LEARN Codes: Inventing Low-Latency Codes via Recurrent Neural Networks
Yihan Jiang, Hyeji Kim, Himanshu Asnani, Sreeram Kannan, Sewoong Oh, and Pramod Viswanath

Abstract—Designing channel codes under low-latency constraints is one of the most demanding requirements in 5G standards. However, a sharp characterization of the performance of traditional codes is available only in the large block-length limit. Guided by such asymptotic analysis, code designs require large block lengths as well as latency to achieve the desired error rate. Tail-biting convolutional codes and other recent state-of-the-art short block codes, while promising reduced latency, are neither robust to channel-mismatch nor adaptive to varying channel conditions. 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 non-trivial performance. In this paper, we first propose an end-to-end learned neural code, obtained by jointly designing a Recurrent Neural Network (RNN) based encoder and decoder. This code outperforms canonical convolutional code under block settings. We then leverage this experience to propose a new class of codes under low-latency constraints, which we call Low-latency Efficient Adaptive Robust Neural (LEARN) codes. These codes outperform state-of-the-art low-latency codes and exhibit robustness and adaptivity properties. LEARN codes show the potential to design 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, robustness, adaptivity.

I. INTRODUCTION

RELIABLE channel codes have had a huge impact on communications in the modern information age. Since its inception in [1] that was powered by the mathematical insights 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 the Shannon limit when operated at large block lengths under Additive White Gaussian Noise (AWGN) channels. These codes were successfully adopted and applied in LTE and 5G data planes [5]. Since 5G is under intensive development, designing codes that have features such as low latency, robustness, and adaptivity has become increasingly important.

A. Motivation

An Ultra-Reliable Low Latency Communication (URLLC) code [6] requires minimal delay constraints, thereby enabling scenarios such as vehicular communication, virtual reality, and remote surgery. When considering low-latency requirements, it is instructive to observe the interplay of three different types of delays: processing delays, propagation delays, and structural delays. Processing and propagation delays are affected mostly by computing resources and varying environments [7]. Low-latency channel coding, the focus of this paper, aims to improve the structural delay caused by the encoder and/or decoder. Encoder structural delay refers to the delay between the encoder’s receiving the information bit and sending it out. Decoder structural delay refers to the delay between the decoder’s receiving and decoding bits from the channel. Traditional AWGN capacity-achieving codes, such as LDPC and Turbo codes with small block lengths, show poor performance for URLLC requirements [7], [11]. There has also been recent interest in establishing theoretical limits and bounds on the reliability of codes at small to medium block lengths [12].

We note that latency is proportional to block length when using a block code; the decoder waits until it receives the entire (noisy) codeword to start the decoding. On the other hand, 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 lengths are not capacity achieving, they can possibly be optimal under the low-latency constraint. Indeed, this possibility was raised by [7] and
observed in [8], [9], [10]. In this work, we advance this hypothesis by showing that we can invent codes similar to convolutional codes that outperform handcrafted codes in the low-latency regime. While convolutional codes are state-of-the-art in this regime, in the moderate latency regime, Extended Bose-Chaudhuri-Hocquenghem codes (eBCH) also perform well [13].

In addition, low-latency constraint channel coding must take channel effects into account in fading channels because pilot bits used for accurate channel estimation increase latency [7]. This calls for incorporating both robustness and adaptivity as desired features for URLLC codes. Robustness refers to the ability to perform with acceptable degradation without retraining when model mismatches occur; adaptivity refers to the ability to learn to adapt to different channel models with retraining. Current and traditional channel coding systems require heuristics to compensate for channel effects, which leads to sub-optimality for model mismatches [14]. In general, channels without clean mathematical analysis lack the theory of an optimal communication algorithm and thus rely on sub-optimal heuristics [15]. In short, current channel coding schemes fail to deliver under the challenges of low latency, robustness, and adaptivity.

