Boosting Belief Propagation for LDPC Codes with Deep Convolutional Neural Network Predictors

Xiwen Chen[1], Huayu Li[1], Junsuo Qu[2], and Abolfazl Razi[1]

[1]School of Informatics, Computing and Cyber systems, Northern Arizona University, AZ, USA
[2]School of Automation, Xi'an University of Posts and Telecommunications, Xi'an, China

2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC) | 978-1-7281-9794-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/CCNC49032.2021.9369460

***Abstract*— Conventional channel codes are designed to recover channel errors by adding controlled redundancy to transmit bits; however, the main underlying assumption is that information bits are independent and identically distributed (i.i.d.). Short term and linear temporal correlations are assumed to be exploited by the preceding source encoders. This assumption is flawed in some scenarios since many types of data (e.g. audio samples, video frames, and sensor measurements) exhibit long-term relations and intricate dependencies that are not exploitable by conventional source encoders. Furthermore, sending plain information is still commonplace in wireless networks. Therefore, it is essential to design channel encoders that accommodate these conditions. It is well-known that the underlying hidden patterns can be captured by deep learning methods. This important capability is not yet fully utilized in channel encoder design.**

This work is a primary step towards developing a predictive channel decoder that learns the intricate dependencies within and between data frames using an embedded learning module at the receiver to enhance the bit decoding performance, especially in high-noise regimes. The learning module is integrated with the *belief propagation* algorithm over bipartite graphs appropriate for low-density parity-check (LDPC) codes. The proposed method is universal since no specific correlation model is adopted and the *learning-based* prediction is performed at the bit level. The proposed method is fully implemented at the receiver side, making it compatible with generic LDPC encoders. Our simulations demonstrate the superior performance of the proposed method compared to standard LDPC decoders. For instance, about 1.7 dB gain at the 10^{-4} BER level is achieved when recovering noisy audio files [1].

I. INTRODUCTION

Wireless transmission is an integral part of many technologies including wireless systems, internet of things, ground and aerial robotics, self-driving cars, and many more. In a wireless system, data transmission is often disrupted by undesired effects like channel noise, fading, interference, loss of synchronization, etc. The basic idea of channel coding is adding controlled redundancy to the transmit bits in order to facilitate recovering transmission errors at the destination. Generally, a commonly adopted assumption is that the input bits are independent identically distributed (i.i.d). However, this assumption is unrealistic in practice since data frames (e.g. audio samples, video frames, sensor readings, text, and etc.) can have long-term relations and intricate hidden dependencies. Conventional source encoders that convert information blocks into binary bit-stream are typically designed to capture temporal short-term correlations and fail in recovering intricate dependencies [1], [2]. Furthermore, source encoders are customized for specific data types with prior known correlation models that restrict their use in general cases. On the other hand, several algorithms are developed by the machine learning (ML) community to exploit intricate and hidden patterns among data samples through data-driven learning methods. For instance, predicting missing video frames based on preceding frames is still a hot topic with continued progress in the ML research community. A seminal work is [3], which proposed a convolutional neural network (CNN) to generate video sequences directly in the pixel domain with given information from the start and end frames. Likewise, [4] intends to predict future video frames based on the preceding frames using a decompositional disentangled predictive auto-encoder with the idea of representing video frames with lower-dimensional latent variables. The power of ML algorithms, unfortunately, is not yet fully utilized by the coding research community, and commercial wireless systems still utilize conventional channel encoders such as turbo and LDPC codes [5]. This paper aims at taking initial steps towards filling this gap between the communication and learning perspectives by introducing the concept of *learning-powered* decoders.

The idea is that data bits can be viewed as the observations of an underlying time-series process with long-term and intricate dependencies. Therefore, the learnability of bits from the history of observation is inherited from the temporal correlation of time series as a universal concept regardless of the information source type. In this work, we design a *learning-powered* decoder which integrates a bit-level prediction module with the sum-product algorithm (SPA), a realization of belief propagation algorithm used to recover bits encoded by the low-density parity-check codes (LDPC). More specifically, we integrate the soft information obtained by the SPA algorithm and the embedded predictor.

