This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCOMM.2020.3002915, IEEE Transactions on Communications

Trainable Communication Systems: Concepts and Prototype

Sebastian Cammerer, *Student Member, IEEE*, Fayçal Ait Aoudia, *Member, IEEE*, Sebastian Dörner, *Student Member, IEEE*, Maximilian Stark, *Student Member, IEEE*, Jakob Hoydis, *Senior Member, IEEE*, and Stephan ten Brink, *Senior Member, IEEE*

(Invited Paper)

***Abstract*—We consider a trainable point-to-point communication system, where both transmitter and receiver are implemented as neural networks (NNs), and demonstrate that training on the bit-wise mutual information (BMI) allows seamless integration with practical bit-metric decoding (BMD) receivers, as well as joint optimization of constellation shaping and labeling. Moreover, we present a fully differentiable neural iterative demapping and decoding (IDD) structure which achieves significant gains on additive white Gaussian noise (AWGN) channels using a standard 802.11n low-density parity-check (LDPC) code. The strength of this approach is that it can be applied to arbitrary channels without any modifications. Going one step further, we show that careful code design can lead to further performance improvements. Lastly, we show the viability of the proposed system through implementation on software-defined radios (SDRs) and training of the end-to-end system on the actual wireless channel. Experimental results reveal that the proposed method enables significant gains compared to conventional techniques.**

***Index Terms*—Autoencoder, end-to-end learning, iterative demapping and decoding, code design, geometric shaping, software-defined radio**

I. Introduction

End-to-end learning of communication systems consists of implementing the transmitter, channel, and receiver as a single neural network (NN), referred to as an autoencoder, and training it to reproduce its input at its output [1]. This approach enables joint optimization of the transmitter and receiver for a specific channel model without extensive mathematical analysis. Autoencoder-based communication systems have first been proposed in the context of wireless communications [1], and have subsequently been extended towards other settings, such as optical fiber [2], optical wireless [3], and molecular communications [4]. Most of these approaches are optimized on the symbol-wise categorical cross entropy (CE), which is equivalent to maximizing the mutual information between the channel input and output [5]. However, practical systems usually rely on bit interleaved coded modulation (BICM) [6] because of its reasonable complexity. With BICM, bit-metric decoding (BMD) [7] is used at the receiver, in which the points forming the channel input constellation are labeled by bit vectors, and bit levels are treated independently. Therefore, the points that form a constellation learned by an autoencoder optimized on the symbol-wise categorical CE must be labeled by bit vectors to be used in a practical BMD-based system. This is typically a delicate task, even for low modulation orders. Moreover, the mutual information between channel input and output is known to *not* be a rate achievable by BMD [7]. This suggests that this metric is not appropriate for trainable communication systems leveraging BMD.

In this work, training on the bit-wise mutual information (BMI) is considered instead. The BMI was proven to be an achievable rate for BMD [8], making it a suitable metric for the optimization of communication systems based on BMD. Moreover, training on the BMI enables joint optimization of constellation shaping and labeling on the transmitter side. On the receiver side, the learned demapper outputs log-likelihood ratios (LLRs) and can therefore be smoothly interfaced with a channel decoder, e.g., based on belief propagation (BP). Since the learned constellations benefit from iterative demapping and decoding (IDD) [9], we present a differentiable neural IDD architecture which unlocks significant gains on the additive white Gaussian noise (AWGN) channel, considering a standard 802.11n low-density parity-check (LDPC) code. We then show how further performance improvements can be achieved by designing an LDPC code "on top" of the trained communication system. We follow the basic idea in [10] which describes, to the best of our knowledge, the first autoencoder with a training procedure that directly optimizes the bit-metrics. A similar *neural receiver structure* has been proposed in [11] for visible light communications and, recently, in [12] for the optical fiber. While the authors of [11] train the system for a symbol-wise output (and calculate the LLRs *manually*), the authors of [12] use a bit-wise *neural equalizer* and show performance gains over the optical fiber by a tailored LDPC code. However, although both schemes consist of a trainable IDD receiver, they both do not train the system end-to-end, i.e., no training at transmitter is involved.

Significant gains by using non-uniform constellations (NUCs) for the BICM scenario have been shown in [13] based on *conventional* optimization techniques. In [10], a BICM autoencoder-based system for the AWGN channel with additive radar interference is presented. It was also observed that the concatenation with an outer channel code simplifies

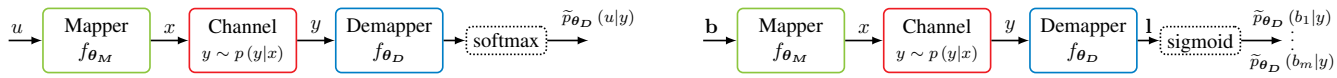
S. Cammerer, S. Dörner, and S. ten Brink are with the Institute of Telecommunications, University of Stuttgart, Pfaffenwaldring 47, 70659 Stuttgart, Germany ({cammerer,doerner,tenbrink}@inue.uni-stuttgart.de). This work has been supported by DFG, Germany, under grant BR 3205/6-1.

F. Ait Aoudia and J. Hoydis are with Nokia Bell Labs, Route de Villejust, 91620 Nozay, France ({faycal.ait_aoudia,jakob.hoydis}@nokia-bell-labs.com).

M. Stark is with the Institute of Communications, Hamburg University of Technology, Hamburg, Germany (maximilian.stark@tuhh.de).

0090-6778 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Exeter. Downloaded on June 18,2020 at 15:32:29 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCOMM.2020.3002915, IEEE Transactions on Communications



(a) Symbol-wise autoencoder

(b) Bit-wise autoencoder

Fig. 1: Symbol-wise and bit-wise autoencoder architectures

if BMD-based autoencoders are used and further improves the overall performance. In this work, we aim to put the bit-metric optimized autoencoder on a theoretical solid ground and introduce the bit-wise autoencoder in the BICM scenario. Although all simulations in this paper were carried out considering an AWGN channel, the proposed approach can be applied without modifications to any channel model. Thus, we demonstrate the universality of the proposed method through its implementation on software-defined radios (SDRs), and by training on an actual wireless channel, using the algorithm proposed in [14] which enables training of the end-to-end system even if the channel gradient is not accessible or not defined. The observed gains confirm the benefits of trainable communication systems in practice. Our learning-based optimization of the full physical layer could be completely automated and might hence be one of the key ingredients of next generation communication systems.

The rest of this paper is organized as follows. Section II motivates training on BMI from an information-theoretical viewpoint. The neural IDD structure is introduced in Section III. In Section IV, code design is performed to achieve further gains, and Section V presents the results of over-the-air experiments. Finally, Section VI concludes the paper.

Notations: Random variables are denoted by capital italic font, e.g., X, Y, with realizations x, y, respectively. $I(X;Y)$, $p(y|x)$ and $p(x,y)$ represent the mutual information, conditional probability, and joint probability distribution of the two random variables X and Y. Multivariate random variables are represented with capital bold font, e.g., $\mathbf{Y} = [Y_0, Y_1]^T$. Vectors are represented using a lower case bold font, e.g., $\mathbf{y}$. The cross entropy for two discrete distributions $p(x)$ and $q(x)$ is $H(p(x), q(x)) := -\sum_x p(x) \log q(x)$.

II. Bit-wise autoencoder

Although it appears as a trivial modification at first glance, the transition from symbol-wise to bit-wise autoencoder-based communication systems turns out to require a carefully adjusted framework. The main difference in the objective is to minimize the bit error rate (BER) instead of the symbol error rate (SER). This immediately leads to the question of how to find the optimal labeling scheme for the learned constellations, which is not part of previous autoencoder implementations.

Fig. 1a shows the architecture of such a typical autoencoder-based end-to-end communication system optimized on the symbol-wise categorical CE. For clarity, complex-unidimensional constellations are considered, as generalization to multidimensional constellations is trivial. The number of bits per channel use is denoted by m. The mapper takes as input a message $u \in \{0, \dots, 2^m - 1\}$, and maps it to a complex baseband symbol x. Therefore, it implements the mapping $f_{\boldsymbol{\theta}_M} : \{0, \dots, 2^m - 1\} \mapsto \mathbb{C}$, with trainable parameters $\boldsymbol{\theta}_M$. The complex baseband symbol is sent over a channel $y \sim p(y|x)$, with $y \in \mathbb{C}$. The received sample is fed to the demapper, which implements the mapping $g_{\boldsymbol{\theta}_D} : \mathbb{C} \mapsto \mathbb{R}^{2^m}$ with trainable parameters $\boldsymbol{\theta}_D$. The 2^m outputs of the demapper are called *logits*. Probabilities over the 2^m messages, denoted by $\widetilde{p}_{\boldsymbol{\theta}_D}(u|y)$, are obtained by applying the softmax function to the logits. For a general introduction of the concept of end-to-end learning we refer the interested reader to [1].

The constellation obtained by training on the symbol-wise CE needs to be labeled to be used in a conventional BICM system. However, since the learned constellation points do not usually form a grid (see Fig. 4), as in a conventional QAM, finding the optimal labeling is a combinatorial problem with $2^m!$ possibilities (neglecting symmetries). Therefore, we can only rely on sub-optimal heuristics to label the constellation points *after* the training process, as, e.g., done in [2].

A. Information theory perspective

An autoencoder-based communication system is typically trained by minimizing the categorical CE between the true posterior distribution $p_{\boldsymbol{\theta}_M}(u|y)$[1] and the one learned by the receiver $\widetilde{p}_{\boldsymbol{\theta}_D}(u|y)$, averaged over all the possible channel outputs y, i.e.,

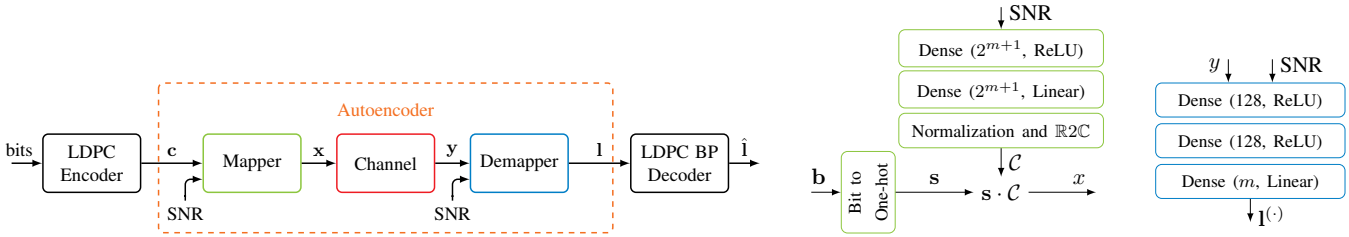
$$\mathbb{E}_y \left[H\left(p_{\boldsymbol{\theta}_M}(u|y), \widetilde{p}_{\boldsymbol{\theta}_D}(u|y)\right)\right] = -\mathbb{E}_y \left[\sum_{u=0}^{2^m-1} p_{\boldsymbol{\theta}_M}(u|y) \log \widetilde{p}_{\boldsymbol{\theta}_D}(u|y) \right]. \quad (1)$$

Assuming that the NN implementing the demapper has a sufficiently high capacity (i.e., is large enough), this is equivalent to maximizing the mutual information between the channel input X and output Y [5]. It is known [7] that achieving the maximum $I(X;Y)$ requires *multistage decoding* at the receiver, together with *multilevel coding* at the transmitter, i.e., the use of an individual binary code on each bit level. As a consequence, most practical systems rely on the simpler BICM encoder [6], with which all bit-levels are encoded by a single binary code. At the receiver, BMD is used, which treats the bit levels independently. For practical use of a constellation learned by an autoencoder trained on (1) with BMD, the constellation needs to be labeled, i.e., each constellation point needs to be mapped to a unique bit vector. This task is non-trivial, and becomes quickly intractable even for small modulation orders. Moreover, $I(X;Y)$ is not an achievable rate by BMD [7]. Therefore, even if an optimal labeling is assumed (e.g., found by exhaustive search), a constellation learned by minimization of (1) is not necessarily optimal for practical BMD receivers.

[1]The posterior depends on the trainable parameters $\boldsymbol{\theta}_M$ of the mapper.