B. Channel Coding Inspired by Deep Learning: Prior Art

On the past decade, advances in deep learning (DL) have greatly benefited engineering fields such as computer vision, natural language processing, and gaming technology [16]. This has generated recent excitement about applying DL methods to communication system design [19], [20]. These methods have typically been successful in settings where there is a significant model-deficiency, i.e., the observed data cannot be well described by a clean mathematical model. Thus, many initial proposals applying DL to communication systems have also focused on problems where there is model uncertainty due to the lack of, say, channel knowledge [21], [22]. In developing codes for the AWGN channel under low-latency constraints, there is no model deficiency since the channel is well-defined mathematically and simple to describe. However, the main challenge is that optimal codes and decoding algorithms are not known; we refer to this as algorithm deficiency.

For algorithm-deficit problems, the following two categories of work apply deep learning to communication systems: (1) designing a neural network decoder (also known as neural decoder) for a given canonical encoder such as LDPC or Turbo codes, and (2) jointly designing both the neural network encoder and decoder, referred to as a Channel Autoencoder (also known as Channel AE) [19] (as illustrated in Figure 1).

Neural decoders show promising performance by mimicking and modifying existing optimal decoding algorithms. Learnable Belief Propagation (BP) decoders for BCH and High-Density Parity-Check (HDPC) codes have been proposed in [24] and [25]. Polar decoding via neural BP is proposed in [26] and [27]. Since mimicking learnable Tanner graphs requires a fully connected neural network, generalizing to longer block lengths is prohibitive. Capacity-achieving performance for Turbo code under an AWGN channel is achieved via Recurrent Neural Networks (RNNs) for arbitrary block lengths [28]. The joint design of neural code (encoders) and decoders via a Channel AE, relevant to the problem under consideration in this paper, has witnessed scant progress. Deep autoencoders have been successfully applied for various problems, such as dimensionality reduction, representation learning, and graph generation [18]. However, Channel AE significantly differs from typical deep autoencoder models in the following two aspects, making its design and training highly challenging:

1) The number of possible message bits $b$ grows exponentially with respect to the block length. Hence, Channel AE must generalize to unseen messages with capacity-restricted encoders and decoders [26].

2) Channel models add noise between the encoder and decoder, and the encoder needs to satisfy power constraints, thus requiring high robustness in the code.

For Channel AE training, [19] and [20] introduce learning tricks emphasizing both channel coding and modulations. Learning Channel AE without a channel gradient is shown in [32]. Modulation gain is reported in [33]. Beyond AWGN and fading channels, [34] extended RNN to design a code for the feedback channel, which outperforms existing state-of-the-art codes [29], [30]. Extending Channel AE to MIMO settings is reported in [23]. 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 rate 4/7 Hamming code with 4 information bits). Furthermore, existing works do not focus on the low-latency, robustness, and adaptivity requirements.

With this backdrop, this paper poses the following fundamental question: Can we improve the Channel AE design to construct new codes that comply with low-latency requirements?

We answer this in affirmative, as described next.

C. Our Contribution

Our primary goal is to design a low-latency code under extremely low structural-latency requirements. As noted, convolutional codes outperform block codes given low structural latency [7], [8], [9], [10]. An 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 a natural generalization across different block lengths in prior work [26], [28]. With a carefully designed learnable structure that uses Bidirectional RNN (Bi-RNN) for both encoder and decoder, as well as a novel training methodology developed specifically for the Channel AE model, we demonstrate that 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 to both the encoder and decoder with an additional low-latency constraint. LEARN improves performance under extremely low latency constraints. Ours is the first work we know of that creates an end-to-end design for a neural code that improves performance on the AWGN channel (in any regime). In summary, the contributions of this paper include:

1) Outperforming convolutional codes: We propose a bi-directional RNN network structure and a tailored learning methodology for Channel AE that outperform canonical convolutional codes. The proposed training methodology results in smoother training dynamics and better generalization (Section II).

2) Improving performance in low-latency settings: We design LEARN code for low-latency requirements with specific network designs. LEARN code improves performance in extremely low-latency settings (Section III).

3) Showing robustness and adaptivity: When channel conditions are varying, LEARN codes demonstrate robustness (ability to work well under unseen channels) and adaptivity (ability to adapt to a new channel with few training symbols), showing an order of magnitude improvement in reliability compared to existing state-of-the-art codes (Section III).

4) Interpreting the neural code: We provide interpretations to aid in the fundamental understanding of why the jointly trained code works better than canonical codes (Section IV).

