Deep Learning Enabled Semantic Communication Systems

Huiqiang Xie, Zhijin Qin, Member, IEEE, Geoffrey Ye Li, Fellow, IEEE,and Biing-Hwang Juang Life Fellow, IEEE

摘要

最近，人们开发了支持深度学习的端到端（E2E）通信系统，以合并传统通信系统中的所有物理层块，从而使联合收发器优化成为可能。在深度学习的推动下，自然语言处理（NLP）在分析和理解大量语言文本方面取得了巨大成功。受这两个领域研究成果的启发，我们的目标是从语义层面对通信系统提供一个新的视角。特别地，我们提出了一个基于深度学习的语义通信系统，名为 DeepSC，用于文本传输。基于该转换器，DeepSC 旨在通过恢复句子的意义，而不是传统通信中的位或符号错误，来最大限度地提高系统容量并最小化语义错误。此外，迁移学习用于确保 DeepSC 适用于不同的通信环境，并加速模型训练过程。为了准确证明语义通信的性能，我们还初始化了一个新的度量，称为句子相似度。大量的仿真结果表明，与不考虑语义信息交换的传统通信系统相比，该系统对信道变化具有更强的鲁棒性，能够在低信噪比下获得更好的性能。

关键词：深度学习；端到端通信；语义通信；迁移学习；Transformer

1 引言

基于 Shannon 和 Weaver, 通信有三个级别

1）符号的传播；

2）传输符号的语义交换；

3）语义信息交换的影响。

第一级通信主要关注符号从发射机成功传输到接收机，其中传输精度主要在比特或符号级别进行测量。

第二级通信处理从发送方发送的语义信息和在接收方解释的含义，称为语义通信。

第三级处理通信的影响，这些影响转化为接收器以发送器所期望的方式执行某些任务的能力。

在过去的几十年里，通信主要集中在如何准确有效地将符号（以比特为单位）从发射机传输到接收机。在这种系统中，误码率（BER：Bit Error Ratio）或 SER（Symbol Error Ratio）表示误符号率。通常被视为性能指标。随着第一代（1G）到第五代（5G）的发展，实现的传输速率提高了数万倍，系统容量逐渐接近单端极限。

最近，各种新的应用出现了，例如自动运输、消费机器人、环境监测和远程健康。这些应用程序的互连将产生惊人的字节数量的数据。此外，这些应用程序需要在有限的频谱资源上支持大规模连接，但需要较低的延迟，这对传统的信源信道编码提出了严峻挑战。

语义通信可以通过提取数据的含义，过滤掉无用、无关和非本质的信息，从而在保留语义的同时进一步压缩数据，从而在语义域中处理数据。

此外，语义通信有望对恶劣的信道环境，即低信噪比（SNR）区域具有鲁棒性，这非常适合要求高可靠性的应用。这些因素促使我们通过考虑数字比特背后的语义来开发智能通信系统，以提高通信

的准确性和效率。

与传统通信不同，语义通信旨在传输与传输目标相关的信息。例如，对于文本传输，意义因此是基本信息，而表达，即是单词的表达，是不必要的。通过这样做，数据流量将显著减少。在典型通信系统中，当带宽有限、信噪比较低或误码率/误码率较高时，这样的系统可能特别有用。

基于深度学习（DL）的自然语言处理（NLP）和通信系统的最新进展启发我们研究语义通信，以实现上述第二级通信。所考虑的语义通信系统主要关注联合语义信道编码和解码，其目的是提取和编码句子的语义信息，而不是简单地一个比特序列或一个单词。

对于语义通信系统，我们面临以下问题：

Question 1: 如何定义 bits?

