Keywords and Phrases: Hierarchical Autoencoder, Bandwidth reduction, Error-resilient Coding, Feature compression, Transmitter Complexity, Wireless Communication, Machine Learning

ACM Reference Format:

First Author's Name, Initials, and Last Name, Second Author's Name, Initials, and Last Name, and Third Author's Name, Initials, and Last Name. 2018. The Title of the Paper: ACM Conference Proceedings Manuscript Submission Template: This is the subtitle of the paper, this document both explains and embodies the submission format for authors using Word. In Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY. ACM, New York, NY, USA, 10 pages. NOTE: This block will be automatically generated when manuscripts are processed after acceptance.

1 INTRODUCTION

This paper presents the design of a neural network based autoencoder that can be used for combining multiple sensor data streams and reducing overall data dimensionality. Such reductions can lead to network bandwidth savings and reduce the transmission energy costs, thus making it suitable for a large slew of wireless sensor network applications including precision agriculture [1], smart infrastructure [2], healthcare [3], and smart manufacturing [4]. In most cases, the corresponding networks are required to transmit multiple independent data streams that are generated by various application-specific sensors (e.g., temperature, humidity, etc. for precision farming in a greenhouse). The proposed approach is to eliminate information redundancies within and across those data streams before transmission, resulting in a lossy compression mechanism. Any information in the original data that is unnecessary for the specific downstream tasks is purged out by the autoencoder during the compression process. This not only minimizes communication overhead but also lessens the transmission and computational load on sensor nodes.

The proposed mechanism addresses another crucial aspect, which is unreliable channel and the resulting errors prevalent in sensors networks formed by low-complexity sensor transceivers. Apart from their usual shortcoming, channel errors can be particularly harmful for dimensionality-reduced data stream with high density of essential task-specific information. The problem is further compounded by the fact that computationally heavy error correction mechanisms are often not practically feasible for low-complexity sensor transceivers. This challenge is addressed in this paper by integrating an error-resilient and neural network enabled transmission coding approach that operates in tandem with the autoencoder-based compression mechanism itself. Through end-to-end training, the proposed framework learns to strategically encodes compressed information into binary codes such that the impact of channel errors is minimized.

This paper demonstrates the proposed approach in the context of greenhouse micro-climate monitoring [5], where an integrated sensor system is equipped with multiple environmental sensors collecting multiple data streams. The application-layer objective is to transmit multiple sensor data streams to a centralized collection unit. Due to strong temporal and spatial correlations within and across streams, the raw data exhibits significant redundancies. To address this, the paper introduces a hierarchical neural network-based autoencoding framework which puts an encoder at sensor node to compresses the high-dimensional, multi-stream data into a single low-dimensional representation that retains essential information. This compact stream is transmitted to the central unit, where a decoder reconstructs the original data streams. The encoder is organized hierarchically, with the initial layer capturing intra-stream temporal correlations to eliminate redundancies within individual streams. Subsequent layers focus on extracting inter-stream dependencies to further reduce data dimensionality. The decoder at the receiver contains a similar hierarchy, which is jointly trained with the encoder at the transmitters side sensor system to enable accurate reconstruction. The proposed architecture is tunable in that the trade-off between transmission bandwidth reduction via task-specific compression and the complexity of the hierarchical autoencoder can be adjusted by tuning various system parameters.

The key contributions of the paper are as follows. First, a hierarchical learning architecture is developed to compress multiple high-dimensional data streams into a single low-dimensional data stream, achieving efficient data compression. Second, a multidimensional trade-off is studied with respect to data transmission performance, bandwidth reduction, and transmitter computational complexity. Third, an innovative learning-based coding scheme is developed in order to to distribute the compressed data over communication channels with error. Finally, the proposed framework is validated using a real-world greenhouse dataset. Extensive simulation experiments with that dataset are conducted for functional validation and performance evaluation of the developed framework.

2 RELATED WORK

Recent research on data compression techniques for wireless sensor networks (WSNs) has primarily emphasized methods suitable for devices with constrained computational power, memory, and energy resources. Traditional approaches include Huffman coding and Discrete Wavelet Transform (DWT)-based frameworks, both widely applied to sensor data to mitigate redundancy and reduce bandwidth usage [11][12]. Huffman coding effectively compresses data by assigning shorter codes to frequently occurring symbols; however, its efficiency significantly diminishes when dealing with sensor data characterized by low entropy, resulting in limited compression ratios. Meanwhile, DWT-based compression techniques typically leverage frequency domain transformations to selectively discard high-frequency coefficients if such components contain less meaningful information. However, this selective removal strategy can inadvertently eliminate subtle yet crucial task-specific features, causing irreversible data loss that negatively impacts downstream analysis and decision-making processes.

More recently, statistical data compression methods, particularly Principal Component Analysis (PCA) and Singular Value Decomposition (SVD), have gained prominence as alternatives capable of overcoming some limitations of traditional compression schemes. PCA-based approaches have shown significant promise due to their ability to identify dominant patterns in high-dimensional datasets and project data onto a lower-dimensional subspace. For instance, PCA has been applied effectively to compress ultrasonic guided wave data in structural health monitoring, demonstrating substantial data reduction while preserving critical structural integrity information [2]. Similarly, SVD-based compression has been employed in the context of smart grid systems, efficiently capturing dominant spatial-temporal patterns in sensor measurements to reduce data transmission overhead [13]. Despite these strengths, both PCA and SVD rely on predetermined hyperparameters, such as the number of principal components or singular values retained, selected based on heuristic criteria or fixed thresholds. Consequently, these approaches often struggle to dynamically adapt to changing sensor data characteristics, especially in complex or multimodal scenarios. Moreover, PCA and SVD compression schemes typically do not explicitly incorporate task-specific criteria, thereby limiting their sensitivity to the unique application contexts and specific analytical goals inherent in various WSN deployments.

In recent advancements, machine learning techniques have been applied to data compression within WSNs to improve performance beyond traditional data compression methods. Many works focus on single-modality compression, where models such as convolutional and recurrent autoencoders are used to compress structured data like images or speech while preserving perceptual quality. For instance, recurrent neural networks have been applied for full-resolution image compression [14][27], and variational frameworks with scale hyperpriors have been developed to enhance entropy modeling [15]. Deep generative models have also been explored for learning-based audio compression [16] [28]. More recently, efforts have expanded toward multi-stream compression, where multiple data streams, such as images, audio, and sensor measurements, are jointly encoded to exploit inter-stream correlations. Representative approaches include factorized representation learning across streams [17] and joint latent space translation frameworks [18], which aim to enhance compression efficiency by capturing shared structure among heterogeneous inputs. However, most of these methods are developed under idealized settings and do not consider the computational load, memory usage, and energy constraints faced by typical WSN systems, limiting their practicality in real-world scenarios.

Parallel to compression, effective coding schemes are critical for ensuring robust data transmission over noisy wireless channels. Traditional handcrafted error-control coding methods, including convolutional codes [19], Turbo codes [20], and low-density parity-check (LDPC) codes [21], have been widely adopted. However, these approaches often fail to adequately address the unique bit-level error sensitivities characteristic of compressed sensor data. Recently, research has focused on learning-based coding schemes that jointly optimize data representations and error-correction mechanisms to

improve resilience against channel noise. A representative approach is the deep Joint Source-Channel Coding (JSCC) framework [22], developed specifically for wireless image transmission. JSCC integrates source coding, channel coding, and modulation into a single end-to-end trainable pipeline. While demonstrating effectiveness under controlled simulation environments, JSCC's tightly integrated architecture significantly restricts its flexibility and practical interoperability. Particularly, JSCC complicates integration with established frequency-domain transmission techniques, such as Orthogonal Frequency-Division Multiplexing (OFDM), which are fundamental for efficient spectrum utilization and robust transmission in realistic wireless deployments. Additionally, due to the implicit learning of modulation, JSCC lacks adaptability to dynamic spectrum allocation and carrier selection.

To mitigate these issues, a neural joint source-channel coding approach termed NECST (Neural Joint Source-Channel Coding) was proposed [23]. NECST optimizes source and channel coding jointly, explicitly modeling the probabilistic behavior of noisy channels. This method achieves lower reconstruction errors compared to traditional benchmarks under identical noise conditions and code lengths. Further enhancing NECST's robustness, an Infomax Adversarial Bit Flip (IABF) regularization strategy was developed [24]. The IABF method integrates mutual information maximization and adversarial training to jointly refine source compression and error correction in an end-to-end neural setting. Nonetheless, those approaches assume symmetric transmitter and receiver complexities, limiting their ability to flexibly balance transmitter-side computational load with bandwidth usage. Furthermore, NECST relies on the Variational Inference for Monte Carlo Objectives (VIMCO) [25] estimator to handle the non-differentiable Bernoulli sampling process. Although VIMCO provides unbiased estimates, its high variance can significantly complicate and destabilize the training process.

To overcome these identified limitations, this paper introduces a hierarchical and asymmetric Autoencoder framework tailored specifically for WSN deployments. Our proposed design clearly delineates intra- and inter-stream compression responsibilities, allowing flexible trade-offs between transmitter complexity and bandwidth utilization. It incorporates dedicated learning-based binary Code Generation and Interpretation modules, enhancing robustness against noisy channel conditions. Additionally, to effectively manage the non-differentiable binary representation operations, the framework utilizes a Straight-Through Estimator (STE), enabling seamless end-to-end training. Consequently, this proposed method offers a unified, adaptable, and practically deployable compression-coding solution explicitly designed for resource-constrained wireless sensor networks.

