

A Lite Distributed Semantic Communication System for Internet of Things

Huiqiang Xie, *Graduate Student Member, IEEE*, and Zhijin Qin, *Member, IEEE*

Abstract—The rapid development of deep learning (DL) and widespread applications of Internet-of-Things (IoT) have made the devices smarter than before, and enabled them to perform more intelligent tasks. However, it is challenging for any IoT device to train and run DL models independently due to its limited computing capability. In this paper, we consider an IoT network where the cloud/edge platform performs the DL based semantic communication (DeepSC) model training and updating while IoT devices perform data collection and transmission based on the trained model. To make it affordable for IoT devices, we propose a lite distributed semantic communication system based on DL, named L-DeepSC, for text transmission with low complexity, where the data transmission from the IoT devices to the cloud/edge works at the semantic level to improve transmission efficiency. Particularly, by pruning the model redundancy and lowering the weight resolution, the L-DeepSC becomes affordable for IoT devices and the bandwidth required for model weight transmission between IoT devices and the cloud/edge is reduced significantly. Through analyzing the effects of fading channels in forward-propagation and back-propagation during the training of L-DeepSC, we develop a channel state information (CSI) aided training processing to decrease the effects of fading channels on transmission. Meanwhile, we tailor the semantic constellation to make it implementable on capacity-limited IoT devices. Simulation demonstrates that the proposed L-DeepSC achieves competitive performance compared with traditional methods, especially in the low signal-to-noise (SNR) region. In particular, while it can reach as large as 40x compression ratio without performance degradation.

Index Terms—Internet of Things, neural network compression, pruning, quantization, semantic communication.

I. INTRODUCTION

With the widely deployed connected devices, Internet of Things (IoT) networks are providing more and more intelligent services, i.e., smart home, intelligent manufacturing, and smart cities, by processing a massive amount of data generated by those connected devices [1], [2]. Deep learning (DL) [3] has demonstrated great potentials in processing various types of data, i.e., images and texts. The DL-enabled IoT devices are capable of exploiting and processing different types of data more effectively as well as handling more intelligent tasks than before. Although some IoT devices have certain capability to process simple DL models, the limited memory, computing, and battery capability still prevent from wide applications of DL [4]. Therefore, the burden of DL model updates is usually transferred to the cloud/edge platform [5]. Particularly, the DL model is trained at the cloud/edge platform based on data from the IoT devices, and then the trained model is distributed

to IoT devices. However, data transmitted over the air could be distorted by wireless channels, which may cause improper trained results, i.e., local optimum. Moreover, the large number of parameters in DL models leads to high latency when distributing the DL models with limited bandwidth. Therefore, transmitting accurate data to the cloud/edge platform over wireless channels for model training and reducing the number of parameters in DL models for lower latency and power consumption at the IoT devices are two crucial problems.

To address the first problem on accurate data transmission in an IoT network, semantic communication system, which interprets information at the semantic level rather than bit sequences [6], is promising. To make a decision based on the received information, there are usually three steps: i) the traditional communication receiver to recover the raw data [7]; ii) the feature extractor to obtain and interpret the meanings of the raw data for the decision [8]; and iii) the effects network to produce the desired effects according to the extracted features [9], [10]. Correspondingly, the communication is categorized into three levels [11], including transmission level to guarantee the transmission accuracy of bit sequence, semantic level to guarantee the exchange of semantic information, and effectiveness level to measure the corresponding effects or caused actions of transmitted information, i.e., network re-configuration, which is illustrated in Fig. 1. The traditional communication system works at the transmission level shown in Fig. 1(a), which aims at transmitting and receiving symbol accurately [12]. The followed feature extractor network and effect networks are designed separately based on applications. However, designing these modules separately may lead to *error propagation* and prevent from reaching joint optimality. For example, the feature network is not able to correct errors from the traditional receiver, which will affect the subsequent decision making in the effect network. Thus, through designing the traditional receiver and feature extractor network jointly (the semantic level) or merging traditional receiver, feature extractor network, and effects network together (the effectiveness level), communication systems have the capability of error correction at the semantic level and effectiveness level, respectively. In this paper, we will focus on distributed semantic communications for IoT networks and leave effectiveness level communication to future research.

With the recent advancements on DL, it is promising to represent a traditional transceiver or each individual signal processing block by a deep neural network (DNN) [13]. Inspired by the autoencoder in DL techniques, an end-to-end (E2E) communication system has been proposed to merge the signal processing blocks in traditional communication [14].

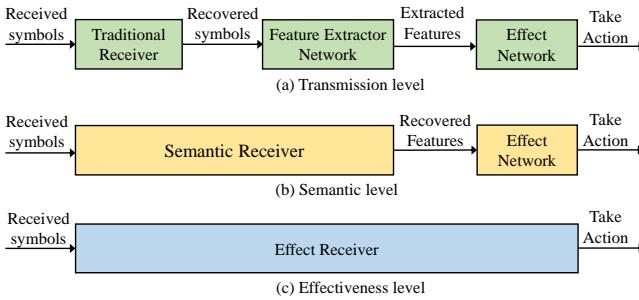


Fig. 1. Illustration of three communication levels at the receiver.

Missing channel gradients becomes the bottleneck of training E2E communication systems. There are several works for mitigating this problem [15]–[17]. Dörner *et al.* proposed a two-phase training processing [15] by training the transceiver with a stochastic channel model firstly, and fine-tuning the receiver over real channels. Aoudia *et al.* estimated the channel gradients by sampling from a relaxed distribution based on stochastic reinforce learning policy [16], where the transmitter and receiver are trained separately. Ye *et al.* proposed generative adversarial network (GAN) to approximate the unknown channel model [17] so that the channel gradients can be estimated by the GAN.

There have been some initial works related to deep semantic communications [18]–[22]. Bourtsoulatze *et al.* [18] proposed joint source-channel coding for wireless image transmission based on the convolutional neural network (CNN), where peak signal-to-noise ratio (PSNR) is used to measure the accuracy of image recovery at the receiver. Taking image classification tasks into consideration, Lee *et al.* [19] developed a transmission-recognition communication system by merging wireless image transmission with the effect network as DNNs, i.e., image classification, which achieves higher image classification accuracy than performing them separately. For texts, Farsad *et al.* [21] designed joint source-channel coding for erasure channel by using a recurrent neural network (RNN) and a fully-connected neural network (FCN), where the system recovers the text directly rather than perform channel and source decoding separately. In order to understand texts better and adapt to dynamic environments, Xie *et al.* [22] developed a semantic communication system based on Transformer, named DeepSC, which clarifies the concepts of semantic information and semantic error at the sentence-level for the first time. In brief, compared with traditional approaches, the semantic communication systems are more robust to channel variation and are able to achieve better performance in terms of source recovery and image classification, especially in the low signal-to-noise (SNR) regime.

To deal with the second problem in reducing the number of parameters, network slimmer has attracted extensive attention to compress DL models without degrading performance since neural networks are usually over-sized [23]. *Parameters pruning* and *quantization* are two main approaches for DL model compression. Parameter pruning is to remove the unnecessary connections between two neurons or important neurons. Han *et al.* [24] proposed an iterative pruning approach, where the

model is trained first, then pruned by a given threshold, and is fine-tuned to recover performance in terms of image classification. This approach could reduce the connections without losing accuracy. Liu *et al.* [25] proposed to prune the filters in CNN by training the model with the L_1 regularization so that redundancy weights converge to zero directly without sacrificing the performance. By analyzing the connection sensitivity among neurons and layers, Li *et al.* [26] remove the insensitive layers, which further increases inference speed. By applying these pruning approaches, DL models can be compressed by 13 to 20 times. Quantization aims to represent a weight parameter with lower precision (fewer bits), which reduces the required bitwidth of data flowing through the neural network model in order to shrink the model size for memory saving and simplify the operations for computing acceleration [27]. With vector quantization, Gong *et al.* [28] quantize the DL models. Similarly, Zhou *et al.* [29] investigated an iterative quantization, which starts with a trained full-resolution model and then quantizes only a portion of the model followed by several epochs of re-training to recover the accuracy loss from quantization. A mix precision quantization by Li *et al.* [30] quantizes weights while keeping the activations at full-resolution. The training algorithm by Jacob *et al.* [31] preserves the model accuracy after post-quantization. With the quantization, the weights can generally be compressed from 32-bit to 8-bit without performance loss. Similarly, pruning and quantizing can be also used in DL-enabled communication systems. For example, Guo *et al.* [32] have shown that model compression can accelerate the processing of channel state information (CSI) acquisition and signal detection in massive multiple-input multiple-output (MIMO) systems without performance degradation.