The proposed solution improves upon the performance of conventional encoders and provides several key advantages including (i) universal decoding and applicability to a wide range of applications (audio, video, text, sensor readings, etc.), since no prior assumptions are made on correlation model, as opposed to the context-specific source encoders, (ii) recovering long-term, non-linear, and intricate dependencies between and within frame bits, and (iii) backward

[1]This material is based upon work supported by the National Science Foundation under Grants No. 1755984 and 2008784

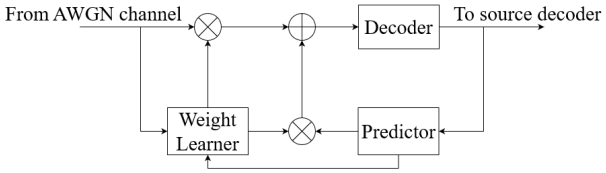

Fig. 1. The overall architecture of the proposed *learning-powered decoder*. The predictor uses the previously decoded messages to predict the current message at bit level. The information is mixed with the decoder soft outputs to yield the final estimates of the information bits. The *weight learner* module assigns proper mixing coefficients based on the fidelity of the prediction module and the channel SNR.

compatibility as it requires no modifications to the encoder.

In this paper, a novel implementation of *learning-based decoding* is proposed by incorporating the predicted information into the conventional LDPC decoding structure with sum-product algorithm (SPA), as shown in Fig.1. The LDPC codes along with turbo codes are among the most commonly used channel coding algorithms that achieve near-Shannon capacity performance with very low error floors. This is why they remained as a top-choice in communication systems during the past several decades. However, like other conventional decoders, its operation is based on the i.i.d. assumption for the input bits and hence does not perform well for recovering predictable information bits. The proposed algorithm addresses this problem by implementing an embedded learning module into the decoder. This approach extends the utility of the LDPC decoder to recovering predictable noisy information and outperforms both the SPA-based channel decoder and the prediction module. The soft information is represented in the Log-likelihood Ratio (LLR) format, which substantially reduces the computation complexity and facilitates building a more interpretable mixed model. The superior performance of our method is verified by experimental results. The proposed mixed model obtains about 1.7 dB gain at the 10^{-4} BER level with a well-designed predictor.

A. Sum-Product Algorithm (SPA)

Here, we review the basics of the SPA algorithm, a popular implementation of the *belief propagation* algorithm, which is the core part of our proposed decoder. This algorithm was first proposed by Judea Pearl in 1982 [6], who formulated it as an exact inference algorithm on trees, which was later extended to polytrees [7]. Essentially, the algorithm is based on *Tanner graphs*, also called bipartite graphs (shown in Fig.2). Tanner graphs are used to describe a family of codes like LDPC codes, where the variable nodes represent the coded bits ,and check nodes represent the algebraic equations implied by the encoder [8]. The sum-product algorithm is the natural result of applying a message passing algorithm over tanner graphs for the iterative update of soft information associated with check nodes and variable nodes [9]. The idea is updating each variable node based on the messages arrived from the relevant check nodes and vice versa. A key property is excluding the impact of the current node on updating itself, and hence the algorithm performs better on short-cycle-free graphs. A message from a check node to a variable node is the product of the associated factors from other variable nodes connected to the check node, marginalized over all variable nodes except the current one, that naturally leads to the sum-product operation.

This basic algorithm is further improved over time by researchers. In [10], the authors proposed a SPA algorithm in the Log-likelihood domain, which substantially reduced the complexity of the algorithm. Some other works investigated the conditions required for the convergence of SPA [11], [12].

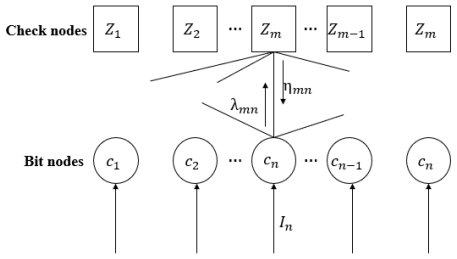

Fig. 2. The illustration of the *message passing* algorithm over *Tanner graphs*, where the check nodes z_m represent the check equations determined by the parity-check matrix, and variable nodes c_n represent the coded bits. Variable nodes are updated based on the messages received from the check nodes, η_{mn}, and vice versa until the convergence conditions are met.

