Applied Soft Computing Journal 159 (2024) 111672

Contents lists available at ScienceDirect

Applied Soft Computing

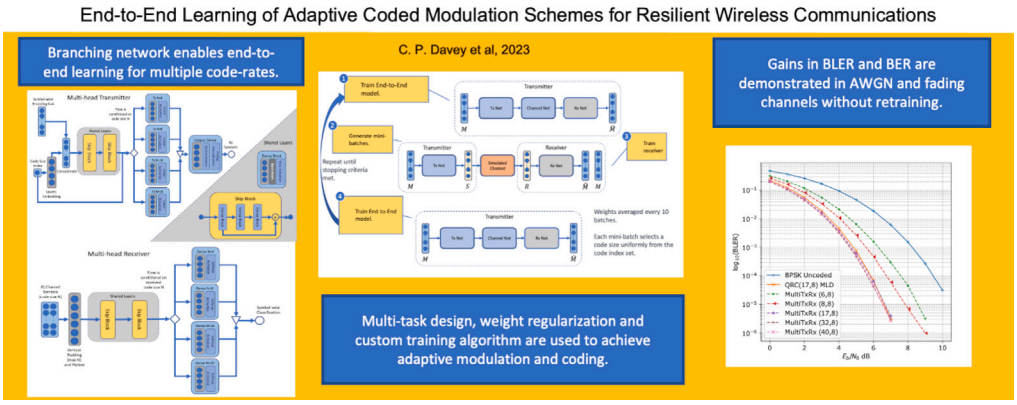
journal homepage: www.elsevier.com/locate/asoc

End-to-end learning of adaptive coded modulation schemes for resilient wireless communications

Christopher P. Davey [a,*], Ismail Shakeel [a,b], Ravinesh C. Deo [a,*], Ekta Sharma [a], Sancho Salcedo-Sanz [c], Jeffrey Soar [d]

[a] School of Mathematics, Physics and Computing, University of Southern Queensland, Springfield, 4300, Queensland, Australia
[b] Spectrum Warfare Branch, Information Sciences Division, Defence Science and Technology Group (DSTG), Edinburgh, 5111, SA, Australia
[c] Department of Signal Processing and Communications, Universidad de Alcalá, Alcalá de Henares, 28805, Spain
[d] School of Business, University of Southern Queensland, Springfield, 4300, Queensland, Australia

GRAPHICAL ABSTRACT



ARTICLE INFO

Keywords:
Wireless communications
Adaptation
Coding design
Deep learning
Multi-task learning

ABSTRACT

Adaptive modulation and coding schemes play a crucial role in ensuring robust data transfer in wireless communications, especially when faced with changes or interference in the transmission channel. These schemes involve the use of variable coding rates, which can be achieved normally through code puncturing or shortening, and have been adopted in 4G and 5G communication standards. In recent works, auto-encoders for wireless communications have demonstrated the ability to learn short code representations that achieve gains over conventional codes. Such a methodology is attractive as it can learn optimal representations under a variety of channel conditions. However, due to its structure the auto-encoder does not currently support multiple code rates with a single model. This article draws upon the discipline of multi-task learning, as it applies to deep learning and therefore devises a branching architecture for the auto-encoder and custom training algorithm in training transmitter and receiver for adaptive modulation and coding. In this article we aim to demonstrate improvements in Block Error Rate over conventional methods in the Additive White Gaussian Noise channel, and to analyse the performance of the model under Rayleigh fading channels without retraining the auto-encoder on the new channel. This article demonstrates a novel approach towards training

* Corresponding authors.
E-mail addresses: christopher.davey@usq.edu.au (C.P. Davey), ravinesh.deo@usq.edu.au (R.C. Deo).

https://doi.org/10.1016/j.asoc.2024.111672
Received 4 April 2023; Received in revised form 14 November 2023; Accepted 15 April 2024
Available online 26 April 2024

auto-encoder models to jointly learn adaptive modulation and coding schemes framed as a multi-task learning problem. The research outcomes extend end-to-end learning approaches to the design of adaptive wireless communications systems.

1. Introduction

Adaptive modulation and coding (AMC) methods can enable wireless communication systems to optimise the transmission of data over a channel with varying operating conditions, message sizes, and data transfer rates. This involves adjusting the modulation scheme and error correction coding rate in real-time based on the channel conditions. AMC algorithms are designed to monitor the channel conditions in real-time and dynamically select a suitable modulation and coding scheme that provides the best trade-off between data rate and error protection for the given channel conditions. The selection of the best modulation and coding scheme is normally made by a look-up table or an algorithm that maps the channel conditions to a particular modulation and coding scheme with respect to transmit power, expected channel use and error rate [1]. The modulator/demodulator and encoder/decoder algorithms for each modulation and code are normally designed and implemented separately on the communication platform [2]. This paper focuses on using machine learning techniques to generate multiple coded modulation schemes from a unified model architecture for channel adaptation.

In communications systems, the primary goal is to transmit a message through a communication channel to a receiver and then reconstruct the original message without error at the receiver. Distortions that are introduced by the channel are the primary obstacle to an error free recovery of the transmitted message.

Fig. 1 illustrates a simple wireless communications system, which is the focus of our article. In such a system, a message M, defined in bits, is formatted and communicated by a transmitter over the air (the channel) to be recovered and interpreted at the receiver. In order to be transmitted, messages may be coded for error correction and modulated for transmission over the channel. The modulation process converts a bit sequence into a waveform where each discrete point (symbol) in the waveform is represented as a series of complex symbols $x(t) \in \mathbb{C}$, having both in-phase and quadrature (IQ) components, where t indicates the discrete time step for the symbol. We use the term "symbol" after modulation because a single symbol can refer to more than 1 bit of the original message. The number of bits mapped to a symbol is referred to as the order of the modulation. However, a modulation is not necessarily sufficient to allow the receiver to recover the message without error.

The subject of this research is coding, which is an essential component of any communication system that enables error detection and correction at the receiver end. A variety of techniques are available to code a particular message, and these include linear block codes and convolutional codes. The resulting code words are longer than the original message block, and the ratio between the original message at length K and the resulting N bit code word is known as the code rate K/N. For smaller code rates, one requires more symbols to be transmitted across the channel. The ability to perform error correction using a coded message can achieve significant improvements over uncoded messages (a coding gain) but this comes with a trade-off in terms of the number of usages required to transmit the message, or in other words, the amount of power required to send the message. However, the ability to receive a message without error reduces the need for re-transmissions.

As illustrated in Fig. 1, the coding and modulation are an integral part of the system. Therefore in this article, we focus on the learning of the coding and modulation process that assumes a perfect synchronisation of the transmitted signal with the receiver. The coding and modulation stages in the wireless communications system are typically designed in isolation of each other, and often, they do not account for the distortions introduced by different types of channels. This method of system design is referred to as the block design, where components are individually optimised and do not consider interactions between components or the channel distortion [3].

The ability to automatically learn each of the stages within a wireless communications system presents an advantage over the block design approach, since each of the stages can be optimised jointly with respect to a given channel condition and hardware imperfection. As such, learning how to transmit and receive coded information over the wireless channel has recently attracted significant attention in the field of wireless communications. Deep learning (DL) methods, and in particular the AE architecture have been demonstrated to jointly optimise both transmitter and receiver with respect to an assumed channel model [2]. Such a joint approach, learns coding and modulation in an end-to-end manner by gradual optimisation of the model parameters to minimise the error produced in symbol-wise classification of individual messages [2].

In end-to-end learning, a transmitter acts as an encoder network to encode a message, while the receiver acts as the decoder network to retrieve the original message [2]. The channel is represented either as an instantaneous function which adds perturbations to the output of the transmitter [2], may be learnt through adversarial techniques [4] or reinforcement learning (RL) is applied without assuming a channel [5]. The design of the AE for wireless communications in [2], does not support learning more than one code rate, in that approach support for multiple code rates requires separate networks. Therefore the subject of this research article investigates how to parameterise and alter the structure of a single AE for wireless communications to enable support for multiple code rates.

Multi-task learning (MTL) in DL research is concerned with the challenge of training a single network architecture to concurrently perform different but related tasks, which may also have a dependency relation between them [6]. Approaches to MTL consider the architecture design, model regularisation and training methods [6]. There is a relationship between negative transfer in MTL and catastrophic forgetting which occurs in sequential learning [7,8]. Negative transfer occurs when certain tasks negatively impact the ability to learn other tasks when learnt concurrently [6]. Hence regularisation techniques acting to minimise negative transfer during training are a key concern of MTL. This is realised through the design of the network architecture (hard sharing) as well as through weight regularisation and loss functions (soft sharing) [6]. In this article we approach AMC from the perspective of MTL and apply both hard sharing in the AE network design as well as soft sharing in the approach to regularisation during the training procedure. We demonstrate that a multi-branching variant of the AE is better suited to learning AMC in comparison to a single path AE network. The model is trained by iterating between end-to-end and receiver only training, and we apply a weight averaging regularisation technique [9] to improve the error rates for each of the resulting code rates.

The main contributions of this paper are:

- To change the structure of the AE for wireless communications to enable learning multiple code rates with a single neural network architecture. A shared path with branching output heads are activated based on a selected code rate parameter to support end-to-end learning for AMC.
- To frame the end-to-end learning of AMC for wireless communications as a multi-task learning problem.
- Propose a training procedure which iterates between end-to-end training and receiver training, producing lower error rates than single step training.

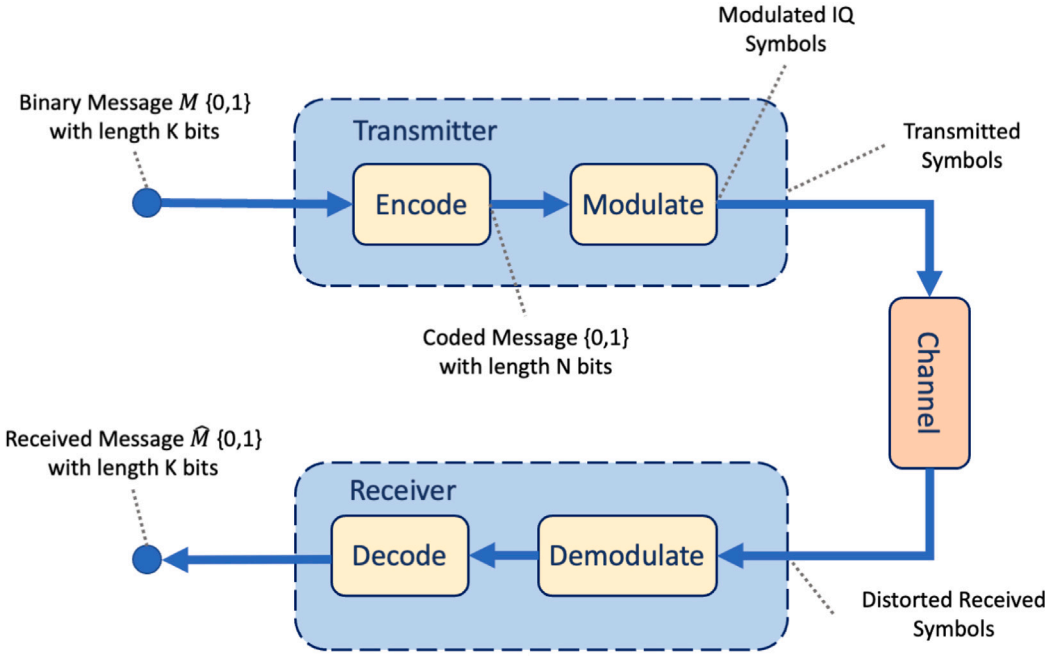


Fig. 1. A simplified schematic of a typical wireless communication system. In the transmitter, a binary message of K information bits is coded as N bits for error correction during the Encode stage. The Modulation stage converts the bits into discrete complex symbols using amplitude, phase or frequency to differentiate bits. On the receiver side, the demodulator converts the received modulation symbols to the original code word of length N and the decoding block converts from the code back to the original K information bits of the message.