Through applying network slimmer into our existing work DeepSC, the aforementioned two challenges in IoT networks can be effectively addressed. Although the above works validate the feasibility, we still face the following issues to make it affordable for IoT devices:

- *Question 1: How to design semantic communication systems over wireless fading channels?*
- *Question 2: How to form the constellation to make it affordable for capacity-limited IoT devices?*
- *Question 3: How to compress semantic models for fast-model transmission and low-cost implementation on IoT devices?*

In this paper, we design a distributed semantic communication system for IoT networks. Especially, a lite DeepSC is proposed (L-DeepSC) to address the above questions. Different from our previous work [22], this work solves the training DeepSC problem over fading channels with imperfect CSI and considers different wireless channel models to show the generalization of our method. Moreover, this work extends [22] to a more practical IoT scenario, where two key problems, model updating, and broadcasting, are solved. This work also addresses the issue of the finite constellation sizes for capacity-constrained IoT devices while [22] assumes infinite constellations. The main contributions of this paper are summarized as follows.

- We design a distributed semantic communication network under power and latency constraints, in which the receiver and feature extractor networks are jointly optimized by overcoming fading channels.
- By identifying the impacts of CSI on DL model training over fading channels, we propose a CSI-aided semantic communication system to speed up convergence, where the CSI is refined by a de-noise neural network. This addresses the aforementioned *Question 1*.
- To make data transmission and receiving affordable for capacity-constrained devices, we design a finite-bits constellation to solve *Question 2*.
- Due to over-parametrization, we propose a model compression algorithm, including network sparsification and quantization, to reduce the size of DL models by pruning the redundancy connections and quantizing the weights, which addresses the aforementioned *Question 3*.

The rest of this paper is organized as follows. The distributed semantic communication system model is introduced and the corresponding problems are identified in Section II. Section III presents the proposed L-DeepSC. Numerical results are used to verify the performance of the proposed L-DeepSC in Section IV. Finally, Section V concludes this paper.

Notation: $\mathbb{C}^{n \times m}$ and $\mathbb{R}^{n \times m}$ represent the sets of complex and real matrices of size $n \times m$, respectively. Bold-font variables denote matrices or vectors. $x \sim \mathcal{CN}(\mu, \sigma^2)$ means variable x follows the circularly-symmetric complex Gaussian distribution with mean μ and covariance σ^2 . $(\cdot)^T$ and $(\cdot)^H$ denote the transpose and Hermitian of a vector or a matrix, respectively. $\Re\{\cdot\}$ and $\Im\{\cdot\}$ refer to the real and the imaginary parts of a complex number.

II. SYSTEM MODEL AND PROBLEM FORMULATION

Text is an important type of source data, which can be sensed from speaking and typing, environmental monitoring, etc. By training DL models with these text data at cloud/edge platform, the DL models based IoT devices have the capability to understand text data and generate semantic feature to be transmitted to the center to perform intelligent tasks, i.e., intelligent assistants, human emotion understanding, and environment humid and temperature adjustment based on human preference [33].

As shown in Fig. 2(a), we focus on distributed semantic communications for IoT networks. The considered system is consisted of various IoT networks with two layers, the cloud/edge platform and distributed IoT devices. The cloud/edge platform is equipped with huge computation power and big memory, which can be used to train the DL model by the received semantic features. The semantic communication enabled IoT devices to perform intelligent tasks by understanding sensed texts, which are with limited memory and power but expected long lifetime, i.e., up to 10 years. Particularly, our considered distributed semantic communication system consists of the following three steps:

- 1) **Model Initialization/Update:** The cloud/edge platform first trains the semantic communication model by initial dataset. The trained model is updated in the subsequent

iterations by the received semantic features from IoT devices.

- 2) **Model Broadcasting:** The cloud/edge platform broadcasts the trained DL model to each IoT device.
- 3) **Semantic Features Upload:** The IoT devices constantly capture the text data, which are encoded by the proposed semantic transmitter shown in Fig. 2(b). The extracted semantic features are then transmitted to the cloud/edge for model update and subsequent processing.

The aforementioned *Questions 1-3* correspond to model initialization/update, semantic features uploading, and model broadcasting, respectively. Different from the traditional information transmission, semantic features can be not only used for recovering the text at the semantic level accurately, but also exploited as the input of other modules, i.e., emotion classification, dialog system, and human-robot interaction, for training effect networks and perform various intelligent tasks directly. The devices can also exchange semantic features, which has been previously discussed in our work in [22]. We focus on the communication between cloud/edge platforms and local IoT devices to make the semantic communication model affordable.

A. Semantic Communication System

The DeepSC shown in Fig. 2(b) can be divided into three parts mainly, transmitter network, physical channel, and receiver network, where the transmitter network includes semantic encoder and channel encoder, and the receiver network consists of semantic decoder and channel decoder.

We assume that the input of the DeepSC is a sentence, $\mathbf{s} = [w_1, w_2, \dots, w_N]$, where w_n represents the n -th word in the sentence. The encoded symbol stream can be represented as

$$\mathbf{X} = C_{\alpha}(S_{\beta}(\mathbf{s})), \quad (1)$$

where $S_{\beta}(\cdot)$ is the semantic encoder network with parameter set β and $C_{\alpha}(\cdot)$ is the channel encoder with parameter set α .

If \mathbf{X} is sent through a wireless fading channel, the signal received at the receiver can be given by

$$\mathbf{Y} = f_{\mathbf{H}}(\mathbf{X}) = \mathbf{H}\mathbf{X} + \mathbf{N}, \quad (2)$$

where \mathbf{H}^1 represents the channel gain between the transmitter and the receiver, and $\mathbf{N} \sim \mathcal{N}(0, \sigma_n^2)$ is additive white Gaussian noise (AWGN).

The decoded signal can be represented as

$$\hat{\mathbf{s}} = S_{\chi}^{-1}(C_{\delta}^{-1}(\mathbf{Y})), \quad (3)$$

where $\hat{\mathbf{s}}$ is the recovered sentence, $C_{\delta}^{-1}(\cdot)$ is the channel decoder with parameter set δ and $S_{\chi}^{-1}(\cdot)$ is the semantic decoder network with parameter set χ , the superscript -1 represents the decoding operation.

¹Here, we have omitted discussion of complex channels. If the complex channel is $\tilde{\mathbf{H}}$, then $\tilde{\mathbf{H}} = [\Re(\mathbf{H}), -\Im(\mathbf{H}); \Im(\mathbf{H}), \Re(\mathbf{H})]$.

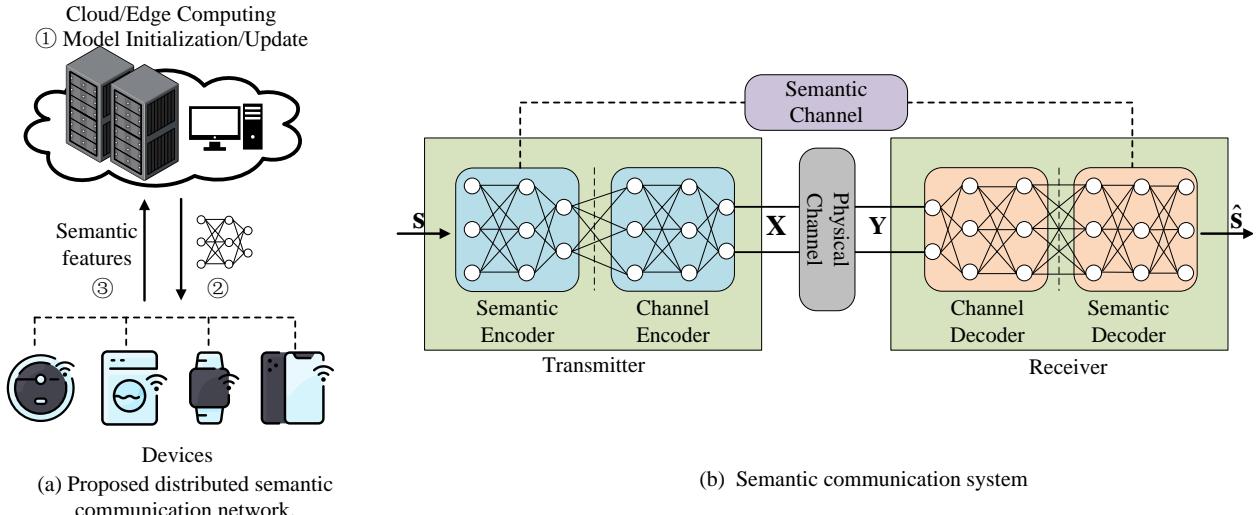


Fig. 2. The framework of semantic communications for IoT networks.

The whole semantic communication can be trained by the cross-entropy (CE) loss function, which is given by

$$\mathcal{L}_{\text{CE}}(\mathbf{s}, \hat{\mathbf{s}}) = \sum_{i=1} (q(w_i) - 1) \log(1 - p(w_i)) - \sum_{i=1} q(w_i) \log(p(w_i)), \quad (4)$$

where $q(w_i)$ is the real probability that the i -th word, w_i , appears in source sentence \mathbf{s} , and $p(w_i)$ is the predicted probability that the i -th word, w_i , appears in $\hat{\mathbf{s}}$. CE can measure the difference between the two distributions. Through minimizing the CE loss, the network can learn the word distribution, $q(w_i)$, in the source sentence, \mathbf{s} . Consequently, the syntax, phrase, and the meaning of words in the context can be learnt by DNNs.

