A Unified Multi-Task Semantic Communication System for Multimodal Data

Guangyi Zhang, Qiyu Hu, Zhijin Qin, Yunlong Cai, Guanding Yu, Xiaoming Tao, and Geoffrey Ye Li

Abstract

Task-oriented semantic communication has achieved significant performance gains. However, the model has to be updated once the task is changed or multiple models need to be stored for serving 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. As the difficulty varies from different tasks, different numbers of neural network layers are required for various tasks. We develop a multi-exit architecture in U-DeepSC to provide early-exit results for relatively simple tasks. To reduce the transmission overhead, we design a unified codebook for feature representation for serving multiple tasks, in which only the indices of these task-specific features in the codebook are transmitted. Moreover, we propose a dimension-wise dynamic scheme that can adjust the number of transmitted indices for different tasks as the number of required features varies from task to task. Furthermore, our dynamic scheme can adaptively adjust the numbers of transmitted features under different channel conditions to optimize the transmission efficiency. According to 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, multi-exit, multimodal data, multi-task semantic communication, unified codebook.

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

Geoffrey Ye Li is with the Department of Electrical and Electronic Engineering, Imperial College London, London SW7 2AZ, UK (e-mail: geoffrey.li@imperial.ac.uk).

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. To address this issue, semantic communications have been considered as a promising technology to achieve better performance [1], [2].

A. Prior Work

Semantic communications have recently received great attention [3]. Different from conventional communications, they only take into account the relevant semantic information to the tasks [4]–[6]. 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 [7]–[14], especially in unfriendly channel environments.

The existing works on semantic communications can be mainly divided into two categories: data reconstruction [7]–[14] and task execution [15]–[19]. For the data reconstruction, the semantic system extracts global semantic information from the source data. Specifically, a so-called DeepSC framework in [7] encodes the text information into various lengths by employing sentence information. In [8], 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 [9] can overcome the cliff-effect. The semantic communication system in [10] 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 [15]–[19]. In particular, a model for image retrieval task under power and bandwidth constraints has been proposed in [15]. In [16], an image classification-oriented semantic communication system has been developed. In [17], a vector quantization-variational autoencoder (VQ-VAE) based robust semantic communication systems have 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

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 [18], 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 [18] 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 [20] has designed a model to handle the image detection and segmentation tasks, but it only handles two tasks with one modality of data.

There have been some works about multi-task learning in the field of computer vision and natural language processing [21], [22]. 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.

B. Motivation and Contributions

The unified multi-task model and overhead control for different modalities of data in wireless communications has not been well investigated. Therefore, we propose a unified DL-based semantic communication system (U-DeepSC) in this paper. To the best of our knowledge, this is the first work on the unified semantic communication system for serving various tasks. The proposed U-DeepSC is able to deal with a number of tasks with three modalities, i.e., image, text, and speech, simultaneously. Since different tasks are with different difficulties and require different numbers of layers to achieve satisfactory performance, the multi-exit architecture is developed by inserting the early-exit modules after the intermediate layers of the decoder to provide early-exit results for relatively simple tasks [23], [24]. In addition, since more decoder layers are required to eliminate the impact of the larger channel noise, we involve the channel noise in the training procedure to enable U-DeepSC to achieve dynamic inferencing with adaptive

layers under different channel conditions.

To reduce the transmission overhead, we adopt the codebook design in [17], 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 [17] where the codebook is for a specific task with single modality, we design a unified codebook with domain adaptation for multimodal data, which makes the output features of different tasks similar and can be implemented by adding the distance constraint in the training procedure. Thus, the output features of different tasks vary on a similar distribution, which is beneficial for a unified codebook with a smaller size to represent the output features compared with the codebook designed separately in [17]. Moreover, there exists redundancy for transmitting all the features since different tasks require different numbers of transmitted features. Generally, the data reconstruction requires more transmitted features than intelligent tasks. If more encoded features are transmitted, the redundancy of the features would lead to better performance against noise but will induce a higher transmission overhead and latency. There is an inherent trade-off between the performance and the number of the transmitted symbols. Therefore, we propose a dynamic channel encoder to remove the redundant features based on different tasks and adjust the number of transmitted features according to the channel conditions. In particular, we add the selection modules at the channel encoder, to generate the selection mask vector based on the tasks and the channel conditions to indicate the features to transmit. To jointly train these modules, we propose a novel training method. Simulation results show that our proposed method achieves comparable performance to the task-oriented semantic communication systems designed for a specific task with much reduced transmission overhead and fewer model parameters. The main contributions of this paper can be summarized as follows.

- We firstly propose U-DeepSC, a unified semantic communication architecture that can handle a number of tasks with multimodal data by jointly learning these tasks.
- The multi-exit architecture is designed to provide early-exit results for different tasks under different channel conditions, which significantly reduces the inference time.
- We develop a novel method with domain adaptation to design a unified codebook for multi-task services, which further reduces the codebook size as well as improves the performance, by projecting the features of different tasks into the same domain.
- We design a dynamic channel encoder, which is empowered with the ability to dynamically

adjust the numbers of transmitted features under different channel conditions and tasks.

C. Organization and Notations

The rest of this paper is organized as follows. Section II introduces the framework of U-DeepSC. Section III presents a unified codebook designed with a novel domain adaptation loss. Section IV develops the task-specific dynamic channel encoder with the selection module. The considered tasks and training method are introduced in Section V. Simulation results are presented in Section VI. Finally, Section VII concludes this paper.

Notations: Scalars, vectors, and matrices are respectively denoted by lower case, boldface lower case, and boldface upper case letters. Notation $\mathbf{I}$ represents an identity matrix and $\mathbf{0}$ denotes an all-zero matrix. For a matrix $\mathbf{A}$, $\mathbf{A}^\top$ and $\|\mathbf{A}\|$ denote its transpose, conjugate and Frobenius norm, respectively. For a vector $\mathbf{a}$, $\|\mathbf{a}\|$ is its Euclidean norm. Finally, $\mathbb{C}^{m\times n}$ $(\mathbb{R}^{m\times n})$ are the space of $m \times n$ complex (real) matrices.

II. Framework of the U-DeepSC

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

A. Overview of U-DeepSC

As shown in Fig. 1, the proposed U-DeepSC is able to handle a number of tasks with three modalities. For example, given a video task that contains the image, text, and speech data, the proposed framework aims to deal with three modalities of data. It mainly consists of four parts: image transmitter, text transmitter, speech transmitter, and unified receiver. DNNs are employed to implement the transmitters and the unified receiver. In particular, each transmitter consists of the a semantic encoder and a channel encoder. The unified receiver consists of the channel decoder and the semantic decoder.

We consider a communication system equipped with N_t transmit antennas and N_r receive antennas. The inputs of the system are image, $\mathbf{x}^I$, text, $\mathbf{x}^T$, and speech, $\mathbf{x}^S$. Each is processed to obtain the corresponding encoded features. Thus, the encoded features of the three modalities of data can be represented by

$$\hat{\mathbf{x}}^I = \mathcal{F}_C^I\big(\mathcal{F}_S^I(\mathbf{x}^I;\boldsymbol{\theta}_S^I);\boldsymbol{\theta}_C^I\big), \tag{1}$$

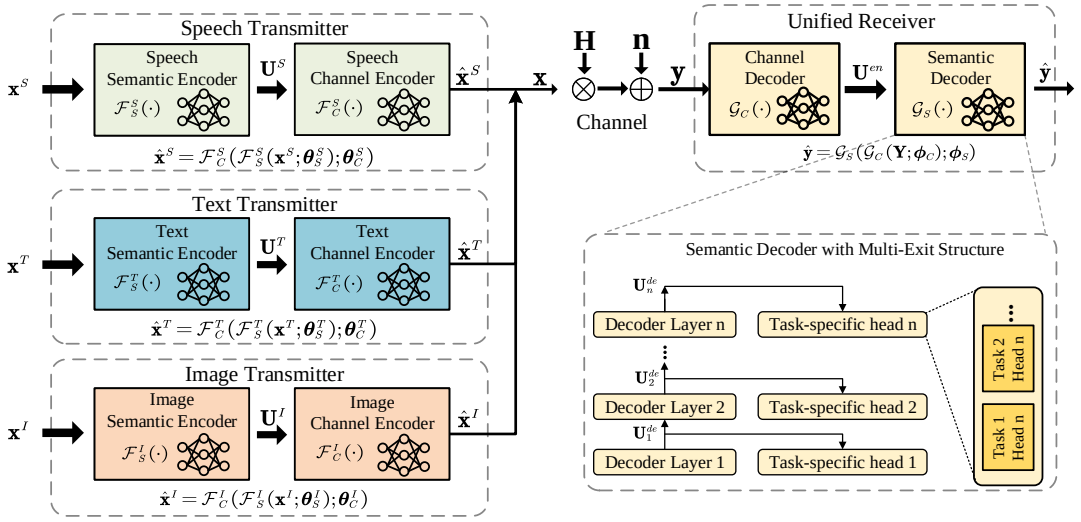


Fig. 1: The framework of the proposed U-DeepSC.

