A Lite Distributed Semantic Communication System for Internet of Things

Huiqiang Xie and Zhijin Qin

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 a DL model 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. Meanwhile, we tailor the semantic constellation training processing to decrease the effects of fading channels on 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 by quantization for the current antenna design. 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 20x 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 these 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 from 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 due to decisions according to the extracted features [9], [10]. Corresponding to the three steps, the communication is categorized into three levels correspondingly [11], including transmission level, semantic level, and effectiveness level, as explained in Fig. 1. The traditional communication system works at the transmission level shown in Fig. 1(a), which aims to 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 the 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]. There have been some initial works related to deep semantic communications [14]–[17]. Bourtsoulatze et al. [14] 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...

Huiqiang Xie and Zhijin Qin are with the School of Electronic Engineering and Computer Science, Queen Mary University of London, London E1 4NS, UK (e-mail: h.xie@qmul.ac.uk, z.qin@qmul.ac.uk).
classification tasks into consideration, Lee et al. [18] developed an 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. [16] 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 apply it in dynamic environments, Xie et al. [17] 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 on reducing the number of parameters, network slimmer has attracted extensive attention to compress DL models without degrading performance since neural networks are trained usually with over-parameters [19]. 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. [20] proposed an iterative pruning approach, where the model is trained first, then pruned by a given threshold, and is finely tuned to recover performance in terms of image classification. This approach could reduce the connections without losing accuracy. Liu et al. [21] proposed to prune the filters in CNN by training the model with the regularization loss function so that redundancy weights converge to zero directly without sacrificing the performance. By analyzing the connection sensitivity among neurons and layers, Li et al. [22] 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 [23]. With vector quantization, Gong et al. [24] quantize the DL models. Similarly, Zhou et al. [25] 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. [26] quantizes weights while keeping the activations at full-resolution. The training algorithm by Jacob et al. [27] 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. [28] have shown that model compression can accelerate the processing of channel state information (CSI) acquisition and signal detection in massive multi-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 for make it affordable for IoT devices:

- **Question 1:** How to design semantic communication system over wireless fading channels?
- **Question 2:** How to form the constellation to reduce the burden on antenna?
- **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. Specially, a lite DeepSC is proposed (L-DeepSC) to address the above questions. 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 aforementioned Question 1.
- To alleviate the burden on antenna for data transmission and receiving, 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 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 $\mu$ and covariance $\sigma^2$. \((\cdot)^T\) 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 generates 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 [29].

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 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 devices.

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 semantic level accurately, but also exploited as the input of others 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 [17]. We focus on the communication between cloud/edge platform 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, $s = [w_1, w_2, \cdots, w_N]$, where $w_n$ represents the $n$-th word in the sentence. The encoded symbol stream can be represented as

$$X = C_\alpha S_\beta (s),$$  \hspace{1cm} (1)

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

If $X$ is sent to a wireless fading channel, the signal received at the receiver can be given by

$$Y = f_H (X) = HX + N,$$  \hspace{1cm} (2)
where $\mathbf{H}^1$ represents the channel gain between the transmitter and the receiver, and $\mathbf{N} \sim \mathcal{CN}(0, \sigma_n^2)$ is additive white Gaussian noise (AWGN).

The decoded signal can be represented as
$$\hat{s} = S_X^{-1} C_{\delta}^{-1}(\mathbf{Y}),$$

where $\hat{s}$ is the recovered sentence, $C_{\delta}^{-1}()$ is the channel decoder with parameter set $\delta$ and $S_X^{-1}()$ is the semantic decoder network with parameter set $\chi$.

The whole semantic communication can be trained by the cross-entropy (CE) loss function, which is given by
$$L_{\text{CE}}(s, \hat{s}) = \sum_{i=1}^{\infty} q(w_i) \log (1 - p(w_i)) - \sum_{i=1}^{\infty} q(w_i) \log (p(w_i)),
$$

where $q(w_i)$ is the real probability that the $i$-th word, $w_i$, appears in source sentence $s$, and $p(w_i)$ is the predicted probability that the $i$-th word, $w_i$, appears in $\hat{s}$. 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, $s$. 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, $s$, in the DeepSC, will cause that the learned constellation is no longer limited to a few points any more. 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
$$\hat{s} = S_X^{-1} C_{\delta}^{-1}(\mathbf{Y}),
$$

**Back-propagation:** Taking semantic encoder as an example, the parameter vector at the $t_{th}$ iteration are is updated by
$$\beta(t) = \beta(t-1) - \frac{\partial L_{\text{CE}}}{\partial \beta},$$

where $\eta$ is the learning rate and $\frac{\partial L_{\text{CE}}}{\partial \beta}$ is the gradient, computed by
$$\frac{\partial L_{\text{CE}}}{\partial \beta} = \frac{\partial L_{\text{CE}}}{\partial \hat{s}} \frac{\partial \hat{s}}{\partial \mathbf{Y}} \frac{\partial \mathbf{Y}}{\partial \mathbf{X}} \frac{\partial \mathbf{X}}{\partial \beta},$$

