A Unified Multi-Task Semantic Communication System for Multimodal Data

Guangyi Zhang, *Graduate Student Member, IEEE*, Qiyu Hu, *Student Member, IEEE*, Zhijin Qin, *Senior Member, IEEE*, Yunlong Cai, *Senior Member, IEEE*, Guanding Yu, *Senior Member, IEEE*, and Xiaoming Tao, *Senior Member, IEEE*

***Abstract*—Task-oriented semantic communications have achieved significant performance gains. However, the employed deep neural networks in semantic communications have to be updated when the task is changed or multiple models need to be stored for performing different tasks. To address this issue, we develop a unified deep learning-enabled semantic communication system (U-DeepSC), where a unified end-to-end framework can serve many different tasks with multiple modalities of data. As the number of required features varies from task to task, we propose a vector-wise dynamic scheme that can adjust the number of transmitted symbols for different tasks. Moreover, our dynamic scheme can also adaptively adjust the number of transmitted features under different channel conditions to optimize the transmission efficiency. Particularly, we devise a lightweight feature selection module (FSM) to evaluate the importance of feature vectors, which can hierarchically drop redundant feature vectors and significantly accelerate the inference. To reduce the transmission overhead, we then design a unified codebook for feature representation to serve multiple tasks, where only the indices of these task-specific features in the codebook are transmitted. According to the simulation results, the proposed U-DeepSC achieves comparable performance to the task-oriented semantic communication system designed for a specific task but with significant reduction in both transmission overhead and model size.**

***Index Terms*—Deep learning, dynamic overhead, multimodal data, multi-task semantic communication.**

I. Introduction

With the rapid development of artificial intelligence, a huge amount of interconnected intelligent applications have appeared in the networks. To support massive connectivity for these applications over limited wireless resources, the conventional communication systems are facing critical challenges [1]–[3]. To address this issue, semantic communications have been considered as a promising technology to achieve better performance [4].

Parts of this paper were presented at the IEEE Global Communications Conference, Rio de Janeiro, Brazil, December 2022.

This work was supported in part by the National Natural Science Foundation of China under Grant U22A2004, 62293484, and 61925105, and also in part by Zhejiang Provincial Key Laboratory of Information Processing, Communication and Networking (IPCAN), Hangzhou 310027, China. *(Corresponding author: Yunlong Cai.)*

G. Zhang, Q. Hu, Y. Cai, and G. Yu are with the College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou 310027, China (e-mail: zhangguangyi@zju.edu.cn; qiyhu@zju.edu.cn; ylcai@zju.edu.cn; yuguanding@zju.edu.cn).

Z. Qin is with the Department of Electronic Engineering, Tsinghua University, Beijing 100084, China, and also with the Beijing National Research Center for Information Science and Technology, Beijing 100084, China (e-mail: qinzhijin@tsinghua.edu.cn).

X. Tao is with the Department of Electronic Engineering, Tsinghua University, Beijing 100084, China (e-mail: taoxm@tsinghua.edu.cn).

A. Prior Work

Semantic communications have recently received great attention [5]. Different from conventional communications, they only take into account the relevant semantic information to the tasks [6]–[8]. With the integration of communications and deep learning (DL), task-related semantic information can be extracted from the source data through deep neural networks (DNNs), and is represented by the encoded features. Recent DL-based studies on semantic communications have shown a great potential to achieve performance gains [9]–[29], especially in unfriendly channel environments.

The existing works on semantic communications can be mainly divided into two categories: data reconstruction [9]–[20] and task execution [21]–[29]. For the data reconstruction, the semantic system extracts global semantic information from the source data. Specifically, a so-called DeepSC framework in [9] encodes the text information into various lengths by employing sentence information. A novel semantic-preserving compression method for text transmission has been developed in [30], which saves the number of bits for message representation. In [10], an attention-based JSCC system operates with different signal-to-noise (SNR) levels during image transmission. As for video transmission, a DL-aided wireless video transmission system in [11] can overcome the cliff-effect. The semantic communication system in [12] converts speech signals into semantic features and decodes the received features into a reconstructed speech waveform. For the task execution applications, only the task-specific semantic information is extracted and encoded at the transmitter [21]–[25]. In particular, a model for image retrieval task under power and bandwidth constraints has been proposed in [21]. In [22], an image classification-oriented semantic communication system has been developed. In [23], a vector quantization-variational autoencoder (VQ-VAE) based robust semantic communication systems has been developed for image classification.

Though the aforementioned semantic communication systems have exhibited satisfactory performance in certain scenarios, they only handle one task with single modality of data. These DNN models are hard to simultaneously serve different tasks with multi-modality in practice for the reasons below: (i) The model has to be updated once the task is changed, which leads to massive gradient transmission for retraining it; (ii) Different models need to be stored for serving different tasks, which might be unrealistic for the edge devices with limited storage resources. Generally, most of the devices require multi-task service, hence developing a unified multi-task semantic communication system is of great importance. In [24], a Transformer-based framework has been proposed to address this issue initially. It is able to share the same transmitter structure for the considered tasks. However, the model in [24] still needs to be retrained separately for different tasks and the transceiver architecture has not been unified for different tasks yet. A recent work in [31] has designed a model to handle the image detection and segmentation tasks, but it only handles two tasks with one modality of data. In [32], a multi-task semantic approach was proposed for joint optimization of completing multiple tasks with multiple receivers, where the considered tasks are mainly image classification tasks.

There have been some works about multi-task learning in the field of computer vision and natural language processing [33]–[36]. Multi-task learning aims at utilizing the task-specific information contained in the training samples of related tasks. Compared with the single-task models, the multi-task models bring the following advantages: (i) The memory space for storing the model can be significantly reduced due to the shared model parameters for multi-task; (ii) It is easier to simultaneously train the model for multiple tasks and improve the performance if some related tasks share the complementary semantic information. Moreover, for data of multiple modalities, multi-representation learning is a significant task, which aims to cope with the consistency and the difference in different modalities of data, and explore modality representations to contain both consistent and complementary information [36].

B. Motivation and Contributions

It is foreseen that wireless networks are expected to provide various intelligent services in terms of generality and efficiency. Though there have been numerous semantic works for specific tasks, a unified multi-task model for different modalities of data in wireless communications has not been thoroughly investigated. Therefore, in this paper, we propose a unified DL-based semantic communication system (U-DeepSC). To the best of our knowledge, this is the first work on a unified semantic communication system designed to serve various tasks. Furthermore, it is important to notice that there is redundancy in transmitting all the features, as different tasks require varying numbers of transmitted features. For instance, data reconstruction typically demands more transmitted features compared to intelligent tasks. While transmitting more encoded features can enhance performance against noise by capitalizing on feature redundancy, this also introduces higher transmission overhead. Consequently, an inherent trade-off between performance and the number of transmitted symbols exists. Nevertheless, the majority of existing methods heavily rely on handcrafted designs, where the output size of employed DNN models remain fixed. This will cause the transmission rate to exceed its required minimum value. Therefore, it is important to determine an optimal transmission rate for each task within a unified multi-task semantic communication system.

To this end, we make the first attempt to devise a unified semantic communication for end-to-end data transmission. Our approach takes into account six widely recognized tasks within the semantic communication community, encompassing both single-modality tasks and multi-modality tasks. The proposed U-DeepSC is capable of simultaneously handling multiple tasks across three distinct modalities: image, text, and speech. In order to equip U-DeepSC with the ability to extract task-specific information for diverse tasks, we introduce task embedding vectors and task query matrices. These components are input alongside the source data for each task, effectively indicating the intended task for the given data. To determine an appropriate transmission overhead tailored to each task, we develop a dynamic channel encoder for U-DeepSC. This encoder is designed to drop redundant feature vectors for specific tasks and adjust the number of transmitted feature vectors based on the channel conditions. More specifically, we implement a lightweight feature selection module (FSM) within the channel encoder. This module generates a selection mask vector, taking into account both the tasks and current channel conditions, to precisely indicate which features need to be transmitted. In addition, the proposed FSM can evaluate the importance of feature vectors and hierarchically prune redundant feature vectors, significantly speeding up the inference. To reduce the transmission overhead as well as enable digital transmission, we adopt the codebook design in [23], where a discrete codebook shared by the transmitter and receiver is designed for encoded feature representation and only the indices of these encoded features in the codebook are transmitted. Different from [23] where the codebook is for a specific task with single modality, we design a unified codebook for multimodal data. Furthermore, we develop a unified decoder for different tasks, where a masked cross-attention method is proposed for parallel training. Simulation results show that our proposed methods achieve comparable performance to the task-oriented semantic communication systems designed for a specific task with much reduced transmission overhead and fewer model parameters.

Specifically, the main contributions of this paper can be summarized as follows.

- We propose a unified semantic communication framework, U-DeepSC, which can handle a number of tasks using a fixed model. The proposed U-DeepSC is a general framework that can support the transmission of three modalities of data.
- We design FSM at the channel encoder, which is empowered with the ability to dynamically adjust the numbers of transmitted features under different channel conditions and tasks. It is able to achieve an adaptive tradeoff between transmission rate and task performance for the considered tasks. Since FSM can hierarchically prune redundant feature vectors, the computation complexity is reduced and inference speed can be accelerated.
- We develop a unified codebook for multi-task services to support digital communication and reduce transmission overhead. Specifically, we introduce vector quantized-variational mechanism for discrete feature representations, along with the utilization of a digital modulation module for digital transmission.
- We devise a unified decoder based on Transformer decoder, where a novel masked cross-attention method is proposed to achieve parallel training. Besides, we also propose a novel two-phase training algorithm to simultaneously learn multiple tasks.

TABLE I
DEFINITIONS OF NOTATIONS.

Notation	Definition
$\boldsymbol{s}^v$, $\boldsymbol{s}^t$, $\boldsymbol{s}^s$	input image, text, and speech signals
L_v, L_t, L_s	length values of image, text, and speech signals
$\boldsymbol{z}^v$, $\boldsymbol{z}^t$, $\boldsymbol{z}^s$	encoded input image, text, and speech features
K_v, K_t, K_s	numbers of channel uses
$\boldsymbol{y}^v$, $\boldsymbol{y}^t$, $\boldsymbol{y}^s$	received input image, text, and speech features
$\boldsymbol{\theta}_v$, $\boldsymbol{\theta}_t$, $\boldsymbol{\theta}_s$	trainable parameters of transmitters
f_v, f_t, f_s	mappings of image, text, and speech transmitter
$\boldsymbol{\phi}_u$, f_u	trainable parameters and mapping of receiver
$h^i, \boldsymbol{n}^i$, $i \in \{v, t, s\}$	channel gain coefficients and noises
$\boldsymbol{X}^v, \boldsymbol{X}^t, \boldsymbol{X}^s$	preprocessed feature matrices
E_v, E_t, E_s	length values of feature vectors in $\boldsymbol{X}^v, \boldsymbol{X}^t, \boldsymbol{X}^s$
N_v, N_t, N_s	numbers of feature vectors in $\boldsymbol{X}^v, \boldsymbol{X}^t, \boldsymbol{X}^s$
$\boldsymbol{w}_{tk}^v, \boldsymbol{w}_{tk}^t, \boldsymbol{w}_{tk}^s$	task-specific embedding vectors
$\hat{\boldsymbol{X}}^v, \hat{\boldsymbol{X}}^t, \hat{\boldsymbol{X}}^s$	feature matrices with $\boldsymbol{w}_{tk}^v, \boldsymbol{w}_{tk}^t, \boldsymbol{w}_{tk}^s$
$\boldsymbol{X}_{se}^v, \boldsymbol{X}_{se}^t, \boldsymbol{X}_{se}^s$	output feature matrices of semantic encoder
$\boldsymbol{X}_{en}^v, \boldsymbol{X}_{en}^t, \boldsymbol{X}_{en}^s$	output feature matrices of channel encoder
$\boldsymbol{F}_g, \boldsymbol{F}_l, \boldsymbol{F}_n$	global, local, noise feature mapping matrices
$\boldsymbol{F}_c$	comprehensive feature matrix
$\boldsymbol{F}^q, \boldsymbol{m}^q$	q-th feature matrix and q-th mask vector
N_d	number of feature vector in $\boldsymbol{F}^q$
$\hat{\boldsymbol{m}}$	final mask vector
$\boldsymbol{P}$, $\boldsymbol{o}$	sampling probability matrix and one-hot vector
$\boldsymbol{e}_j^i, i \in \{v, t, s\}$	j-th basis vector of codebooks
N_q	number of FSM
f_{vq}	mapping for selecting basis vectors in codebook
f_n	mapping for prefined masking ratio
δ	predefined ratio if transmitted feature vectors
$\boldsymbol{U}_{cd}$	feature matrix after channel decoder
$\boldsymbol{U}_{de}^i$	feature matrix at i-th semantic decoder layer
$\boldsymbol{W}_{ry}$	task-specific query matrix
$\boldsymbol{A}, \boldsymbol{M}$	attention score and attention mask matrices
$\boldsymbol{B}$	cross-attention matrix

C. Organization and Notations

The rest of this paper is organized as follows. Section II introduces the framework of U-DeepSC and the considered tasks. Section III presents the detailed architecture of semantic encoder. Section IV introduces the dynamic channel encoder and FSM. The unified codebook and decoder are introduced in Section V. The masked cross-attention for parallel training and training method are provided in Section VI. Simulation results are presented in Section VII. Finally, Section VIII concludes this paper.