$$\hat{\mathbf{x}}^T = \mathcal{F}_C^T\big(\mathcal{F}_S^T(\mathbf{x}^T;\boldsymbol{\theta}_S^T);\boldsymbol{\theta}_C^T\big), \tag{2}$$

and

$$\hat{\mathbf{x}}^S = \mathcal{F}_C^S\big(\mathcal{F}_S^S(\mathbf{x}^S;\boldsymbol{\theta}_S^S);\boldsymbol{\theta}_C^S\big), \tag{3}$$

where $\hat{\mathbf{x}}^I \in \mathbb{C}^{N_t\times 1}$, $\hat{\mathbf{x}}^T \in \mathbb{C}^{N_t\times 1}$, $\hat{\mathbf{x}}^S \in \mathbb{C}^{N_t\times 1}$, $\boldsymbol{\theta}_S^I$ and $\boldsymbol{\theta}_C^I$ denote the trainable parameters of the image semantic encoder, $\mathcal{F}_S^I$, and the image channel encoder, $\mathcal{F}_C^I$, respectively, $\boldsymbol{\theta}_S^T$ and $\boldsymbol{\theta}_C^T$ denote the trainable parameters of the text semantic encoder, $\mathcal{F}_S^T$, and the text channel encoder, $\mathcal{F}_C^T$, respectively, $\boldsymbol{\theta}_S^S$ and $\boldsymbol{\theta}_C^S$ denote the trainable parameters of the speech semantic encoder, $\mathcal{F}_S^S$, and the speech channel encoder, $\mathcal{F}_C^S$, respectively. We concatenate the encoded features to obtain the transmitted symbol streams expressed as $\mathbf{x} = \big[\hat{\mathbf{x}}^I, \hat{\mathbf{x}}^T, \hat{\mathbf{x}}^S\big]$. Then, the received signal at the receiver is given by

$$\mathbf{Y} = \mathbf{H}\mathbf{x} + \mathbf{n}, \tag{4}$$

where $\mathbf{H} \in \mathbb{C}^{N_r\times N_t}$ represents the channel matrix and $\mathbf{n} \sim \mathcal{CN}(\mathbf{0}, \sigma^2\mathbf{I})$ is the additive white Gaussian noise (AWGN).

At the receiver, the decoded signal can be represented as

$$\hat{\mathbf{Y}} = \mathcal{G}_S\big(\mathcal{G}_C(\mathbf{Y};\phi_C);\phi_S\big), \tag{5}$$

where ϕ_C and ϕ_S denote the trainable parameters of the channel decoder, $\mathcal{G}_C$, and the semantic decoder, $\mathcal{G}_S$, respectively. Finally, the obtained features are further processed by light-weight task-specific heads to execute downstream tasks. Particularly, the task-specific head refers to some

simple layers that reshape the decoded features into the intended dimension of output, e.g., the number of classes for a classification task. Moreover, the multi-exit architecture is developed at the receiver. Particularly, we insert the early-exit modules after the intermediate layer of the decoder, which provides early-exit results adaptively based on the tasks and channel conditions.

B. Semantic Encoder

Since the data from different modalities have totally different statistical characteristics and have different semantic information and encoded features, we have to design image, text, and speech semantic encoders for image, text, and speech, respectively.

1) Image semantic encoder: The image-only and multimodal tasks take an image, $\mathbf{I}$, as input, and preprocess it into the preliminary features, $\mathbf{x}^I$. 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. Then, the image semantic encoder maps $\mathbf{x}^I$ to encoded image features, $\mathbf{U}^I$. Moreover, since different tasks require the semantic encoder to extract different features, a task embedding vector, $\mathbf{w}^I_{task}$, is added to the semantic encoder given as $[\mathbf{x}^I, \mathbf{w}^I_{task}]$. It indicates the task to perform with the given sample, $\mathbf{I}$, and to allow it to extract task-specific information. Particularly, the task embedding vectors are employed to perform convolution or attention operation together with the encoded image features [25]. Then, we obtain the encoded image feature matrix $\mathbf{U}^I = \{\mathbf{u}^I_1, ..., \mathbf{u}^I_L\}$, where $\mathbf{u}^I_i$ for $i = 1, 2, ..., L$, denotes the encoded image feature vector.

2) Text semantic encoder: As for text-only and multimodal tasks, we preprocess the input text into a sequence of O features $\mathbf{x}^T = \{\mathbf{w}^T_1, ..., \mathbf{w}^T_O\}$. Subsequently, $\mathbf{x}^T$ is encoded by the text semantic encoder. Similar to the image semantic encoder, we also add a trainable task embedding vector $\mathbf{w}^T_{task}$ by concatenating it with $\mathbf{x}^T$. Then, the concatenated sequence, $[\mathbf{x}^T, \mathbf{w}^T_{task}]$, is input into the text semantic encoder, which generates the encoded text features, $\mathbf{U}^T = \{\mathbf{u}^T_1, ..., \mathbf{u}^T_O\}$.

3) Speech semantic encoder: As for speech-only and multimodal tasks, the input of the proposed U-DeepSC is denoted by $\mathbf{x}^S$, which is the speech samples drawn from the corresponding dataset. Then, the input samples are framed for training before passing through the speech semantic encoder [10]. The speech semantic learns the D encoded speech features $\mathbf{U}^S = \{\mathbf{u}^S_1, ..., \mathbf{u}^S_D\}$ from the concatenated sequence $[\mathbf{x}^S, \mathbf{w}^S_{task}]$, where $\mathbf{w}^S_{task}$ is the task embedding vector trained with the whole network.

C. Unified Semantic Decoder

The received encoded features are firstly processed by the channel decoder, whose output is denoted as $\mathbf{U}^{en}$. For image-only tasks, text-only tasks, and speech-only tasks, the input to the decoder can be represented by $\mathbf{U}^{en} = \hat{\mathbf{U}}^I$, $\mathbf{U}^{en} = \hat{\mathbf{U}}^T$, and $\mathbf{U}^{en} = \hat{\mathbf{U}}^S$, respectively, where $\hat{\mathbf{U}}^I$, $\hat{\mathbf{U}}^T$, and $\hat{\mathbf{U}}^T$ denote the decoded image features, text features, and speech features of channel decoder, respectively. For multimodal tasks, we concatenate the decoded features from the corresponding modalities of data into a sequence, e.g., $\mathbf{U}^{en} = [\hat{\mathbf{U}}^I, \hat{\mathbf{U}}^T]$ 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. 1. The semantic decoder takes the output of channel decoder, $\mathbf{U}^{en}$, and the task-specific query vector, $\mathbf{q}_{task}$, as input. The task-specific query vector is employed to indicate the task to be handled by the semantic decder, which is also a conventional input required by the Transformer decoder. Then, we obtain the output of the i-th decoder layer, $\mathbf{U}_i^{de}$, which will be processed by the multi-exit task-specific heads to output early-exit results.

D. Multi-Exit Module with Task-Specific Heads

In general, different tasks under different channel conditions require different numbers of layers for the reasons below: (i) Each task is of different difficulty and requires different numbers of layers to achieve satisfactory performance; (ii) Different tasks require different levels of semantic information from various layers; (iii) As for the better channel conditions, fewer layers are required to eliminate the influence of channel noise. To design the multi-exit architecture, we firstly attach the task-specific heads to these intermediate layers, as shown in Fig. 2(a). It is called the confidence-based method, which provides early-exit results when the confidence for the result at a certain layer is greater than the predefined threshold, δ_e. In this way, the inference time can be dynamically reduced.

1) Training method: As for the multi-exit architecture as shown in Fig. 2(a), we need to involve more than one task-specific heads in the training procedure. To train these task-specific heads, a straightforward way is to minimize the sum of the task-specific loss funtions of each head. In particular, denote the loss function for the task-specific head in the i-th layer as $\mathcal{L}_i = \mathcal{C}(g_i(\mathbf{U}_i^{de}; \boldsymbol{\psi}_i), \boldsymbol{y})$, where $\boldsymbol{y}$ is the label of the task, g_i is the task-specific head of the i-th layer,

$\boldsymbol{\psi}_i$ is the parameter of g_i, and $\mathcal{C}$ is the task-specific loss function, e.g., the cross-entropy function for the classification tasks. Then, we can simply minimize $\sum_{i=1}^{n} \mathcal{L}_i$ to train these task-specific heads. However, this method is limited in U-DeepSC for the reasons below:

- The decoder layers have to provide intermediate features for two competing purposes: immediate inference at the adjacent task-specific heads and gradual feature extraction for future heads. Thus, the loss functions of different task-specific heads negatively interfere with each other.
- We need to take the results of the last layer as the model output when the confidence for the result at every layer is less than the predefined threshold. It occurs in the inference phase frequently, hence the task-specific head at the last decoder layer plays an important role in the model performance. Therefore, preserving the performance for the task-specific head at the last decoder layer is important. It cannot be achieved by the simple sum of these loss functions, where there is no preference for the task-specific head at the last decoder layer.