II. PROPOSED METHOD

A. Problem Formulation

Consider a point to point communication system. A Message W within the index set $\{1, 2, ..., M\}$ is encoded by the transmitter to X^n ($f : W \mapsto X^n$) and is sent to the receiver. The receiver receives a noisy version of it $Y^n \sim p(y^n|x^n)$ under a conditional probability $p(y|x)$ determined by the channel model, then estimates the message W by an appropriate decoding rule $\hat{W} = g(Y^n)$.

Let's consider the general case of a discrete memoryless channel (DMC) but with a relaxed i.i.d assumption on the input messages $W_1, W_2, \ldots$, which implies dependencies on the resulting channel symbols $X_1, X_2, \ldots$. First, we define the channel configuration with a uniformly distributed index W on the set $\mathcal{W} = \{1, 2, ..., 2^{nR}\}$. Thus, a Markov chain is formed by $W \rightarrow X^n(W) \rightarrow Y^n \rightarrow \hat{W}$. The transmission error can be described as $P_e^{(n)} = Pr(W \neq \hat{W})$. With the above assumptions, the data symbols with bit-wise dependency are transmitted through the channel, which means that the mutual information $I(X_i; X_j)$ between the transmitter and the receiver is non-negative when $i, j \leq n$ for some n based on the application of interest.Therefore, if the data with noise power $N_0/2$ is transmitted, the optimal coder design problem can be stated as:

$$\begin{aligned} \underset{f,g}{\text{minimize}}\ & P_e^{(n)} \\ s.t.\ & \frac{1}{n}\sum_{i=1}^{n} x_i^2 \leq E_b \\ & y_i \sim \mathcal{N}(x_i, N_0/2) \\ & I(X_i; X_j) > 0 \quad \text{for } i, j \leq n \end{aligned} \tag{1}$$

where f and g describe the encoder and decoder functions, respectively, x_i represents the ith transmitted symbol

according to the utilized codebook, and y_i represents the received symbol, and E_b denotes the power constraint of the transmitter. The constraint of the power is necessary, since otherwise the error can be made arbitrarily small by choosing an infinite subset of inputs far apart [13]. Our goal in this paper is to reduce the error $P_e^{(n)}$ by offering a practical decoding algorithm for LDPC encoders.

B. learning-powered decoder structure

Here, we propose a *learning-powered* decoding algorithm for LDPC codes under AWGN channel that can capture the embedded long-term correlations among information bits to enhance the decoding performance. The prediction approach is based on sequential learning, meaning that the prediction model is refined and improved over time by receiving more frames at the receiver. The L previously decoded messages $X_{k-L+1}, \ldots, X_{k-1}$ are reused in the predictor to predict the next message $\tilde{X}_k$. Then, the bits corresponding to the predicted message are used to initialize the SPA-based decoder to help the decoder converge to a better local optima point.

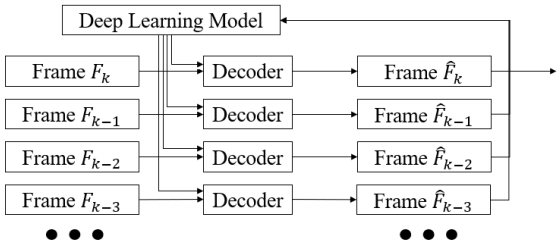

Fig. 3. The predictor captures the long-term correlation using hidden feature layers from previously decoded messages. The model is used to predict the next message and use it to initialize the SPA decoder for enhanced performance.

It is notable that incorporating predictive information should be regulated properly. In general, when the channel SNR is high, the resulting error rate approaches zero, hence we should assign more weight to the decoder results when mixing the information. Our approach is to estimate the reliability of the SPA algorithm and use it to tune the mixing weights by an empirically obtained formula to achieve the highest performance. In this regard, we define

$$\begin{aligned} \eta_1 &= SNR, \\ \eta_2 &= var(|\mathbf{I}_{i,j}|) \\ \alpha &= \frac{\eta_1}{\lambda \eta_2}, \lambda > 0, \end{aligned} \tag{2}$$

where $\mathbf{I}_{i,j}$ is the jth bit channel output information for the ith frames, N is the total number of frames in one segment, and λ is an attenuation factor set in advance. The parameter $\alpha \in [0, 1]$ is a decreasing function of the channel SNR and used to suppress the weights of the decoder.

On the other hand, the predictor performance is somewhat independent of the channel and reflects the predictability of the information, and depends on different factors such as modeling accuracy, bias to the training samples, intrinsic unpredictability and natural randomness of the samples, and the error in the estimation of preceding messages used to predict the upcoming message. This parameter is between 5%-20% in our simulations.