Notations: Scalars, vectors, and matrices are respectively denoted by lower case, boldface lower case, and boldface upper case letters. For a matrix $\boldsymbol{A}$, $\boldsymbol{A}^\top$ and $\|\boldsymbol{A}\|$ denote its transpose, conjugate, and Frobenius norm, respectively. For a vector $\boldsymbol{a}$, $\|\boldsymbol{a}\|$ is its Euclidean norm. Additionally, $\odot$ is the element-wise multiplication of two matrices, i.e., Hadmard product. Finally, $\mathbb{C}^{m\times n}$ ($\mathbb{R}^{m\times n}$) are the space of $m \times n$ complex (real) matrices. Most of the key notations are listed in Table I.

II. FRAMEWORK OF UNIFIED SEMANTIC COMMUNICATION

In this section, we design the framework of U-DeepSC. The U-DeepSC consists of the semantic/channel encoders for each modality, and the unified semantic/channel decoder with light-weight task-specific heads.

A. System Model

As shown in Fig. 1, the proposed U-DeepSC is able to handle a number of tasks with three modalities. We assume that there is a candidate task set that contains various tasks, e.g. $\mathrm{T1}, \mathrm{T2}, ..., \mathrm{T10}$. These tasks may involve different numbers of modalities, including image, text, and speech. We then take T4 that contains image, text, and speech data as an example to present the overall process of U-DeepSC. In particular, after determining the involved modality, the proposed framework will activate the corresponding transmitters based on the selected modalities. It mainly consists of four parts: an image transmitter, a text transmitter, a speech transmitter, and a unified receiver. DNNs are employed to implement the transmitters and the unified receiver. In particular, each transmitter consists of a semantic encoder and a dynamic channel encoder. The unified receiver consists of the channel decoder and the semantic decoder.

The inputs of the system are image $\boldsymbol{s}^v \in \mathbb{R}^{L_v}$, text $\boldsymbol{s}^t \in \mathbb{R}^{L_t}$, and speech $\boldsymbol{s}^s \in \mathbb{R}^{L_s}$, where L_v, L_t, and L_s are the length values of image, text, and speech signals, respectively. Specifically, L_v is the size of an image, L_t signifies the word count in a sentence, and L_s is the number of sampling points in the speech signal. Subsequently, each of these signals undergoes processing to obtain the corresponding encoded features. For simplicity in notation, we treat the semantic encoder and channel encoder collectively. In particular, each signal undergoes initial encoding by the semantic encoder to extract semantic features. Subsequently, the dynamic channel encoder is designed to execute feature selection, regulating the size of the channel symbols. As a result, the encoded features post the channel encoders for the three data modalities are denoted as $\boldsymbol{z}^v \in \mathbb{C}^{K_v}$, $\boldsymbol{z}^t \in \mathbb{C}^{K_t}$, and $\boldsymbol{z}^s \in \mathbb{C}^{K_s}$, where K_v, K_t, and K_s denote the numbers of channel uses of image, text, and speech signals, respectively. The encoding process can be expressed as

$$\boldsymbol{z}^i = f_i(\boldsymbol{s}^i; \boldsymbol{\theta}_i), i \in \{v, t, s\}, \tag{1}$$

where $f_v(\cdot)$, $f_t(\cdot)$, and $f_s(\cdot)$ denote the mappings of image, text, and speech transmitters, respectively. Additionally, $\boldsymbol{\theta}_v$, $\boldsymbol{\theta}_s$, and $\boldsymbol{\theta}_t$ are the trainable parameters in image, text, and speech transmitters, respectively. We define the bandwidth ratios for three modalities as $\rho_i \triangleq K_i/L_i, i \in \{v, t, s\}$. Moreover, the transmitted signals are subject to a power constraint P, i.e., $\frac{1}{K_i}\mathbb{E}\|\boldsymbol{z}^i\|^2 \leq P$, $i \in \{v, t, s\}$.

The encoded vectors, $\boldsymbol{z}^i$, $i \in \{v, t, s\}$, are then sent to the receiver for decoding, and the received vectors are denoted as $\boldsymbol{y}^i, i \in \{v, t, s\}$. Specifically, the received vector at the receiver is given by

$$\boldsymbol{y}^i = h^i \boldsymbol{z}^i + \boldsymbol{n}^i, i \in \{v, t, s\}, \tag{2}$$

where $h^i \in \mathbb{C}$ represents the channel gain coefficient and $\boldsymbol{n}^i \sim \mathcal{CN}(\boldsymbol{0}, \sigma^2)$ is the additive white Gaussian noise (AWGN).

At the receiver, the decoder firstly processes the corrupted complex-valued signal and the obtained features are further processed by light-weight task-specific head to execute downstream tasks. The decoding process can be formulated as

$$\hat{\boldsymbol{s}} = f_u(\boldsymbol{y}^v, \boldsymbol{y}^t, \boldsymbol{y}^s; \boldsymbol{\phi}_u), \tag{3}$$

where $f_u(\cdot)$ denotes the mapping of the receiver, including channel and semantic decoder, and $\boldsymbol{\phi}_u$ is the trainable pa-

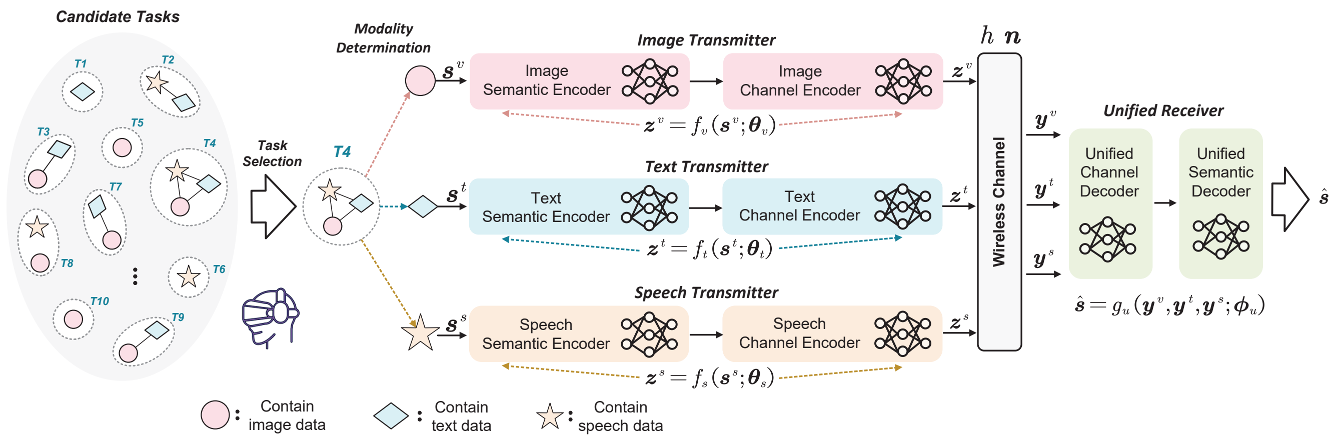


Fig. 1. Framework of the proposed U-DeepSC.

rameters of the receiver. Moreover, as different tasks may involve different modalities of data, there might not always be a necessity to simultaneously feed all three modalities of data into $f_u(\cdot)$. For instance, for tasks like image classification that solely require image data, only $\boldsymbol{y}^v$ is necessary.

B. Task Description

To conduct a comprehensive analysis of U-DeepSC and present compelling results illustrating its efficacy, we will address a diverse set of significant tasks across various domains. These tasks encompass text classification, visual question answering, image classification, video sentiment analysis, image data reconstruction, and text data reconstruction. They involve three fundamental data modalities: image, text, and speech, which constitute a substantial component of contemporary multimedia communication. Additionally, we include cross-modality tasks such as visual question answering, aligning with prevalent themes in existing semantic communication research. The selected tasks serve as a representative demonstration of the proposed scheme's effectiveness, with their relevance evident in established semantic communication systems [9], [21], [24], [30].

(i) **Text classification:** The purpose of the text classification task is to classify whether the sentiment of a given sentence is positive or negative. It is essentially a binary classification problem. Thus, we take classification accuracy as the performance metric for text classification, and the cross-entropy as the loss function to train the model.

(ii) **Visual question answering:** In visual question answering task, the images and questions in text are processed by the model to classify which answer is right. Thus, we take answer accuracy as the performance metric and the cross entropy as the loss function.

(iii) **Video sentiment analysis:** The video sentiment analysis task is about leveraging multimodal signals for an effective understanding of the videos generated by users [37]. As for the evaluation criterion, classification accuracy is selected as the metric. Additionally, the binary cross-entropy is used for the loss function.

(iv) **Image classification:** The image classification task aims at classifing which category the given image belongs to. To evaluate the performance of image classification task, the classification accuracy is adopted as the performance evaluation metric. To learn this task, we adopt the cross-entropy loss function.

(v) **Image reconstruction:** The performance of the image reconstruction task is quantified by the peak signal-to-noise ratio (PSNR). The PSNR measures the ratio between the maximum possible power and the noise, which is given by

$$\mathrm{PSNR} = 10 \log_{10} \frac{\mathrm{MAX}^2}{\mathrm{MSE}} (\mathrm{dB}), \tag{4}$$

where $\mathrm{MSE} = d(\boldsymbol{x}, \hat{\boldsymbol{x}})$ denotes the mean square error (MSE) between the source image $\boldsymbol{x}$, and the reconstructed image $\hat{\boldsymbol{x}}$, and MAX is the maximum possible value of the pixels. Moreover, the MSE is adopted as the training loss.

(vi) **Text reconstruction:** As for the text reconstruction task, the bi-lingual evaluation understudy (BLEU) score is adopted to measure the performance. The BLEU takes the n-gram matching criterion to measure the performance. BLEU score is a scalar between 0 and 1, which evaluates the similarity between the reconstructed text and the source text, with 1 representing the highest similarity. We take the cross entropy as the loss function since the BLEU score is non-differentiable.