0090-6778 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Exeter. Downloaded on June 18,2020 at 15:32:29 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCOMM.2020.3002915, IEEE Transactions on Communications



(a) Setup overview (b) Mapper (c) Demapper

Fig. 2: Bit-wise autoencoder setup with channel coding, where $\mathbf{c} = \left[\mathbf{b}^{(1)\mathsf{T}}, \ldots, \mathbf{b}^{(s)\mathsf{T}}\right]^{\mathsf{T}}$ and $\mathbf{l} = \left[\mathbf{l}^{(1)\mathsf{T}}, \ldots, \mathbf{l}^{(s)\mathsf{T}}\right]^{\mathsf{T}}$. For simplicity, the interleaver/deinterleaver are considered part of the LDPC graph, and are therefore not shown.

An achievable rate by BMD is the BMI [8]

$$R \coloneqq H(\mathbf{B}) - \sum_{j=1}^{m} H(B_j|Y) \leq I(X;Y) \tag{2}$$

where $\mathbf{B}$ denotes the binary random variable associated with the input bit vector $\mathbf{b}$ of length m. This result motivates the bit-wise autoencoder shown in Fig. 1b. In the bit-wise autoencoder, the mapper takes as input a bit vector of length m, and the demapper outputs m logits (one per bit) which form a vector denoted by $\mathbf{l} \in \mathbb{R}^m$. Probabilities over the m bits, denoted by $\widetilde{p}_{\boldsymbol{\theta_D}}(b_j|y),\ j = 1, \ldots, m$, are obtained by element-wisely applying the sigmoid function

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

to the corresponding logits. One can see that the logits correspond to LLRs[2]:

$$\forall j \in \{1, \ldots, m\},\ \mathrm{LLR}(j) \coloneqq \log \frac{1 - \widetilde{p}_{\boldsymbol{\theta_D}}(b_j = 1|y)}{\widetilde{p}_{\boldsymbol{\theta_D}}(b_j = 1|y)} \tag{3}$$

$$= l_j \tag{4}$$

where

$$\widetilde{p}_{\boldsymbol{\theta_D}}(b_j = 0|y) = \sigma\left(l_j\right). \tag{5}$$

Optimization of the bit-wise autoencoder is done by minimizing the total binary CE

$$\mathcal{L}(\boldsymbol{\theta_M}, \boldsymbol{\theta_D}) \coloneqq \sum_{j=1}^{m} \mathbb{E}_y \left[H\left(p_{\boldsymbol{\theta_M}}(b_j|y), \widetilde{p}_{\boldsymbol{\theta_D}}(b_j|y)\right)\right] \tag{6}$$

$$= \sum_{j=1}^{m} \mathbb{E}_{y,b_j} \left[-\log \widetilde{p}_{\boldsymbol{\theta_D}}(b_j|y)\right] \tag{7}$$

which is closely related to the BMI

$$\mathcal{L}(\boldsymbol{\theta_M}, \boldsymbol{\theta_D}) = H(\mathbf{B}) - R + \sum_{j=1}^{m} \mathbb{E}_y \left[\mathrm{D}_{\mathrm{KL}}\left(p_{\boldsymbol{\theta_M}}(b_j|y) || \widetilde{p}_{\boldsymbol{\theta_D}}(b_j|y)\right)\right] \tag{8}$$

where D_{KL} is the Kullback-Leibler (KL) divergence, and $p_{\boldsymbol{\theta_M}}(b_j|y),\ j = 1, \ldots, m$, are the true posterior distributions.

[2]Note that, we define the logits such that $\widetilde{p}_{\boldsymbol{\theta_D}}(b_j = 0|y) = \sigma\left(l_j\right)$ instead of the more common definition $\widetilde{p}_{\boldsymbol{\theta_D}}(b_j = 1|y) = \sigma\left(l_j\right)$. However, this only impacts the sign but allows to use logits and LLRs equally throughout this paper

Interestingly, according to [8], (6) itself is an achievable rate for an imperfect receiver. Rewriting (6) as (8) allows to connect the actual BMI to the achievable rate for imperfect receivers. In turn, it becomes clear how trainable communication systems are able to easily outperform conventional systems with imperfect receivers. In more detail, the first term on the right-hand side of (8) is the entropy of the bit vector generated by the source, which is typically constant and equals the amount of bits m mapped to one symbol (assuming uniformly distributed bits). The second term is the BMI, and the third term is the sum of the KL divergences between the true posterior distributions $p_{\boldsymbol{\theta_M}}(b_j|y)$ and the ones learned by the demapper $\widetilde{p}_{\boldsymbol{\theta_D}}(b_j|y)$, and accounts for the sub-optimality of the receiver. Training of the end-to-end system by minimizing $\mathcal{L}$ therefore corresponds to maximizing R, which is suited to BMD, while minimizing the KL divergence between the optimal demapper and the one learned at the receiver. Moreover, because the mapper assigns each bit vector to a constellation point, joint geometric shaping and bit labeling is performed when minimizing $\mathcal{L}$. The NN implementing the demapper should approximate the posterior distributions $p_{\boldsymbol{\theta_M}}(b_j|y)$ of a constellation maximizing the BMI with high precision. This avoids learning a constellation where the posterior distributions are well-approximated, but the BMI is not maximized. This can be ensured by choosing the NN implementing the demapper large enough, so that it is (in principle) capable of approximating a wide range of posterior distributions. Unfortunately, even with a very large NN, there is no guarantee that the learning procedure converges to such a solution for every channel.

B. Simulation setup

We will now evaluate the autoencoder on an AWGN channel with two-sided noise-power spectral-density N_0, i.e.,

$$p(y|x) = \frac{1}{\pi N_0} \exp\left(-\frac{|y-x|^2}{N_0}\right). \tag{9}$$

Fig. 2a shows the end-to-end system. A bitstream is fed to an LDPC encoder which generates codewords $\mathbf{c}$ of length n. The number of bits per channel use is m and it is assumed that n is a multiple of m. Each codeword is broken apart into $s = \frac{n}{m}$ bit vectors $\mathbf{b}$ of length m, i.e., $\mathbf{c} = \left[\mathbf{b}^{(1)\mathsf{T}}, \ldots, \mathbf{b}^{(s)\mathsf{T}}\right]^{\mathsf{T}}$. Each bit vector $\mathbf{b}^{(i)}, i = 1, \ldots, s$, is mapped into a complex baseband symbol $x_i \in \mathbb{C}$, and is sent over the channel. On the receiver

0090-6778 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Exeter. Downloaded on June 18,2020 at 15:32:29 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCOMM.2020.3002915, IEEE Transactions on Communications

side, the demapper processes each received sample $y_i \in \mathbb{C}$ and generates LLRs $\mathbf{l}^{(i)} \in \mathbb{R}^m$. Finally, the LLRs of the entire codeword $\mathbf{l} = \left[\mathbf{l}^{(1)\mathsf{T}}, \ldots, \mathbf{l}^{(s)\mathsf{T}}\right]^{\mathsf{T}}$ are fed into a BP decoder. For simplicity, the interleaver/deinterleaver are considered part of the LDPC graph, and are therefore not shown.

The architectures of the NNs that implement the mapper and the demapper are shown in Fig. 2b and Fig. 2c, respectively. The mapper includes an NN that generates a continuum of constellations $\mathcal{C} \in \mathbb{C}^{2^m}$ that are determined by the signal-to-noise ratio (SNR). Note that $\mathcal{C}$ determines the whole constellation set, i.e., contains 2^m complex-valued symbols and the one-hot vector $\mathbf{s}$ selects the corresponding symbol. The NN included in the mapper is made of two dense layers of 2^{m+1} units each, one with rectified linear unit (ReLU) activations and the other with linear activations. The 2^{m+1} outputs of the second dense layer correspond to the real and imaginary parts of the 2^m constellation points. The last layer is a normalization layer, which ensures that $\mathbb{E}_x\left[|x|^2\right] = 1$. A bit vector $\mathbf{b}$ is mapped to one of the constellation points. To that aim, it is first converted to its one-hot representation denoted by $\mathbf{s}$, i.e., the vector of size 2^m with all elements set to zero, except the one whose index has $\mathbf{b}$ as binary representation set to one. The vector $\mathbf{b}$ is then mapped to a constellation point x by taking the product of $\mathbf{s}$ and $\mathcal{C}$. Note that this is only the training procedure and the later implementation of the mapper can be further optimized in terms of implementation complexity. The demapper is made of three dense layers, two with ReLU activations and 128 units each, and the last one with linear activations and m units. Moreover, the demapper takes as input the SNR (in dB). This was motivated by the observation that the posterior distribution depends on the SNR.

We want to emphasize that the constellation set in our setup is SNR dependent, i.e., mapper and demapper both need to know the SNR. However, this is just a generalization of the case when re-training per SNR is performed as for trainable systems the SNR is always implicitly part of the training data. In case that feedback of the SNR is not possible, the same setup can be trained with fixed SNR input at the price of a slightly reduced BER performance. Recall that in *classical* communication systems (e.g., in the 5G standard), we typically define the modulation and coding scheme (MCS) and, thus, the constellations (and accordingly the LDPC code) can be also adapted.

An 802.11n [15] irregular LDPC code was considered, with rate $r = \frac{1}{2}$ and codewords of length $n = 1296$ bit. The SNR is defined as

$$\text{SNR} = \frac{1}{rmN_0} \tag{10}$$

which corresponds to the energy per information-bit per noise-power spectral-density ratio since the normalization ensures $\mathbb{E}_x\left[|x|^2\right] = 1$. Training was done with the Adam [16] optimization algorithm and we have initialized the weights of the mapper using the *Glorot initializer* [17]. We have summarized the training parameters in Table I.

TABLE I: Training Parameters

Parameter	Value
Batch-size	500
Learning rate	decreasing from 10^{-3} to 10^{-5}
Training SNR	4 dB around waterfall region of LDPC code
Optimizer	Adam [16]
Initializer	Glorot [17]

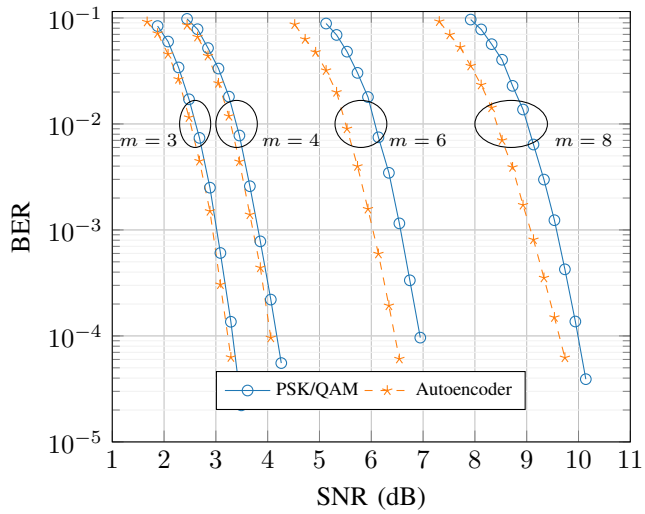


Fig. 3: BER achieved by PSK/QAM constellations and the bit-wise autoencoder for $m = 3,\ 4,\ 6$ and 8. The baseline is PSK for $m = 3$, and QAM otherwise.

The loss function $\mathcal{L}$ was estimated by

$$\mathcal{L}(\boldsymbol{\theta_M}, \boldsymbol{\theta_D}) \approx -\frac{1}{|\mathcal{B}|} \sum_{\mathbf{b} \in \mathcal{B}} \sum_{j=1}^{m} \Big(b_j \log\left(\widetilde{p}_{\boldsymbol{\theta_D}}\left(b_j|y\right)\right) + (1 - b_j) \log\left(1 - \widetilde{p}_{\boldsymbol{\theta_D}}\left(b_j|y\right)\right) \Big) \tag{11}$$

where $\mathcal{B}$ is a finite set of samples of bit vectors. The *batch-size* $|\mathcal{B}|$, i.e., the number of samples used to estimate $\mathcal{L}$, was set to 500, whereas the learning rate was progressively decreased from 10^{-3} to 10^{-5}. At training, the SNR was randomly and uniformly selected for each example from a 4 dB range centered on the waterfall region of the code. Note that, at training, the channel encoder and decoder are not needed, as the loss function (6) is based on the output of the demapper. At evaluation, the number of iterations performed by the BP decoder was set to 40.

