Machine Learning Scaled Belief Propagation for Short Codes

Matthias Hummert, Dirk Wübben and Armin Dekorsy
Department of Communications Engineering
University of Bremen, 28359 Bremen, Germany
Email: {hummert, wuebben, dekorsy}@ant.uni-bremen.de

***Abstract*—The problem of finding good error correcting codes for short block lenghts and its corresponding decoders is an open research topic. A frequently applied soft decoder is the Belief Propagation (BP) decoder, however with degraded performance in case of short loops in the Tanner graph. This is especially problematic for short length codes as loops of small length are more likely to occur. In this paper, we propose the Machine Learning Scaled Belief Propagation (MLS-BP) to mitigate the performance loss of BP decoding for short length codes by introducing a learned scaling factor for the receive signals. The key point of this approach is the fact that the implementation of the BP decoder is not changed and the simple scaling leads to performance results comparable to other proposed BP improvements.**

***Index Terms*—Supervised Machine Learning, Belief Propagation, error-correcting codes, short block length**

I. Introduction

In a time where short packet transmissions are getting more and more important, especially for 5G, the research in finding good error correcting codes and corresponding decoders for these data transmissions is of huge importance. In general, the performance of channel codes improves with the block length, therefore finding good performing codes for short block length is not an easy task. Furthermore, optimum maximum likelihood decoding is too complex for most practical scenarios. The Belief Propagation (BP) decoder is a frequently applied soft decoder in particular for Low-Density-Parity-Check (LDPC) codes [1]. This decoder has shown to perform well for long block length, but is in general suboptimal due to loops in the Tanner graph. These loops harm the decoding performance as the reliability of the messages is overestimated due to the inherent dependencies in the code. Therefore, the question arises, if the BP decoder can be modified for decoding short codes.

As reported in [2], Tanner already proposed to scale the messages in each iteration to compensate for the loops in the decoder. For minimizing the BER, [3] proposed to use bruteforce search to find scaling factors in the check to variable messages of the BP and [4] exploited the consistency condition of Log-Likelihood-Ratios (LLRs) to scale the messages. In order to avoid this bruteforce search or complicated analytical analyis, a machine learning (ML) search has been proposed to find these scaling factors in [5].

Recently, the idea of learning general complete neural networks (NN) for soft decoding has raised significant attention [6]. However this approach only works for small dimension of codes since the procedure of learning a new soft input decoder by using neural networks is doomed by the curse of dimensionality. In order to overcome this drawback several strategies have been proposed e.g. [7]. For sequential decoding numerous approaches exists to learn the decoding of convolutional codes and Turbo codes [8].

Another approach is to interprete the BP decoder as a NN [9], [10]. The so-called Neural Belief Propagation (N-BP) decoder will be discussed in Subsection II-D and used as the benchmark.

In this paper, we propose an alternative approach by introducing a scaling factor for the calculation of the LLR of the receive signals. The key aspect is that the subsequent BP decoder remains unchanged, i.e., no additional scaling factors are incorporated in the passed messaged, only the input of the decoder is changed. This is especially interesting for hardware implementation purposes as an existing implementation of the BP decoder can be used without any modification of the decoder itself. The proposed approach can hence easily be integrated in existing schemes. A supervised learning procedure is introduced to adapt this proposed input scaling which is trained offline without adding extra computations to the online processing.

The paper is structured as follows: after discussing the system model in Section II, the Machine Learning Scaled Belief Propagaion (MLS-BP) and the training procedure are presented in Section III. In Section IV the performance is investigated and the conclusions are provided in Section V.

II. Preliminaries

A. System model

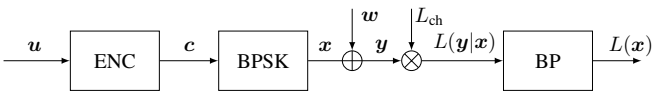

Fig. 1: System model of coded BPSK transmission over AWGN channel and LLR calculation prior to BP decoding

This work was partly funded by the German ministry of education and research (BMBF) under grant 16KIS1180K (FunKI).