B. Problem Description

Instead of bits, the input sentence, \mathbf{s} , in the DeepSC, will cause that the learned constellation is no longer limited to a few points anymore. After transmitting \mathbf{X} , the fading channel increases the difficulty of model training compared with the AWGN channel. Meanwhile, the huge number of parameters, $\alpha, \beta, \chi, \delta$, indicates the complexity of the whole model. These factors limit DeepSC for IoT networks and incur the aforementioned *Questions 1-3*, including feasible constellation design, training for fading channel, and model compression.

1) *Training of fading channel*: In DL, the training process can be divided forward-propagation to predict the target and back-propagation to converge the neural network, as stated in the following.

Forward-propagation: From the received signal to recover semantic information, the estimation sentence is given by

$$\hat{\mathbf{s}} = S_{\chi}^{-1}(C_{\delta}^{-1}(\mathbf{Y})), \quad (5)$$

Back-propagation: Taking semantic encoder as an example, the parameter vector at the t_{th} iteration are updated by

$$\beta(t) = \beta(t-1) - \eta \frac{\partial \mathcal{L}_{\text{CE}}}{\partial \beta}, \quad (6)$$

where η is the learning rate and $\frac{\partial \mathcal{L}_{\text{CE}}}{\partial \beta}$ is the gradient, computed by

$$\begin{aligned} \frac{\partial \mathcal{L}_{\text{CE}}}{\partial \beta} &= \frac{\partial \mathcal{L}_{\text{CE}}}{\partial \hat{\mathbf{s}}} \frac{\partial \hat{\mathbf{s}}}{\partial \mathbf{Y}} \frac{\partial \mathbf{Y}}{\partial \mathbf{X}} \frac{\partial \mathbf{X}}{\partial \beta} \\ &= \frac{\partial \mathcal{L}_{\text{CE}}}{\partial \hat{\mathbf{s}}} \frac{\partial \hat{\mathbf{s}}}{\partial \mathbf{Y}} \mathbf{H} \frac{\partial \mathbf{X}}{\partial \beta}. \end{aligned} \quad (7)$$

In (7), \mathbf{H} will introduce stochasticity during weight updating. For an AWGN channel, $\mathbf{H} = \mathbf{I}$ will not affect it. However, for fading channels, \mathbf{H} is random, which may lead to that β fails to converge to the global optimum while the forward-propagation in (5) is unable to recover semantic information accurately based on the local optimum. Thus, it is critical to design the training process to mitigate the effects of \mathbf{H} , which also makes the DeepSC applicable for fading channels.

2) *Feasible constellation design*: Generally, the DL models run on floating-point operations (FLOPs), which means that the input, output, and weights are in a large range of $\pm 1.40129 \times 10^{-45}$ to $\pm 3.40282 \times 10^{+38}$ [34]. Although DeepSC can learn the constellations from the source information and channel statistics, the learned constellation points, such as cluster constellation [35], are disordered in the range of $\pm 1.40129 \times 10^{-45}$ to $\pm 3.40282 \times 10^{+38}$, which brings additional burden to the hardware of IoT devices, for instance, the high-resolution phase-shift and amplitude-shift pose high requirements on the circuit. Therefore, it is desired to form feasible constellations with only finite points for the current radio frequency (RF) systems. In other words, we have to design a smaller constellation for the DeepSC.

3) *Model communication*: The more parameters DeepSC has, the stronger the signal processing ability, which however increases computational complexity and model size and results in high power consumption. In the distributed DeepSC system, the trained DeepSC model deployed at local IoT devices is frequently updated to perform intelligent tasks better. The IoT application limits the bandwidth and cost of distributing the DeepSC model. Furthermore, to extend the IoT network lifetime, especially the battery lifetime, most local devices

are with finite storage and computation capability, which limits the size of DeepSC. Therefore, compressing DeepSC not only reduces the latency of model transmission between the cloud/edge platform and local devices but also makes it possible to run the DL model on local devices.

III. PROPOSED LITE DISTRIBUTED SEMANTIC COMMUNICATION SYSTEM

To address the identified challenges in Section II, we propose a lite distributed semantic communication system, named L-DeepSC. We analyze the effects of CSI in the model training under fading channels and design a CSI-aided training process to overcome the fading effects, which successfully deals with *Question 1*. Besides, the weight pruning and quantization are investigated to address *Question 2*. Finally, our finite-points constellation design solves *Question 3*, effectively.

A. Deep De-noise Network based CSI Refinement and Cancellation

The most common method to reduce the effects of fading channels in wireless communication is to use the known channel properties of a communication link, CSI. Similarly, CSI can also reduce the channel impacts in training L-DeepSC. Next, we will first analyze the role of CSI in L-DeepSC training.

In order to simplify the analysis, we assume the transmitter and the receiver are with one-layer dense with sigmoid activation, where transmitter has an additional untrainable embedding layer, and receiver also has an untrainable de-embedding layer. The IoT devices are with the trained transmitter model and the cloud/edge platform works as the receiver, as shown in the system model Fig. 2. The IoT devices and cloud/edge platform are equipped with the same number of antennas. After the embedding layer, the source message, \mathbf{s} , is embedded into, \mathbf{S} . Then, encode \mathbf{S} into

$$\mathbf{X} = \sigma(\mathbf{W}_T \mathbf{S} + \mathbf{b}_T), \quad (8)$$

where \mathbf{X}^2 is the semantic features transmitted from the IoT devices to the cloud/edge platform. \mathbf{W}_T and \mathbf{b}_T are the trainable parameters to extract the features from source message \mathbf{s} , and $\sigma(\cdot)$ is the sigmoid activation function.

The received symbol at the cloud/edge platform is affected by channel \mathbf{H} and AWGN as in (2). From the received symbol, the cloud/edge platform recovers the embedding matrix by

$$\hat{\mathbf{S}} = \sigma(\mathbf{W}_R \mathbf{Y} + \mathbf{b}_R), \quad (9)$$

where the estimated source message, $\hat{\mathbf{s}}$, can be obtained after de-embedding layer. \mathbf{W}_R and \mathbf{b}_R can learn to recover \mathbf{s} . The L-DeepSC can be optimized by the loss function in (4). The fading channels not only contaminates the gradients in the back-propagation, but also restricts the representation power in the forward-propagation.

²Here, we have avoided discussion of complex signal. If the complex signal is $\tilde{\mathbf{X}}$, then $\tilde{\mathbf{X}} = [\Re(\mathbf{X}), \Im(\mathbf{X})]$.

Back-propagation: It updates parameter \mathbf{W}_T by its gradient

$$\frac{\partial \mathcal{L}_{\text{CE}}(\hat{\mathbf{s}}, \mathbf{s})}{\partial \mathbf{W}_T} = (\mathbf{F}_R \mathbf{W}_R \mathbf{H} \mathbf{F}_T)^T \nabla_{\hat{\mathbf{s}}} \mathcal{L}_{\text{CE}}(\hat{\mathbf{s}}, \mathbf{s}) \mathbf{s}^T, \quad (10)$$

where $\mathbf{F}_R \sim \text{diag}(\sigma'(\mathbf{W}_R \mathbf{y} + \mathbf{b}_R))$ and $\mathbf{F}_T \sim \text{diag}(\sigma'(\mathbf{W}_T \mathbf{s} + \mathbf{b}_T))$. In (10), the \mathbf{H} is untrainable and random, therefore it will cause perturbation for the weight updating, i.e., the weight updating with higher variance. If the transmitter consists of very deep neural networks, the perturbation will affect the back-propagation of the whole transmitter network, where the perturbation will propagate to the whole transmitter network by the chain rule.

Forward-propagation: With the received signal \mathbf{W}_R , the source messages can be recovered by

$$\begin{aligned} \hat{\mathbf{S}} &= \sigma(\mathbf{W}_R \mathbf{Y} + \mathbf{b}_R) \\ &= \sigma(\mathbf{W}_R \mathbf{H} \mathbf{X} + \mathbf{W}_R \mathbf{N} + \mathbf{b}_R). \end{aligned} \quad (11)$$

In (11), \mathbf{W}_R has to learn how to deal with the channel effects and decode at the same time, which increases training burden and reduces network expression capability. Meanwhile, the errors caused by channel effects also propagate to the subsequent layers for the L-DeepSC receiver with multiple layers.

The impacts of channel can be mitigated by exploiting CSI at the cloud/edge. If channel \mathbf{H} is known, then the received symbol can be processed by

$$\tilde{\mathbf{Y}} = (\mathbf{H}^H \mathbf{H})^{-1} \mathbf{H}^H \mathbf{Y} = \mathbf{X} + \tilde{\mathbf{N}}, \quad (12)$$

where $\tilde{\mathbf{N}} = (\mathbf{H}^H \mathbf{H})^{-1} \mathbf{H}^H \mathbf{N}$. In (12), the channel effect is transferred from multiplicative noise to additive noise, $\tilde{\mathbf{N}}$, which provides the possibility of stable back-propagation as well as the stronger capability of network representation. With (12), back-propagation and forward-propagation can be performed by setting $\mathbf{H} = \mathbf{I}$ in (10) and (11), respectively. Therefore, the channel effects can be completely removed.