In (7), $\mathbf{H}$ will introduce stochasticity during weight updating. For an AWGN channel, $\mathbf{H} = \mathbf{I}$ will not affect it. The DL model, thus, can achieve the global optimum. However, for fading channels, $\mathbf{H}$ is random, which leads to that $\beta$ 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 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 $\pm 3.40282 \times 10^{38}$. Although DeepSC can learn the constellations from the source information and channel statistics, the learned constellation points, such as cluster constellation [30], 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 antenna design for IoT devices. Therefore, it is desired to form feasible constellation 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 its signal processing ability, which however increase computational complexity and model size and result 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 channel in wireless communication is to use 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, $s$, is embedded into, $S$. Then, the IoT devices encodes $S$ into

$$X = \sigma (W_T S + b_T),$$  \hfill (8)

where $X$ is the semantic features transmitted from the IoT devices to the cloud/edge platform. $W_T$ and $b_T$ are the trainable parameters to extract the features from source message $s$, and $\sigma(\cdot)$ is the sigmoid activation function.

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

$$\hat{S} = \sigma (W_R Y + b_R),$$  \hfill (9)

where the estimated source message, $\hat{s}$, can be obtained after de-embedding layer. $W_R$ and $b_R$ can learn to recover $s$. 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.

**Back-propagation:** It updates parameter $W_T$ by its gradient

$$\frac{\partial \mathcal{L}_{CE} (\hat{s}, s)}{\partial W_T} = (F_R W_R H F_T)^T \nabla_s \mathcal{L}_{CE} (\hat{s}, s) s^T,$$  \hfill (10)

where $F_R \sim \text{diag}(\sigma'(W_R Y + b_R))$ and $F_T \sim \text{diag}(\sigma'(W_T S + b_T))$. In (10), the $H$ is untrainable and random, therefore it will cause perturbation. If the transmitter consists of very deep neural networks, the gradient contamination will affect the back-propagation of the whole transmitter network.

**Forward-propagation:** With the received signal $W_R$, the source messages can be recovered by

$$\hat{S} = \sigma (W_R Y + b_R)$$
$$= \sigma (W_R H X + W_R N + b_R).$$  \hfill (11)

In (11), $W_R$ 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 $H$ is known, then the received symbol can be processed by

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

where $\tilde{N} = (H^H H)^{-1} H^H N$. In (12), the channel effect is transferred from multiplicative noise to additive noise, $\tilde{N}$, 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 $H = I$ 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, by 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 LS estimator requires no prior channel information.

For simplicity, we initially use the LS estimator. Then, we adopt the deep de-noise network to increase the resolution of LS estimator as in [31] shown in Fig. 3. Particularly, the rough CSI estimated by LS estimator with few pilots first denoted by

$$H_{\text{rough}} = H + N.$$  \hfill (13)

From (13), $H_{\text{rough}}$ consists of exact $H$ and the noise, $N$. Deep noise neural networks are used to recover $H$ more accurately from $H_{\text{rough}}$ by considering $H$ and $H_{\text{rough}}$ as the original picture and noisy picture, respectively. Here, we exploit attention-guided denoising convolutional neural network (ADNet) [32] to refine CSI, where the refined CSI, $H_{\text{refine}}$ denoted by

$$H_{\text{refine}} = \text{ADNet}(H_{\text{rough}}).$$  \hfill (14)

In (14), the $\text{ADNet}(\cdot)$ is trained the the loss function,

$$\mathcal{L}(H_{\text{refine}}, H) = \frac{1}{2} \|H_{\text{refine}} - H\|_F^2.$$  

