LEARN Codes: Inventing low-latency codes via recurrent neural networks

Yihan Jiang
ECE Department
University of Washington
Seattle, United States
yij021@uw.edu

Hyeji Kim
Samsung AI Center Cambridge
Cambridge, United Kingdom
hkim1505@gmail.com

Himanshu Asnani
ECE Department
University of Washington
Seattle, United States
asnani@uw.edu

Sreeram Kannan
ECE Department
University of Washington
Seattle, United States
ksreeram@ee.washington.edu

Sewoong Oh
ISE Department
University of Illinois at Urbana Champaign
Illinois, United States
swoh@illinois.edu

Pramod Viswanath
ECE Department
University of Illinois at Urbana Champaign
Illinois, United States
pramodv@illinois.edu

Abstract—**Designing channel codes under low latency constraints is one of the most demanding requirements in 5G standards. However, sharp characterizations of the performances of traditional codes are only available in the large block lengths limit. Code designs are guided by those asymptotic analyses and require large block lengths and long latency to achieve the desired error rate. Furthermore, when the codes designed for one channel (e.g. Additive White Gaussian Noise (AWGN) channel) are used for another (e.g. non-AWGN channels), heuristics are necessary to achieve any non trivial performance - thereby severely lacking in robustness as well as adaptivity.**

Obtained by jointly designing recurrent neural network (RNN) based encoder and decoder, we propose an end-to-end learned neural code which outperforms canonical convolutional code under block settings. With this gained experience of designing a novel neural block code, we propose a new class of codes under low latency constraint - *Low-latency Efficient Adaptive Robust Neural (LEARN) codes*, which outperform the state-of-the-art low latency codes as well as exhibit robustness and adaptivity properties. LEARN codes show the potential of designing new versatile and universal codes for future communications via tools of modern deep learning coupled with communication engineering insights.

Index Terms—**Channel Coding, Low Latency, Communications, Deep Learning.**

I. Introduction

Channel coding has emerged as an impactful field of research for the modern information age. Since its inception in [1], powered by the mathematical insight of information theory and principles of modern engineering, several capacity-achieving codes such as Polar, Turbo and LDPC codes [2] [3] [4] have come close to Shannon limit with large block lengths under Additive White Gaussian Noise (AWGN) channels. These then have been successfully adopted and applied in LTE and 5G data plane [5]. As 5G is under intensive development, designing codes that have features such as *low latency*, *robustness*, and *adaptivity* has become increasingly important.

A. Motivation

Ultra-Reliable Low Latency Communication (URLLC) code [6] requires minimal delay constraints, thereby enabling scenarios such as vehicular communication, virtual reality, and remote surgery. While speaking of low-latency requirements, it is instructive to observe that there is an interplay of three different types of delays: processing delay, propagation delay, and structural delay. Processing and propagation delays are affected mostly by computing resources and varying environment [7]. Low latency channel coding, as is the case in this paper, aims to improve the structural delay caused by encoder and/or decoder. Encoder structural delay refers to the delay between receiving the information bit and sending it out by the encoder. Decoder structural delay refers to the delay between receiving the bits from the channel and decoding the corresponding bits. Traditional AWGN capacity-achieving codes such as LDPC and Turbo codes with small block lengths show poor performance for URLLC requirement [7], [8]. There has also been recent interest in developing finite-block information theory to understand bounds on reliability of codes at small to medium block length regime [9].

We note that latency translates directly to block-length when using a block code, however, when using a convolutional code, latency is given by the decoding window length. Thus there is an inherent difference between block codes and convolutional codes when considering latency. Since the latter incorporates locality in encoding, they can also be locally decoded. While convolutional codes with small constraint length are not capacity achieving, it is possible that they can be optimal under the low-latency constraint. Indeed, this possibility was raised by [7], who showed that convolutional codes beat a known converse on the performance of block codes. In this work, we develop further on this hypothesis and show that we can invent codes similar to convolutional codes that beat the performance of all known codes in the low-latency regime. While convolutional codes are state-of-the-art in the low latency regime, in the moderate latency regime, Extended Bose-Chaudhuri-Hocquenghem Code (eBCH) is shown to perform well [10].

In addition, low latency constraint channel coding must take channel effects into account under non-AWGN settings, as pilot bits used for accurate channel estimation increase latency [7]. This calls for incorporating robustness and adaptivity, both, as desired features for URLLC codes. A practical channel coding system requires heuristics to compensate for channel effects, which leads to sub-optimality when there exists model mismatch [11]. Robustness refers to the ability to perform with acceptable degradation without retraining when model mismatches. Adaptivity refers to the ability to learn to adapt to different channel model with retraining. In general, channels without clean mathematical analysis lack the theory of an optimal communication algorithm, thus relying on suboptimal heuristics [12]. In short, current channel coding schemes fail to deliver under the challenges of low latency, robustness, and adaptivity.

