Learning Joint Detection, Equalization and Decoding for Short-Packet Communications

Sebastian Dörner®, Graduate Student Member, IEEE, Jannis Clausius®, Graduate Student Member, IEEE, Sebastian Cammerer®, Member, IEEE, and Stephan ten Brink®, Fellow, IEEE

Abstract—We propose and practically demonstrate a joint detection and decoding scheme for short-packet wireless communications in scenarios that require to first detect the presence of a message before actually decoding it. For this, we extend the recently proposed serial Turbo-autoencoder neural network (NN) architecture and train it to find short messages that can be, all “at once”, detected, synchronized, equalized and decoded when sent over an unsynchronized channel with memory. The conceptional advantage of the proposed system stems from a holistic message structure with superimposed pilots for joint detection and decoding without the need of relying on a dedicated preamble. This results not only in a higher spectral efficiency, but also translates into the possibility of shorter messages compared to using a dedicated preamble. We compare the detection error rate (DER), bit error rate (BER) and block error rate (BLER) performance of the proposed system with a hand-crafted state-of-the-art conventional baseline and our simulations show a significant advantage of the proposed autoencoder-based system over the conventional baseline in every scenario up to messages conveying \( k = 96 \) information bits. Finally, we practically evaluate and confirm the improved performance of the proposed system over-the-air (OTA) using a software-defined radio (SDR)-based measurement testbed.

Index Terms—Digital communication, wireless communication, modulation coding, signal detection, machine learning.

I. INTRODUCTION

In the last decades, wireless communication systems have been continuously pushed towards their fundamental physical limits and roughly 70 years after Shannon has quantified the achievable performance bounds in his seminal work [1], we virtually achieve his capacity limits over many channels [2]. However, increased connectivity and the expected omnipresence of smart devices in almost all fields of our daily life becomes a challenging – but yet exciting – requirement for future communication systems and upcoming wireless communication standards [3], [4]. From a wireless engineering perspective, these Internet of Things (IoT) and massive machine-type communications (mMTC) networks are mostly characterized by their large number of devices. It is also expected that many of those low-rate devices do not require the transmission of large amounts of data, but short messages with sporadic channel accesses [3].

Yet, when considering the actual end-to-end performance of today’s systems for short packet transmissions, we are far away from Shannon’s achievable rates as the evaluation of systems only based on their symbol-wise spectral efficiency over well-defined and synchronized channels does often not constitute their practical performance. Especially for short messages where a conventional dedicated preamble constitutes severe overhead, a carefully designed procedure is required for detection, channel estimation and to synchronize transmitter and receiver. However, from an information theoretical perspective it is known that the achievable coding rate diminishes for short block lengths [5] and the conventional separation of a transmission into a dedicated pilot preamble for detection/estimation and a payload sequence carrying the information, as depicted in Fig. 1, is provably suboptimal [6], [7]. The theoretical results in [6] for the binary-input additive white Gaussian noise (AWGN) channel further indicate, that “in the short-packet regime, joint detection and decoding
The main contributions of this paper are synchronization, equalization and decoding (see Fig. 1) by following the concept of autoencoder-based communications [8].

The rest of this paper is organized as follows. Section II provides a short overview of related literature. Section III introduces the system model, the channel model and the corresponding conventional baseline system. The proposed holistic end-to-end framework is introduced in Section IV. In Section V we present the results of our simulations and over-the-air measurements and Section VI concludes the paper.

A. Notations

Random variables are denoted by capital italic font, e.g., $X, Y$, with realizations $x, y$, respectively. Vectors are represented using a lower case bold font, e.g., $y$. $\mathcal{U}(a, b)$ denotes a uniform distribution in the range $[a, b]$ and $\mathcal{C}(\mu, \sigma^2)$ denotes a complex-valued normal distribution with mean $\mu$ and variance $\sigma^2$.

II. RELATED WORK

Autoencoder-based communication [8] has provided a new paradigm of how to design future communication systems – instead of individually optimized signal-processing blocks, we can now jointly optimize the whole transceiver w.r.t. a single end-to-end performance metric. The first synchronization for end-to-end learning has already been proposed in [10], however, the performance was limited due to the very short messages of only a few channel uses and the synchronization neural network (NN) was trained consecutively after the messages were learned, thus, no strict joint optimization. For most of the follow-up work [11], [12], [13], [14], [15], the focus so far has mostly been on the advantages of joint signal processing and the possibilities of data-driven system optimization [15], [16], [17], i.e., the system can be trained with real-world data including all actual hardware and channel impairments. Also, with the emergence of the Turbo-autoencoder architecture [12], which allows an increased amount of information bits carried within a message, the focus was mostly on learning a coded modulation scheme for the AWGN channel that can compete with state-of-the-art short length channel codes or even outperform state-of-the-art for channels with feedback [18]. This concept has been extended to serially concatenated Turbo-autoencoders in [9] for AWGN channels. Using the autoencoder scheme for static channels with memory and inter-symbol-interference was investigated in [19]. Further, a CNN-based autoencoder [20] for non-static channels with memory was proposed in [21]. However, both [19] and [21] did not include detection or synchronization. It is worth mentioning that the concept of hyper-dimensional modulation [22] describes a similar, but non-differentiable, idea of learning holistic messages, which could be potentially also extended by an outer synchronization component. Also the authors of [23] identified a similar mMTC scenario and addressed the task of joint detection, channel estimation and decoding of conventional messages with a Turbo-receiver scheme using a bilinear generalized approximate message passing (BiG-AMP) algorithm. In this work, we want to shed light on the question of how to materialize the full end-to-end learning potential of the autoencoder concept by not just focusing on joint signal-processing
capabilities and raw coding performance, but also on the advantages of learning a holistic message structure that is inherently introduced by the Turbo-autoencoder architecture as visualized in Fig. 1.

III. SYSTEM MODEL

We assume a single-input single-output (SISO) system model with single-carrier modulation as depicted in Fig. 2. The objective is to transmit \( k \) bits within a short-packet message of \( n \) complex-valued channel uses (symbols) including all required overhead for synchronization. In this scenario, the channel model adds a random offset \( \tau_{\text{off}} \in (0, n) \) to the message starting point to model initial random access without any prior signaling. In conventional systems this is commonly done using a message that consists of a dedicated preamble sequence followed by the individual payload. Typically, the payload modulation consists of symbols chosen from a set of alphabets, e.g., quadrature phase shift keying (QPSK), 16-quadrature amplitude modulation (QAM) or 256-QAM which, however, may be a user specific choice, e.g., depending on the individual SNR. The problem can be further complicated by additional carrier frequency offset (CFO) and sampling frequency offset (SFO) as transmitter and receiver oscillators do not share the exact same frequency.