III. Semantic Encoding Design for Multimodal Data

In this section, we elaborate the detailed design of the transmitter, including the semantic encoders for three modalities of data. Since the data from different modalities have totally different statistical characteristics and have different semantic information, we have to design image, text, and speech semantic encoders for image, text, and speech, respectively. The detailed architecture of the transmitter is shown in Fig. 2. Specifically, the overall process of transmitter is depicted as

$$\boldsymbol{s}^i \rightarrow \boldsymbol{X}^i \xrightarrow{\boldsymbol{w}^i_{tk}} \hat{\boldsymbol{X}}^i \rightarrow \boldsymbol{X}^i_{se} \xrightarrow{\boldsymbol{m}^q} \boldsymbol{X}^i_{en}, \; i \in \{v, t, s\}, \tag{5}$$

where the definitions of notations can be found in the following sections.

A. Transformer-Based Encoding

We mainly use Transformer encoder to construct the encoders of the transmitter. The Transformer encoder constitutes the fundamental building block in the landscape of signal processing [38]–[40]. The architecture of a basic Transformer encoder layer is depicted in the upper part of Fig. 2. Each Transformer encoder layer, as illustrated in Fig. 2, comprises a multi-head self-attention (MSA) module and a multi-layer perceptron (MLP) layer. The MSA mechanism enables the model to capture long-range dependencies among different feature vectors, facilitating efficient global information integration. It assigns higher weights to relevant feature vectors while simultaneously attending to different positions in the sequence. After the MSA, the feed forward neural (FFN) layer introduces non-linearity and captures complex relationships within the feature vectors.

As depicted in Fig. 2, we input the intermediate feature matrix $\boldsymbol{F}_{in} \in \mathbb{R}^{L\times D}$ into the Transformer encoder layer. The generated feature matrix is represented by $\boldsymbol{F}_{out} \in \mathbb{R}^{L\times D}$, and both $\boldsymbol{F}_{in}$ and $\boldsymbol{F}_{out}$ share the shape $L \times D$. Overall, the encoding procedure of a Transformer encoder layer [39] can be denoted as

$$\boldsymbol{F}_{out} = MSA(\boldsymbol{F}_{in}) + FFN(MSA(\boldsymbol{F}_{in})). \tag{6}$$

Furthermore, GeLU activation and layer normalization operations are employed prior to MSA and MLP modules.

The MSA module consists of N_a self-attention modules with a residual connection, i.e.,

$$MSA(\boldsymbol{X}_{in}) = \boldsymbol{F}_{in} + concat\underbrace{\left(SA(\boldsymbol{F}_{in}), SA(\boldsymbol{F}_{in}), ..., SA(\boldsymbol{F}_{in})\right)}_{N_a}\boldsymbol{W}, \tag{7}$$

where $concat(\cdot)$ denotes the concatenation operation, $SA(\boldsymbol{F}_{in}) \in \mathbb{R}^{L\times D_a}$ is self-attention operation on $\boldsymbol{F}_{in}$, $\boldsymbol{W} \in \mathbb{R}^{D_a N_a \times D}$ is linear projection matrix, and $D_a = D/N_a$ is the output dimension. Moreover, $SA(\cdot)$ is formulated as

$$SA(\boldsymbol{F}_{in}) = softmax\left(\frac{\boldsymbol{Q}\boldsymbol{K}^{\top}}{\sqrt{D}}\right)\boldsymbol{V}, \tag{8}$$

where $\boldsymbol{Q} \in \mathbb{R}^{L\times D_a}$, $\boldsymbol{K} \in \mathbb{R}^{L\times D_a}$, and $\boldsymbol{V} \in \mathbb{R}^{L\times D_a}$ are generated from three linear projections as

$$\boldsymbol{Q} = \boldsymbol{F}_{in}\boldsymbol{W}_Q, \quad \boldsymbol{K} = \boldsymbol{F}_{in}\boldsymbol{W}_K, \quad \boldsymbol{V} = \boldsymbol{F}_{in}\boldsymbol{W}_V, \tag{9}$$

where $\boldsymbol{W}_Q \in \mathbb{R}^{D\times D_a}$, $\boldsymbol{W}_K \in \mathbb{R}^{D\times D_a}$, $\boldsymbol{W}_V \in \mathbb{R}^{D\times D_a}$, and $softmax(\cdot)$ denotes the Softmax function. Moreover, all the introduced projection matrices are set as trainable parameters. In addition to MSA, the FFN module consists of a multi-layer MLP and a GeLU activation function.

B. Semantic Encoder Design

1) Image Semantic Encoder: The image-only and multi-modal tasks take an image $\boldsymbol{s}^v$ as input. Then, it is preprocessed [39] through patch embedding into the preliminary features $\boldsymbol{X}^v = [\boldsymbol{x}_1^v, \boldsymbol{x}_2^v, \ldots, \boldsymbol{x}_{N_v}^v]^{\top} \in \mathbb{R}^{N_v\times E_v}$, where N_v and E_v denote the number and length of the feature vectors, respectively. Additionally, for the video tasks that involve processing multiple frames of images at one time, we concatenate the preliminary features of different frames as the input of the image semantic encoder. U-DeepSC is designed as multi-task model that can simultaneously deal with several tasks with just a set of parameters. Besides, there is no need to update the parameters when the task changes. In this case, the encoder of U-DeepSC needs to identify the current task so that it can perform the feature extraction pertinently.

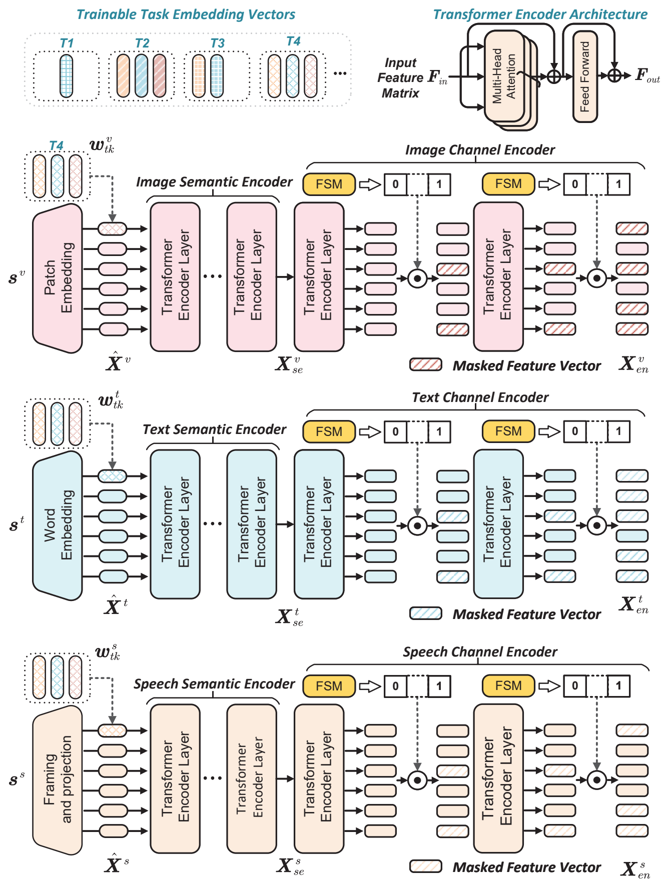


Fig. 2. The transmitter architecture of the proposed U-DeepSC. It mainly consists of three transmitters and each transmitter is comprised of two parts: semantic encoder and channel encoder. The task embedding vectors consist of randomly generated trainable parameters, which are continually updated along with the model. Additionally, multiple tasks will share a common transmitter.

To this end, we introduce the task embedding vector to the semantic encoder, which is generated as trainable parameter and updated with the whole model. Specifically, we would generate one task embedding vector for each task at the image encoder. For instance, in the case of two tasks related to image data, two task embedding vectors are randomly generated. It is worth noting that the task embedding vector is quite similar to the CLS token embedding vector in the vanilla transformer [38], [39]. This is achieved by generating a set of random trainable parameter vectors, as illustrated in Fig. 2, and jointly learning them across the entire network. During the training phase, we select the task embedding vector corresponding to the specific task and input it along with the image. This action signifies the task to be performed, enabling the semantic encoder to extract task-specific information. In the inference phase, the learned task embedding vectors are selected for specific tasks. For instance, if a task involves three modalities, three learned task embedding vectors, such as T4, are selected. As shown in Fig. 2, the task embedding vector $\boldsymbol{w}_{tk}^v \in \mathbb{R}^{E_v}$ is added to the semantic encoder. It is concatenated with $\boldsymbol{X}^v$, and the concatenated feature matrix $\hat{\boldsymbol{X}} \in \mathbb{R}^{(N_v+1)\times E_v}$ is given by

$$\hat{\boldsymbol{X}}^v = concat\left(\boldsymbol{X}^v, \boldsymbol{w}^v_{tk}\right). \tag{10}$$

Particularly, the task embedding vectors are employed to perform convolution or attention operation together with the encoded image feature vectors [41]. Then, we obtain the encoded image feature matrix through the image semantic encoder, which is represented by $\boldsymbol{X}^v_{se}$.

2) Text Semantic Encoder: As for text-only and multimodal tasks, we preprocess the input text by word embedding into a sequence of N_t feature vectors, $\boldsymbol{X}^t = [\boldsymbol{x}^t_1, \boldsymbol{x}^t_2, \ldots, \boldsymbol{x}^t_{N_t}]^\top \in \mathbb{R}^{N_t\times E_t}$, where N_t and E_t denote the number and length of the feature vectors, respectively. Subsequently, $\boldsymbol{X}^t$ is encoded by the text semantic encoder. Similar to the image semantic encoder, we also add a trainable task embedding vector $\boldsymbol{w}^t_{tk} \in \mathbb{R}^{E_t}$ by concatenating it with $\boldsymbol{X}^t$. Then, the concatenated sequence $\hat{\boldsymbol{X}}^t \in \mathbb{R}^{(N_t+1)\times E_t}$ is input to the text semantic encoder, which generates the encoded text feature matrix $\boldsymbol{X}^t_{se} \in \mathbb{R}^{(N_t+1)\times E_t}$.

3) Speech Semantic Encoder: As for speech-only and multimodal tasks, the input speech signal of the proposed system is obtained from the speech set. Similarly, we first preprocess speech signal $\boldsymbol{s}^s$ into $\boldsymbol{X}^s = [\boldsymbol{x}^s_1, \boldsymbol{x}^s_2, \ldots, \boldsymbol{x}^s_{N_s}]^\top \in \mathbb{R}^{N_s\times E_s}$, where N_s and E_s denote the number and length of the feature vectors, respectively. Then, the speech semantic encoder learns the N_s encoded speech feature matrix $\boldsymbol{X}^s_{se} \in \mathbb{R}^{(N_s+1)\times E_s}$ from the concatenated sequence $\hat{\boldsymbol{X}}^s = concat\left(\boldsymbol{X}^s, \boldsymbol{w}^s_{tk}\right) \in \mathbb{R}^{(N_s+1)\times E_s}$, where $\boldsymbol{w}^s_{tk} \in \mathbb{R}^{E_s}$ is the task embedding vector trained with the whole network.

IV. Feature Selection-Based Dynamic Channel Encoder

In this section, we introduce the task-specific dynamic overhead by developing the dynamic channel encoder with FSM.

A. Hierarchical Feature Selection

Transmitting all the feature vectors introduces excessive redundancy of semantic information and different tasks require different numbers of transmitted features. Thus, the transmission overhead in U-DeepSC can be reduced by selecting a specific number of feature vectors for each task. Although excessive redundancy generally induces a high transmission overhead and latency, it leads to better performance against noise if more encoded features are transmitted. It is mainly because when certain features are seriously disturbed, the other features that are not disturbed can help to maintain the performance. Therefore, we need to balance the performance and the number of transmitted symbols. To achieve this goal, we design a channel encoder to dynamically adjust the number of output feature vectors for different tasks under different channel conditions in U-DeepSC, which is able to dynamically achieve satisfactory performance by transmitting the least number of features.