B. Deep Learning inspired Channel Coding : Prior Art

In the past decade, advances in *deep learning* have greatly benefitted several fields in engineering such as computer vision, natural language processing as well as gaming technology [13].

This has generated recent excitement in applying deep learning methods to communication system design [16] [17]. Deep learning methods have been typically successful in settings where there is significant model-deficit [18], i.e., the observed data cannot be well described by a clean mathematical model. Thus, many of the initial proposals applying deep learning to communication systems have also focused on this regime, where there is model uncertainty due to lack of say channel knowledge [19], [20]. In order to develop codes for the AWGN channel under low-latency constraints, there is no model deficit, since the channel is well-defined mathematically and quite simple to describe. However, the main challenge is that optimal codes and decoding algorithms are not known - we term this regime as algorithm-deficit. In this regime, there has been very little progress in applying deep learning to communication system design. Indeed, there is no known code beating state-of-the-art codes in canonical channels. We construct the first state-of-the-art code for the AWGN channel in this paper (in the low-latency regime).

Broadly, the following have been two categories of works that apply deep learning to communications: (a) designing a neural network decoder (or neural decoder in short) for a given canonical encoder such as LDPC or turbo codes; (b) jointly designing both the neural network encoder and decoder, referred to as Channel Autoencoder (Channel AE) design [16] (as illustrated in Figure 1).

Neural decoder shows promising performance by mimicking and modifying existing optimal decoding algorithm. Learnable Belief Propagation (BP) decoders for BCH and High-Density Parity-Check (HDPC) code have been proposed in [22] and [23]. Polar decoding via neural BP is proposed in [24] and [25]. As mimicking learnable Tanner graphs requires a fully connected neural network, generalizing to longer block lengths is prohibitive. Capacity-achieving performance for Turbo Code under AWGN channel is achieved via Recurrent Neural Network (RNN) for arbitrary block lengths [26].

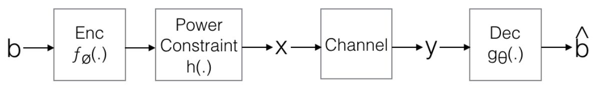


Fig. 1. Channel AE Block Diagram

The joint design of neural code (encoders) and decoders via Channel Autoencoder (AE), which is relevant to the problem under consideration in this paper, has witnessed scanty progress. Deep Autoencoder has been successfully applied for various problems such as dimensionality reduction, representation learning, and graph generation [15]. However, Channel AE significantly differs from the typical deep autoencoder models in the following two aspects, thereby making it highly challenging:

1) The number of possible message bits b grows exponentially with respect to block length, thus Channel AE must generalize to unseen messages with capacity-restricted encoder and decoder [24].
2) Channel model adds noise between the encoder and the decoder, and encoder needs to satisfy power constraint - thus requiring a high robustness in the code.

For the Channel AE training, [16] and [17] introduce learning tricks emphasizing both channel coding and modulations. Learning Channel AE without channel gradient is shown in [28]. Modulation gain is reported in [29]. Beyond AWGN and fading channels, [30] extended RNN to design code for the feedback channel, which outperforms existing state-of-the-art code. Extending Channel AE to MIMO settings is reported in [21]. Despite the successes, existing research on Channel AE under canonical channels is currently restricted to very short block-lengths (for example, achieving the same performance as a (7,4) Hamming code).

Furthermore, existing works do not focus on the low-latency, robustness, and adaptivity requirements. In this paper, we ask the fundamental question:

Can we improve Channel AE design to construct new codes that comply with low-latency requirements?

We answer this in affirmative as shown in the next subsection.

C. Our Contribution

The primary goal is to design a low latency code under extremely low latency requirements. As pointed out earlier, convolutional codes beat block codes under the low latency regime [7]. RNN is a constrained neural structure with a natural connection to convolutional codes, since the encoded symbol has a locality of memory and is most strongly influenced by the recent past of the input bits. Furthermore, RNN based codes have shown natural generalization across different block lengths in prior work [26] [24]. We demonstrate that with carefully designed learnable structure using Bidirectional RNN (Bi-RNN) for both encoder and decoder, as well as training methodology developed specifically for the Channel AE model, our Bi-RNN based neural code outperforms convolutional code.

We then propose *Low-latency Efficient Adaptive Robust Neural (LEARN) code*, which applies learnable RNN structures for both the encoder and the decoder with an additional low-latency constraint. LEARN achieves state-of-the-art performance under extremely low latency constraints. To the best of our knowledge, this is the first work that achieves an end-to-end design for a neural code achieving state-of-the-art performance on the AWGN channel (in any regime). In summary, the contributions of the paper are:

1. *Beating convolutional codes:* We propose the Bi-RNN network structure and a tailored learning methodology for Channel AE that can beat canonical convolutional codes. The proposed training methodology results in smoother training dynamic and better generalization, which are required to beat convolutional codes (Section II).
2. *State-of-the-art performance in low latency settings*: We design **LEARN code** for low latency requirements with specific network design. LEARN code results in beating SOTA performance in extremely low latency requirements (Section III).
3. *Robustness and Adaptivity*: When the channel conditions are varying, LEARN codes show robustness (ability to work well under unseen channel) as well as adaptivity (adapt to new channel with few training symbols), showing an order of magnitude improvement in reliability over state-of-the-art codes (Section III).