- To show that the proposed method achieves results closely matching and improving upon performance of maximum likelihood decoding (MLD) with conventional codes over several code rates and channels.

The remainder of this article is structured in the following manner: Section 2 provides an overview of the research into end-to-end learning for wireless communications as well as describing multi-task learning and adaptive modulation and coding. Section 3 describes the multi-rate AE, the method of regularisation and its custom training procedure. Section 4 reports on the BER and BLER of the model under several channel environments. Section 5 describes the limitations of the model and discusses the generalisation capability. The article concludes in Section 6 where the summary of findings is given and further directions for research are proposed.

2. Background and related work

The use of AE neural networks for learning an end-to-end wireless communications system was first proposed in [2]. An AE was demonstrated to learn optimal short codes for the AWGN channel. The AE model architecture as described by the article is illustrated in Fig. 2. The model consists of symbol-wise encoding for 2^K possible messages, a transmitter containing multiple dense blocks (layers) and an energy normalisation constraint, transmitter output IQ symbols for a code size N, an assumed channel function, and a receiver (also containing multiple dense blocks) whose task is to predict the received message index [2]. In a typical AE the middle layers are applied to find a compressed set of features for the input, whereas in the wireless communications design, the middle layers typically represent the output of the transmitter, which are influenced by the distortion provided by the channel function. By applying DL to the design of wireless communications systems the article introduced the potential use of generalised hardware platforms such as graphical processing units (GPUs), and enabled the opportunity for optimisation against complex channels without requiring an analytic mathematical model for the channel [2]. The article demonstrated that an AE with a code rate of $K/N = 4/7$ could achieve equal performance to MLD decoding for the Hamming(7,4) code and that an AE trained on an interference channel could achieve lower error rate than a quadrature amplitude modulation (QAM) time sharing modulation scheme for equivalent code sizes [2]. However, the design of the classifier architecture is limited to a fixed number of message bits K and a fixed code size N, and is constrained to the domain of short length burst transmissions. Such codes are beneficial for use in energy constrained communications such as in the internet of things (IoT). The AE model as presented in [2] serves as the canonical model for DL end-to-end communications systems under simplified constraints. Related work stemming from this initial research extends the AE architecture and examines applications to end-to-end learning, over-the-air learning, and the use of custom training algorithms.

A key assumption of the AE model as described in [2] requires that a differentiable channel function for a given channel is predefined to permit back-propagation between the transmitter and receiver. An alternate approach is to train a generative adversarial network (GAN) using observed perturbations from the true channel, thereby removing the assumption of a predetermined channel. The approach described in [4], applies this technique to approximate an unknown channel function and provides a fully differentiable channel model for training of the AE. The architecture is trained and evaluated on several channels including the AWGN, Rayleigh Fading and Frequency Selective multipath channels. Under the AWGN and Rayleigh fading channels, the AE-GAN model is compared with the end-to-end AE from [2], as well as conventional coding methods, demonstrating the effectiveness of the GAN in approximating the channel during training [4]. Each component (transmitter, channel GAN and receiver) is trained in succession using an iterative algorithm, where components not participating in each training cycle had their weights frozen [4]. The architecture assumed a constant size of K message bits for the AE input and output, and while it was able to approximate bit-wise output leveraging Convolutional Neural Network (CNN) layers to support differing message lengths, it did not address the effect of altering the code size N and was not designed to produce multiple code rates without retraining.

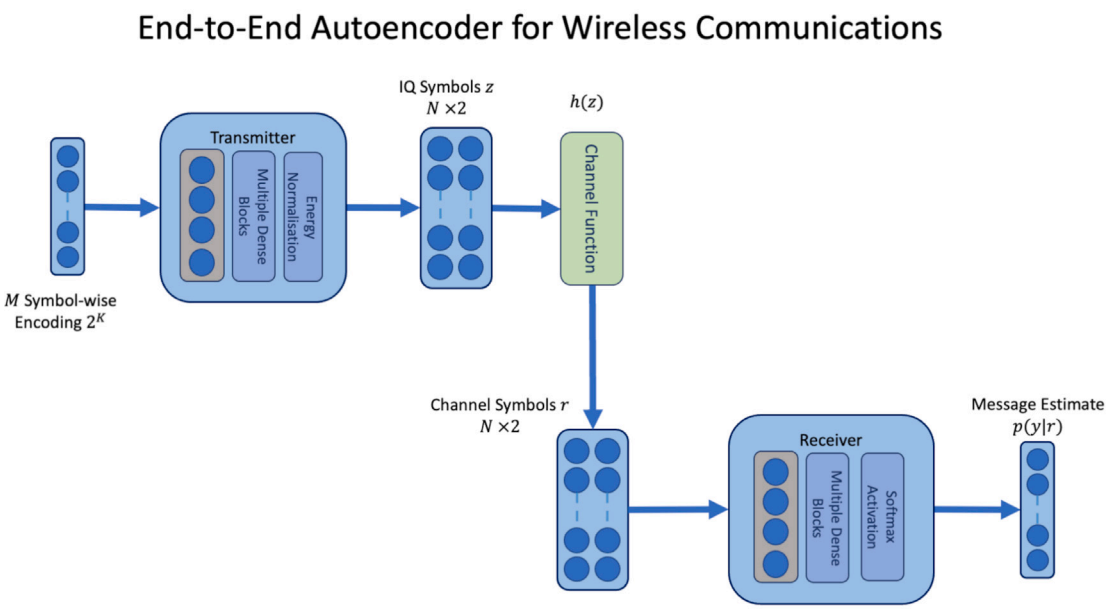


Fig. 2. The neural network architecture of the end-to-end AE for wireless communications described in [2]. The architecture of transmitter and receiver each consist of multiple dense layers or blocks, and the transmitter includes an energy normalisation layer. The model is defined for a predefined number of message bits K and a single code rate N, as well as an assumed channel function $h(z)$. The transmitter learns to encode message M from one of 2^K messages into IQ symbols which are sent over the assumed channel function and the receiver learns to estimate the probability of the message M given the received output of the channel r.

Another method for addressing the channel assumption was proposed by leveraging an iterative training algorithm based on RL in [5, 10]. The training algorithm first trains the receiver by back-propagation and in the second step applies the receiver loss to an approximation of the gradient to perform back-propagation at the transmitter [5]. The advantage of this method is that it is agnostic to the true channel, although does require reliable feedback of losses from the receiver, and applied an equalisation method from [2] in order to train against the Rayleigh Block Fading (RBF) channel [5]. This latter point indicates the dependency of DL methods on the perturbations that are applied to their training data, in wireless communications systems, these perturbations result from the channel environment. In [10] the RBF channel is evaluated with and without equalisation, indicating a slight difference in performance between the approaches. Changes made to the channel, outside of that applied during training, will have a negative impact on the performance of the model, hence further investigation of approaches to regularisation for end-to-end training are required for adaptability to changing channel conditions. While the research focus for AE in wireless communications has addressed the assumed channel function, we propose that adaptability can be achieved by also considering an AMC scheme.

One approach to address adaptation, post training, is to deploy and retrain the receiver independently of the transmitter on the true channel, also referred to as tuning or transfer learning. This technique is described in [3] where an iterative approach is applied to train the AE end-to-end and secondly to fine-tune the receiver. The method is demonstrated on a software defined radio (SDR) implementation during an over the air (OTA) training phase [3]. OTA training allows a more realistic channel environment as opposed to end-to-end training, and requires additional stages to support synchronisation such as filtering, timing, phase, and carrier frequency offset corrections [3]. Two separate sub-networks were incorporated prior to the receiver/decoder model to correct for timing and phase offsets as well as learning new features to assist in decoding [3]. While the trained system did not perform as well as the comparative communications system, the work demonstrates that receiver tuning OTA has the potential to improve the adaptation of the overall system post end-to-end training and emphasises the mismatch between the analytic and actual channel models. This is further evidence of the sensitivity of DL models to perturbations during training. While we do not investigate an OTA implementation, we do investigate the performance of an end-to-end AE on several channels without retraining or tuning, and propose that an architecture capable of generating AMC schemes is advantageous in environments for which it was not initially trained.

Extensions of the AE architecture have been made to incorporate concatenated coding techniques for reliable communications in [11] where the AE learns the inner code and the outer code is implemented with a low density parity check (LDPC) code. Such an outer code is capable of variable code rates, independent of the AE model. In this method, the AE performs bit-wise encoding and decoding as opposed to the classification of a symbol-wise output in [2]. The role of the receiver is to estimate the log-likelihood ratio for each bit and is trained in a manner equivalent to maximising the achievable rate of the communications system [11]. Both the encoder and decoder are parameterised by the signal to noise ratio (SNR) and it is shown that learnt constellations are correlated with the given SNR [11]. The association of constellation and SNR enables a form of adaptive coding which does not vary the code length, but may rearrange the constellation points instead. In our approach we instead modify the AE model architecture to allow parameterisation for a code index which permits the model to learn a mapping between the code index parameter and a variable code length, thereby achieving AMC by varying code rates.

Given an assumed channel, and a measure of the communication error rate, it is possible to iteratively search for an optimal code rate. A technique for this type of search is presented in [12]. The main contribution of the article is first to address the issue of overfitting in the end-to-end AE and propose an additional regularisation term that maximises the mutual information between the transmitter symbols and the output of an assumed channel function [12]. This regularisation term is applied to the loss term of the AE and is approximated by training a separate neural network. The search algorithm, described as capacity driven AE, iterates over multiple SNR and trains AEs at incremental code rates K/N until improvement in the mutual information over previous AE falls below a given threshold [12]. However, long training durations, and the limitations around sampling for large message bit lengths are a disadvantage for AE. An exhaustive search is feasible for short messages, but less feasible as K increases. The ability to design a single network architecture that can support multiple code

rates could reduce the overall duration of such a search algorithm. To make these changes to the AE architecture, we propose framing the task of training multiple code rates as a MTL problem.

MTL seeks to regularise a network to perform several related but distinct tasks through the network architecture (hard sharing) or through regularisation methods constraining weights in matching layers (soft sharing) [6]. The simplest hard sharing approach uses a common single path with multiple outputs to demonstrate that the relatedness between tasks benefits network regularisation through the transfer of inductive bias between those tasks [13]. This type of architecture also has the advantage of limiting the number of parameters required by multiple networks, since a common path is shared between separate branches rather than requiring an individual network for each task [13]. Training of such architectures is performed while learning multiple tasks simultaneously, whereas in our approach we train successively for single tasks (code rates) using a common architecture. However successive training on different tasks is well known to suffer from catastrophic forgetting [7] also termed negative transfer in the MTL literature [6]. The challenge of MTL for sequential learning is approached in [14] which proposes a dynamic architecture comprising of shared and task specific paths which is trained in a sequential manner. This approach is demonstrated to reduce the negative transfer between tasks on the shared path [14]. During training and inference the structure of the network is altered dynamically and enables the execution of one task at a time [14], in this manner the parameterisation of each task is implicit to the current organisation of the network. In our approach we define tasks as code rates and dynamically reconfigure the network structure during training and inference while supplying a code index parameter to indicate which code rate the transmitter should output. To regularise the shared path between tasks we make use of a simple weight averaging regularisation [9].