C. Results

The (coded) BER achieved by the bit-wise autoencoder was compared to that of phase-shift keying (PSK) for $m = 3$ and to QAM for $m = 4$, 6, and 8. For the considered baselines, maximum-likelihood demapping was used. Fig. 3 shows the BER achieved by the compared approaches with the setup presented in Section II-B. It can be seen that the schemes learned by the bit-wise autoencoder outperform the baselines. Moreover, the gains achieved by the learning schemes increase with the modulation order, reaching 0.8 dB for $m = 8$, compared to 0.3 dB for $m = 3$.

0090-6778 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Exeter. Downloaded on June 18,2020 at 15:32:29 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCOMM.2020.3002915, IEEE Transactions on Communications

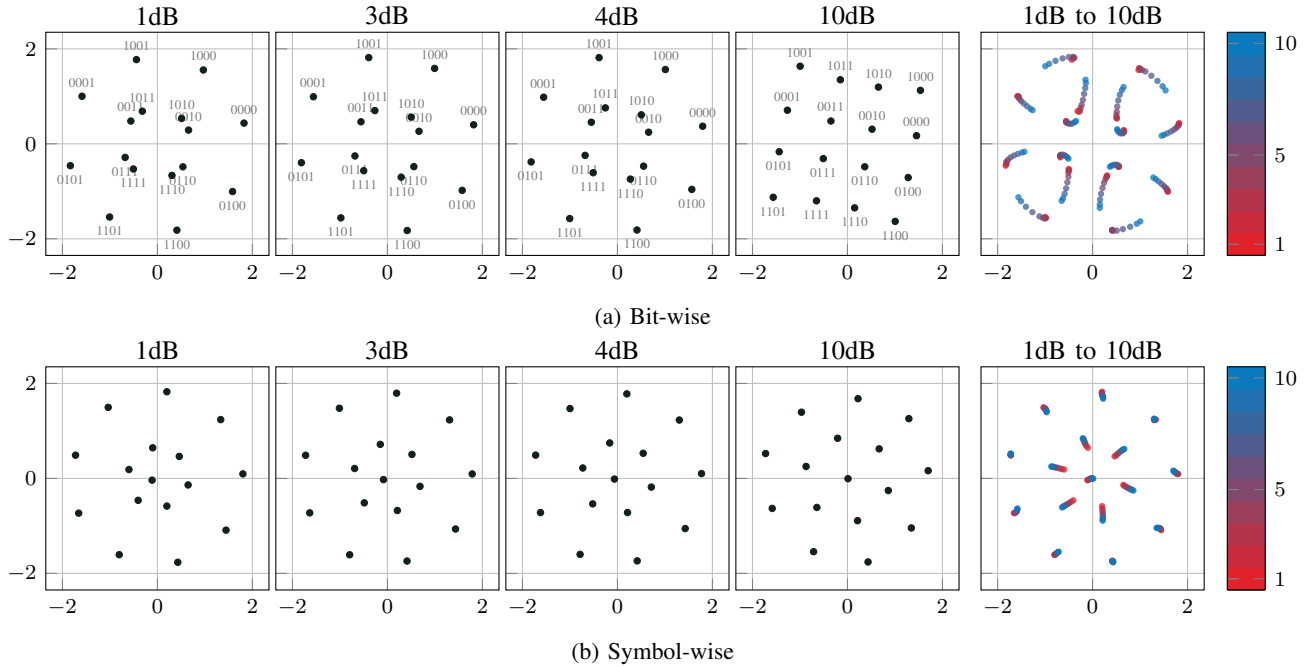


Fig. 4: Constellations learned for $m = 4$ bit at varying SNR. The learned labeling of the bit-wise autoencoder is shown in gray.

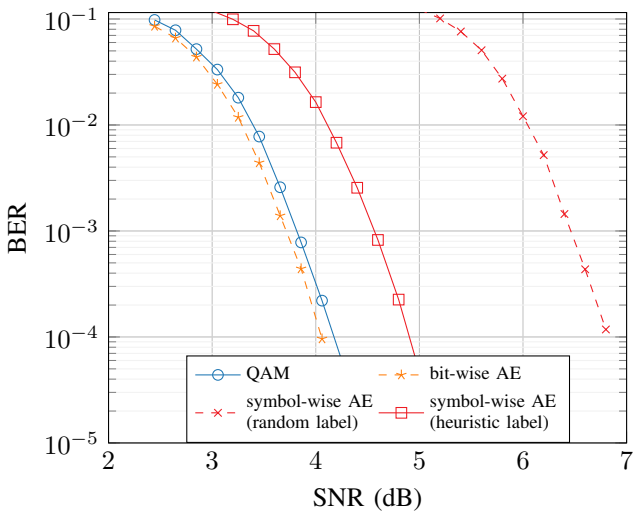


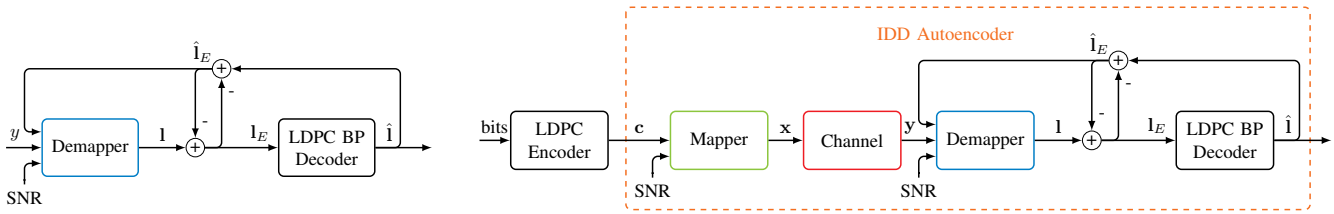
Fig. 5: Achieved BER with $m = 4$ bit for baseline quadrature amplitude modulation (QAM), bit-wise autoencoder and symbol-wise autoencoder (with random and heuristic labeling).

To get insight into the learned constellation geometries, Fig. 4 shows the constellations learned for $m = 4$, when leveraging the bit-wise and symbol-wise autoencoder, respectively. It can be seen that training on the BMI (Fig. 4a) leads to a constellation which differs significantly from the one obtained from training on the symbol-wise CE (Fig. 4b). Recall that the use of the constellation learned with the symbol-wise autoencoder (Fig. 4b) in a BMD-based system requires an additional heuristic labeling step, which is typically not trivial.

Besides the labeling, the optimal position of the constellation points can differ under different metrics. An intuitive example is depicted in Fig. 4 where, for the low SNR region, the bit-wise optimized autoencoder (Fig.4a) *clusters* constellation points into groups that only differ in one bit position. This effectively weakens the reliability of this bit position while improving the other positions and, thereby, optimizes the overall achievable information rate. However, in the corresponding symbol-metric a confusion is unavoidable within the clustered symbols and, therefore, a degraded symbol-wise performance can be observed. When looking at the corresponding symbol-metric optimized constellation in Fig.4b, we see that such a clustering does not occur for the exact same channel parameters after training. While a symbol-wise trained constellation improves the SER, it degrades the achievable BER. Fig. 5 shows the BER performance of a symbol-wise trained autoencoder system with randomly chosen labels and heuristic labels compared with a QAM baseline with Gray labeling and a bit-wise optimized autoencoder. We want to emphasize that our heuristic labeling is not necessarily optimal, but underlines the difficulties in finding such a labeling. As it can be seen, the BER performance of the symbol-wise optimized autoencoder with heuristic labels is ≈ 0.7dB worse than the conventional QAM baseline and even ≈ 0.8dB worse than the bit-wise optimized autoencoder. These results suggest that one should train on the BMI when using BMD. Moreover, in the case of the bit-wise autoencoder, the learned labeling is also shown in Fig. 4a. One can see that a form of Gray labeling was learned, where two points close to each other are assigned labels that differ in only one bit.

0090-6778 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Exeter. Downloaded on June 18,2020 at 15:32:29 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCOMM.2020.3002915, IEEE Transactions on Communications



(a) Iterative demapping and decoding

(b) IDD autoencoder setup

Fig. 6: Bit-wise autoencoder and iterative receiver structure. Each LDPC codeword $\mathbf{c}$ is mapped into s bit vectors $\mathbf{b}^{(i)}$, i.e., s autoencoder transmissions.

III. Bit-wise iterative autoencoder

In this section, the bit-wise autoencoder is extended to IDD receivers [9]. The key idea of IDD is to apply the Turbo Principle to soft demapping together with channel decoding, as illustrated in Fig. 6a. With IDD, the demapper takes as input the received samples y together with prior knowledge on the bits denoted by $\hat{\mathbf{l}}_E$, and generates soft decisions $\mathbf{l}$. $\hat{\mathbf{l}}_E$ is subtracted from $\mathbf{l}$, which results in the so-called *extrinsic information* $\mathbf{l}_E$ that is fed to the decoder. The decoder then computes improved LLRs based on the error-correcting code. These are denoted by $\hat{\mathbf{l}}$. The prior knowledge fed to the demapper is the extrinsic information $\hat{\mathbf{l}}_E$ generated by the decoder, defined as $\hat{\mathbf{l}}_E = \hat{\mathbf{l}} - \mathbf{l}_E$. At the first iteration, $\hat{\mathbf{l}}_E$ is set to the null vector, as no prior information is assumed to be available on the bits. IDD compensates for the information loss caused by BMD, as the use of additional prior knowledge about the other bits result in a more reliable bit estimate. Demapping with no prior knowledge gives separate estimates per bit, with no other prior knowledge than the channel output. However, the extrinsic information feedback from the channel decoder allows to refine the demapper's estimate, which then refines the decoder estimate and vice versa.

The remainder of this section shows how the bit-wise autoencoder can be extended to IDD. This enables training of the end-to-end system including channel decoding with IDD for any channel model.

Algorithm 1: Iterative demapping and decoding

Input : $y_1, \ldots, y_s$

Output: $\hat{\mathbf{l}} = \left[\hat{l}_1, \ldots, \hat{l}_n\right]^{\mathsf{T}}$

/* Initialization */

1 $\hat{\mathbf{l}}_E^{(j)[0]} \leftarrow \mathbf{0},\ \forall j = 1, \ldots, s$

2 $\mu_{c,v}^{[0]} \leftarrow 0,\ \forall (c,v) \in \mathcal{E}$

/* Iterative demapping & decoding */

3 **For** $i = 1, \ldots, I$ **do**

/* Demapping */

4 $\mathbf{l}^{(j)[i]} \leftarrow g_{\boldsymbol{\theta}_D}\left(y_i, \hat{\mathbf{l}}_E^{(j)[i-1]}\right),\ \forall j = 1, \ldots, s$

5 $\mathbf{l}_E^{(j)[i]} \leftarrow \mathbf{l}^{(j)[i]} - \hat{\mathbf{l}}_E^{(j)[i]},\ \forall j = 1, \ldots, s$

6 $\mathbf{l}_E^{[i]} \leftarrow \left[\mathbf{l}_E^{(1)[i]\mathsf{T}}, \ldots, \mathbf{l}_E^{(s)[i]\mathsf{T}}\right]^{\mathsf{T}}$

/* Decoding */

7 $\mu_{v,c}^{[i]} \leftarrow l_{E,c}^{[i]} + \sum_{c' \in \mathcal{C}_v \setminus \{c\}} \mu_{c',v}^{[i-1]},\ \forall (v,c) \in \mathcal{E}$

8 $\mu_{c,v}^{[i]} \leftarrow 2\tanh^{-1}\left(\prod_{v' \in \mathcal{V}_c \setminus \{v\}} \tanh\left(\frac{\mu_{v',c}^{[i]}}{2}\right)\right),\ \forall (c,v) \in \mathcal{E}$

9 $\hat{l}_j^{[i]} \leftarrow \hat{l}_{E,j}^{[i]} + \sum_{c' \in \mathcal{C}_j} \mu_{c,j}^{[i]},\ \forall j = 1, \ldots, n$

10 $\hat{\mathbf{l}}_E^{[i]} \leftarrow \hat{\mathbf{l}}^{[i]} - \mathbf{l}_E^{[i]}$

11 **end**

12 **return** $\hat{\mathbf{l}} = \left[\hat{l}_1^{[I]}, \ldots, \hat{l}_n^{[I]}\right]^{\mathsf{T}}$

A. Extending the autoencoder with IDD

It is assumed that LDPC codes are used together with BP decoding. However, the proposed approach can be generalized to other coding schemes and decoding algorithms. The autoencoder with IDD receiver is shown in Fig. 6b, where $\mathbf{l}_E^{(j)}$ and $\hat{\mathbf{l}}_E^{(j)}$ denote the extrinsic information [18] corresponding to the jth complex baseband symbol, generated by the demapper and decoder, respectively. The internal structure of the demapper remains the same as in Fig. 2c. Only the number of inputs, and therefore the number of weights in the first layer, is increased.