Here, we define the parameter strength $s(F_n)$ as the mean of the absolute values of the predicted data bits in terms of the LLRs of the n most recent frames, namely

$$s(F_n) = \frac{1}{n \ \text{len}(F_i)} \sum_{i=1}^{n} \sum_{j=1}^{\text{len}(F_i)} |LLR_{i,j}^{(po)}|, \tag{3}$$

where $\text{len}(F_i)$ is the number of bits in frame F_i and $LLR_{i,j}^{(po)} = \log\big(p(b_{i,j}=0)/p(b_{i,j}=1)\big)$ represent the LLR of the jth output bit in frame F_i.

The larger values of $s(F_n)$ mean higher confidence about the prediction results. Therefore, the predictor's output should be properly scaled before mixing with the channel outputs to initialize the SPA decoder.

It is observed that over-emphasizing the prediction results by selecting large values of $s(F_n)$ (i.e. $s(F_n) \geq 15$) is not desired, as it strongly biases the SPA decoder and prevents it from converging to the global optima. The under-weighting the prediction results (i.e. $s(F_n) \leq 3$) is not suitable either since it deteriorates the performance of the decoder. To capture this effect into one parameter, we define an auxiliary parameter β as

$$\beta = \begin{cases} 0 & \text{if } s(F_n) \leq 3 \\ 4 & \text{if } 3 \leq s(F_n) \leq 15 \\ 60/s(F_n) & \text{if } s(F_n) \geq 15 \end{cases} \tag{4}$$

to properly weight the prediction information when mixing according to Eq (7). As show in Fig. 2, our decoder operates based on applying SPA [14] to the *tanner graph*. In our notations, η_{mn}^k denotes a message sent from the check node z_m to variable node c_n in the kth iteration. Likewise, λ_{nm}^k represent a message sent by variable node c_n to check node z_m. If I_n is the channel output (in terms of LLR) for variable node c_n, then the belief propagation messages at the kth iterations are defined as:

$$\lambda_{nm}^k = I_n + \sum_{m' \in M_{n,m}} \eta_{m'n}^{k-1}, \tag{5}$$

$$\eta_{mn}^k = 2\tanh^{-1}\Big(\prod_{n' \in N_{m,n}} \tanh\big(\frac{\lambda_{n'm}^k}{2}\big)\Big) \tag{6}$$

Here, N_m is the set of variable nodes that are connected to check node z_m (equivalently, if H is the parity check matrix, it include the non-zero elements of the mth row of H, i.e. $N_m = \{n : H_{mn} = 1\}$). Similarly, M_n is the set of check nodes that are connected to variable node c_n (the non-zero elements of the nth column of H). Also, $N_m \setminus n$ is N_m excluding the variable node v_n ($N_m \setminus n = \{n' : H_{mn'} = 1, n' \neq n\}$), and $M_n \setminus m$ is M_n excluding the check node z_m ($M_n \setminus m = \{m' : H_{mn'} = 1, m' \neq m\}$).

The only exception for Eq (5) is the first iteration, where we initialize the variable nodes with the channel output, i.e. $\lambda_{nm} = I_n$. In the proposed method, we incorporate the predictor results p_n here using the following equation:

$$\lambda_{nm}^1 = \alpha(\beta P_n) + I_n \tag{7}$$

instead of Eq (5), where I_n and P_n are the channel output and the predictor output related to variable node c_n, and α and β are scaling parameters defined in Eq (2,4).

The SPA algorithm continues until the convergence conditions are met or a predefined number of iterations is reached. The convergence point is typically defined by obtaining a fully consistent codeword $\mathbf{c}$ ($\mathbf{c} * H^T = \mathbf{0}$), or when the change in the belief of the variable nodes is negligible. At this point, the final estimates of the bits are obtained by hard-thresholding as

$$\lambda_{nm}^k = I_n + \sum_{m \in M_n} \eta_{mn}^k, \quad c_n = \begin{cases} 1, & \lambda_{nm}^k < 0 \\ 0, & \lambda_{nm}^k > 0. \end{cases} \tag{8}$$

A summary of the proposed algorithm is shown below.