Typically, a preamble sequence is used for detection and to establish timing synchronization and only if this is ensured, the processing of the remaining data is possible. One example of such preamble sequences are Zadoff-Chu sequences, which are also used in the 3GPP 5G standard [24]. The preamble sequence can then also be used for channel estimation to enable equalization and decoding of the following payload data. Additional pilots may be embedded in the message structure to further improve this process. However, in particular for systems that only aim to transmit a short-packet message (i.e., few information bits, e.g., \( k < 100 \)), the signaling overhead to establish timing synchronization between transmitter and receiver and for piloting of the channel can be large.

Given this general description, we want to quickly clarify the terminology throughout this work by defining the meaning\(^1\) of the following terms:

- **Message:** A message \( x \) consists of \( n \) complex-valued channel uses (symbols), carrying \( k \) information bits in total. This results in a message’s information rate \( R = k/n \).\(^2\)
- **Detection:** The receiver scans received sequences \( y \) of length \( n_{\text{det}} = 2 \cdot n \) and must be able to detect whether a full message lies within such a received sequence of complex-valued symbols or not.
- **Synchronization:** The receiver needs to synchronize if a message has been correctly detected. This means the receiver must estimate the exact time offset \( \tau_{\text{off}} \) of the start/end of the message within the received sequence \( y \).

\(^1\)Note that some terms are not clearly defined in literature and their exact meaning often depends on the application. However, these are the definitions we use throughout this work.

\(^2\)\( R \) should not be confused with the code rate \( R_c = k/n_{\text{coded}} \) of the low-density parity-check (LDPC) code that is only used in the conventional baseline system, where \( n_{\text{coded}} \) is the amount bits that can be used as payload.

Fig. 3. Channel model used for training and performance evaluation.

- **Decoding:** Recovery of the originally transmitted information bits. We define the reliability of the system by the bit error rate (BER) and block error rate (BLER) performance after successful detection and synchronization.

A. Channel Model

For performance simulations and training we employ a stochastic multi-path propagation channel model derived from a Proakis type C tap-delay-line model with five coefficients [25]. The \( n_{\text{taps}} = 5 \) complex-valued channel taps \( h_i \) are randomly drawn for each channel realization, following a normal-distribution with Proakis weighting coefficients according to \( h_i = w_{\text{Proakis},i} \cdot t_i \), where \( t_i \sim \mathcal{CN}(0,1) \) and \( w_{\text{Proakis}} = [0.227, 0.46, 0.688, 0.46, 0.227] \). The full channel model is depicted in Fig. 3. In a first step the random channel access of a device is modeled by adding a random delay of \( \tau_{\text{off}} \sim \mathcal{U}(0, n - n_M) \), where \( n_M \) is the memory length of the channel, to the transmitted message \( x \). The resulting output \( x^{(2n-n_M)} \) is a sequence of \( 2n - n_M \) symbols consisting of the message \( x \) with random delay according to \( \tau_{\text{off}} \) and zero padding. This sequence is then convolved with a randomly drawn channel impulse response \( h \), resulting in \( y^{(2n-1)}_h \) of length \( 2n - 1 \) due to a full convolution. To simulate a random sampling time offset (STO) at the receiver, as the result of un-synchronized oscillators between transmitter and receiver, \( y^{(2n-1)}_h \) is then convolved with a randomly phase-shifted raised-cosine-filter vector \( g_{\text{STO}}^{(n_{\text{span}})} \) with a span of \( n_{\text{span}} = 16 \) sampled from \( g(t + \tau_{\text{STO}}) \) with

\[
g(t) = \begin{cases} 
\frac{1}{\pi} \sin \left( \frac{\pi}{2\beta} \right) \cos \left( \frac{3\pi t}{T} \right) \sin \left( \frac{\pi t}{T} \right) & \text{if } |t| = \frac{T}{2\beta} \\
0 & \text{otherwise}
\end{cases}
\]

Note that we assume, that the STO adds a single significant tap to the impulse response which results in a total memory of the channel of \( n_M = n_{\text{taps}} - 1 + 1 \) and the signal \( y^{(2n)}_g \) of length \( 2n \). Finally, we add white Gaussian noise \( n \sim \mathcal{N}(0, \sigma^2) \) to \( y^{(2n)}_g \) resulting in \( y^{(2n)} \) as our channel model’s output sequence. We decided not to model CFO and SFO as the influence of SFO is negligible for ultra short sequences and the compensation of notable CFO would pose a significant challenge for the conventional baseline system since it would probably require additional pilot symbols at the end of a
message. Also, considering the “not clock-synced” SDR-based testbed, which was used to evaluate actual deployment, the influence of frequency offsets was existent but negligible throughout our measurements.

### B. Conventional State-of-the-Art Baseline System

To address the task of transmitting \( k \) bits within \( n \) complex-valued channel uses, one can come up with a variety of different conventional systems. We chose a single carrier short message system consisting of a Zadoff-Chu-based preamble sequence [24] followed by an QPSK modulated payload as a conventional baseline system for performance comparison. 

For the specific parameters of \( k = 64 \) bits within \( n = 64 \) complex channel uses we empirically ended up with a preamble of length \( n_{ZC} = 20 \) symbols and a payload of \( n_{bl,payload} = 44 \) symbols as best performing system. An exemplary baseline system message is shown in Fig. 1a. Thus, the QPSK modulated payload sequence conveys \( k_{\text{coded}} = 88 \) coded bits. To protect the \( k = 64 \) information bits that have to be transmitted, we use a 5G NR compliant LDPC code \(^3\) of rate \( R_c = 64/88 = 8/11 \). The code is based on the second base graph with a lifting factor of 11, where the first 22 bits and the last 9 bits are punctured and the bits 65 and 66 are shortened.