The operations performed by the IDD receiver are shown in Algorithm 1. In this algorithm, I denotes the number of iterations performed by the receiver, and $\mathcal{E}$ denotes the set of edges of the Tanner graph associated to the considered code. $\mathcal{C}_v(\mathcal{V}_c)$ denote the neighboring check (variable) nodes of the variable (check) node v (c). Some short background on LDPC codes and terminology will be provided in Section IV. The superscript $[i]$ is used to refer to the ith iteration. At each iteration, and for each received sample $y_j,\ j = 1, \ldots, s$, the demapper $g_{\boldsymbol{\theta}_D}$ computes LLRs on the bits, denoted by $\mathbf{l}^{(j)}$, and using the previously computed extrinsic information generated by the decoder $\hat{\mathbf{l}}_E^{(j)}$ (line 4), initially set to zero (line 1). The extrinsic information generated by the demapper $\hat{\mathbf{l}}_E^{(j)}$ for each received sample is then computed (line 5). The extrinsic information generated by the demapper for the entire codeword is constructed by concatenating the ones of individual symbols (line 6). Next, one iteration of BP is performed from the demapper extrinsic information (lines 7 to 9). Note that this requires knowledge of the values of messages from check nodes to variable nodes of the previous iteration, which are initially set to 0 (line 2). Finally, the extrinsic information generated by the decoder is computed (line 10), which corresponds to the prior knowledge fed to the demapper at the next iteration.

0090-6778 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Exeter. Downloaded on June 18,2020 at 15:32:29 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCOMM.2020.3002915, IEEE Transactions on Communications

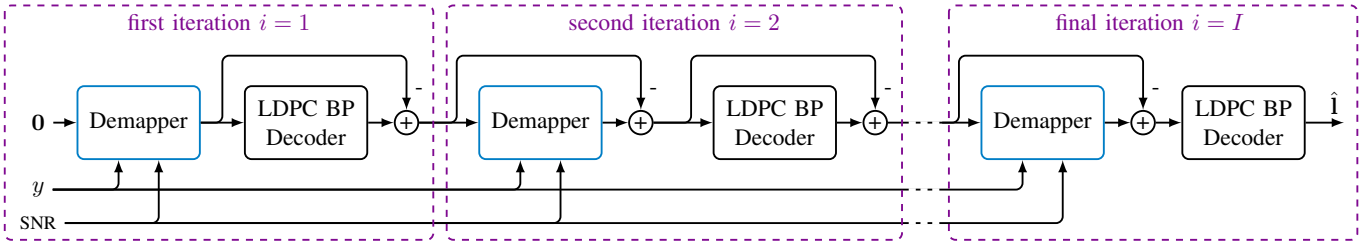


Fig. 7: Computational graph of the unfolded IDD receiver.

As done in [19] with conventional BP, the iterative algorithm depicted in Algorithm 1 is unfolded. This is illustrated in Fig. 7, where the first two iterations and the final iteration are shown, while intermediate iterations are straightforward and are therefore indicated by dashed lines. The so obtained end-to-end system forms a feedforward NN, which contains only differentiable operations. Therefore, it can be trained in an end-to-end manner with usual backpropagation. By doing so, the constellation, labeling, and demapper are jointly optimized for the considered channel and number of iterations. Although Algorithm 1 uses the same trainable demapper $g_{\boldsymbol{\theta_D}}$ in each iteration, one could also consider a different demapper in each iteration. Additionally, one could enrich the BP equations with trainable weights, as done in [19]. Also lower-complexity BP variants, such as min-sum decoding, can be used as long as they are differentiable.

Compared to the bit-wise autoencoder introduced in the previous section, which operates on individual complex baseband symbols carrying m bits each, the autoencoder extended to IDD operates on entire codewords of n bits. The loss (6) can be extended to such a system by

$$\sum_{j=1}^{n} H\left(p_{\boldsymbol{\theta_M}}(c_j|\mathbf{y}), \widetilde{p}_{\boldsymbol{\theta_D}}(c_j|\mathbf{y})\right) \tag{12}$$

where summation is performed over the bits forming a codeword, and $\mathbf{y} = [y_1, \ldots, y_s]^{\mathsf{T}}$. The value of a single bit c_j depends on multiple baseband samples as channel decoding is considered as part of the autoencoder. It was experimentally observed that training the end-to-end system by minimizing (12) leads to poor performance. Therefore, the following loss is used, which is obtained by summing the binary CE computed at the output of the demapper over all iterations:

$$\mathcal{J}(\boldsymbol{\theta_M}, \boldsymbol{\theta_D}) := \sum_{i=1}^{I}\sum_{j=1}^{n} H\left(p_{\boldsymbol{\theta_M}}(c_j|\mathbf{y}), \widetilde{p}^{[i]}_{\boldsymbol{\theta_D}}(c_j|\mathbf{y})\right) \tag{13}$$

$$= \sum_{i=1}^{I}\sum_{j=1}^{n} \mathbb{E}_{\mathbf{y},c_j}\left[-\log \widetilde{p}^{[i]}_{\boldsymbol{\theta_D}}(c_j|\mathbf{y})\right] \tag{14}$$

where $\widetilde{p}^{[i]}_{\boldsymbol{\theta_D}}(c_j|\mathbf{y})$ is the estimated posterior distribution at the ith iteration obtained by applying the sigmoid function to the logits generated by the demapper $\mathbf{l}^{[i]} = \left[\mathbf{l}^{(1)[i]\mathsf{T}}, \ldots, \mathbf{l}^{(s)[i]\mathsf{T}}\right]^{\mathsf{T}}$.

B. Simulation setup and results

The iterative bit-wise autoencoder with IDD was evaluated considering an AWGN channel and the same channel coding scheme as in Section II-B. Loop unrolling was applied to train through the differentiable IDD receiver as shown in Fig. 7. The number of iterations was set to $I = 40$, and the same set of weights was used in the demapper for all iterations. A benefit of this is that the number of weights does not increase with the number of iterations. As in II-B, the Adam optimization algorithm [16] was used and the learning rate was progressively decreased from 10^{-3} to 10^{-5} (see Tab I). The batch-size was set to 500 codewords.

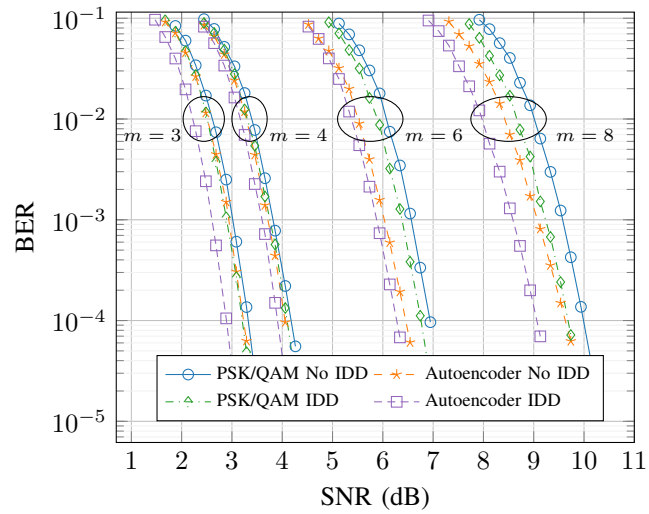


Fig. 8: BER achieved by PSK/QAM constellations with and without IDD and the bit-wise autoencoder with IDD for $m = 3,\ 4,\ 6$, and 8. The baseline is PSK for $m = 3$, and QAM otherwise.

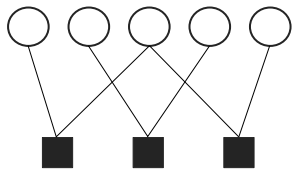
Fig. 8 shows the BER achieved by PSK/QAM and the bit-wise autoencoder with and without IDD. Unsurprisingly, IDD improves the performance of both the baseline and the autoencoder. Moreover, one can see that the autoencoder achieves significant gains over standard modulations when considering IDD. Interestingly, the autoencoder without IDD achieves BERs lower than the ones of the baseline with IDD for high modulation orders (64 ($m = 6$) and 256 ($m = 8$)).

IV. Code optimization

The previous sections considered an LDPC code from the 802.11n standard [15]. This section investigates how further gains can be achieved by optimization of the code after the system is trained. As we aim to communicate over arbitrary channels, off-the-shelf codes are not necessarily optimal and,

0090-6778 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Exeter. Downloaded on June 18,2020 at 15:32:29 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCOMM.2020.3002915, IEEE Transactions on Communications



(a) A Tanner graph: Circles depict VNs and squares CNs, respectively

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix}$$

(b) Parity-check matrix equivalent to the Tanner graph in (a)

Fig. 9: A Tanner graph and its corresponding parity-check matrix

as such, may limit the achievable overall performance. An LDPC code can be described by a Tanner graph which is composed of two types of nodes, variable nodes (VNs) and check nodes (CNs), representing bits and parity-check constraints, respectively. An exemplary Tanner graph and its equivalent parity-check matrix are shown in Fig. 9. The node degree corresponds to the number of edges that are connected to a node. In this example, all CNs have a degree of two, while the VNs have a degree of either one or two. The degree distribution considering all nodes is referred to as degree profile.

The conventional LDPC code design method (e.g., [20],[21]) is applied in this section to materialize the improved IDD performance. This approach is based on the construction of ensembles of codes whose average behavior approaches the fundamental Shannon limit [22]. An ensemble is characterized by its degree profile, i.e., codes from an ensemble share the same VN and CN degree distributions. Hereby, the optimization of ensembles is motivated by the concentration-around-ensemble-average property [23], which states that, as the length of codewords increases, codes from an ensemble tend to behave close to the ensemble average. The whole task of the code optimization is to find a degree distribution that allows successful iterative decoding while maximizing the rate of the code.

As usually done, we introduce the polynomials $\rho(z) = \sum_{j\geq 1} \rho_j z^{j-1}$ and $\lambda(z) = \sum_{i\geq 1} \lambda_i z^{i-1}$, where ρ_j and λ_i denote the fractions of edges that connect to a CN of degree j and a VN of degree i, respectively. Also describing the same code ensemble as the previously introduced *node degree distribution*, the edge perspective simplifies the later analysis. Note that $\forall j, \rho_j \geq 0, \forall i, \lambda_i \geq 0$, and $\sum_{j\geq 1} \rho_j = \sum_{i\geq 1} \lambda_i = 1$. Further, the average VN degree v_{avg} can be calculated as $v_{\text{avg}} = 1/\int_0^1 \lambda(x)\mathrm{d}x = 1/\sum_{i\geq 2} \frac{\lambda_i}{i}$ and the average CN degree c_{avg} can be calculated as $c_{\text{avg}} = 1/\sum_{j\geq 2} \frac{\rho_j}{j}$, respectively (see [23]). A well-performing ensemble is constructed by optimizing ρ and λ, such that the design rate

$$r = 1 - \frac{v_{\text{avg}}}{c_{\text{avg}}} = 1 - \frac{\sum_{j\geq 2} \frac{\rho_j}{j}}{\sum_{i\geq 2} \frac{\lambda_i}{i}} \tag{15}$$

is maximized, while BP decoding still converges to a zero average bit error probability. This is achieved using the extrinsic information transfer (EXIT) chart approach [21]. With this approach, it is assumed that the LLRs fed to the demapper as prior knowledge, denoted by $\hat{\mathbf{l}}_E$, as well as those fed to the decoder are Gaussian distributed. This is a common and valid assumption considering a large enough number of iterations and sufficiently long codewords.

A. EXIT analysis

Recall that for iterative receiver implementations the extrinsic information denotes the new information gained by a receiver component (e.g., channel decoder, due to exploiting redundancy of the code) using *a priori* information and, if available (e.g. for demapper, detection front-ends) the channel observations. For the concept of iterative receivers (cf. *Turbo* principle [18]), only the extrinsic information per component should be forwarded. However, if we directly maximize the output mutual information (MI) of the autoencoder as in Sec. III-A, the resulting MI violates this condition as the a priori information will be part of the demapper's output. Thus, we explicitly subtract the a priori knowledge in the IDD setup in Fig. 6. A theoretical derivation of the demapper's extrinsic information becomes mathematically intractable due to the trainable weights in the receiver and we have to rely on Monte-Carlo sampling of the mutual information. This procedure will be described in the following and we refer to [18], [24] for details.