3 SYSTEM MODEL

Figure 1 depicts the architectural details of the proposed framework. The sensor unit is deployed at a greenhouse planting area [5]. The sensor modalities include temperature (T), light intensity (L), photosynthetically active radiation (PAR) p, humidity (H), and voltage (V) of the sensor system power source. A microprocessor in the sensor unit performs data sampling (with rate f), compresses the multi-stream data, encodes it into an error-resilient binary format, and transmits it over a noisy wireless channel. At the monitoring station, the receiver reconstructs the original data streams from the received error-resilient binary representation for downstream analysis. It is assumed that the transmitter here operates under the constraints of limited energy and computational capacity. In contrast, the receiver is assumed to have no such constraints. This asymmetry motivates a design that ensures low computation at the sensor node, minimal transmission overhead to save energy and bandwidth, and robustness against channel noise, while maintaining accurate reconstruction of the original sensor data. The end-to-end data and processing pipeline can be summarized as follows.

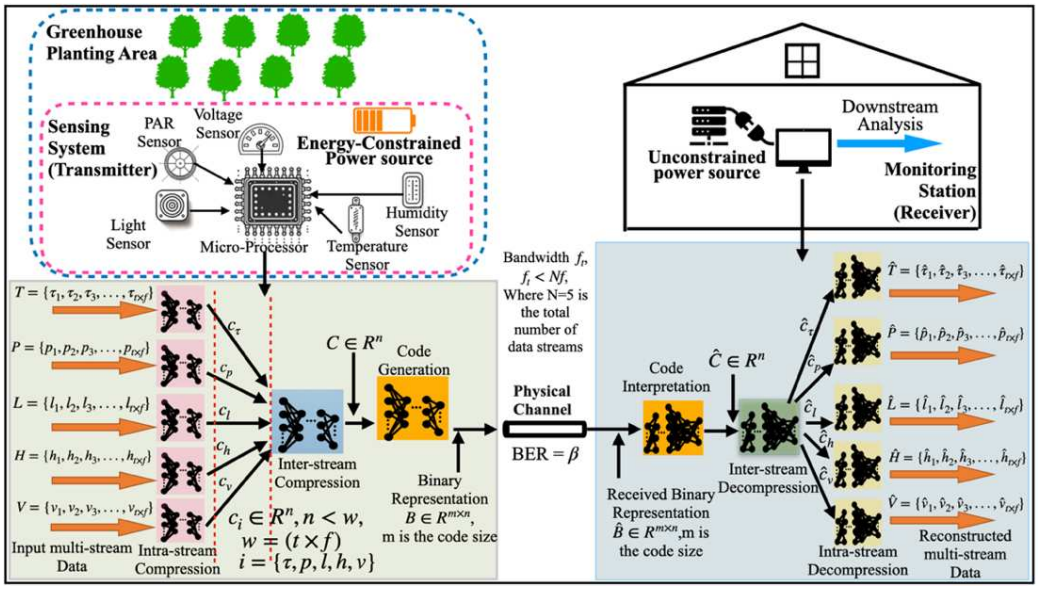

Fig. 1 Proposed System Architecture: Components, Workflow, and Data Flow

3.1 Transmitter Side Pipeline

The sensor unit with transmitter collects sensor modalities $\{T, p, L, H, V\}$ at sampling rate f. In time duration t, each modality generates a w −dimensional data stream, where $w = f \times t$. Note that the sampling rates for different modalities do not need to be uniform.

For data compression, the following steps are implemented in the transmitter side sensor unit.

1. Intra-Stream Compression: Each w −dimensional data stream is processed by a neural network (NN) to achieve compression, resulting in an n −dimensional compressed representation c_i, where $n < w$. Here, $i = \{\tau, p, l, h, v\}$, c_i, represents the compressed representation of the temperature, PAR, light intensity, humidity, and voltage respectively.
2. Inter-Stream Compression: A second NN-based encoder is employed to combine the n −dimensional compressed representations for all the data modalities into a single unified n −dimensional representation C for efficient transmission.
3. Code Generator: After obtaining the unified compressed representation C, an NN-based code generator module deployed in order to convert the compressed representation to an error-resilient binary representation B, where $B \in R^{n \times \mathrm{m}}$ and m is the code size. Eventually, the transmitter sends the binary representation to the receiver through the noisy channel.

3.2 Receiver Side Pipeline

Receiver Side Pipeline: At the receiver, the received binary representation is first decoded by the NN-based code interpretation module to retrieve the unified compressed representation $\hat{C}$. To reconstruct the original multi-stream data for all modalities, the receiver implements hierarchical NN-based decoders as deployed at the transmitter. First, an NN-based inter-stream decoder processes the reconstructed unified representation $\hat{C}$ to recover the individual n −dimensional

compressed representation $\hat{c}_i$ or each modality. Subsequently, each $\hat{c}_i$ is passed through another NN-based intra-stream decoders to reconstruct the original w −dimensional data streams corresponding to the modalities, namely, temperature ($\hat{T}$), light intensity ($\hat{L}$), photosynthetically active radiation ($\hat{P}$), humidity ($\hat{H}$), and voltage ($\hat{V}$). Such reconstructed data is then passed on to any downstream data analysis task.

4 FRAMEWORK FOR MULTI-MODAL COMPRESSION

4.1 Multi-Modal Compression and Transmission Framework

The primary functionality of the transmitter in the proposed framework is to read multi-modal data and send a compressed version that can be successfully decompressed at the receiver to retrieve data for all modalities. As the example use case in this work, data is collected from five separate streams, each sampled at f samples per second. The resolution for compression is t second long time window that contains $w = f \times t$ samples. This windowing is applied uniformly across all five data streams. As shown in Figure 1, these data streams are fed into an intra-stream compression module on a window-by-window basis. In this module, each data stream S_i is independently compressed by an NN encoder. The resulting compressed representation c_i for each modality is given by:

$$c_i = E_i^{n,o}(S_i; \theta_i) \qquad (2)$$

where $c_i \in R^n,\ and\ n < w$ is the compressed variable capturing the essential temporal information of each data stream. Here, $E_i^{o,n}(S_i; \theta_i)$ denotes the encoder parametrized by θ_i, with an n-dimensional output layer. The superscript 'o' denotes the computational complexity of the model, which is quantified by the number of multiplications operations required per inference cycle. In the proposed framework, the complexity o is tunable by adjusting either the number of hidden layers or the number of neurons in the internal layers. The objective of this step is to extract intra-stream information that is essential for accurate reconstruction.

Following the intra-stream compression stage, as discussed before (also refer Figure 1), the compressed representations from all individual modalities $\{c_\tau, c_p, c_l, c_h, c_v\}$ are combined in the Inter-Stream Compression Module. In this module, an NN encoder is implemented to eliminate cross-modality correlations and further reduce the overall data volume, which produces a unified compressed vector. Specifically, let $c_i = \{c_{i,1}, c_{i,2}, \dots, c_{i,n}\}$ denote the n −dimensional compressed representation for modality i. The NN encoder process the nth element from all modalities simultaneously to generate the corresponding compressed value C_n. This operation not only preserves temporal alignment across modalities but also enhances compression efficiency by exploiting inter-modality correlations. Formally, this process can be written as:

$$C_n = G\left(c_{\tau,n}, c_{p,n}, c_{l,n}, c_{h,n}, c_{v,n}; \phi\right) \qquad (3)$$

where $C_n \in R$, denotes the compressed value at index n in the unified representation C. The full compressed vector is given by $C = \{C_1, C_2, .., C_n\} \in R^n$. $G\left(c_{\tau,n}, c_{p,n}, c_{l,n}, c_{h,n}, c_{v,n}; \phi\right)$ is the intra-stream encoder parametrized by ϕ.

Once the unified compressed vector C is obtained from the Inter-Stream Compression Module, it is transmitted over a noisy communication channel. Transmission impairments such as interference, power limitations, and path loss can compromise data integrity, making continuous-valued representations vulnerable to distortion and less robust against errors. To enhance reliability, C needs to be converted into a binary form before the transmission. A Key challenge in noisy communication channels is the occurrence of bit errors, where transmitted bits are corrupted, typically flipped from 0 to 1 or vice versa, due to random noise in the channel. To model this behavior, we adopt a Binary Symmetric Channel (BSC),

where each bit has an independent probability ρ of being flipped. varying ρ allow us to simulate different levels of channel noise.

To improve robustness to channel noise, we introduce a learning-based coding scheme jointly optimized with the overall system. By adapting the encoded structure to both feature content and channel characteristics, the model better preserves critical information under transmission errors. The process involves three main steps: generating logits, applying probabilistic mapping, and threshold-based binarization. These steps are illustrated in Equations 4 to 6.

As shown in Figure 1, first, a neural network $\mathcal{C}(C_n; \psi)$ generates a real-valued logit matrix b from C. Specifically, for each element C_n in unified compressed variable C, the neural network outputs a logit vector b_n, which can be expressed as:

$$b_n = \mathcal{C}(C_n; \psi) \tag{4}$$

where $b_n \in R^m$, m is the code size that is selected empirically based on the compression ratio, downstream task requirements, and the strength of channel noise. Collectively The logit matrix $b = \{b_1, b_2, \dots b_n\} \in R^{n \times m}$, consists of real-valued scores that represent the unnormalized likelihood of each encoded bit being 1 before binarization, reflecting how strongly the model "believes" each bit should be activated based on the input.