Particularly, we design a dynamic channel encoder to conduct vector-wise feature selection by adjusting the number of the transmitted feature vectors. The proposed dynamic channel encoder has the following advantages:

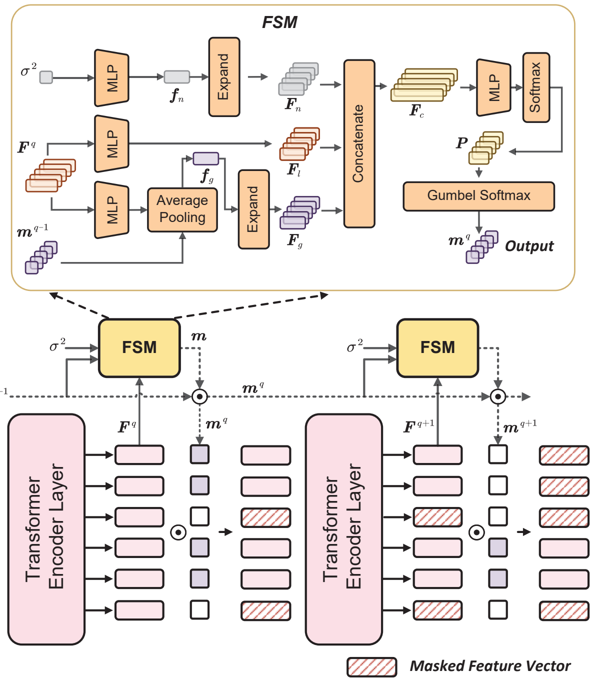


Fig. 3. The architecture of the proposed channel encoder and FSM. The FSM takes σ^2, $\boldsymbol{F}^q$, and $\boldsymbol{m}^{q-1}$ as input and outputs $\boldsymbol{m}^q$.

- The transmitter can identify the task-related features and omit the task-unrelated features, which leads to satisfactory performance.
- In Transformer architecture, each encoded feature corresponds to portion of the input, such as one small patch of the input image or one word of a sentence. Thus, it is also of great interpretability to see which features are task-unrelated.
- With the vector-wise selection design, the transmission overhead can be dynamically adjusted and significantly reduced.

Since the output of the semantic encoder is a sequence of encoded feature vectors. We omit the task-unrelated features and transmit the informative task-related features to the receiver. In particular, as shown in Fig. 3, the FSM is inserted into the original channel encoder layer to generate the mask vector, which is used to indicate the features to be kept/dropped for each task. Inspired by [42], we conduct hierarchical selection by inserting multiple FSMs at certain layers, to perform gradually dropping. Moreover, hierarchical selection gradually drops the relatively unimportant features and can avoid mistakenly dropping important features directly. It is equivalent to dividing a complex selection problem into several simple selection problems. After dropping a small part of features each time, the model can adaptively adjust the next selection according to the existing unmasked features, to achieve a higher tolerance for error selection than the single selection performed at the end of the transmitter.

B. FSM Design

1) Architecture Details: To realize hierarchical feature selection, i.e., gradually drop the uninformative feature vectors, we introduce the selection mask vector, $\boldsymbol{m}^q$ =

$[m_1^q, m_2^q, ..., m_{N_d}^q] \in \{0,1\}^{N_d}$, which indicates whether to transmit each feature vector and N_d is the number of the encoded feature vectors. The elements in the selection mask vector are firstly initialized to 1 and updated progressively in the N_q FSMs, as shown in Fig. 3. We denote the feature matrix $\boldsymbol{F}^q = [\boldsymbol{f}_1^q, \boldsymbol{f}_2^q, \dots, \boldsymbol{f}_{N_d}^q]^\top \in \mathbb{R}^{N_d \times E_d}$, as the input of the q-th selection module, where N_d and E_d denote the number and the size of the feature vectors, respectively. The q-th selection module takes the previous selection mask vector $\boldsymbol{m}^{q-1}$ and $\boldsymbol{F}^q$, as input, and outputs the probabilities of keeping the feature vectors.

Firstly, the encoded feature vectors are projected by a MLP: $\boldsymbol{F}_l = \mathit{MLP}(\boldsymbol{F}^q) \in \mathbb{R}^{N_d \times E_l}$, where $\mathit{MLP}(\cdot)$ denotes the MLP layer, $\boldsymbol{F}_l$ denotes the information feature matrix with a smaller size than the original feature matrix, and E_l is smaller than E_d. In addition, to determine whether to drop or keep a feature vector, we also need to consider the information of the existing feature vectors that are not masked. It is employed to perform information fusion with $\boldsymbol{F}_l$ to evaluate the information loss caused by the mask operation. Therefore, the global feature vector $\boldsymbol{f}_g$ that aggregates the information of all existing feature vectors that have not been masked can be computed by

$$\boldsymbol{f}_g = g\big(\mathit{MLP}(\boldsymbol{F}^q), \boldsymbol{m}^{q-1}\big) \in \mathbb{R}^{E_l}, \tag{11}$$

where $g(\cdot)$ is the function to extract the information of the existing feature vectors. $g(\cdot)$ can be implemented by the average pooling

$$g(\boldsymbol{F}, \boldsymbol{m}) = \sum_{i=1}^{N_d} m_i \cdot \boldsymbol{f}_i / \sum_{i=1}^{N_d} m_i, \tag{12}$$

where i is the index of the vector, $\boldsymbol{F} = [\boldsymbol{f}_1, \boldsymbol{f}_2, \dots, \boldsymbol{f}_{N_d}]^\top$, and $\boldsymbol{m} = [m_1, m_2, \dots, m_{N_d}]^\top$. We further expand $\boldsymbol{f}_g$ as $\boldsymbol{F}_g = [\boldsymbol{f}_g, \boldsymbol{f}_g, \dots, \boldsymbol{f}_g]^\top \in \mathbb{R}^{N_d \times E_l}$ to perform knowledge fusion between $\boldsymbol{F}_g$ and $\boldsymbol{F}_l$. The channel noise also affects probability of keeping the feature vectors, we employ another MLP layer to extract the noise feature $\boldsymbol{f}_n = \mathit{MLP}(\sigma^2) \in \mathbb{R}^{E_l}$, where σ^2 is the noise variance. Then, we expand $\boldsymbol{f}_n$ in the same way as $\boldsymbol{f}_g$ to obtain $\boldsymbol{F}_n \in \mathbb{R}^{N_d \times E_l}$. As a result, we combine the local, global, and noise feature to obtain comprehensive feature matrix

$$\boldsymbol{F}_c = \mathit{concat}\left(\boldsymbol{F}_l, \boldsymbol{F}_g, \boldsymbol{F}_n\right) \in \mathbb{R}^{N_d \times 3E_l}, \tag{13}$$

and feed it to another MLP to predict the probabilities of keeping the feature vectors,

$$\boldsymbol{P} = \mathit{softmax}(\mathit{MLP}(\boldsymbol{F}_c)) \in \mathbb{R}^{N_d \times 2}. \tag{14}$$

Then, we can obtain the current mask vector $\boldsymbol{m}$ by sampling from $\boldsymbol{P}$. Note that this module will be inserted into different layers of the channel encoder to perform gradually dropping feature vectors. Accordingly, the true mask vector $\boldsymbol{m}^q$ at the next layer is updated by $\boldsymbol{m}^q = \boldsymbol{m}^{q-1} \odot \boldsymbol{m}$, where $\odot$ denotes Hadamard product. Therefore, once a feature is dropped, it will never be used in the following layers.

2) Differentiable Sampling: Although the design above is able to perform feature vector selection, it is difficult to implement during training. The main obstacle is that the sampling operation from $\boldsymbol{P}$ to obtain the selection mask vector $\boldsymbol{m}$ is non-differentiable. Note that $\boldsymbol{P} = [\boldsymbol{p}_1, \boldsymbol{p}_2, \dots, \boldsymbol{p}_{N_d}]^\top \in \mathbb{R}^{N_d \times 2}$ denotes the probability matrix and the elements of the first rank and second rank are the probabilities of keeping and dropping these N_d feature vectors, respectively. We take the j-th feature vector $\boldsymbol{f}_j^q$ in $\boldsymbol{F}^q$ as an example, and a straightforward way to determine whether to keep it is to sample from $\boldsymbol{p}_j \in \mathbb{R}^2$. However, the sampling operation is non-differentiable, which hinders the back propagation of the gradients. To address this problem, the Gumbel-Softmax technique is adopted. It is differentiable and makes it possible to train the selection module. Firstly, with the Gumbel-Max, discrete one-hot sample from the distribution, $\boldsymbol{p}_j = [p_j^1, p_j^2]$, can be expressed as

$$\boldsymbol{o} \triangleq [o_1, o_2] = \mathit{oneshot}\Big(\arg\max_{i=1,2}\big(\log(p_j^i) + g_i\big)\Big), \tag{15}$$

where g_i for $i = 1, 2$ are samples drawn from $\mathit{gumbel}(0,1)$, the operation $\mathit{oneshot}(n)$ denotes generating a one-hot vector, where the n-th element equals 1. For instance, $\mathit{oneshot}(1) = [1, 0]$. The $\mathit{gumbel}(0,1)$ distribution can be sampled using inverse transform sampling by drawing u from a uniform distribution $U(0,1)$ and computing $g = -\log(-\log(\mathrm{u}))$.

To approximate the non-differentiable $\arg\max(\cdot)$, we further use the softmax function as a continuous, differentiable approximation [43]:

$$\hat{o}_i = \frac{\exp\left(\left(\log(p_j^i) + g_i\right)/\tau\right)}{\sum_{k=1}^{2} \exp\left(\left(\log(p_j^k) + g_k\right)/\tau\right)}, \tag{16}$$

for $i = 1, 2$, where τ is a temperature parameter that controls the discreteness. As the softmax temperature τ approaches 0, sample from the Gumbel distribution, i.e., $\hat{\boldsymbol{o}} = [\hat{o}_1, \hat{o}_2]$, becomes one-hot. At higher temperatures, it is no longer one-hot, and becomes uniform. It is differentiable and makes it possible to train the selection module. Therefore, the i-th element in selection vector $\boldsymbol{m}$ can be written as

$$m_i = \hat{o}_1 = \mathit{gumbel\text{-}softmax}\ (\boldsymbol{p}_i)[1]. \tag{17}$$

In particular, the output of $\mathit{gumbel\text{-}softmax}(\cdot)$ is an approximate one-hot vector with the same shape as the input, where element 1 indicates the sampled result. Thus, the output one-hot vector $\mathit{gumbel\text{-}softmax}(\boldsymbol{p}_i)$ has two elements, and we take its first element as the result. Therefore, $\boldsymbol{m}$ will be a mask vector sampled from $\boldsymbol{F}^q$, with each element being 0 or 1.