We are interested in the MI between the random variable associated with the input bit sequence B and another random variable of their corresponding LLR representations L_E at the demapper's output. Unfortunately, this requires knowledge of the a posterior distribution $p(L_E|B = b)$ (see [25] for details) which could be approximated by measuring the corresponding histograms, or, more simply, by the following approximation from [18]

$$I(B; L_E) = \left(1 - \frac{1}{n}\sum_{i=1}^{n}\left[\log_2\left(1 + e^{-l_{E,i}(-2b_i+1)}\right)\right]\right)$$

where n denotes a large enough amount of samples.

For the EXIT-chart analysis, it is convenient to assume the transmission of the all-zero codeword. However, this is only valid for symmetric channels which is obviously not the case for autoencoder-based transmission, i.e., in general $p(Y = y|c = 0) \neq p(Y = -y|c = 1)$. We ensure symmetric channels (during code design) by introducing a pseudo random scrambling sequence known to both transmitter and receiver, e.g., see [26]. This essentially means that random bits $\mathbf{c}$ are transmitted, but at the receiver (i.e., the demapper output) the signs of the corresponding LLRs $\mathbf{l}_E$ are flipped if $c_i = 1$ (as if the all-zero codeword was transmitted). Note that, for readability, this detail is omitted in the following and the code optimization is explained as if the all-zero codeword was transmitted.

To make the mutual information accessible for Monte-Carlo simulations, the relation between mutual information and LLRs is required. Under the all-zero codeword assumption, the mutual information shared between the transmitted bits

0090-6778 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Exeter. Downloaded on June 18,2020 at 15:32:29 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCOMM.2020.3002915, IEEE Transactions on Communications

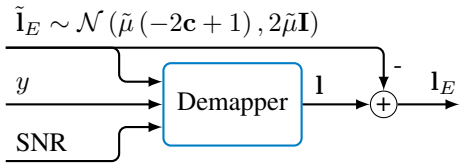


Fig. 10: EXIT-analysis of the demapper assuming Gaussian distributed a priori LLRs $\tilde{\mathbf{l}}_E$

and Gaussian distributed LLRs with the mean value μ (and variance 2μ [21]) can be quantified by

$$J(\mu) = 1 - \int_{-\infty}^{\infty} \frac{e^{-(\tau-\mu)^2/(4\mu)}}{\sqrt{4\pi\mu}} \log(1+e^{-\tau}) d\tau \tag{16}$$

thus, leading to $I(B_j; L_j) = J\left(\mathbb{E}_{l_j}\left[l_j\right]\right)$. Note that $J(.)$ takes values in the interval $(0,1)$, and a numerical approximation is given in [21].

EXIT chart-based *degree profile matching* requires knowledge of the mutual information between the output LLRs of the demapper and the transmitted bits[3] which can be simulated via Monte-Carlo methods as shown later. In the case of IDD, the demapper observes y and additionally $\hat{\mathbf{l}}_E$ as side information (so-called *a priori knowledge* denoted by the mutual information $\tilde{I}_A$). However, for the code design phase (all-zero codeword) we replace $\hat{\mathbf{l}}_E$ by Gaussian distributed LLRs $\tilde{\mathbf{l}}_E$ such that the mean of these a priori LLRs becomes $\tilde{\mu} = J^{-1}(\tilde{I}_A)$ (see [21] for a numerical approximation), which, again, is much simpler to simulate. The demapper's output is a vector with soft-estimates on all m transmitted bits. However, in the BICM case, these m estimates are treated equally and the demapper's output can be described by the mean value of its output LLRs $\mathbf{l}_E$.

The demapper's EXIT characteristic is denoted by $T(.)$ and describes the input/output relation of the *extrinsic* mutual information of the iterative demapper [21], i.e., for a given a priori mutual information, how much (additional) extrinsic information the demapper can contribute by observing the channel output. This EXIT characteristic is defined[4] as

$$\begin{aligned} &T\left(\mathrm{SNR}, \tilde{\mu} = J^{-1}\left(\tilde{I}_A\right)\right) \\ &\qquad = J^{-1}\left(\frac{1}{m}\sum_{i=1}^{m} I\left(B_i; L_{E,i}|\tilde{L}_{E,i}\right)\right) \end{aligned} \tag{17}$$

where $L_{E,i}$ denotes the demapper's extrinsic soft estimate on the *ith* bit and $\tilde{L}_{E,i}$ is chosen such that $\tilde{I}_A = I(B_i; \tilde{L}_{E,i})$. This can be achieved by sampling such that $\tilde{\mathbf{l}}_E \sim \mathcal{N}\left(\tilde{\mu}\left(-2\mathbf{c}+1\right), 2\tilde{\mu}\mathbf{I}\right)$ as shown in Fig. 10. Note, that from code design perspective, this transfer function $T(.)$ characterizes the *channel front-end*, i.e., $T(.)$ depends on the trained systems so that the optimum code can only be defined after training. As such, the training of the autoencoder impacts the optimal degree profile.

[3]We assume a random interleaver between demapper and decoder.

[4]For readability, we define $T(\mathrm{SNR}, \mu)$ as the input/output relation of the mean of the LLRs. However, this directly relates to the mutual information via the $J(.)$-function.

A closed form solution for $T(.)$ is difficult to find, however, for a fixed SNR and given $\tilde{I}_A$, Monte Carlo-based approaches are possible based on measured LLRs $l_{E,i}$. We follow [18] for a numerical approximation of the mutual information for simulated LLRs $l_{E,i}$ leading to

$$\begin{aligned} &T\left(\mathrm{SNR}, J^{-1}\left(\tilde{I}_A\right)\right) \\ &\quad \approx J^{-1}\left(1 - \frac{1}{n}\sum_{i=1}^{n}\left[\log_2\left(1+e^{-l_{E,i}(-2c_i+1)}\right)\right]\right) \end{aligned} \tag{18}$$

where c_i denotes the codeword bit corresponding to the LLR $l_{E,i}$ and n is sufficiently large. For the $J(.)$ function, we use the numerical approximation from [21]. This turns out to be more precise in non-Gaussian scenarios than the naive mean-based estimation.

B. Code design method

Following the conventional method (e.g., [20],[21]), optimization of the code ensemble can be reduced to a linear optimization problem by fixing one of the two polynomials defining its degree profile and optimizing the respective other (cf. [27]). We choose a check-concentrated degree distribution $\rho_c = 1$ for some integer c, i.e., a CN-regular code of degree c (e.g., $c = 2$ in Fig. 9a). Therefore, maximizing the design rate (15) is equivalent to maximizing $\sum_{i=2}^{d} \frac{\lambda_i}{i}$, where d is the maximum VN degree. Optimization of the VN degree distribution is performed by solving the following optimization problem:

$$\begin{aligned} \underset{\lambda_2,\ldots,\lambda_d}{\text{maximize}} \quad & \sum_{i=2}^{d} \frac{\lambda_i}{i} \\ \text{subject to} \quad & \lambda_i \geq 0, \ \forall i \in \{2,\ldots,d\} \\ & \sum_{i=2}^{d} \lambda_i = 1 \\ & \sum_{i=2}^{d} \lambda_i J\left(T\left(\mathrm{SNR}, J^{-1}\left(h\right)\right) + (i-1)J^{-1}\left(h\right)\right) \\ & > 1 - J\left(\frac{1}{c-1}J^{-1}\left(1-h\right)\right), \ \forall h \in (0,1) \end{aligned} \tag{19a}$$

$$\lambda_2 \leq \beta \tag{19b}$$

The constraint (19a) guarantees convergence to a zero average bit error probability (see, e.g., [20] for more details). The constrained (19b) provides a stability condition on λ_2 to ensure convergence close to the point $(1,1)$ in the EXIT chart as intersections between the VN and CN EXIT curves are numerically hard to track in this region. However, these intersections may translate into a non-negligible error-floor or BER performance degradation. For the AWGN case, it can be shown that $\beta \leq \frac{\exp\left(\frac{1}{N_0}\right)}{c-1}$. However, β also allows to generally control the amount of degree-2 VNs to simplify the later code construction phase as these degree-2 nodes require a careful (and complex) placement to avoid high error-floors.

0090-6778 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Exeter. Downloaded on June 18,2020 at 15:32:29 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCOMM.2020.3002915, IEEE Transactions on Communications

Algorithm 2: Code design algorithm

Input : Initial SNR, $r^*, \Delta_{\text{SNR}}, \epsilon, c, D$
Output: $\lambda_2^*, \ldots, \lambda_v^*$

1 **repeat**
2 Sample: B codewords $\mathbf{c}^{(1)}, \ldots, \mathbf{c}^{(B)}$
3 Sample: corresponding observations $\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(B)}$
4 **For** $i = 1, \ldots, D$ **do**
5 $\tilde{\mu} \leftarrow J^{-1}(\frac{i}{D})$
6 Sample: $\tilde{\mathbf{l}}_E^{(1)}, \ldots, \tilde{\mathbf{l}}_E^{(B)}$, $\tilde{\mathbf{l}}_E^{(j)} \sim \mathcal{N}\left(\tilde{\mu}\left(-2\mathbf{c}^{(j)}+1\right), 2\tilde{\mu}\mathbf{I}\right)$
7 Estimate $T\left(\text{SNR}, J^{-1}\left(\frac{i}{D}\right)\right) \approx$
8 $J^{-1}\left(1 - \frac{1}{Bn}\sum_{k=1}^{B}\sum_{j=1}^{n}\left[\log_2\left(1 + e^{-l_{E,j}^{(k)}\cdot\left(-2c_j^{(k)}+1\right)}\right)\right]\right)$
9 **end**
10 Set $\lambda_2, \ldots, \lambda_v$ by solving (19)
11 $r \leftarrow 1 - \frac{1}{c\sum_{j=2}^{v}\frac{\lambda_j}{j}}$
12 **if** $r > r^*$ **then**
13 SNR $\leftarrow$ SNR $- \Delta_{\text{SNR}}$
14 $\lambda_2^*, \ldots, \lambda_v^* \leftarrow \lambda_2, \ldots, \lambda_v$
15 **else**
16 SNR $\leftarrow$ SNR $+ \Delta_{\text{SNR}}$
17 $\Delta_{\text{SNR}} \leftarrow \frac{\Delta_{\text{SNR}}}{2}$
18 **if** $\Delta_{SNR} < \epsilon$ **then**
19 **return** $\lambda_2^*, \ldots, \lambda_v^*$

In practice, the optimization problem (19) is made tractable by uniformly discretizing the interval $(0, 1)$ with D values. Thus, (19a) is replaced by the set of constraints

$$\sum_{i=2}^{d}\lambda_i J\left(T\left(\text{SNR}, J^{-1}\left(\frac{j}{D}\right)\right) + (i-1)J^{-1}\left(\frac{j}{D}\right)\right) > 1 - J\left(\frac{1}{c-1}J^{-1}\left(1-\frac{j}{D}\right)\right),\ j = 0, \ldots, D-1. \quad (20)$$

Moreover, because $T(.)$ depends on the SNR, (19) is specific to a given SNR value. However, one wants to find a code which achieves a desired target rate denoted by r^* with the lowest possible SNR. To achieve this goal, (19) is repeatedly solved until the SNR cannot be decreased without achieving a rate lower than r^*, as shown in Algorithm 2. In this algorithm, $T(.)$ is numerically estimated by the previously introduced Monte-Carlo approach (lines 2 to 8) for a fixed SNR. Using this estimation, the optimization problem (19), with (19a) is replaced by (20), is solved using a conventional linear solver (line 10). The rate of the so-obtained solution $\lambda(z)$ is computed (line 11). As (19) aims at finding the degree profile $\lambda(z)$ that maximizes the rate for the given SNR and under the constraint that successful decoding is possible (constraint (19a) and (19b)), this solution provides the largest possible rate for the given SNR, c and demapper $T(.)$. If the rate is higher than the targeted rate r^*, the SNR is decreased for the next iteration by a predefined amount Δ_{SNR} (lines 12 to 14). Otherwise, it is increased (lines 15 and 16). This describes a simple bisection search in a certain SNR interval. The optimization is repeated until Δ_{SNR} is below a predefined threshold ϵ.[5] The same algorithm can be repeated for multiple values of c until the best threshold is achieved.