Next, this logit vector is passed through a sigmoid function which yields:

$$z = \sigma(b) \tag{5}$$

where,

$$z \in R^{n \times m} \tag{5a}$$

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{5b}$$

This step is crucial because the sigmoid function maps the logit values to the range [0,1] that represents the probability of each encoded bit being 1. Physically, this transformation allows the model to express uncertainty or confidence in each bit, which supports a probabilistic and learnable binarization process in the next stage.

Finally, an element-wise threshold of ε is applied to z to produce the binary code $B \in R^{n \times \mathrm{m}}$:

$$B_{j,k} = \begin{cases} 1, \textit{ if } z_{j,k} > \varepsilon, \\ 0, \textit{ otherwise.} \end{cases} \tag{6}$$

The threshold ε is chosen based on the desired bit distribution. A lower ε yields dense code with more redundancy, while a higher ε leads to sparse code with lower overhead but increases vulnerability to channel errors. The resulting binary code B is then transmitted to the receiver through a noisy channel. In essence, by training the NN model $\mathcal{C}(C_n; \psi)$, the compressed representation C is converted to an error-resilient binary format, enhancing the robustness of the transmitted data and enables accurate reconstruction despite potential channel noise, which will be discussed next.

During transmission, channel noise may cause bit flips in the binary code B, resulting in the corrupted binary representation $\hat{B} = \{\hat{b}_1, \hat{b}_2, \dots, \hat{b}_n\}$, where each $\hat{b}_n \in R^m$, corresponding to corresponding to nth element of original compressed vector C. At the receiver, as shown in Fig. 1, the Code Interpretation Module uses a neural network decoder $\mathcal{C}'(\hat{b}_n; \psi')$, parametrized by ψ', to reconstruct the unified compressed representation $\hat{C}$ with each element computed as:

$$\hat{C}_n = \mathcal{C}'(\hat{b}_n; \psi') \tag{7}$$

Here, $\mathcal{C}'$Collectively the reconstructed unified data can be represented as $\hat{C} = \{\hat{C}_1, \hat{C}_2, \ldots, \hat{C}_n\} \in R^n$.

Once the unified compressed vector $\hat{C}$ is obtained, it is passed to the Inter-Stream Decompression Module, which reconstructs the individual compressed representations $\{\hat{c}_\tau, \hat{c}_p, \hat{c}_l, \hat{c}_h, \hat{c}_v\}$ for each sensor modality from the unified vector $\hat{C}$. A neural network decoder $G'(\hat{C}_n; \phi')$, parameterized by ϕ', is implemented and mirrors the Inter-Stream encoder at the transmitter, as shown in Fig 1. Formally, let $\hat{C} = \{\hat{C}_1, \hat{C}_2, \ldots, \hat{C}_n\} \in R^n$ denote the decoded unified representation. For each element $\hat{C}_n$ in $\hat{C}$, the decoder outputs:

$$\left(\hat{c}_{\tau,n}, \hat{c}_{p,n}, \hat{c}_{l,n}, \hat{c}_{h,n}, \hat{c}_{v,n}\right) = G'(\hat{C}_n; \phi') \qquad (8)$$

where each $\hat{c}_{i,n} \in$ R, represents the nth element of recovered compressed representation $\hat{c}_i$ for modality $i \in \{\tau, p, l, h, v\}$. This step ensures that the cross-modal information captured during inter-stream compression is accurately separated back into its respective modalities.

The output of the Inter-Stream Decoding Module $\{\hat{c}_\tau, \hat{c}_p, \hat{c}_l, \hat{c}_h, \hat{c}_v\}$ are then passed to the Intra-Stream Decoding Module (see Figure 1). This module uses modality-specific neural network decoders $D_i^{n,o'}(\widehat{c_i}; \theta'_i)$, parameterized by θ'_i, to reconstruct the original data stream $\{\hat{S}_\tau, \hat{S}_p, \hat{S}_l, \hat{S}_h, \hat{S}_v\}$. The decoding process for each modality is expressed as:

$$\hat{S}_i = D_i^{n,o'}(\widehat{c_i}; \theta'_i) \qquad (9)$$

where $\hat{S}_i \in R^w$ represents the reconstructed w-dimensional data stream for modality i. The complexity o' of each decoder can be tuned empirically, allowing flexibility to balance the reconstruction accuracy and computational cost. By mirroring the functionality of the Intra-Stream Compression process at the transmitter side, this module ensures high-fidelity recovery of the original multi-stream sensor data.

In practice, the transmitter and receiver store pretrained models of varying complexity and compression levels. At runtime, the system selects the most suitable encoder-decoder pair based on resource availability, and performance needs. Training details are provided in the next subsection.

4.2 End-to-End Training Methodology for Robust Multi-Modal Compression

To ensure efficient and robust performance under real-world conditions, the entire framework is trained end-to-end, as illustrated in Fig. 2. This approach allows all neural network modules (as shown in Fig. 1) to be jointly optimized to minimize the reconstruction error.

The training objective is to minimize the Mean Absolute Error (MAE) between reconstructed sensor data $\hat{\mathrm{S}}_i$ and the original sensor data S_i for all modalities $i \in \{\tau, p, l, h, v\}$:

$$\mathcal{L}(\theta_i, \phi, \psi, \psi', \phi', \theta'_i) = \sum_{i \in \{\tau,p,l,h,v\}} \frac{1}{w} \left|\hat{\mathrm{S}}_i - S_i\right| \qquad (10)$$

where w is window size, and $|\cdot|$ denotes the $l1$-norm. By backpropagating the MAE loss through the entire pipeline, all components are jointly optimized to minimize reconstruction error, ensuring cohesive operation and robust performance.

However, a key subtlety in this end-to-end pipeline is the thresholding operation in Equation 6 to produce binary codes. While crucial for generating noise-resilient codes, it introduces zero gradient and a non-differentiable point at $z_{j,k} = \epsilon$. Formally, the thresholding operation can be expressed as a Heaviside step function [6] $H(z_{j,k} - \epsilon)$, where:

$$H(z_{j,k} - \epsilon) = \begin{cases} 0, & for\ z_{j,k} - \epsilon < 0\ , \\ 1, & for\ z_{j,k} - \epsilon > 0. \end{cases} \qquad (12)$$

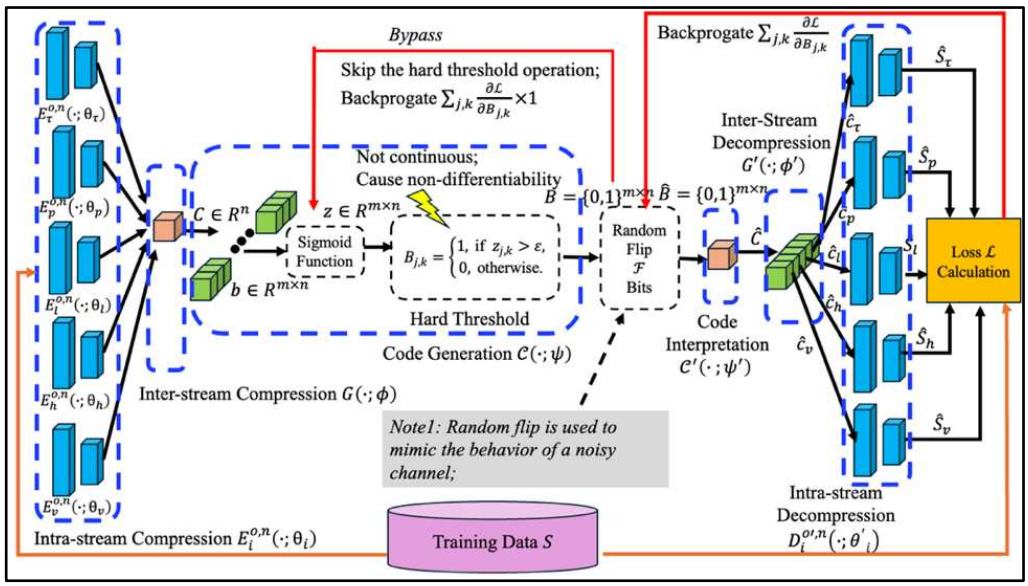

Fig. 2 End-to-end training procedure of proposed framework

The derivation of $H(z_{j,k}-\epsilon)$ with respect to $z_{j,k}$ is zero almost everywhere and undefined at the threshold:

$$\frac{\partial H(z_{j,k}-\varepsilon)}{\partial z_{j,k}}=\begin{cases}0, & for\, z_{j,k}\neq\varepsilon,\\ undefined, & for\, z_{j,k}=\varepsilon.\end{cases} \quad (11)$$

We define $W=\{\theta_i, \phi, \psi\}$ as set of all trainable parameters at transmitter side, which include θ_i in Instra-stream Compression Module, ϕ in Intra-Stream Compression Module, and ψ in Code Generation Module. during backpropagation, the gradient of loss $\mathcal{L}$ with respect to the parameter set W is computed using chain rule:

$$\frac{\partial \mathcal{L}}{\partial W}=\sum_{j,k}\frac{\partial \mathcal{L}}{\partial B_{i,j}}\times\frac{\partial B_{i,j}}{\partial z_{j,k}}\times\frac{\partial z_{j,k}}{\partial b_{j,k}}\times\frac{\partial b_{j,k}}{\partial W} \quad (12)$$

Since $\frac{\partial B_{i,j}}{\partial z_{j,k}}=\frac{\partial H(z_{j,k}-\varepsilon)}{\partial z_{j,k}}=0$ (or undefined at $z_{j,k}=\varepsilon$), standard backpropagation yields zero or meaningless gradients, preventing updates to upstream parameters θ_i, ϕ, ψ.