II. DESIGNING NEURAL CODES THAT OUTPERFORM CONVOLUTIONAL CODES

The reliability of neural codes relies heavily on two factors: (1) a network architecture, and (2) a training methodology. In this section, we provide guidelines for these two factors and demonstrate that neural codes designed and trained using our guidelines have superior reliability compared to convolutional codes which are the best performing codes in the literature.

A. Network Architecture

The performance (coding gain) of traditional channel codes improves with block length. Recent research on the Channel AE model does not show coding gain for even moderate block lengths [19], [26] with fully connected neural networks, even with nearly unlimited training examples. We argue that a Recurrent Neural Network (RNN) architecture is a more suitable DL structure for Channel AE. For a brief introduction to RNN and its variants – such as Bidirectional RNN (Bi-RNN), Gated Recurrent Unit (GRU), or Long Short Term Memory (LSTM) – please refer to Appendix A in the supplementary material. In this paper, we use the terms GRU and RNN interchangeably.

RNN-based Encoder and Decoder Design: Our empirical results comparing different Channel AE structures (Figure 2) show that for long enough block lengths, RNN outperforms a Fully Connected Neural Network (FCNN) for Channel AE. The FCNN curve in Figure 2 refers to using FCNN for both encoder and decoder. RNN in Figure 2 refers to using Bi-RNN for both encoder and decoder. The training steps are kept the same for a fair comparison. The repetition code and extended Hamming code performances are shown as a reference for both short and long block lengths.

Figure 2 (left) shows that for short block lengths (4), the performance of FCNN and RNN are close to each other since enumerating all possible code is not prohibitive. On the other hand, for a longer block length (100), Figure 2 (right) shows that in using FCNN, the Bit Error Rate (BER) is even worse than repetition codes, which shows a failure in generalization; RNN outperforms FCNN due to its generalization via parameter sharing and adaptive learnable dependency length. Hence, in this paper, we model the encoder and decoder as RNNs in order to gain generalization across block lengths. Figure 2 also shows that RNNs with tailored training methodologies outperform simply applying RNN or FCNN for Channel AE; we illustrate this training methodology in Section II-B.

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 after the RNN encoding layer, as shown in Figure 1. Power normalization enforces that the output code has unit power by normalizing the output as \( E[x^2] = 1 \). More detail is in the Appendix in the supplementary material.

B. Training Methodology

We find in this paper that the following training methods result in a faster learning trajectory and better generalization with the learnable structure discussed above.

- Training with a large batch size
- Using Binary Crossentropy (BCE) loss
- Training encoder and decoder separately
- Adding a minimum distance regularizer on encoder.
- Using the Adam optimizer
- Adding more capacity (parameters) to the decoder than the encoder

Some of the training methods are not common in deep learning due to the unique structure of Channel AE. Appendix C in the supplementary material shows the empirical evidence, where we reason about the better performance of the above training methods.

C. Performance of RNN-Based Channel AE: AWGN Setting

Applying the network architecture guidelines and the training methodology improvements thus far proposed, we design a neural code with Bi-GRU for both encoder and decoder, as shown in Figure 3. The hyperparameters are shown in Figure 4.

Tail-biting Convolutional Code (TBCC) has proven to be the state of the art under a short block length regime [7], [8], [9], [10]. We compare the performance of TBCC with RNN-based channel code on block code settings. The BER performance in the AWGN channel under various code rates is shown in Figure 5. The TBCC BER curve is generated by convolutional code with constraint length up to \( m = 7 \) by the best generator function from Figure 7, with traceback length equals \( 5(m + 1) \); we measure the BERs for all TBCCs in Figure 7 empirically using CommPy simulator [52]...
Fig. 2. Channel AE performance on code rate 1/2, where block length (number of information bits) is 4 (left) and 100 (right). TBCC shown on the right is $m = 2$, with $g_{11} = 5$, $g_{12} = 7$ with feedback 7.

Fig. 3. RNN-based Channel AE encoder (top left), decoder (top right), and network structures (bottom).

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 hyperparameters.