II. Designing Neural Code to Beat Convolutional Code

In order to beat convolutional code using a learnt neural-network code, the network architecture as well as training methodology need to be carefully crafted. In this section, we provide guidelines for designing learning architecture as well as training methods that are key to achieve a high reliability of neural codes. Finally, we demonstrate that the aforementioned codes outperform Convolutional Code under block coding setup with solely end-to-end learning.

A. Network Structure Design

Recent research on Channel AE does not show coding gain for even moderate block lengths [16] [24] with fully connected neural network, even with nearly unlimited training examples. We argue that a Recurrent Neural Network (RNN) architecture is more suitable as a deep learning structure for Channel AE.

Introduction of RNN

As illustrated in Figure 2 (left), RNN is defined as a general function $f(.)$ such that $(y_t, h_t) = f(x_t, h_{t-1})$ at time t, where x_t is the input, y_t is the output, h_t is the state sent to the next time slot and h_{t-1} is the state from the last time slot. RNN can only emulate causal sequential functions. Indeed, it is known that an RNN can capture a general family of measurable functions from the input time-sequence to the output time-sequence [27]. Illustrated in Figure 2 (right), Bidirectional RNN (Bi-RNN) combines one forward and backward RNN and can infer current state by evaluating through both past and future. Bi-RNN is defined as $(y_t, h_t^f, h_t^b) = f(x_t, h_{t-1}^f, h_{t+1}^b)$, where h_t^f and h_t^b stands for the state at time t for forward and backward RNNs [13].

RNN is a restricted structure which shares parameters between different time slots across the whole block, which makes it naturally generalizable to a longer block length. Moreover, RNN can be considered as an overparameterized non-linear convolutional code for both encoder and decoder, since convolutional code encoder can be represented by causal RNN and BCJR forward-backward algorithm can be emulated by Bi-RNN [26]. There are several parametric functions $f(.)$ for RNN, such as vanilla RNN, GRU, or LSTM. Vanilla RNN is known to be hard to train due to diminishing gradients, in this paper we use Gated Recurrent Unit (GRU) as our primary network structure [32]. In this paper we use the terms GRU and RNN interchangeably.

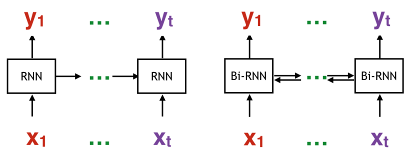


Fig. 2. Basic RNN structure (left), Bi-RNN (right)

RNN based Encoder and Decoder Design

Our empirical results comparing different Channel AE structures in Figure 3 shows that for longer block length, RNN outperforms simply applying fully connected neural network (FCNN) for Channel AE (both encoder and decoder). RNN in Figure 3 refers to using Bi-RNN for both encoder and decoder. The repetition code and extended Hamming code performances are shown as a reference for both short and long block length cases.

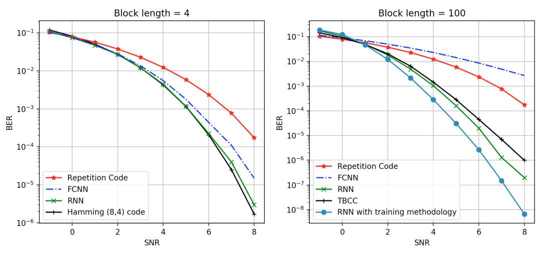


Fig. 3. Encoder and Decoder Structure Design. Channel AE for block length 4 (left), and block length 100 (right)

Figure 3 (left) shows that for short block length (4), the performance of FCNN and RNN are close to each other, since for short block length enumerating all possible code is not prohibitive. On the other hand, for longer block length (100), Figure 3 (right) shows that in using FCNN for longer block length, the Bit Error Rate (BER) is even worse than repeat code, which shows failure in generalization. RNN outperforms FCNN due to its generalization via parameter sharing and adaptive learnable dependency length. Thus in this paper, we use RNN for both the encoder and the decoder to gain generalization across block length. We can also see from Figure 3 that RNN with a tailored training methodology outperforms simply applying RNN or FCNN for Channel AE; we illustrate those training methodology used in Section B below.