To address these issues, we propose a novel two-phase training strategy for the multi-task training with multi-exit architecture. In the first phase, we update model with the loss function of the task-specific head at the last decoder layer alone. In the second phase, we update the model with the sum of these loss functions. The objectives in the two stages are as follows.

$$\begin{aligned} &\text{Phase 1: } \min \ \lambda_n \mathcal{L}_n, \\ &\text{Phase 2: } \min \ \sum_{i=1}^{n} \lambda_i \mathcal{L}_i, \end{aligned} \tag{6}$$

where λ_i for $i = 1, \cdots, n$ is the introduced weight to denote the importance of each loss, n denotes the number of task-specific heads. While for the multi-task training, we perform the above two-phase training for each task in turn. Moreover, the performance of the task-specific head at the last layer can be well improved, since we train it alone in the first phase specifically.

2) Exiting decision: As for the multi-exit structure, the inference is terminated when the model is certain enough to its prediction at an intermediate layer. To obtain the predicted certainty, we propose a learnable certainty module (LCM) that takes into account both channel condition, i.e., SNR, and features output by the intermediate layers of the decoder. Particularly, the LCM consists of a simple multilayer perceptron (MLP). It takes the intermediate features, $\mathbf{U}_i^{de}$, and the variance of the signal noise, σ^2, as its input and outputs the predicted certainty, c_i, at the

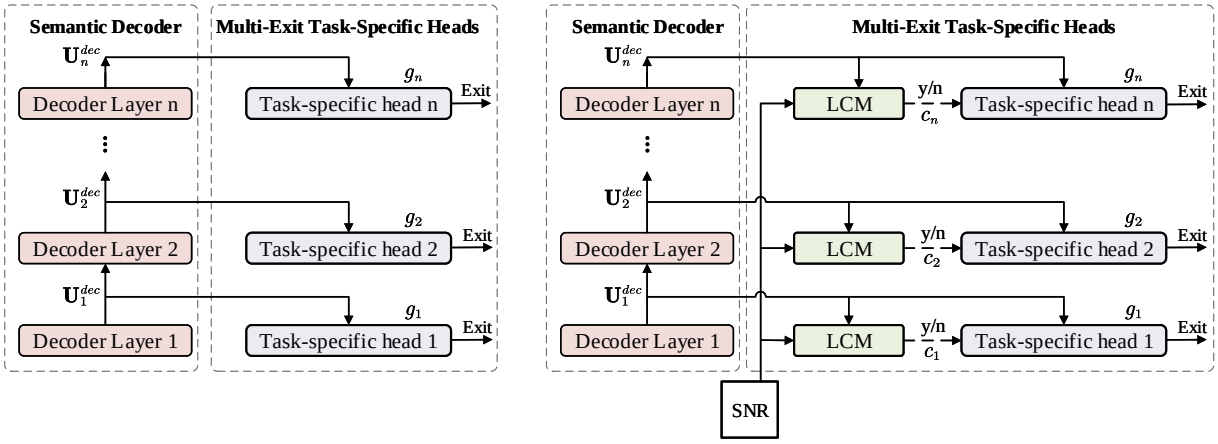


Fig. 2: The structure of semantic decoder with multi-exit architecture: (a) Illustration of the confidence-based multi-exit architecture; (b) Illustration of the learning-based multi-exit architecture.

i-th layer. Moreover, the predicted certainty is constrained between 0 and 1, with 1 denoting the highest certainty. Therefore, we take the sigmoid$(\cdot)$ as the activation function for LCM and the predicted certainty is given by

$$c_i = \text{sigmoid}\left(f\left(\mathbf{U}_i^{de}, \sigma^2\right)\right), \tag{7}$$

where $f(\cdot)$ denotes the MLP.

To train LCM, the loss function is designed as the mean squared-error (MSE) between c_i and the ground truth certainty, $\hat{c}_i$, at the i-th layer, which is given by $\mathcal{L}_i^{LCM} = ||c_i - \hat{c}_i||_2^2$. Particularly, we train the LCM and these task-specific heads simultaneously by adding $\mathcal{L}_i^{LCM}$ to the loss function of the corresponding heads, i.e., $\mathcal{L}_i^{LCM} + \mathcal{L}_i$. For the classification task, the ground truth certainty is determined by whether the task-specific head makes the correct prediction. If a certain task-specific head makes the correct prediction, the ground truth certainty for this head equals 1 while the ground truth certainty equals 0 for the wrong prediction. Therefore, the ground truth certainty for the i-th layer is given by

$$\hat{c}_i = \mathbf{1}\left(\arg\max_j \; g_i^j\left(\mathbf{U}_i^{de}\right) = \boldsymbol{y}\right), \tag{8}$$

where g_i denotes the predicted result output by the task-specific head, and j denotes its j-th dimension, $\mathbf{1}(\cdot)$ is 1 when the condition in the bracket is satisfied and 0 otherwise. As for the

data reconstruction task, the ground truth certainty can be evaluated by the absolute error of the predicted results. In particular, if the reconstruction is great at a certain decoder layer, i.e., the reconstruction result is very close to the label data, $\boldsymbol{y}$, then we obtatin a high ground truth certainty that approaches 1. While for the worst reconstruction, the ground truth certainty approaches 0. We adopt the $\tanh(\cdot)$ as the activation function to obtain the ground truth certainty for the data reconstruction task since it has an output range from 0 to 1 for the input greater than 0. Thus, the ground truth certainty for the i-th layer is expressed as

$$\hat{c}_i = 1 - \tanh\left(\left|\left|g_i\left(\mathbf{U}_i^{de}\right) - \boldsymbol{y}\right|\right|\right). \tag{9}$$

To apply LCM and provide the same metric to evaluate the intermediate features of different layers, we share only one LCM among all decoder layers. Moreover, as for multi-task applications, we design an LCM for each task, whch is jointly trained with the model. At the inference phase, the predicted certainty of the output at each layer is compared with the threshold, δ_e. If it is larger than the threshold at a certain layer, i.e. $c_i > \delta_e$, the model outputs the early-exit result and the subsequent layers will not be executed.

III. Unified Codebook with Domain Adaptation

In this section, we first design the unified codebook for the multi-task applications and then propose a novel method with domain adaptation to reduce its size.

A. Unified Codebook for Multimodal Data

We aim to design a unified codebook for all considered tasks with different modalities of data. The codebook consists of a number of basis vectors and the encoded features 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. 3. Different tasks of the same modality share the same sub-codebook.

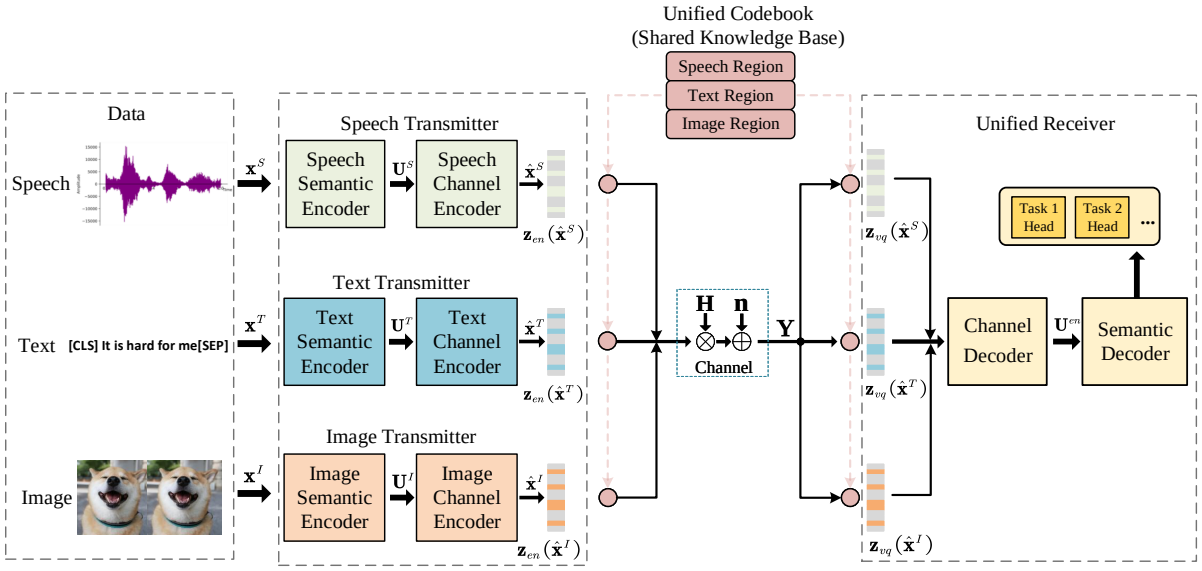


Fig. 3: The architecture of the proposed U-DeepSC.