The receiver of the conventional baseline system is shown in Fig. 4 and consists of two separated steps, namely detection and decoding. In a first step the detector tries to find a valid message within the received symbol vector \( y_{(2n)} \) using a combination of preamble sequence correlation and energy detection. If a certain threshold\(^4\) is reached, a message has been detected and the decoder forwards the estimated channel impulse response (which is the result of the preamble sequence auto-correlation) and the snippet \( y_{(n+n_m)} \) of the detected message to the BCJR equalizer. This concludes the first step of message detection/synchronization and starts the second step of iterative equalization, demapping and decoding (IEDD) [28]. Therefore, a max-log BCJR equalizer approximates the maximum a posteriori (MAP) [29], [30], [31] sequence estimate of the \( k_{\text{coded}} = 88 \) coded bits and forwards their log-likelihood ratios (LLRs) \( L_{\hat{x}} \) to a damped min-sum belief propagation (BP) decoder with a damping factor of \( \lambda = 0.7 \). The BP decoder then decodes the codeword for \( \ell_{\text{BP}} \) iterations and feeds the gained extrinsic information \( L_{\hat{x},\text{ext}} \) back to the BCJR equalizer. While the BCJR equalizer could not make use of any \textit{a priori} information in the first iteration (as there has not been any feedback from the decoder yet, \( L_{\hat{x},\text{ext}} = 0 \)), it can now estimate the most likely bit sequence in form of \( L_{\hat{x}} \) again using the extrinsic information \( L_{\hat{x},\text{ext}} \) provided by the decoder. The decoder feedback is considered in the equalizer by a weighted addition to the branch metrics. However, the optimal weighting factor is a function of the signal-to-noise-ratio (SNR) which is assumed to be unknown. Therefore, we empirically optimize the weight and set it to 0.2. Note, we found that an SNR estimation based on these short messages does not yield a gain in performance. This process can be repeated for several \( \ell_{\text{IEDD}} \) iterations resulting in a very competitive best-in-class conventional receiver system that can be considered state-of-the-art when it comes to bit-information retrieval at receiver side. Finally, after \( \ell_{\text{IEDD}} \) iterations, the decoder outputs the decoded bit sequence \( \hat{x} \).

### IV. Serial Turbo-Autoencoder-Based System

In this section, we introduce the proposed serial Turbo-autoencoder-based holistic short message system architecture for PHY-level communication. For the sake of convenience we will refer to this system as \textit{PHY-AE} in the following. Fig. 5 shows the transmitter part of the PHY-AE which comprises the learning of a piloting scheme for detection and equalization as well as learning of an powerful forward error correction (FEC) encoding to protect information bits during transmission. Note, that no explicit outer FEC code is used within the PHY-AE system and an error protection scheme must be implicitly learned. In Fig. 6 the two staged receiver part that comprises signal detection, synchronization, channel equalization and iterative decoding is shown. Further, a detailed documentation of the employed CNN architectures and hyperparameters is given in Appendix A.
of complex-valued channel uses $x$ is a change of the final output activation to obtain a sequence.

Compared to this structure, the key adjustment to the encoder to be able to construct state-of-the-art short block codes is the architecture in [9] that showed, for the AWGN channel, serial Turbo-autoencoder’s encoder part and closely follows a related to the receiver part of the serial Turbo-autoencoder structure is necessary in terms of the dimensions of the inputs and outputs. In general, the dimensions of the exchanged information between inner and outer decoder can be chosen arbitrarily. However, as we use one-dimensional CNNs, a suitable number of dimensions is 2. We interpret the first dimension as a positional dimension and the second dimension as depth with $F$ entries per position. A practical simplification is to set the length of the positional dimension to $n$. As a result, we can choose $F$ freely and concatenate inputs to a CNN via the depth. However, the a priori information and the synchronized channel observations do no share the length, but are of length $k = n$ and $n'$ respectively. One way, that works well, is to align the lengths of the positional dimension via zero padding. Note, that we set the number of bits $k$ and the number of channel uses $n$ to the same value ($k = n$).

### C. Training

We train the PHY-AE by optimizing the binary cross-entropy (BCE) loss $L_{BCE,AE}$ between the originally transmitted input bit sequence $u$ and the estimated bit sequence $\hat{u}$ at the receiver’s output. In parallel, the detection CNN preceding the Turbo-decoder outputs a softmax activated prediction $p^{(n-n_{RM}+1)}_r$ on the starting point of a detected message within the received sequence $y^{(2n)}$, as schematically shown in Fig. 7. On the basis of this prediction $p^{(n-n_{RM}+1)}_r$ and the actual offset $\tau$ used by the channel model, we calculate a secondary categorical cross-entropy (CCE) loss $L_{CCE,det.}$ which is added to the BCE loss. The detection loss $L_{CCE,det.}$ can further be weighted by a hyperparameter $\alpha$ to balance decodability and detectability. Thus, the total loss is given by

$$L_{tot.} = L_{BCE,AE} + \alpha \cdot L_{CCE,det.}. \quad (2)$$

As a result, all components including encoder and decoder with preceding detection CNN contribute to the same loss $L_{tot}$ and are thereby trained in an end-to-end manner via stochastic gradient descent (SGD). This ensures the learning of holistic messages which are both optimized for detectability and decoding performance. The training process itself is similar to [9], [12] and the main hyperparameters are given in Tab. I.

### Table I

| Parameter        | Value        |
|------------------|--------------|
| Loss             | $10^{-4} - 10^{-6}$ |
| $T_{TX}$         | 100          |
| $T_{RX}$         | 500          |
| Batchsize        | $500 - 4000$ |
| Optimizer        | ADAM         |
| Learning rate    | $10^{-4}$    |
| Encoder SNR      | 15.0dB       |
| Decoder SNR      | 10.0 – 15.0dB|
| Loss-weight $\alpha$ | 0.01        |
model with memory. First, the detector needs to be included in the alternating training process of updating the encoder without the decoder and vice versa. To insure that the goals of the detector and the decoder are non-contradictory, the encoder weights are set to non-trainable during detector and decoder updates. Yet, while keeping the alternating training schedule w.r.t. to the encoder/decoder training, the encoder (and thereby the learned messages) is jointly optimized for detection and decoding as the loss is calculated depending on both detector and decoder CNNs. Note, that the trade-off parameter $\alpha$ also balances the contradicting goals of decodability and detectability in the encoder. Another notable difference to the AWGN case is, that the autoencoder over-fits to the interleaver that is used during training. This means, the interleaver cannot be exchanged after training (which is possible in the AWGN scenario where a TurboAE generalizes to any interleaver [12]) and must be chosen accordingly beforehand. This behavior might be due to the memory induced disturbances in the channel and the resulting position-aware sequence estimation and tracking of the decoder.