Adaptation for both modulation and coding has been demonstrated to achieve more reliable communications under varying levels of interference when compared to adaptation for modulation only [15]. AMC-enabled systems have also been shown to produce higher data transfer rates over various communication environments [16,17]. AMC is implemented by selecting a combination of modulation scheme and error-correcting code to achieve a target BER under a given SNR partition [16,18]. Different code sizes may be constructed from the same family of codes so that the minimum distance of the code remains constant over varying SNR and channel fading conditions [19]. Such codes can be formed by shortening, that is reducing both information and code word bits, or puncturing by removing some of the parity check bits from each code word [20]. Cyclic codes are well suited to shortening and puncturing since the original decoding procedure can be applied to the resulting code [20]. This category of codes includes the Hamming code [21], Bose–Chaudhuri–Hocquenghem (BCH) code [21,22] and the quadratic residue code (QRC) [21]. Rather than shortening a family of codes, we augment the end-to-end AE for wireless communications to jointly learn multiple code rates. The advantage of jointly learning modulation and coding would enable AMC schemes to be tuned specifically for target channel conditions.

3. Methodology

In our work we consider the AE architecture in [2] as the canonical DL architecture for jointly learning modulation and coding in a wireless communications system. However the structure of the canonical AE is limited to a single message bit size K and a single code size N. In this article we make several alterations to the original AE architecture to support end-to-end learning for AMC with multiple code sizes N. We modify the network architecture so that it is able to learn several predefined code sizes by adding branching outputs at the transmitter and receiver. We also add a parameter to select the code size index during training and inference in the transmitter. In the main path of the network we include skip connections [23] between blocks of dense units to enable a slightly larger network to aid in learning multiple code sizes.

This section includes the details for the changes to the AE architecture (Section 3.1), the approach to training for the transmitter and receiver models (Section 3.2) and the selected channel functions that are applied during training and evaluation (Section 3.3).

3.1. Model architecture

The AE, described in [2], consists of a single path through the network and a channel function implemented as an AWGN layer (Fig. 2). Estimation is performed as a classification for the corresponding one-hot encoded input message M. One-hot encoding represents each binary message M by defining an input vector of the same length as the number of unique binary messages (in 2^K messages). Each unique message is represented as a 1 at its corresponding index in the input vector (a value from 0 to 2^K-1) with all other positions of the vector set to 0. Symbol-wise classification is performed at the receiver by learning to estimate the probability of a given message at the corresponding vector index $p(M|r(t))$ given a set of channel symbols (in-phase and quadrature values) $r(t)$. The index with the highest probability is mapped from the index to the estimated binary message $\hat{M} = \arg\max p(M|r(t))$.

The branching structure of our proposed model is inspired by hard sharing in MTL. In the simplest form of hard sharing, a common network path is followed by a set of branches that correspond to each distinct task [6]. The common path learns shared features for all tasks. In our architecture a task, corresponding to a network branch, represents a different code rate K/N due to the change in the code size N. The common path contains two dense blocks composed of a feed-forward layer, batch-normalisation [24] and either rectified linear unit (ReLU) [25] or Swish [26] activation functions. The two dense blocks feature residual connections to form skip blocks which assist in preventing extremes in the gradient during back-propagation of deeper networks [23]. Although these networks are relatively shallow, we have found that the skip connections do improve the performance of the network.

In the transmitter, the common path accepts a concatenation between the one-hot encoded message and an embedding representing the code index. The code size index is provided to a discrete gate that determines which branch of the network architecture should receive features from the common path during the forward pass. Each branch contains a feed-forward layer followed by a linear activation. The output of the transmitter consists of a feed-forward layer followed by a tanh activation and an energy normalisation layer that is applied prior to the channel function. An overview of the transmitter model is shown in Fig. 3. The architecture is parameterised by a set of code rates $i \in \{1_1, \ldots, i_n\}$ that are used by the branch node to select which of the output branches are active during training and inference. The branch node is indicated by the *Discrete Gate* in the transmitter, Fig. 3 and the receiver, Fig. 4.

The architecture of the receiver is illustrated in Fig. 4 and follows a similar pattern to that of the transmitter. Instead of receiving an additional code size parameter during inference, the length of the received channel symbols is stored at the input to the network. Zero padding is applied to the received symbols up to the maximum allowed code size. The padding layer is followed by a series of skip blocks having the same structure as those in the transmitter, prior to the discrete gate. The *Discrete Gate* receives the stored length parameter and uses this to determine which output branch (*Dense Layer* i) to activate on the forward pass. Each output branch consists of a feed-forward layer and a soft-max activation layer.

The number of units in the input layers of the transmitter relate to the 2^K possible messages while for the receiver the input units depend on the code size that is selected in the transmitter. To determine the number of units for the shared path of each network, a stepwise approach was applied. Starting from the value of K the number of units

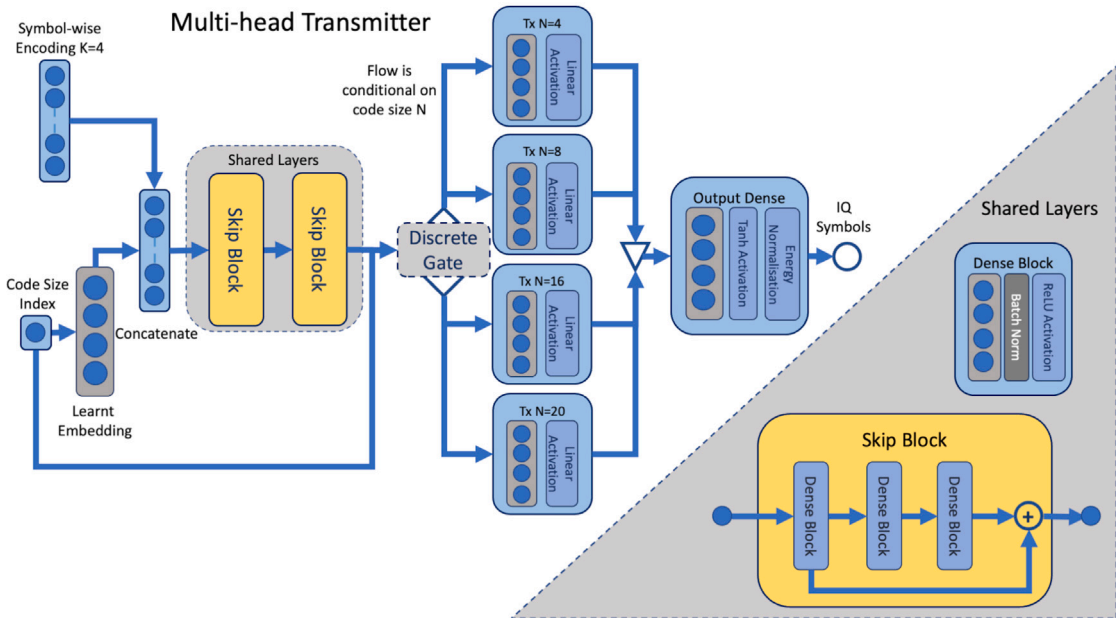


Fig. 3. The transmitter contains a common path followed by a discrete gate which switches between the set of selected code sizes prior to merging and energy normalisation. The transmitter sizes for N provide an example of how each branch represents a different code rate, and are an example of 4 bit model configuration.

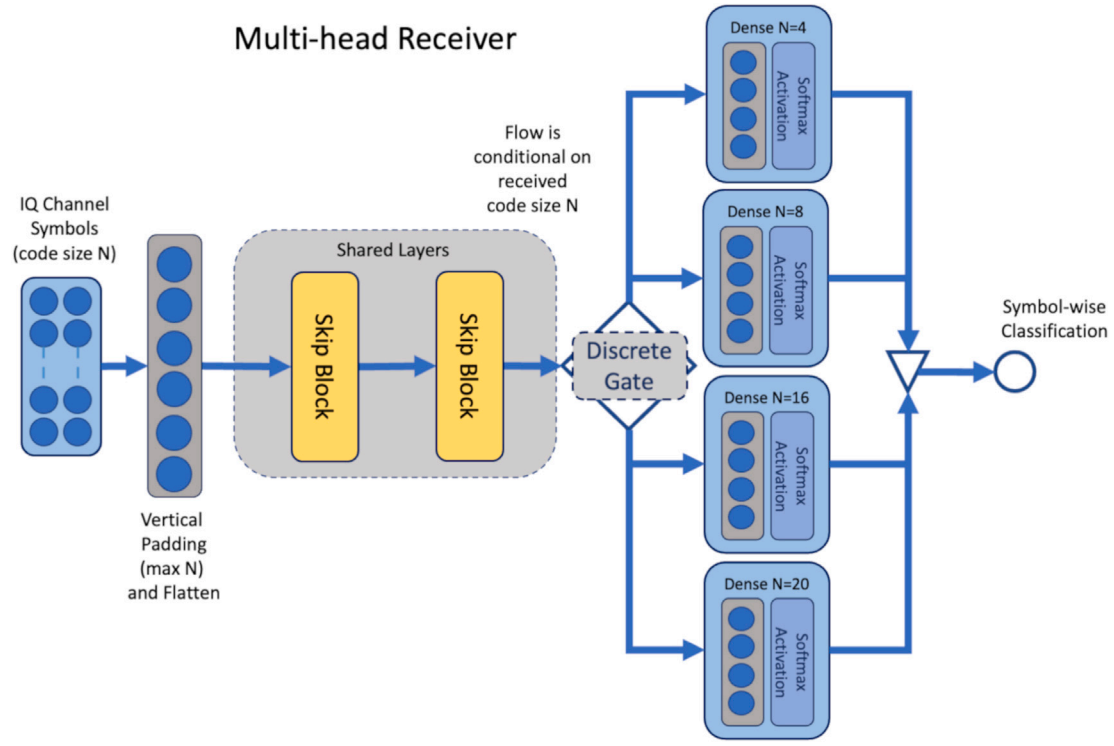


Fig. 4. The receiver architecture follows a similar approach to the transmitter where the shared path is followed by a discrete gate and a separate classification layer corresponding to each code size.

was gradually increased until an acceptable performance was achieved during training. The list of layers and the number of units in each layer of the transmitter network is shown in Table 1 and is shown for the receiver network in Table 2.

Each branch in the transmitter and receiver represents a code of size N. The code size parameter was supplied to the network as a choice from a set of code sizes $i \in \{i_i, \ldots, i_n\}$. Several variants of the network architecture were trained to evaluate the effect of additional code sizes (or increased number of tasks) on the overall performance of the model. The branching transmitter and receiver networks were trained for three message sizes, $K = 4$ bits, $K = 7$ bits and $K = 8$ bits. For each case, the network was trained to map K information bits to multiple code sizes of N channel symbols for transmission. For the $K = 4$ bit message size, the code sizes included $N = 4, 8, 16$ and 20, similarly for the $K = 7$ bit message size, code sizes $N = 11, 15,$ and 34 and finally for the $K = 8$ bit message configuration, code size $N = 6, 8, 17, 32$ and 40 were selected. The configurations for each model, message and code size are listed in Table 3. The table also lists the total number of trainable parameters in each neural network.

Table 1
The number of units in each layer of the transmitter model. The list of choices for code size indices i are provided as a parameter to the architecture. During training and inference the one-hot encoded message and the selected code size index i are provided as input to the network.