Note that the RNN-based Channel AE code is continuous; the performance gain is from both coding gain and high-order modulation gain, as shown in Figure 5 right; the performance gap at a higher SNR is larger. Binarized code leads to a fairer comparison; however, binarizing with the sign function is not differentiable. Bypassing non-differentiability using a straight-through estimator (STE) [50] degrades performance in channel coding [51]. Binarizing code and comparing to high order modulation are deferred to future research.

D. Performance of RNN-Based Channel AE: Non-AWGN Setting

We test the robustness and adaptivity of RNN-based Channel AE 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 heavy-tailed noise distribution models bursty interference.
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. $p = 0.05$. (Assume $\sigma_1 \ll \sigma_2$). This noise model appears when there is bursty interference, for example, when Radar interferes with LTE [15], [39].

**Robustness:** Robustness refers to the property that when RNN-based Channel AE is trained for the AWGN channel, the test performance with no re-training on a different channel (ATN and Radar) should not degrade significantly. Most existing codes are designed under AWGN since it has a clean mathematical abstraction, and AWGN is the worst case noise under a given power constraint [1]. When both the encoder and decoder are unaware of the non-AWGN channel statistics, the BER performance degrades. Robustness ensures both the encoder and decoder perform well under channel mismatch, which is a typical use case for the low-latency scheme when channel estimation and corresponding receiver calibration are not accurate [5].

**Adaptivity:** Adaptivity allows RNN-based Channel AE to learn a decoding algorithm from sufficient data (includes retraining) even when the channel statistics does not admit a simple mathematical characterization [28]. We train RNN-based Channel AE under ATN and Radar channels with and plot the best performing one. Figure 5 shows that RNN-based Channel AE outperforms all convolutional codes up to constraint length 7. RNN-based Channel AE empirically demonstrates the advantage of jointly optimizing encoder and decoder over the AWGN channel.
the same hyperparameters shown in Figure 4 and the same amount of training data to ensure that the RNN-based Channel AE converges. With both learnable encoder and decoder, two cases of adaptivity are tested. First is decoder adaptivity, where the encoder is fixed and the decoder can be further trained. Second is the full adaptivity of both encoder and decoder. In our findings, encoder adaptivity shows no further advantage and is thus omitted.

We now evaluate RNN-based Channel AE for robustness and adaptivity on ATN and Radar channels. The BER performance is shown in Figure 6. Note that under non-AWGN channels, RNN-based Channel AE trained on the AWGN channel outperforms the convolutional code with the Viterbi decoder. It also shows more robust decoding ability for channel mismatching compared to the best convolutional code. As shown in Figure 6, RNN-based Channel AE with decoder-only adaptivity improves over the RNN-based Channel AE robust decoder, while RNN-based Channel AE with full adaptivity with both trainable encoder and decoder shows the best performance.

The fully adapted RNN-based Channel AE outperforms the convolutional code even with CSIR, which utilizes the log-likelihood of T-distribution noise. Thus, designing jointly by utilizing the encoder and decoder further optimizes the code under given channels. Even when the underlying mathematical model is far from a cleaner abstraction, RNN-based Channel AE can learn the underlying functional code via self-supervised back-propagation.

RNN-based Channel AE is the first neural code to our knowledge that outperforms existing canonical codes under the AWGN channel coding setting, which opens a new field of constructing efficient neural codes under canonical settings. Furthermore, RNN-based Channel AE can be applied to channels when, especially, closed form mathematical analysis cannot be performed.

III. Designing Low Latency Neural Codes: LEARN

Designing codes for low latency constraints is challenging since many existing block codes require inevitably long block lengths. To address this challenge, we now propose a new RNN-based encoder and decoder architecture that satisfies a low latency constraint, which we call LEARN. While our encoder and decoder architectures are based on RNNs, we introduce a new block in the decoder architecture so that the decoder can satisfy the extreme low-latency constraint. We show that LEARN code is: (1) significantly more reliable than convolutional codes, which are state-of-the-art under extreme low-latency constraints [7], and (2) more robust and adaptive for various channels beyond AWGN channels. In the following, we first define latency and review the literature for 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 receives $y_{t+D}$. 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 [35] with a decoding window of length $w$, which uses only the next $w$ future branches in the trellis to compute the current output. For code rate $R = k - 1/kw$ [11]. We use code rate $k = 1$ with $n = 2, 3, 4$; the structural decoder delay is $D = w$. Convolutional code is state-of-the-art code under extreme low latency, where $D \leq 50$ [7].