Further, we want to emphasize that the BCE loss optimizes the BER and not, as often desired, the BLER. As a result, we observed a significant number of block errors with only a single erroneous bit. Thus, we propose a to add a single parity bit and flip the least reliable estimate in case the parity bit is not matching. To account for the parity bit we chose to transmit one more bit, i.e., the autoencoder transmits $k_{\text{PHY AE, actual}} = k + 1$ bits over $n = k$ channel uses. Thus, this does not affect the final information rate $R$. Further implementational details and the training procedure of a single epoch Alg. 1 are given in Appendix A.

D. Discussion of Learned Messages and Power Limitation

Before we take a look at the BER and BLER performance results, we first analyze the learned messages and discuss the impact of symbol power limitation with respect to the peak-to-average power ratio (PAPR). Note, that it is not informative or even possible to inspect each learned message individually due to the sheer amount of $2^k$ differently modulated messages. Further, it is instructive to realize that the learned transmitter performs a non-linear encoding. Thus, to get a better insight into the learned message sequences, Fig. 8 provides a scatter plot (heatmap histogram) of several thousand randomly generated messages, exemplary with parameters $k = 64$ and $n = 64$. As can be seen, the transmitter learns a “croissant”-shaped Gaussian distribution with several distinct pilot positions and one outside of the center at around $2.3 - 2.3j$. The fact that the learned distribution is not perfectly Gaussian or perfectly symmetrical indicates that a trade-off between detection/synchronization, channel estimation/equalization and information transfer was learned. This trade-off is further analyzed in [34]. Within the learned distribution, we observe four more frequent positions (highlighted with green circles) which form an almost straight line together with the piloting position at $2.3 - 2.3j$. We assume that symbols at these positions mainly serve the purpose of detection, synchronization and channel estimation and can thereby be seen as superimposed pilot positions, while the remaining positions within the “croissant”-shape are most likely information carrying. Interestingly, the observed characteristics like a rotational asymmetric symbol distribution, a few obvious pilot positions in a straight line and a tendency to U-formed or “croissant”-formed distribution shapes was reproducible in our experiments. Starting with a random initialization, we trained the system multiple times from scratch and each time the learned messages exhibited those similar characteristics.

Also if we take a look at the average power $\mathbb{E} \left[ |x_i|^2 \right]$ and variance $\text{Var} \left( x_i \right)$ allocated to each symbol position $i$, as shown in Fig. 9, we can clearly identify 5 distinct pilot
symbols at the end of the learned messages. These final 5 symbols do not carry significant information as their average variance is close to zero, meaning they consist of constant values just like traditional explicit pilot symbols. Their average power also clearly differs from information carrying symbols. Within the subset of investigated messages, the last symbol $x_{n-1}$ always corresponded to position $2.3 - 2.3j$, which is visible in Fig. 8, and also the remaining 4 designated pilot symbols correspond to the more frequently used positions identified in Fig. 8 (green circles). This result of the PHY-AE apparently learning to use a dedicated pilot sequence is of particular interest as it contradicts the information theoretical argument, that a dedicated preamble and payload are sub-optimal for detection and decoding [6], [7]. However, the task of the chosen scenario goes beyond slotted detection and decoding and also includes synchronization and channel estimation, which likely benefits from dedicated piloting symbols. It is also not possible to determine if the PHY-AE solely relies on these dedicated pilots or if a superimposed piloting scheme is also present in the remaining symbols of a message. Which can be assumed to some extent, due to the unsymmetrical non-Gaussian distribution of the remaining symbols.

Furthermore, we can see in Fig. 9, that the transmitter learns to allocate a significant part of the available power per message to the last symbol $x_{n-1}$, presumably to easily measure the channel’s impulse response. Unfortunately, such high symbol power peaks lead to a large PAPR, which is disadvantageous or even prohibitive for some hardware components, e.g., more expensive power amplifiers are required. Especially with regard to cost- and power-efficient user equipments (UEs) either a limited PAPR or even a constant envelope modulation can be required [35]. For example the learned messages shown in Fig. 8, which have only been constrained by an average power normalization (per message), exhibit a higher PAPR than the conventional baseline system, as can be seen by the complementary cumulative distribution function (CCDF) of the PAPR depicted in Fig. 11.5 However, the PAPR can easily be reduced by limiting the individual symbol power $|x_i|^2$ to a desired range during training. In the most extreme case the power of each symbol can thereby be limited to the unit circle, resulting in the symbol distribution depicted by Fig. 10. Consequently, the learned symbol power constrained PHY-AE messages shown in Fig. 10 entail an PAPR reduction of more than 5 dB, as shown in Fig. 11, resulting in a PAPR performance comparable to that of the QPSK-based conventional baseline. As can be seen in the heatmap of Fig. 10, the symbols of these messages are also not uniformly or symmetrically distributed over the unit circle, however, the assumed pilot positions (green circles), which we determined by their close to zero variance $\text{Var}(x_i)$, are not the most frequently used positions anymore. Yet, due to the reduction of the available modulation space to the unit circle, the density increases and positions are therefore used more often. The consequences of symbol power limitation are, however, a more restricted modulation space that leads to less shaping gain, hindered channel estimation/equalization, and hindered detection, as will be shown in the next section.

V. RESULTS

We first evaluate the detection and decoding performance of the PHY-AE over the simulated channel model and, then, over-the-air within a real world SDR-based deployment. We compare the performance with the hand-crafted conventional baseline system introduced in Sec. III-B.

A. Detection Performance

We define a detection error as the event where the detector is not able to detect and localize a message within a
window of $n_{\text{det}}$ symbols to an accuracy of $\pm n_M$ symbols (mis detections + synchronization errors). A detection error also occurs in the event of wrongly detecting a message where a “none-message” $y_{\text{none}}$, a recording of $n_{\text{det}}$ symbols without any transmitted message, was simulated (false alarm). As a decision for a threshold cannot optimize both types of errors jointly, we set the threshold such that the false alarm rate is lower than $0.1\%$ according to the 5G NR specifications for the physical random access channel (PRACH) [27]. This has been ensured empirically by simulations to find the optimal threshold for the conventional baseline system and by adjusting $\alpha$ and the amount of none-messages during training of the PHY-AE.