As presented in Fig. 4, we design the unified codebook as

$$\mathcal{E} \triangleq \left\{\mathcal{E}^I, \mathcal{E}^T, \mathcal{E}^S\right\} \triangleq \left\{\{\mathbf{e}_i\}_{i=1}^{K_I}, \{\mathbf{e}_i\}_{i=K_I+1}^{K_I+K_T}, \{\mathbf{e}_i\}_{i=K_I+K_T+1}^{K_I+K_T+K_S}\right\} \in \mathbb{R}^{K\times D}, \tag{10}$$

where K_I, K_T, and K_S are the size of the image sub-codebook, $\mathcal{E}^I$, text sub-codebook, $\mathcal{E}^T$, and speech sub-codebook, $\mathcal{E}^S$, respectively, D is the dimension of each basis vector, $\mathbf{e}_i$, and $K = K_I + K_T + K_S$, is the total size of the unified codebook. The input data, $\mathbf{s}$, passes through an encoder to produce the encoded feature vector, $\mathbf{z}_{en}(\mathbf{s})$. Then, it is replaced by a basis vector, $\mathbf{z}_{vq}(\mathbf{s})$, which is the nearest vector in the codebook [26],

$$\mathbf{z}_{vq}(\mathbf{s}) = \arg\min_{\mathbf{e}_j} \left\|\mathbf{z}_{en}(\mathbf{s}) - \mathbf{e}_j\right\|_2, \forall \mathbf{e}_j. \tag{11}$$

The basis vectors, $\{\mathbf{e}_j, \forall j\}$, in the codebook, $\mathcal{E}$, are trained together with the parameters of encoder and decoder. However, the operation in (11) is non-differentiable. Hence, the gradients are copied from the input of the decoder, $\mathbf{z}_{vq}(\mathbf{s})$, to the output of the decoder, $\mathbf{z}_{en}(\mathbf{s})$. In this way, the gradient, $\nabla_{\mathbf{z}_{vq}(\mathbf{s})}\mathcal{L}_c$, 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

$$\mathcal{L}_c(\mathbf{s}, \mathbf{z}; \boldsymbol{\theta}, \mathbf{e}_j) = \left\|\mathrm{ng}\left[\mathbf{z}_{en}(\mathbf{s})\right] - \mathbf{e}_j\right\|_2^2 + \beta\left\|\mathbf{z}_{en}(\mathbf{s}) - \mathrm{ng}\left[\mathbf{e}_j\right]\right\|_2^2, \tag{12}$$

where $\mathbf{s}$, $\hat{\mathbf{s}}$, and $\mathbf{z}$ are the input, output, and true label, respectively, $\boldsymbol{\theta}$ is the trainable parameters of the original DNN, and β denotes the hyper-parameter. Symbol $\mathrm{ng}\left[\mathbf{z}_{en}(\mathbf{s})\right]$ represents stop-

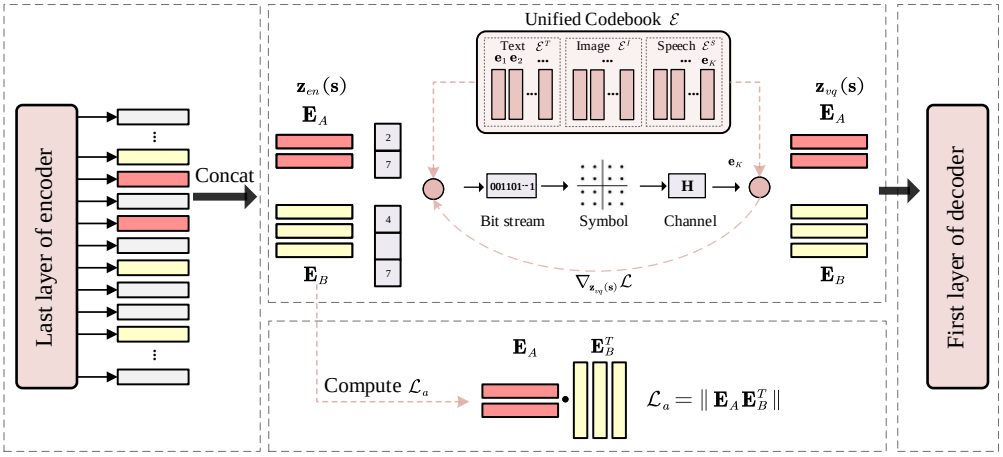


Fig. 4: The architecture of unified codebook with domain adaptation.

gradient operator that has zero gradient during backward propagation and is the identity layer during the forward propagation. It constrains $\mathbf{z}_{en}(\mathbf{s})$ to be a non-updated constant.

B. Improvement with Domain Adaptation

Although the codebook can be trained with loss function (12), it is limited in multi-task applications. This is mainly because that the data from different modalities and tasks has different distributions, which leads to great difference of the encoded features. Thus, a significantly large codebook is required to handle these tasks. To address this issue, we leverage the domain adaptation [27] to design a unified codebook with a much smaller number of basis vectors for multiple tasks. In particular, a domain adaptation loss is proposed for the training procedure, which improves the performance and accelerates the convergence speed.

As shown in Fig. 4, we denote the feature matrix of tasks A and B as $\mathbf{E}_A$ and $\mathbf{E}_B$, respectively. The domain adaptation loss is designed to increase the similarity between the encoded features of task A and B. The most commonly used methods for evaluate the similarity are maximum mean discrepancy (MMD), correlation alignment (CORAL), and Kullback Leibler (KL) divergence [27], [28]. However, they are not suitable for U-DeepSC as the dimensions of $\mathbf{E}_A$ and $\mathbf{E}_B$ are generally different. To address this issue, we propose an efficient similarity criterion for domain adaptation. In particular, we define the similarity of the features as the Frobenius norm of the product of corresponding feature matrices, i.e., $\|\mathbf{E}_A\mathbf{E}_B^\top\|$, where a larger value indicates a higher

similarity. Then, the domain adaptation loss is given by

$$\mathcal{L}_a = -\|\mathbf{E}_A \mathbf{E}_B^\top\|. \tag{13}$$

It enables the model to project the encoded features of different tasks into the same domain, hence the output features for different tasks become similar, which leads to a much smaller codebook size.

IV. Task-Specific Dynamic Channel Encoder

In this section, we introduce the task-specific dynamic overhead by developing the dynamic channel encoder with the selection module. In addition, the channel condition is incorporated into the task-specific dynamic channel encoder to determine which features should be transmitted under different SNRs. Furthermore, we propose a novel loss function to train the dynamic channel encoder.

A. Element-Wise and Dimension-Wise 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 the 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 that 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 features 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.

There exist some model pruning methods [29]–[31], which aim to prune the unimportant parameters of the model. The pruning of these parameters generally reduces the length of the feature vector. In particular, it would drop the redundant elements of the encoded feature vector and can achieve element-wise feature selection. As shown in Fig. 5(a), the elements represented by the white squares would be dropped. These pruning methods inspire us to investigate the way to conduct feature selection in U-DeepSC to reduce the redundancy.

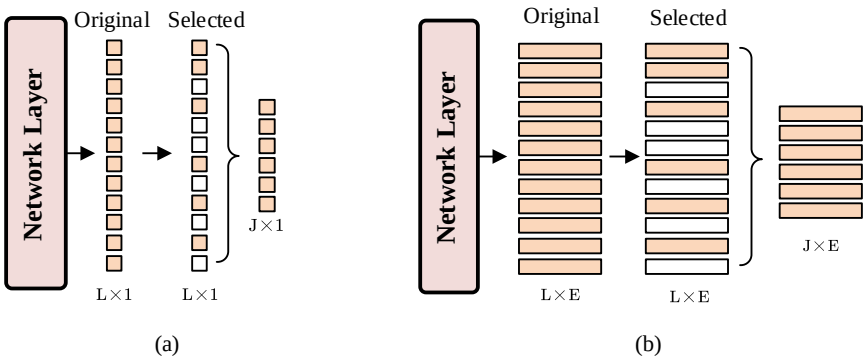


Fig. 5: (a) Element-wise method; (b) Dimension-wise method.

Although we can perform feature selection by simply employing the model pruning method, there are two drawbacks: (i) The elements are dropped permanently and the model is fixed after being pruned, which cannot achieve dynamic overhead; (ii) Since the encoded features are represented by the indices of the basis vectors, we generally need to drop all the elements in a feature vector. To address these issues, we further propose a dimension-wise scheme as shown in Fig. 5(b), where we drop some feature vectors represented by the white rectangle at. Particularly, we design a dynamic channel encoder to adjust the redundancy by adjusting the number of the transmitted features. The proposed dynamic channel encoder has the following advantages:

- The transmitter can identify the task-related features and omit the task-unrelated features, which leads to satisfactory performance.
- With the feature dimension-wise design, the transmission overhead can be dynamically adjusted and significantly reduced.