3) Training and Inference: The training process for U-DeepSC encompasses the dynamic channel encoder, ensuring its capability to make decisions regarding the transmission of specific feature vectors. To optimize dynamic transmission overhead across varying channel conditions, the ratio of transmitted feature vectors is governed by a predetermined variable value $\delta = f_n\left(\sigma^2; \boldsymbol{\psi}\right)$, where $\boldsymbol{\psi}$ denotes the trainable parameters and σ^2 denotes the variable channel noise. In practical communication systems, the receiver has the capability to gain awareness of channel conditions through channel feedback. Consequently, the encoding of features can be enhanced by integrating the knowledge of channel conditions into the process. In particular, when $f_n\left(\sigma^2; \boldsymbol{\psi}\right)$ is configured as a non-negative, monotonically increasing function implemented by DNNs, and a greater number of feature vectors will be chosen for larger σ^2. To implement this non-negative increasing function, we employ a series of multi-layer perceptron (MLP) layers. The mapping of an MLP layer can be represented by

$$\mathit{MLP}(\boldsymbol{x}) = \mathit{tanh}(\boldsymbol{W}\boldsymbol{x}), \tag{18}$$

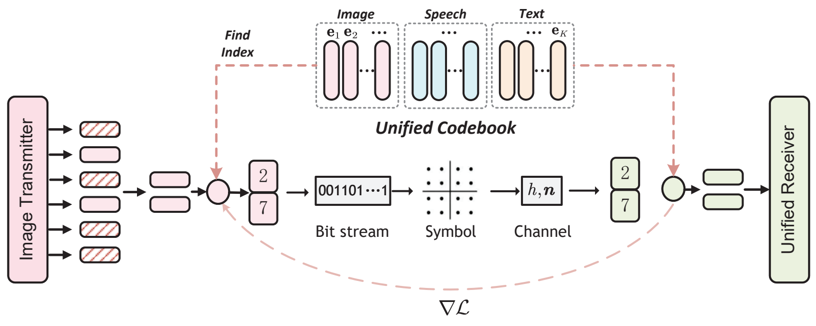


Fig. 4. The unified codebook for digital data transmission.

where we employ $tanh(\cdot)$ as the activation function, $\boldsymbol{W}$ denotes the trainable parameters, and $\boldsymbol{x}$ represents the input vector. Since we aim to design f_n as a non-negative increasing function with respect to the channel noise, we let $\boldsymbol{W} = abs(\bar{\boldsymbol{W}})$ for each MLP layer, where $\bar{\boldsymbol{W}}$ is the actual trainable parameters of each MLP layer. The monotonicity can be easily acknowledged by deriving the derivatives with respect to $\boldsymbol{x}$.

Given the specific channel noise σ^2, we obtain a set of target ratios for N_q corresponding selection modules, i.e., $[\delta, \delta^2, ..., \delta^{N_q}]$. We apply the MSE loss to supervise the prediction module:

$$\mathcal{L}_{FSM} = \frac{1}{N_q}\sum_{q=1}^{N_q}\left(\delta^q - \frac{1}{N_d}\sum_{i=1}^{N_d} m_i^q\right)^2, \qquad (19)$$

where $\frac{1}{N_d}\sum_{i=1}^{N_d} m_i^q$ denotes the true ratio of transmitting feature vectors. By supervising the true ratio with the target ratio, only the target ratio of feature vectors will be processed by the decoder for training. However, if we apply loss (19) directly, more feature vectors will be selected as the training goes on, since the model tends to keep more feature vectors to improve the performance. To balance the performance and the number of transmitted symbols, we add the l_1-norm of δ to loss (19), as given by

$$\mathcal{L}_{FSM} = \frac{1}{N_q}\sum_{q=1}^{N_q}\left(\delta^q - \frac{1}{N_d}\sum_{i=1}^{N_d} m_i^q\right)^2 + \|\delta\|. \qquad (20)$$

It makes the model keep fewer feature vectors with the decrement of δ, and enables the model to achieve a good performance by only transmitting a part of feature vectors.

During the inference phase, with knowledge of the channel noise variance, we can selectively discard less informative feature vectors based on probabilities generated by the selection modules. This ensures that only $N_d\delta^{N_q}$ feature vectors are transmitted to the receiver. In particular, only the top $N_d\delta^{N_q}$ feature vectors with the highest probabilities are transmitted. In addition, only the important parts of the features are retained after FSM, while the unimportant parts of feature vectors are dropped directly. Furthermore, the FSM retains only the essential features, discarding unimportant parts of the feature vectors. As a result, the transmitter transmits only the retained feature vectors without the need to transmit the mask. At the receiver, the received feature vectors are concatenated and directly fed into the decoder.

V. Unified Codebook and Receiver

In this section, we first design the unified codebook for the multi-task applications. Then, we design the unified receiver based on Transformer decoder.

A. Unified Codebook for Multimodal Data

We aim to design a unified codebook for all considered tasks with different modalities of data to enable digital transmission. The codebook consists of a number of basis vectors and the encoded feature vectors will be represented by the basis vectors. Since the data characteristics of different modalities differ from each other, it is difficult to share the basis vectors among the data of different modalities. In contrast, the semantic information of different tasks from the same data modality may overlap. For example, the image reconstruction task and image classification task can share some basis vectors since the class semantic information must be included in global semantic information, which can be employed to reconstruct the data. It poses a potential to share the basis vectors in the codebook among these tasks, which leads to a much smaller codebook size for multiple tasks. Therefore, we aim to design a unified codebook with three sub-codebooks for the image, text, and speech, as shown in Fig. 4(a). Different tasks of the same modality share the same sub-codebook.

As presented in Fig. 4(a), we design a unified codebook as $\mathcal{E} \triangleq \{\mathcal{E}^v, \mathcal{E}^t, \mathcal{E}^s\}$. Specifically,

$$\mathcal{E}^i \triangleq \{\boldsymbol{e}_j^i\}_{j=1}^{M_i}, i \in \{v, t, s\}, \qquad (21)$$

where M_v, M_t, and M_s are the sizes of the image sub-codebook $\mathcal{E}^v$, text sub-codebook $\mathcal{E}^t$, and speech sub-codebook $\mathcal{E}^s$ respectively. $M_c = M_v + M_t + M_s$, is the total size of the unified codebook. Then, we take the image data as an example to show the way to represent the encoded feature vectors with the codebook. Recall that the input data $\boldsymbol{s}^v$ passes through the encoder to produce the encoded feature matrix $\boldsymbol{X}_{en}^v = [\boldsymbol{x}_1^v, \boldsymbol{x}_2^v, ..., \boldsymbol{x}_{N_v+1}^v] \in \mathbb{R}^{(N_v+1)\times E_v}$. Then, it is represented by a group of basis vectors. Specifically, each feature vector $\boldsymbol{x}_i^v$ is represented by $f_{vq}(\boldsymbol{x}_i^v)$, which is the nearest vector in the codebook [44],

$$f_{vq}(\boldsymbol{x}_i^v) = \arg\min_{\boldsymbol{e}_j^v \in \mathcal{E}^v} \left\|\boldsymbol{x}_i^v - \boldsymbol{e}_j^v\right\|_2. \qquad (22)$$

The basis vectors in the codebook are trained together with the parameters of encoder and decoder. However, the operation in (22) is non-differentiable. Hence, the gradients are copied from the input of the decoder to the output of the encoder. In this way, the gradient is passed to the encoder to enable the back propagation. In particular, the trainable parameters of the encoder, decoder, and codebook are updated via the loss function below

$$\mathcal{L}_c(\boldsymbol{x}_i^v;\boldsymbol{e}_j^v) = \left\|\mathrm{ng}\left[\boldsymbol{x}_i^v\right] - \boldsymbol{e}_j^v\right\|_2^2 + \beta\left\|\boldsymbol{x}_i^v - \mathrm{ng}\left[\boldsymbol{e}_j^v\right]\right\|_2^2, \quad (23)$$

where β denotes the hyper-parameter, and symbol $\mathrm{ng}\left[\boldsymbol{u}_i^v\right]$ represents stop-gradient operator that has zero gradient during backward propagation and is the identity layer during the forward propagation. Additionally, if the task involves more than one modality, we need to calculate the sum of codebook losses of involved modalities. Since we use the straight gradient estimation of mapping from $\boldsymbol{x}_i^v$ to $f_{vq}(\boldsymbol{x}_i^v)$, the basis vectors $\{\mathbf{e}_j^v, \forall j\}$ have no gradients from the loss function. Therefore, to learn the basis vectors, we employ the l_2 error to move the basis vectors towards the encoded features, $\boldsymbol{x}_i^v$, as shown in the first term of (23). Since the volume of the encoded feature space is dimensionless, the codebook can grow arbitrarily and cause the training process to diverge if the basis vectors, $\{\mathbf{e}_j^v, \forall j\}$, are not trained as fast as the encoder parameters [44]. To address this issue, we add the second term in (23). In summary, the basis vectors are optimized by the first loss term, and the encoder is optimized by the second loss term. These details have been elucidated more explicitly in the revised manuscript.

B. Unified Receiver Design

1) Transformer-Based Decoding: As depicted in Fig. 5, the Transformer decoder primarily consists of two components: MSA and multi-head cross-attention (MCA). In contrast to MSA, MCA focuses on attending to the extra input to capture relevant information. By incorporating both self-attention and cross-attention mechanisms, the Transformer decoder layer can adaptively process the received information. In particular, the decoding procedure of a Transformer can be denoted as

$$\boldsymbol{F}_{out} = \mathit{MSA}(\boldsymbol{F}_{in1}) + \mathit{FFN}(\mathit{MCA}(\mathit{MSA}(\boldsymbol{F}_{in1}), \boldsymbol{F}_{in2})), \quad (24)$$

where $\boldsymbol{F}_{out} \in \mathbb{R}^{L_1 \times D}$ denotes the output matrix, $\boldsymbol{F}_{in1} \in \mathbb{R}^{L_1 \times D}$ and $\mathbf{F}_{in2} \in \mathbb{R}^{L_2 \times D}$ are input matrices. The operation $\mathit{MCA}(\cdot)$ can be described as follows. Firstly, for simplicity, we let $\boldsymbol{F}_1 = \mathit{MSA}(\boldsymbol{F}_{in1})$ and $\boldsymbol{F}_2 = \boldsymbol{F}_{in2}$. Then, similar to MSA, the operation $\mathit{MCA}(\cdot)$ generally takes two matrices, $\boldsymbol{F}_1 \in \mathbb{R}^{L_1 \times D}$ and $\mathbf{F}_2 \in \mathbb{R}^{L_2 \times D}$, as input, and the procedure can be expressed as

$$\begin{aligned}&\mathit{MCA}(\boldsymbol{F}_1, \boldsymbol{F}_2) = \\ &\boldsymbol{F}_1 + \mathit{concat}\underbrace{\left(\mathit{CA}(\boldsymbol{F}_1, \boldsymbol{F}_2), \mathit{CA}(\boldsymbol{F}_1, \boldsymbol{F}_2), ..., \mathit{CA}(\boldsymbol{F}_1, \boldsymbol{F}_2)\right)}_{N_s}\boldsymbol{W}.\end{aligned} \quad (25)$$

Specifically, the cross-attention operation is given by

$$\mathit{CA}(\boldsymbol{F}_1, \boldsymbol{F}_2) = \mathit{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{D}}\right)\boldsymbol{V}, \quad (26)$$

where $\boldsymbol{Q} \in \mathbb{R}^{L_1 \times D_s}$, $\boldsymbol{K} \in \mathbb{R}^{L_2 \times D_a}$, and $\boldsymbol{V} \in \mathbb{R}^{L_2 \times D_a}$ are generated from three linear projections as

$$\boldsymbol{Q} = \boldsymbol{F}_1\boldsymbol{W}_Q, \quad \boldsymbol{K} = \boldsymbol{F}_2\boldsymbol{W}_K, \quad \boldsymbol{V} = \boldsymbol{F}_2\boldsymbol{W}_V, \quad (27)$$

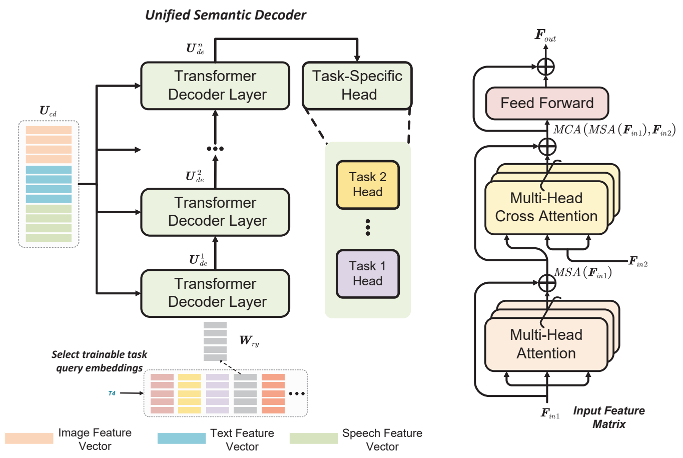


Fig. 5. The receiver architecture of the proposed U-DeepSC.

where $\boldsymbol{W}_Q \in \mathbb{R}^{D \times D_a}$, $\boldsymbol{W}_K \in \mathbb{R}^{D \times D_a}$, and $\boldsymbol{W}_V \in \mathbb{R}^{D \times D_a}$. By connecting multiple Transformer decoder layer, a Transformer decoder is obtained.