The above discussion shows the importance of CSI in model training. However, CSI can be only estimated generally, i.e., least-squared (LS), linear minimum mean-squared error (LMMSE), or minimum mean-squared error (MMSE) estimators. Due to exploiting prior channel statistics, LMMSE and MMSE estimators usually perform better than the LS estimators. Thus, LMMSE and MMSE estimators are sensitive to the accuracy of channel statistic while the LS estimator requires no prior channel information. Meanwhile, DL techniques can also be used to improve the performance of channel estimation [36], [37].

For simplicity, we initially use the LS estimator. Then, we adopt the deep de-noise network to increase the resolution of the LS estimator as in [38] shown in Fig. 3. Particularly, the rough CSI estimated by the LS estimator with few pilots first denoted by

$$\mathbf{H}_{\text{rough}} = \mathbf{Y}_p \mathbf{X}_p^H = \mathbf{H} + \mathbf{N} \mathbf{X}_p^H, \quad (13)$$

where $\mathbf{Y}_p = \mathbf{H} \mathbf{X}_p + \mathbf{N}$, \mathbf{Y}_p is the received pilot signal, \mathbf{X}_p is the transmitted pilot signals. Then, (13) can be represented as

$$\mathbf{H}_{\text{rough}} = \mathbf{H} + \hat{\mathbf{N}}, \quad (14)$$

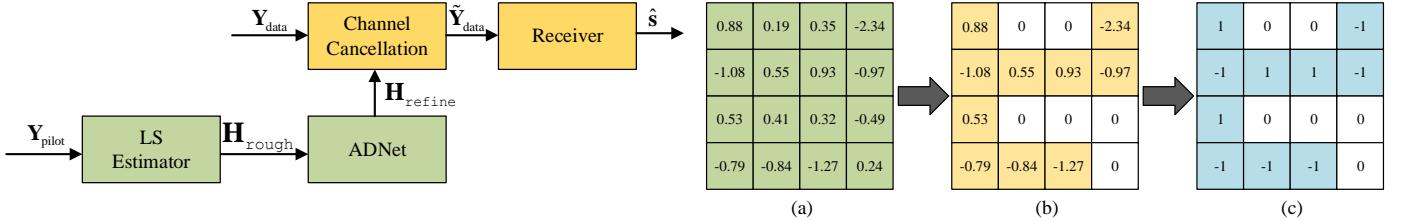


Fig. 3. The proposed CSI refinement and cancellation based on de-noise neural networks.

$$\text{where } \hat{\mathbf{N}} = \mathbf{N} \mathbf{X}_p^H.$$

From (14), $\mathbf{H}_{\text{rough}}$ consists of exact \mathbf{H} and the noise, $\hat{\mathbf{N}}$. De-noise neural networks are used to recover \mathbf{H} more accurately from $\mathbf{H}_{\text{rough}}$ by considering \mathbf{H} and $\mathbf{H}_{\text{rough}}$ as the original picture and noisy picture, respectively. Here, we exploit attention-guided denoising convolutional neural network (ADNet) [39] to refine CSI. ADNet includes four blocks, a sparse block, a feature enhancement block, an attention block, and a reconstruction block. After the input image, the sparse block is used to extract useful features from the given noisy image. Attention block can extract the noise information hidden in the complex background and is integrated into the feature enhancement block to reduce the complexity. Finally, the de-noised image is reconstructed by the reconstruction block.

The refined CSI, $\mathbf{H}_{\text{refine}}$ denoted by

$$\mathbf{H}_{\text{refine}} = \text{ADNet}(\mathbf{H}_{\text{rough}}). \quad (15)$$

In (15), the $\text{ADNet}(\cdot)$ is trained the the loss function, $\mathcal{L}(\mathbf{H}_{\text{refine}}, \mathbf{H}) = \frac{1}{2} \|\mathbf{H}_{\text{refine}} - \mathbf{H}\|_F^2$. Since the performance of the LS estimator is similar to that of LMMSE and MMSE estimators in the high SNR region, we pay more attention to the low SNR region when training ADNet. With proper training, ADNet can mitigate the impacts from noise but without any prior channel information, especially in the low SNR region. Such a design provides a good solution for *Question 1*.

B. Model Compression

Through applying CSI into model training, the cloud/edge platform can extract the semantic features from L-DeepSC. However, the size and complexity of the trained L-DeepSC model are still very large, which causes high latency for the cloud/edge platform to broadcast updated L-DeepSC. Note that both weights pruning and quantization can reduce the model size and complexity, therefore, we compress the DeepSC model by a joint pruning-quantization scheme to make it affordable for IoT devices. As shown in Fig. 4, the original weights are first pruned at a high-precision level by identifying and removing the unnecessary weights, which makes the network sparse. Quantization is then used to convert the trained L-DeepSC model into a low-precision level. The proposed network sparsification and quantization can address *Question 3* and are introduced in detail in the following.

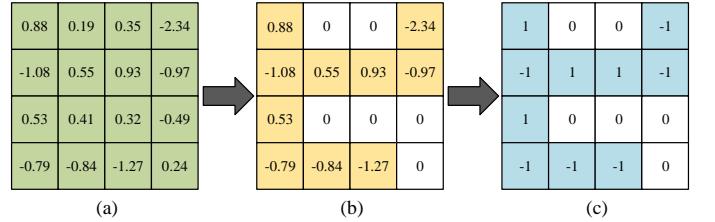


Fig. 4. Flowchart of the proposed joint pruning-quantization, (a) the original weights matrix; (b) the weights after pruning, where the example pruning function is $x = 0$ for $x < 0.5$; (c) the weights after quantization, where the example quantization function is $x = \text{sign}(x)$.

Algorithm 1 Network Sparsification.

Input: The pre-trained weights \mathbf{W} , the sparse ratio γ .

Output: The pruned weights $\mathbf{W}_{\text{pruned}}$.

- 1: Count the the total number of connections, M .
 - 2: Sort the whole connections from small to large, \mathbf{s} .
 - 3: Obtain the threshold by (17) with M and γ , w_{thre} .
 - 4: **for** $n = 1$ to N **do**
 - 5: Prune the connections by (16), $\mathbf{W}_{\text{pruned}}^{(n)}$.
 - 6: **end for**
 - 7: Fine-tune the pruned model by loss function (4).
-

1) Network Sparsification: A proper criterion to disable neural connections is important. Obviously, the connections with small weight values can be pruned. Therefore, the pruning issue here turns into setting a proper pruning threshold.

As shown in Fig. 2(b), the DeepSC consists with neural networks, $\alpha, \beta, \chi, \delta$, where each includes multiple layers. As the DeepSC mainly consists of dense layers, we choose unstructured pruning method in this paper, where the computation workload of sparse model can be reduced by the sparsity algorithm and FPGA design [40], [41], i.e., sparse matrix-vector multiplication. Assume there are total N layers in the pre-trained DeepSC model with $\mathbf{W}_{i,j}^{(n)}$ being the weight of connection between the i_{th} neuron of the $(n+1)_{th}$ layer and j_{th} neuron of n_{th} layer. With a pruning threshold w_{thre} , the model weights can be pruned by

$$\mathbf{W}_{i,j}^{(n)} = \begin{cases} \mathbf{W}_{i,j}^{(n)}, & \text{if } |\mathbf{W}_{i,j}^{(n)}| > w_{\text{thre}}, \\ 0, & \text{otherwise,} \end{cases} \quad (16)$$

We determine the pruning threshold by

$$w_{\text{thre}} = \mathbf{s}_M \times \gamma, \quad (17)$$

where $\mathbf{s} = \text{sort}([\mathbf{W}^{(1)}, \mathbf{W}^{(2)}, \dots, \mathbf{W}^{(N)}])$, is the sorted weights value from least important one to the most important one, M is the total number of connections, and γ , the sparsity ratio between 0 and 1, indicates the proportion of zero values in weights. The weight pruning can be divided into two steps, weight pruning to disable some neuron connections and fine-tine to recover the accuracy, as shown in Algorithm 1.

2) Network Quantization: The quantization includes weight quantization and activation quantization. The weights, $\mathbf{W}_{i,j}^{(n)}$, from a trained model, can be converted from 32-bit float point

Algorithm 2 Network Quantization.

Input: The pre-trained weights \mathbf{W} , the quantization level m , the correlation coefficient c , and the calibration data \mathcal{K} .
Output: The pre-trained weights $\mathbf{W}_{\text{quantized}}$ and the range of activation x_{\min} and x_{\max} .