As a result, Fig. 12 shows the simulated DER for the conventional baseline system and the PHY-AE systems. As can be seen, the unconstrained PHY-AE shows the best DER performance. We assume that the possibility to allocate more energy into single symbols enables better detection. Yet, also the maximal symbol power-constrained PHY-AE constr. with $|x_i|^2 = 1$ significantly outperforms the correlation and energy detection-based conventional baseline. The overall better detection performance of the PHY-AE’s CNN-based detector shows the advantage of a system that is optimized for both data transmission and detection from end-to-end.

B. Decoding Performance

After successful detection of a received message, the receivers needs to equalize $y'$ and estimate the originally transmitted bit sequence $u$. Fig. 13 shows the achieved BLER and BER performance of

- The conventional Baseline system
- An unconstrained Deep CNN-based autoencoder architecture
- The unconstrained PHY-AE
- The maximal symbol power constrained ($|x_i|^2 = 1$) PHY-AE constr.

for parameters $k = 64$ and $n = 64$. As a reference, we also show the maximal achievable performance Baseline (full CSI) of the baseline system where the CSI, i.e., $h$ and $\tau_{\text{STO}}$, is perfectly known at the receiver side. One can observe that the previously discussed DER of all systems (Fig. 12) is $\approx 6$ dB better than their respective BLER. This means that detection errors have almost no impact on the overall performance for the simulated channel. We can also see that the PHY-AE performs up to 1 dB better in terms of BLER and more than 2 dB better in terms of BER compared to the conventional Baseline. One particularly interesting observation is that the symbol power constrained PHY-AE constr. is almost on par with the unconstrained PHY-AE, which shows that symbol power limitation only results in slightly degraded end-to-end performance. Another interesting result is that the PHY-AE even outperforms the Baseline (full CSI) with perfect CSI in terms of BER in the low SNR regime. These large BER gains indicate that the PHY-AE must have learned a holistic coding scheme with notable coding gain considering the short block length of $k = 64$.

C. The Advantage of the Interleaver

To better illustrate the advantage of employing a serial Turbo autoencoder architecture, Fig. 13 also shows the performance of a Deep CNN-based autoencoder system that can be compared to the systems described in [20] and [21]. The Deep CNN system shares exactly the same structure and hyperparameters as the PHY-AE with the exception of not having an interleaver between the inner and outer encoder and the inner and outer decoder parts, which basically results in an autoencoder consisting of a Deep CNN on the encoder and decoder side. It has been shown in [12] for the parallel TurboAE that the introduction of an interleaver can cause roughly 1 dB of coding gain in the AWGN scenario. Accordingly, in our scenario we observe in Fig. 13 that the Turbo architecture with an interleaver (PHY-AE) also improves the BLER performance.
significantly by about 1.3dB compared to the Deep CNN structure. Although the performance of the Deep CNN system is already quite good, as it can compete with the state-of-the-art conventional system, the Turbo architecture with interleave allows the PHY-AE to significantly outperform the conventional system.

D. Universality and Scope of Application

A common misconception in terms of model-based training is that NN-based PHY-layer components optimized over a channel model will only work for the exact same channel parameters. However, if the channel model relies on stochastically drawn parameters, it covers an ensemble of many different scenarios. In the case of the chosen channel model described in Sec. III-A, where the channel taps \( h \) are randomly drawn, the PHY-AE also encounters simpler channel instantiations with effectively \( n_{\text{taps}} \leq 5 \) channel taps during training (as some instantiations of \( h \) can be close to zero). From this perspective, the employed channel model acts more like a hyperparameter of the PHY-AE itself, as it defines an upper design limit for the PHY-AE, which learns a robust signaling for up to \( n_{\text{taps}} = 5 \) channel taps, but can also operate under simpler conditions. This is similar to the process of designing the length of a dedicated preamble for a conventional system with respect to the desired scope of application. Fig. 14 therefore depicts the averaged BLER performance of the PHY-AE and the baseline system over the channel model with explicitly reduced numbers of simulated channel taps from \( n_{\text{taps}} = 1, \ldots, 5 \). As can be seen, the PHY-AE also outperforms the conventional baseline over simulated channels with \( n_{\text{taps}} \leq 5 \) channel taps.  

6The differing slope and shifted BLER performance visible in Fig. 14 compared to the results for \( n_{\text{taps}} = 5 \) shown in Fig. 13 are due to a higher SNR variance for simulated channel instantiations with \( n_{\text{taps}} < 5 \).

Fig. 14. Simulated BLER vs. SNR over the channel model with a varying number of \( n_{\text{taps}} = 1,5 \) (solid) and \( n_{\text{taps}} = 7 \) (dashed) random channel taps \((k = 64, n = 64)\), while both systems are optimized for \( n_{\text{taps}} = 5 \).

Fig. 15. Simulated BLER vs. block length \( n \) evaluated at an SNR of 18dB and fixed information rate of \( R = b/n = 1 \) for each block length \( n \).

taps (dashed lines), which is out of the scope of application for both systems. As expected, both systems show a degraded BLER performance for \( n_{\text{taps}} = 7 \) while the PHY-AE still outperforms the baseline system, indicating a generalization to the task. To be able to operate over channels with up to \( n_{\text{taps}} = 7 \) channel taps, the PHY-AE would require re-training and the baseline system would have to be redesigned w.r.t. preamble length and payload encoding.

To provide a better understanding of the general superiority of learned holistic messages and joint CNN-based detection, synchronization, equalization and decoding in the field of short-packet communications, Fig. 15 shows the BLER evaluated at a fixed SNR of 18dB for various block lengths \( n \). For this evaluation the PHY-AE has been retrained and in the handcrafted baseline the number of pilots \( n_{\text{bl, ZF}} \) has been empirically optimized for each \( n \) to transmit \( k = n \) information bits. Further implementational details are given in Appendix A. As can be seen, the PHY-AE outperforms the conventional baseline for all simulated block lengths up to \( n = 96 \). Especially for shorter block lengths the gain of the PHY-AE compared to the baseline further increases. On the one hand, this is linked to the practical problem of how to reasonably design a conventional system with explicit preamble and payload allocation for ultra short block lengths, as short as \( n = 32 \). On the other hand, this result can be easily explained by the information theoretical advantages of holistic messages and joint processing over provably sub-optimal dedicated preambles [6] for short block lengths.