Algorithm 1: *learning-powered* Sum-Product Algorithm (LP-SPA)

Require: The check matrix H; soft information from the demodulator $\mathbf{I}$; the maximum number of iterations $Iter_{max}$, tuning parameter λ, L previously recovered frames $(F_{n-L+1}, \ldots, F_{n-1})$.

Ensure: Decoded message $\mathbf{c}^n = (c_1^n, c_2^n, \ldots)$.

1: Initialize mixing parameters α
2: Estimate the channel SNR
3: Update mixing coefficients η_1, η_2, α using (2)
4: Predict the current frame $\tilde{F}_n$ from previous frames $(F_{n-L+1}, \ldots, F_{n-1})$
5: Estimate the strength of predictor $(s(F_n))$ using (3)
6: Estimate the mixing parameter β using (4)
7: Initialize the SPA decoder using (7)
8: Set iter=1
9: **while** $k \leq Iter_{max}$ **and** $\mathbf{c}_n \cdot H^T \neq 0$ **do**
10: Variable-node update at kth iteration:

$$\lambda_{nm}^k = I_n + \sum_{m' \in M_{n,m}} \eta_{m'n}^{k-1}$$

11: Check-node update at kth iteration:

$$\eta_{mn}^k = 2tanh^{-1}\Big(\prod_{n' \in N_{m,n}} tanh\big(\frac{\lambda_{n'm}^k}{2}\big)\Big)$$

12: k++
13: **end while**
14: **Hard decision:**

$$\lambda_{nm}^k = I_n + \sum_{m \in M_n} \eta_{mn}^k, c_n = \begin{cases} , 1, & \lambda_{nm}^k < 0 \\ 0, & \lambda_{nm}^k > 0 \end{cases}$$

C. Deep Learning Model

As we discussed in Section II, CNN can perform well in capturing hidden features. For instance, [15] used CNN to predict Gray coded audio signal samples. Our strategy in this work is using CNN in a novel way of predicting bit-level information based on the history of the received frames, which makes it applicable to a broad range of applications. More specifically, we predict the value of each data-bit of the current frame using the decoded version of the preceding ten frames. We took necessary considerations so that the predictor can deal with information bits and check bits simultaneously. Additionally, since the prediction process is performed segment by a segment, transfer learning is applied in the modeling phase, meaning that the trainable parameters from the previous segments can be used to initialize the current network (as shown in Fig.4), which substantially improves the training efficiency.

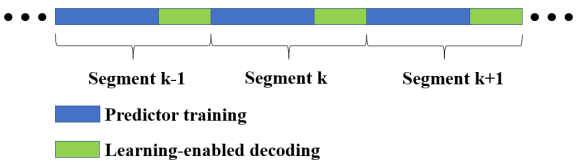

Fig. 4. The illustration of prediction and decoding phases within a segment.

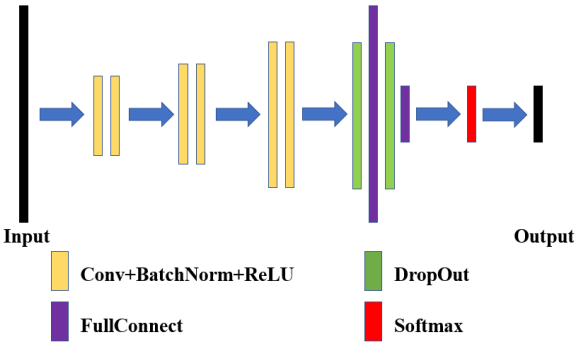

Fig. 5. The utilized neural network architecture includes six stacked 1-dimensional convolutional layers followed by two fully connected layers and a softmax layer. We skip pooling layers due to the relatively small size of the 1-dimensional network to avoid eliminating valuable information.

The deep neural network architecture adopted in this paper is shown in Fig. 5. To promise a near real-time communication, a low-complexity network with a few trainable parameters. The input is a $\mathbb{R}^{1 \times N_1 \times N_2 \times N_3}$ tensor, in which N_1 is the length of a codeword, N_2 is the number of feature map channels, and N_3 is the number of frames. Batch-normalization is used for stable training with enhanced performance. A dropout with a rate of 0.35 is deployed to avoid over-fitting. Softmax activation is used on the output layer to achieve a bit-wise classification. It is notable that because of the predictor implemented in the receiver, noise caused by the channel is unavoidable, hence the prediction accuracy is typically below 90%.