2) Unified Semantic Decoder: The received encoded features are firstly processed by the channel decoder, whose output is denoted as $\boldsymbol{U}_{cd}$, as shown in Fig. 5. For image-only tasks, text-only tasks, and speech-only tasks, the input to the decoder can be represented by $\boldsymbol{U}_{cd} = \boldsymbol{U}^v$, $\boldsymbol{U}_{cd} = \boldsymbol{U}^t$, and $\boldsymbol{U}_{cd} = \boldsymbol{U}^s$, respectively, where $\boldsymbol{U}^v$, $\boldsymbol{U}^t$, and $\boldsymbol{U}^s$ denote the decoded image, text, and speech feature matrices processed by channel decoder, respectively. For multimodal tasks, we concatenate the decoded features from the corresponding modalities of data into a sequence, e.g., $\boldsymbol{U}_{cd} = \mathit{concat}\left(\boldsymbol{U}^v, \boldsymbol{U}^t\right)$ for image-and-text tasks.

Unlike the separate design for each modality at the transmitter, the semantic decoder is built upon the unified Transformer decoder structure. As shown in Fig. 5, the semantic decoder takes the output of channel decoder, $\boldsymbol{U}_{cd} \in \mathbb{R}^{N_p \times E_p}$, and the task-specific query matrix, $\boldsymbol{W}_{ry} \in \mathbb{R}^{N_r \times E_p}$, as the input, where N_p is the total number of feature vectors, N_r and E_p denote the number and length of task-specific query embedding vectors, respectively. The task-specific query embedding matrix serves as an indicator of the task assigned to the semantic decoder, akin to the conventional input required by the Transformer decoder [38]. Similar to the task embedding vectors, we would generate several task-specific query embedding matrices, and each of which corresponds to one task. During the testing phase, only the task-specific query embedding matrix relevant to the ongoing task is utilized. To illustrate, if there are a total of 6 tasks, we create 6 trainable task-specific query embedding matrices. In the inference phase, we select the corresponding matrix as input to indicate the intended task, as illustrated in Fig. 5. Let $\boldsymbol{F}_{in1} = \boldsymbol{W}_{ry}$ and $\boldsymbol{F}_{in2} = \boldsymbol{U}_{cd}$. Then, we obtain the output of the i-th decoder layer $\boldsymbol{U}_{de}^i$. Therefore, the output of the first decoder layer $\boldsymbol{U}_{de}^1$ can be denoted by

$$\boldsymbol{U}_{de}^1 = \mathit{MSA}(\boldsymbol{W}_{ry}) + \mathit{FFN}(\mathit{MCA}(\mathit{MSA}(\boldsymbol{W}_{ry}), \boldsymbol{U}_{cd})). \quad (28)$$

Then, for the later layers, the output of the i-th decoder layer $\boldsymbol{U}_i^{de}$ can be iteratively denoted as

$$\boldsymbol{U}_{de}^i = \mathit{MSA}(\boldsymbol{U}_{de}^{i-1}) + \mathit{FFN}(\mathit{MCA}(\mathit{MSA}(\boldsymbol{U}_{de}^{i-1}), \boldsymbol{U}_{cd})). \quad (29)$$

VI. Masked Cross-Attention and Joint Training

In this section, we design the masked cross-attention for parallel training. The overall training algorithm is also introduced.

A. *Masked Cross-Attention for Parallel Training*

Recall that FSM can select informative feature vectors using Gumbel-Softmax sampling. However, the sampling operation is inherently random, leading to an indeterminate number of selected features that can vary across different inputs. Consequently, the received feature matrices $\mathbf{U}_{cd}$, which consist of selected feature vectors for distinct inputs (such as different images), exhibit varying shapes. The inherent variability in feature vector sizes poses a challenge to achieving parallel batch-wise training, as conventional deep learning frameworks often demand uniform feature shapes within a batch. Diverging from the approach presented in [45], which utilizes multiple decoders to handle varying feature vector sizes, our solution for efficient parallel training involves the introduction of a masked cross-attention mechanism.

To this end, we do not directly drop the redundant feature vectors, instead, all feature vectors are kept in the training phase and we only drop the redundant feature vectors in the inference phase feature. Thus, we must keep the number of feature vectors unchanged in the training phase, while cutting down the interactions between the redundant feature vectors and other informative feature vectors. Nevertheless, merely zero-out the redundant feature vectors according to the mask vector is not feasible, since the zeroed feature vectors will still influence other vectors through the Softmax operation, as given in (26). In order to drop the redundant feature vectors, we mask the attention scores calculated by the redundant feature vectors. That is, the main procedure of the masked cross-attention is described as follows. We first denote the attention score matrix as

$$\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{K}^T/\sqrt{D} \in \mathbb{R}^{L_1 \times L_2}. \tag{30}$$

Then, we generate a mask matrix $\boldsymbol{M} \in \mathbb{R}^{L_1 \times L_2}$ at the decoder for $\boldsymbol{A}$ based on the final mask vector $\hat{\boldsymbol{m}} \in \mathbb{R}^{L_2}$:

$$\boldsymbol{M}[i] = \hat{\boldsymbol{m}},\ 1 \leq i \leq L_1. \tag{31}$$

Accordingly, we denote the cross-attention matrix in (26) as $\boldsymbol{B}$, which can be calculated by

$$\boldsymbol{B}[i,j] = \frac{\exp\left(\boldsymbol{A}[i,j]\right)\boldsymbol{M}[i,j]}{\sum_{l=1}^{L_2}\exp\left(\boldsymbol{A}[i,l]\right)\boldsymbol{M}[i,l]}. \tag{32}$$

Particularly, $\boldsymbol{M}[i,j] = 1$ indicates that the j-th feature vector will influence the i-th feature vector. In this way, the redundant feature vectors will not contribute to the informative feature vectors, so that they can be directly dropped in the inference phase.

B. *Joint Training Algorithm*

To jointly learn the considered tasks, we propose an efficient method to train the modules in the U-DeepSC system, which can be divided into two phases.

- In the first phase, we fix parameters of f_n, $\boldsymbol{\psi}$, and jointly train the other parameters. Specifically, we randomly choose a task and sample data from the corresponding dataset. After determining the involved modalities, we activate the required modules and update the parameters of encoders and decoders using the loss $\mathcal{L}\left(\boldsymbol{s}^v, \boldsymbol{s}^t, \boldsymbol{s}^s\right)$. In addition, the codebook for feature vector representation is not included in this phase. After convergence, the model is generally capable of extracting task-specific semantic information and achieving satisfactory task performance.
- In the second phase, we fine-tune the entire U-DeepSC model, building upon the parameter trained in the first phase. Furthermore, the codebook loss $\mathcal{L}_c$ and the FSM loss $\mathcal{L}_{FSM}$ are both incorporated into the overall loss function $\mathcal{L}\left(\boldsymbol{s}^v, \boldsymbol{s}^t, \boldsymbol{s}^s\right)$. This phase aims to optimize the complete system, aiming for global optimization. Consequently, a balanced trade-off between transmission overhead and task performance can be achieved for different tasks.

The detailed training procedures are summarized in Algo-

Algorithm 1: Two-phase training algorithm

```
1  Input: Training datasets consist of input image, text, speech data and labels. The numbers of training epochs for two phases, $N_1$ and $N_2$, the learning rate.
2  Output: Optimized parameters $\{\boldsymbol{\theta}_v^*, \boldsymbol{\theta}_s^*, \boldsymbol{\theta}_t^*, \boldsymbol{\phi}_u^*\}$.
3  First Phase:
4  Fix parameters of $f_n$.
5  for $i \leftarrow 1$ to $N_1$ do
6      Choose one task and generate a batch of samples.
7      Generate the selection mask vectors.
8      Compute FSM loss $\mathcal{L}_{FSM}$ based on (19).
9      Continue forward propagation with the generated mask vectors.
10     Compute the task-specific loss $\mathcal{L}_p$ according to the type of task.
11     Compute the total loss $\mathcal{L}\left(\boldsymbol{s}^v, \boldsymbol{s}^t, \boldsymbol{s}^s\right) = \mathcal{L}_p + \mathcal{L}_{FSM}$.
12     Update parameters of U-DeepSC, $\{\boldsymbol{\theta}_v, \boldsymbol{\theta}_s, \boldsymbol{\theta}_t, \boldsymbol{\phi}_u\}$, using $\mathcal{L}\left(\boldsymbol{s}^v, \boldsymbol{s}^t, \boldsymbol{s}^s\right)$.
13 end
14 Second Phase:
15 Load the parameters trained in the first phase.
16 for $i \leftarrow 1$ to $N_2$ do
17     Choose one task and generate a batch of samples.
18     Sample the channel variance $\sigma^2$ from the given SNR range.
19     Sample the channel gain coefficient $h$.
20     Generate the selection mask vectors before transmitting.
21     Compute FSM loss $\mathcal{L}_{FSM}$ based on (20).
22     Compute codebook loss $\mathcal{L}_c$ based on (23).
23     Continue forward propagation with the generated mask vectors.
24     Compute the task-specific loss $\mathcal{L}_p$ according to the type of task.
25     Compute the total loss $\mathcal{L}\left(\boldsymbol{s}^v, \boldsymbol{s}^t, \boldsymbol{s}^s\right) = \mathcal{L}_p + \mathcal{L}_{FSM} + \mathcal{L}_c$.
26     Update parameters of U-DeepSC, $\{\boldsymbol{\theta}_v, \boldsymbol{\theta}_s, \boldsymbol{\theta}_t, \boldsymbol{\phi}_u, \boldsymbol{\psi}\}$, using $\mathcal{L}\left(\boldsymbol{s}^v, \boldsymbol{s}^t, \boldsymbol{s}^s\right)$.
27 end
```

rithm 1. The proposed U-DeepSC is a general framework, and can support various tasks of different modalities of data, e.g., object detection and speech classification. Firstly, it necessitates the design of a task embedding vector and task-specific query embedding matrices customized to the specific characteristics of the new task. Secondly, the creation of a task-specific head dedicated to the new task is essential. Finally, the model must undergo joint learning to effectively support the new task.

VII. Simulation Results

In this section, we demonstrate the superiority of the proposed U-DeepSC by numerical results.

