CS230

Deep Learning for Wireless Channel Coding

Yun Liao
Department of Electrical Engineering
Stanford University
yunliao@stanford.edu

Abstract

Channel coding is a key to improve transmission reliability in wireless communication. A channel coding scheme specifies an encoder that constructs a redundant version of the original message to be sent through a noisy channel, and a decoder that recovers the original message from the redundant noisy received signal. Most existing channel coding schemes are developed based on additive white Gaussian noise (AWGN) channel, or require precise channel state information (CSI) in implementation, which may not be the case in real world scenarios. This project mainly investigates the potential of using deep neural networks (DNNs) to improve the channel coding in wireless communication systems without knowing the channel a priori. Two major cases are considered: (1) fixed length polar encoder with DNN decoder for short messages; (2) recurrent neural network (RNN)-based encoder and decoder for long messages or messages with flexible length. For the first case, DNN decoder significantly outperforms conventional decoder in non-AWGN channel. For the second case, the trained RNN network shows lower error rate than widely used convolutional code in highly noisy channel.

1 Introduction

In a typical wireless communication system, signals are transmitted through highly noisy and dynamic wireless channels. Achieving the highest possible data rate while maintaining satisfying transmission reliability has always been the goal for the research in this area. Conventionally, the transmission reliability is ensured by channel coding and modulation: the original message passes through an encoder that maps it to a longer binary code containing parity (extra) information, and then a modulator maps the code to a real or complex symbol sequence to be sent over the wireless channel. The receiver receives the noisy symbol sequence and uses a demodulator and a decoder to recover the original message. Conventionally, modulation is a rigid mapping (i.e., $0 \to 1, 1 \to -1$), and the codes are developed separately from the modulation, with additive white Gaussian noise (AWGN) channel or perfect channel state information (CSI) assumption.

In real world implementation, however, the underlaying channel may well not be AWGN, and acquiring precise CSI in real time is very costly. In this case, conventional methods may not work well. On the other hand, jointly optimizing them in a mathematical way under non-AWGN channel or under channels with no known model would be extremely hard. These observations motivate us to apply deep learning methods to this problem by replacing the conventional hand-designed transmission schemes with deep neural networks (DNNs) and evaluate how well deep learning can do.

In the rest of this report, a transmitter refers to the combination of an encoder and a decoder, i.e., it denotes a mapping from a binary message $\boldsymbol{x} = [x_1, x_2, \ldots, x_k]$ to a real sequence $\boldsymbol{c} =$

CS230: Deep Learning, Fall 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

$[c_1, c_2, \ldots, c_n]$. The code rate is defined as $r = k/n$. A receiver denotes a mapping from the noisy received sequence $\hat{\boldsymbol{c}}$ to a recovered message $\hat{\boldsymbol{x}}$. The system model is depicted in Fig. 1.

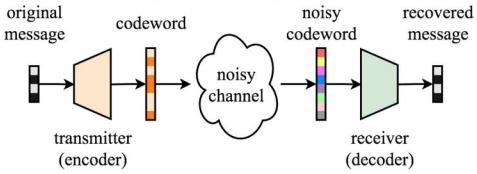

Figure 1: Block diagram of a typical wireless communication system.

In this project, we mainly consider the following two scenarios: (1) use a DNN receiver for a conventional transmitter; (2) replace both the transmitter and receiver with DNNs. For the first scenario, the input of the DNN is $\hat{\boldsymbol{c}}$, a real sequence of length n denoting the distorted symbol sequence, and the output is either a binary sequence of length k or a one-hot indicator of length 2^k. For the second scenario, the input of the DNN-transmitter is a k-bit binary sequence, and the output is a real sequence of length n. The output of the DNN-transmitter is connected to the input of the DNN-receiver through a noisy channel. The output of the DNN-receiver is a k-bit sequence.