In this paper, we confine our scope to investigating extreme low latency with no encoder delay under 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 a convolutional code with variable memory length. Under an unbounded block length setting, longer memory improves performance; however, under a low latency constraint, longer memory may not necessarily mean better performance since the decoding window is short [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 (top), with the convolutional code encoder shown in Figure 7 (bottom). The decoder is Viterbi with a decoding window $w = D$.

B. LEARN Network Architecture

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

Applying the Bi-RNN decoder for low latency code requires the computation of 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. It has two GRUs: one GRU runs till the current time slot, and another GRU runs further for $D$ steps; the outputs of the two GRUs are then summarized by a FCNN. The LEARN decoder ensures that all the information bits satisfying the delay constraint can be utilized with the forward pass only. When decoding a received signal, each GRU needs to process only 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, resulting in $O(D)$ computation for decoding each bit. Although GRU has a large computational constant due to the complexity of the neural network, the computation time is expected to diminish [38] with emerging AI chips. The hyperparameters of LEARN differ from block code settings. To summarize the differences: (1) the encoder and decoder use GRU instead of Bi-GRU, (2) the number of training epochs is reduced to 120, and (3) we do not use a partial minimum distance regularizer.

C. Performance of LEARN: AWGN Setting

Figure 9 shows the BER of a LEARN code and state-of-the-art RSC codes with varying memory lengths as illustrated in Figure 7 (top) for rates 1/2, 1/3, and 1/4 as a function of SNR under latency constraints $D = 1$ and $D = 10$. As the figure shows, for rates 1/3 and 1/4 under the AWGN channel, LEARN code under extreme delay ($D = 1$ to $D = 10$) shows better performance in Bit Error Rate (BER) compared to the state-of-the-art RSC codes from varying memory lengths in Figure 7 (top). LEARN outperforms all RSC codes listed in Figure 7 (top) with $D \leq 10$ with code rates 1/3 and 1/4, demonstrating a promising application of neural code under the low latency constraint.

For higher code rates (such as $R = 1/4$ and $D \geq 5$), LEARN shows comparable performance to convolutional codes but...
D. Robustness and Adaptivity

The performance of LEARN with reference to robustness and adaptivity is shown in Figure 10 for 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; and (3) delay $D = 10$, code rate $R = 1/2$, with the Radar ($p = 0.05$, $\sigma_2 = 5.0$) channel. As shown in Figure 10 (left), with ATN ($\nu = 3$), which has a heavy-tail noise, LEARN with robustness outperforms convolutional codes. An improved adaptivity is achieved when both the encoder and the decoder are trained, compared to when only decoder is trained. By exploring a larger space of codes, neural designed coding schemes can match canonical convolutional codes with Channel State Information at Receiver (CSIR) at a low code rate ($R = 1/2$) and outperform convolutional codes with CSIR at a high code rate ($R = 1/3$).

As for Figure 10 (middle) ATN ($\nu = 3$) channel with code rate $R = 1/3$ and Figure 10 (right) Radar ($\sigma_2 = 5.0$) channel with code rate $R = 1/4$, the same trend holds. Note that under the Radar channel, we apply the heuristic proposed in [15]. We observe that LEARN with full adaptation gives an order-of-magnitude improvement in reliability over the convolutional code heuristic [15]. This 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 that can be applied to any statistical channels and ensure good performance.

IV. INTERPRETABILITY OF DEEP MODELS

The promising performance of LEARN and the RNN-based Channel AE leads to a question: how do we interpret what the encoder and decoder have learned? Answering this can inspire future research as well as help us find caveats and limitations. In this section, we present interpretation analysis via local perturbation for LEARN and RNN-based Channel AE encoders and decoders.