B. Overview of the Dynamic Channel Encoder

Since the encoded features are represented by the basis vectors in the codebook, we perform feature selection to dynamically adjust the number of transmitted indices of basis vectors. We omit the task-unrelated features and transmit the informative task-related features to the receiver. In particular, the selection module is inserted into the original channel encoder layer to generate the probabilities of dropping/keeping the features. We conduct hierarchical selection through the whole network at certain layers, to perform gradually dropping. For example, as for a 4-layer channel encoder, we can conduct feature selection at the 2nd and 4th layers. It enables the model to gradually recognize and keep the important features through multiple selections.

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.

C. Hierarchical Feature Selection by Sampling

1) Hierarchical feature selection: To perform hierarchical feature selection, i.e., gradually drop the uninformative features, we introduce the selection mask vector $\mathbf{m} \in \{0,1\}^N$ to indicate whether to transmit each feature, where N is the number of the encoded features. The elements in the selection mask vector are firstly initialized to 1 and we update the selection mask vector progressively in the Q selection modules, as shown in Fig. 6. We denote the feature map, $\mathbf{F}^q \in \mathbb{R}^{N\times E}$, as the input of the q-th selection module, where N and E denote the number and the dimension of a feature, respectively. The q-th selection module takes the current selection mask vector, $\hat{\mathbf{m}}$, and $\mathbf{F}^q$, as input, and outputs the probabilities to drop/keep the features. Firstly, the encoded features are projected by a simple MLP: $\mathbf{Z}^l = f_1(\mathbf{F}^q) \in \mathbb{R}^{N\times E^{'}}$, where $f_1(\cdot)$ denotes the MLP layer, $\mathbf{Z}^l$ denotes the information features with a smaller size than the original features, $E^{'}$ is a smaller dimension than E. In addition, to determine whether to drop or keep a feature, we also need to consider the information of the existing features that are not masked. It is employed to perform information fusion with $\mathbf{Z}^l$ to evaluate the information loss caused by the mask operation. Therefore, the global feature, $\mathbf{z}^g$, that aggregates the information of all existing features that are not masked can be computed by

$$\mathbf{z}^g = \eta\big(f_2(\mathbf{F}^q), \mathbf{m}\big) \in \mathbb{R}^{E^{'}}, \tag{14}$$

where $\eta(\cdot)$ is the function to extract the information of the existing features, $f_2(\cdot)$ denotes the MLP layer. It can be implemented by the average pooling

$$\eta(\mathbf{U}, \hat{\mathbf{m}}) = \frac{\sum_{i=1}^{N} \hat{\mathbf{m}}[i]\mathbf{u}_i}{\sum_{i=1}^{N} \hat{\mathbf{m}}[i]}, \tag{15}$$

where i is the index of the vector and $\mathbf{U} = [\mathbf{u}_1; \mathbf{u}_2; ...; \mathbf{u}_N]$. We expand the global feature as $\mathbf{Z}^g = [\mathbf{z}^g; \mathbf{z}^g; ...; \mathbf{z}^g] \in \mathbb{R}^{N\times E^{'}}$ to perform knowledge fusion between $\mathbf{Z}^g$ and $\mathbf{Z}^l$. The channel noise also affects probability for keeping the features, we employ a MLP layer, $f_3(\cdot)$, to extract

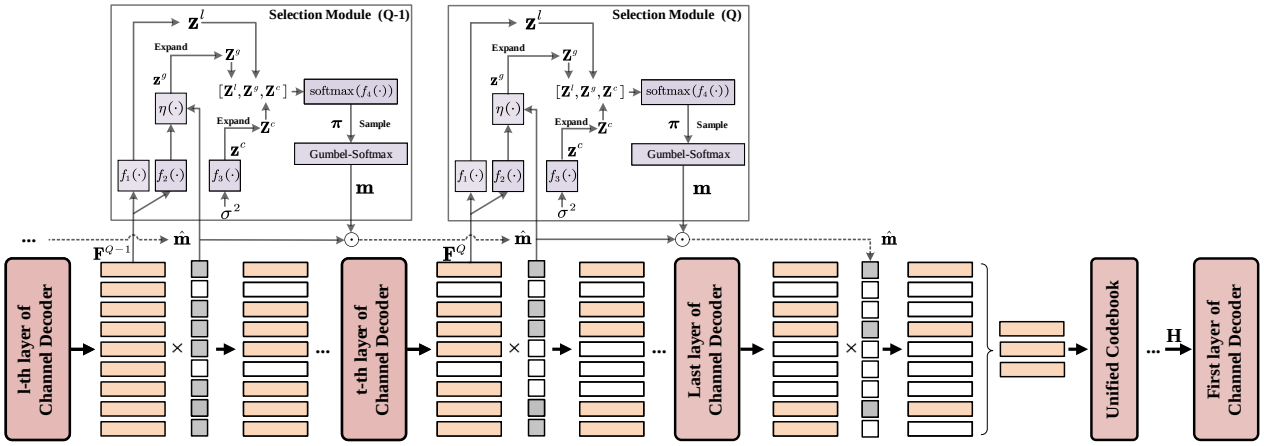


Fig. 6: Dynamic channel encoder for U-DeepSC.

the noise feature, $\mathbf{z}^c = f_3(\sigma^2) \in \mathbb{R}^{E'}$, where σ^2 is the noise variance. Then, we expand $\mathbf{z}^c$ in the same way as $\mathbf{z}^g$, to obtain $\mathbf{Z}^c \in \mathbb{R}^{N \times E'}$. Therefore, we combine the local, global, and noise features to obtain comprehensive features, $\mathbf{Z} = \left[\mathbf{Z}^l, \mathbf{Z}^g, \mathbf{Z}^c\right] \in \mathbb{R}^{N \times 3E'}$ and feed them to another MLP, $f_4(\cdot)$, to predict the probabilities for keeping the features,

$$\boldsymbol{\pi} = \mathrm{softmax}(f_4(\mathbf{Z})) \in \mathbb{R}^{N \times 2}. \tag{16}$$

Then, we obtain the current mask vector, $\mathbf{m}$, by sampling from $\boldsymbol{\pi}$. Note that this module will be inserted into different layers of the transmitter to perform gradually dropping features. Then, the true mask vector, $\hat{\mathbf{m}}$, at the next layer is updated by $\hat{\mathbf{m}} = \hat{\mathbf{m}} \odot \mathbf{m}$, where $\odot$ denotes Hadamard product, it indicates that once a feature is dropped, it will never be used in the following layers.

2) Differentiable sampling: Although the design above is able to achieve feature dimension-wise selection, it is difficult to implement during training. The main obstacle is that the sampling operation from $\boldsymbol{\pi}$ to obtain the selection mask, $\hat{\mathbf{m}}$, is non-differentiable, where $\boldsymbol{\pi} \in \mathbb{R}^{N \times 2}$ denotes the probability matrix, and the elements of the first and second rank is the probability of keeping and dropping these N features, respectively. We take the i-th feature vector as an example, a straightforward way to determine whether to keep it is to sample from $\boldsymbol{\pi}[i, 1]$ and $\boldsymbol{\pi}[i, 2]$. However, the sampling operation is non-differentiable, which hinders the back propagation of the gradients. To address this problem, the Gumbel-Softmax is adopted. It is differentiable and makes it possible to train the selection module. In particular, the selection vector, $\mathbf{m}$, can be obtained by

$$\mathbf{m}[i] = \text{Gumbel-Softmax}(\boldsymbol{\pi}[i])[1]. \tag{17}$$

The input of Gumbel-Softmax is a probability vector with each dimension indicating the probability of sampling the case corresponding to it. Besides, the output of Gumbel-Softmax is an approximate one-hot vector with the same shape as the input, where the element 1 indicates the sampled result. Thus, the output one-hot vector, Gumbel-Softmax($\boldsymbol{\pi}[i]$), has two elements, and we take its first element as the result, as shown by (16). Therefore, $\mathbf{m}$ will be a mask vector sampled from $\boldsymbol{\pi}$, and each element of which is 0 or 1.