Layer	Units $K = 4$	Units $K = 7$	Units $K = 8$	Group
Input 1 $M = 2^K$ units	16	128	256	Input layers
Input 2 code size index i	1	1	1	
Code index embedding $2K$ units	8	14	16	
Concatenate $2K + 2^K$ units	24	142	272	
Dense layer	32	512	512	Skip block
Batch normalisation	–	–	–	
Activation (ReLU or Swish)	–	–	–	
Dense layer	64	64	64	
Batch normalisation	–	–	–	
Activation (ReLU or Swish)	–	–	–	
Dense layer	32	512	512	
Batch normalisation	–	–	–	
Activation (ReLU or Swish)	–	–	–	
Dense layer	32	512	512	Skip block
Batch normalisation	–	–	–	
Activation (ReLU or Swish)	–	–	–	
Dense layer	64	64	64	
Batch normalisation	–	–	–	
Activation (ReLU or Swish)	–	–	–	
Dense layer	32	512	512	
Batch normalisation	–	–	–	
Activation (ReLU or Swish)	–	–	–	
Gate layer	–	–	–	
Dense layer	$i = 4$, units $= 8$	$i = 11$, units $= 22$	$i = 6$, units $= 12$	Code size $i \in \{i, \ldots, i_n\}$ branches
Dense layer	$i = 8$, units $= 16$	$i = 15$, units $= 30$	$i = 8$, units $= 16$	
Dense layer	$i = 16$, units $= 32$	$i = 34$, units $= 68$	$i = 17$, units $= 34$	
Dense layer	$i = 20$, units $= 40$	–	$i = 32$, units $= 64$	
Dense layer	–	–	$i = 40$, units $= 80$	
Linear activation	–	–	–	Transmitter output for code size i
Reshape layer	$[2 \times i]$	$[2 \times i]$	$[2 \times i]$	
Dense layer	$[2 \times i]$	$[2 \times i]$	$[2 \times i]$	
Tanh activation	–	–	–	
Energy normalisation	–	–	–	

We compare the different model configurations against several conventional codes. The $K = 4$ bit model is compared with the MLD performance of a system that uses uncoded binary phase shift keying (BPSK) modulation with extended Hamming (8,4) code. The 7 bit model is compared with three BCH coded systems, two of which use shortened codes s-BCH(11,7) and s-BCH(34,7) derived from mother codes BCH(15,11) and BCH(63,36) respectively, with the additional code being BCH(15,7). The K = 8 bit model is compared with uncoded BPSK and a QRC(17,8) code.

In both architectures, the gate function at layer l, $f_g^{(l)}$ is parameterised by the code rate index i and input to the current layer $h^{(l)} = f^{(l-1)}\left(h^{(l-1)}\right)$ which selects from a set of branches comprising the next layer $f_i^{(l+1)} \in \left\{f_0^{(l+1)}, f_1^{(l+1)}, \ldots f_n^{(l+1)}\right\}$ (Eq. (1)). In the transmitter, the code rate index is supplied as an explicit parameter to the network, whereas in the receiver the code rate index is determined based on the number of symbols received from the channel. During the forward pass only one path through the branch is active (at the branch layer, the active branch will be $f_i^{(l+1)}$). During back-propagation, no gradients exist for the inactive branches, hence only the active branch receives the gradient update. Each of the respective shared paths in the transmitter and receiver, participate in back-propagation.

$$f_g^{(l)}\left(i, h^{(l)}\right) = f_i^{(l+1)}\left(h^{(l)}\right), \quad \text{where } f_i^{(l+1)} \in \left\{f_0^{(l+1)}, f_1^{(l+1)}, \ldots f_n^{(l+1)}\right\} \tag{1}$$

3.2. Training algorithm

It is important to consider a suitable regularisation approach to prevent negative impact on overall network performance between tasks. This is achieved with two approaches, randomised sampling for code size during training, and weight regularisation. Training consists of mini-batches (32 messages per batch) and code sizes are selected from a random uniform distribution each mini-batch. The update of each model is performed with back-propagation each mini-batch and the gradient is calculated for the selected code size. Over the course of learning, the weights for each layer in the network are stored and are averaged across mini-batches every ten iterations. This latter approach to regularisation is based on the stochastic weight averaging (SWA) performed in [9], and in the results we have observed that training using SWA produces better performance as opposed to those networks trained without SWA. SWA combined with a cyclical learning rate schedule [27] is demonstrated in [9] to improve the generalisation of the network. During training back-propagation is performed with the Adam optimiser [28] combined with the cyclical learning rate between 0.0001 and 0.001.

In addition to the sampling scheme and SWA regularisation, an alternating training algorithm is applied in four steps described in Algorithm 1 and Fig. 5. These steps consist of: (1) train the end-to-end network, (2) generate mini-batches using the transmitter, (3) train the receiver against a simulated channel and record the loss, and (4) update the end-to-end network. In Step 1, back-propagation is run on the end-to-end model which contains both the receiver and transmitter models. This updates the weights in both the receiver and transmitter models, and the AWGN channel is simulated directly as part of the end-to-end model architecture. During Step 2, the transmitter is used to generate the transmitter symbols and the channel is simulated independently of both transmitter and receiver models. In Step 3, the receiver is then trained using the channel response as the receiver input, and during back-propagation the receiver loss is calculated against the true messages. This allows the transmitter and receiver to be evaluated independently and the resulting receiver loss is used to coordinate

Algorithm 1 During training, the end-to-end model is trained iteratively with the receiver model.

Input: epochs ▷ The number of training iterations.
Input: batchSize ▷ The size of each training or validation batch.
Input: transmitModel ▷ The transmitter model.
Input: receiveModel ▷ The receiver model.
Input: endToEndModel ▷ The end-to-end model containing transmitter, channel and receiver models.
Input: channel ▷ The channel simulation function.
Input: snr ▷ An initial SNR dB for perturbation of training data.
Input: codeSizeList ▷ The set of allowed code lengths.
Output: endToEndModel ▷ The end-to-end model updated after training.

$loss \leftarrow \infty$
$weightsList \leftarrow []$

for $i \leftarrow 1, epochs$ **do**
  if Train with random SNR **then**
    $snr \leftarrow$ RANDOM-UNIFORM(0,9) ▷ Use randomised SNR to perturb data for training.
  end if
  codeSize <- RANDOM-UNIFORM(codeSizeList) ▷ Randomly select the code size for the training batch.
  TRAIN-ENDTOEND($batchSize, codeSize, snr, endToEndModel$) ▷ Fig. 5 (Step 1)
  $messages \leftarrow$ TRANSMIT-SAMPLES($batchSize, codeSize, snr, transmitModel, channel$) ▷ Fig. 5 (Step 2)
  $receiverLoss \leftarrow$ TRAIN-RECEIVER($messages, codeSize, receiveModel$) ▷ Fig. 5 (Step 3)
  if $receiverLoss < loss$ **then**
    $loss \leftarrow receiverLoss$
    SAVE($endToEndModel, transmitModel, receiveModel$) ▷ Save models each time learning improves at the receiver.
  end if
  if i mod 10 equals 0 **then**
    $w \leftarrow$ AVERAGE-WEIGHTS($weightsList$)
    SET-WEIGHTS($endToEndModel, w$) ▷ Apply weight averaging every 10 iterations.
  else
    $w \leftarrow$ GET-WEIGHTS($endToEndModel$)
    APPEND($weightsList, w$)
  end if
  TRAIN-ENDTOEND($batchSize, codeSize, snr, endToEndModel$) ▷ Fig. 5 (Step 4)
end for

return $endToEndModel$

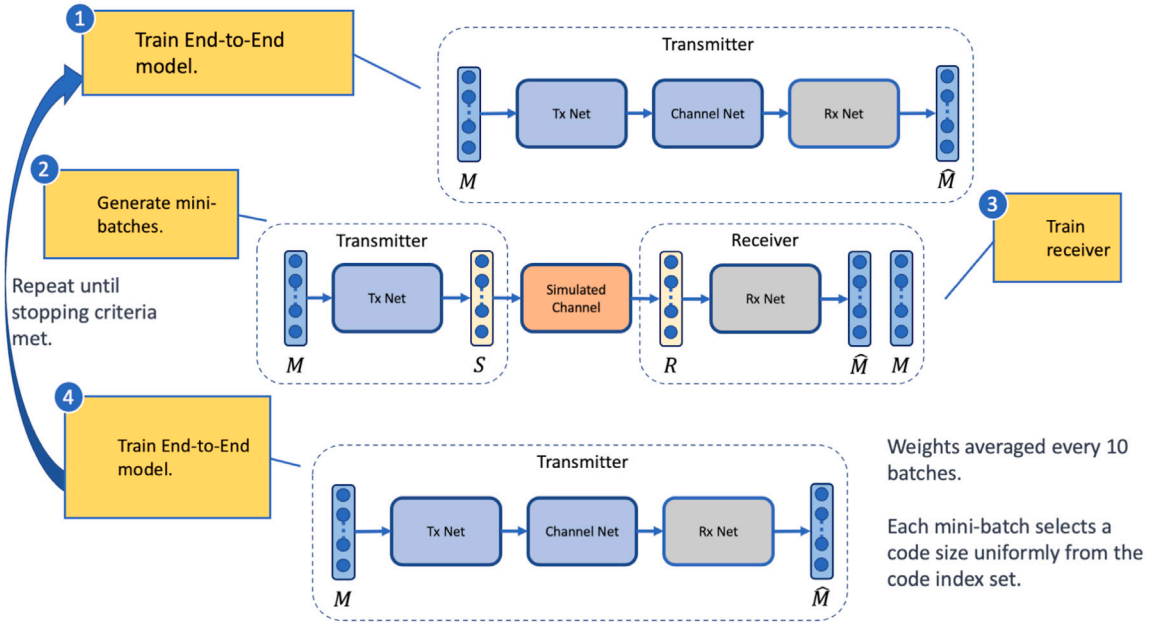


Fig. 5. Custom training algorithm consisting of several stages interleaving training of receiver and end-to-end model.

Table 2
The number of units in each layer of the receiver model. The list of choices for code size indices i are provided as a parameter to the architecture, and selected at runtime based on the length of the received channel symbols.

Layer	Units $K = 4$	Units $K = 7$	Units $K = 8$	Group
Input 1 $[2 \times i]$ units	$[2 \times i]$	$[2 \times i]$	$[2 \times i]$	Input from channel for code size i
Flatten Layer	$2i$	$2i$	$2i$	
Dense layer	32	512	512	Skip block
Batch normalisation	–	–	–	
Activation (ReLU or Swish)	–	–	–	
Dense layer	64	64	64	
Batch normalisation	–	–	–	
Activation (ReLU or Swish)	–	–	–	
Dense layer	32	512	512	
Batch normalisation	–	–	–	
Activation (ReLU or Swish)	–	–	–	
Dense layer	32	512	512	Skip block
Batch normalisation	–	–	–	
Activation (ReLU or Swish)	–	–	–	
Dense layer	64	64	64	
Batch normalisation	–	–	–	
Activation (ReLU or Swish)	–	–	–	
Dense layer	32	512	512	
Batch normalisation	–	–	–	
Activation (ReLU or Swish)	–	–	–	
Gate layer	–	–	–	
Dense layer 2^K units	16	128	256	Output $i \in \{i, \ldots, i_n\}$ Branches
Softmax activation	–	–	–	

Table 3
Multiple variations of the model are trained with separate configurations for message bits K and code size N. The total trainable parameters for each neural network counts all weights, biases, and batch normalisation parameters. The final column lists the conventional codes included in comparisons of BER and BLER selected channel conditions.

Configuration	Model variant	K	N	Transmitter parameters	Receiver parameters	Comparison codes
1	4 bit single model	4	$i \in \{4\}$	3831	4880	Uncoded BPSK
2	4 bit multi-rate model	4	$i \in \{4, 8, 16, 20\}$	14 471	13 888	Uncoded BPSK extended Hamming(8,4)
3	7 bit multi-rate model	7	$i \in \{11, 15, 34\}$	671 151	767 616	s-BCH(11,7) BCH(15,7) s-BCH(34,7)
4	8 bit multi-rate code	8	$i \in \{6, 8, 17, 32, 40\}$	649 125	1 101 696	Uncoded BPSK QRC(17,8)

intermittent checkpointing of both models. Finally, Step 4 updates the end-to-end network after weight averaging before the next training iteration.