Power Constraint Module
The output of the RNN encoder can take arbitrary values and does not necessarily satisfy the power constraint. To impose the power constraint, we use a power constraint layer followed by the RNN. As shown in Figure 1, before transmitting codewords, we force the output of power constraint module to generate codewords s.t. $Ex_i^2 = 1$ and $Ex_i = 0$, which is referred as bit-wise normalization. During training phase bit-wise normalization applies $x_i = \frac{x_i - mean(x_i)}{std(x_i)}$ where $mean(x_i)$ and $std(x_i)$ refer to the mean and standard deviation (std) given the training mini-batch. During testing phase, the mean and std are precomputed by passing through many samples to ensure that the estimation is accurate.

B. Training Methodology

The following training methods result in a faster learning trajectory and better generalizations with the learnable structure discussed above.

1) Train with large batch size.
2) Train encoder and decoder separately. Train encoder once, and then train decoder 5 times.
3) Adding minimum distance regularizer on encoder.
4) The decoder has more capacity (parameters) than the encoder.

Some of the training methods are not common in deep learning due to the unique structure of Channel AE. We briefly discuss three of the most significant enhancements to the training methodology here.

Large Batch Size
Empirically the batch size for Channel AE has to be larger than 1000 to generalize well. Large batch size gives a better gradient estimations. Also with a large batch size, power constraint module offers a better estimation of mean and std [33], which makes the output of the encoder less noisy, thus the decoder can be optimized accordingly.

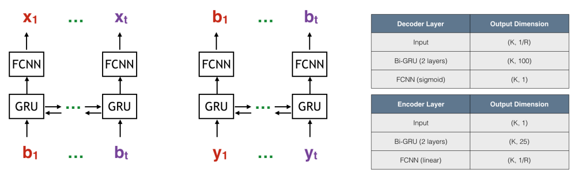
Separately Train Encoder and Decoder
Training encoder and decoder jointly with end-to-end back-propagation leads to saddle points. We argue that training Channel AE entails separately training encoder and decoder [28]. The accurate gradient of the encoder can be computed when the decoder is optimal for a given encoder. Thus after training encoder, training decoder until convergence will make the gradient of encoder more trustable. However, at every step training decoder till convergence is computationally expensive. Empirically we find training encoder once and training decoder 5 times shows the best performance.

Adding Minimum Distance Regularizer
Naively optimizing Channel AE results in paired local optimum: a bad encoder and a bad decoder can be locked in a saddle-point. Adding regularization to loss is a common method to escape local optima [13]. Coding theory suggests that maximizing minimum distance between all possible input messages [5] improves coding performance. However since the number of all possible messages increases exponentially with respect to code block length, computing loss with maximized minimum distance for long block code becomes prohibitive.

Exploiting the locality inherent to RNN codes, we introduce a different loss term solely for the encoder which we refer to as the partial minimum code distance regularizer. Partial minimum code distance $dist(u_s)$ is the minimum distance among all possible codewords with length s, which contains 2^s codes. Computing pairwise distance requires $O(\binom{2^s}{2})$ computations. Partial minimum code distance is a compromise over computation, which still guarantees large minimum distance under small block length s, while hoping the minimum distance on longer block length would still be large. Empirically partial minimum code distance improves block code performance significantly.

C. Design to beat Convolutional Code

Applying the network architecture guidelines and the training methodology improvements proposed hitherto, we design neural code with Bi-GRU for both encoder and decoder as shown in Figure 4. The hyperparameters are shown in Figure 5.



Decoder Layer	Output Dimension
Input	(K, 1/R)
Bi-GRU (2 layers)	(K, 100)
FCNN (sigmoid)	(K, 1)

Encoder Layer	Output Dimension
Input	(K, 1)
Bi-GRU (2 layers)	(K, 25)
FCNN (linear)	(K, 1/R)

Fig. 4. RNN-based Channel AE encoder (left), decoder (middle), and Network Structures (right)

Encoder	2-layer Bi-GRU with 25 units
Decoder	2-layer Bi-GRU with 100 units
Power constraint	bit-wise normalization
Batch size	1000
Learning rate	0.001, decay by 10 when saturate
Num epoch	240
Block length	100
Batch per epoch	100
Optimizer	Adam
Loss	Binary Cross Entropy (BCE)
Min Dist Regularizer	0.0
Train SNR at rate 1/2	mixture of 0 to 8dB
Train SNR at rate 1/3	mixture of -1 to 2dB
Train SNR at rate 1/4	mixture of -2 to 2dB
Train method	train encoder once decoder 5 times
Min Distance Regularizer	0.001 ($s = 10$)

Fig. 5. RNN-based Channel AE hyperparameters

D. *Performance of RNN-based Channel AE: AWGN Setting*

We design the block code under short block lengths and compare the performance with Tail-biting Convolutional Code (TBCC), as TBCC bridges the gap between run-length code and block code. The BER performance in AWGN channel under various code rates is shown in Figure 6. The TBCC BER curve is generated by the best generator function from Figure 7 (left). Figure 6 shows that RNN-based Channel AE outperforms all convolutional codes under memory size 7. RNN-based Channel AE empirically shows the advantage of jointly optimizing encoder and decoder over AWGN channel.