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 trained L-DeepSC model are still very large, which cause high latency for the cloud/edge platform to broadcast updated L-DeepSC. Note that both

```
---

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

---

\textbf{Question 1.}

---

\textbf{Question 2.}

---

\textbf{Question 3.}

---

\textbf{Question 4.}

---
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.

1) Network Sparsification: A proper criterion to disable neural connections is important. Obviously, the connections with small weight value 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. Assume there are total $N$ layers in the pre-trained DeepSC model with $W_{i,j}^{(n)}$ 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 $w_{\text{thre}}$, the model weights can be pruned by

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

(15)

We determine the pruning threshold by

$$w_{\text{thre}} = s_M \times \gamma,$$

(16)

where $s = \text{sort}([W^{(1)}, W^{(2)}, \ldots, W^{(N)}])$, is the sorted weights value from least important one to the most important one, $M$ is the total number of connections, and $\gamma$, 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-tune to recover the accuracy, as shown in Algorithm 1.

2) Network Quantization: The quantization includes weight quantization and activation quantization. The weights, $W_{i,j}^{(n)}$, from a trained model, can be converted from 32-bit float point to $m$-bits integer through applying the quantization function by

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

(17)

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

$$q_w = \frac{2^m - 1}{\max \left( W^{(n)} \right) - \min \left( W^{(n)} \right)}.$$  

(18)

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 same. Therefore, accumulators are usually implemented with higher bit-widths, for example, INT32 += INT8 $\times$ 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 statistically 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 into 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
\[ x_{\min}^{(n)}(t + 1) = (1 - c) x_{\min}^{(n)}(t) + c \min (X^{(n)}(t)), \tag{19} \]
and
\[ x_{\max}^{(n)}(t + 1) = (1 - c) x_{\max}^{(n)}(t) + c \max (X^{(n)}(t)), \tag{20} \]
where \( x_{\min}^{(n)}(t + 1) \) and \( x_{\max}^{(n)}(t + 1) \) are used for the range of activation quantization, and \( x_{\min}^{(n)}(1) = \min (X^{(n)}(1)) \), \( x_{\max}^{(n)}(1) = \max (X^{(n)}(1)) \), (\( X^{(n)}(t) \)) is the output of activations at \( n_{th} \) layer with \( t_{th} \) batch data, \( c \in [0, 1] \) represents the correlation between the current \( x_{\min}^{(n)}(t) \) and \( x_{\max}^{(n)}(t) \) 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
\[ \tilde{X}^{(n)} = \text{clamp} \left( \text{round} \left( q_x \left( X^{(n)} - x_{\min}^{(n)} \right) \right) ; -M, M \right), \tag{21} \]
where \( q_x = (2^m - 1)/(x_{\max}^{(n)} - x_{\min}^{(n)}) \) is the scale-factor and clamp(.) is used to eliminate the quantized outliers, which is given by
\[ \text{clamp} \left( X^{(n)}; -T, T \right) = \min \left( \max (X^{(n)}, -T), T \right), \tag{22} \]
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 (17) directly. In phase 2, 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, straight-through estimator (STE) is used to estimate the gradient of quantized weights in the back-propagation [33].

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, the antenna of IoT devices is not able to create high-resolution wave, in other words, the antenna cannot afford a large number of constellation points close to each other.

Different from bits, the source message, \( s \), is more complicated and the learned constellation will not be limited to few points, which brings additional burden on antenna design. Besides, the DL model 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 \( X \) is the learned constellation while \( X \) 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
\[ X_{\text{dequantize}} = \frac{X_{\text{quantize}}}{q_x} + x_{\min}, \tag{23} \]
where \( X_{\text{quantize}} \) is the quantized \( X \) from (21), \( q_x \) is the scale-factor and \( x_{\min} \) is the obtained by (19) and \( X_{\text{dequantize}} \) is the dequantized \( X \).

First, we quantize the \( X \) into \( m \)-bits integer so that the range of \( X \) is narrowed to the size of \( 2^m \). For example, when \( m = 8 \), the size of constellation is reduced to 256. Then, \( X_{\text{quantize}} \) is dequantize to restore \( X \). Such an \( X_{\text{dequantize}} \) has the similar distribution as \( X \) but is with fewer constellation points, which is helpful to simplify antenna design at receiver 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 \( CN(0, 1) \); for the Rician fading channel, it follows \( CN(\mu, \sigma^2) \) with \( \mu = \sqrt{E/(k + 1)} \) and \( \sigma = \sqrt{1/(k + 1)} \). where \( k \) is Rician coefficient and we use \( k = 2 \) in our simulation.

The transmitter of L-DeepSC is the same as that of DeepSC in [17]. 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 1 and Dense 3 is the input of LayerNorm layer.

The adopted dataset is the proceedings of the European Parliament [34], 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.

![Table I](image-url)
(Huffman coding) and fixed-length coding (5-bit) for source coding, Reed-Solomon (RS) coding [35] for channel coding, and quadrature amplitude modulation (QAM). The bilingual evaluation understudy (BLEU) score is used to measure the performance [36].