To address, we use Straight-Through Estimator (STE) [7] to approximates the corresponding gradients during training. The forward pass remains the same (as shown in Figure 2): each element in the logit vector z is pushed through a hard threshold to obtain a binary output. However, during back propagation, the thresholding operation's derivative, which is normally zero or undefined, is replaced by the simple identity (i.e., $\frac{\partial B_{i,j}}{\partial z_{j,k}}=1$).. While this approximation is biased and ignores the true derivative, it enables end-to-end training of proposed framework, preserving the advantages of binarization while still updating the parameters via gradient-based optimization.

To further improve robustness, in the training process, we randomly flip a subset of bits in the coded compressed representation B. This controlled noise injection improves the model's resilience to channel errors without significantly degrading reconstruction accuracy. The effectiveness of this strategy is analyzed in Section V. The overall training algorithm for our proposed framework is summarized in Algorithm 1

ALGORITHM 1: Training algorithm for the proposed framework

Initialization:

1.1 Define the Intra-Stream Compression module $E_i^{n,o}$ with a compressed dimension n and complexity o.

1.2 Define the Intra-Stream Decompression module $D_i^{n,o'}$, with a compressed dimension n and complexity o'.

1.3 Initialize all parameters $\{\theta_i, \phi, \psi, \psi', \phi', \theta'_i\}$ with random weights value.

1.4 Set learning rate α,

1.5 Set number of flipped bits $\mathcal{F}$.

1.6 Set total epochs $\mathcal{N}$.

1.7 Set current epoch $e = 0$.

While $e < \mathcal{N}$**:**

//Intra-stream compression at transmitter side

for each modality $i \in \{\tau, p, l, h, v\}$ **do**:

$c_i \leftarrow E_i^{n,o}(S_i; \theta_i)$

end for

//Intra-stream compression at transmitter side

$C \leftarrow G(c_\tau, c_p, c_l, c_h, c_v; \phi)$

//Code generation at transmitter side

$b \leftarrow \mathcal{C}(C; \psi)$

$z \leftarrow \sigma(b)$, *where* $z \in R^{m \times n}$, $\sigma(x) = \frac{1}{1+e^{-x}}$

$B \leftarrow$ *zero matrix of dimension* (m,n)

//Binarize z with threshold ε:

for each element $z_{j,k}$ in z **do**:

if $z_{j,k} > \varepsilon$ **then**:

$B_{j,k} = 1$

else:

$B_{j,k} = 0$

end if

end for

//Random Bit-Flipping

$num\_flip \leftarrow 0$

$\hat{B} \leftarrow B$

$Flipped\_position \leftarrow \emptyset$

while $num\_flip < \mathcal{F}$ **do**:

Repeat: randomly select (j, k)

Until (j, k) not in $Flipped\_position$

$\hat{B}_{j,k} \leftarrow 1 - \hat{B}_{j,k}$

Add (j, k) to $Flipped\_position$

$num\_flip \leftarrow num\_flip + 1$

//Decoding at receiver side

$\hat{C} \leftarrow \mathcal{C}'(\hat{B}; \psi')$

19: $(\widehat{c_\tau}, \widehat{c_p}, \widehat{c_l}, \widehat{c_h}, \widehat{c_v}) \leftarrow G'(\hat{C}; \phi')$

20: **for each modality** $i \in \{\tau, p, l, h, v\}$ **do**:

$\widehat{S_i} \leftarrow D_i^{n,o\prime}(\widehat{c_i}; \theta'_i)$

//Compute loss and Backprop

21: $\mathcal{L}(\theta_i, \phi, \psi, \psi', \phi', \theta'_i) \leftarrow \sum_i^N \frac{1}{w} |\hat{S}_i - S_i|$

// Update the parameter set p $\in \{\theta_i, \phi, \psi, \psi', \phi', \theta'_i\}$

22: $p \leftarrow p - \alpha \nabla_p \mathcal{L}(\theta_i, \phi, \psi, \psi', \phi', \theta'_i)$

23: $\boldsymbol{e \leftarrow e + 1}$

4.3 Single-Head vs. Multi-Head Autoencoders for Intra-Stream Compression

In our proposed framework, the Intra-Stream Compression module utilizes an independent encoder $E_i^{o,n}(\cdot; \theta_i)$ for each modality $i \in \{\tau, p, l, h, v\}$, effectively creating a multi-head architecture that compresses each modality separately (see Figure 1). Similarly, on the receiver side, the Intra-Stream Decompression module employs an independent decoder$D_i^{o',n}(\cdot; \theta_i)$ for each modality to reconstruct them individually. This mechanism allows proposed framework to be more tunable. The model complexity of each encoder and decoder can be adjusted independently based on the data properties. For example, for the sensing modalities that exhibits weak temporal dependency, the extraction of temporal feature can be difficult. In this scenario, implementing a more complex encoder and decoder architectures tailored to these modalities could enhance both compression and reconstruction efficiency. Moreover, this mechanism compresses and reconstruct each modality independently without assuming the correlations exists in different data streams.

However, this approach also increases the number of learnable parameters due to the need for multiple encoders and decoders, which can place a heavier burden on resource-constrained transmitters. Moreover, large number of trainable parameters can complicate model convergence, often leading to higher variance, especially when employing STE during training, given the model's non-differentiability. An alternative solution is to employ a single encoder-decoder pair in the Intra-Stream Compression and Decompression modules, thereby compressing and reconstructing all modalities with fewer parameters. The encoding process can be written as:

$$c_i = E^{n,o}(S_i; \theta_i) \qquad (13)$$

where n is the compressed dimension and o is the model complexity. Similarly, on the receiver side, the decoding process is:

$$\widehat{S_i} = D^{n,o'}(\widehat{c_i}; \theta'_i) \qquad (14)$$

where o' also determined empirically under similar constraints.

In our implementation, when strong inter-modal dependencies exist, a single-head design can improve performance by simplifying training and enhancing convergence. A detailed comparison with the multi-head setup is provided in Section V.

5 EXPERIMENT SETUP AND EVALUARION METRICS

5.1 Dataset Development

The proposed framework was validated using data from a greenhouse micro-climate monitoring system [5][8], where IoT sensor nodes measured temperature, light intensity, PAR, humidity, and system voltage, then transmitted the data for

analysis. The dataset was collected from a custom-built, solar-powered agricultural sensing platform [5][8], deployed at Michigan State University Greenhouses. Two such platforms have been operating continuously for several years. This study analyzes a subset of the collected data, comprising 338,860 samples per parameter.

5.2 Implementation

In our implementation, time series data is divided into windows of size w, where w is ranges from 100 to 200 in increments of 10. Then employ a series of neural network (NN) models (see Fig. 1) is applied to perform intra-stream compression, inter-stream compression, code generation, code interpretation, inter-stream decompression, and intra-stream decompression in sequence. Table 1 summarizes the architectures of these NN models, specifying the number of hidden layers and the number of neurons in each layer.

Table 1: The Architecture of models

Modules	Number of Hidden Layers	Number of Neurons in Each Layer
Intra-Stream Compression	2	$w_1 \times$ Compressed Dimension
Inter-Stream Compression	1	1
Code Generation	3	$32 \times 64 \times$ Code Size
Code Interpretation	3	$128 \times 64 \times 32$
Inter-Stream Decompression	1	5
Intra-Stream Decompression	2	$w_2 \times$ Compressed Dimension

To adjust transmitter complexity, the number of neurons in the first hidden layer of the Intra-Stream Compression module (w_1) can be changed. Similarly, to adjust receiver complexity, the corresponding layer in the Intra-Stream Decompression module (w_2) can be modified. We examine the impacts of the compressed dimension by varying its value from 10 to 100 in increments of 10. We also investigate the effects of code size by setting it to 32, 64, and 128 respectively. During experimental phase, all modules are trained jointly. The training parameters are shown in Table 2.

Table 2: The Training Parameters

Parameter	Value
Optimizer	Adam
Learning Rate	0.0001
K-fold Validation	10
Train-test split	90:10 (%)

To investigate the proposed framework's robustness to the channel noisy, we introduce bit flips (from 0 to 1 or from 1 to 0) during testing stage. Specifically, when the code size is m, and compressed dimension is n, transmitting a single window yield $n \times m$ bits. for BSC channel with BER β, the probability that exactly k bits are flipped is given by:

$$P(k) = \binom{k}{n \times m} \beta^k (1-\beta)^{(n \times m)-k} \qquad (15)$$

According to equation 15, we test the proposed framework for all possible number of flips (i.e., for which $P(k)$>0) using10-fold validation, and then compute the average results across these trials.