Another observation is that the performance of both systems increases with the block length \( n \) and the number of transmitted information bits \( k \), which is also expected as it can be explained by the increasing achievable coding rate for increasing \( k \) [5]. Yet, we know from [9], that the coding gain of the Turbo-autoencoder architecture does not scale as well as for conventional coding schemes with increasing \( k \) for \( k \gg 100 \). Therefore, we expect for increasing \( n \), as soon as detection and estimation can be handled by a sufficiently

long preamble sequence and coding gain predominates, that a conventional system will, at some point, surpass the PHY-AE. However, this is not the main focus of this work as for long block lengths of $n \gg 100$ the disadvantages of a dedicated preamble vanish and conventional systems with state-of-the-art coding schemes are already operating close to capacity.

E. Actual Deployment and Measurement Testbed

To demonstrate the applicability of the proposed system, we deploy the PHY-AE on an SDR-based testbed and evaluate the over-the-air (OTA) performance on a real world channel. Therefore, we fix the weights of the transmitter part, which had been optimized via end-to-end SGD updates through the stochastic channel model. Although it was shown in [16] and [17], that it is also possible to further optimize the transmitter through the actual channel using reinforcement learning (RL) methods or, respectively, a generative adversarial network (GAN)-based channel model, we choose to not further adapt the transmitter signaling. We see multiple motivations behind this decision. First, re-optimizing the transmitter means a potential loss of generalization as it could lead to overfitting to a certain dataset or channel condition. To counter (or leverage) overfitting, periodical RL-based retraining steps would be required to follow changing channel conditions throughout the whole topological area of operation and different environments. A similar problem holds for the GAN-based approach, where a dataset of channel conditions for the entire scope of application is required to be able to train a generative model without overfitting effects. Secondly, training the transmitter weights in-the-field would require significant computational power at the transmitter devices, which contradicts a scenario of low-power machine-type communications (MTC) devices. And thirdly, both transmitter training approaches would require some kind of feedback channel to the transmitter devices to update their weights, which requires additional bandwidth and also contradicts an application scenario of massive amounts of ultra low-budget transmit-only devices. This is why we deploy the transmitter weights learned with the stochastic channel model described in Sec. III-A, which was deliberately designed to generate harsh channel conditions for the targeted scope of application.

At the receiver side, on the other hand, we also show the performance of a finetuned receiver according to [10]. As has been shown in [10] and [15], the performance loss due to unavoidable mismatch between the synthetic channel model used for training and the actual channel can be significantly mitigated by finetuning the receiver. We used $12.5 \cdot 10^6$ recorded messages and their corresponding genie labels to finetune the receiver via re-training. However, in practice the required labels for the receiver re-training process can also be easily obtained by the periodical transmission of pilot messages or by leveraging the idea of error-correcting code (ECC)-based label recovery formulated in [36].

Our measurement testbed consists of two universal software radio peripherals (USRPs) B210 from Ettus Research, transmitting at a carrier-frequency of 2.35GHz and a bandwidth of 40.0MHz.\(^7\) The USRPs are placed in a static indoor office environment without line of sight and the measured channel impulse response is depicted in Fig. 16. Transmitter and receiver USRP are not clock-synced and during measurements we rely on each USRP’s internal oscillator clock with a declared frequency accuracy of ±2.0 ppm. Except the USRP’s internal hardware digital to analog converters we do not use over-sampling at the transmitter as well as at the receiver side, i.e., $x$ is directly fed to the transmitter USRP and $y$ is recorded at the same sampling frequency at the receiver USRP.

F. Over-the-Air Results

As with the simulated results before, we first have a look at the detection error rate for over-the-air measurements with the SDR testbed. Fig. 17 shows the OTA DER for the conventional Baseline system, the $|x_i|^2 = 1$ symbol power constrained PHY-AE constr. and the unconstrained PHY-AE. It also shows the performance of the PHY-AE with a finetuned receiver. As a first important observation we can see that both variants PHY-AE and PHY-AE constr., which have only been trained and optimized for the synthetic channel

\(^7\)As our testbed was optimized for carrier frequencies in the WiFi spectrum, we had to use a high bandwidth to experience notable delay spread. The goal was to mimic similar conditions that IoT devices find at typically used unlicensed bands $\sim 900$MHz for bandwidths of $\sim 2000$kHz.
model, perform – out-of-the-box – better than the correlation and energy detection-based conventional Baseline. Similar to the simulated results, we can also see that the unconstrained PHY-AE achieves an even better performance than the symbol power constrained PHY-AE constr. variant. If we then finetune the receiver of the PHY-AE via re-training to further fit the NN to all measurement and hardware impairments of the SDR testbed, we can see a significant performance gain of about 0.5dB for the PHY-AE w finetuned RX and about 1dB for the constrained PHY-AE constr. w finetuned RX. Based on these results we can state that the PHY-AE is able to confirm its superior detection performance over-the-air (despite the channel mismatch) and the PHY-AE with a retrained receiver can outperform the conventional system even more due to adaptation to hardware insufficiencies.

When looking at the OTA BLER performance shown in Fig. 18, we can see similar results. Both the unconstrained PHY-AE and the $|x_i|^2 = 1$ constrained PHY-AE constr. show a better BLER performance compared to the Baseline system out-of-the-box. This shows, again, that the chosen synthetic channel model is accurate enough to enable the PHY-AE to generalize to real world channels. If we further finetune the PHY-AE’s receiver part, we can again observe a significant performance gain of about 2-2.5dB for both variants, while the constrained PHY-AE seems to profit slightly more from finetuning. Note, that finetuning in this case also means deliberate overfitting to our static measurement environment, and for real world application periodic finetuning steps would be necessary to follow a changing environment [37].

In conclusion to these measurement results we can state that the PHY-AE is as reliable as the handcrafted state-of-the art conventional system for $n = k = 64$, but already at up to 3.6dB lower SNR.

VI. OUTLOOK AND CONCLUSION

We have proposed an end-to-end optimized autoencoder-NN-based solution for joint detection, synchronization, equalization and decoding in short-packet communications. Our proposed system shows better DER, BER and BLER performance than a handcrafted state-of-the-art conventional baseline system over a simulated multipath channel model for block lengths of up to $k \sim 100$ information bits. We further demonstrated the applicability of the PHY-AE in actual over-the-air measurements within an SDR-based testbed, where the superior performance was confirmed and gains were further increased by additional receiver finetuning. Due to improved performance in the domain of short-packet communications and the conceptual simplicity of learned holistic messages, we think this approach renders a viable option for future mMTC systems and, potentially, also for initial channel access schemes.