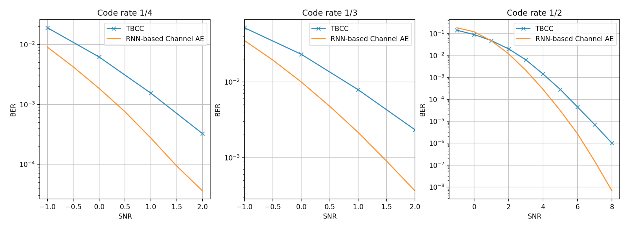


Fig. 6. Comparing RNN-based Channel AE with Conv Code on rate 1/4 (left), 1/3 (middle), 1/2 (right). Block length = 100

III. Design Low Latency Codes: LEARN

Designing codes for low latency constraints is challenging as many existing block codes require inevitably long block lengths. In this section, to address this challenge, we propose a novel RNN based encoder and decoder architecture that satisfies low latency constraint, which we call LEARN. We show that the LEARN code is (a) significantly more reliable than convolutional codes, which are state-of-the-art under extreme low latency constraint [7], and (b) more robust and adaptive for various channels beyond AWGN channels. In the following, we first define the latency and review the literature under the low latency setting.

A. *Low Latency Convolutional Code*

Formally, decoder structural delay D is understood in the following setting: to send message b_t at time t, the causal encoder sends code x_t, and the decoder has to decode b_t as soon as it received y_{t+D}. The decoder structural delay D is the number of bits that the decoder can look ahead to decode. The convolutional code has 0 encoder delay due to its causal encoder, and the decoder delay is controlled by the optimal Viterbi Decoder [31] with a decoding window of length w which only uses the last w future branches in the trellis to compute the current output. For code rate $R = \frac{k}{n}$ convolutional code, the structural decoder delay is $D = k - 1 + kw$ [8]. When information bit is $k = 1$, the structural decoder delay is $D = w$. Convolutional code is the state-of-the-art code under extreme low latency where $D \leq 50$ [7].

In this paper, we confine our scope at investigating extreme low latency with no encoder delay under extremely low structural decoder delay $D = 1$ to $D = 10$ with code rates 1/2, 1/3, and 1/4. The benchmark we are using is convolutional code with variable memory length. Under unbounded block length setting, longer memory results in better performance, however under low latency constraint longer memory might not necessarily mean better performance since the decoding window is shorter [7]. Hence we test for all memory lengths under 7 to get the state-of-the-art performance of the Recursive Systematic Convolutional (RSC) Code, whose generating functions are shown in Figure 7 (left), with convolutional code encoder shown in Figure 7 (right). The decoder is Viterbi Decoder with decoding window $w = D$.

	$R=\frac{1}{2}$		$R=\frac{1}{3}$			$R=\frac{1}{4}$			
m	g_{11}	g_{12}	g_{11}	g_{12}	g_{13}	g_{11}	g_{12}	g_{13}	g_{14}
1	2	3	1	3	3	1	1	3	3
2	5	7	5	7	7	5	7	7	7
3	15	17	13	15	17	13	15	15	17
4	23	35	25	33	37	25	27	33	37
5	53	75	47	53	75	53	67	71	75
6	133	171	133	145	175	135	135	147	163
7	247	371	225	331	367	237	275	313	357

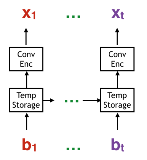
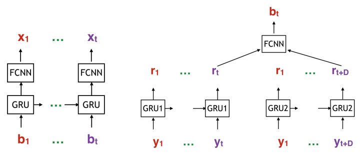
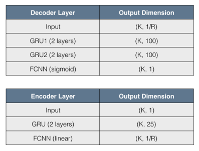


Fig. 7. Convolutional Code generator matrix (left) and Encoder (right)

B. *LEARN network structure*

Following the network design proposed in previous section, we propose a novel RNN based neural network architecture for the LEARN (both the encoder and the decoder) that satisfies the low latency constraint. Our proposed LEARN encoder is illustrated in Figure 8 (left). The causal neural encoder is a causal RNN with two layers of GRU added to Fully Connected Neural Network (FCNN). The neural structure ensures that the optimal temporal storage can be learnt and extended to non-linear regime. The power constraint module is bit-wise normalization as described in previous section.