2 Related work

People started to apply deep learning to the physical layer of communication systems very recently. Existing works can be roughly divided into three categories: use deep learning to improve the decoder for existing codes (1; 2; 3); use deep learning to understand channel effect better or decode from non-AWGN channel outputs (4; 5); and use deep learning to design end-to-end communication system (6; 7). In (1; 2), the authors used deep learning to improve the conventional belief propagation decoding algorithm. In (3), the authors compared how well can deep learning do in decoding structured code and random code. In (4), deep neural network was used to detect symbol sequence transmitted through a molecular channel without explicit mathematical model. In (5) used plain DNN to decode polar code from nonlinear channel. The authors claimed very good results by training their DNNs under exactly the same channel condition as the test sets, which may be impractical in real world scenario. In (6), the channel was modeled mathematically, based on which end-to-end communication scheme was learned, and dense neural networks were used for both encoder and decoder. The major problem is that the system is not scalable, and the number of trainable parameters is prohibitive. The authors in (7) trained an end-to-end communication system without channel model. They implemented dense neural networks for both transmitter and receiver, and trained them iteratively. The key issue is still the scalability: each piece of their message is only 8 bits.

3 Dataset and Features

Synthetic data are used for the experiments. For both scenarios, we start with AWGN channels. The main reasons behind this are: (1) AWGN channel is symmetric and memoryless. It is easy to generate synthetic samples; (2) most theories in communication theories are developed based on AWGN channel. There are mature ways to demodulate/decode the received signal to achieve theoretically guaranteed performance, which can be viewed as benchmarks.

3.1 First Scenario

For the first scenario, we set $k = 8$ and $n = 16$. Polar encoder (8) and binary phase-shift keying (BPSK) modulation $\{0 \to 1, 1 \to -1\}$ are used at the transmitter end. Eight training sets, each containing 2,000,000 samples, are constructed by selecting the channel type from {AWGN, nonlinear}, and training signal-to-noise ratio (SNR) from {4dB, 8dB, 10dB, ∞}. For AWGN channel, white Gaussian noise is added independently to the output of the transmitter. The variance of the noise is set according to the chosen SNR. For the nonlinear channel, we adopt the channel model used in (5). Specifically, the dispersive model is

$$y_i = 0.3482c_i + 0.8704c_{i-1} + 0.3482c_{i-2}, \tag{1}$$

and the nonlinear distortion is described as

$$|g(v)| = |v| + 0.2|v|^2 - 0.1|v|^3 + 0.5cos(\pi|v|). \tag{2}$$

White Gaussian noise is added after the nonlinear distortion and dispersion. Each sample $(\boldsymbol{x}, \hat{\boldsymbol{c}})$ in a certain training set is generated as follows: the message $\boldsymbol{x}$ (label) is a randomly generated 8-bit binary sequence. We take the message through the polar encoder, the modulator, and the channel to obtain the input of the DNN-receiver $\hat{\boldsymbol{c}}$.

We evaluate 44 test sets of size 100,000 samples for this scenario. Each test set is constructed in a similar way as the training set: we select the channel type from {AWGN, nonlinear} and test SNR from $\{i/2, i = 0, 1, \ldots, 21\}$dB for the set. Then, we randomly generate the messages $\boldsymbol{x}$'s, and pass them through the transmitter and channel to get corresponding $\hat{\boldsymbol{c}}$'s.

3.2 Second Scenario

The training set for this scenario contains 1,000,000 samples. Each sample contains a 100-bit binary sequence $\boldsymbol{x}$ and an AWGN channel realization vector $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \sigma^2)$. The training SNR for each sample is randomly picked from $[1, 5]$ dB.

For testing, 26 test sets of size 5000 are evaluated. In each set, the message length is either 100 bits or 200 bits, and the SNR is picked from $\{0, 0.5, \ldots, 5.0\}$ dB.