D. Training and Inference

The training of U-DeepSC includes training the dynamic channel encoder such that it can produce decisions on whether to transmit some features. In order to provide dynamic transmission overhead under different channel conditions, the ratio of transmitting features is constrained to be a prefined variable value, $\boldsymbol{\rho}\left(\sigma^2\right)$, with respect to the variable channel noise, σ^2. In practical communication systems, the receiver could be aware of the channel conditions via channel feedback. Therefore, the channel condition can be incorporated into the process of feature encoding. In particular, $\boldsymbol{\rho}\left(\sigma^2\right)$ is designed as a non-negative increasing function implemented by the DNNs, then more features will be selected for higher σ^2. Given the specific channel noise, σ^2, we obtain a set of target ratios for Q corresponding selection modules, i.e., $\boldsymbol{\rho}\left(\sigma^2\right) = [\rho^1, ..., \rho^Q]$. We apply the MSE loss to supervise the prediction module:

$$\mathcal{L}_{\mathrm{r}} = \frac{1}{Q}\sum_{q=1}^{Q}\left(\rho^q - \frac{1}{N}\sum_{i=1}^{N}\hat{\mathbf{m}}[i]^q\right)^2, \tag{18}$$

where $\frac{1}{N}\sum_{i=1}^{N}\hat{\mathbf{m}}[i]^q$ denotes the true ratio of transmitting features. By supervising the true ratio with the target ratio, only the target ratio of features will be processed by the decoder for training. However, if we apply loss (17) directly, more features will be selected as the training goes on, since the model tend to keep more features to improve the performance. To balance the performance and the number of transmitted symbols, we add the l_1-norm of $\boldsymbol{\rho}\left(\sigma^2\right)$ to loss (17). It makes the model keep fewer features with the decrement of the $\boldsymbol{\rho}\left(\sigma^2\right)$, and enables the model to achieve a good performance by only transmitting a part of features.

In the inference phase, given the variance of channel noise, we can directly drop the less informative features via the probabilities produced by the selection modules such that only $N\rho^Q$ features are transmitted to the receiver. In particular, only the top $N\rho^Q$ features with the highest probability will be transmitted for each task.

V. Task Description and Training

A. Task Description

To provide a thorough analysis of U-DeepSC and sufficient results to demonstrate the effectiveness, we will experiment with jointly handling prominent tasks from different domains, including sentiment analysis, visual question answering (VQA), image classification, video sentiment analysis, image data reconstruction, and text data reconstruction tasks. Note that these tasks have been widely considered in the existing semantic comunication systems [7], [15], [18].

1) Sentiment analysis: The purpose of the sentiment analysis 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 sentiment analysis, and the cross-entropy as the loss function to train the model.

2) VQA: In VQA 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.

3) Video sentiment analysis: The video sentiment analysis task is about leveraging multimodal signals for an effective understanding of the videos generated by users [32]. As for the evaluation criterion, classification accuracy is selected as the metric. Additionally, the cross-entropy is used for the loss function.

4) 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.

5) 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

$$\text{PSNR} = 10\log_{10}\frac{\text{MAX}^2}{\text{MSE}}(\text{dB}), \tag{19}$$

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

Algorithm 1: Training procedures of the U-DeepSC

Input : Six training datasets consist of input images, texts, speech with their labels.

Output : The trained U-DeepSC model with encoders and decoders.

for $i \leftarrow 1$ **to** N **do**
- Randomly choose two tasks and generate two mini-batch samples.
- Sample the channel variance, σ^2, from the given SNR range.
- Generate the selection mask vectors, $\mathbf{m}^A$ and $\mathbf{m}^B$, before transmitting, respectively.
- Compute the prediction loss, $\mathcal{L}_r = \mathcal{L}_r^A + \mathcal{L}_r^B$, based on (18).
- Continue forward propagation with the generated mask.
- **if** *i%2==1* **then**
  - Compute the task-specific losses, $\mathcal{L}_{p_1}^A$ and $\mathcal{L}_{p_1}^B$, based on (6).
  - Compute the unified codebook losses, $\mathcal{L}_c^A$ and $\mathcal{L}_c^B$, based on (12).
  - Compute the total loss $\mathcal{L} = \mathcal{L}_{p_1}^A + \mathcal{L}_{p_1}^B + \mathcal{L}_c^A + \mathcal{L}_c^B + \mathcal{L}_r$.
- **else**
  - Compute the task-specific losses, $\mathcal{L}_{p_2}^A$ and $\mathcal{L}_{p_2}^B$, based on (6).
  - Compute the unified codebook losses, $\mathcal{L}_c^A$ and $\mathcal{L}_c^B$, based on (12).
  - Compute the total loss, $\mathcal{L} = \mathcal{L}_{p_2}^A + \mathcal{L}_{p_2}^B + \mathcal{L}_c^A + \mathcal{L}_c^B + \mathcal{L}_r$.
- **end**
- **if** *task A and task B share the same modality* **then**
  - Compute the domain adaptation loss, $\mathcal{L}_a$, based on (13).
  - Compute the total loss, $\mathcal{L} = \mathcal{L} + \mathcal{L}_a$.
- Update the parameters of U-DeepSC with $\mathcal{L}$.

end

6) 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.

B. Training Method

To jointly learn the above six tasks, we propose an efficient method to train the modules in the U-DeepSC system with domain adaptation. We will randomly take samples from two tasks, task

A and task B at one time, and the task with a larger dataset will be assigned a higher sampling probability since they are generally more difficult to learn. We employ the superscript to indicate which task the loss function belongs to, e.g., $\mathcal{L}^A$ denotes the loss of task A. Firstly, to learn the channel encoder, we sample the noise variance from the considered SNR range, and supervise the selection module with the loss function, $\mathcal{L}_r$. Then, for each task, denote the objective function in Problem (6) as $\mathcal{L}_{p_1}$ and $\mathcal{L}_{p_2}$, respectively. We first train the model with $\mathcal{L}_{p_1}$, and then employ the $\mathcal{L}_{p_1}$ at the second phase. They are added by the loss of the unified codebook, $\mathcal{L}_c$, at the corresponding phase. Moreover, we learn task A and task B with the domain adaptation loss, $\mathcal{L}_a$, if they share the same modality. The detailed training procedures are summarized in Algorithm 1.

VI. Simulation Results

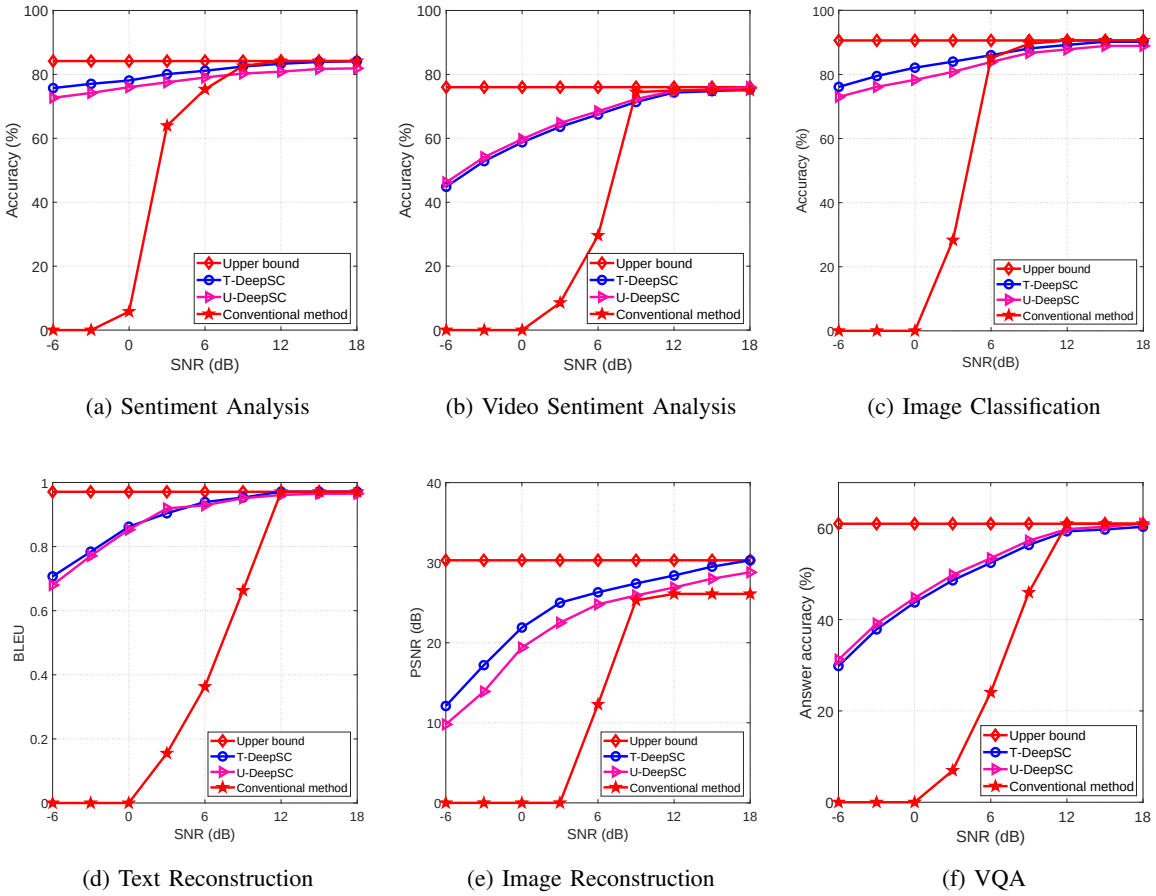
In this section, we demonstrate the superiority of the proposed U-DeepSC by numerical results.