A. Simulation Setup

The setting of the training procedure is: the AdamW optimizer with learning rate 1×10^{-5}, batch size 50, weight decay 5×10^{-31}, where we find that the smaller learning rate and the larger batch size lead to better performance. The number of FSM is $N_q = 2$. To verify the effectiveness of U-DeepSC, we test our U-DeepSC on the aforementioned six tasks, each corresponding to a dataset. Specifically, the CIFAR-10 dataset is utilized for both image classification and image reconstruction tasks. For text-related tasks such as text reconstruction and text classification, the SST-2 dataset is employed. Additionally, the VQAv2 dataset is chosen for the visual question answering task, and the MOSEI dataset is utilized for the video task. In the context of image reconstruction, a patch size of 4 is set, while for image classification, it is set to 32. The text-related tasks utilize BERT-base-uncased as the text embedding tool. For video data's vision stream, Facet, an analytical tool based on the facial action coding system (FACS), is used to extract facial features. Speech features are extracted using COVAREP, a professional acoustic analysis framework. The image and text transmitters are initialized using a pretrained vision transformer and the BERT model, respectively. Furthermore, in the simulation, our focus is on AWGN and Rayleigh fading channels.

For comparison, three benchmarks are considered.

- Conventional methods: This is the conventional separate source-channel coding. For the image data, the joint photographic experts group (JPEG) and low-density parity-check code (LDPC) are adopted as image source coding and image channel coding, respectively. In addition, for video, we adopt the H.264 video compression codecs for source coding. For the text data, the 8-bit unicode transformation format (UTF-8) encoding and the Turbo coding are adopted as the text source coding and text channel coding, respectively. For the speech signal, 16-bits pulse code modulation (PCM) and LDPC are employed as the source coding and channel coding, respectively. Moreover, the coding rate of channel coding is selected as $1/2$.
- T-DeepSC: The task-oriented deep learning enabled semantic communication (T-DeepSC) designed for a specific task with the same architecture as U-DeepSC and is implemented by separately trained U-DeepSC.
- Upper bound: Results obtained via delivering noiseless image, speech, and text features to the receiver based on the T-DeepSC.

[1]The code is available at *github.com/zhang-guangyi/t-udeepsc*.

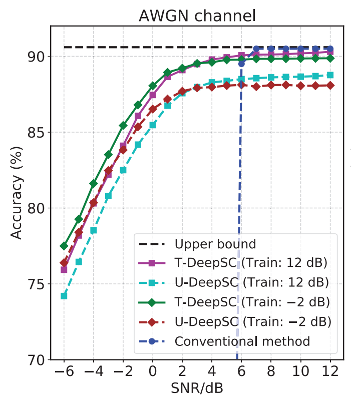
TABLE II
The number of transmitted symbols.

Task	Number of symbols
Image Classification	8
Image Reconstruction	8 (each patch)
Text Classification	4
Text Reconstruction	12 (each word)
Visual Question Answering	30
Video Sentiment Analysis	19

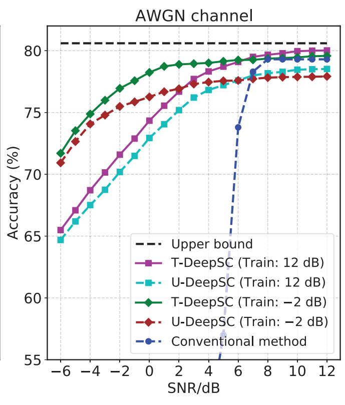
B. Task Performance

Fig. 6 illustrates the performance of the investigated schemes versus the SNR for different tasks. T-DeepSC and U-DeepSC are both trained at the same SNR, i.e., -2 dB and 12 dB. The average number of transmission symbols of U-DeepSC equals that of T-DeepSC. Specifically, the numbers of transmitted symbols for these tasks are shown in Table II. The proposed U-DeepSC and T-DeepSC are tested in SNR from -6 dB to 12 dB (We conduct testing on the image reconstruction task starting from -4 dB, as -6 dB is deemed too low for an image compression task that demands a higher channel capacity.). It is readily seen that both the U-DeepSC and T-DeepSC outperform the conventional schemes and the U-DeepSC approaches the upper bound at high SNR. Moreover, the proposed U-DeepSC achieves close performance to the T-DeepSC in all considered tasks. Therefore, our proposed U-DeepSC is able to simultaneously handle six tasks with comparable performance to the task-oriented models designed for a specific task. Given that UDeepSC transmits only a specific subset of the overall features for different tasks, its satisfactory performance underscores the ability of U-DeepSC to effectively capture and specify task-specific semantic information. By comparing the results of training at -2 dB with that of training at 12 dB, we observe that the performance of U-DeepSC does not saturate immediately with improvements in channel conditions. This characteristic enhances the robustness of U-DeepSC to fluctuations in channel quality, proving advantageous, especially in scenarios involving transmission over time-varying channels.

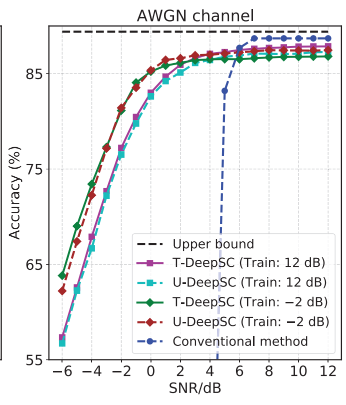
We then train the proposed U-DeepSC model in the AWGN channel with SNR $=$ 12 dB and test it in the Rayleigh fading channel for SNR from -6 dB to 12 dB. The channel information is both known at the transmitter and receiver. To show results in the same figure, we calculate the normalized performance of these tasks as the performance metric, which is obtained by dividing the actual performance by the performance achieved under error free transmission, i.e., upper bound. The results of the considered tasks are shown in Fig. 7. As we can see, the fading effect would lead to performance loss and the performance of U-DeepSC increases with SNR. It can be also found that the proposed model trained in AWGN could well generalize to different channels with different system settings. Moreover, the performance gap between U-DeepSC and T-DeepSC does not decrease a lot as the channel distribution changes. We further find that when we



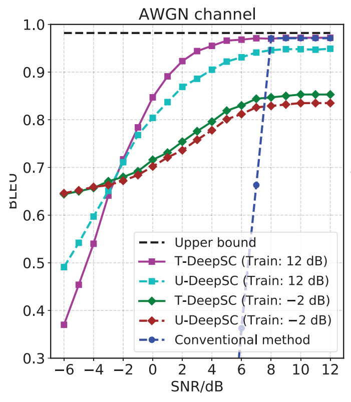
(a) Text Classification



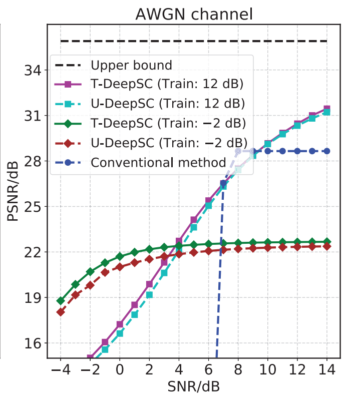
(b) Video Sentiment Analysis



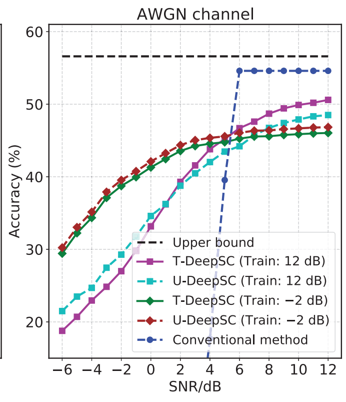
(c) Image Classification



(d) Text Reconstruction

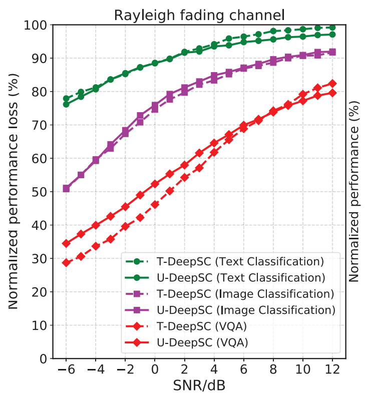


(e) Image Reconstruction

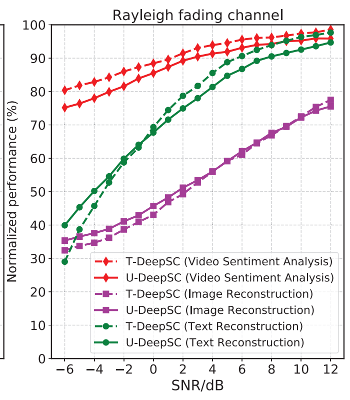


(f) Visual Question Answering

Fig. 6. The performance of six tasks versus SNR, including text classification, video sentiment analysis, image classification, text classification, image reconstruction, and visual question answering. T-DeepSC and U-DeepSC are both trained at SNR $= -2$ dB and SNR $= 12$ dB under AWGN channels.

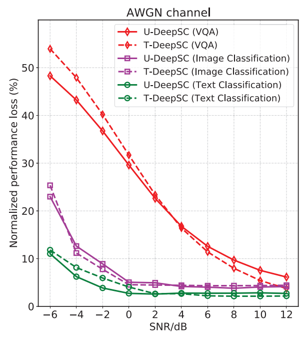


(a)

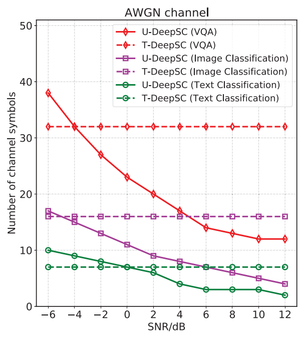


(b)

Fig. 7. The performance of six tasks versus SNR under Rayleigh fading channel.



(a)



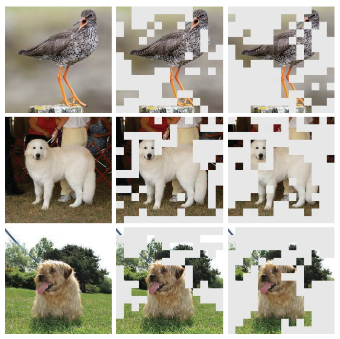
(b)

Fig. 8. Normalized performance loss and number of transmitted symbols as a function of the channel SNR in dynamic channel conditions.

jointly learn multiple tasks, the model tends to perform better at low SNR regimes. This is mainly because the training of the other tasks acts as a perturbation, which is similar to training the model with the lower SNR.

C. *Effectiveness of FSM*

Fig. 8(a) and Fig. 8(b) illustrate the number of transmitted symbols and normalized performance loss versus SNR. From the figure, the normalized performance losses for both U-DeepSC and T-DeepSC decrease with SNR and so does the number of transmitted symbols of U-DeepSC. Compared with T-DeepSC that has a fixed transmission overhead, the proposed method achieves comparable performance with less transmission overhead. The proposed dynamic channel encoder can adaptively adjust the number of transmitted features according to the channel noise levels, thus it can significantly reduce the transmission overhead at the higher SNR regime. Particularly, when the channel conditions are unfavorable, the

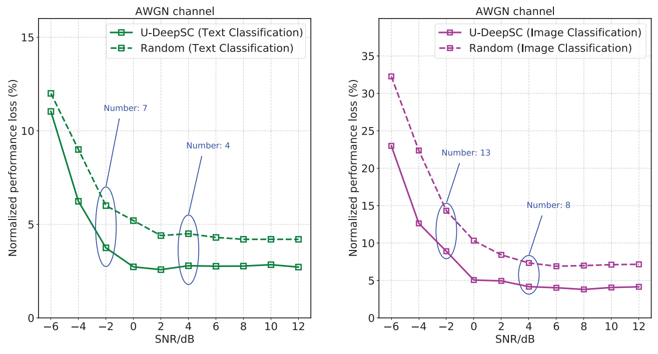


(a) Image

It 's not a bad premise , just a bad movie .
It 's not a bad premise , just a bad movie .
It 's not a bad premise , just a bad movie .
It all adds up to good fun .
It all adds up to good fun .
It all adds up to good fun .
Too bad Maggio could n't come up with a better script .
Too bad Maggio could n't come up with a better script .
Too bad Maggio could n't come up with a better script .