Question 2: 如何衡量句子的语义错误？

Question 3: 如何联合设计语义和信道编码？

本文将 NLP 中的机器翻译技术应用到物理层通信中，研究语义通信系统。具体来说，我们提出了一个支持 DL 的语义通信系统（DeepSC）来应对上述挑战。

2 相关工作

本节简要回顾了 E2E 物理层通信系统和 NLP 中采用的深度神经网络（DNN）技术的相关工作。

2.1 端到端物理层通信系统

DL 技术在处理各种智能任务（即计算机视觉和 NLP）方面显示出巨大的潜力。同时，由于硬件计算能力的提高，可以训练神经网络并在移动设备上运行它们。在通信领域，已经对基于 DL 的 E2E 物理层通信系统进行了一些开创性工作，这些系统合并了传统通信系统中的块[1-7]。通过采用 DL 中的自动编码器结构和去除块结构，E2E 系统中的发射机和接收机作为 E2E 重建任务被联合优化。

2.2 自然语言处理中的语义表示

NLP 使机器理解人类语言，主要目标是理解语法和文本。最初，自然语言可以根据上下文通过联合概率模型来描述[8]。因此，语言模型提供上下文来区分具有相似语义的单词和短语。

3 本文方法

此部分对本文将要复现的工作进行概述，所考虑的系统模型包括两个层次：语义层次和传输层次，如图 1所示。语义层处理用于编码和解码的语义信息处理，以提取语义信息。传输层保证语义信息可以在传输介质上正确交换。总体而言，我们考虑具有随机物理信道的智能 E2E 通信系统，其中发射机和接收机具有一定的背景知识，即不同的训练数据。对于不同的应用场景，背景知识可能是不同的。

3.1 问题概述

如图 1所示，发射机将一个句子 **s** 映射到一个复杂的符号流 **x** 中，然后将其通过具有传输损害（如失真和噪声）的物理信道。接收到的 **y** 在接收器处解码以估计原始句子 **s**。我们使用 DNN 共同设计发射器和接收器，因为 DL 使我们能够训练输入可变长度句子和不同语言的模型。

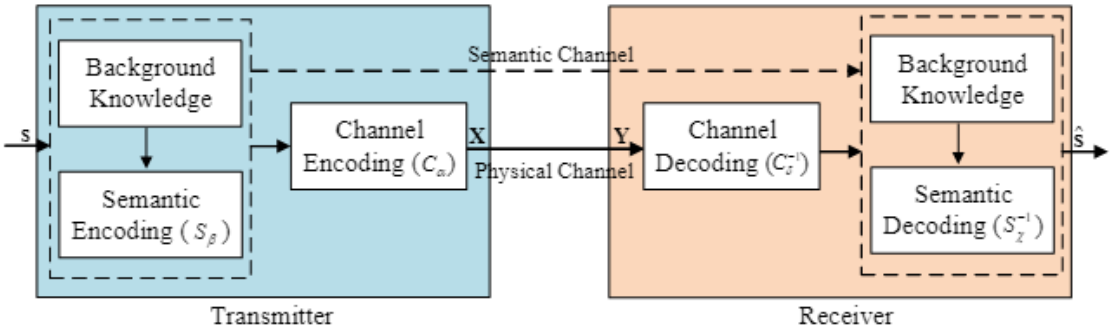

图 1: 提出的支持 DL 的语义通信系统 DeepSC 的框架

特别地，我们假设 DeepSC 的输入是一个句子，$\mathbf{s} = [w_1, w_2, \cdot\cdot, w_L]$，其中 w_l 表示句子中的第 l 个单词。如图 1所示，发射机由两个部分组成，即语义编码器和信道编码器，以从中提取语义信息，并保证语义信息在物理信道上的成功传输。编码符号流可以表示为

$$\mathbf{x} = C_\alpha(S_\beta(\mathbf{s})), \tag{1}$$

其中 $\mathbf{x} \in C^{M\times 1}$，$S_\beta(\cdot)$ 是具有参数集 β 的语义编码器网络，$C_\alpha(\cdot)$ 为具有参数集 α 的信道编码器。为了简化分析，我们假设相干时间为 M。如果发送 **x**，则接收器接收的信号将为

$$\mathbf{y} = h\mathbf{x} + \mathbf{n}, \tag{2}$$

其中 $\mathbf{y} \in C^{M\times 1}$，$h$ 表示具有 $\mathcal{CN}(0,1)$ 和 $\mathbf{n}\sim \mathcal{CN}(0,\sigma_n^2)$ 的瑞利衰落信道。对于编码器和解码器的 E2E 训练，信道必须允许反向传播。物理信道可以由神经网络表示。例如，可以使用简单的神经网络来模拟 **AWGN** 信道、乘法高斯噪声信道和擦除信道[6]。而对于衰落信道，需要更复杂的神经网络[4]。在本文中，为了简单起见，我们主要考虑 **AWGN** 信道和瑞利衰落信道，同时关注语义编码和解码。