Consider the given communication chain in Fig. 1 where a binary information word [1] $\boldsymbol{u} \in \mathbb{F}_2^k$ of length k is encoded (ENC) into the codeword $\boldsymbol{c} \in \mathbb{F}_2^n$ of length n by a linear block code Γ of code rate $R_c = k/n$. BPSK modulation is applied to the codeword by $\boldsymbol{x} = 1 - 2\boldsymbol{c}$, Additive White Gaussian Noise (AWGN) $\boldsymbol{w}$ is added by the channel leading to the receive vector $\boldsymbol{y} \in \mathbb{R}^n$ given by $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{w}$. For the transmission of BPSK symbols over AWGN channels the LLRs for the receive signals are given by

$$L(y_i|x_i) = \log\left(\frac{P(y_i|x_i=1)}{P(y_i|x_i=-1)}\right) = \frac{2}{\sigma_w^2} y_i = L_{\text{ch}} y_i \ . \quad (1)$$

This LLR equals the receive signal $\boldsymbol{y}$ scaled by the so called channel reliability $L_{\text{ch}} = 2/\sigma_w^2$ with the noise variance σ_w^2. Thus, the input of the BP decoder is a scaled version of the received signal vector $\boldsymbol{y}$, dependent on the Signal-to-Noise-Ratio (SNR).

B. Belief Propagation (BP) Decoder

In the following we will briefly summarize the BP decoder [11]. In general, each linear block code is completely described by its parity check matrix $\boldsymbol{H} \in \mathbb{F}_2^{m \times n}$. Each row of $\boldsymbol{H}$ represents a parity check equation of the code and each column stands for a code bit in the code. The parity check matrix can be graphically represented by a Tanner graph consisting of variable nodes v_i $(i = 1, \ldots, n)$ and check nodes chk_j $(j = 1, \ldots, m)$. An edge between variable node v_i and check node chk_j exists, if the code bit c_i participates in the jth check equation, i.e. $h_{ji} = 1$. The BP decoder now uses these parity checks and the input LLRs $L(y_i|x_i)$ (1) to calculate extrinsic information $L_{\text{e}}(x_i)$ for every code bit c_i by exchanging messages between the check and variable nodes [1]. Every BP iteration consists of two steps. At first, the check node chk_j collects the corresponding data from the connected variable nodes v_i and calculate extrinsic information $L_{\text{e}}^j(x_i)$ for every connected variable node v_i using the boxplus operation. For the second step, every variable node v_i sums together all the generated extrinsic information from the connected check nodes to calculate the complete extrinsic information $L_{\text{e}}(x_i) = \sum_{j \in v_i} L_e^j(x_i)$. The extrinsic information $L_{\text{e}}(x_i)$ is then added to the received LLR $L(y_i|x_i)$ (1) to finish a decoder iteration ($L(x_i) = L(y_i|x_i) + L_{\text{e}}(x_i)$). This iterative decoding process is repeated until a valid codeword is found or another stopping criterion is met. The final soft output of the BP decoder is hence given by $L(\boldsymbol{x})$ and for estimating the codeword $\hat{\boldsymbol{c}}$ a hard decision is performed $\hat{\boldsymbol{c}} = \text{sgn}\left(L(\boldsymbol{x})\right)$. We note the maximum number of iterations by N_{it}. Due to loops in the Tanner graph, the BP decoder is in general suboptimal. A loop is formed, if an information flow in the Tanner graph reaches its source node again. This is especially noticeable for very short code length as short loops are more likely and the performance is harmed. The shortest loop present in a code is called girth. This suboptimality is caused by the fact that the BP assumes statistical independence of the passed messages and this assumption is strictly speaking only true for infinite long codes. Due to this assumption LDPC codes are designed in a sparse fashion and in general, longer codes show better performance as the assumption is less violated.

Based on the decoder output $L(x_i)$ the expected value for each code bit x_i can be calculated by

$$\tilde{x}_i = \mathbb{E}\{x_i\} = \tanh\left(L(x_i)/2\right). \quad (2)$$

The elementwise $\tanh$ function provides a soft estimate $\tilde{\boldsymbol{x}}$ for the transmit word $\boldsymbol{x}$ and will be used to train our approach.