5.3 Evaluation Metric

The system is evaluated using normalized reconstruction error (NER), model complexity, and compression ratio. In addition, expected normalized reconstruction error is used to measure the framework's robustness to channel noise. The equations for each metric are outlined as follows:

Normalized Reconstruction Error: Let $\hat{S}_i$ denote the reconstructed stream for modality i, and S_i is the ground truth. The normalized reconstruction error for modality i is calculated as:

$$NRE_i = \frac{1}{w}\frac{\left|\hat{S}_i - S_i\right|}{S_{max}^{(i)} - S_{min}^{(i)}} \tag{16}$$

where $|\cdot|$ denote $l1$-norm. $S_i, \hat{S}_i \in R^w$, w is the window size. $S_{max}^{(i)}$ and $S_{min}^{(i)}$ are the global maximum and minimum values for modality i. To quantify the overall reconstruction quality, we define the Normalized Reconstruction Error (NRE) as the mean value of normalized reconstruction error of all modalities:

$$NRE = \frac{1}{N}\sum_{i\in\{\tau,p,l,h,v\}}\frac{1}{w}\frac{|\hat{S}_i - S_i|}{S_{max}^{(i)} - S_{min}^{(i)}} \tag{17}$$

where N is the total number of sensing modalities.

Model Complexity: Runtime complexities at the transmitter and the receiver are quantified by the number of multiplication operations required for each inference cycle. This metric indicates the computational load of neural network and is crucial for assessing their feasibility in practical scenarios.

Compression Ratio: Assume each element in a w-dimensional sample is represented using K-bits, transmitting the uncompressed sample in T seconds requires bandwidth:

$$\beta = \frac{w\times K}{T} \tag{18}$$

In the proposed framework, data are compressed to n-dimensions, where $n < w$, and each data point is coded to m-bits. the resulting transmission bandwidth is:

$$\beta' = \frac{n\times m}{T} < \beta \tag{19}$$

The compression ratio ρ can be quantified as:

$$\rho = \frac{\beta - \beta'}{\beta} = \frac{w\times K - n\times m}{w\times K} = 1 - \frac{n\times m}{w\times K} \tag{20}$$

This formulation highlights how reducing both dimensionality and bit-width lowers the required bandwidth proportionally.

Expected Normalized Reconstruction Error: Given BSC channel with BER β, the probability that exactly k bits are flipped during the transmission of a single window is given by equation 15. For N windows, the expected number N_k of transmissions that have k bits flipped is given by:

$$N_k = N\binom{k}{n\times m}\beta^k(1-\beta)^{(n\times m)-k} \tag{21}$$

When N_k transmissions have k bits flipped, the expected NER contributed by those N_k transmissions are calculated as:

$$NER_k^N = N\binom{k}{n \times m}\beta^k(1-\beta)^{(n\times m)-k}NER_k \quad (22)$$

Where NER_k^N is the expected NER from k bits flipped in N transmissions. NER_k denotes the expected NER corresponding to k bits flipped during a single transmission, which can be estimated from results aggregated over multiple runs.

Then the expected NER for given BER β is calculated as the mean of expected NER caused by all possible number of bit flips divided by the total number of transmissions:

$$E(NER) = \frac{1}{N}\sum_{k=1}^{n\times m} N\binom{k}{n \times m}\beta^k(1-\beta)^{(n\times m)-k}NER_k$$

$$= \sum_{k=1}^{n\times m}\binom{k}{n \times m}\beta^k(1-\beta)^{(n\times m)-k}NER_k \quad (22)$$

Thus, by evaluating $E(NER)$for different values of β, one can comprehensively assess how bit errors affect overall system performance.

5.4 Relevant Benchmarks

We evaluated the proposed framework's performance in multi-stream compression and robustness to channel noise against three benchmarks. First, we compared it to a naïve autoencoder that directly compresses concatenated streams, assessing model complexity, compression ratio, and reconstruction error.

To test noise robustness, we used two additional baselines: 1) raw data encoded via IEEE754 and protected using convolutional coding; 2) compressed data from the proposed framework, also IEEE754-encoded and convolutionally coded. Both benchmarks use a convolutional code with memory 3 and Viterbi decoding. All methods were tested under equal data rate conditions for fairness.

6 RESULTS AND DISCUSSION

To evaluate the proposed framework, we compare multi-head and single-head autoencoders on greenhouse data, showing simplification is possible with correlated streams. We then assess hierarchical compression under ideal channels, analyzing trade-offs among reconstruction error, compression ratio, and model complexity. Finally, we evaluate robustness by adding the coding modules and testing under varying BERs.

6.1 The comparison between multi-head Autoencoder and Single-head Autoencoder

Figure 3 compares multi-head and single-head Autoencoders across compression ratios, while Figure 4 shows the correlation matrix among data streams, revealing clear inter-stream correlations. In Fig. 3, We maintain identical model complexity for both multi-head and single-head Autoencoders. However, due to architectural differences, the number of trainable parameters differs significantly. The multi-head Autoencoder uses separate encoder-decoders for each stream in the Intra-Stream Compression and Decompression Modules, leading to five times more parameters in these modules. Other modules have identical parameter counts. Results show that the single-head Autoencoder achieves lower reconstruction

error for the same transmitter complexity by leveraging shared patterns across streams, making it efficient for correlated data.

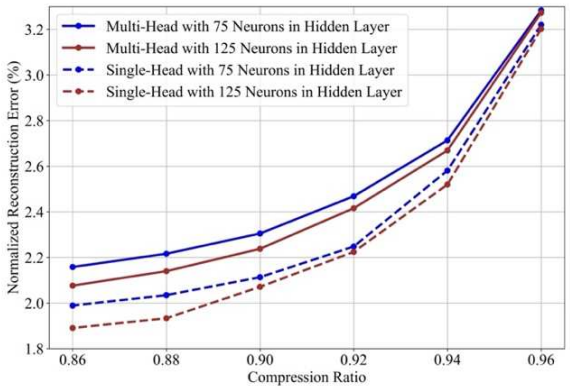

Fig. 3. Comparison of single-head and multi-head Autoencoders on a highly correlated greenhouse monitoring dataset.

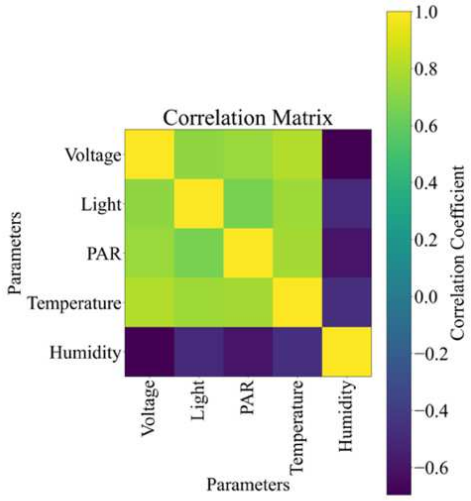

Fig. 4: Correlation matrix illustrating positive and negative correlations among different sensor data streams (Voltage, Light, PAR, Temperature, and Humidity).

Table 3: The comparison between proposed hierarchical Autoencoder and Naïve Autoencoder

Architecture	Complexity	Normalized Reconstruction Error (%)	Normalized Voltage Error (%)	Normalized Light Error (%)	Normalized PAR Error (%)	Normalized Temperature Error (%)	Normalized Humidity Error (%)	Compression Ratio (%)
Hierarchical	72400	0.46	1.41	0.05	0.80	0.03	0.01	0.84
Naive	72500	0.89	3.09	0.05	1.23	0.06	0.01	0.84

Naive	145000	0.87	3.11	0.05	1.13	0.06	0.01	0.84
Hierarchical	72400	0.48	1.46	0.06	0.82	0.03	0.01	0.88
Hierarchical	72400	0.60	1.86	0.06	1.02	0.04	0.01	0.92
Hierarchical	72400	0.93	2.98	0.09	1.50	0.05	0.01	0.96

6.2 The comparison between multi-head Autoencoder and Single-head Autoencoder

We compare the hierarchical Autoencoder with a naïve approach that merges and compresses all streams as a single input. To ensure fairness, each stream is independently normalized using a Min-Max Scaler. Table 3 presents the normalized reconstruction errors for both approaches under various transmitter complexities and compression ratios, with receiver complexity fixed. The hierarchical Autoencoder consistently outperforms the naïve approach, achieving lower reconstruction error even with reduced model complexity and more aggressive compression. This is attributed to its separate handling of intra-stream and inter-stream compression, which better captures per-stream and cross-stream correlations.