- 1: **Phase 1:** Weights Quantization.
- 2: **for** $n = 1$ to N **do**
- 3: Compute the range of weights, $\max(\mathbf{W}^{(n)})$ and $\min(\mathbf{W}^{(n)})$.
- 4: Quantize the weights by (18), $\tilde{\mathbf{W}}^{(n)}$.
- 5: **end for**
- 6: **Phase 2:** Activations Quantization.
- 7: **for** $t = 1$ to \mathcal{K} **do**
- 8: **for** $n = 1$ to N **do**
- 9: Update the dynamic range of activation by (20) and (21), $x_{\min}^{(n)}(t)$ and $x_{\max}^{(n)}(t)$.
- 10: **end for**
- 11: **end for**
- 12: Quantize the activations by (22).
- 13: Fine-tune the quantized model by STE and loss function (4).

to m -bits integer through applying the quantization function by

$$\tilde{\mathbf{W}}_{i,j}^{(n)} = \text{round}\left(q_w\left(\mathbf{W}_{i,j}^{(n)} - \min(\mathbf{W}^{(n)})\right)\right), \quad (18)$$

where q_w is the scale-factor to map the dynamic range of float points to an m -bits integer, which is given by

$$q_w = \frac{2^m - 1}{\max(\mathbf{W}^{(n)}) - \min(\mathbf{W}^{(n)})}. \quad (19)$$

For activation quantization, the results of matrix multiplication are stored in accumulators. Due to the limited dynamic range of integer formats, it is possible that the accumulator overflows quickly if the bit-width for the weights and activation is the same. Therefore, accumulators are usually implemented with higher bit-widths, for example, INT32 += INT8 × INT8. Besides, the range of activations is dynamic and dependent on the input data. Therefore, the output of activations has to re-quantize into m -bits integer for the subsequent calculation. Unlike weights that are constant, the output of activations usually includes elements that are statistical outliers, which expand the actual dynamic range. For example, even if 99% of the data is distributed between -100 and 100, an outlier, 10,000, will extend the dynamic range from -100 to 10,000, which significantly reduces the mapping resolution. In order to reduce the influence from the outliers, an exponential moving average (EMA) is used by

$$x_{\min}^{(n)}(t+1) = (1 - c)x_{\min}^{(n)}(t) + c \min(\mathbf{X}^{(n)}(t)), \quad (20)$$

and

$$x_{\max}^{(n)}(t+1) = (1 - c)x_{\max}^{(n)}(t) + c \max(\mathbf{X}^{(n)}(t)), \quad (21)$$

where $x_{\min}^{(n)}(t+1)$ and $x_{\max}^{(n)}(t+1)$ are used for the range of activation quantization, and $x_{\min}^{(n)}(1) = \min(\mathbf{X}^{(n)}(1))$,

$x_{\max}^{(n)}(1) = \max(\mathbf{X}^{(n)}(1))$, $\mathbf{X}^{(n)}(t)$ is the output of activations at n_{th} layer with t_{th} batch data, $c \in [0, 1]$ represents the correlation between the current $x_{\min}^{(n)}/x_{\max}^{(n)}$ with its past value. The effects from outliers can be mitigated by the past normal values. After $t+1$ epochs, the $x_{\min}^{(n)}$ and $x_{\max}^{(n)}$ are fixed based on $x_{\min}^{(n)}(t+1)$ and $x_{\max}^{(n)}(t+1)$. Then, the output of the activations can be quantized by

$$\tilde{\mathbf{X}}^{(n)} = \text{clamp}\left(\text{round}\left(q_x\left(\mathbf{X}^{(n)} - x_{\min}^{(n)}\right)\right); -M, M\right), \quad (22)$$

where $q_x = (2^m - 1)/(x_{\max}^{(n)} - x_{\min}^{(n)})$ is the scale-factor and $\text{clamp}(\cdot)$ is used to eliminate the quantized outliers, which is given by

$$\text{clamp}\left(\mathbf{X}^{(n)}; -T, T\right) = \min\left(\max\left(\mathbf{X}^{(n)}, -T\right), T\right), \quad (23)$$

where $T = 2^m - 1$, which is the border of the m -bits integer format.

As shown in Algorithm 2, the network quantization includes two phases: i) weight quantization; ii) activations quantization. In phase 1, the weights of each layer can be quantized by (18) directly. In phase 2, the calibration process is applied by running a few calibration batches in order to get the activations statistics. In each batch, $x_{\min}^{(n)}(t)$ and $x_{\max}^{(n)}(t)$ will be updated based on the activations statistics from the previous batches. These quantization processes might lead to slight accuracy degradation. The quantization-aware training (QAT) is required to re-train for minimizing the loss of accuracy. Since the rounding operation is not derivable, a straight-through estimator (STE) is used to estimate the gradient of quantized weights in the back-propagation [42].

C. Constellation Design with Fewer Quantization Bits

The cloud/edge platform can further reduce the size of L-DeepSC with model compression after the model is trained, which not only reduces the latency significantly for broadcasting the updated DeepSC to IoT devices, but also changes DeepSC to L-DeepSC with low complexity. However, high-resolution waveform poses high requirements cost-sensitive IoT devices. In other words, the cost-sensitive IoT devices are usually capacity-limited and cannot afford a large number of constellation points which are with phase and amplitude close to each other.

Different from bits, the source message, s , is more complicated and the learned constellation will not be limited to a few points, which brings additional burden on hardware. Besides, the DL models generally run in FP32, which also expands the range of constellation. Thus, we aim to reduce the size of learned constellation without degrading performance, where the output of \mathbf{X} is the learned constellation while \mathbf{X} is also the output of activation of last layer at the local IoT devices. Inspired from the network quantization, we convert the learned high-resolution constellation into low-resolution one with few points. Thus, we use two-stage quantization to narrow the range of constellations, which is represented by

$$\mathbf{X}_{\text{dequantize}} = \frac{\mathbf{X}_{\text{quantize}}}{q_x} + x_{\min}, \quad (24)$$

where $\mathbf{X}_{\text{quantize}}$ is the quantized \mathbf{X} from (22), q_x is the scale-factor and x_{\min} is the obtained by (20) and $\mathbf{X}_{\text{dequantize}}$ is the dequantized \mathbf{X} .

First, we quantize the \mathbf{X} into m -bits integer so that the range of \mathbf{X} is narrowed to the size of 2^m . For example, when $m = 8$, the size of the constellation is reduced to 256. Then, $\mathbf{X}_{\text{quantize}}$ is dequantize to restore \mathbf{X} . Such an $\mathbf{X}_{\text{dequantize}}$ has a similar distribution as \mathbf{X} but is with fewer constellation points, which is helpful to lower the hardware cost at transmitter and preserves the performance as much as possible and therefore provides the solution for *Question 2*.

In summary, by exploiting the solutions for the aforementioned *Questions*, we develop a lite distributed semantic communication system, named L-DeepSC, which could reduce the latency for model exchange under limited bandwidth, run the models at IoT devices with low power consumption, and deal with the distortion from fading channels when uploading semantic features. As a result, the proposed L-DeepSC becomes a good candidate for the IoT networks.

IV. NUMERICAL RESULTS

In this section, we compare the proposed L-DeepSC with traditional methods under different fading channels, including Rayleigh and Rician fading channels. The weights pruning and quantization are also verified under fading channels. For the Rayleigh fading channel, the channel coefficient follows $\mathcal{CN}(0, 1)$; for the Rician fading channel, it follows $\mathcal{CN}(\mu, \sigma^2)$ with $\mu = \sqrt{k/(k+1)}$ and $\sigma = \sqrt{1/(k+1)}$, where k is the Rician coefficient and we use $k = 2$ in our simulation.

The transmitter of L-DeepSC is the same as that of DeepSC in [22]. The parameters for the decoding network at the receiver are shown in Table I for the fading channels, where the sum of the outputs of Dense 3 and Dense 5 is the input of the LayerNorm layer. The Transformer encoder and decoder are semantic encoder and decoder [22], respectively, which enables the systems to understand text and extract semantic information. We also prune the whole network since we consider the communications between cloud/edge platform and each IoT devices as well as the communications between IoT devices.

TABLE I
THE SETTING OF L-DEEPC TRANSCEIVER.

	Layer Name	Units	Activation
Transmitter	Embedding layer	128	None
	4×Transformer Encoder	128 (8 heads)	None
	Dense 1	256	Relu
	Dense 2	16	None
Receiver	Dense 3	128	Relu
	Dense 4	512	Relu
	Dense 5	128	None
	LayerNorm	None	None
	4×Transformer Decoder	128 (8 heads)	None
	Prediction Layer	Dictionary Size	Softmax

The output features are with 8 symbols per word. We initialize the learnable embedding matrix from $\mathcal{N}(0, 1)$ with shape (vocab size, embedding-dim). The embedding dim is set to 128 in our program and the vocab size depends on the training dataset. The batch size is 64, learning rate is

$128^{-0.5} \min(\text{step}^{-0.5}, \text{step} \times 4000^{-1.5})$, where step is the counting number in the back-propagation. This corresponds to increasing the learning rate linearly for the first 4000 training steps and decreasing it thereafter proportionally to the inverse square root of the step number. We also adopt the L_2 regularization and the Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\varepsilon = 10^{-8}$.