C. Neural Belief Propagation (N-BP) Decoder

The N-BP decoder [9], [10] unfolds the message exchange in the Factor graph over the iterations and interprets it as a NN with trainable weights in order to scale the exchanged messages. To design the NN the number of iterations needs to be fixed in advance and the number of weights increases linearly with the number of BP iterations. The training of all these weights leads to a significant training complexity. Our implementation of the N-BP is based on [12].

III. Machine Learning Scaled Belief Propagation(MLS-BP)

A. Learning Model

In the MLS-BP approach the input of a common BP decoder is rescaled and adapted via supervised learning. For the MLS-BP we propose to feed scaled LLRs

$$\tilde{L}(y_i|x_i) = \beta L(y_i|x_i) = \beta \frac{2}{\sigma_w^2} y_i = \underbrace{\beta L_{\text{ch}}}_{L_{\text{MLS}}} y_i \quad (3)$$

to the BP decoder and determine the non-negative scaling factor β by ML. In general, the scaling factor will depend on the channel code Γ and on the SNR. To simplify the nomenclature, we have introduced in (3) the scaling variable $L_{\text{MLS}} = \beta L_{\text{ch}}$ which can be interpreted as a "learned channel reliability". In Fig. 2, the training scheme to determine β and L_{MLS} is depicted.

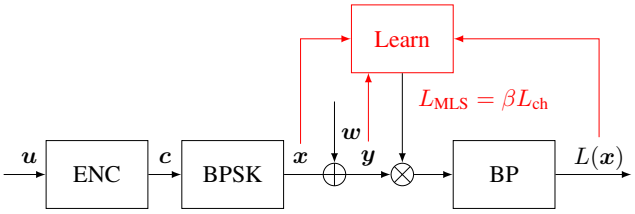

Fig. 2: MLS-BP learning schematic for the scaling constant L_{MLS}

The learning procedure is used during offline training to adapt the rescaling factor β. Therefore the training is carried out only once and the learned β is applied for inference afterwards. It follows the supervised learning principle where the BP decoder serves as a part of a function that is used to train β accordingly. In (3), the scaling factor L_{MLS} varies with

[1] We will note vectors by bold symbols $\boldsymbol{x}$, matrices by upper bold symbols $\boldsymbol{H}$ and an entry of the vector with subscripts, e.g., x_i for the ith entry. Soft estimates are noted as $\tilde{\boldsymbol{x}}$ and hard decisions are marked with $\hat{\boldsymbol{x}}$.

the SNR as the noise variance σ_w^2 is inverse proportional to the SNR. As a consequence, we should learn one scaling factor for every interesting SNR, respectively.

However, in order to further simplify the MLS-BP approach, we investigate the training of only one scaling factor $L_{\text{MLS}}^{\text{range}}$ for a *range* of interesting SNRs. We will observe, that this strategy leads only to a minor performance gap.

As a direct comparison to e.g. the N-BP with thousands of learned scaling factors, the MLS-BP approach needs to learn only one scalar factor β and the BP decoder remains untouched. Therefore the computational complexity of our approach is significantly lower as only one weight is adapted.

B. Supervised Learning Approach

The general idea of supervised learning is to tune learnable parameters of a function using input data (observations) to match the corresponding labeled output data [13]. Here the observations are the received signals $\boldsymbol{y}$ and the labels are the transmitted symbols $\boldsymbol{x}$. Our goal is to minimize a loss $\mathcal{L}(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ so that the BPSK soft estimates $\tilde{\boldsymbol{x}}$ (2) are as close as possible to the corresponding labeled data $\boldsymbol{x}$. The loss reflects some form of distance between the output of the function $\tilde{\boldsymbol{x}}$ and the labels $\boldsymbol{x}$. In order to minimize the loss, we use a variant of Stochastic Gradient Descent (SGD). SGD calculates the derivative w.r.t. to learnable parameters (one parameter β here) and tunes them to minimize the loss function iteratively.