(b) Text

Fig. 9. The visualization of the feature selection.



(a) (b)

Fig. 10. Performance comparison between FSM and random selection. The "Number " marked in the figure indicates the number of transmitted channel symbols.

dynamic channel encoder tends to select more features for transmission, making the received features robust to maintain the performance. It is analogous to adding redundancy for error correction in conventional channel coding techniques. On the contrary, when the channel conditions are good, the dynamic channel encoder tends to transmit fewer features to reduce the communication overhead.

We conduct feature visualization in Fig. 9. Note that in the Transformer architecture, each encoded feature corresponds to one small patch of the input image or one word of a sentence. As for the image, we use gray patches to represent the patches that are masked. It is readily seen that the dynamic channel encoder is able to identify the informative features, which are drawn from the important task-specific patches. Moreover, we observe that the patches in the middle of the image have a higher probability to be kept, which is mainly because that in most images, the objects are located in the center. As for the text, the discarded words are marked in red. We can see that

TABLE III
THE NUMBER OF PARAMETERS.

Task	T-DeepSC	U-DeepSC
Image Classification	7.65M	42.85M
Image Reconstruction	8.20M	42.85M
Text Classification	29.30M	42.85M
Text Reconstruction	33.69M	42.85M
Visual Question Answering	37.60M	42.85M
Video Sentiment Analysis	37.23M	42.85M
Stored Parameters	153.67M	42.85M

the features corresponding to the emotional words, e.g., good and bad, are more likely to be kept in the text classification task. These emotional words usually play an important role in identifying the sentiment of the sentence.

Next, we verify the effectiveness of our proposed FSM by comparing with the random selection strategy. We employ the model trained at SNR $=$ 12 dB for this experiment. Specifically, the feature vectors for proposed FSM are selected based on the probability matrix $\boldsymbol{P}$, where only the top $N_d\delta_q^N$ feature vectors with the highest probability will be selected. For comparison, we randomly select $N_d\delta_q^N$ feature vectors in the inference phase. According to Fig. 10, we find that FSM significantly outperforms the random selection strategy. It is mainly because the feature vectors with the higher sampling probability contributes much more to the final result, i.e., classification results. From the results, we can conclude that FSM makes semantic extraction more efficient and can focus on the content that is truly relevant to the results. Hence, the proposed U-DeepSC is more flexible and can significantly improve the inference speed.

D. *Model Parameters*

As shown in Table III, the total stored number of parameters of T-DeepSC is 153.67M, which is obtained by adding the parameters required for each task. For our proposed U-DeepSC, the number of stored model parameters is only 42.85M for six tasks, which is 28.54% of the T-DeepSC. The U-DeepSC is able to provide satisfactory performance with much-reduced model parameters. It is of great significance towards a practical semantic communication system for scenarios with limited spectrum and storage resources.

VIII. CONCLUSION

In this paper, we first proposed a general framework for U-DeepSC. Particularly, we considered six popular tasks and jointly trained these tasks with a unified model. To control the transmission overhead, we developed a novel vector-wise FSM to make U-DeepSC adaptive to the tasks, where the number of the transmitted features can be dynamically adjusted for different tasks under different SNR regimes. Then, the unified codebook has been proposed for the feature representation of multiple tasks. Simulation results showed that our proposed model had satisfactory performance in the low SNR regime and achieved comparable performance to the task-oriented model designed for a specific task with significant reductions in both transmission overhead and model size.

REFERENCES

[1] W. Saad, M. Bennis, and M. Chen, "A vision of 6G wireless systems: Applications, trends, technologies, and open research problems," *IEEE Netw.*, vol. 34, no. 3, pp. 134–142, May 2020.

[2] C. S. Emilio and B. Sergio, "6G networks: Beyond shannon towards semantic and goal-oriented communications," *Commun. Com. Inf. Sc.*, vol. 190, pp. 107–930, May 2021.

[3] Z. Qin, X. Tao, J. Lu, W. Tong, and G. Y. Li, "Semantic communications: Principles and Challenges," *arXiv preprint arXiv:2201.01389*, 2022.

[4] G. Shi, Y. Xiao, Y. Li, and X. Xie, "From semantic communication to semantic-aware networking: Model, architecture, and open problems," *IEEE Commun. Mag.*, vol. 59, no. 8, pp. 44–50, Aug. 2021.

[5] W. Tong and G. Y. Li, "Nine challenges in artificial intelligence and wireless communications for 6G," *IEEE Wireless Commun.*, pp. 1–10, Aug. 2022.

[6] K. Niu, J. Dai, S. Yao, S. Wang, Z. Si, X. Qin, and P. Zhang, "A paradigm shift toward semantic communications," *IEEE Commun. Mag.*, vol. 60, no. 11, pp. 113–119, Nov. 2022.

[7] M. Kountouris and N. Pappas, "Semantics-empowered communication for networked intelligent systems," *IEEE Commun. Mag.*, vol. 59, no. 6, pp. 96–102, Jun. 2021.

[8] M. Kalfa, M. Gok, A. Atalik, B. Tegin, T. M. Duman, and O. Arikan, "Towards goal-oriented semantic signal processing: Applications and future challenges," *Digit. Signal Process.*, pp. 103–134, Dec. 2021.

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

[10] J. Xu, B. Ai, W. Chen, A. Yang, P. Sun, and M. Rodrigues, "Wireless image transmission using deep source channel coding with attention modules," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 4, pp. 2315–2328, Apr. 2022.

[11] T.-Y. Tung and D. Gündüz, "DeepWiVe: Deep-learning-aided wireless video transmission," *IEEE J. Select. Areas Commun.*, vol. 40, no. 9, pp. 2570–2583, Sep. 2022.

[12] Z. Weng and Z. Qin, "Semantic communication systems for speech transmission," *IEEE J. Select. Areas Commun.*, vol. 39, no. 8, pp. 2434–2444, Aug. 2021.

[13] M. Yang, C. Bian, and H.-S. Kim, "OFDM-guided deep joint source channel coding for wireless multipath fading channels," *IEEE Trans. Cognit. Commun. Netw.*, vol. 8, no. 2, pp. 584–599, Jun. 2022.

[14] M. Wang, J. Li, M. Ma, and X. Fan, "Constellation design for deep joint source-channel coding," *IEEE Signal Process. Lett.*, vol. 29, pp. 1442–1446, Jun. 2022.

[15] A. Li, X. Liu, G. Wang, and P. Zhang, "Domain knowledge driven semantic communication for image transmission over wireless channels," *IEEE Wireless Commun. Lett.*, vol. 12, no. 1, pp. 55–59, Jan. 2023.

[16] H. Zhang, H. Wang, Y. Li, K. Long, and A. Nallanathan, "DRL-driven dynamic resource allocation for task-oriented semantic communication," *IEEE Trans. Commun.*, to appear, 2023, doi: 10.1109/TCOMM.2023.3274145.

[17] D. Huang, F. Gao, X. Tao, Q. Du, and J. Lu, "Towards semantic communications: Deep learning-based image semantic coding," *arXiv preprint arXiv:2208.04094*, 2022.

[18] M. Jankowski, D. Gündüz, and K. Mikolajczyk, "Joint device-edge inference over wireless links with pruning," in *Proc. IEEE 21st Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, May 2020, pp. 1–5.

[19] D. Huang, X. Tao, F. Gao, and J. Lu, "Deep learning-based image semantic coding for semantic communications," in *in Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2021, pp. 1–6.

[20] S. Wang, J. Dai, Z. Liang, K. Niu, Z. Si, C. Dong, X. Qin, and P. Zhang, "Wireless deep video semantic transmission," *IEEE J. Select. Areas Commun.*, vol. 41, no. 1, pp. 214–229, Jan. 2023.

[21] M. Jankowski, D. Gündüz, and K. Mikolajczyk, "Wireless image retrieval at the edge," *IEEE J. Select. Areas Commun.*, vol. 39, no. 1, pp. 89–100, Jan. 2021.

[22] C. H. Lee, J. W. Lin, P. H. Chen, and Y. C. Chang, "Deep learning-constructed joint transmission-recognition for Internet of Things," *IEEE Access*, vol. 7, pp. 76 547–76 561, Jun. 2019.

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

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

[25] D. B. Kurka and D. Gündüz, "DeepJSCC-f: Deep joint source-channel coding of images with feedback," *IEEE J. Select. Areas Inf. Theory*, vol. 1, no. 1, pp. 178–193, May 2020.

[26] J. Shao, Y. Mao, and J. Zhang, "Learning task-oriented communication for edge inference: An information bottleneck approach," *IEEE J. Select. Areas Commun.*, vol. 40, no. 1, pp. 197–211, Jan. 2022.

[27] S. Ma, W. Qiao, Y. Wu, H. Li, G. Shi, D. Gao, Y. Shi, S. Li, and N. Al-Dhahir, "Task-oriented explainable semantic communications," *IEEE Trans. Wireless Commun.*, to appear, 2023, doi: 10.1109/TWC.2023.3269444.

[28] Y. Xu, H. Zhou, and Y. Deng, "Task-oriented semantics-aware communication for wireless UAV control and command transmission," *IEEE Commun. Lett.*, vol. 27, no. 8, pp. 2232–2236, Aug. 2023.

[29] J. Huang, D. Li, C. Huang, X. Qin, and W. Zhang, "Joint task and data oriented semantic communications: A deep separate source-channel coding scheme," *IEEE Internet Things J.*, to appear, 2023, doi: 10.1109/JIOT.2023.3293154.

[30] E. Kutay and A. Yener, "Semantic text compression for classification," *arXiv preprint arXiv:2309.10809*, 2023.

[31] M. Wang, Z. Zhang, J. Li, M. Ma, and X. Fan, "Deep joint source-channel coding for multi-task network," *IEEE Signal Process. Lett.*, vol. 28, pp. 1973–1977, Sep. 2021.

[32] Y. E. Sagduyu, T. Erpek, A. Yener, and S. Ulukus, "Multi-receiver task-oriented communications via multi-task deep learning," *arXiv preprint arXiv:2308.06884*, 2023.

[33] S. Vandenhende, S. Georgoulis, W. Van Gansbeke, M. Proesmans, D. Dai, and L. Van Gool, "Multi-task learning for dense prediction tasks: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 7, pp. 3614–3633, Sep. 2021.

[34] M. Crawshaw, "Multi-task learning with deep neural networks: A survey," *arXiv preprint arXiv:2009.09796*, 2020.

[35] R. Hu and A. Singh, "Unit: Multimodal multitask learning with a unified transformer," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 1419–1429.

[36] W. Yu, H. Xu, Z. Yuan, and J. Wu, "Learning modality-specific representations with self-supervised multi-task learning for multimodal sentiment analysis," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 12, May 2021, pp. 10 790–10 797.

[37] D. Hazarika, R. Zimmermann, and S. Poria, "MISA: Modality-invariant and-specific representations for multimodal sentiment analysis," in *Proc. ACM Int. Conf. Multimedia*, 2020, pp. 1122–1131.

[38] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[39] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[40] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Aug. 2020, pp. 213–229.

[41] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, Jun. 2022, pp. 16 000–16 009.

[42] Y. Rao, W. Zhao, B. Liu, J. Lu, J. Zhou, and C.-J. Hsieh, "Dynamicvit: Efficient vision transformers with dynamic token sparsification," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2021, pp. 13 937–13 949.

[43] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," in *Proc. Int. Conf. Learn. Representations (ICLR)*, Feb. 2017.

[44] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, "Neural discrete representation learning," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, Dec. 2017, pp. 6309–6318.

[45] M. Mortaheb, M. A. Khojastepour, S. T. Chakradhar, and S. Ulukus, "Semantic multi-resolution communication," *arXiv preprint arXiv:2308.11604*, 2023.