III. EXPERIMENTS

A. Data preparation and experimentation setup

To perform our simulations, we choose segments of two different sample audio files (*Canon in C* [16] and *A Comme Amour*) [17] in our experiments to assess the performance of the decoder for different scenarios. Our strategy is to quantize the audio sample into 8 bits, then

convert them to Gray code, which results in a lower sample (in our case audio sample, or an image pixel) error rate under a given bit error rate [15]. Finally, we encode the bit-stream using an LDPC encoder with a coding rate 1/2. To simplify the information fusion between the predictor and the decoder, a systematic architecture is desirable, therefore we manipulate the parity-check matrix to obtain a systematic generator matrix. In the receiver, we split the received noisy frames into segments, where each segment includes 20000 frames. For each segment, the first 75% of the frames are used in the training phase (50% for training and 25% for validation), and the remaining 25% to produce prediction bits for the *learning-powered* decoder. We calculate the prediction error using the original error-free data) and present the prediction accuracy versus the channel SNR=E_b/N_0. Note that the training is performed on the noisy data and the test is performed with respect to the noise-free data frames, so the use of SNR is relevant. We investigate the convergence of the mixing parameters $\alpha, \beta, s(\mathbf{F}_n)$, as well as the average performance gain of the proposed *learning-powered* decoder under different values for the hyper-parameter λ.

B. Simulation Results

First, we demonstrate that the parameter η_2 defined in Eq (2) converges to its final value for a sufficiently large number of the received frames. This is technically important since η_2 is used to regulate the mixing coefficients. More specifically, we send 16-bit frames with random bits and set different levels of SNRs in the AWGN channel with BPSK modulation, then investigate the value of η_2 versus the number of frames, as shown in Fig. 6. As expected, η_2 converges to a constant value after receiving enough number of frames (about 1000 frames for audio files). This number is relatively low in real-world applications, where extremely long files can be exchanged (e.g., a video streaming scenario). After this period, we achieve a higher gain for using learning-enabled decoding. The parameter η_2 converges to a larger value for higher SNRs, which results in a smaller value for α. This implies the suppression of the prediction results and emphasizing the received bits, a reasonable choice for the high-SNR regime.

Similarly, the behavior of α versus the channel SNR and λ is presented in Fig. 7. Firstly, it is noticeable that α decreases with the channel SNR as intentionally planned to suppress the weight of the prediction results and put more emphasis on the channel outputs when mixing the information to initialize the SPA decoder. Also, this parameter decreases with the tuning parameter λ, which should be tuned based on the system characteristics. For instance, it is a known concept that the performance of LDPC codes improves with the larger block lengths. Therefore it is advantageous to assign higher values for λ for longer blocks to obtain lower values of α in order to put more emphasis on the channel decoder compared to the prediction results.

Next, we analyze the performance of the two predictors. Table I presents the results of the prediction module applied to the two aforementioned test scenarios. The resulting BER

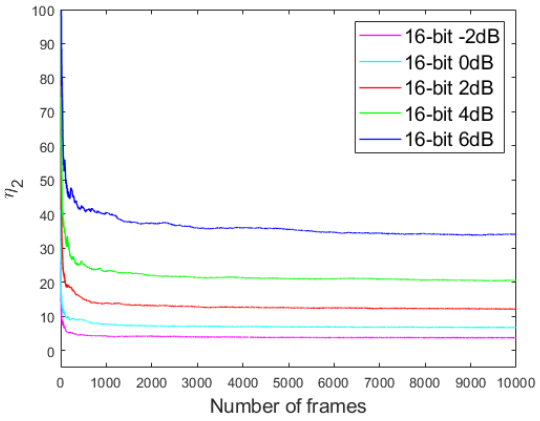

Fig. 6. The convergence of η_2 after receiving frames, at different SNR levels (-2dB, 0dB, 2dB, 4dB, and 6dB).