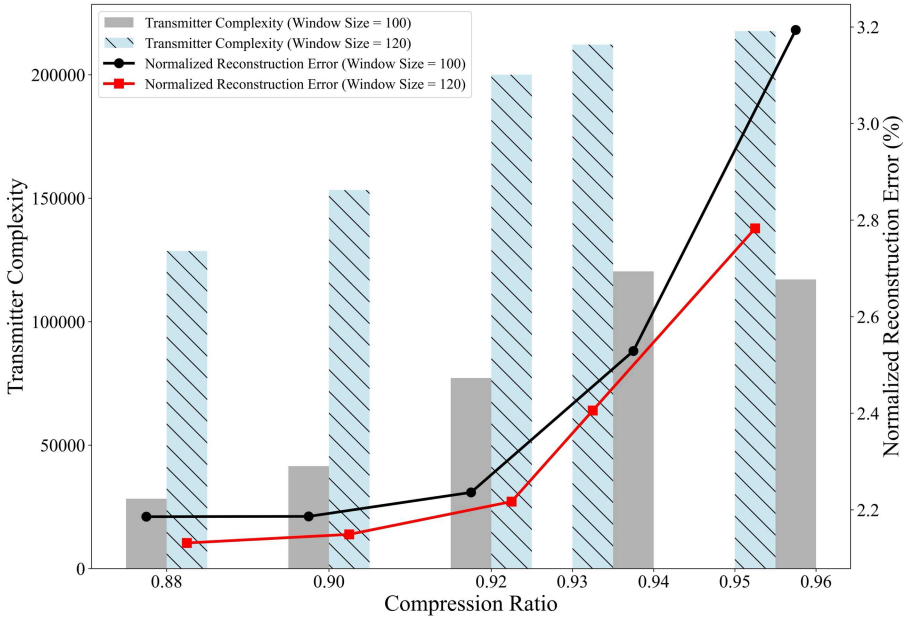

Fig. 5. The trade-off between compression ratio and normalized MAE under different window size

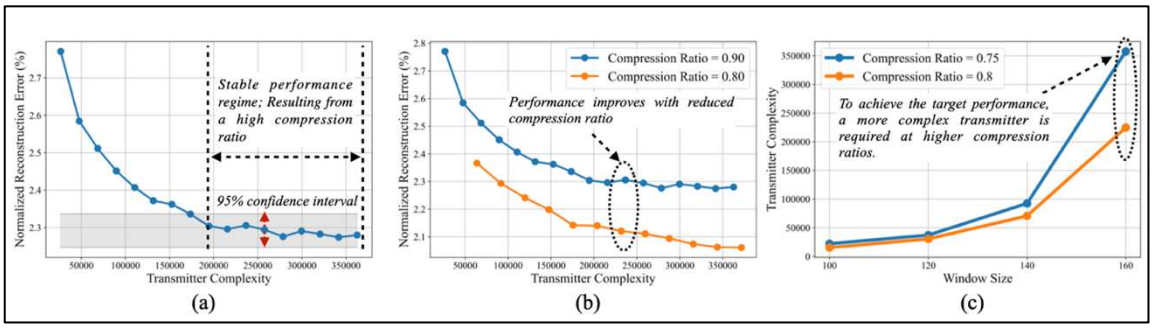

Fig. 6. The trade-off between window size and transmitter complexity for different compression ratio.

6.3 The trade-off between model complexity, compression ratio, and reconstruction error

Fig. 5 illustrates the trade-off between compression ratio and normalized reconstruction error for various window sizes. A key observation is that incremental compression can be achieved with minimal performance degradation within a certain sustainable range of compression ratios. Beyond this range, further compression leads to a noticeable increase in reconstruction error. Comparisons across different window sizes show that larger windows can yield better performance by capturing more temporal information. However, this improvement comes at the cost of higher computational expense. It is also important to emphasize that larger window sizes increase system latency, as it takes more time to collect a complete window of data at a given sampling frequency.

Fig. 6 illustrates the trade-off between compression ratio and transmitter complexity. Fig. 6(a) shows that increasing transmitter complexity initially reduces reconstruction error. However, beyond a threshold, further increases yield no additional gains, and the system enters a stable regime. The gray shading marks the 95% confidence interval for a representative point in this plateau, indicating the optimal performance range. This occurs because both transmitter complexity and compressed dimension affect how much information is captured; once the encoder is sufficiently powerful, the fixed compressed dimension becomes the bottleneck. Fig. 6(b) further shows that, in the stable regime, reducing the compression ratio leads to noticeable performance improvement, highlighting the influence of compression ratio

Fig. 6(c) shows the minimum transmitter complexity needed to reach the stable regime in Fig. 6(a), across different window sizes and compression ratios with fixed receiver complexity. A key observation is that Higher compression ratios require more transmitter complexity, and this difference grows with window size. Achieving target performance requires encoding enough information within the compressed variable. Higher compression ratios reduce dimensionality of the compressed data, demanding a more complex transmitter to capture key features with limited dimensionality. As window size increases, more temporal data must be processed, and although compressed dimensionality also grows, it lags behind the rising input volume, widening the complexity gap between high and low compression ratios.

6.4 The trade-off between transmitter complexity, receiver complexity and reconstruction error

Fig. 7 illustrates the trade-offs among transmitter complexity, receiver complexity, and reconstruction error. Fig. 7(a) and 7(c) show that after reaching a stable regime with sufficient transmitter complexity, increasing receiver complexity initially reduces reconstruction error. However, beyond a threshold, the benefit saturates, as the compression ratio limits the total transmittable information. Figure 7(b) shows that when both transmitter and receiver complexities are high, lowering the compression ratio significantly improves reconstruction by enabling the compressed variable to carry more information. These results highlight that when transmitter resources are constrained, performance can still improve by increasing receiver complexity. Once both sides reach saturation, further improvement relies on reducing the compression ratio.

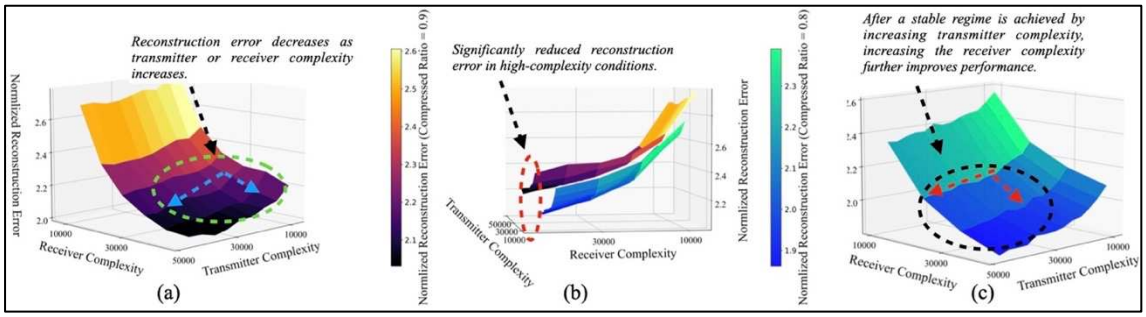

Fig. 7. Trade-off relationship between transmitter complexity and receiver complexity and reconstruction error for different compressed dimension.

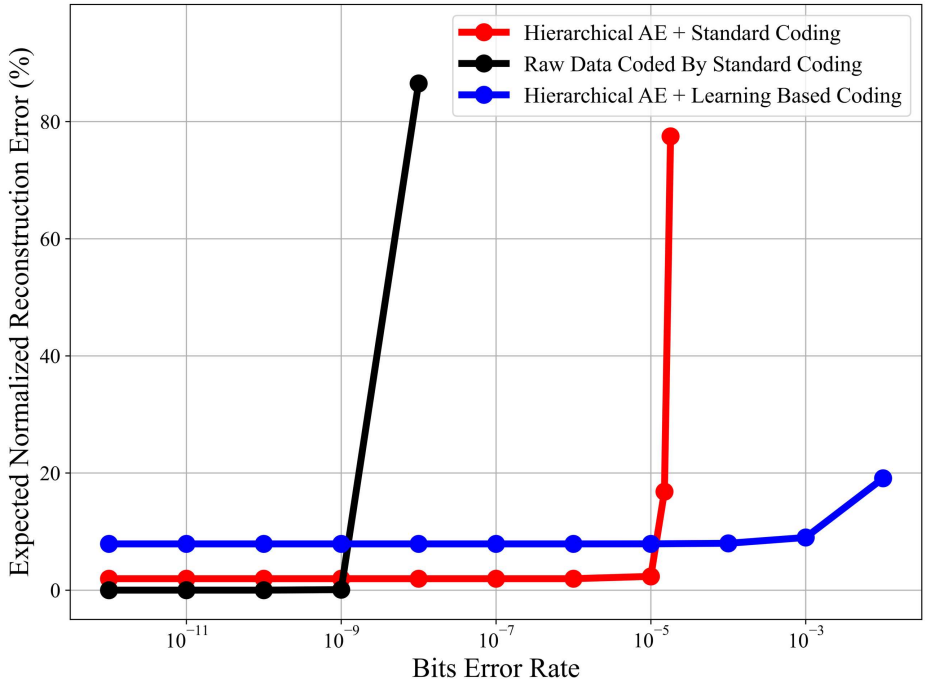

Fig. 8. Comparison of channel noise robustness with baseline approaches

6.5 Robustness to the channel error

To assess robustness to channel errors, we compare the proposed framework with Code Generation and Interpretation Modules (as shown in Fig. 1) against two benchmarks from Section IV. As shown in Fig. 8, when BER less than 10^{-9} raw data transmission yields the lowest reconstruction error but requires high bandwidth and introduces latency. The