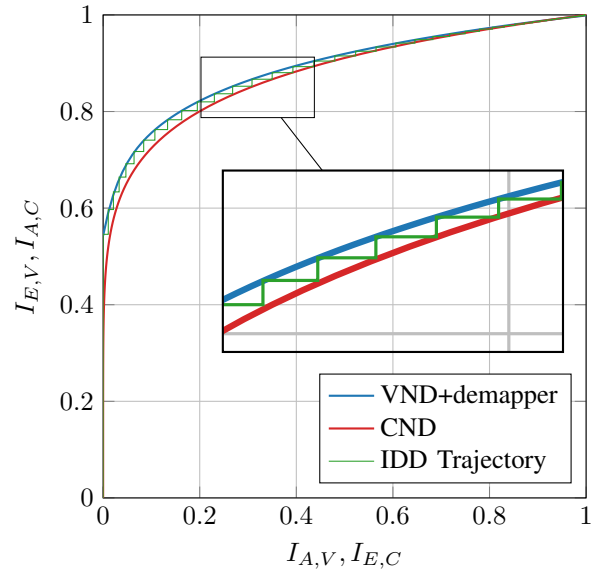


Fig. 11: EXIT chart of an optimized LDPC code including the actual trajectory during decoding for the bit-wise autoencoder with $m = 6$. The schemes use IDD with 80 iterations.

Once the ensemble degree distribution is optimized, a single code from this ensemble is selected using conventional code construction methods (see, e.g., [20]).

C. Evaluation

We use a progressive edge-growth (PEG)-based code construction [28] to ensure that an optimized decoding threshold also translates into an improved BER performance. The intuition behind PEG is to maximize the *girth*, i.e., to minimize short cycles in the graph and, thus, to avoid a performance degradation due to correlated messages during decoding. The baseline uses the conventional 802.11n code of the same length of $n = 1944$ bits. Further, we limit the amount of degree-2 VNs to be less than the number of CNs (e.g., in this case $\beta = 0.26$). This helps to reduce error-floors at the price of a slightly decreased waterfall performance. However, in particular for short (to medium) code lengths, the explicit graph construction becomes a challenging task.

Fig. 11 shows the corresponding EXIT-chart of the optimized code and the actual decoding trajectories during decoding. Note that $I_{A,V}, I_{A,C}$ and $I_{E,V}, I_{E,C}$ denote the a priori and extrinsic mutual information of the VN and CN, respectively. The VN and CN curves can be analytically described for a given ensemble (i.e., degree profile; see [21]) and describe the input/output characteristic of the extrinsic mutual information of the VN decoder (+demapper) and the CN decoder update, respectively. For a given input mutual information I_A, the curves trace how much new (i.e., extrinsic)

[5]Note that the algorithm could be modified such that the SNR is constant and the rate is maximized; depending on the exact setup, this can be beneficial, e.g., for transmissions where the SNR is given.

0090-6778 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Exeter. Downloaded on June 18,2020 at 15:32:29 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCOMM.2020.3002915, IEEE Transactions on Communications

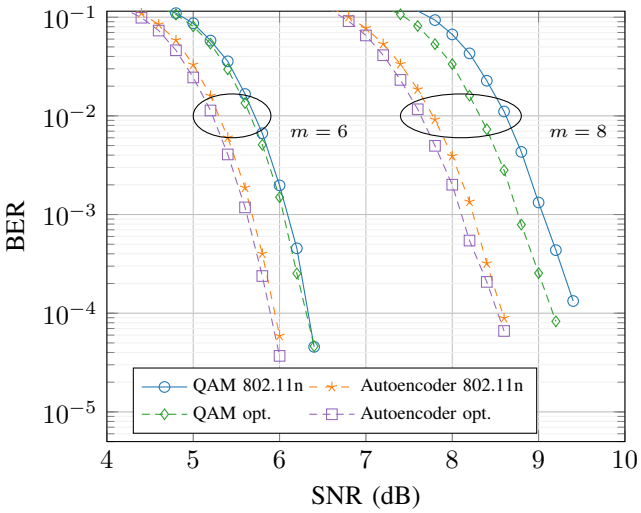


Fig. 12: BER achieved by baseline QAM constellations and the bit-wise autoencoder for $m = 6$ and $m = 8$ bits per symbol. We compare between 802.11n LDPC codes [15] and optimized LDPC codes of the same length $n = 1944$. All schemes use IDD with 80 iterations.

mutual information I_E can be gained with an VN or CN decoding update, respectively. If the VN and CN curves intersect, no successful decoding is possible as the iterative update of VNs and CNs does not generate sufficient extrinsic knowledge to progress convergence. However, the plotted trajectory is the actual simulated performance during decoding. As can be seen, the iterative decoding trajectory step-wise tracks the predicted curves and, thus, follows its expected behavior. Further, it can be seen that the EXIT-curves are well-matched while maintaining an open tunnel during all decoding iterations, verifying the effectiveness of our code optimization.

Fig. 12 shows the BER performance of the optimized code with maximum VN degree $v_{max} = 12$ for an AWGN channel and for the 802.11n code of length $n = 1944$ as a baseline. For both codes, both a QAM constellation and, as comparison, the bit-autoencoder for $m = 6$ and $m = 8$ are used. The code is optimized for the QAM and autoencoder individually to ensure the best performance of each system. Note that, for this canonical scenario, the baseline already provides a well-designed coding scheme. Interestingly, also the baseline benefits from the code optimization which is plausible as the 802.11n code was not specifically designed for IDD receivers and, thus, can also be outperformed by a code carefully matched to the IDD receiver. However, as proof-of-concept, the proposed approach outperforms the baseline by approx. 0.1 dB for $m = 6$ and 0.2 dB for $m = 8$, respectively, but larger gains are expected whenever m increases, or when the channel becomes more sophisticated, as will become obvious in the next section.

V. Over-the-air experiments

This section shows how the previously described system can be trained over an actual channel. In previous sections, training of the autoencoder was performed considering an AWGN channel model. However, it was shown in [29] that training using a channel model leads to significant loss of performance once the system is deployed on the actual channel. This loss of performance was attributed to the unavoidable mismatch between the channel model and the actual channel, and was observed despite important efforts put on the conception of an accurate channel model. These results motivated the design of a training method for autoencoder-based communication systems in [14] that does not require a model of the channel. Moreover, this method can be used for training over channels with non-differentiable components, such as quantization.

A. Training over the actual channel

Following [14], training of the end-to-end system is done in an alternating fashion in which each iteration is composed of two stages, one for the receiver and one for the mapper. Training of the receiver, shown in Fig.13a, does not require differentiation of the channel and is therefore conventional.

First, a batch of codewords is sampled, and each codeword $\mathbf{c} = \left[\mathbf{b}^{(1)^\mathsf{T}}, \ldots, \mathbf{b}^{(s)^\mathsf{T}}\right]^\mathsf{T}$ is mapped onto a vector of channel symbols $\mathbf{x} = f_{\boldsymbol{\theta}_M}(\mathbf{c}) = \left[f_{\boldsymbol{\theta}_M}(\mathbf{b}^{(1)}), \ldots, f_{\boldsymbol{\theta}_M}(\mathbf{b}^{(s)})\right]^\mathsf{T}$. Thus, $\mathbf{x}$ is sent over the channel and the received symbols $\mathbf{y}$ are fed to the IDD-based receiver, generating LLRs $\hat{\mathbf{l}}$. Training is done using stochastic gradient descent (SGD) on the loss (14). At training, it is assumed that the receiver has knowledge of the transmitted codewords $\mathbf{c}$ which can be achieved, e.g., through pseudorandom number generators initialized with the same seed on both ends.

Training of the transmitter is less straightforward as differentiation of the channel is not possible through conventional backpropagation. Inspired by reinforcement learning, the method introduced in [14] relies on the addition of random perturbations w to the output of the transmitter at training. This approach enables approximation of the gradient of the loss function with respect to (w.r.t.) the transmitter parameters - despite the lack of a channel model. From the viewpoint of reinforcement learning the addition of perturbations enables exploration of neighboring solutions that may lead to better performance. Formally, the transmitter output is relaxed to a normal distribution by adding a zero-mean Gaussian perturbation $w \sim \mathcal{CN}(\mathbf{0}, \sigma_w^2)$

$$x = \sqrt{1 - \sigma_w^2} f_{\boldsymbol{\theta}_M}(\mathbf{b}) + w \tag{21}$$

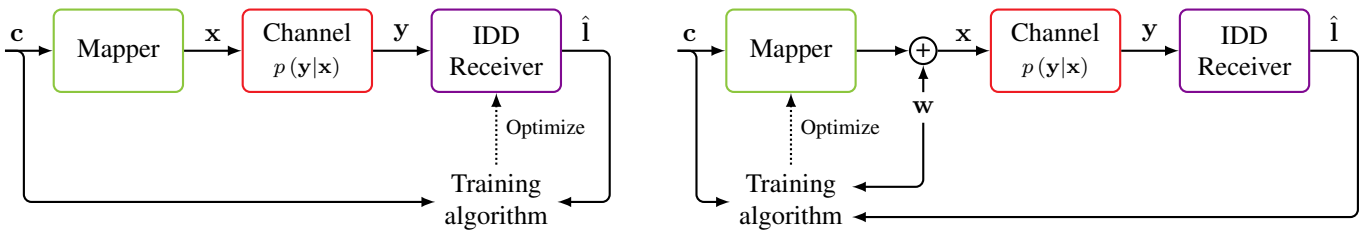
where scaling is done to ensure the conservation of the average energy, and σ_w must be in the interval $(0, 1)$. Training of the transmitter is performed on the loss

$$\widehat{\mathcal{J}}(\boldsymbol{\theta}_M) \triangleq \sum_{i=1}^{I} \sum_{j=1}^{n} \mathbb{E}_{c_j, \mathbf{w}, \mathbf{y}} \left[-\log \widetilde{p}_{\boldsymbol{\theta}_D}^{[i]}(c_j | \mathbf{y}) \right] \tag{22}$$

instead of (14), where $\mathbf{w} = [w_1, \ldots, w_S]^\mathsf{T}$ is the vector of independent and identically distributed (i.i.d.) perturbations added to the baseband symbols generated by the transmitter for a given codeword. Note that, compared to (14), the expected

0090-6778 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Exeter. Downloaded on June 18,2020 at 15:32:29 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCOMM.2020.3002915, IEEE Transactions on Communications



(a) Training of the receiver

(b) Training of the mapper without channel model

Fig. 13: Training of the end-to-end system without a channel model is done by alternating between (a) conventional training of the receiver, and (b) training of the mapper using a method inspired by reinforcement learning.

value is also over the perturbations $\mathbf{w}$, as $\mathbf{y} \sim p(\mathbf{y}|f_{\boldsymbol{\theta}_M}(\mathbf{c}) + \mathbf{w})$. The gradient of $\widehat{\mathcal{J}}$ is estimated by

$$\nabla_{\boldsymbol{\theta}_M} \widehat{\mathcal{J}}(\boldsymbol{\theta}_M) \approx \frac{1}{|\mathcal{V}|} \sum_{\mathbf{c}\in\mathcal{V}} \sum_{j=1}^{n} \sum_{i=1}^{I} \Bigg(\frac{2\sqrt{1-\sigma_w^2}}{\sigma_w^2} \cdot \Big(c_j \log\Big(\widetilde{p}^{[i]}_{\boldsymbol{\theta}_D}(c_j|\mathbf{y})\Big) + (1-c_j)\log\Big(1-\widetilde{p}^{[i]}_{\boldsymbol{\theta}_D}(c_j|\mathbf{y})\Big)\Big) \cdot \nabla_{\boldsymbol{\theta}_M} f_{\boldsymbol{\theta}_M}(\mathbf{c})\mathbf{w} \Bigg) \quad (23)$$

where $\mathcal{V}$ is a finite set of samples of codewords, $\nabla_{\boldsymbol{\theta}_M} f_{\boldsymbol{\theta}_M}(\mathbf{c}) = \left[\nabla_{\boldsymbol{\theta}_M} f_{\boldsymbol{\theta}_M}(\mathbf{b}^{(1)}), \ldots, \nabla_{\boldsymbol{\theta}_M} f_{\boldsymbol{\theta}_M}(\mathbf{b}^{(s)})\right]$, and the so called "log-trick" $\nabla_{\mathbf{x}} \log p(\mathbf{y}|\mathbf{x}) = \frac{\nabla_{\mathbf{x}} p(\mathbf{y}|\mathbf{x})}{p(\mathbf{y}|\mathbf{x})}$ was used. It was shown in [14] that in cases where the channel is differentiable, $\nabla_{\boldsymbol{\theta}_M} \widehat{\mathcal{J}}(\boldsymbol{\theta}_M)$ converges to the gradient of (14) $\nabla_{\boldsymbol{\theta}_M} \mathcal{J}(\boldsymbol{\theta}_M)$ as σ_w goes to zero. Unfortunately, reducing σ_w leads to higher variance of the gradient estimator (23), causing slow convergence. Therefore, σ_w controls a tradeoff between the accuracy of the loss function gradient approximation and the estimator variance.