4 Methods

4.1 First Scenario

Since the message is short (8 bits), we have two choices for the output layer of the DNN receiver: 8 output units or $2^8 = 256$ output units. For these two choices, we adopt the dense neural network architectures in Fig. 2:

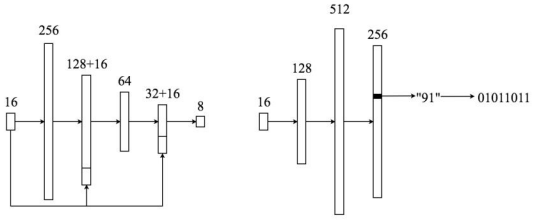

Figure 2: Architectures for DNN polar decoder. Left: DNN-receiver with 8 output units. Right: DNN-receiver with 2^8 output units.

The leaky-relu activation is used in all hidden layers for both architectures. In the left DNN, skip connections from the input layer to the second hidden layer and the fourth hidden layer are used to prevent gradient diminishing problem. The output layer uses sigmoid activation and binary cross-entropy loss:

$$\mathcal{L}_{CE}(\boldsymbol{x}, \hat{\boldsymbol{x}}) = -\frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{k} \left[x_j^{(i)} \log(\hat{x}_j^{(i)}) + (1 - x_j^{(i)}) \log(1 - \hat{x}_j^{(i)}) \right], \tag{3}$$

where $\boldsymbol{x}$ is the original 8-bit binary message, and $\hat{\boldsymbol{x}}$ is the output of the network. The choices of output layer activation function and loss are due to the fact that the target output is a binary sequence, so the learning task can be viewed as a multi-class classification, where the classes are not mutually exclusive.

In the right DNN, we use softmax as the activation function for the output layer, and categorical cross-entropy is used:

$$\mathcal{L}_{CCE}(c, \boldsymbol{p}) = -\frac{1}{m} \sum_{i=1}^{m} \log(p_{c^{(i)}}^{(i)}), \tag{4}$$

where $c \in \{0, 1, \ldots, 2^k - 1\}$ denotes the true message, and $\boldsymbol{p}$ is the output of the network indicating the probability of each class. Here, the labels c's are obtained by converting the binary sequence $\boldsymbol{x}$ to decimal numbers. We choose categorical cross-entropy loss for this architecture because the 2^k output classes are mutually exclusive.

Note that minimizing the loss in eq. (3) is directly related to minimizing bit error rate, while minimizing the loss in eq. (4) is directly related to minimizing block error rate.

4.2 Second Scenario

In this case, the message is long or even has flexible length, so it is impractical to take 2^k output units. It is even impractical to see all possible 2^k messages in the training set. Thus, it is critical to choose an architecture that is easily scalable w.r.t. various message length. Taking the sequential nature of messages into account, the multi-layer RNN structure in Fig. 3 is used for this scenario.

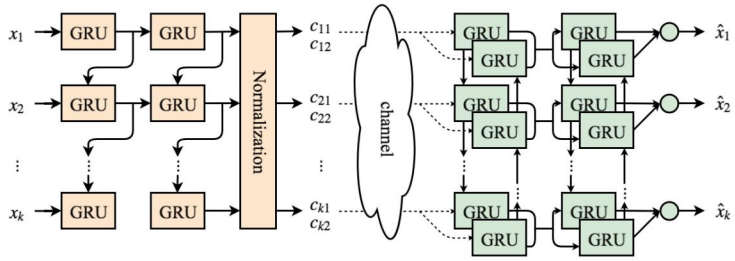

Figure 3: Architecture for joint DNN-transmitter and DNN-receiver learning.

The GRUs at the DNN-transmitter end have 20 hidden units, and the GRUs at the receiver end have 100 hidden units. In practical communication system, the transmit power is constrained. Thus, we use tanh activation for the second GRU layer in DNN-transmitter, and normalize the $\ell 2$-norm of transmitted signal. The outputs of the second layer GRU at the receiver end are fed into a dense layer with sigmoid activation function to get predictions.