To formalize this, we define the training set consisting of T training samples as $(\boldsymbol{y}^t, \boldsymbol{x}^t)$ for $t = \{1, ..., T\}$, so the training set contains T vectors of length n for $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively. The observations $\boldsymbol{y}$ and the BPSK soft estimates $\tilde{\boldsymbol{x}}$ are related via the received LLR scaling and BP decoding function $\tilde{\boldsymbol{x}} = f(\boldsymbol{y})$. For an initial value β the whole decoding chain is executed and soft symbols $\tilde{\boldsymbol{x}}$ are calculated for the loss $\mathcal{L}(\boldsymbol{x}, \tilde{\boldsymbol{x}})$. From the loss the derivative w.r.t. β is determined and the parameter β updates via SGD. The most basic form of SGD is given by the iteration

$$\beta^{\ell+1} = \beta^{\ell} - \frac{\eta}{|\mathcal{S}_\ell|} \sum_{t \in \mathcal{S}_\ell} \nabla_\beta \mathcal{L}(\boldsymbol{x}^t, \tilde{\boldsymbol{x}}^t) \tag{4}$$

where ℓ denotes the iteration index, η is the step size or learning rate and $\nabla_\beta L(\boldsymbol{x}, \hat{\boldsymbol{x}})$ is the gradient of the loss function $\mathcal{L}(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ w.r.t. the learning paramter β. We divide the training set into subsets $|\mathcal{S}_\ell| < T$ of equal size in the way that no training data is present twice in any subset. These subsets $\mathcal{S}_\ell$ are called batches and $|\mathcal{S}_\ell|$ is named batch size. When all subsets $\mathcal{S}_\ell$ are processed, the whole training set T has been used once which is called an epoch. Usually many epochs are used to train ML algorithms.

C. Learning procedure for MLS-BP

Considering our system model in Fig. 2 we generate new training vector sets $(\boldsymbol{y}^t, \boldsymbol{x}^t)$ with size T in every epoch and show our optimizer new training data in each iteration ℓ to make the training as efficient as possible. For given loss function and SNR, the processing chain is evaluated and the parameter β adopts accordingly. Here, the Adam optimizer [14] is used, which is a variant of the SGD method. This supervised learning procedure is implemented using Keras [15] and Tensorflow [16].

Two different loss functions are considered. First the Mean-Square Error (MSE) loss evaluates the Euclidian distance between the labels $\boldsymbol{x}^t$ and the soft estimates $\tilde{\boldsymbol{x}}^t$

$$\mathcal{L}_{\text{MSE}}\left(\boldsymbol{x}^t, \tilde{\boldsymbol{x}}^t\right) = \left\|\boldsymbol{x}^t - \tilde{\boldsymbol{x}}^t\right\|_2^2. \tag{5}$$

Second, the MI loss considers the negative of the approximate mutual information (MI) between $\boldsymbol{x}^t$ and the LLR $L(\boldsymbol{x}^t)$ given by

$$\begin{aligned} &\mathcal{L}_{\text{MI}}\left(\boldsymbol{x}^t, L(\boldsymbol{x}^t)\right) = -\text{MI}\left(\boldsymbol{x}^t, L(\boldsymbol{x}^t)\right) \\ &= -\left(1 - \frac{1}{n}\sum_{i=1}^{n} \log_2\left(1 + \exp\left(-x_i L(x_i)\right)\right)\right). \end{aligned} \tag{6}$$

The approximation is motivated by the ergodic theorem and has also been used for EXtrinsic Information Transfer (EXIT) charts to approximate mutual information [17]. The negative MI is considered in order to obtain a minimization problem.

IV. Simulation Results

A. Basic Parameters

For evaluating the MLS-BP we investigate the performance for a Bose-Chaudhuri-Hocquenghem (BCH) code, which has been used for evaluating the N-BP in [9] and [10], and a LDPC code from [18]. For all testing purposes a size of $|\mathcal{S}_\ell| = 100$ is used, a learning rate of $\eta = 0.01$ is applied and the BP decoder uses a maximum of $N_{\text{it}} = 5$ iterations, if not mentioned otherwise.