3.3. Channel functions

The assumed channel function that is applied during training of the proposed model is the AWGN channel. Evaluation for the BER and BLER is made on three channels, the AWGN and two variants of Rayleigh fading differing in duration, the first applies fading to the entire block (Block fading), and the second varies symbol to symbol (Bit fading). When evaluation is carried out, the proposed models are not retrained or tuned for the two additional fading channels.

In AWGN (Eq. (2)) additive Gaussian noise $n(t)$ is added to the output of the transmitter $z(t)$, where t is the discrete time step of the transmitter output.

$$r(t) = z(t) + n(t) \tag{2}$$

Eq. (3) shows the Rayleigh fading coefficient $a(t)$, at each discrete time t, applied to the transmitted signal $z(t)$, prior to addition of additive noise $n(t)$. The fading coefficient $a(t) = \frac{1}{\sqrt{2}}|a|e^{j\psi}$, is drawn from a complex standard normal distribution $a \sim \mathcal{CN}(0, 1)$, and it's argument multiplied with the exponential waveform with phase parameter ψ, we assume a constant phase $\psi = 0$. The duration of the coefficient varies under block or bit fading. In addition we assume no channel estimation to reverse the effect of fading on the receiver.

$$r(t) = a(t)z(t) + n(t) \tag{3}$$

The additive Gaussian noise $n(t)$ is drawn from the complex normal distribution $n(t) \sim \mathcal{CN}(0, \sigma^2)$. The variance σ^2 is derived from the desired SNR and the final output of the transmitter layer $z(t)$, having $t = [1 \cdots T]$ discrete time steps. A desired level of noise is first supplied to the channel simulation as the ratio of energy per bit to noise E_b/N_0 dB. To account for the selected code rate k/n, the E_b/N_0 dB is converted to the ratio of energy per symbol to noise E_s/N_0 dB $= E_b/N_0$ dB $+ 10log_{10}(k/n)$. The components for E_s and N_0 are then estimated from the transmitter symbols $z(t)$ where $E_s = \frac{\sum_{t=1}^{T} z(t)^2}{T}$ and $N_0 = \frac{E_s}{E_s/N_0}$. The parameter E_s/N_0 is also commonly referred to as SNR. The variance is then estimated as $\sigma^2 = N_0/2$ and used to sample from the complex normal distribution.

The output at the transmitter is normalised by the energy constraint $\|x\|_2^2 \leq 1$ implemented in Eq. (4) where $x(t) \in \mathbb{C}$ is the sequence of complex symbols output by the tanh activation layer and T the number of time steps in the sequence. During training it is possible to vary the SNR dB randomly or to train at a constant SNR dB. A fixed SNR of 6 dB performed best for the 4 bit message, however 7 bit and 8 bit messages

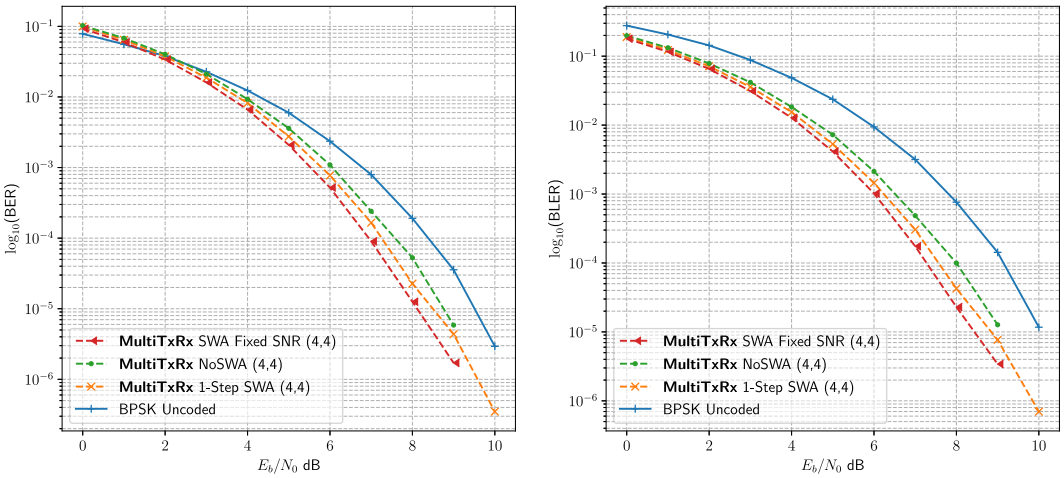


Fig. 6. BER and BLER in AWGN of several training methods, training with a standard back-propagation algorithm and SWA weight averaging (MultiTxRx 1-Step SWA), multi-step training without SWA (MultiTxRx NoSWA) and multi-step training with SWA and fixed SNR (MultiTxRx SWA Fixed SNR). Improvement in performance is indicated with the addition of SWA as well as when training with the multi-step training procedure as opposed to the standard back-propagation.

were trained with random SNR between $0-9$ dB.

$$z(t) = \frac{x(t)}{\sqrt{\sum_{i=1}^{T} x(i)^2/T}} \tag{4}$$

4. Results

In this section we report the empirical evaluation for the proposed model at different code rates, listed in Table 3, for the AWGN and the two fading channels. In each case we refer to the proposed model as the *MultiTxRx* model to indicate a multi-branching transmitter and receiver. The first evaluation investigates the performance of the proposed training algorithm. The second set of evaluations reports on the performance of different variations of the architecture. These two sets of evaluations are used to examine the design choices for the model structure and training approach. After this we compare the set of code size configurations from Table 3 with the conventional codes in the AWGN channel, and evaluate performance without retraining or tuning in the fading channels.

The performance of several training algorithms are evaluated in Fig. 6, with message size $K = 4$ bits and code size $N = 4$ (from configuration 1, Table 3). Uncoded BPSK performance is included for reference. The *MultiTxRx 1-Step SWA* model was trained using standard back-propagation and included weight averaging. *MultiTxRx NoSWA* (no weight averaging) and *MultiTxRx SWA* where trained with the multi-step algorithm described in Fig. 5. The multi-step algorithm with weight averaging (*MultiTxRx SWA*) produces lower BER and lower BLER in comparison to standard back-propagation (*MultiTxRx 1-Step SWA*). Training without weight averaging produces higher BER and BLER.

Next, we compare the changes made to the AE architecture in Fig. 7 by training on multiple code sizes from configuration 2 in Table 3. The different types of architecture shown in Fig. 7 include a *Single Path* AE, *Single Tx MultiRx*, *MultiTx SingleRx*, *MultiTxRx* and *MultiTxRx Residual*. The *Single Path* model consists of a single common path in the network. The *Single Tx MultiRx* model applies a single path with a pooling layer to realise multiple codes in the transmitter, and classifies multiple codes using branching in the receiver. The structure is reversed in the *MultiTx SingleRx*. When trained with multiple code sizes, *MultiTxRx* is similar to the proposed architecture but does not feature skip connections and the *MultiTxRx residual* is the proposed architecture, including skip connections. The *MultiTxRx Residual* model performs better than the other models and is close to the extended Hamming(8,4) BER. Both versions of the *MultiTxRx* model exhibit similar BLER.

All code rates from configuration 2 in Table 3 are compared under in the AWGN channel in Figs. 8 and 9 and in the Block Fading and Bit Fading channels in Figs. 10 and 11 respectively. In the AWGN channel, it is difficult to see the difference in performance between code rates in relation to E_b/N_0 (except for $K = 4, N = 4$). In contrast, Fig. 9 displays the BER and BLER related to the energy per symbol (E_s/N_0) SNR dB. This example demonstrates that the smaller code rates can achieve lower BER and BLER as the channel noise increases at the cost of increased channel usage due to the increase in code size N. The aim of an AMC scheme is to maintain performance by trading off channel use in varying SNR.

In the Block Fading channel (Fig. 10) we observe that the BER is much higher than the uncoded BPSK while the BLER is much lower. The BER is higher because symbol-wise classification does not perform error correction of individual bits. An error on a code word may contain more incorrect bits in a single forward pass estimation. However, this approach achieves better BLER, as it can accurately classify, or map, the entire code word for a corresponding message. In the Block Fading channel, the entire code word is impacted by the channel fading. Bit-interleaving techniques can be applied in this circumstance which can produce an effect that is similar to a Bit Fading channel prior to decoding. The difference between the code rates is most noticeable in the Bit Fading channel (Fig. 11), performance improves as the code rate decreases (at the expense of channel use). In this channel, the $K = 4, N = 8$ code is slightly better than the baseline extended Hamming(8,4) code, as opposed to the AWGN channel. In the AWGN channel, code rate $K = 4, N = 8$ is close to the baseline extended Hamming(8,4) code, but differs slightly in higher SNR. The model is not retrained on either of the fading channels and is able to perform close to or better than the baseline.

Fig. 12 displays the performance of the $K = 7$ bit message and code rates from configuration 3 of Table 3 in the AWGN channel. The figure shows slight gains for BLER over the shortened s-BCH(11,7) and BCH(15,7) codes, and similar performance to the shortened s-BCH(34,7) code. There is less difference in BER performance for these codes. Under the Block Fading channel in Fig. 13 the BER is again higher, but the BLER is lower in comparison to the reference codes. In the Bit Fading channel, shown in Fig. 14, incremental gains are achieved on all code rates in comparison to the BCH and shortened codes (Fig. 14).

Configuration 4 from Table 3 for the $K = 8$ bit message and selected code sizes, is compared with the uncoded BPSK and the QRC code in the AWGN (Fig. 15), Block Fading (Fig. 16) and Bit Fading (Fig. 17) channels. In AWGN the BER produced by the model at the lower code

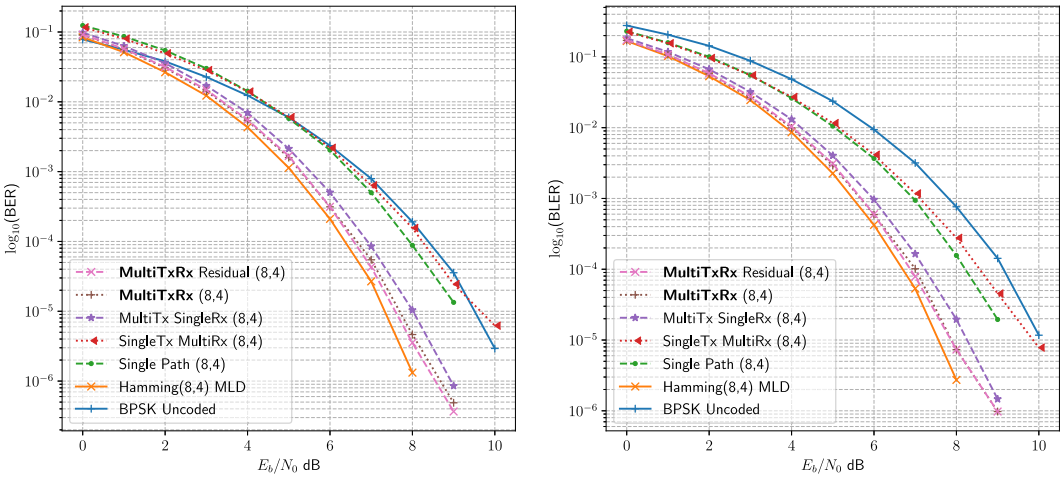


Fig. 7. The multi-branching Tx Rx architecture is compared with four variants of the architecture, a non-branching single path architecture (Single Path), a single branch transmitter with multi branch receiver (SingleTx MultiRx), the multi-branch transmitter and single receiver (MultiTx SingleRx) and multiple branching transmitter receiver with and without residual connections (MultiTxRx Residual and MultiTxRx). The choice of network architecture influences performance for the multi-task estimation of multiple code rates.