For this case, we tried two loss functions, namely binary cross-entropy loss given in eq. (3), and mean squared error:

$$\mathcal{L}_{MSE} = \frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{k}(\hat{x}_j^{(i)} - x_j^{(i)})^2. \tag{5}$$

Experiments show that the model trained under mean squared error loss achieve better performance. The results shown in the next section are all trained under mean squared error loss.

5 Results and Discussion

5.1 First Scenario

The bit error rate (BER) curve and block error rate (BLER) curve under AWGN channel are shown in Fig. 4 and Fig. 5, respectively.

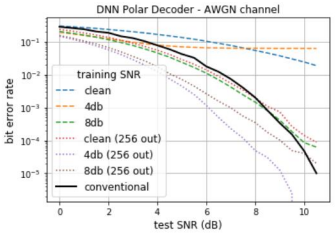

Figure 4: BER curve of DNN-receiver trained and tested under AWGN channels.

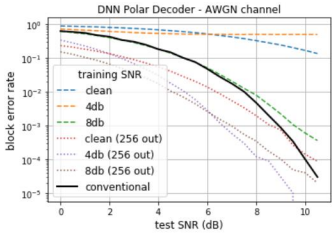

Figure 5: BLER curve of DNN-receiver trained and tested under AWGN channels.

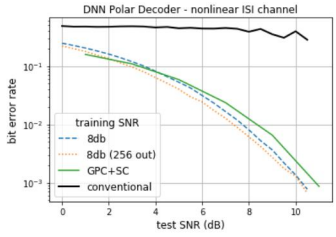

Figure 6: BER curve of DNN-receiver trained and tested under nonlinear channels.

As shown in the figures, different training SNR leads to models with very different performance, and there exists an intermediate value (8dB) of training SNR that achieves the best overall performance. (Besides the curves shown in the figures, we also trained models with training SNR=0dB, 10dB, 12dB. We removed some of the curves to make sure the figures are not too messy). This can be explained considering the regularization effect of the noise: by adding proper amount of noise, the

model is forced to learn more robust decoding schemes, but if the noise level is too high, the noise may occlude the underlaying structure of the code.

Comparing to the conventional receiver suggested in (8), the best DNN-receiver with 8 output units performs comparably well to the conventional one, while the DNN-receivers with 256 output units performs significantly better, especially when comparing the BLER. This makes sense because the DNN-receiver with 256 output is built to minimize the BLER (It becomes more clear if we take a close look at the curve with legend "clean (256 out)": it's BER curve is close to the conventional one, but the BLER is much lower). This observation indicates a difference in error distribution. However, the gap between the two DNN structures are not yet clear.

Fig. 6 shows the BER curves of the models trained under nonlinear channel. The conventional curve is obtained by implementing conventional channel equalization to compensate for channel distortion, and then decode as if the channel is AWGN. The other more advanced benchmark (GPC+SC) uses Gaussian process for classification (GPC) for equalization and successive cancellation algorithm for decoding (9). As we can see, when the channel is non-AWGN, the conventional method fails. The DNN-receiver performs comparably to the GPC+SC method in low SNR regime, and outperforms it when SNR exceeds 5dB. This indicates that when the noise level is too high, it may worth some delicate effort to compensate for the channel and suppress the noise before decoding.

5.2 Second Scenario

We compare our results for this scenario with the widely used convolutional code. Note that the convolutional code has maximum likelihood decoding algorithm, and it is proved to have error correcting capability as long as the errors are below certain threshold. In particular, we compare the performance of the joint DNN transmitter and receiver with the convolutional encoder + maximum likelihood decoder (Viterbi decoder), as well as the convolutional encoder + DNN receiver. It can be observed from the BER curves in Fig. 7 and BLER curves in Fig. 8 that the DNN receiver itself can achieve near optimal performance given the convolutional encoder. The joint DNN transmitter and receiver is able to get lower error rate in the low SNR regime, meaning that our model is more robust for highly noisy channel.