The adopted dataset is the proceedings of the European Parliament [43], which consists of around 2.0 million sentences and 53 million words. The dataset is pre-processed into lengths of sentences with 4 to 30 words and is split into training data and testing data with 0.1 ratio. The benchmark approach is based on separate source coding and channel coding technologies, which adopt variable-length coding (Huffman coding) for source coding, where we build the Huffman codes by counting the frequency of letters and punctuation so that the look-up table is not large. Turbo coding and Reed-Solomon (RS) coding [44] for channel coding, where turbo decoding method is log-MAP algorithm with 5 iterations, and quadrature amplitude modulation (QAM). The bilingual evaluation underway (BLEU) score is used to measure the performance [45].

A. Constellation Design

Fig. 5 compares the full-resolution constellation and the 4-bits constellation. The full-resolution constellation points in Fig. 5(a) contain more information due to the higher resolution, but require complicated hardware, which is almost impossible to design. Through mapping the full-resolution constellation into a finite space, the 4-bits constellation points in Fig. 5(b) become simplified, which makes it possible to implement in the existing RF system. Note that the 4-bits constellation keeps a similar distribution with the full-resolution constellation. For example, there exist certain blank regions at the edge of the constellation in Fig. 5(a), while the 4-bits constellation shows a similar trend in Fig. 5(b). Such similar distribution prevents sharp performance degradation when the resolution of constellation decreases significantly.

Fig. 6 shows the BLEU scores versus SNR for different constellation sizes under AWGN, including 4-bits constellation, 8-bits constellation, and full-resolution constellation. All of them could achieve very similar performance when $\text{SNR} > 9 \text{ dB}$, which demonstrates the constellation design is effective and cause no significant performance degradation. Full resolution and 8-bits constellations perform slightly better than 4-bits constellation when SNR is low. This is because some weights information used for denoising is lost when the resolution of the constellation is small.

B. Performance over Fading Channels

Fig. 7 compares the channel estimation MSEs of LS, MMSE, and ADNet-aided LS estimator versus SNR under the Rayleigh fading channels. Note that MMSE equals to LMMSE for the AWGN channels. The MMSE and LS estimators have similar accuracy in the high SNR region, thus the range of training SNRs for the ADNet is set from 0 dB to 10 dB to improve the performance of the LS estimator in the low SNR

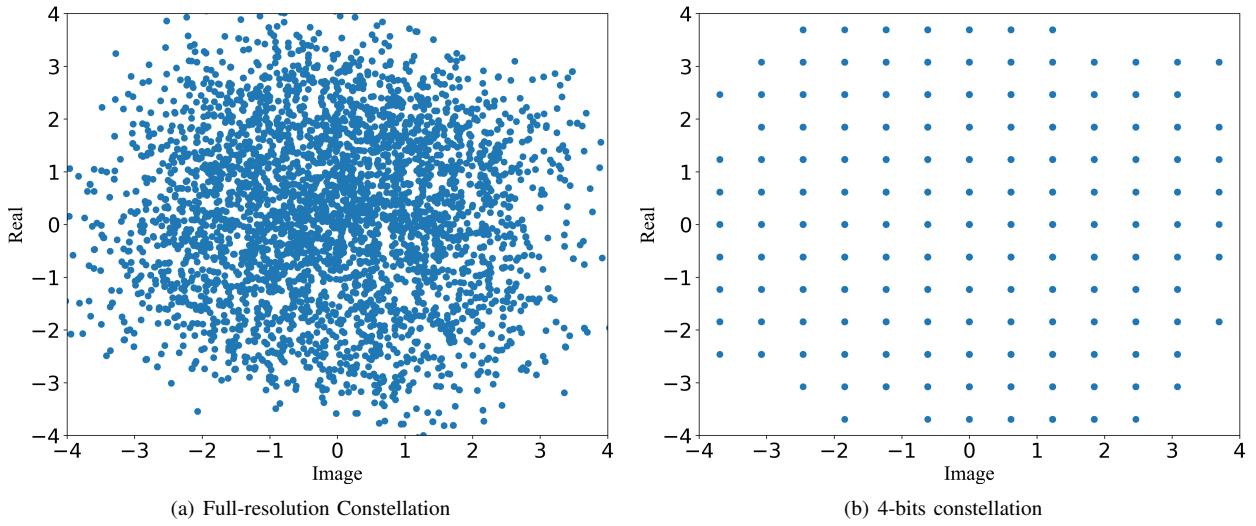


Fig. 5. The comparison between the full-resolution constellation and 4-bits constellation.

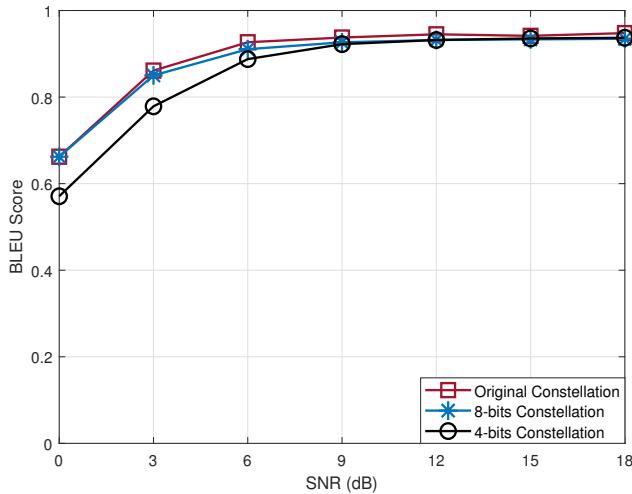


Fig. 6. The BLEU scores of different constellation sizes versus SNR under AWGN.

region. As a result, the MSE of ADNet based LS estimator is significantly lower than that of LS and MMSE estimators when SNR is low. With increasing SNR, the MSE of ADNet based LS estimator approaches to that of the LS and MMSE estimators. Therefore, the ADNet based LS estimator can be substituted by the LS estimator to reduce the complexity in the high SNR region.

Fig. 8 and Fig. 9 illustrate the relationship between BLEU score and SNR with the 4-bits constellation over the Rician and the Rayleigh fading channels, respectively, where DeepSC is trained with perfect CSI and the L-DeepSC is trained with perfect CSI, rough CSI by (14), refined CSI by (15) and without CSI, respectively. The traditional approaches are Huffman coding with (5,7) RS and with turbo coding (rate 1/2), both with 64-QAM. We observe that all DL-enabled approaches are more competitive under the fading channels. RS coding is better than turbo coding in terms of BLEU score. This is because RS coding is linear block coding with long

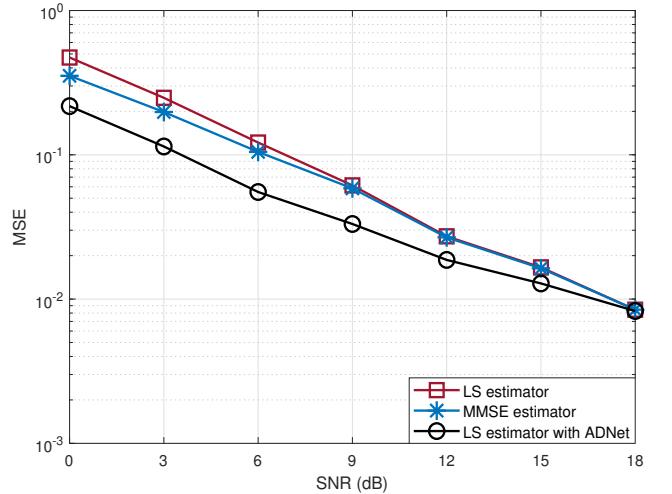


Fig. 7. The MSE for MMSE estimator, LS estimator, and the proposed ADNet based LS estimator.

block-length, which can correct long bit sequences, however, turbo coding is convolution coding with short block-length, where the coded bits only are related with previous m bits, i.e., $m = 3$, so that the adjacent words result in higher error rate. The performance of L-DeepSC is very close to that of DeepSC in terms of BLEU score, but requires much less bandwidth for communications. The system trained without CSI performs worse than those trained with CSI, especially under the Rayleigh fading channels, which also confirms the analysis of (10) and (11). Without CSI, the performance difference between the Rayleigh channels and the Rician channels is caused by the line-of-sight (LOS), which can help the systems recognize the semantic information during training. Besides, with the aid of CSI, the effects of the fading channels are mitigated significantly, as we have analyzed before. When SNR is low, the system with perfect CSI or refined CSI outperforms that with rough CSI. As SNR increases, all these systems, L-DeepSC with perfect CSI, refined CSI, and rough

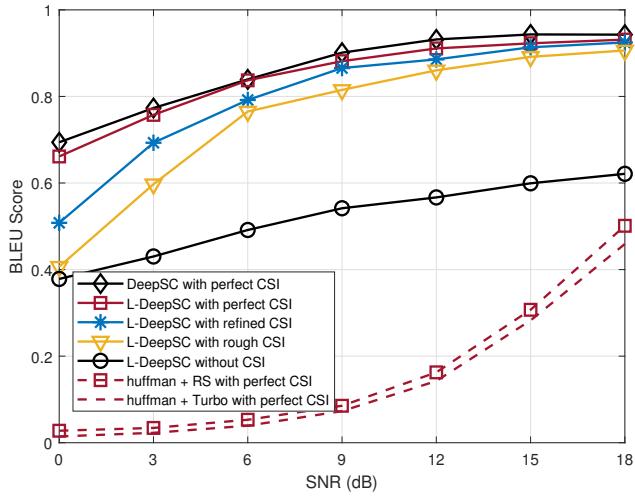


Fig. 8. The BLEU scores versus SNR under Ricain fading channels, with perfect CSI, rough CSI, refined CSI, and no CSI.