A. Encoder Interpretability

The significant recurrent length of RNNs is a recurrent capacity indicator that helps to interpret neural encoder and decoders. The RNN encoder’s significant recurrent length is defined, at time $t$, as how long a sequence the input $u_t$ can impact as RNN operates recurrently. Assume two input sequences, $u_1 = u_{1,1}, \ldots, u_{1,t}, \ldots, u_{1,T}$ and $u_2 = u_{2,1}, \ldots, u_{2,t}, \ldots, u_{2,T}$, where only $u_{1,t}$ and $u_{2,t}$ differ. Taking a batch of $u_1$ and $u_2$ as input for the RNN encoder, we compare the output absolute difference along the whole block between $x_1 = f(u_1)$ and $x_2 = f(u_2)$ to investigate how long the input flip at time $t$ can affect. To investigate LEARN’s RNN encoder, we flip only the first bit (position 0) of $u_1$ and $u_2$. The code position refers to the block bit positions, and the y-axis shows the averaged difference between two different sequences. Figure 11 (top left) shows that for an extremely short delay $D = 1$, the encoder’s significant recurrent length is short. The effect of the current bit diminishes after 2 bits. As the delay constraint increases, the encoder’s significant recurrent length increases, accordingly. The LEARN encoder learns to encode locally to optimize under the low-latency constraint.
For RNN-based Channel AE with a Bi-RNN encoder, the block length is 100, and the flip is applied at the middle 50th bit position. Figure 11 (top right) shows the encoder trained under the AWGN and ATN channels. The encoder trained on ATN shows a longer significant dependency length. ATN is a heavy-tail noise distribution. To alleviate the effect of extreme values, increasing the dependency helps. Note that even the longest significant recurrent length is only backward 10 steps and forward 16 steps; thus, the GRU encoder actually did not learn a very long recurrent dependency. AWGN capacity-achieving codes have some inbuilt mechanisms to improve long-term dependency. For example, the Turbo encoder uses an interleaver to improve the long-term dependency [3]. Improving the significant recurrent length via a more learnable structure design is an interesting future research direction.

**B. Decoder Interpretability**

The decoder’s significant recurrent length can illustrate how it copes with different constraints and channels. Assume two noiseless coded sequences, $y_1 = y_{1,1}, \ldots, y_{1,t}, \ldots, y_{1,T}$ and $y_2 = y_{2,1}, \ldots, y_{2,t}, \ldots, y_{2,T}$, $y_1$, and $y_2$ equals $y_1$ other than at time $t$, where $y_{1,t} = y_{2,t} + p$, where $p$ is the large deterministic pulse noise; $p = 5.0$ for our experiment. We compare the output absolute difference along the whole block between $\hat{u}_1 = g(y_1)$ and $u_2 = g(y_2)$ to investigate how long the pulse noise can affect. For the LEARN decoder, we inject a pulse noise at the starting position. Figure 11 (bottom left) shows that for all delay cases, the noise most significantly affected the position equal to the delay constraint. This shows that the LEARN decoder learns to coordinate with the causal LEARN encoder. Since $D = 1$, the maximized decoder difference along the block is at position 1; when $D = 10$, the maximized decoder difference along the block is at position 10. Other code bits have a less significant but non-zero decoder difference.

The LEARN decoder’s significant recurrent length implies that it not only learns to utilize the causal encoder’s immediate output, but it also utilizes outputs in other time slots to help decoding. Note that the maximized significant recurrent length is approximately twice the delay constraint; after less than approximately $2D$, the impact diminishes. The LEARN decoder learns to decode locally to optimize under different low latency constraints.

For RNN-based Channel AE with the Bi-RNN encoder, the perturbation is applied at the middle 50th position, still with block length 100. Figure 11 (bottom right) shows the decoder trained under the AWGN and ATN channels. The AWGN-trained decoder is more sensitive to pulse noise with extreme values than the ATN-trained decoder. By reducing the sensitivity for extreme noise, the ATN-trained decoder learns to alleviate the effect of non-Gaussian noise. The RNN-based Channel AE decoder learns to optimize under different channel settings.