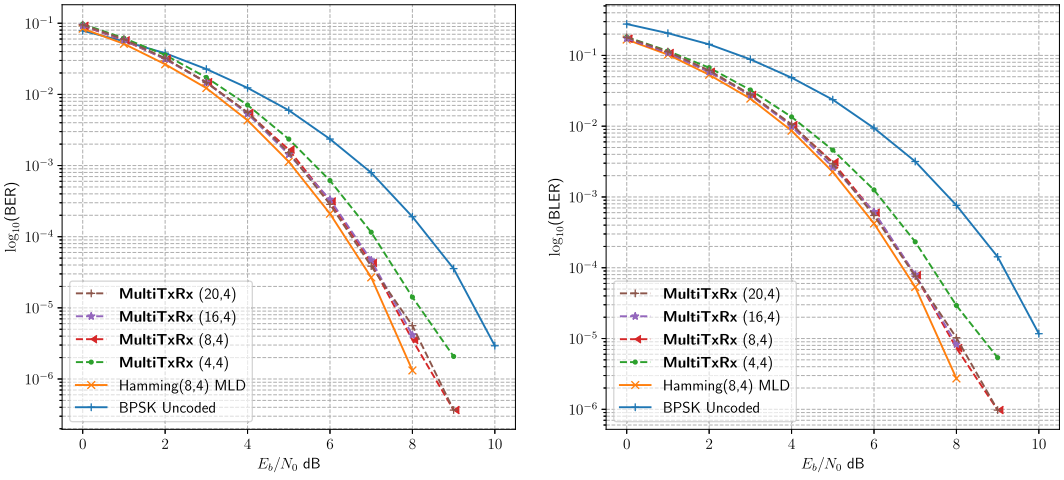


Fig. 8. BER and BLER in AWGN for MultiTxRx model with K = 4 bits and N = [4, 8, 16, 20] compared with BPSK uncoded and extended Hamming(8,4) maximum likelihood decoding (MLD).

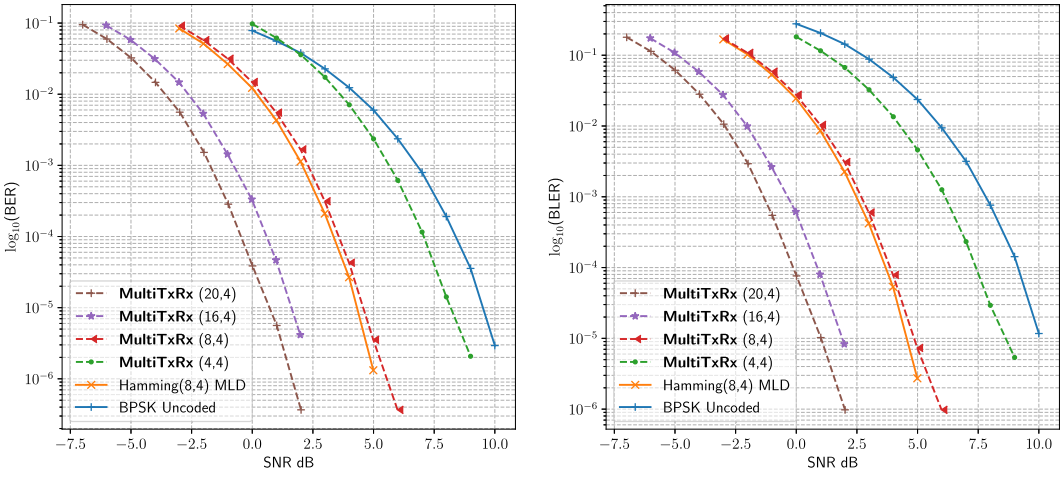


Fig. 9. The coding gain for each respective code rate is visible when plotting the BER and BLER in AWGN over the energy per symbol (E_s/N_0) SNR dB. The advantage of learning multiple codes enables operation under increased noise in the channel.

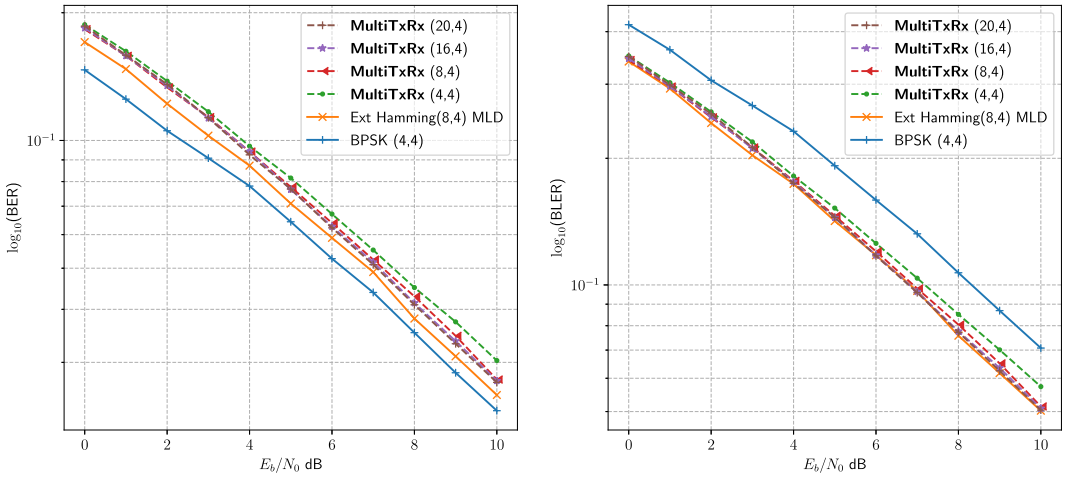


Fig. 10. BER and BLER in the Block Fading channel for MultiTxRx model with K = 4 bits and N = [4, 8, 16, 20] compared with BPSK uncoded and extended Hamming(8,4) maximum likelihood decoding (MLD). The proposed module was originally trained on the AWGN channel and is not trained for the Block Fading channel.

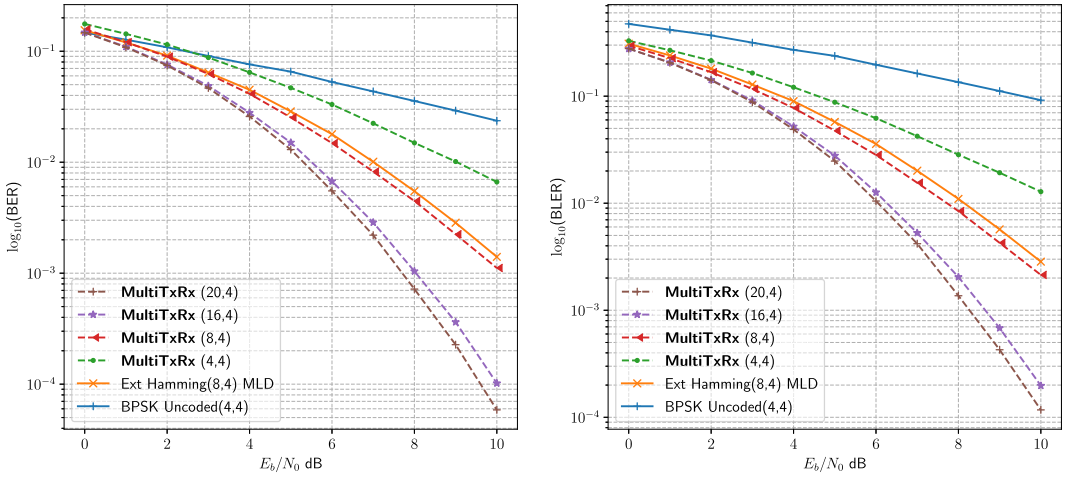


Fig. 11. BER and BLER in the bit fading channel for MultiTxRx model with K = 4 bits and N = [4, 8, 16, 20] compared with BPSK uncoded and extended Hamming(8,4) maximum likelihood decoding (MLD). The proposed module was originally trained on the AWGN channel and is not trained for the bit fading channel.

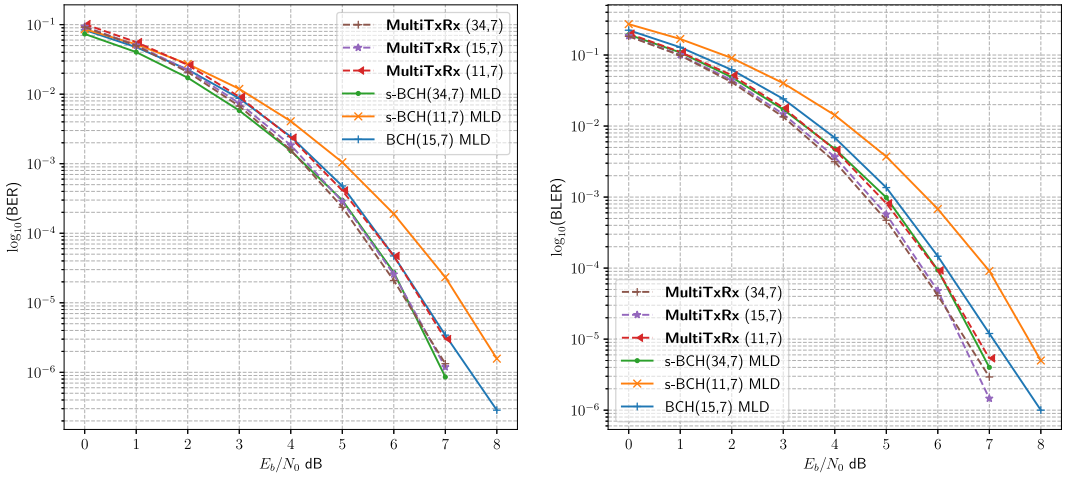


Fig. 12. BER and BLER in the AWGN channel for MultiTxRx model with K = 7 bits and N = [11, 15, 34] compared with shortened BCH codes s-BCH(11,7), s-BCH(34,7) and BCH code (15,7) maximum likelihood decoding (MLD), and trained with random SNR.

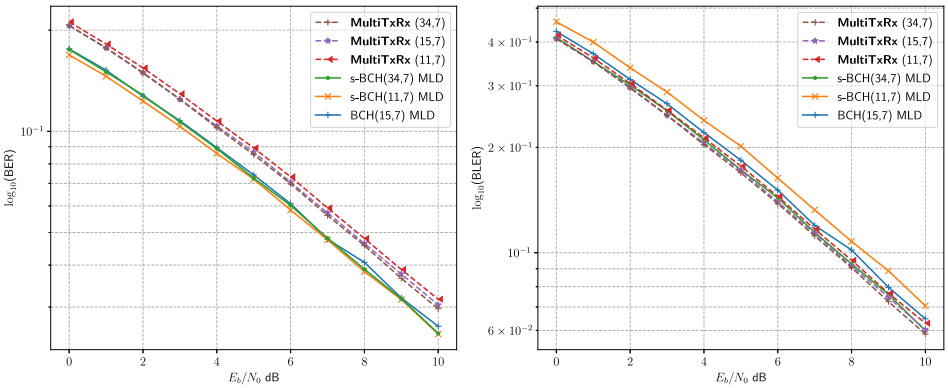


Fig. 13. BER and BLER in the Block Fading channel for MultiTxRx model with K = 7 bits and N = [11, 15, 34] compared with shortened BCH codes s-BCH(11,7), s-BCH(34,7) and BCH code (15,7) maximum likelihood decoding (MLD). The proposed module was originally trained on the AWGN channel and is not trained for the Block Fading channel.