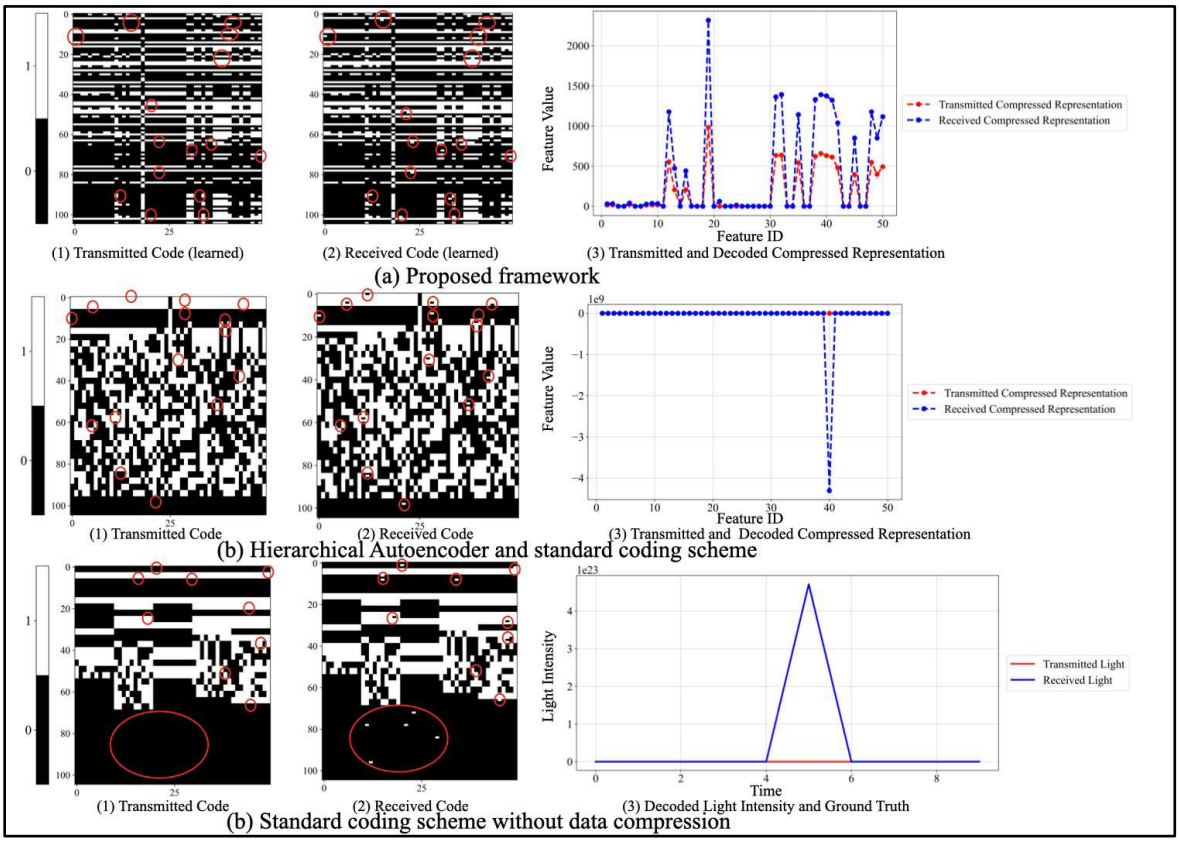

Fig 9. Comparison of transmitted binary code, received binary code, and decoded data under 15 bits flips. (Top) Proposed Coding Scheme, (Mid) Hierarchical Autoencoder and standard coding scheme, (Bottom) Standard coding scheme without data compression

hierarchical autoencoder reduces bandwidth by discarding redundancy, introducing minor errors even at low BER. Adding the coding modules slightly increases reconstruction error when BER lower than 10^{-5} due to quantization, which limits representational capacity.

However, these modules significantly improve robustness by learning error-resilient binary encoding and decoding. At higher BERs, standard coding fails and leading to sharp performance drops, even with convolutional correction. It is important to note that the proposed method maintains low error rates with only 64-bit codes per value, compared to 105 bits in the baselines, demonstrating greater efficiency and robustness. To provide a more in-depth comparison among the three methods, Fig. 9 shows the transmitted and received binary sequences, and corresponding decoded information, under 15 flipped bits, with all methods using 105-bit codes. It is observed that the proposed method maintains the same trend in the decoded output as the transmitted data, while benchmarks suffer from severe reconstruction errors.

6.6 The influence of code size

Fig 10 shows the influence of code size on the reconstruction error under various BER with compression ratio of 0.9. The observation is that for given BER, larger code size gives lower reconstruction error. This is because, with a larger code

size, the proposed framework encodes information in a way that reduces the impact of individual bit errors, improving robustness to channel noise.

Fig 11 shows the cumulative distribution functions (CDF) of the number of flipped bits at a fixed channel BER for various code size, demonstrating the improved resilience with larger codes. In this experiment, the compressed dimension is set to 50, which means each transmission consists of 50 floating-point numbers, which yields totally $N = m \times 50$ bits, where m is the code size. We define n_{max}^{m} as the minimal number of flips for which the CDF exceeds 0.99 at code size m,

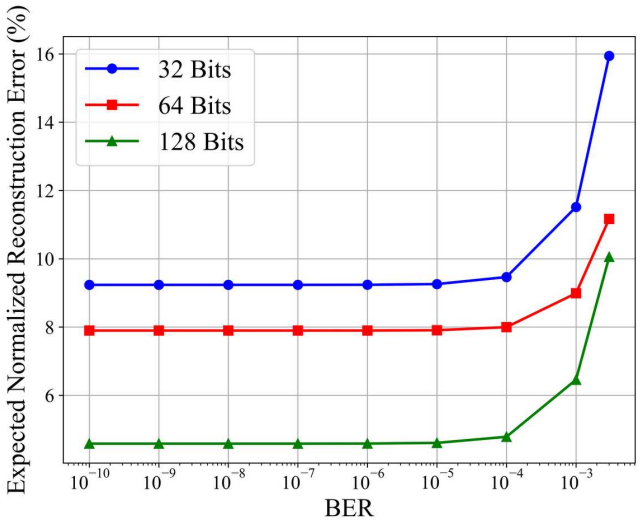

Fig. 10. The influence of code size

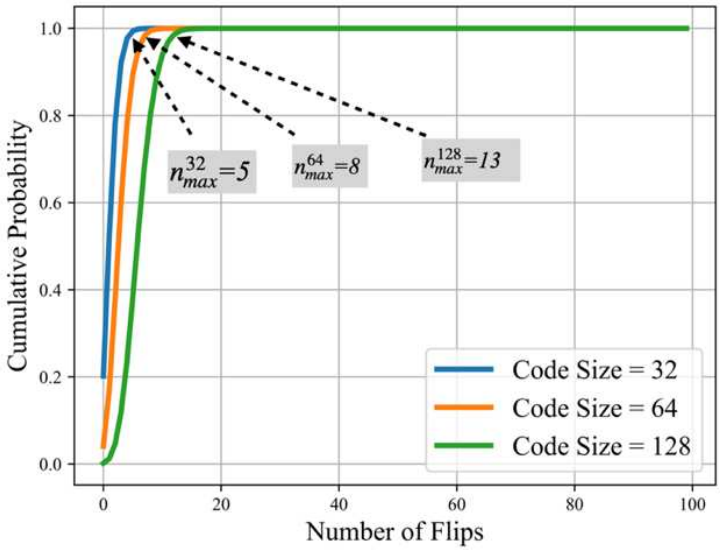

Fig. 11. the cumulative distribution functions of the number of flipped bits

which indicates an upper bound of number of flips for single transmission when code size is m, beyond which additional flips are extremely unlikely. The practical maximum BER when code size is m can be calculated as $\beta_{max}^{m} = \frac{n_{max}^{m}}{N}$. According to Fig 11, the n_{max}^{m} and β_{max}^{m} for various code size is shown in Table 4.

Table 4: The Training Parameters

Code Size (bits)	n_{max}^{m} (bits)	β_{max}^{m} (%)
32	5	0.0031
64	8	0.0025
128	13	0.0020

According to Table IV, the β_{max}^{m} is lower for larger code size, indicating that a larger code size can improve the proposed framework's robustness to the channel error, which is consistent with Shannon's capacity theorem [23], which states that reliable communication is achievable up to a certain limit when sufficient redundancy is added.

6.7 The impact of flips in training stage

As introduced in section 4, during training stage, randomly flipping a subset of bits in the coded compressed representation can enhance the model's robustness to channel. Fig 12 presents the performance of models trained with different number of flips under various BER, the observation is that the model trained with larger number of flips gives a lower reconstruction error. The model trained with bit flipping effectively learns how to decode corrupted representations. Consequently, a model trained with a larger number of flips becomes more robust to channel error, resulting in lower reconstruction errors.