Decoder Layer	Output Dimension
Input	(K, 1/R)
GRU1 (2 layers)	(K, 100)
GRU2 (2 layers)	(K, 100)
FCNN (sigmoid)	(K, 1)

Encoder Layer	Output Dimension
Input	(K, 1)
GRU (2 layers)	(K, 25)
FCNN (linear)	(K, 1/R)

Fig. 8. LEARN encoder (left), LEARN decoder (middle), and Network Structures (right)

Applying Bi-RNN decoder for low latency code requires to compute lookahead instances for each received information bit, which is computationally expensive in both time and memory. To improve efficiency, the LEARN decoder uses two GRU structures instead of Bi-RNN structures. The LEARN decoder has two GRUs: one GRU runs till the current time slot, another GRU runs further for D steps, then the outputs of two GRUs are summarized by a FCNN. LEARN decoder ensures that all the information bits satisfying delay constraint can be utilized with the forward pass only. When decoding a received signal, each GRU just needs to process one step ahead, which results in decoding computation complexity $O(1)$. Viterbi and BCJR low latency decoders need to go through the trellis and backtrack to the desired position, which requires going forward one step and backward with delay constraints steps, thus resulting in $O(D)$ computation for decoding each bit. Although GRU has a large computational constant due to the complexity of the neural network, with emerging AI chips the computation time is expected to diminish [34]. The hyper-parameters of LEARN are different from block code settings and is shown in Figure 9.

Encoder	One 2-layer GRU with 25 units
Decoder	Two 2-layer GRU with 100 units
Num epoch	120
Min Dist Regularizer	0.0

Fig. 9. LEARN hyperparameters

C. Performance of LEARN: AWGN Setting

Figure 10 shows the BER of LEARN code and SOTA RSC code from varying memory lengths in Figure 7 (left) for rates 1/2, 1/3, and 1/4 as a function of SNR under latency constraints $D = 1$ and $D = 10$. As we can see from the figure, for rates 1/3 and 1/4 under AWGN channel, LEARN code under extreme delay ($D = 1$ to $D = 10$) shows better performance in Bit Error Rate (BER) as compared to the SOTA RSC code from varying memory lengths from Figure 7 (left). LEARN outperforms all RSC code listed in Figure 7 (left) with $D \leq 10$ with code rates 1/3 and 1/4, demonstrating a very promising application of neural code under low latency constraint.

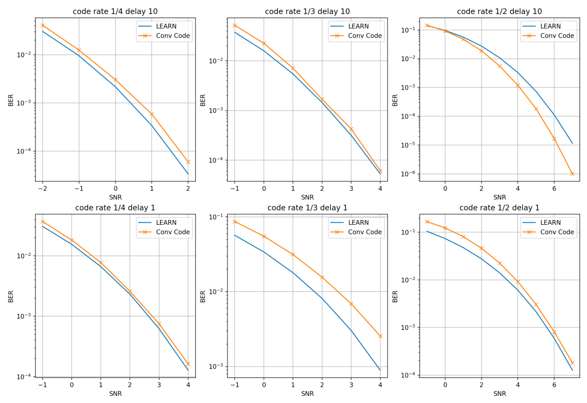


Fig. 10. BER curves comparing Low latency Convolutional Code vs LEARN under AWGN channel with rate 1/4, 1/3, and 1/2.

For higher code rates such as $R = \frac{1}{2}$ and $D \geq 5$, LEARN shows comparable performance to convolutional codes but degrades at high SNR. We expect further improvements can be made via improved structure design and hyperparameter optimization, especially at higher rates.

D. Robustness and Adaptivity

We test the robustness and adaptivity of LEARN on three families of channels:

1) AWGN channel: $y = x + z$, where $z \sim N(0, \sigma^2)$.
2) Additive T-distribution Noise (ATN) channel: $y = x+z$, where $x \sim T(v, \sigma^2)$, for $v = 3, 5$. This noise is a model for heavy-tailed distributions.
3) Radar channel: $y = x+w+z$, where $z \sim N(0, \sigma_1^2)$ and $w \sim N(0, \sigma_2^2)$, w.p. p. (Assume $\sigma_1 \ll \sigma_2$). This noise model shows up when there is bursty interference, for example when a Radar interferes with LTE [12], [35]

Robustness
Robustness shows when LEARN is trained for AWGN channel, the test performance with no re-training on a different channel (ATN and Radar) should not degrade much. Most existing codes are designed under AWGN since AWGN has a clean mathematical abstraction, and AWGN is the worst case noise under a given power constraint [1]. When both the encoder and the decoder are not aware of the non-AWGN channel, the BER performance degrades. Robustness ensures both the encoder and the decoder perform well under channel mismatch, which is a typical use case for low latency scheme when channel estimation and compensation are not accurate [5].

Adaptivity
Adaptivity allows LEARN to learn a decoding algorithm from enough data even under no clean mathematical model [26]. We train LEARN under ATN and Radar channels with the same hyperparameter as shown in Figure 9 and with the same amount of training data to ensure LEARN converges. With both encoder and decoder learnable, two cases of adaptivity are tested. First is the decoder adaptivity, where encoder is fixed and decoder can be further trained. Second is the full adaptivity on both encoder and decoder. In our findings, encoder adaptivity doesn't show any further advantage, and is thus omitted.