3.2 信道编码器和解码器的设计

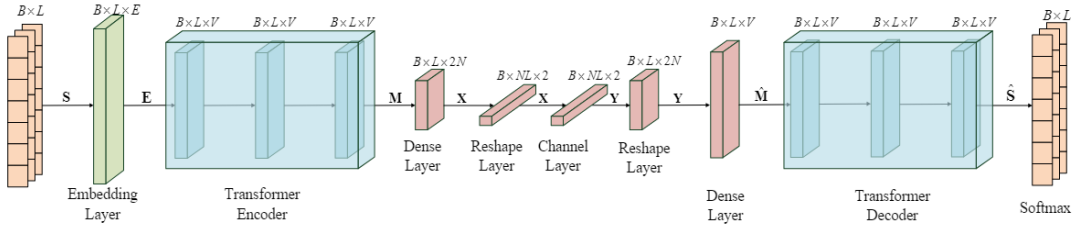

图 2: 语义通信系统的神经网络结构。

3.3 损失函数定义

$$\mathcal{L}_{total} = \mathcal{L}_{CE}(s,\hat{s};\alpha,\beta,\chi,\sigma) - \lambda\mathcal{L}_{MI}(x,y;\mathbf{T},\alpha,\beta), \tag{3}$$

其中第一项是考虑句子相似性的损失函数，其目的是通过训练整个系统来最小化 **s** 和 $\hat{\mathbf{s}}$ 之间的语义差异。第二个是互信息的损失函数，它在发送端训练期间最大化所获得的数据速率。参数 λ（0≤λ≤1）是第二项的权重。

4 复现细节

4.1 与已有开源代码对比

与开源代码相比，我主要是增加了数据预处理工作，因为开源代码没有这一部分的内容，故而给复现论文增添了难度。主要工作是去除文本数据的冗余信息和构造词典等。以下是部分自己的工作代码：

```
def clean_lines(lines):
    cleaned = list()
    # prepare regex for char filtering
    re_print = re.compile('[^%s]' % re.escape(string.printable))
    # prepare translation table for removing punctuation
    table = str.maketrans('', '', string.punctuation)
    for line in lines:
        # normalize unicode characters
        line = normalize('NFD', line).encode('ascii', 'ignore')
        line = line.decode('UTF-8')
        # tokenize on white space
        line = line.split()
        # convert to lower case
        line = [word.lower() for word in line]
        # remove punctuation from each token
        line = [word.translate(table) for word in line]
        # remove non-printable chars form each token
        line = [re_print.sub('', w) for w in line]
        # remove tokens with numbers in them
        line = [word for word in line if word.isalpha()]
        # store as string
        cleaned.append(' '.join(line))
    return cleaned

def update_dataset(lines, vocab):
    new_lines = list()
    for line in lines:
        new_tokens = list()
        for token in line.split():
            if token in vocab:
                new_tokens.append(token)
            else:
                new_tokens.append('unk')
        new_line = ' '.join(new_tokens)
        new_lines.append(new_line)
    return new_lines
```

部分开源代码核心内容如下：

```
class SemanticCommunicationSystem(nn.Module):  # pure DeepSC
    def __init__(self):
        super(SemanticCommunicationSystem, self).__init__()
        self.embedding = embedding(36214, 128)  # which means the corpus has
           36214 kinds of words and
        # each word will be coded with a 128 dimensions vector
        self.frontEncoder = nn.TransformerEncoderLayer(d_model=128, nhead=8)  #
            according to the paper
        self.encoder = nn.TransformerEncoder(self.frontEncoder, num_layers=3)
        self.denseEncoder1 = dense(128, 256)
        self.denseEncoder2 = dense(256, 16)