B. Performannce Evaluation for a BCH code

In Fig. 3 we depict the BER for the BCH code (63, 45) versus the scaling factor L_{MLS} for different SNRs. For a fixed SNR, the bruteforce (BF) search sweeps the scaling factor over a fixed range, determines the corresponding BER and selects L_{BF}^* resultung in the miniumun BER ($\triangle$). The scaling factors L_{MLS} found by using the MI loss (circle) and the MSE loss ($\diamond$) are marked, respectively. For both loss functions, the MLS-BP factor nearly matches the best BER using the bruteforce approach. This bruteforce search yields the best results but needs more computational power. On top our MLS-BP approach optimizes e.g. an information theoretic criterion like the MI (6) and has hence a more convenient motivation.

In Fig. 4 we validate this observation by showing the BER versus the SNR. We compare the BP decoder without scaling ("L_{ch}") with the found scaling factors L_{BF}^* obtained by bruteforce search and the learned scaling factors $L_{\text{MLS,MI}}$ for the MI loss. Compared to the common BP decoder without scaling, the demanding N-BP [9] decoder achieves a significant performance improvement of 1.5 dB at BER of 10^{-4}. This performance gap indicates the suboptimality of common BP decoding for codes of small girth, here the girth is just 4. However, the same performance improvement is achieved by all scaling approaches with significant reduction in computational complexity in comparison to the N-BP as many

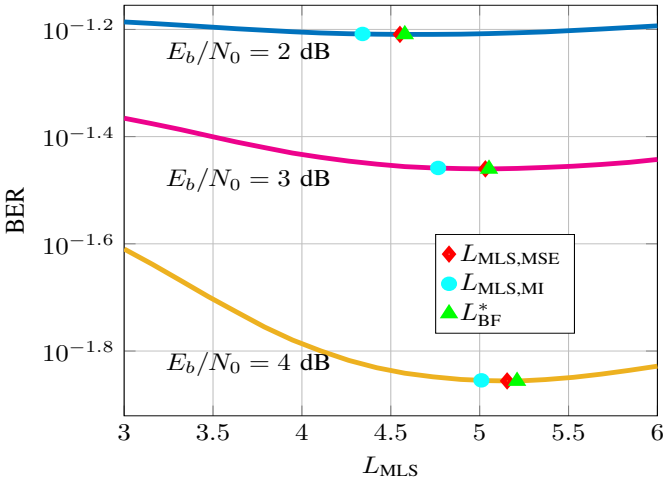

Fig. 3: BER performance for various scaling factors L_{MLS} for the (63, 45) BCH code

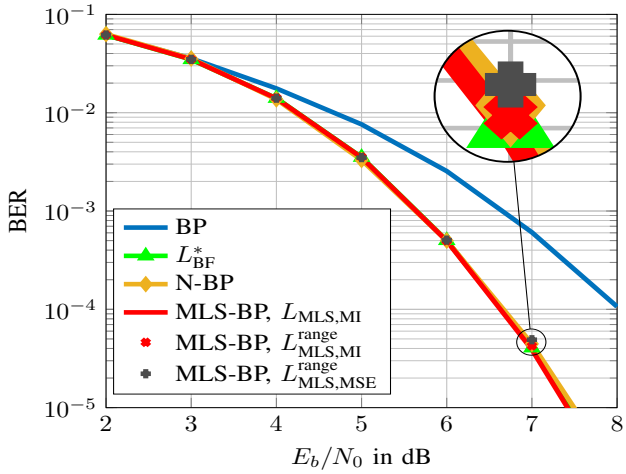

Fig. 4: (63, 45) BCH code, BER over E_b/N_0 compared for the different approaches

extra weight multiplications need to be executed. The MLS-BP with learned MI loss achieves virtually the performance of the scaled BP with bruteforce search for the optimum scaling factor L^*_{BF}. Interestingly, even the adaptation of *only* one learned scaling factor $L^{\text{range}}_{\text{MLS}}$ trained for the SNR *range* 2, ..., 8 dB leads to almost the same BER for both, the MI and the MSE loss function. This is due to the fact that the receive LLR tend to be "more" overestimated for high SNR and hence even one scaling factor for all SNR works nearly equally well. As the difference in BER performance of the MI and the MSE loss is negligible, we will concentrate on the MSE loss (5) subsequently.