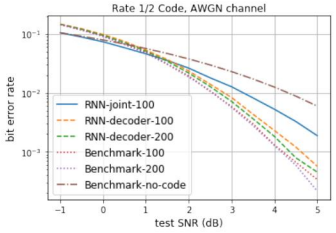

Figure 7: BER curve of joint DNN transmitter and receiver.

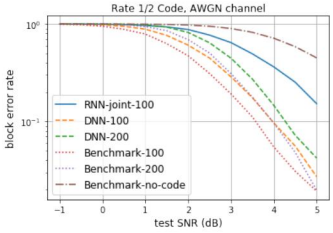

Figure 8: BLER curve of joint DNN transmitter and receiver.

6 Conclusion and Future Work

In this project, we investigated the possibility of using deep learning to design novel channel coding/decoding schemes for wireless communications. Results showed that deep learning can help improve the existing channel coding scheme in the case when channel model is unknown or imprecise, or when no optimal decoding algorithm is developed. Also, we successfully trained a channel coding scheme from scratch, which outperformed widely used convolutional code in highly noisy channel. This reveals the potential of deep learning methods to further optimization of current wireless transmission schemes.

Moving forward, there are multiple open problems unsolved in this project. For example, for the unconstrained length code case, we observed that it is much easier to reduce the BER to a satisfying level, but it is very hard to get competitive BLER comparing to existing schemes. It would be worthy to try more loss functions and architectures to see how to effectively reduce BLER if time allows. Also, given the universal approximation capability of neural network, we expected the DNNs to perform as well as the conventional methods, which is not true in the second case. It would be interesting to dig deeper for the reasons.

7 Contributions

This project is done by Yun Liao. The topic is suggested by Prof. Andrea Goldsmith.

8 Codes

Please refer to `https://github.com/yunl140/CS230_channel_coding`.

In the code, we modified a commpy library (`https://github.com/veeresht/CommPy`), which contains basic communication modules. Keras is used to build neural networks.

References

[1] E. Nachmani, E. Marciano, L. Lugosch, W. Gross, D. Burshtein, and Y. Be'ery, "Deep learning methods for improved decoding of linear codes," *IEEE Journal of Selected Topics in Signal Processing,* vol. 12, no. 1, pp. 119-131, Feb. 2018.

[2] F. Liang, C. Shen, and F. Wu, "An iterative BP-CNN architecture for channel decoding," *IEEE Journal of Selected Topics in Signal Processing,* vol. 12, no. 1, pp. 144-159, Feb. 2018.

[3] T. Gruber, S. Cammerer, J. Hoydis, and S. Brink, "On deep learning-based channel decoding," in *IEEE Annual Conference on Information Sciences and Systems*, Baltimore, MD, Mar. 2017.

[4] N. Farsad, and A. Goldsmith, "Sliding bidirectional recurrent neural networks for sequence detection in communication systems," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Calgary, AB, Canada, Apr. 2018.

[5] H. Ye, and G. Y. Li, "Initial results on deep learning for joint channel equalization and decoding," in *IEEE Vehicular Technology Conference (VTC-Fall)*, Toronto, ON, Canada, Sept. 2017.

[6] T. J. O'Shea, K. Karra, T. C. Clancy, "Learning to communicate: channel auto-encoders domain specific regularizers and attention", in *IEEE Int. Symp. Signal Process. Inf. Technol.*, Limassol, Cyprus, Dec. 2016.

[7] F. A. Aoudia and J. Hoydis, "End-to-end learning of communications systems without a channel model," `https://arxiv.org/abs/1804.02276`.

[8] E. Arikan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Trans. Inform. Theory*, vol. 55, no. 7, pp. 3051-3073, 2009.

[9] P. Olmos, J. Murillo-Fuentes, and F. Perez-Cruz, "Joint nonlinear channel equalization and soft LDPC decoding with Gaussian processes," *IEEE Trans. Signal Process.*, vol. 58, no. 3, pp. 1183-1192, Mar. 2010.