**V. Conclusion**

In this paper, we demonstrated the power of neural network-based architectures to achieve state-of-the-art performance for joint encoder and decoder design. We showed that our learned codes significantly outperform convolutional codes in short to medium block lengths. However, in order to outperform state-of-the-art codes such as Turbo or LDPC codes, we require additional mechanisms such as interleaving to introduce long-term dependencies. This promises to be a fruitful direction for future exploration.
In the low-latency regime, we achieved state-of-the-art performance with LEARN codes. Furthermore, LEARN codes beat existing codes by an order of magnitude in reliability where there is a channel mismatch. Our present design is restricted to very low structural delay; however, with additional mechanisms for introducing longer term dependencies [40], [41], we believe that it is possible to extend these designs to cover a larger range of delays. This is another interesting direction for future work. Finally, we have focused only on very short structural delay. Latency depends on other factors as well (e.g., computational complexity). Optimization of neural decoders to reduce other attributes of latency poses an interesting open problem.

ACKNOWLEDGMENT

The authors would like to thank reviewers for the helpful and supportive comments.

REFERENCES

[1] C. E. Shannon, “A mathematical theory of communication,” Bell Syst. Tech. J., vol. 27, no. 3, pp. 379–423, 1948.
[2] E. Arikan, “A performance comparison of polar codes and Reed–Müller codes,” IEEE Commun. Lett., vol. 12, no. 6, pp. 447–449, Jun. 2008.
[3] C. Berrou, A. Glavieux, and P. Thitimajshima, “Near Shannon limit error-correcting coding: Turbo-codes. 1,” in Proc. Commun. ICC, 1993, pp. 248–287.
[4] D. J. C. MacKay and M. N. Radford, “Near Shannon limit performance of low density parity check codes,” Electron. Lett., vol. 32, no. 18, pp. 457–458, Mar. 1997.
[5] T. Richardson and R. Urbanke, Modern Coding Theory. Cambridge, U.K.: Cambridge Univ. Press, 2008.
[6] M. Sybist et al., “Channel coding for ultra-reliable low-latency communication in 5G systems,” in Proc. IEEE 8th Veh. Technol. Conf. (VTC-Fall), 2016, pp. 1–5.
[7] C. Rachinger, J. B. Huber, and R. R. Müller, “Comparison of convolutional and block codes for low structural delay,” IEEE Trans. Commun., vol. 63, no. 12, pp. 4629–4638, Dec. 2015.
[8] G. Liva, L. Gaudio, T. Ninacs, and T. Jerkovic, “Code design for short blocks: A survey,” Oct. 2016. [Online]. Available: arXiv:1610.00873.
[9] G. Liva, G. Durisi, M. Chiani, S. S. Ullah, and S. C. Liew, “Short codes with mismatched channel state information: A case study,” in Proc. IEEE Int. Conf. Wireless Commun. Sys. (ICWCS), Jul. 2017, pp. 1–5.
[10] G. Durisi, T. Koch, and P. Popovski, “Toward massive, ultrareliable, and low-latency wireless communication with short packets,” IEEE Trans. Commun., vol. 104, no. 9, pp. 1711–1726, Aug. 2016.
[11] S. V. Maiya, D. J. Costello, and T. E. Fujii, “Low latency coding: Convolutional codes vs. LDPC codes,” IEEE Trans. Commun., vol. 60, no. 5, pp. 1215–1225, May 2012.
[12] Y. Polyanskiy, H. V. Poor, and S. Verdu, “Channel coding rate in the finite blocklength regime,” IEEE Trans. Inf. Theory, vol. 56, no. 5, pp. 2307–2359, May 2010.
[13] M. Shirvani-Moghadamt et al., “Short block-length codes for ultra-reliable low latency communications,” IEEE Commun. Mag., vol. 57, no. 2, pp. 130–137, Dec. 2018.
[14] J. Li, X. Wu, and R. Laroya, OFDMA Mobile Broadband Communications: A Systems Approach. Cambridge, U.K.: Cambridge Univ. Press, 2013.
[15] H. A. Safavi-Naeini, C. Ghosh, E. Visotsky, R. Ratasuk, and S. Roy, “Impact and mitigation of narrow-band radar interference in downlink LTE,” in Proc. IEEE Int. Conf. Commun. (ICC), Jun. 2015, pp. 2644–2649.
[16] I. J. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA, USA: MIT Press, Nov. 2016.
[17] S. Han, H. Mao, and W. J. Dally, “Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding,” 2015. [Online]. Available: arXiv:1510.00149.
[18] G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” Science, vol. 313, no. 5786, pp. 504–507, 2006.
Yihan Jiang received the B.S. degree from the Beijing Institute of Technology in 2012 and the M.S. degree in electrical and computer engineering from UC San Diego in 2014. He is currently pursuing the Ph.D. degree with the Electrical and Computer Engineering Department, University of Washington, Seattle, WA, USA. His research interests are in the areas of channel coding, information theory, deep learning, and federated learning.