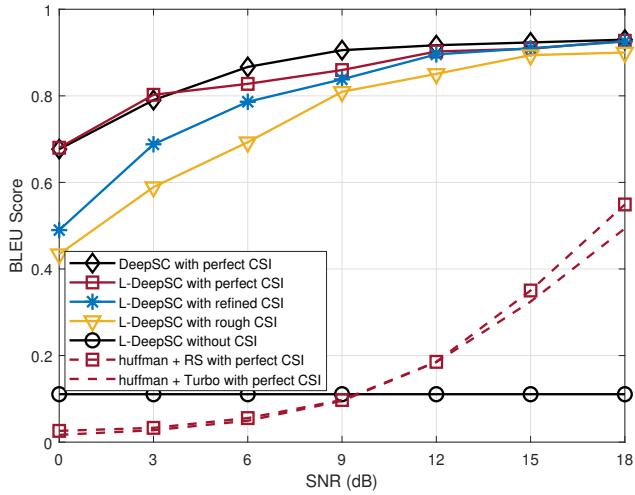


Fig. 9. The BLEU scores versus SNR under Rayleigh fading channels, with perfect CSI, rough CSI, refined CSI, and no CSI.

CSI, converge to similar performance gradually.

C. Model Compression

In this experiment, we investigate the performance of network slimmer, including network sparsification, network quantization, and the combination of both. The pre-trained model used for pruning and quantization is trained with 4-bits constellation under the Rician fading channels.

Fig. 10 shows the influences of network sparsity ratio, γ , on the BLEU scores with different SNRs under the Rician fading channels, where the system is pruned directly when γ increases from 0 to 0.9 and is pruned with fine-tuning when γ increases to 0.99 continually. The proposed L-DeepSC achieves almost the same BLEU scores when the γ increases from 0 to 0.9, which shows that there exists a mass of weights redundancy in the trained DeepSC model. When the γ increases to 0.99, the BLEU scores still drop slightly due to the processing of fine-tuning, where the performance loss at 0 dB and 6 dB is larger than that at 12 dB and 18 dB. Thus,

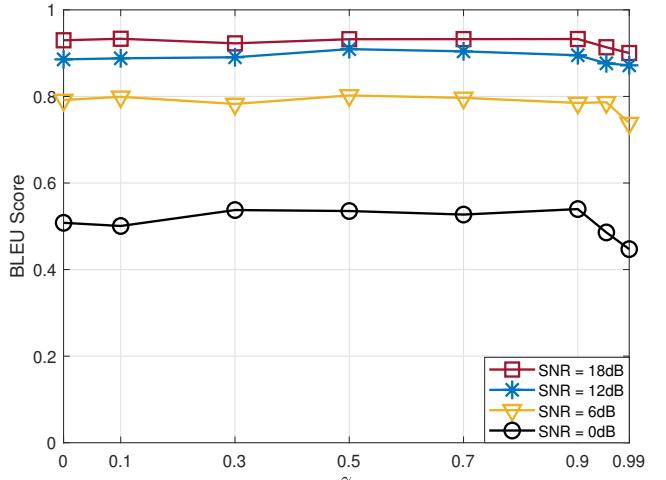


Fig. 10. The BLEU scores of different SNRs versus sparsity ratio, γ , under Rician fading channel with the refined CSI.

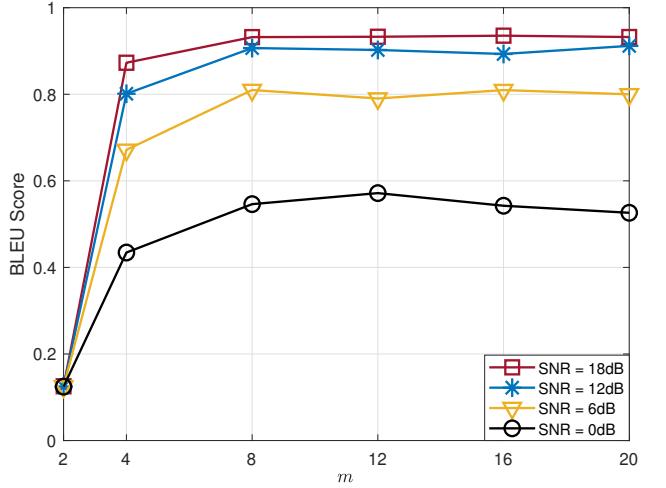


Fig. 11. The BLEU scores of different SNRs versus quantization level, m , under Rician fading channels with the refined CSI.

for the high SNR cases, the model can be pruned directly with only slight performance degradation. For the low SNR region, it is possible to prune 99% weights without significant performance degradation when the system is sensitive to power consumption.

Fig. 11 demonstrates the relationship between the BLEU score and the quantization bit number, m , under the Rician fading channels, where m is defined in (19), and the system is quantized with QAT when the m is smaller than 2. The performance with $m = 8$ to $m = 20$ is similar, which indicates that the effectiveness of low-resolution neural networks. If the system is more sensitive to power consumption and can tolerate to certain performance degradation, the resolution of the neural networks can be further reduced to 4-bits level. However, the BLEU score decreases dramatically from $m = 4$ to $m = 2$ over the whole SRN range since most of the key information is removed in the low-resolution neural network.

Table II compares the BLEU scores and compression ratios under different combinations of weights pruning and weights

TABLE II
THE BLEU SCORE AND COMPRESSION RATIO, ψ , COMPARISONS VERSUS DIFFERENT SPARSITY RATIO, γ , AND QUANTIZATION LEVEL, m , IN SNR = 12dB.

Pruned Model	BLEU score with $m = 4$	ψ	BLEU score with $m = 8$	ψ	BLEU score with $m = 12$	ψ	BLEU score with $m = 16$	ψ	BLEU score with $m = 32$	ψ
$\gamma = 0$	0.811194	8	0.906763	4	0.902354	2.667	0.903089	2	0.895602	1
$\gamma = 0.3$	0.838967	11.429	0.892745	5.714	0.908537	3.81	0.910184	2.857	0.89851	1.429
$\gamma = 0.6$	0.835863	20.0	0.897143	10.0	0.90815	6.667	0.900468	5.0	0.9093	2.5
$\gamma = 0.9$	0.810322	80.0	0.895306	40.0	0.898784	26.667	0.910554	20.0	0.89515	10
$\gamma = 0.95$	0.779685	160.0	0.875814	80.0	0.873426	53.333	0.877221	40.0	0.87653	20

TABLE III
THE COMPARISON BETWEEN L-DEEPC AND DEEPC TRANSCEIVER IN PARAMETERS, SIZE, RUNTIME, AND BLEU SCORE.

	Parameters	Size	Runtime	BLEU score
$\gamma = 0,$ $m = 32$	3,333,120	12.3 MB	20ms	0.895602
$\gamma = 0.6,$ $m = 8$	1,333,247	1.28 MB	18ms	0.897143

quantization with SNR = 12 dB, where the compression ratio is computed by

$$\psi = \frac{M \times 32}{M_{\text{pruned}} \times m}, \quad (25)$$

where M is the number of weights before pruning and M_{pruned} is the number of weights remaining after pruning, 32 is the number of required bits for FP32 and m is the number of the required bits after quantization. The performance decreases when γ increases or m decreases, which are consistent with Fig. 10 and Fig. 11. From the table, different compression ratios could lead to similar performance. For example, the BLEU score with $\gamma = 30\%$ and $m = 8$ is similar to that with $\gamma = 90\%$ and $m = 12$, but the compression ratio is about five times different, i.e., 5.714 and 26.667. By properly choosing a suitable sparsity ratio and a quantization level, the same performance can be achieved but with a high compression ratio.

Table III compares the DeepSC and L-DeepSC with 60% weights sparsity and 8-bit quantization when SNR is 12 dB, where we mainly consider the transmission of the weights. The simulation is performed in CPU by the computer with Intel Core i7-9700CPU@3.00GHz. After network slimmer, the model size is reduced from 12.3 MB to 1.28 MB while achieving a similar BLEU score, which means the bandwidth resource can be saved significantly without degrading the performance. Besides, the runtime slightly decreases from 20ms to 18ms since the unstructured pruning method is employed, and there exists the communication time between flash memory and some operation that can not be optimized. If the model size is bigger, the L-DeepSC could save more runtime.

V. CONCLUSION

In this paper, we proposed a lite distributed semantic communication system, named L-DeepSC, for the Internet of Things (IoT) networks, where the participating devices are usually with limited power and computing capabilities.