Performance
The performance of LEARN with reference to robustness and adaptivity is shown in Figure 11 with three different settings: (1) delay $D = 10$, code rate $R = 1/2$, with ATN($\nu = 3$) channel; (2) delay $D = 2$, code rate $R = 1/3$, with ATN($\nu = 3$) channel; (3) delay $D = 10$, code rate $R = 1/2$, with Radar($p = 0.05$, $\sigma_2 = 5.0$) channel. As shown in Figure 11 (left), with ATN ($\nu = 3$) that has a heavy-tail noise, LEARN with robustness outperforms convolutional code. Adaptivity with both encoder and decoder performs best, and is better than only decoder is adaptive. By utilizing the degree of freedom of designing encoder and decoder, neural designed coding scheme can match canonical convolutional codes with Channel State Information at Receiver (CSIR) at low code rate ($R = 1/2$), and outperform convolutional codes with CSIR at high code rate ($R = 1/3$).

As for Figure 11 (middle) ATN ($\nu = 3$) channel with code rate $R = 1/3$ and Figure 11 (right) Radar ($\sigma_2 = 5.0$) channel with code rate $R = 1/4$, the same trend holds. Note that under Radar channel, we apply the heuristic proposed in [12]. We observe that LEARN with full adaptation gives an order-of-magnitude improvement in reliability over the convolutiona code heuristic [12]. The experiment shows that by jointly designing both encoder and decoder, LEARN can adapt to a broad family of channels. LEARN offers an end-to-end low latency coding design method which can be applied to any statistical channels and ensure good performance.

IV. Conclusion

In this paper, we have demonstrated the power of neural network based architectures in achieving state-of-the-art performance for simultaneous code and decoder design. In the long block length case, we showed that our learnt codes can significantly outperform convolutional codes. However, in

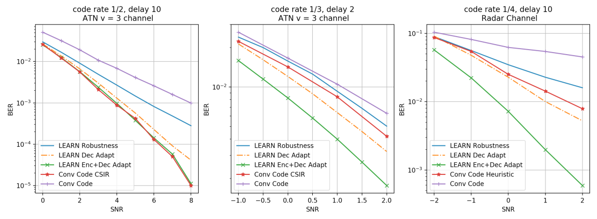


Fig. 11. LEARN robustness and adaptivity in ATN ($\nu = 3$, $R = 1/2$, $D = 10$)(left), ATN ($\nu = 3$, $R = 1/3$, $D = 2$)(middle), and Radar ($p = 0.05$, $\sigma_2 = 5.0$, $R = 1/4$, $D = 10$)(right) channels

order to beat state-of-the-art codes such as Turbo or LDPC codes, we require additional mechansims such as interleaving to introduce long-term dependence. This promises to be a fruitful direction for future exploration.

In the low-latency regime, we showed that we can achieve state-of-the-art performance with LEARN codes. Furthermore, we showed that LEARN codes beat existing codes by an order of magnitude in reliability when there is channel mismatch. Our present design is restricted to extreme low latency, however, with additional mechanisms for introducing longer term dependence [36], [37], it is possible to extend these designs to cover a larger range of delays. This is another interesting direction for future work.

We refer the reader to our full paper [38] for further details on training methodology as well as interpretation of learnt codes.

References

[1] Shannon, C.E. “A mathematical theory of communication.” Bell system technical journal 27.3 (1948): 379-423.

[2] Arikan, Erdal. “A performance comparison of polar codes and Reed-Muller codes.” IEEE Communications Letters 12.6 (2008).

[3] Berrou, Claude, Alain Glavieux, and Punya Thitimajshima. “Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1.” Communications, 1993. ICC’93.

[4] MacKay, D. JC, and Radford M. N. “Near Shannon limit performance of low density parity check codes.” Electronics letters 32.18 (1996).

[5] Richardson, Tom, and Ruediger Urbanke. Modern coding theory. Cambridge university press, 2008.

[6] Sybis, Michal, et al. “Channel coding for ultra-reliable low-latency communication in 5G systems.” Vehicular Technology Conference (VTC-Fall), IEEE 84th. IEEE, 2016.

[7] Rachinger, Christoph, Johannes B. Huber, and Ralf R. Mller. “Comparison of convolutional and block codes for low structural delay.” IEEE Transactions on Communications 63.12 (2015): 4629-4638.

[8] Maiya, Shashank V., Daniel J. Costello, and Thomas E. Fuja. “Low latency coding: Convolutional codes vs. LDPC codes.” IEEE Transactions on Communications 60.5 (2012): 1215-1225.

[9] Polyanskiy, Yury, H. Vincent Poor, and Sergio Verd. ”Channel coding rate in the finite blocklength regime.” IEEE Trans. on Info. Theory 56.5 (2010): 2307-2359.

[10] Shirvanimoghaddam, Mahyar, et al. “Short Block-length Codes for Ultra-Reliable Low-Latency Communications.” arXiv:1802.09166 (2018).

[11] Li, Junyi, Xinzhou Wu, and Rajiv Laroia. OFDMA mobile broadband communications: A systems approach. Cambridge University Press, 2013.