C. Performance Evaluation for a short LDPC Codes

In Fig. 5, simulation results of the common BP and the MLS-BP with $L^{\text{range}}_{\text{MLS,MSE}}$ trained between 2 dB and 8 dB are shown for a short (32, 16) LDPC code and varying number of maximum iterations $N_{\text{it}} = 5,\ 10$. Obviously, the gains of

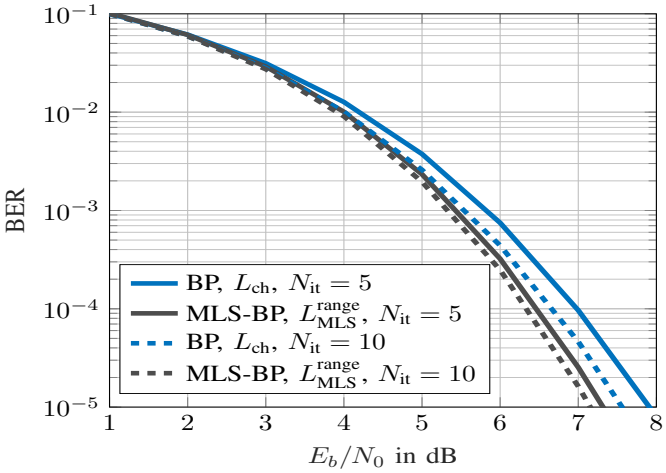

Fig. 5: (32, 16) LDPC code, BER over E_b/N_0 compared for the learned scaling factors $L^{\text{range}}_{\text{MLS}}$ over different number of iterations compared to the channel constant L_{ch}

scaling the receive signals are smaller in comparison to Fig. 4 as this LDPC code has a higher girth. For example, at a BER of 10^{-5} the MLS-BP achieves a gain of approximately 0.8 dB for a maximum of 5 BP iterations and 0.6 dB for 10 iterations, respectively. However, the MLS-BP with $N_{\text{it}} = 5$ iterations even outperforms the common BP with $N_{\text{it}} = 10$ iterations. Thus, complexity is again saved by our approach.

D. Usability for longer codes

In Fig. 6, we show simulations for LDPC codes that have lengths of $n = 128,\ 256,\ 512$ and 1008 with $R_c = 1/2$. We train the scaling factors $L^{\text{range}}_{\text{MLS}}$ for the interesting E_b/N_0 range of $2, \ldots, 4$ dB. As expected the gain decrease for increasing block length due to the higher girth. For the shown codes the performance gains of the approach vanish at the code length of $n = 1008$ in comparison to the common BP. This should be considered as an indication until which codelength this approach is applicable.

V. Summary and Future Work

In this paper, we propose the Machine Learning Scaled Belief Propagation (MLS-BP) to learn an input scaling factor of the BP decoder to improve the performance for short codes. The MLS-BP approach has only one trainable weight that is learned for each SNR or, with slightly degraded performance, for a SNR range. This very simple approach improves the BER performance of BP decoding for short codes at no additional cost and can directly been used with existing BP implementations.

References

[1] R. Gallager, "Low-density Parity-Check Codes," *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, Jan. 1962.

[2] D. J. MacKay and M. C. Davey, "Evaluation of Gallager codes for short block length and high rate applications," in *Codes, Systems, and Graphical Models*. Springer, 2001, pp. 113–130.