Specially, the receiver and feature extractor were designed jointly for text transmission. Firstly, we analyzed the effectiveness of CSI in forward-propagation and back-propagation during system training over the fading channels. The analytical results reveal that the fading channels contaminate the weights update and restrict model representation capability. Thus, a refined LS estimator with fewer pilot overheads was developed to eliminate the effects of fading channels. Besides, we map the full-resolution original constellation into finite bits constellation to lower the cost of IoT devices, which was verified by simulation results. Finally, due to the limited narrow bandwidth and computational capability in IoT networks, two model compression approaches have been proposed: 1) the network sparsification to prune the unnecessary weights, and 2) network quantization to reduce the weights resolution. The simulation results validated that the proposed L-DeepSC outperforms the traditional methods, especially in the low SNR regime, and has provided insights into the balance among compression ratio, sparsity ratio, and quantization level. Therefore, our proposed L-DeepSC is a promising candidate for intelligent IoT networks, especially in the low SNR regime.

REFERENCES

- [1] L. Atzori, A. Iera, and G. Morabito, "The Internet of Thingss: a survey," *Computer Netw.*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.
- [2] T. Qiu, N. Chen, K. Li, M. Atiquzzaman, and W. Zhao, "How can heterogeneous Internet of Things build our future: A survey," *IEEE Commun. Surv. Tutorials*, vol. 20, no. 3, pp. 2011–2027, Feb. 2018.
- [3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.
- [4] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, "Deep learning for IoT big data and streaming analytics: A survey," *IEEE Commun. Surv. Tutorials*, vol. 20, no. 4, pp. 2923–2960, Jun. 2018.
- [5] H. Li, K. Ota, and M. Dong, "Learning iot in edge: Deep learning for the Internet of Things with edge computing," *IEEE Network*, vol. 32, no. 1, pp. 96–101, Jan. 2018.
- [6] R. Carnap, Y. Bar-Hillel *et al.*, *An Outline of A Theory of Semantic Information*. RLE Technical Reports 247, Research Laboratory of Electronics, Massachusetts Institute of Technology., Cambridge MA, Oct. 1952.
- [7] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. Cambridge University Press, 2005.
- [8] I. Guyon, S. Gunn, M. Nikravesh, and L. A. Zadeh, *Feature Extraction: Foundations and Applications*. Springer, 2008, vol. 207.
- [9] R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer Science & Business Media, 2010.
- [10] N. Indurkha and F. J. Damerau, *Handbook of Natural Language Processing*. CRC Press, 2010, vol. 2.
- [11] C. E. Shannon and W. Weaver, *The Mathematical Theory of Communication*. The University of Illinois Press, 1949.
- [12] D. Tse and P. Viswanath, *Fundamentals Wireless Communication*. Cambridge University Press, 2005.

- [13] Z. Qin, H. Ye, G. Y. Li, and B.-H. F. Juang, "Deep learning in physical layer communications," *IEEE Wireless Commun.*, vol. 26, no. 2, pp. 93–99, Apr. 2019.
- [14] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Trans. Cogn. Comm. & Networking*, vol. 3, no. 4, pp. 563–575, Oct. 2017.
- [15] S. Dörner, S. Cammerer, J. Hoydis, and S. ten Brink, "Deep learning based communication over the air," *IEEE J. Sel. Topics Signal Processing*, vol. 12, no. 1, pp. 132–143, Dec. 2018.
- [16] F. A. Aoudia and J. Hoydis, "Model-free training of end-to-end communication systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 11, pp. 2503–2516, Aug. 2019.
- [17] H. Ye, L. Liang, G. Y. Li, and B.-H. Juang, "Deep learning-based end-to-end wireless communication systems with conditional gans as unknown channels," *IEEE Trans. Wireless Commun.*, vol. 19, no. 5, pp. 3133–3143, Feb. 2020.
- [18] E. Bourtsoulatze, D. B. Kurka, and D. Gündüz, "Deep joint source-channel coding for wireless image transmission," *IEEE Trans. Cogn. Commun. Netw.*, vol. 5, no. 3, pp. 567–579, May 2019.
- [19] C. Lee, J. Lin, P. Chen, and Y. Chang, "Deep learning-constructed joint transmission-recognition for internet of things," *IEEE Access*, vol. 7, pp. 76 547–76 561, Jun. 2019.
- [20] M. Jankowski, D. Gündüz, and K. Mikolajczyk, "Joint device-edge inference over wireless links with pruning," in *Proc. IEEE Int'l Workshop Signal Process. Advances Wireless Commun. (SPAWC)*, Atlanta, GA, USA, Aug. 2020, pp. 1–5.
- [21] N. Farsad, M. Rao, and A. Goldsmith, "Deep learning for joint source-channel coding of text," in *Proc. IEEE Int'l. Conf. Acoustics Speech Signal Process. (ICASSP)*, Calgary, AB, Canada, Apr. 2018, pp. 2326–2330.
- [22] H. Xie, Z. Qin, G. Y. Li, and B.-H. Juang, "Deep learning enabled semantic communication systems," *arXiv:2006.10685*, 2020. [Online]. Available: <https://arxiv.org/abs/2006.10685>
- [23] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, Montreal, Quebec, Canada, Dec. 2014, pp. 1269–1277.
- [24] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, Montreal, Quebec, Canada, Dec. 2015, pp. 1135–1143.
- [25] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *Proc. IEEE Int'l. Conf. on Comput. Vis. (ICCV)*, Venice, Italy, Oct. 2017, pp. 2755–2763.
- [26] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," in *Proc. IEEE Int'l. Conf. on Learning Representations (ICLR)*, Toulon, France, Apr. 2017.
- [27] R. Krishnamoorthi, "Quantizing deep convolutional networks for efficient inference: A whitepaper," *arXiv:1806.08342*, 2018. [Online]. Available: <https://arxiv.org/abs/1806.08342>
- [28] Y. Gong, L. Liu, M. Yang, and L. Bourdev, "Compressing deep convolutional networks using vector quantization," *arXiv:1412.6115*, 2014. [Online]. Available: <https://arxiv.org/abs/1412.6115>
- [29] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, "Incremental network quantization: Towards lossless cnns with low-precision weights," in *Proc. IEEE Int'l. Conf. on Learning Representations (ICLR)*, Toulon, France, Apr. 24–26, 2017.
- [30] F. Li, B. Zhang, and B. Liu, "Ternary weight networks," *arXiv:1605.04711*, 2016. [Online]. Available: <https://arxiv.org/abs/1605.04711>
- [31] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. G. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Salt Lake City, UT, USA, Jun. 2018, pp. 2704–2713.
- [32] J. Guo, J. Wang, C.-K. Wen, S. Jin, and G. Y. Li, "Compression and acceleration of neural networks for communications," *IEEE Wireless Commun.*, vol. 27, no. 4, pp. 110–117, July 2020.
- [33] D. Gil, A. Ferrández, H. Mora-Mora, and J. Peral, "Internet of Things: A review of surveys based on context aware intelligent services," *Sensors*, vol. 16, no. 7, p. 1069, Jul. 2016.
- [34] "IEEE standard for floating-point arithmetic," *IEEE Std 754-2008*, pp. 1–70, 2008.
- [35] B. Zhu, J. Wang, L. He, and J. Song, "Joint transceiver optimization for wireless communication phy using neural network," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1364–1373, Mar. 2019.
- [36] K. Thakkar, A. Goyal, and B. Bhattacharyya, "Deep learning and channel estimation," in *Proc. Int'l Conf. on Adv. Comput. and Commun. Systems (ICACCS)*, Coimbatore, India, Mar. 2020, pp. 745–751.
- [37] E. Balevi, A. Doshi, and J. G. Andrews, "Massive MIMO channel estimation with an untrained deep neural network," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 2079–2090, Jan. 2020.
- [38] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a Gaussian denoiser: Residual learning of deep cnn for image denoising," *IEEE Trans. Image Process.*, vol. 26, no. 7, pp. 3142–3155, Feb. 2017.
- [39] C. Tian, Y. Xu, Z. Li, W. Zuo, L. Fei, and H. Liu, "Attention-guided cnn for image denoising," *Neural Netw.*, vol. 124, pp. 117–129, Apr. 2020.
- [40] R. Dorrance, F. Ren, and D. Marković, "A scalable sparse matrix-vector multiplication kernel for energy-efficient sparse-blas on fpgas," in *Proc. ACM/SIGDA Int'l sym. Field-programmable gate arrays*, Feb. 2014, pp. 161–170.
- [41] L. Zhuo and V. K. Prasanna, "Sparse matrix-vector multiplication on fpgas," in *Proc. ACM/SIGDA Int'l sym. Field-programmable gate arrays*, Feb. 2005, pp. 63–74.
- [42] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv:1308.3432*, 2013. [Online]. Available: <http://arxiv.org/abs/1308.3432>
- [43] P. Koehn, "Europarl: A parallel corpus for statistical machine translation," in *MT Summit*, vol. 5, 2005, pp. 79–86.
- [44] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *J. Society Industrial Applied Math.*, vol. 8, no. 2, pp. 300–304, Jan. 1960.
- [45] K. Papineni, S. Roukos, T. Ward, and W. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proc. Annual Meeting Assoc. Comput. Linguistics (ACL)*, Philadelphia, PA, USA, Jul. 2002, pp. 311–318.