Hyeji Kim received the B.S. degree (Hons.) in electrical engineering from KAIST in 2011, and the M.S. and Ph.D. degrees in electrical engineering from Stanford University in 2013 and 2016, respectively. In 2018, she joined Samsung AI Research Cambridge as a Researcher. She was a Postdoctoral Research Associate with the University of Illinois at Urbana–Champaign. She is a recipient of the Stanford Graduate Fellowship and participant in the Rising Stars at EECS Workshop in 2015.

Himanshu Asnani received the B.Tech. degree from IIT Bombay in 2009 and the M.S. and Ph.D. degrees from the Electrical Engineering Department, Stanford University in 2011 and 2014, respectively, where he was a Stanford Graduate Fellow. He worked briefly as a System Architect with Ericsson Silicon Valley following which he was involved in a couple education startups. He has also held visiting faculty appointments with Stanford University and IIT Bombay. He is currently a Reader (equal to tenure-track Assistant Professor) with the School of Technology and Computer Science, University of Washington, Seattle, where he was a Research Associate. His current research interests include machine learning, causal inference, and information and coding theory. He was a recipient of the 2014 Marconi Society Paul Baran Young Scholar Award, the 2018 Amazon Catalyst Fellow Award, the Best Paper Award at Mobihoc 2009, and was the Finalist for the Student Paper Award at ISIT 2011.

Pramod Viswanath (Fellow, IEEE) received the Ph.D. degree in electrical engineering and computer science from the University of California at Berkeley in 2000. He is a Professor of electrical and computer engineering with the University of Illinois at Urbana-Champaign. He has coauthored with David Tse the text *Fundamentals of Wireless Communication*, which has been used in over 60 institutions worldwide. His current research interests include blockchain technologies from a variety of angles: networking protocols, consensus algorithms, payment channels, distributed coded storage, and incentive designs. He has received the Elihu Jury Award, the Bernard Friedman Prize, the NSF CAREER Award, and the Best Paper Award at the Sigmetrics Conference in 2015. He is a Co-Founder and the CEO of applied protocol research, a startup focused on developing core blockchain technologies.

Sreeram Kannan received the M.S. degree in mathematics and the Ph.D. degree in electrical engineering from the University of Illinois at Urbana- Champaign. He is currently an Assistant Professor with the University of Washington, Seattle. He was a Postdoctoral Scholar with the University of California at Berkeley from 2012 to 2014. His research interests include the applications of information theory and learning to blockchains, computational biology, and wireless networks. He was a recipient of the 2019 UW ECE Outstanding Teaching Award, the 2017 NSF Faculty Early CAREER Award, the 2013 Van Valkenburg Outstanding Dissertation Award from UIUC, the 2010 Qualcomm (CTO) Roberto Padovani Outstanding Intern Award, and the SVC Aiya Medal from the Indian Institute of Science in 2008, and a co-recipient of the 2010 Qualcomm Cognitive Radio Contest First Prize and the Intel India Student Research Contest First Prize in 2006.

Sewoong Oh received the Ph.D. degree from the Department of Electrical Engineering, Stanford University in 2011 under the supervision of A. Montanari. He worked as a Postdoctoral Researcher with the Laboratory for Information and Decision Systems, MIT under the supervision of D. Shah. His research interest is in theoretical machine learning. In 2019, he joined the Paul G. Allen School of Computer Science and Engineering, University of Washington as an Associate Professor. He has been an Assistant Professor with the Department of Industrial and Enterprise Systems Engineering, University of Illinois at Urbana-Champaign since 2012. He was co-awarded the ACM SIGMETRICS Best Paper Award in 2015, the NSF CAREER Award in 2016, the ACM SIGMETRICS Rising Star Award in 2017, and the GOOGLE Faculty Research Award in 2017 and 2020.