Figure 13b shows the training of the mapper. Each codeword $\mathbf{c} \in \mathcal{V}$ is mapped to a vector of baseband symbols $f_{\boldsymbol{\theta}_M}(\mathbf{c})$. A perturbation vector $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \sigma_w^2 \mathbf{I})$ is added to $f_{\boldsymbol{\theta}_M}(\mathbf{c})$ to form the vector of symbols $\mathbf{x}$ that is sent over the channel. The received vector of symbols $\mathbf{y}$ is demapped and decoded by the IDD-based receiver, and the so-obtained LLRs $\hat{\mathbf{l}}$ are sent to the mapper using a reliable link required only at training. Finally, one step of optimization is performed with the loss gradient estimated by (23). By alternating between a few steps of receiver training (Fig. 13a) and a few steps of transmitter training (Fig. 13b), optimization of the end-to-end system is carried out - despite the lack of a channel model.

B. Experimental results

A wireless communication system consisting of two Universal Software Radio Peripherals (USRPs) B210 from Ettus Research with carrier-frequency of 2.35 GHz and an effective bandwidth of 15.94 MHz was trained in a static indoor office environment. An Orthogonal Frequency-Division Multiplexing (OFDM)-based autoencoder with cyclic prefix (CP) of ratio $1/8$ and 64 subcarriers (50 of which are used for data transmission) as introduced in [30] was considered. The channel was mostly frequency flat with varying SNR between sub-carriers of at most $\approx$ 2dB. Although, the OFDM framework was only used for synchronization and the CP was not accessible to the NN. From the autoencoder's point of view it was operating as a single carrier system (within a sub-carrier of the OFDM framework) and other sub-carriers and their channel conditions were unknown to it. Linear minimum mean square error (MMSE) equalization of the received symbols was performed on a per-subcarrier-basis prior to NN-based demapping.[6] We did not compensate for any other potential analog or digital impairments such as distortion, quantization or clipping. To estimate the SNR, we first calculate the error vector magnitude (EVM) between the originally sent symbols and the equalized received symbols. We then calculate an average SNR per subcarrier, defined by the mean over the EVM, and feed this SNR estimation to the demapper of the corresponding sub-

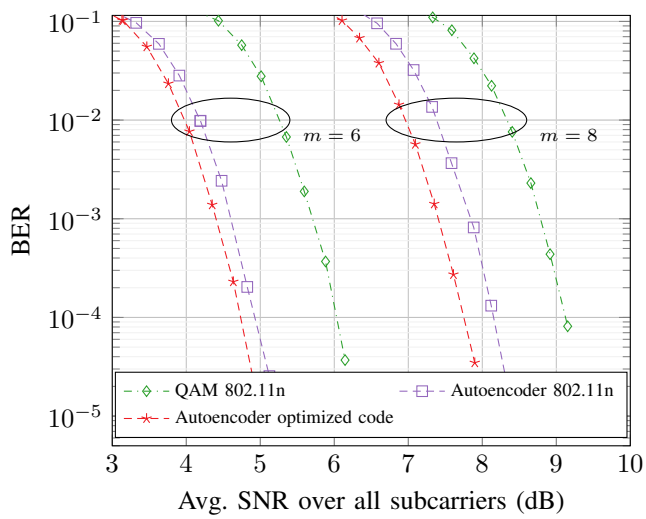
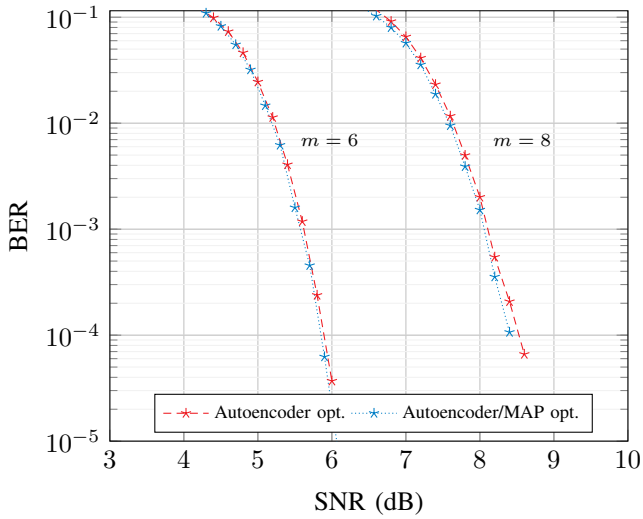


Fig. 14: Actual over-the-air (OTA) BER achieved by baseline QAM constellations and the bit-wise autoencoder for $m = 6$ and 8. The baseline uses the 802.11n code of same length $n = 1944$ as the optimized (opt.) code (see Section IV). The SNR is averaged over all subcarriers of the OFDM system. All schemes use IDD with 80 iterations.

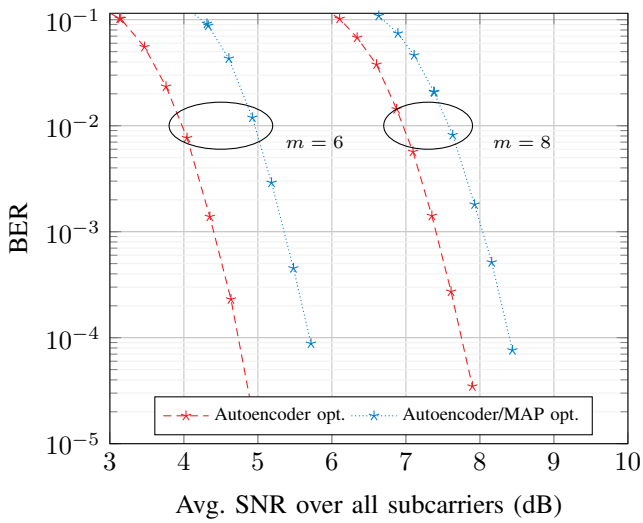
[6] In [30], it has been shown that the autoencoder can learn MMSE equalization, however, this requires more than one complex-valued channel use. For simplicity and a fair comparison with the QAM-baseline, this case is not considered here, yet an extension is straightforward.

0090-6778 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Exeter. Downloaded on June 18,2020 at 15:32:29 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCOMM.2020.3002915, IEEE Transactions on Communications



(a) Simulated AWGN channel



(b) Actual over-the-air channel

Fig. 15: Comparison between autoencoder/MAP, using a maximum a posteriori (MAP) demapper, and a pure autoencoder, using an NN-based (learned) demapper. The upper plot shows the BER for the simulated AWGN channel and the lower plot shows the OTA BER achieved for $m = 6$ and $m = 8$. The SNR is averaged over all subcarriers of the OFDM system for the OTA results. All schemes use IDD with 80 iterations.

carrier. To perform code optimization (see Section IV), the EXIT curves were measured based on data from actual over-the-air transmissions. The NN parameters were the same as in the previous sections, and σ_w^2 was progressively decreased from 0.05 to 0.01 at training. The only exception is that we did not feed the current SNR to the autoencoder encoder, as this would require a precise SNR sounding prior to the actual transmission. Instead for all OTA results, we use a single fixed constellation trained within a certain SNR range centered around the LDPC's waterfall region.

The obtained BER curves are shown in Fig. 14. Obviously, the autoencoder maintains its superiority when compared to the QAM baselines and can even increase the performance gap to approximately 1.0 dB. Code optimization provides further improvements of 0.2 dB and 0.4 dB for $m = 6$ and 8, respectively. This can be intuitively explained by the fact that the 802.11n LDPC code was optimized for an AWGN channel, while for OTA training, different constellations (and, thus, different demapper characteristics) may turn out to be optimal.

Next, we will focus on the reasons of why a trained system outperforms the *classical* baseline over the actual channel. Let us quickly recap the reasons for potential performance gains which are superimposed in Fig. 14:

1) Optimized constellations. This typically requires IDD to materialize the full performance gains.
2) An optimal demapper for the actual channel. For the OTA channel the AWGN-MAP demapper is a *mismatched* demapper and, thus, sub-optimal for the actual channel. The AE can learn the underlying channel transition probabilities and, thereby, provides better LLR estimates (i.e., closer to the actual true *a posteriori probability*).
3) BER gains due to the *matched* LDPC code, i.e., the IDD receiver can recover more information from the received sequence.

We compare the BER achieved by an explicit AWGN-MAP demapper for IDD (see [9]), only optimal for the AWGN channel, and the learned autoencoder constellation as shown in Fig 15. Note that, this requires knowledge of the underlying channel statistics and only serves as an optimal baseline for the canonical AWGN channel, i.e., it provides a lower bound of the achievable AWGN performance. It can be seen that on the AWGN, the autoencoder-based receiver virtually approaches the same BER performance, and, thus, can be claimed to be optimal for the given AWGN channel. For the OTA transmission, the learned demapper yields a gain of 0.6 dB compared to the MAP demapper for the AWGN channel period.[7]. This can be intuitively explained as even after equalization, the actual channel is not necessarily AWGN, resulting in suboptimal performance for the MAP demapper when an AWGN channel is assumed. The learned demapper was, on the other hand, optimized on the actual channel, leading to its superior BER performance.

VI. Conclusion

We have described and experimentally validated training-based optimization of the physical layer of a point-to-point communication system, including an outer channel code. The key novelty is training on the bit-wise mutual information which enables seamless integration with BMD receivers, widely used in practice. The performance improvements over classic QAM baselines result from geometric constellation

[7]To account for alterations after deployment, it is possible to continuously finetune the autoencoder demapper by a simple supervised training step as shown in [31] The labels for such a re-training can be directly taken from the BP decoder's output $\hat{\mathbf{I}}$ which can be assumed to be (nearly) error-free for a typical operation point. Thereby, continuous receiver finetuning without any additional piloting overhead is possible.

0090-6778 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Exeter. Downloaded on June 18,2020 at 15:32:29 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCOMM.2020.3002915, IEEE Transactions on Communications

shaping as well as learning of the optimal demapper. Further gains can be achieved through iterative demapping and decoding which has been elegantly integrated into the end-to-end learning procedure. Once the system is trained, LDPC code optimization can lead to additional improvements.

Simulations as well as over-the-air experiments, in which the system was optimized from scratch over an actual wireless channel, have demonstrated the universality and practical benefits of the proposed method. The trained system outperforms a 256-QAM baseline with an 802.11n LDPC code in a realistic setup by approximately 1.3 dB while maintaining the same communication rate.

Looking into the future, we believe that learning-based optimization of the full physical layer for a point-to-point link could be completely automated and may be one of the key ingredients of next generation communication systems. Due to the necessary overhead for training, such an approach seems particularly attractive for fixed wireless or optical channels. Interesting directions for further research comprise learning of probabilistic shaping [5] as well as schemes for multiuser communications, i.e., multiple access and broadcast channels [32]. Also, the idea of meta learning for fast online learning seems very promising [33].

Acknowledgments

The authors would like to thank Laurent Schmalen for his help with the LDPC code design algorithms and Marc Gauger for his help with the OTA setup.

References

[1] T. O'Shea and J. Hoydis, "An Introduction to Deep Learning for the Physical Layer," *IEEE Trans. Cogn. Commun. Netw.*, vol. 3, no. 4, pp. 563–575, Dec. 2017.

[2] B. Karanov, M. Chagnon, F. Thouin, T. A. Eriksson, H. Bülow, D. Lavery, P. Bayvel, and L. Schmalen, "End-to-End Deep Learning of Optical Fiber Communications," *J. Lightw. Technol.*, vol. 36, no. 20, pp. 4843–4855, Oct. 2018.

[3] Z. Zhu, J. Zhang, R. Chen, and H. Yu, "Autoencoder-Based Transceiver Design for OWC Systems in Log-Normal Fading Channel," *IEEE Photon. J.*, vol. 11, no. 5, pp. 1–12, Oct. 2019.