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 antenna, 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 the similar distribution with the full-resolution constellation. For example, there exists certain blank region in the edge of constellation in Fig. 5(a), while the 4-bits constellation shows the 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 SNR > 9 dB, which demonstrate 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 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 LS estimator in the low SNR 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 the L-DeepSC is trained with perfect CSI, rough CSI by (13), refined CSI by (14) and without CSI, respectively. The traditional approaches are Huffman coding with (5,7) RS and 5-bit coding with (7,9) RS, both with 64-QAM. We observe that all DL-enabled approaches are more competitive under the fading
channels. 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 light-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 outperform that with rough CSI. As SNR increases, all these systems, L-DeepSC with perfect CSI, refined CSI and rough CSI, converge to similar performance gradually.

C. Model Compression

In this experiment, we investigate the performance of network slimmer, including network sparcification, 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, $\gamma$, on the BLEU scores with different SNRs under the Rician fading channels, where the system is pruned directly when $\gamma$ increases from 0 to 0.9 and is pruned with fine-tuning when $\gamma$ increases to 0.99 continually. The proposed L-DeepSC achieves almost the same BLEU scores when the $\gamma$ increases from 0 to 0.9, which shows that there exists a mass of weights redundancy in the trained DeepSC model. When the $\gamma$ 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, 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 scores and network sparsity ratio, $\gamma$. The BLEU scores drop significantly as the network sparsity ratio increases from 0 to 0.9, which shows that there exists a mass of weights redundancy in the trained DeepSC model. When the $\gamma$ 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, 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.
Table II

| Pruned Model | BLEU score with $m = 4$ | $\psi$ | BLEU score with $m = 8$ | $\psi$ | BLEU score with $m = 12$ | $\psi$ | BLEU score with $m = 16$ | $\psi$ |
|--------------|-----------------------|-------|-----------------------|-------|-----------------------|-------|-----------------------|-------|
| $\gamma = 0.3$ | 0.838967 | 11.429 | 0.892745 | 5.714 | 0.908537 | 3.81 | 0.910184 | 2.857 |
| $\gamma = 0.6$ | 0.835863 | 20.0 | 0.897143 | 10.0 | 0.90815 | 6.667 | 0.900468 | 5.0 |
| $\gamma = 0.9$ | 0.810322 | 80.0 | 0.895306 | 40.0 | 0.898784 | 26.667 | 0.910584 | 20.0 |
| $\gamma = 0.95$ | 0.779685 | 160.0 | 0.875814 | 80.0 | 0.873426 | 53.333 | 0.877221 | 40.0 |

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

The BLEU score and the quantization bit number, $m$, under the Rician fading channels, where $m$ is defined in (18), 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 SNR range since most key information are removed in the low-resolution neural network.

Table II compares the BLEU scores and compression ratios under different combinations of weights pruning and weights quantization with SNR = 12 dB, where the compression ratio is computed by

$$\psi = \frac{M \times 32}{M_{pruned} \times m}, \quad (24)$$

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 $\gamma$ increases or $m$ decreases, which are consistent with Fig. 15 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 high compression ratio.

V. Conclusion

In this paper, we proposed a lite distributed semantic communication system, named L-DeepSC, for 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 less pilot overheads was developed to eliminate the effects from fading channels. Besides, we map the full-resolution original constellation into finite bits constellation to match the current antenna design, 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 in 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 things: a survey,” Computer Networks, 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.
R. Szeliski, Computer Vision: Algorithms and Applications. Springer Science & Business Media, 2010.

N. Indurkhya and F. J. Damerau, Handbook of Natural Language Processing. CRC Press, 2010, vol. 2.

C. E. Shannon and W. Weaver, The Mathematical Theory of Communication. The University of Illinois Press, 1949.

D. Tse and P. Viswanath, Fundamentals of Wireless Communication. Cambridge University Press, 2005.

Z. Qin, H. Ye, G. Y. Li, and B.-H. P. Juang, “Deep learning in physical layer communications,” IEEE Wireless Commun., vol. 26, no. 2, pp. 93–99, Apr. 2019.

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.

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.

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.

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.

R. Krishnamoorthi, “Quantizing deep convolutional networks for efficient inference: A whitepaper,” arXiv:1806.08342, 2018. [Online]. Available: http://arxiv.org/abs/1806.08342

Y. Gong, L. Liu, M. Yang, and L. Bourdev, “Compressing deep convolutional networks using vector quantization,” arXiv:1412.6115, 2014. [Online]. Available: http://arxiv.org/abs/1412.6115

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.

F. Li, B. Zhang, and B. Liu, “Ternary weight networks,” arXiv:1605.04711, 2016. [Online]. Available: http://arxiv.org/abs/1605.04711

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. 18-22, 2018, pp. 2704–2713.

J. Guo, J. Wang, C.-K. Wen, S. Jin, and G. Y. Li, “Compression and acceleration of neural networks for communications,” IEEE Wireless Commun., Early Access.

D. Gil, A. Ferrández, H. Mora-Mora, and I. Peral, “Internet of things: A review of surveys based on context aware intelligent services,” Sensors, vol. 16, no. 7, p. 1069, Jul. 2016.

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.

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.

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.