Several open questions remain, e.g.: a) It would be advantageous to extend the end-to-end loss function to explicitly optimize the (often) desired BLER metric instead of the BER via BCE. b) Further analysis within the framework of [34] on the trade-off between information transfer and detection/estimation could allow a better explanation of the learned symbol distributions. c) Adaptive finetuning of transmitter and receiver for changing channel conditions could be an interesting investigation. And d) also an extension of the PHY-AE to waveform-level signal modulation appears to be straightforward as it would allow for direct PAPR and adjacent channel leakage ratio (ACLR) targets during training [38].

APPENDIX

In the appendix we want to provide further implementation details of the autoencoder. All following parts of the system are based on a basic CNN-structure as shown in Fig. 19. This structure consists of 5 one-dimensional convolution-layers with same-padding that are parameterized by the triple $[N_{\text{filters}}, L_{\text{kernel}}, f_{\text{activation}}]$ which is the number of filters, the length of the kernel and the used activation function respectively. The output layer is chosen according to the use case within the PHY-AE system. Further, Alg. 1 shows a more detailed (but probably still incomplete) summary of the training process.

A. PHY-AE Encoder

The output layer of the outer encoder is a one-dimensional convolutional-layer with the hyperparameters

![Fig. 19. Flowchart of the basic CNN block.](image-url)
The task of the layer is to combine the channels per position to the desired output of $F = 10$ coded features per bit-position. Similarly, the output layer of the inner encoder is a one-dimensional convolutional-layer with the hyperparameters $[N_{\text{filters}}, L_{\text{kernel}}, f_{\text{activation}}] = [2, 1, \text{None}]$. In this case the channels are combined to resemble the real part and imaginary part of a complex symbol. The final processing step in the transmitter concerns the normalization. The basic normalization step ensure that each symbol sequence $x$ has a zero mean with unit energy via

$$x = \frac{x' - \mu_{x'}}{\sigma_{x'}}$$  \hspace{1cm} (3)

where $\mu_{x'}$ is the mean and $\sigma_{x'}$ the standard deviation of $x'$. Furthermore, during inference of the symbol power constrained PHY-AE, we only consider the phase of the symbol and set the magnitude to $|x| = 1$. While during training, we only clip $x_i$ according to a power-threshold per symbol to ensure a stable training process, instead of considering the phase. To achieve convergence we successively decrease this power-threshold to $|x_i|^2 = 2$, $|x_i|^2 = 1.1$ and $|x_i|^2 = 1.01$. For each threshold we train until the test-loss saturates.

**B. Detector**

Before we describe the output layer of detector-CNN in detail, we want to focus on the input to the network. A necessary pre-processing step is some form of normalization to account for different received power levels. For this, we chose to normalize the channel observations $y$ to unit power. Otherwise a varying transmit power would be required during simulation.

The output layer of the detector actually contains three layers. The first layer is a one-dimensional convolutional layer with hyperparameters $[N_{\text{filters}}, L_{\text{kernel}}, f_{\text{activation}}] = [1, 1, \text{None}]$ to combine the channels. The second layer is a flatten layer and the third layer is a dense layer with $n - n_M + 1$ neurons followed by a softmax activation function. Due to the softmax activation, each entry of $p_i^{(n-n_M+1)}$ can be interpreted as a probability of a specific integer offset $\tau_{\text{off}} = [0, 1, \ldots, n-n_M]$ and the final entry is the estimate whether a message is present in the observed sequence or not.

For the training of the detector we use a cross-entropy (CE)-loss. As the detector-CNN needs to find a trade-off between missed detections (MDs) and false alarms (FAs) we can split the loss function in two terms. Note, that errors due to wrong synchronization are interpreted as a MD. Thus, the loss can be rewritten as

$$L_{\text{CE}} = L_{\text{CE,MD}} + \gamma \cdot L_{\text{CE,FA}}$$  \hspace{1cm} (4)

where $L_{\text{CE,MD}}$ is the loss due to a MD, $\gamma$ is the ratio none-messages in the dataset and $L_{\text{CE,FA}}$ is the loss due to a FA. Therefore, we can target a specific false alarm rate (FAR) by choosing $\gamma$ accordingly. For this, we propose to adapt $\gamma$ during training based on the FAR. At the end of each epoch we check whether the FAR is smaller or larger than the target FAR and increase or decrease $\gamma$, respectively.

Further, we want to emphasize that we use different SNRs for detection and decoding during training, since the interesting SNR region for detection is roughly $6\text{dB}$ lower than for decoding.

**C. Decoder**

While Fig. 6 already shows the iterative structure of the decoder, two implementation details are left out. Both are shown in Fig. 20. The first one regards the padding of the a priori information. In order to concatenate the a priori information with the channel observations we pad it from the left and the right with $2n_M$ zeros in total. The second detail concerns the calculation of the extrinsic information. It is calculated by subtracting the a priori information from the output of a decoder-CNN. Conveniently, this acts as a residual connection in the network that might facilitate the training process.

**D. Achieving Variable Message Lengths**

In order to achieve the BLER results for various message lengths $n$, as shown in Fig. 15, we had to retrain the PHY-AE and slightly re-design the conventional baseline system. The PHY-AE was retrained from scratch for all lengths $n$, except for $n = 32$ as we examined that a retrained $n = 64$-variant performed better at $n = 32$ than the $n = 32$-variant learned.

![Diagram](image-url)
from scratch. For the conventional baseline, we empirically optimized the trade-off between pilots and payload w.r.t. each message length $n$. Thus, we still use QPSK modulation and a 5G LDPC code with the respective length and rate. As a result, we chose $n_{bl,ZF} = 16$ pilot symbols for $n = 40$ and $n = 48$, $n_{bl,ZF} = 20$ for $n = 56$ and $n = 64$, and finally $n_{bl,ZF} = 24$ for $n = 96$. Especially for short lengths $n < 56$, reducing the number of pilot symbols led to further performance degradation of the baseline system as can be seen in Fig. 15.