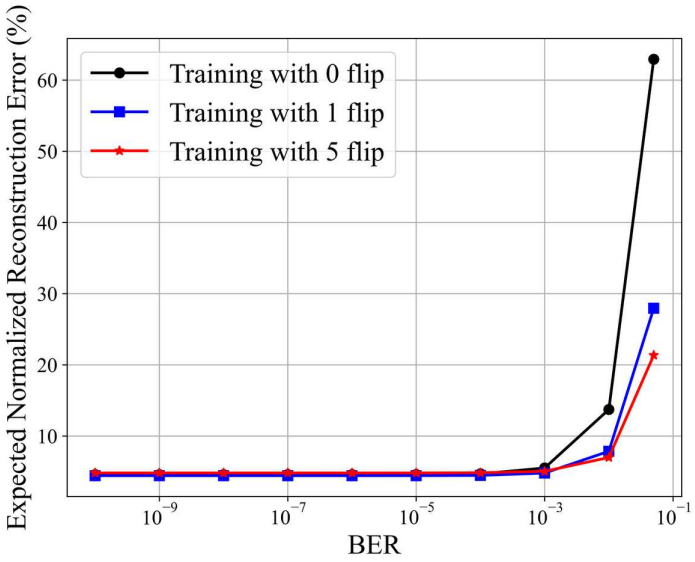

Fig. 12. The influence of random flips in training stage

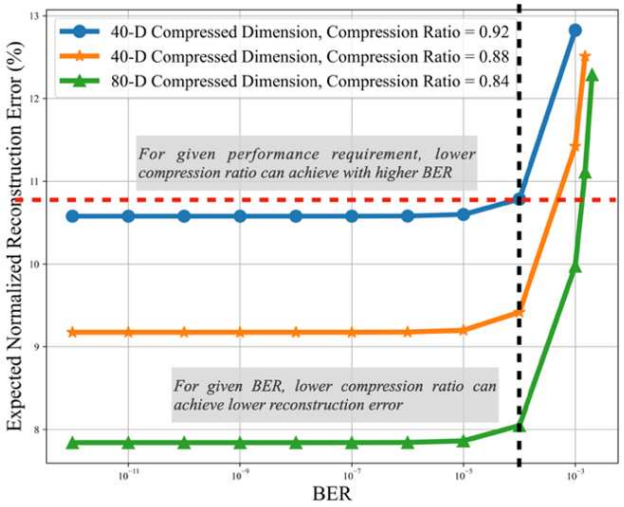

Fig. 13. The influence of compressed dimension under various BER

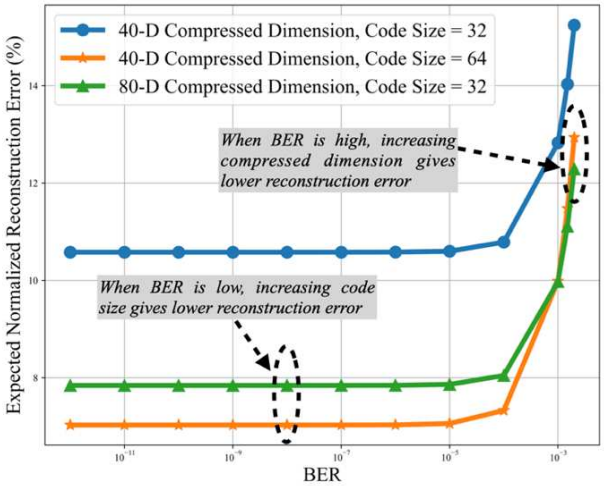

Fig. 14. The influence of code size under various BER

6.8 The influence of compression ratio under noisy channel

As defined in equation 22, the compression ratio can be adjusted by adopting the compressed dimension or code size. Fig 13 demonstrates the influence of compressed dimension on the reconstruction error under various BER and fixed code size of 32. The observation is that for a given BER, a larger compressed dimension yields better reconstruction. Conversely, to meet a fixed performance level, a larger compressed dimension tolerates higher BERs. This is because, with fixed code size, increasing compressed dimension spreads information across more bits, reducing per-bit information density and minimizing the impact of bit errors.

As shown in Equation 22, the compression ratio can be adjusted by changing either the code size or the compressed dimension. Figure 14 compares their effects with a fixed window size of 100. Starting from a compressed dimension of 40

and code size of 32, halving the compression ratio can be achieved by doubling either parameter. The key observation is that when BER is low, increasing code size can give lower reconstruction error. However, when BER is high, a larger compressed dimension can give the lowest reconstruction error. This is because at low BER, fewer bit flips occur, so increasing code size reduces per-bit information density and limits the impact of errors. At high BER, errors are widespread, increasing compressed dimension lowers per-variable information density, enhancing system's robustness.

7 SUMMARY AND CONCLUSION

This paper presents a hierarchical autoencoder framework for bandwidth-efficient and error-resilient data transmission in wireless sensor networks. By decoupling intra-stream and inter-stream compression, the architecture captures both modality-specific features and cross-stream correlations, achieving higher compression ratios with lower model complexity than naïve autoencoders. System-level analysis shows that transmitter complexity should be increased only up to a performance plateau, beyond which improvements arise from enhancing receiver complexity or reducing the compression ratio. The proposed binary code generation and interpretation modules significantly outperform conventional convolutional coding under channel noise. Overall, the framework enables accurate reconstruction with reduced bandwidth and enhanced robustness, offering a practical solution for resource-constrained and error-prone sensing scenario.

REFERENCES

[1] Kassim M R M, Mat I, Harun A N. Wireless Sensor Network in precision agriculture application[C]//2014 international conference on computer, information and telecommunication systems (CITS). IEEE, 2014: 1-5.

[2] Yang K, Kim S, Harley J B. Guidelines for effective unsupervised guided wave compression and denoising in long-term guided wave structural health monitoring[J]. Structural Health Monitoring, 2023, 22(4): 2516-2530.

[3] Upreti K, Kumar N, Alam M S, et al. Machine learning-based Congestion Control Routing strategy for healthcare IoT enabled wireless sensor networks[C]//2021 Fourth International Conference on Electrical, Computer and Communication Technologies (ICECCT). IEEE, 2021: 1-6.

[4] [Chang V, Loganathan S, Golightly L, et al. Internet of things in manufacturing: impact of wireless sensor networks on machine health monitoring[J]. International Journal of Business Information Systems, 2023.

[5] Dutta H, Bhuyan A K, Gao K, et al. Cross-Modality Multivariate Regression for Energy-Bandwidth Economy in Resource-Constrained Agricultural IoTs[C]//2025 IEEE 22nd Consumer Communications & Networking Conference (CCNC). IEEE, 2025: 1-6.

[6] E. W. Weisstein, "Heaviside step function," From MathWorld–A Wolfram Web Resource. 2008. [Online]. Available: http://mathworld.wolfram.com/HeavisideStepFunction.html

[7] Yin P, Lyu J, Zhang S, et al. Understanding straight-through estimator in training activation quantized neural nets[J]. arXiv preprint arXiv:1903.05662, 2019.

[8] Williams R J. Simple statistical gradient-following algorithms for connectionist reinforcement learning[J]. Machine learning, 1992, 8: 229-256.

[9] Feng, Dezhi, et al. "Energy-efficient and secure data networking using chaotic pulse position coded PDUS." IEEE Transactions on Green Communications and Networking 4.2 (2019): 375-386.

[10] Han S, Mao H, Dally W J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding[J]. arXiv preprint arXiv:1510.00149, 2015.

[11] G. Antoniol and P. Tonella, "EEG data compression techniques," in IEEE Transactions on Biomedical Engineering, vol. 44, no. 2, pp. 105-114, Feb. 1997, doi: 10.1109/10.552239.

[12] B. A. Rajoub, "An efficient coding algorithm for the compression of ECG signals using the wavelet transform," in IEEE Transactions on Biomedical Engineering, vol. 49, no. 4, pp. 355-362, April 2002, doi: 10.1109/10.991163.

[13] Hashemipour N, Aghaei J, Kavousi-Fard A, et al. Optimal singular value decomposition based big data compression approach in smart grids[J]. IEEE Transactions on industry applications, 2021, 57(4): 3296-3305.

[14] Toderici G, Vincent D, Johnston N, et al. Full resolution image compression with recurrent neural networks[C]//Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. 2017: 5306-5314.

[15] Ballé J, Minnen D, Singh S, et al. Variational image compression with a scale hyperprior[J]. arXiv preprint arXiv:1802.01436, 2018.

[16] Kumar R, Seetharaman P, Luebs A, et al. High-fidelity audio compression with improved rvqgan[J]. Advances in Neural Information Processing Systems, 2023, 36: 27980-27993.

[17] Tsai Y H H, Liang P P, Zadeh A, et al. Learning factorized multimodal representations[J]. arXiv preprint arXiv:1806.06176, 2018.

[18] Li M, Huang P Y, Chang X, et al. Video pivoting unsupervised multi-modal machine translation[J]. IEEE Transactions on Pattern Analysis and Machine

Intelligence, 2022, 45(3): 3918-3932.

[19] Proakis J G, Salehi M. Digital communications[M]. McGraw-hill, 2008.

[20] Berrou C, Glavieux A. Near optimum error correcting coding and decoding: Turbo-codes[J]. IEEE Transactions on communications, 1996, 44(10): 1261-1271.

[21] Gallager R. Low-density parity-check codes[J]. IRE Transactions on information theory, 2003, 8(1): 21-28.

[22] Bourtsoulatze E, Kurka D B, Gündüz D. Deep joint source-channel coding for wireless image transmission[J]. IEEE Transactions on Cognitive Communications and Networking, 2019, 5(3): 567-579.

[23] Choi K, Tatwawadi K, Grover A, et al. Neural joint source-channel coding[C]//International Conference on Machine Learning. PMLR, 2019: 1182-1192.

[24] Song Y, Xu M, Yu L, et al. Infomax neural joint source-channel coding via adversarial bit flip[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2020, 34(04): 5834-5841.

[25] Mnih A, Rezende D. Variational inference for monte carlo objectives[C]//International Conference on Machine Learning. PMLR, 2016: 2188-2196.

[26] Shannon C E. A mathematical theory of communication[J]. The Bell system technical journal, 1948, 27(3): 379-423.

[27] Toderici G, O'Malley S M, Hwang S J, et al. Variable rate image compression with recurrent neural networks[J]. arXiv preprint arXiv:1511.06085, 2015.

[28] Huang Q, Liu T, Wu X, et al. A generative adversarial net-based bandwidth extension method for audio compression[J]. Journal of the Audio Engineering Society, 2019, 67(12): 986-993.