A. Simulation Setup

The image semantic encoder of U-DeepSC has four Transformer encoder layers, the text semantic encoder is designed with four Transformer encoder layers, and the speech semantic encoder consists of six Transformer encoder layers. The unified semantic decoder consists of eight Transformer decoder layers. The setting of the training procedure is the AdamW optimizer with learning rate 1×10^{-4}, batch size 32, weight decay 5×10^{-3}. Moreover, in the simulation, we will focus on AWGN channels.

To verify the effectiveness of U-DeepSC, we test our U-DeepSC on the aforementioned six tasks, and each corresponding to a datasets. In particular, CIFAR-10 datasets are adopted for image classification and image reconstruction tasks, respectively. For the text reconstruction and sentiment analysis, the proceedings of the European Parliament and the SST-2 datasets are used, respectively. Moreover, the VQAv2 dataset is used for the VQA task. For the video task, we employ the MOSI dataset. 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 the low-density parity-check code (LDPC) are adopted as image source coding and image channel coding, respectively. In addition, for the images in the video, we adopt the H.264 video compression codecs for



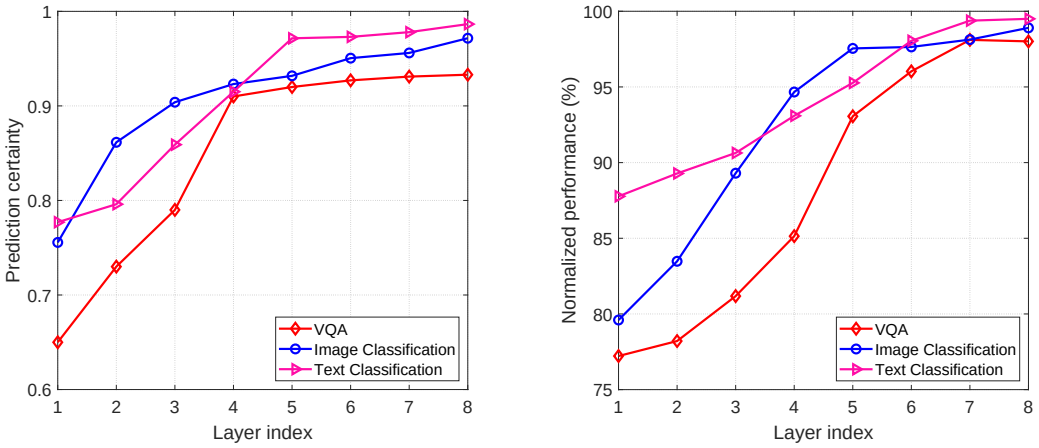
(a) Sentiment Analysis (b) Video Sentiment Analysis (c) Image Classification

(d) Text Reconstruction (e) Image Reconstruction (f) VQA

Fig. 7: The performance of six tasks versus SNR.

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.



(a) Predicted certainty at different decoder layers.

(b) The normalized performance of different tasks that exit at a fixed layer.

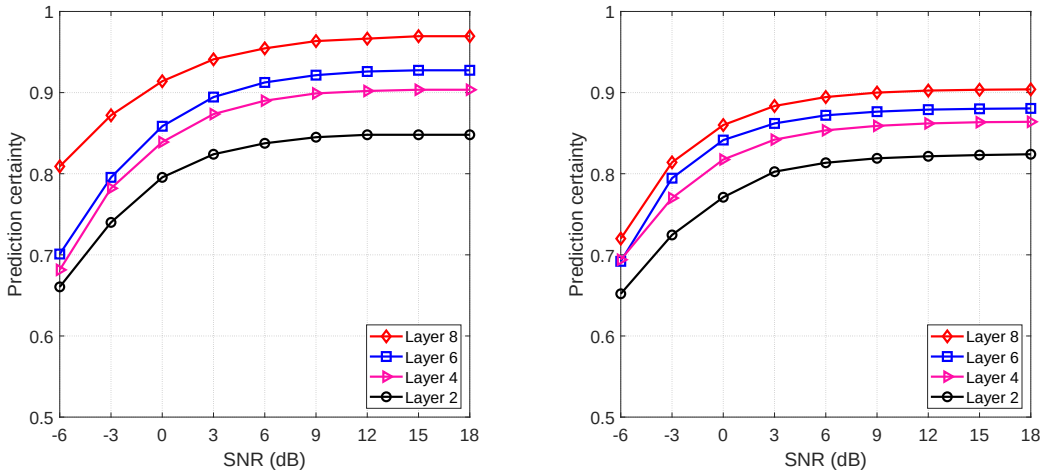
Fig. 8: The predicted certainty and the normalized performance of different tasks at different decoder layers.

B. Task Performance

Fig. 7 illustrates the performance of the investigated schemes versus the SNR for different tasks. The proposed U-DeepSC is trained with SNR ranging from 0 dB to 12 dB, and tested in SNR from -6 dB to 18 dB. 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 is close to the performance of 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. Since only a specific part of the overall features are transmitted in U-DeepSC for different tasks, the satisfactory performance of U-DeepSC shows that the task-specific semantic information can be well specified by U-DeepSC.

C. Multi-Exit Performance

As shown in Fig. 8(a), as the layer index increases, the predicted certainty output by the LCM firstly increases dramatically, then the increasing trend slows down after certain layer, e.g., 4 for the VQA task. It indicates the required number of layers for different samples from different



(a) Predicted certainty of image classification task at different layers versus SNR.

(b) Predicted certainty of VQA task at different layers versus SNR.

Fig. 9: The predicted certainty of the tasks versus SNR.

tasks. Besides, it also shows that the proposed multi-exit architecture can dynamically identify the proper early-exit layers and achieve satisfactory trade-off between the performance and the executed number of layers. Fig. 8(b) depicts the normalized performance of the task output at a fixed layer, i.e., all samples are required to exit at this layer. Note that the normalized performance is adopted so that different tasks with different performance metrics can be put into one figure for comparison, which is computed by dividing the achieved performance by the upper bound. From the figure, the normalized performance of all of the tasks increases as the network gets deeper. Moreover, by comparing the curves in Fig. 8(a) and Fig. 8(b), the predicted certainty and normalized performance are positively correlated, which demonstrates the effectiveness of the LCM.

Fig. 9 presents the predicted certainty for the image classification task and the VQA task versus SNR at different layers of the decoder. From the figure, the predicted certainty increases with SNR, which provides a metric to perform early-exit under different channel conditions. In particular, a larger number of layers is able to provide higher certainty, which is required to eliminate the impact of the channel noise in the low SNR scenario.

In Fig. 10, we compare the proposed U-DeepSC with *Fixed Layer*, where all samples exit at a certain given layer. Given a fixed index of layer, we control the threshold, δ_e, to adjust the

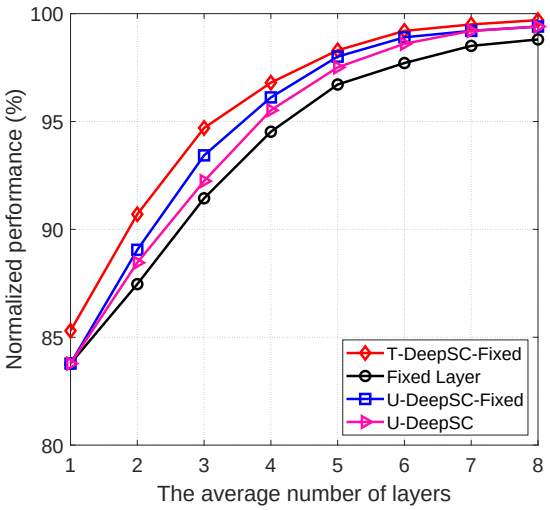


Fig. 10: Normalized performance versus the average number of layers.

average number of layers of all testing samples in U-DeepSC. Moreover, the *U-DeepSC-Fixed* and *T-DeepSC-Fixed* in the figure refer to training the model with a fixed number of layers. The *U-DeepSC-Fixed* can be considered as an upper bound for the U-DeepSC since the model is trained specifically to achieve the best performance with the fixed layer. Moreover, the normalized performance of these schemes increases with the average number of layers. We can see that the U-DeepSC achieves better performance than the *Fixed Layer*, demonstrating the effectiveness of the dynamic channel encoder to dynamically handle different samples and tasks with the adaptive number of layers of the decoder.