[12] Safavi-Naeini, Hossein-Ali, et al. “Impact and mitigation of narrow-band radar interference in down-link LTE.” 2015 IEEE ICC, 2015.

[13] Goodfellow, Ian, et al. Deep learning. Vol. 1. Cambridge: MIT press, 2016.

[14] Han, Song, Huizi Mao, and William J. Dally. “Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding.” arXiv:1510.00149 (2015).

[15] Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. “Reducing the dimensionality of data with neural networks.” science 313.5786 (2006): 504-507.

[16] O’Shea, Timothy J., Kiran Karra, and T. Charles Clancy. “Learning to communicate: Channel auto-encoders, domain specific regularizers, and attention.” Signal Processing and Information Technology (ISSPIT), 2016 IEEE International Symposium on. IEEE, 2016.

[17] O’Shea, Timothy, and Jakob Hoydis. “An introduction to deep learning for the physical layer.” IEEE Transactions on Cognitive Communications and Networking 3.4 (2017): 563-575.

[18] Kannan, Sreeram and Kim, Hyeji and Oh, Sewoong, “Deep learning and Information theory: An emerging interface.” Tutorial at IEEE International Symposium on Information Theory (ISIT), June 2018.

[19] Farsad, Nariman and Goldsmith, Andrea, “Neural Network Detection of Data Sequences in Communication Systems.” IEEE Transactions on Signal Processing, Jan 2018.

[20] Dörner, Sebastian and Cammerer, Sebastian and Hoydis, Jakob and Brink, Stephan ten, “Deep learning-based communication over the air.” arXiv preprint arXiv:1707.03384

[21] O’Shea, Timothy J., Tugba Erpek, and T. Charles Clancy. “Deep learning based MIMO communications.” arXiv preprint arXiv:1707.07980 (2017).

[22] Nachmani, Eliya, Yair Be’ery, and David Burshtein. “Learning to decode linear codes using deep learning.” Communication, Control, and Computing (Allerton), 2016 54th Annual Allerton Conference on. IEEE, 2016.

[23] Nachmani, Eliya, et al. “Deep learning methods for improved decoding of linear codes.” IEEE Journal of Selected Topics in Signal Processing 12.1 (2018): 119-131.

[24] Gruber, Tobias, et al. “On deep learning-based channel decoding.” Information Sciences and Systems (CISS), 2017 51st Annual Conference on. IEEE, 2017.

[25] Cammerer, Sebastian, et al. “Scaling deep learning-based decoding of polar codes via partitioning.” GLOBECOM 2017-2017 IEEE Global Communications Conference. IEEE, 2017.

[26] Kim, Hyeji and Jiang, Yihan and Rana, Ranvir and Kannan, Sreeram and Oh, Sewoong and Viswanath, Pramod. ”Communication Algorithms via Deep Learning” Sixth International Conference on Learning Representations (ICLR).

[27] Hammer, Barbara, “On the approximation capability of recurrent neural networks.” Neurocomputing Vol. 31, pp. 107–123, 2010.

[28] Aoudia, Fayal Ait, and Jakob Hoydis. “End-to-End Learning of Communications Systems Without a Channel Model.” arXiv preprint arXiv:1804.02276 (2018).

[29] Felix, Alexander, et al. “OFDM-Autoencoder for End-to-End Learning of Communications Systems.” arXiv preprint arXiv:1803.05815 (2018).

[30] Kim, Hyeji, et al. “Deepcode: Feedback codes via deep learning.” arXiv preprint arXiv:1807.00801 (2018).

[31] Viterbi, Andrew. “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm.” IEEE transactions on Information Theory 13.2 (1967): 260-269.

[32] Chung, Junyoung, et al. “Empirical evaluation of gated recurrent neural networks on sequence modeling.” arXiv:1412.3555 (2014).

[33] Ioffe, Sergey, and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift.” arXiv preprint arXiv:1502.03167 (2015).

[34] Ovtcharov, Kalin, et al. “Accelerating deep convolutional neural networks using specialized hardware.” MSR Whitepaper 2.11 (2015).

[35] Sanders, Geoffrey A, “Effects of radar interference on LTE (FDD) eNodeB and UE receiver performance in the 3.5 GHz band.” US Department of Commerce, National Telecommunications and Information Administration, 2014.

[36] Sutskever, Ilya and Vinyals, Oriol and Le, Quoc V, “Sequence to sequence learning with neural networks.” NIPS 2014.

[37] Jaderberg, Max and Simonyan, Karen and Zisserman, Andrew and others, “Spatial transformer networks.” NIPS 2015.

[38] Y. Jiang, H. Kim, H. Asani, S. Kannan, S. Oh, and P. Viswanath, “LEARN Codes: Inventing low-latency codes via recurrent neural networks.” to appear on arXiv. https://infotheory.ece.uw.edu/papers/learn_2018.pdf