```

```
        self.denseDecoder1 = dense(16, 256)
        self.denseDecoder2 = dense(256, 128)
        self.frontDecoder = nn.TransformerDecoderLayer(d_model=128, nhead=8)
        self.decoder = nn.TransformerDecoder(self.frontDecoder, num_layers=3)

        self.prediction = nn.Linear(128, 36214)
        self.softmax = nn.Softmax(dim=2)  # dim=2 means that it calculates
           softmax in the feature dimension

    def forward(self, inputs):
        embeddingVector = self.embedding(inputs)
        code = self.encoder(embeddingVector)
        denseCode = self.denseEncoder1(code)
        codeSent = self.denseEncoder2(denseCode)
        codeWithNoise = AWGN_channel(codeSent, 12)  # assuming snr = 12db
        codeReceived = self.denseDecoder1(codeWithNoise)
        codeReceived = self.denseDecoder2(codeReceived)
        codeSemantic = self.decoder(codeReceived, code)
        codeSemantic = self.prediction(codeSemantic)
        info = self.softmax(codeSemantic)
        return info


class MutualInfoSystem(nn.Module):  # mutual information used to maximize
   channel capacity
    def __init__(self):
        super(MutualInfoSystem, self).__init__()
        self.fc1 = nn.Linear(32, 256)
        self.fc2 = nn.Linear(256, 256)
        self.fc3 = nn.Linear(256, 1)
        # nn.init.normal_(self.fc1.weight, std=0.02)  # init weight with normal
            distribution and mean is 0, std is 0.02
        # nn.init.constant_(self.fc1.bias, 0)  # init bias with constant num 0
        # nn.init.normal_(self.fc2.weight, std=0.02)
        # nn.init.constant_(self.fc2.bias, 0)
        # nn.init.normal_(self.fc3.weight, std=0.02)
        # nn.init.constant_(self.fc3.bias, 0)  # which may not be necessary to
           initialize weight manually

    def forward(self, inputs):
        output = F.relu(self.fc1(inputs))
        output = F.relu(self.fc2(output))
        output = F.relu(self.fc3(output))
        return output
```

5 实验结果分析

本部分对实验所得结果进行分析，详细对实验内容进行说明，实验结果进行描述并分析。

5.1 性能指标

首先是 BLEU 分数，在 E2E 通信系统中，发送端和接收端通常以误码率作为训练目标，这有时会忽略通信的其他方面目标。对于文本传输，误码率不能很好地反映性能。除了通过人工判断来确定句子之间的相似性外，机器翻译中通常使用双语评估研究（Bilingual Evaluation Understudy：BLEU）分数来衡量结果，这将作为本文的性能指标之一。

$$log\mathbf{BLEU} = min(1 - \frac{l_{\hat{s}}}{l_s}) + \sum_{n=1}^{N} u_n log p_n \tag{4}$$

然而，BLEU 分数只能比较两句话中单词之间的差异，而不能比较它们的语义信息。因此，我们初始化了一个新的度量，称为句子相似度，以根据两个句子的语义信息来描述它们的相似程度。

$$match(\hat{s}, s) = \frac{B_\phi(s) \cdot B_\phi(\hat{s})^T}{\| B_\phi(s) \cdot \| \| B_\phi(\hat{s}) \|} \tag{5}$$

其中，代表 BERT 的 B_ϕ，是一个庞大的预训练模型，包括数十亿个用于提取语义信息的参数。定义的句子相似性是一个个于 0 和 1 之间的数字，表示解码后的句子与传输的句子有多相似。