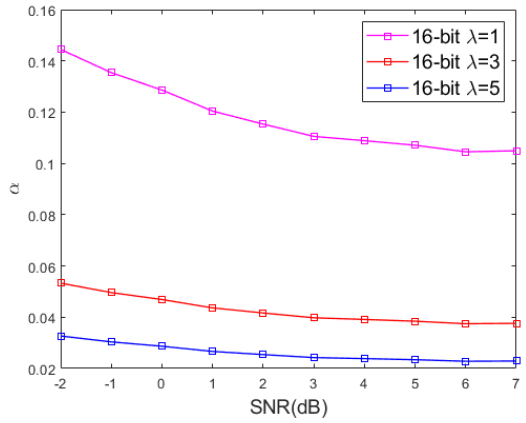

Fig. 7. The behavior of mixing weight α versus the channel SNR for different values of λ (1, 3, and 5).

decreases with the channel SNR:E_b/N_0, as expected. Also, the strength of the prediction denoted by $s(F)$ increases with the channel SNR but remains above 3 under the given SNR range that confirms the advantage of incorporating the predictor's results into the channel decoder initialization to boost the error recovery performance.

TABLE I

THE RELATIVE STRENGTH OF THE PREDICTORS $s(\mathbf{F})$ AS WELL AS THE PREDICTION BER (BIT ERROR RATE) FOR THE TWO TEST SCENARIOS.

TestE_b/N_0	0 dB	1 dB	2 dB	3 dB	4 dB
Test 1 $s(F)$	4.9	5.6	6.5	7.2	7.9
Test 1 BER	0.087	0.076	0.068	0.065	0.062
Test 2 $s(F)$	6.6	7.1	7.8	8.2	8.5
Test 2 BER	0.187	0.176	0.165	0.161	0.157

The simulation results for the two test scenarios are presented in Fig. 8 and Fig. 9. In Test 1, the proposed method with different λ values achieves a considerable improvement for different channel SNR values, which quantifies the gain achieved by using the prediction module. The highest performance gain is achieved for $\lambda = 1$ in this test. Test 2 presents a similar trend, but the achieved gain is slightly lower than test 1. Also, this test is less sensitive to the value of λ.

These results uncover many unknowns in designing *learning-powered* decoders and optimal parameter tuning, which is worthy of further investigations. The achieved gain in terms of the power reduction for the two test scenarios is given in Table II. We assess the models at 10^{-2}, 10^{-3}, and 10^{-4} BER levels. The power gain is the amount of the reduction in the transmit power (in dB scale) by the proposed *learning-powered* decoder to achieve the same BER performance of the standard SDA decoder. The achieved performance is considerable and varies between 0.5dB to 2dB. The achieved gain depends on the design parameters like $lambda$ and urges for a more in-depth analysis.

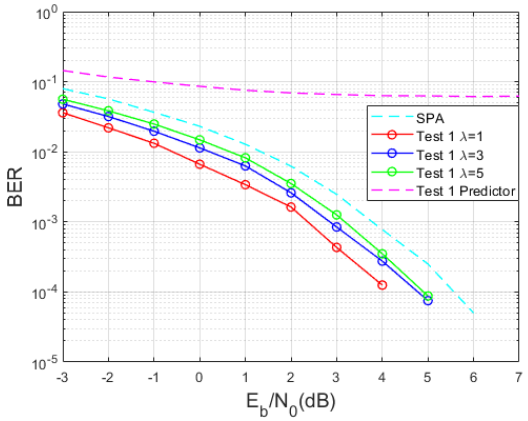

Fig. 8. This is simulation results for Test 1. The maximum number of iterations is set to 30 and λ to 1, 3, and 5.

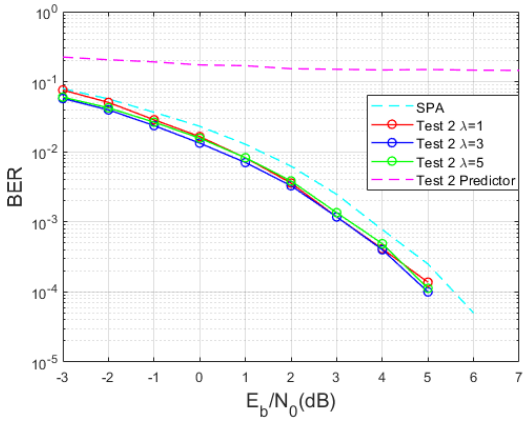

Fig. 9. This is simulation results for Test 2. The maximum number of iterations is set to 30 and λ to 1, 3, and 5..

TABLE II

THE ACHIEVED POWER REDUCTION GAIN BY THE PROPOSED DECODER AT DIFFERENT BER LEVELS.