D. Effectiveness of Unified Codebook

Fig. 11 depicts the performance of U-DeepSC versus different codebook sizes. In particular, *Joint* refers to jointly training these tasks with just one codebook, *Joint+DA* refers to jointly training these tasks with just one codebook aided by domain adaptation, *Sep-M* refers to designing a codebook for each modality separately, and *Sep-T* denotes equipping each task with a codebook. Moreover, U-DeepSC is the proposed method with domain adaptation. From the figure, the U-DeepSC significantly outperforms other benchmarks, and the performance gap between *Joint* and *Sep-M* demonstrates the necessity to design a codebook for each modality. It is mainly because that the statistics of data from different modalities differ from each other significantly. Moreover,

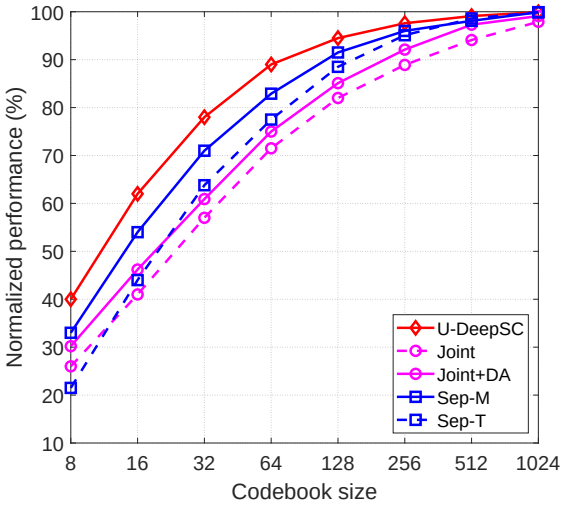
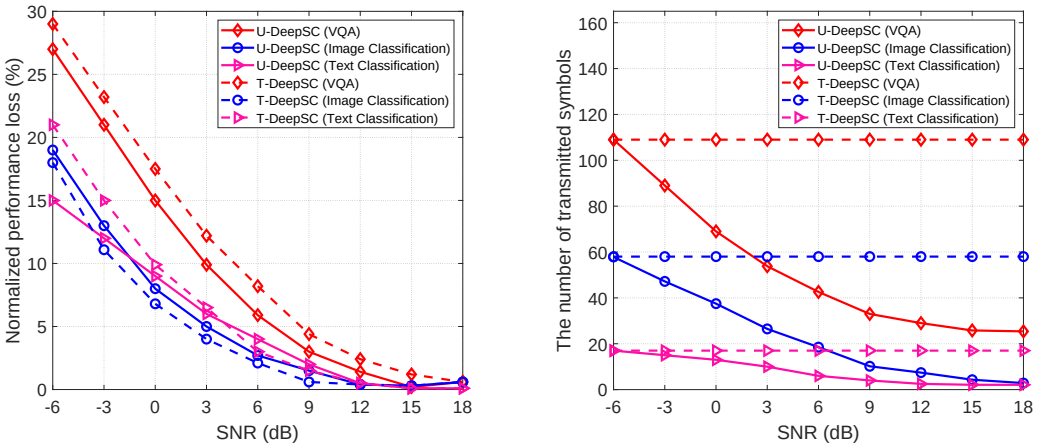


Fig. 11: Normalized performance versus the size of codebook.

the *Sep-M* is close to the performance of U-DeepSC and outperforms *Sep-T*. Therefore, the semantic information among different tasks from one modality can be shared.



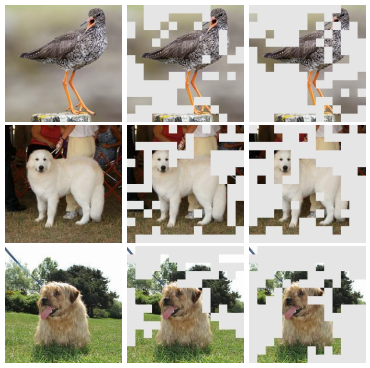
(a) Normalized performance loss versus SNR.

(b) The number of transmitted symbols versus SNR.

Fig. 12: Normalized performance loss and transmitted symbols as a function of the channel SNR in dynamic channel conditions.

E. Overhead Analysis

Fig. 12 illustrates 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 soes 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 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.



(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. 13: The visualization of the feature selection.

We conduct feature visualization in Fig. 13. 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 the features corresponding to the emotional words, e.g., good and bad, are more likely

to be kept in the text sentiment analysis task. These emotional words usually play an important role in identifying the sentiment of the sentence.

TABLE I: The number of transmitted symbols for one image/sentence in different tasks.

Task	Conventional method	U-DeepSC
Image Reconstruction	53,760	1,568
Image Classification	53,760	6
Text Reconstruction	580	590
Sentiment Analysis	335	4
VQA	123,430	25

Table I presents the transmission overhead of U-DeepSC and the conventional method. For image-related tasks, the proposed U-DeepSC significantly reduces the transmission overhead, especially for the image classification task. For text-related task, the U-DeepSC transmit a similar number of symbols compared with the benchmark in text reconstruction task, since there is little redundancy for a precise text reconstruction task. In contrast, the proposed U-DeepSC decrease the transmitted symbols in the sentiment analysis task, where only the semantic information of the emotional words is required. Moreover, U-DeepSC significantly reduced the transmitted symbols in the multimodal VQA task. Our proposed method can handle these tasks with a much lower number of transmitted symbols.

F. Model Parameters

TABLE II: The number of parameters.

Task	T-DeepSC	U-DeepSC
Image Classification	32.6M	84.1M
Image Reconstruction	32.9M	84.1M
Sentiment Analysis	48.3M	84.1M
Text Reconstruction	48.2M	84.1M
VQA	63.0M	84.1M
Video Sentiment Analysis	83.8M	84.1M
Stored Parameters	308.8 M	84.1M

As shown in Table II, the total stored number of parameters of T-DeepSC is 308.8M, which is obtained by adding the parameters required for each task. For our proposed U-DeepSC, the

number of stored model parameters is only 84.1M for six tasks, which is 72.7% less than that 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.

VII. 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. Then, we developed a multi-exit architecture for U-DeepSC to provide early-exit results for relatively simple tasks and good channel conditions. To reduce the transmission overhead, the unified codebook with domain adaptation has been proposed for the feature representation of multiple tasks. Additionally, we further developed a novel dimension-wise selection module 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. Simulation results showed that our proposed model had satisfactory performance in 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] Z. Qin, X. Tao, J. Lu, W. Tong, and G. Y. Li, "Semantic communications: Principles and Challenges," *arXiv preprint arXiv:2201.01389*, 2021.

[2] 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.

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

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

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

[6] 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.

[7] 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.

[8] 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.

[9] T.-Y. Tung and D. Gündüz, "DeepWiVe: Deep-learning-aided wireless video transmission," *arXiv preprint arXiv:2111.13034*, 2021.

[10] 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.

[11] 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)*, 2021, pp. 1–6.

[12] 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)*, 2020, pp. 1–5.

[13] S. Wang, J. Dai, Z. Liang, K. Niu, Z. Si, C. Dong, X. Qin, and P. Zhang, "Wireless deep video semantic transmission," *arXiv preprint arXiv:2205.13129*, 2022.

[14] P. Jiang, C. K. Wen, S. Jin, and G. Y. Li, "Wireless semantic communications for video conferencing," *arXiv preprint arXiv:2204.07790*, 2022.

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

[16] 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.

[17] Q. Hu, G. Zhang, Z. Qin, Y. Cai, G. Yu, and G. Y. Li, "Robust semantic communications with masked VQ-VAE enabled codebook," *arXiv preprint arXiv:2206.04011*, 2022.

[18] H. Xie, Z. Qin, X. Tao, and K. B. Letaief, "Task-oriented multi-user semantic communications," *IEEE J. Select. Areas Commun.*, pp. 1–1, 2022.

[19] D. B. Kurka and D. Gunduz, "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.

[20] M. Wang, Z. Zhang, J. Li, M. Ma, and X. Fan, "Deep joint source-channel coding for multi-task network," *arXiv preprint arXiv:2109.05779*, 2021.

[21] 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, 2021.

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

[23] A. Bakhtiarnia, Q. Zhang, and A. Iosifidis, "Multi-exit vision transformer for dynamic inference," *arXiv preprint arXiv:2106.15183*, 2021.

[24] J. Xin, R. Tang, Y. Yu, and J. J. Lin, "BERxiT: Early exiting for bert with better fine-tuning and extension to regression," in *Proc. Conf. European Chapter of the Assoc. for Computational Linguistics (EACL)*, 2021.

[25] K. He, X. Chen, S. Xie, Y. Li, P. Dollar, and R. Girshick, "Masked autoencoders are scalable vision learners," *arXiv preprint arXiv:2111.06377*, 2021.

[26] 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.

[27] M. Wang and W. Deng, "Deep visual domain adaptation: A survey," *Neurocomputing*, vol. 312, pp. 135–153, 2018.

[28] B. Sun and K. Saenko, "Deep CORAL: Correlation alignment for deep domain adaptation," in *Proc. Eur. Conf. Comput. Vis. Workshop (ECCVW)*, 2016, pp. 443–450.

[29] D. Molchanov, A. Ashukha, and D. Vetrov, "Variational dropout sparsifies deep neural networks," in *Proc. Int. Conf. Mach. Learn*, 2017, pp. 2498–2507.

[30] J. M. Alvarez and M. Salzmann, "Learning the number of neurons in deep networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2016, pp. 2270–2278.

[31] 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.

[32] 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.