[4] S. Mohamed, J. Dong, A. R. Junejo, and D. C. Zuo, "Model-Based: End-to-End Molecular Communication System Through Deep Reinforcement Learning Auto Encoder," *IEEE Access*, vol. 7, pp. 70 279–70 286, May 2019.

[5] M. Stark, F. Ait Aoudia, and J. Hoydis, "Joint Learning of Geometric and Probabilistic Constellation Shaping," in *Proc. IEEE Globecom Conf. Workshops (GC Wkshps)*, Dec. 2019.

[6] G. Caire, G. Taricco, and E. Biglieri, "Bit-interleaved Coded Modulation," *IEEE Trans. Inform. Theory*, vol. 44, no. 3, pp. 927–946, May 1998.

[7] G. Böcherer, F. Steiner, and P. Schulte, "Bandwidth Efficient and Rate-Matched Low-Density Parity-Check Coded Modulation," *IEEE Trans. Commun.*, vol. 63, no. 12, pp. 4651–4665, Dec. 2015.

[8] G. Böcherer, "Achievable Rates for Probabilistic Shaping," *arXiv preprint arXiv:1707.01134*, 2017.

[9] S. ten Brink, J. Speidel, and Ran-Hong Yan, "Iterative Demapping and Decoding for Multilevel Modulation," in *Proc. IEEE Globecom Conf.*, vol. 1, Nov. 1998, pp. 579–584.

[10] F. Alberge, "Deep Learning Constellation Design for the AWGN Channel with Additive Radar Interference," *IEEE Trans. Commun.*, vol. 67, no. 2, pp. 1413–1423, 2018.

[11] Y. He, M. Jiang, X. Ling, and C. Zhao, "Robust BICM Design for the LDPC Coded DCO-OFDM: A Deep Learning Approach," *IEEE Trans. Commun.*, pp. 1–1, 2019.

[12] T. Koike-Akino, Y. Wang, D. S. Millar, K. Kojima, and K. Parsons, "Neural Turbo Equalization: Deep Learning for Fiber-Optic Nonlinearity Compensation," *arXiv preprint arXiv:1911.10131*, 2019.

[13] N. S. Loghin, J. Zöllner, B. Mouhouche, D. Ansorregui, J. Kim, and S.-I. Park, "Non-uniform Constellations for ATSC 3.0," *IEEE Trans. Broadcast.*, vol. 62, no. 1, pp. 197–203, 2016.

[14] F. Ait Aoudia and J. Hoydis, "Model-Free Training of End-to-End Communication Systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 11, pp. 2503–2516, Nov 2019.

[15] "IEEE Standard for Information Technology – Local and Metropolitan Area Networks – Specific Requirements – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 5: Enhancements for Higher Throughput," *IEEE Std 802.11n-2009 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, and IEEE Std 802.11w-2009)*, pp. 1–565, Oct. 2009.

[16] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *preprint arXiv:1412.6980*, 2014.

[17] X. Glorot and Y. Bengio, "Understanding the Difficulty of Training Deep Feedforward Neural Networks," in *Proc. Int. Conf. AI and Statistics*, 2010, pp. 249–256.

[18] J. Hagenauer, "The Turbo Principle in Mobile Communications," in *Proc. Int. Symp. on Inform. Theory Appl.*, 2002.

[19] E. Nachmani, Y. Be'ery, and D. Burshtein, "Learning to Decode Linear Codes Using Deep Learning," in *Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sept. 2016, pp. 341–346.

[20] T. Richardson and R. Urbanke, *Modern Coding Theory*. Cambridge University Press, 2008.

[21] S. ten Brink, G. Kramer, and A. Ashikhmin, "Design of Low-density Parity-check Codes for Modulation and Detection," *IEEE Trans. Commun.*, vol. 52, no. 4, pp. 670–678, April 2004.

[22] S.-Y. Chung, G. D. Forney, T. J. Richardson, R. Urbanke *et al.*, "On the Design of Low-density Parity-check Codes within 0.0045 dB of the Shannon Limit," *IEEE Comm. Letters*, vol. 5, no. 2, pp. 58–60, 2001.

[23] T. J. Richardson and R. L. Urbanke, "The Capacity of Low-density Parity-check Codes under Message-passing Decoding," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 599–618, 2001.

[24] S. ten Brink, "Convergence Behavior of Iteratively Decoded Parallel Concatenated Codes," *IEEE Trans. Commun.*, vol. 49, no. 10, pp. 1727–1737, 2001.

[25] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. John Wiley & Sons, 2012.

[26] J. Hou, P. H. Siegel, L. B. Milstein, and H. D. Pfister, "Capacity-approaching Bandwidth-efficient Coded Modulation Schemes Based on Low-density Parity-check Codes," *IEEE Trans. Inform. Theory*, vol. 49, no. 9, pp. 2141–2155, 2003.

[27] L. Schmalen, "Degree Distribution Optimization of LDPC Codes," Institute of Telecommunications, University of Stuttgart, Germany, Oct 2019, webdemo. [Online]. Available: http://webdemo.inue.uni-stuttgart.de

[28] X.-Y. Hu, E. Eleftheriou, and D.-M. Arnold, "Regular and Irregular Progressive Edge-growth Tanner Graphs," *IEEE Trans. Inform. Theory*, vol. 51, no. 1, pp. 386–398, 2005.

[29] S. Dörner, S. Cammerer, J. Hoydis, and S. ten Brink, "Deep Learning Based Communication Over the Air," *IEEE J. S. Topics in Signal Process.*, vol. 12, no. 1, pp. 132–143, Feb 2018.

[30] A. Felix, S. Cammerer, S. Dörner, J. Hoydis, and S. ten Brink, "OFDM-Autoencoder for End-to-end Learning of Communications Systems," in *Proc. IEEE Int. Workshop on Signal Processing Advances in Wireless Commun. (SPAWC)*, 2018, pp. 1–5.

[31] S. Schibisch, S. Cammerer, S. Dörner, J. Hoydis, and S. ten Brink, "Online Label Recovery for Deep Learning-based Communication Through Error Correcting Codes," in *Proc. IEEE Int. Symp. on Wireless Commun. Systems (ISWCS)*, 2018, pp. 1–5.

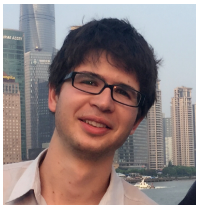
[32] E. Stauffer, A. Wang, and N. Jindal, "Deep Learning for the Degraded Broadcast Channel," *arXiv preprint arXiv:1903.08577*, 2019.

[33] S. Park, O. Simeone, and J. Kang, "Meta-Learning to Communicate: Fast End-to-End Training for Fading Channels," *arXiv preprint arXiv:1910.09945*, 2019.

0090-6778 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Exeter. Downloaded on June 18,2020 at 15:32:29 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCOMM.2020.3002915, IEEE Transactions on Communications

Sebastian Cammerer (S'16) received the B.Sc. and M.Sc. degree (with distinction) in electrical engineering and information technology from University of Stuttgart, Germany, in 2013 and 2015, respectively. During his years of study he worked as a research assistant at multiple institutes of University of Stuttgart. Since 2015 he is a member of research staff at Institute of Telecommunications, University of Stuttgart, and is pursuing his Ph.D. His main research topics are machine learning for communications and channel coding. Further research interests are in the areas of modulation, parallelized computing for signal processing and information theory. He is recipient of the IEEE SPS Young Author Best Paper Award 2019, the Best Paper Award of the University of Stuttgart 2018, the Anton- und Klara Röser Preis 2016, the Rohde&Schwarz Best Bachelor Award 2015, the VDE-Preis 2016 for his master thesis and third prize winner of the Nokia Bell Labs Prize 2019.



Fayçal Ait Aoudia received the M.Sc. degree in computer science engineering from the National Institute of Applied Sciences of Lyon (INSA Lyon) in 2014, and the Ph.D. degree in signal processing from the University of Rennes 1 in 2017. His Ph.D. work has focused on energy management and communication protocols design for autonomous wireless sensor networks. Since 2017, he has been a Researcher with Nokia Bell Labs, France, where he is involved in machine learning for wireless communications. He received the 2018 Nokia AI Innovation Award, the 2018 and 2019 Nokia France Top Inventor Awards, and the 2018 ICT Best Demonstrator Award.

Sebastian Dörner (S'18) received the B.Sc. and M.Sc. degrees in electrical engineering and information technology in 2014 and 2017, respectively, from the University of Stuttgart, Germany, where he has been a member of research staff with the Institute of Telecommunications since April 2017 and is currently working toward the Ph.D. degree. His main research topic is applying machine learning techniques to different areas of communication systems. Further research interests include modulation, parallelized GPU computing and signal processing, channel coding and SDRs. He is recipient of the IEEE SPS Young Author Best Paper Award 2019, the University of Stuttgart's Best Paper Award 2018, the Dr. Gunter Woysch Award 2017 for his master thesis and third prize winner of the Nokia Bell Labs Prize 2019.

Maximilian Stark (S'14) received Bachelor of Science and Master of Science degrees in Electrical Engineering from the Hamburg University of Technology (TUHH) in 2014 and 2017, respectively. In 2016 he studied at Chalmers University of Technology during an Erasmus semester. Since 2017 he is with the Institute of Communications at TUHH as research assistant pursing his Ph.D. degree. In 2019 he was with Nokia Bell Labs, France, as visiting researcher in the area of deep learning in communications. His research is in the area of machine learning methods for communications and signal processing, with particular interest in channel coding, the information bottleneck method and coarsely quantized signal processing. He was recipient of the Karl H. Dietze award in 2017 for his master thesis.

Jakob Hoydis (S'08-M'12-SM'19) received the diploma degree (Dipl.-Ing.) in electrical engineering and information technology from RWTH Aachen University, Germany, and the Ph.D. degree from Supélec, Gif-sur-Yvette, France, in 2008 and 2012, respectively. He is currently head of a research department at Nokia Bell Labs, France, focusing on radio systems and artificial intelligence. Prior to this, he was co-founder and CTO of the social network SPRAED and worked for Alcatel-Lucent Bell Labs in Stuttgart, Germany. His research interests are in the areas of machine learning, cloud computing, SDR, large random matrix theory, information theory, signal processing, and their applications to wireless communications. He is a co-author of the textbook "Massive MIMO Networks: Spectral, Energy, and Hardware Efficiency" (2017). He is recipient of the 2019 VTG IDE Johann-Philipp-Reis Prize, the 2019 IEEE SEE Glavieux Prize, the 2018 IEEE Marconi Prize Paper Award, the 2015 IEEE Leonard G. Abraham Prize, the IEEE WCNC 2014 best paper award, the 2013 VDE ITG Förderpreis Award, and the 2012 Publication Prize of the Supélec Foundation. He has received the 2018 Nokia AI Innovation Award, the 2018 and 2019 Nokia France Top Inventor Award, and has been nominated as an Exemplary Reviewer 2012 for the IEEE Communication Letters. He is currently chair of the IEEE COMSOC Emerging Technology Initiative on Machine Learning as well as Editor of the IEEE Transactions on Wireless Communications.

Stephan ten Brink (M'97–SM'11) has been a faculty member at the University of Stuttgart, Germany, since July 2013, where he is head of the Institute of Telecommunications. From 1995 to 1997 and 2000 to 2003, Dr. ten Brink was with Bell Laboratories in Holmdel, New Jersey, conducting research on multiple antenna systems. From July 2003 to March 2010, he was with Realtek Semiconductor Corp., Irvine, California, as Director of the wireless ASIC department, developing WLAN and UWB single chip MAC/PHY CMOS solutions. In April 2010 he returned to Bell Laboratories as Department Head of the Wireless Physical Layer Research Department in Stuttgart, Germany. Dr. ten Brink is a recipient and co-recipient of several awards, including the Vodafone Innovation Award, the IEEE Stephen O. Rice Paper Prize, and the IEEE Communications Society Leonard G. Abraham Prize for contributions to channel coding and signal detection for multiple-antenna systems. He is best known for his work on iterative decoding (EXIT charts), MIMO communications (soft sphere detection, massive MIMO), and deep learning applied to communications.

0090-6778 (c) 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: University of Exeter. Downloaded on June 18,2020 at 15:32:29 UTC from IEEE Xplore. Restrictions apply.