TestGainBER level	10^{-2}	10^{-3}	10^{-4}
Test 1 $\lambda = 1$	1.9	1.5	1.7
Test 1 $\lambda = 3$	1.2	1.0	0.8
Test 1 $\lambda = 5$	0.8	0.7	0.7
Test 2 $\lambda = 1$	0.7	0.7	0.6
Test 2 $\lambda = 3$	1.0	0.7	0.6
Test 2 $\lambda = 5$	0.7	0.5	0.6

IV. CONCLUSION

In this work, we posed the idea of boosting the performance of channel decoders by including embedded learning modules. The idea is utilizing the learnability of data bits to enhance the performance of bit recovery algorithms. It does not replace the main task of a channel encoder, namely recovering channel errors by exploiting the embedded coding patterns, rather it complements the error recovery by exploiting the power of learning algorithms to guess the values of the lost and corrupted bits. As an exemplary implementation, we designed a SPA algorithm that uses an embedded deep learning module to decoded LDPC encoded data frames. The key challenges are tuning learning parameters, translating the learnability of the information bits into coded bits, model sharing between multiple transmitter-receiver pairs, and finding a universal way for tuning mixing coefficients. To the best of our knowledge, this is the first work in this area and yet we achieved a significant gain (0.5 to 2 dB in transmit power reduction) with our implementation. The gain is higher for noisier conditions, where the conventional decoders fail in recovering the information bits.

REFERENCES

[1] J. C. Kieffer, "A survey of the theory of source coding with a fidelity criterion," *IEEE Transactions on Information Theory*, vol. 39, no. 5, pp. 1473–1490, 1993.

[2] A. Razi, K. Yasami, and A. Abedi, "On minimum number of wireless sensors required for reliable binary source estimation," in *2011 IEEE Wireless Communications and Networking Conference*, pp. 1852–1857, IEEE, 2011.

[3] Y. Li, D. Roblek, and M. Tagliasacchi, "From here to there: Video inbetweening using direct 3d convolutions," *arXiv preprint arXiv:1905.10240*, 2019.

[4] J.-T. Hsieh, B. Liu, D.-A. Huang, L. F. Fei-Fei, and J. C. Niebles, "Learning to decompose and disentangle representations for video prediction," in *Advances in Neural Information Processing Systems*, pp. 517–526, 2018.

[5] *4G multi-mode turbo decoder*. A newTec Company.

[6] J. Pearl, *Reverend Bayes on inference engines: A distributed hierarchical approach*. Cognitive Systems Laboratory, School of Engineering and Applied Science ..., 1982.

[7] J. Kim and J. Pearl, "A computational model for causal and diagnostic reasoning in inference systems," in *International Joint Conference on Artificial Intelligence*, pp. 0–0, 1983.

[8] R. Tanner, "A recursive approach to low complexity codes," *IEEE Transactions on information theory*, vol. 27, no. 5, pp. 533–547, 1981.

[9] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on information theory*, vol. 47, no. 2, pp. 498–519, 2001.

[10] X.-Y. Hu, E. Eleftheriou, D.-M. Arnold, and A. Dholakia, "Efficient implementations of the sum-product algorithm for decoding ldpc codes," in *GLOBECOM'01*, vol. 2, pp. 1036–1036E, IEEE, 2001.

[11] S. C. Tatikonda, "Convergence of the sum-product algorithm," in *Proceedings 2003 IEEE Information Theory Workshop (Cat. No. 03EX674)*, pp. 222–225, IEEE, 2003.

[12] J. M. Mooij and H. J. Kappen, "Sufficient conditions for convergence of the sum–product algorithm," *IEEE Transactions on Information Theory*, vol. 53, no. 12, pp. 4422–4437, 2007.

[13] T. M. Cover and J. A. Thomas, *Elements of information theory*. John Wiley & Sons, 2012.

[14] T. K. Moon, *Error correction coding: mathematical methods and algorithms*. John Wiley & Sons, 2005.

[15] F. F. Sellers, H. M. Yue, and L. W. Bearnson, "Error detecting logic for digital computers," 1968.

[16] J. Pachelbel, "Canon in c." https://www.youtube.com/watch?v=trNolL4i6hw, 1680.

[17] P. Senneville and O. Toussaint, "A comme amour." https://www.youtube.com/watch?v=4Vs$_C wiB3co$, 1978.