REFERENCES

[1] C. E. Shannon, “A mathematical theory of communication,” Bell Syst. Tech. J., vol. 27, no. 3, pp. 379–423, Jul./Oct. 1948.
[2] S.-Y. Chang, G. D. Forney, J. T. Richardson, and R. Urbanke, “On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit,” IEEE Commun. Lett., vol. 5, no. 2, pp. 58–60, Feb. 2001.
[3] F. Boccardi, R. W. Heath, A. Lozano, T. J. Marzetta, and P. Popovski, “Five disruptive technology directions for 5G,” IEEE Commun. Mag., vol. 52, no. 2, pp. 74–80, Feb. 2014.
[4] X. You et al., “Towards 6G wireless communication networks: Vision, enabling technologies, and new paradigm shifts,” Sci. China Inf. Sci., vol. 64, no. 1, pp. 1–74, Nov. 2020.
[5] Y. Polyanskiy, H. V. Poor, and S. Verdú, “Channel coding rate in the finite blocklength regime,” IEEE Trans. Inf. Theory, vol. 56, no. 5, pp. 2307–2359, Apr. 2010.
[6] A. Lachno, J. Östman, and G. Durisi, “On joint detection and decoding in short-packet communications,” in Proc. IEEE Global Commun. Conf. (GLOBECOM), Dec. 2021, pp. 1–6.
[7] A.-S. Bana, K. F. Trillingsgaard, P. Popovski, and E. De Carvalho, “Short packet structure for ultra-reliable machine-type communication: Tradeoff between detection and decoding,” in Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP), Apr. 2018, pp. 6608–6612.
[8] 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.
[9] J. Clausius, S. Dörner, S. Cammerer, and S. ten Brink, “Serial vs. parallel turbo-autoencoders and accelerated training for learned channel codes,” in Proc. 11th Int. Symp. Topics Coding (ISTC), Aug. 2021, pp. 1–5.
[10] S. Dörner, S. Cammerer, J. Hoydis, and S. ten Brink, “Deep learning based communication over the air,” IEEE J. Sel. Topics Signal Process., vol. 12, no. 1, pp. 132–143, Feb. 2018.
[11] F. A. Aoudia and J. Hoydis, “Trimming the fat from OFDM: Pilot- and CP-less communication with end-to-end learning,” in Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP), Jun. 2021, pp. 1–6.
[12] Y. Jiang, H. Kim, H. Asnani, S. Kannan, S. Oh, and P. Viswanath, “Turbo autoencoder: Deep learning based channel codes for point-to-point communication channels,” in Proc. Adv. Neural Inf. Process. Syst., vol. 32. Red Hook, NY, USA: Curran Associates, 2019, pp. 1–11.
[13] B. Karanov et al., “End-to-end deep learning of optical fiber communications,” J. Lightw. Technol., vol. 36, no. 20, pp. 4843–4855, Oct. 15, 2018.
[14] E. Bourtsoulatze, D. B. Kurka, and D. Gündüz, “Deep joint source-channel coding for wireless image transmission,” IEEE Trans. Cogn. Commun. Netw., vol. 5, no. 3, pp. 567–579, Sep. 2019.
[15] S. Cammerer, F. A. Aoudia, S. Dörner, M. Stark, J. Hoydis, and S. ten Brink, “Trainable communication systems: Concepts and prototype,” IEEE Trans. Commun., vol. 68, no. 9, pp. 5489–5503, Sep. 2020.
[16] F. A. 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.
[17] S. Dörner, M. Henninger, S. Cammerer, and S. ten Brink, “WGAN-based autoencoder training over-the-air,” in Proc. IEEE 21st Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC), May 2020, pp. 1–5.
[18] Y. Jiang, H. Kim, H. Asnani, S. Oh, S. Kannan, and P. Viswanath, “Feedback turbo autoencoder,” in Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP), May 2020, pp. 8559–8563.
[19] Y. Zhang, H. Wu, and M. Coates, “On the design of channel coding autoencoders with arbitrary rates for ISI channels,” IEEE Wireless Commun. Lett., vol. 11, no. 2, pp. 426–430, Feb. 2022.
[20] H. Ye, L. Liang, and G. Y Li, “Circular convolutional auto-encoder for channel coding,” in Proc. IEEE 20th Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC), Jul. 2019.
[21] H. Ye, G. Ye Li, and B.-H.-F. Jiang, “Bilinear convolutional autoencoder based pilot-free end-to-end communication systems,” in Proc. IEEE Int. Conf. Commun. (ICC), Jun. 2020.
[22] H.-S. Kim, “HDM: Hyper-dimensional modulation for robust low-power communications,” in Proc. IEEE Int. Conf. Commun. (ICC), May 2018, pp. 1–6.
[23] X. Bian, Y. Mao, and J. Zhang, “Joint activity detection and data decoding in massive random access via a turbo receiver,” in Proc. IEEE 22nd Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC), Sep. 2021, pp. 361–365.
[24] NR: Physical Channels and Modulation, document TS 38.211, version 17.1.0, 3rd Generation Partnership Project, 2022.
Jannis Clausius (Graduate Student Member, IEEE) received the B.Sc. and M.Sc. (Hons.) degrees in electrical engineering and information technology from the University of Stuttgart, Germany, in 2018 and 2021, respectively, where he is currently pursuing the Ph.D. degree. Since March 2021, he has been a Research Staff Member at the Institute of Telecommunications, University of Stuttgart. His research focuses on the application of deep learning for modulation and channel coding, signal processing, and information theory.

Sebastian Cammerer (Member, IEEE) received the Ph.D. degree in electrical engineering and information technology from the University of Stuttgart, Germany, in 2021. He is currently a Research Scientist at NVIDIA working on the intersection of wireless communications and machine learning. He is one of the Maintainer and a Core Developer of the Sionna open-source link-level simulator. His main research topics are machine learning for wireless communications and channel coding, parallel computing for wireless signal processing, and information theory. He was a recipient of the VDE ITG Dissertationspreis in 2022, the IEEE SPS Young Author Best Paper Award in 2019, the Best Paper Award of the University of Stuttgart in 2018, the Anton- und Klara Röser Preis in 2016, the Rohde&Schwarz Best Bachelor Award in 2015, and Third Prize Winner of the Nokia Bell Laboratories Prize in 2019.

Stephan ten Brink (Fellow, IEEE) has been a Faculty Member of the University of Stuttgart, Germany, since July 2013, he has been the Head of the Institute of Telecommunications. From 1995 to 1997 and 2000 to 2003, he was with the Bell Laboratories, Holmdel, NJ, USA, conducting research on multiple antenna systems. From July 2003 to March 2010, he was with Realtek Semiconductor Corporation, Irvine, CA, USA, as the Director of the Wireless ASIC Department, developing WLAN or UWB single chip MACPHY CMOS solutions. In April 2010, he returned to the Bell Laboratories as the Department Head of the Wireless Physical Layer Research Department, Stuttgart, Germany. He was 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 and massive MIMO), and deep learning applied to communications.