5.2 与原论文结果对比

在图 3中，右边图中黑线是原论文方法实验结果，左边两幅图是复现出来的实验结果，因为数据预处理部分可能有所不同，所以实验结果数据有一定的差异，但相差不大。

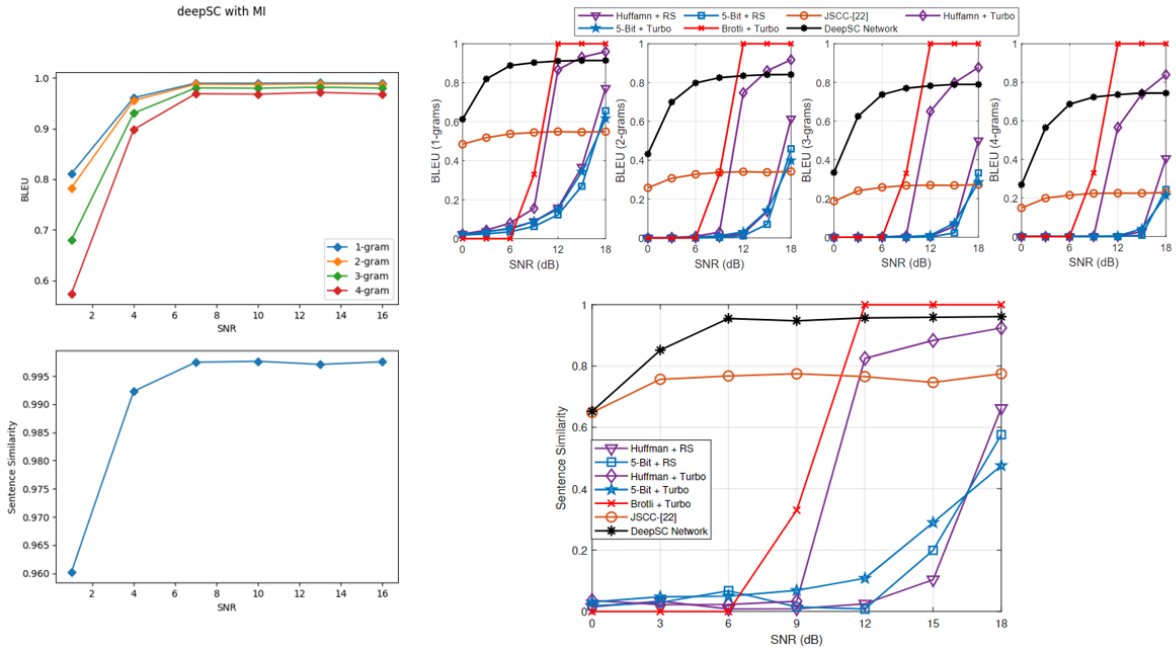

图 3: 复现结果与论文实验结果对比示意

下图是对比有无互信息作为损失项的实验，可以看到考虑互信息损失项的效果无论是句子相似度还是 BLEU 分数都能取得更好的效果。

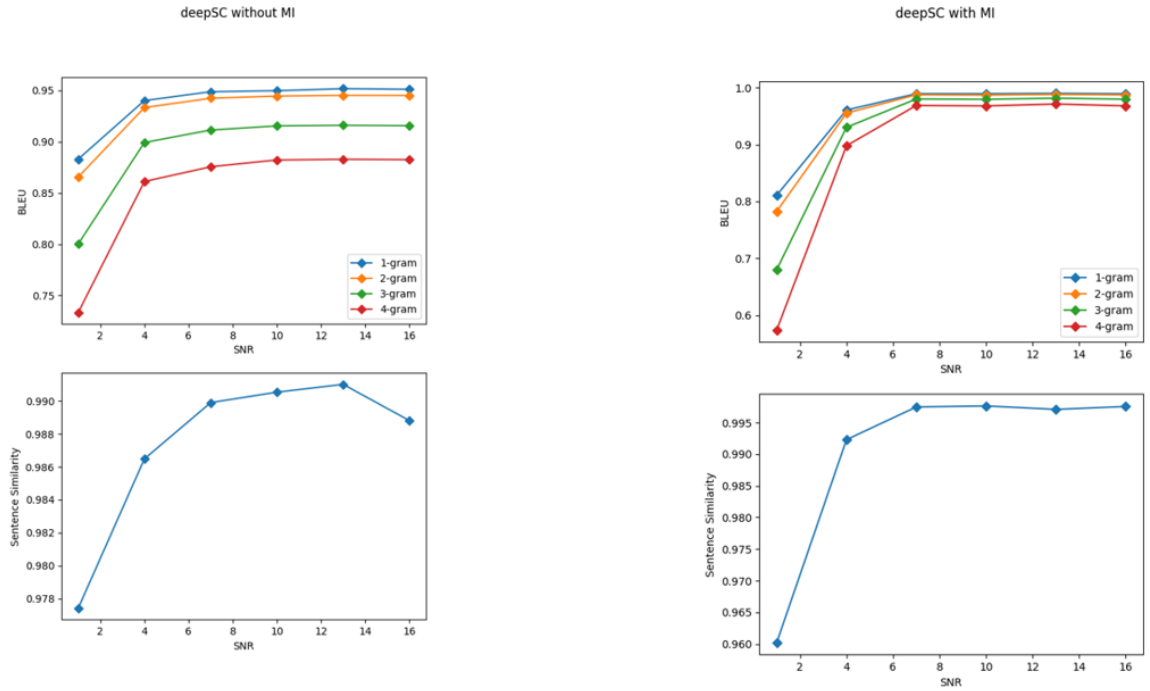

图 4: 互信息对比实验

6 总结与展望

本部分对整个文档的内容进行归纳并分析目前实现过程中的不足以及未来可进一步进行研究的方向。原论文提出了一个语义通信系统，名为 DeepSC，在 DeepSC 中，输入文本和输出符号的长度是可变的，互信息被视为损失函数的一部分，以实现更高的数据速率。此外，还采用了深度迁移学习来满足不同的传输条件，并利用预先训练好的模型中的知识加快新网络的训练。此外，还将句子相似性作为一个新的语义错误性能指标进行了初始化，这是一个更接近人类判断的指标。仿真结果表明，DeepSC 的性能优于各种基准，尤其是在低信噪比情况下。所提出的迁移学习辅助的 DeepSC 已经证明了它能够以较快的收敛速度适应不同的渠道和知识。此次所复现的结果与论文中所列结果大体一致。目前的实现仍缺少一定的创新，且复现论文的语义通信方法较为简单，是由数据驱动的语义通信，我认为这还不是真正的语义通信，未来的研究方向可以是有知识驱动的带有一定推理能力的语义通信。

参考文献

[1] O'SHEA T, HOYDIS J. "An introduction to deep learning for the physical layer[J]. IEEE Trans. Cogn. Comm. & Networking, 2017, 3: 563-575.

[2] DORNER S, CAMMERER S, HOYDIS J, et al. Deep Learning Based Communication Over the Air[J]. IEEE Journal of Selected Topics in Signal Processing, 2018.

[3] AOUDIA F A, HOYDIS J. Model-free Training of End-to-end Communication Systems[J]. arXiv: Information Theory, 2018.

[4] YE H, LIANG L, LI G Y, et al. Deep Learning based End-to-End Wireless Communication Systems with Conditional GAN as Unknown Channel[J]. Cornell University - arXiv, 2019.

[5] PARK S, SIMEONE O, KANG J. End-to-End Fast Training of Communication Links Without a Channel Model via Online Meta-Learning[J]. Cornell University - arXiv, 2020.

[6] FARSAD N, RAO M, GOLDSMITH A. Deep Learning for Joint Source-Channel Coding of Text[J]. arXiv: Information Theory, 2018.

[7] BOURTSOULATZE E, KURKA D B, GUNDUZ D. Deep Joint Source-Channel Coding for Wireless Image Transmission[J]. IEEE Transactions on Cognitive Communications and Networking, 2018.

[8] KNESER R, NEY H. Improved backing-off for M-gram language modeling[J]. International Conference on Acoustics, Speech, and Signal Processing, 1995.