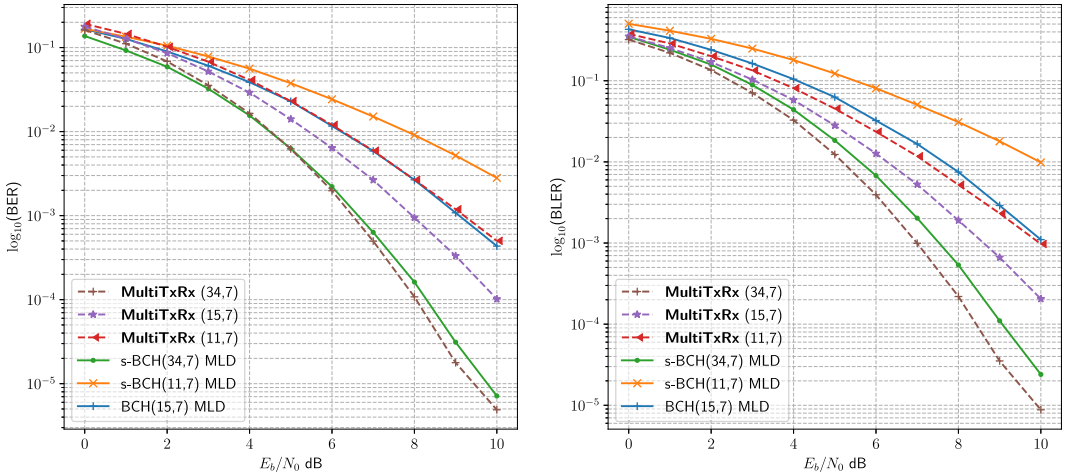


Fig. 14. BER and BLER in the bit fading channel for MultiTxRx model with K = 7 bits and N = [11, 15, 34] compared with shortened BCH codes s-BCH(11,7), s-BCH(34,7) and BCH code (15,7) maximum likelihood decoding (MLD). The proposed module was originally trained on the AWGN channel and is not trained for the bit fading channel.

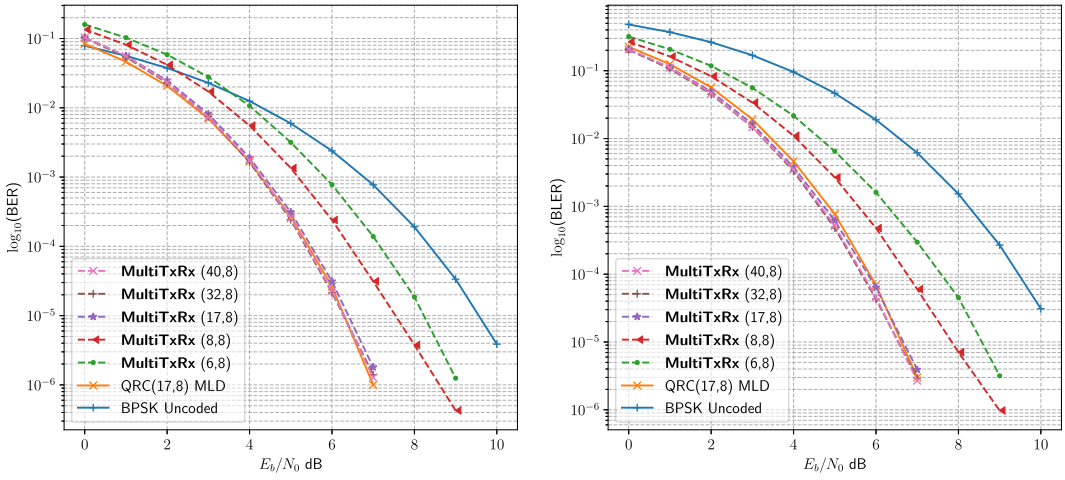


Fig. 15. BER and BLER in the AWGN channel for MultiTxRx model with K = 8 bits and N = [6, 8, 17, 32, 40] compared with BPSK uncoded and Quadratic Residue Code (QRC) K = 8, N = 17 maximum likelihood decoding (MLD), and trained with random SNR. The code (6,8) provides a higher channel usage than uncoded BPSK at 1.33 bits per channel usage.

rates is similar to the baseline code QRC(17,8) and BLER is slightly lower. The BER in the Block Fading channel, shown in Fig. 16, is worse than the target baseline QRC code, however, as we have seen in the other configurations, the BLER for the same code size and lower code rates is slightly better than the reference code. In the Bit Fading channel, both BER and BLER achieve equal or better performance than the reference QRC(17,8) code. However, the BER for higher code rates $K = 8, N = 8$ and $K = 8, N = 6$ do not perform as well as the uncoded BPSK in lower SNR, but do achieve gains for the BLER. This is also apparent in the AWGN channel.

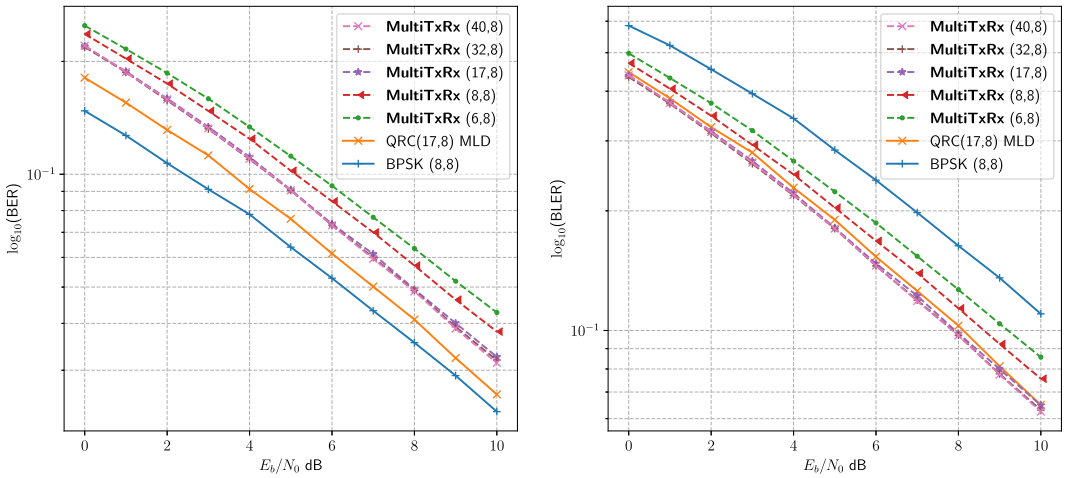


Fig. 16. BER and BLER in the Block Fading channel for MultiTxRx model with K = 8 bits and N = [6, 8, 17, 32, 40] compared with BPSK uncoded and Quadratic Residue Code (QRC) K = 8, N = 17 maximum likelihood decoding (MLD). The proposed module was originally trained on the AWGN channel and is not trained for the Block Fading channel.

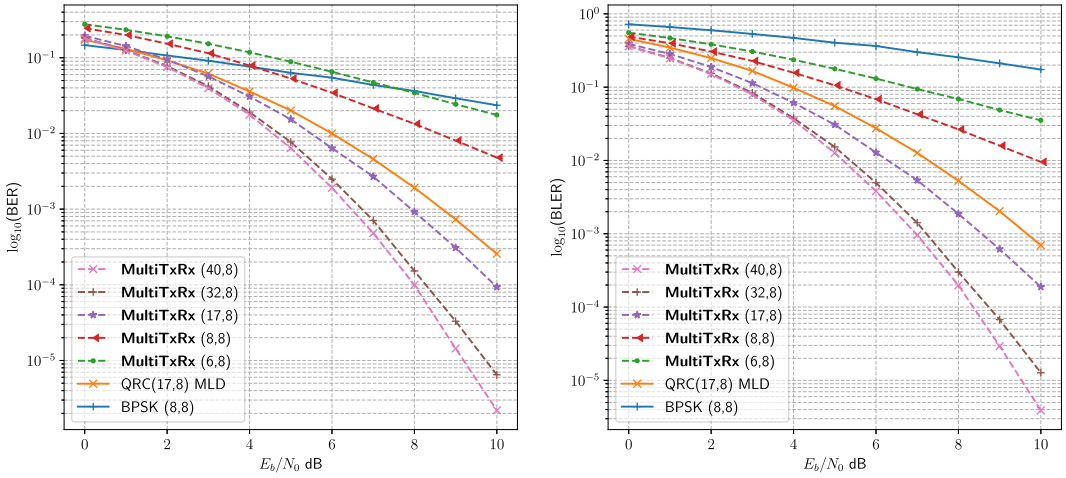


Fig. 17. BER and BLER in the bit fading channel for MultiTxRx model with K = 8 bits and N = [6, 8, 17, 32, 40] compared with BPSK uncoded and Quadratic Residue Code (QRC) K = 8, N = 17 maximum likelihood decoding (MLD). The proposed module was originally trained on the AWGN channel and is not trained for the bit fading channel.

5. Discussion

Comparison of the proposed model with the selected codes s-BCH(11,7), BCH(15,7), s-BCH(34,7) and QRC(17,8), demonstrated lower BLER in each of the channels and notably under the Bit Fading channel without retraining. While the BLER was close to the extended Hamming(8,4) code in each of the channels. In both the AWGN and Block Fading channels the BER was often poorer than the comparison code. As noted this is due to the classification for an entire code word rather than at the bit level and for the Block Fading channel, this effect of fading can be mitigated through the use of bit interleaving. However, the performance of a code is also dependent on the smallest minimum distance between all code words. Since the transmitter learns continuous codes, instead of binary codes, the minimum Euclidean distance is more appropriate measure of distance for those codes.

Fig. 18 shows the Euclidean distances between each of the learnt code words in the $K = 4, N = 8$ code. Ideally the transmitter should learn a constellation related to the distance between messages. In some cases, there is a larger distance between message code words with a message Hamming distance of 1, than those message code words with a larger message Hamming distance. For example, the Euclidean distance between code words for messages 0000 and 0001, a Hamming distance of 1, is larger than the Euclidean distance between code words for messages 0000 and 0111 with a Hamming distance of 3. The confusion matrix for the classifier is shown in Fig. 19, for messages 0000 and 0111 the percentage of incorrect classifications is approximately 3%, slightly higher than the incorrect classification between 0000 and 0001. The minimum Euclidean distance of the code does appear to be related to the performance of the learnt code. For those codes which have a lower BLER than the comparative code, the minimum Euclidean distance and mean Euclidean distances are close to or exceed that of the corresponding code. Table 4 lists the minimum, mean and variance of the Euclidean distance d_E calculated for the constellations of the learnt and comparison codes.

The changes to the AE to support multiple code rates does require an increase in the number of parameters overall within the neural network. This is to support generalisation over multiple code rates. However, the use of a common shared path for multiple codes does reduce the total number of parameters required in comparison to training separate models. The size of four single AE neural networks are shown in Table 5. The proposed branching model requires less parameters in a branching AE that can produce four different code rates in comparison to four separate AE.

The proposed models produced gains in BLER in comparison to the conventional codes under each of the channels. However it is not clear whether to attribute this gain to the learnt code or the inference supported by the AE. To investigate this, we developed a table based transmitter and MLD receiver for the code rate $K = 7, N = 15$. Symbols

Table 4
Computed minimum, mean and variance of euclidean distances for learnt and BPSK modulated reference codes.