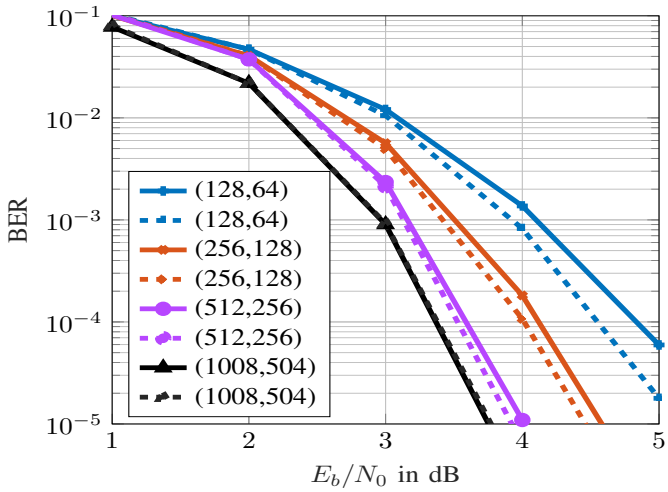

Fig. 6: different LDPC codes, dashed lines represent the MLS-BP ($L_{\text{MLS}}^{\text{range}}$) and solid lines show the standard BP

[3] M. Yazdani, S. Hemati, and A. H. Banihashemi, "Improving belief propagation on graphs with cycles," *IEEE Communications Letters*, vol. 8, no. 1, pp. 57–59, Jan. 2004.

[4] A. Alvarado, V. Núñez, L. Szczecinski, and E. Agrell, "Correcting Suboptimal Metrics in Iterative Decoders," in *Proceedings of IEEE International Conference on Communications, (ICC), Dresden, Germany*, June 2009.

[5] M. Lian, F. Carpi, C. Häger, and H. D. Pfister, "Learned Belief-Propagation Decoding with Simple Scaling and SNR Adaptation," in *IEEE International Symposium on Information Theory (ISIT)*, June 2019, pp. 161–165.

[6] T. Gruber, S. Cammerer, J. Hoydis, and S. ten Brink, "On deep learning-based channel decoding," in *51st Annual Conference on Information Sciences and Systems, (CISS), Baltimore, MD, USA*. IEEE, March 2017.

[7] S. Cammerer, T. Gruber, J. Hoydis, and S. ten Brink, "Scaling Deep Learning-Based Decoding of Polar Codes via Partitioning," in *2017 IEEE Global Communications Conference, (GLOBECOM), Singapore*. IEEE, Dec. 2017.

[8] H. Kim, S. Oh, and P. Viswanath, "Physical Layer Communication via Deep Learning," *IEEE Journal on Selected Areas in Information Theory*, vol. 1, no. 1, pp. 5–18, 2020.

[9] E. Nachmani, E. Marciano, L. Lugosch, W. J. Gross, D. Burshtein, and Y. Be'ery, "Deep Learning Methods for Improved Decoding of Linear Codes," *J. Sel. Topics Signal Processing*, vol. 12, no. 1, pp. 119–131, 2018.

[10] E. Nachmani, Y. Be'ery, and D. Burshtein, "Learning to Decode Linear Codes using Deep Learning," in *54th Annual Allerton Conference on Communication, Control, and Computing, Monticello, IL, USA*, Sept. 2016.

[11] J. Pearl, "Reverend Bayes on Inference Engines: A Distributed Hierarchical Approach," in *Association for the Advancement of Artificial Intelligence (AAAI)*, 1982.

[12] J. Hoydis, "Neural Network BP Decoder," https://colab.research.google.com/drive/1T8qP0fdXQjm7wH7qaBLuSFwHXmtT4VkE.

[13] O. Simeone, "A Very Brief Introduction to Machine Learning With Applications to Communication Systems," *IEEE Trans. Cogn. Comm. & Networking*, vol. 4, no. 4, pp. 648–664, Nov. 2018.

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, (ICLR), San Diego, CA, USA, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., May 2015.

[15] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[16] M. Abadi *et al.*, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/

[17] S. ten Brink, "Convergence of iterative decoding," *Electronics Letters*, vol. 35, no. 10, pp. 806–808, May 1999.

[18] M. Helmling, S. Scholl, F. Gensheimer, T. Dietz, K. Kraft, S. Ruzika, and N. Wehn, "Database of Channel Codes and ML Simulation Results," www.uni-kl.de/channel-codes.