Code rate	$d_{E_{min}}$	$E[d_E]$	$Var[d_E]$	Reference code	Code $d_{E_{min}}$	Code $E[d_E]$	Code $Var[d_E]$
K = 4, N = 8	3.83	4.12	0.08	Ext Hamming(8,4)	4	4.11	0.17
K = 7, N = 11	3.67	4.69	0.2	s-BCH(11,7)	3.46	4.66	0.47
K = 7, N = 15	4.38	4.38	0.19	BCH(15,7)	4.47	5.46	0.44
K = 7, N = 34	6.91	8.3	0.29	s-BCH(34,7)	6.63	8.25	0.53
K = 8, N = 17	4.34	5.82	0.23	QRC(17,8)	4.9	5.8	0.47

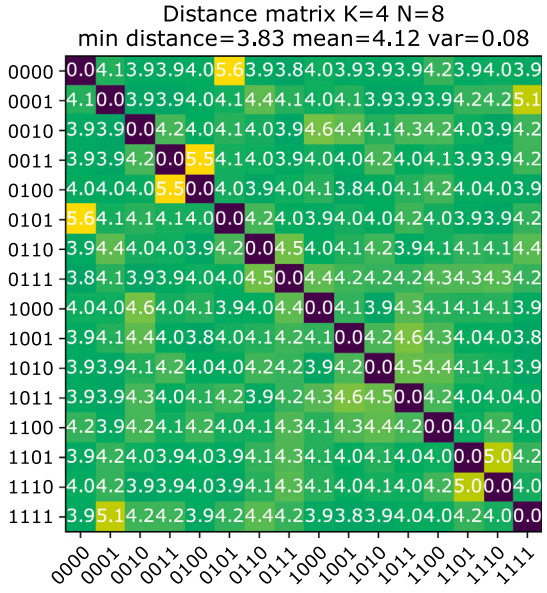


Fig. 18. Euclidean distances between pairwise codewords for each input sequence, learnt by the K = 4, N = 8 MultiTxRx auto-encoder.

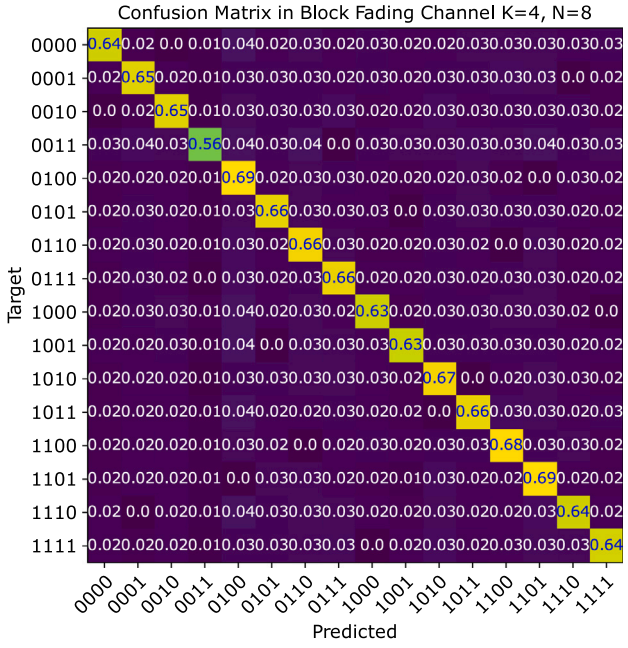


Fig. 19. The confusion matrix for the K = 4, N = 8 code rate under the block fading channel. Figures are relative to the predicted labels. While the classifier achieves a high level of accuracy on the BLER, there is sufficient difference between messages to cause high BER.

for corresponding 7 bit messages output by the *MultiTxRx* K = 7, N = 15 model were stored in a lookup table and transmitted over a AWGN channel. If the gain was solely due to the learnt receiver, we expect the MLD receiver to exhibit higher BLER. The MLD receiver performed nearest neighbour decoding for received channel values against the table of modulated symbols. The performance of the MLD receiver matched the performance of the proposed branching AE model in the corresponding channel (Fig. 20). This indicates that the gains observed are generated due to the learnt constellations resulting from training. This approach demonstrates potential use of DL for wireless communications as a method for code design which may be applied independently of the trained model.

Table 5
The number of parameters for combined separate code rate models versus the multi-task shared path model. The shared path architecture provides less total parameters than separate models for each code rate.

Model variant	K	N	Parameters
K = 4 N = 4 bit model	4	$i \in \{4\}$	8711
K = 4 N = 8 bit model	4	$i \in \{8\}$	9951
K = 4 N = 16 bit model	4	$i \in \{16\}$	12 431
K = 4 N = 20 bit model	4	$i \in \{20\}$	13 671
Total			44 764
4 bit multi-rate model	4	$i \in \{4, 8, 16, 20\}$	28 359

6. Conclusions and future work

In this article we have presented a branching AE architecture capable of automatically learning multiple code rates for AMC schemes. The proposed branching architecture extends applications of the AE architecture beyond the learning of a single code rate to the learning of multiple code rates. The choice of assumed channel during training is highly influential to the resulting performance of the AE in other channels. As a result, the ability to train a receiver separately on a real channel provides the ability to further optimise the system performance after deployment. The proposed branching AE for multiple code rates, is demonstrated to perform well under a variety of changing channel conditions, achieving gains in BLER compared to the selected conventional codes. By leveraging an AMC scheme the approach offers the potential to mitigate the requirement of receiver tuning in AE for wireless communications. However, there remains a number of limitations for the practical application of the DL approach requiring further investigation.

First, in this article we have assumed perfect synchronisation at the receiver. While it is possible to apply conventional methods for synchronisation with learnt modulation and coding schemes, it is desirable that synchronisation be addressed as part of the end-to-end AE architecture.

Second, classification based architectures not only do not scale to higher message lengths, but cannot provide error correction functionality. Hence work on bit-wise decoding for longer message lengths, either as part of a concatenated code, or as a standalone network will be a significant part of the practical application of such models, some of this work has already been described in the related work section of this paper.

Third, there has been work investigating the sensitivity of such architectures to their training conditions and whether they are brittle in terms of adversarial attacks. While we do not directly explore this concern, there is a connection between network regularisation and training methods required to mitigate adversarial attacks. In [29]

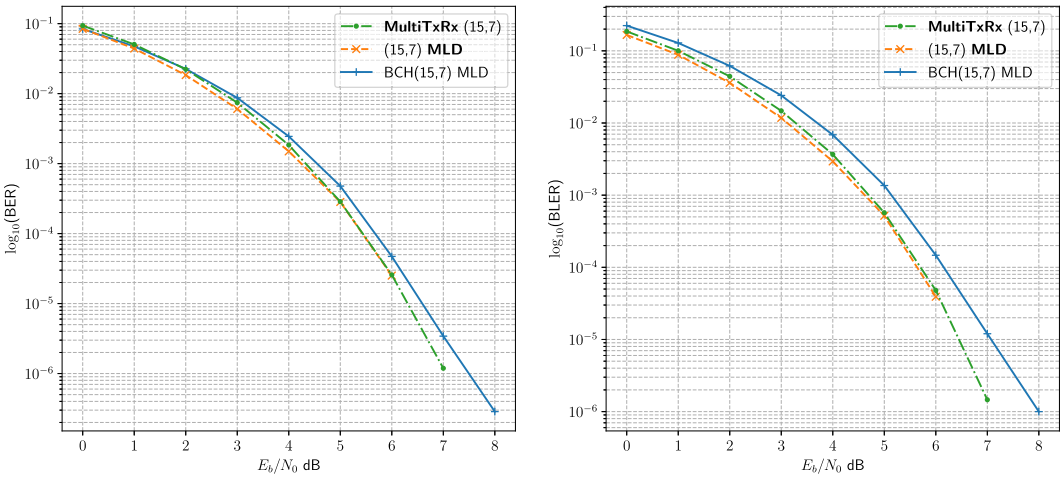


Fig. 20. BER and BLER in the AWGN channel of the learnt constellation for the (15,7) code produced by a table based transmitter and MLD receiver compared with the end-to-end model and BCH(15,7) code.

conventional Hamming codes are shown to be more robust under adversarial and jamming attacks than AEs. It is suggested in [30] that adversarial examples are transferable across different models, thereby enabling black-box attacks. This raises the importance for future investigation into regularisation methods for end-to-end learning in wireless communications and evaluation under adversarial interference.

Fourth, as we have discussed, the Euclidean distances between messages for neighbouring codes may be larger those several message bits away. This negatively impacts the performance of the BER, as misclassification results in a higher number of incorrect bits. Future work should investigate the ability of the transmitter to learn distance based relationships between source messages. In addition, while we have assumed no channel information at the receiver, it may be possible to incorporate or learn such information to enhance receiver performance in the end-to-end learning scenario.

The tuning of the receiver over varying channel conditions would be time consuming in a deployed system and may lead to poor performance on the original channel, for which it was first trained. Whether it is practical to tune a receiver over the air, and how much training is required, is a matter for consideration. A practical solution may be to use a branching AE with multiple code rates under changing conditions. This would permit operation whilst a separate model is adapted in the background. The question of how to update such a model while mitigating catastrophic forgetting in changing channel conditions deserves further investigation.

Finally, the mapping between channel environment and choice of code rate relies on measurements such as expected BER and BLER over associated SNR. It is feasible to imagine the joint learning of AMC and channel performance mapping, extending the work described in [31,32]. More recent research in the industrial internet of things (IIoT) consider wireless communications as part of a joint optimisation objective in seeking to reduce energy consumption over the collective sensor network [33,34]. A potential application would be to learn energy efficient communication schemes for the IIoT setting, which are adaptive to operational constraints in addition to channel conditions, in an end-to-end manner.

The flexibility of the AE architecture provides competitive performance not only in learning a single code rate, but also as we have shown, in learning AMC schemes with varying error-rate performance and spectral efficiencies. By framing the learning problem as MTL, the proposed architecture enables the deployment of a single model, instead of requiring multiple separate models for each code rate.

CRediT authorship contribution statement

Christopher P. Davey: Conceptualisation, Methodology, Implementation. **Ismail Shakeel:** Conceptualisation, Coordination, Editorial. **Ravinesh C. Deo:** Coordination, Editorial. **Ekta Sharma:** Editorial. **Sancho Salcedo-Sanz:** Coordination, Editorial. **Jeffrey Soar:** Coordination, Editorial.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data used in this article is generated through simulation. The simulation process is described in the methodology section of the article, alongside the energy constraints and channel distortions.

Acknowledgements

This research is supported by UniSQ-DSTG Postgraduate Research Scholarship 2021–2024 on the 'Design of Efficient Artificial Intelligence Algorithms for Future Communication Systems'. It is funded by the Department of Defence, Commonwealth of Australia under a DSP Scholarship (Project-Based) Agreement 10254.

References

[1] G. Caire, K.R. Kumar, Information theoretic foundations of adaptive coded modulation, Proc. IEEE 95 (12) (2007) 2274–2298.
[2] Timothy O'Shea, Jakob Hoydis, An introduction to deep learning for the physical layer, IEEE Trans. Cogn. Commun. Netw. 3 (4) (2017) 563–575.
[3] S. Dörner, S. Cammerer, J. Hoydis, S. t. Brink, Deep learning based communication over the air, IEEE J. Sel. Top. Sign. Proces. 12 (1) (2018) 132–143.
[4] Hao Ye, Le Liang, Geoffrey Ye Li, Biing-Hwang Juang, Deep learning-based end-to-end wireless communication systems with conditional GANs as unknown channels, IEEE Trans. Wirel. Commun. 19 (5) (2020) 3133–3143.
[5] F.A. Aoudia, J. Hoydis, End-to-end learning of communications systems without a channel model, in: 2018 52nd Asilomar Conference on Signals, Systems, and Computers, 2018, pp. 298–303.
[6] Michael Crawshaw, Multi-task learning with deep neural networks: A survey, 2020, arXiv preprint arXiv:2009.09796.
[7] Michael McCloskey, Neal J. Cohen, Catastrophic interference in connectionist networks: The sequential learning problem, in: Psychology of Learning and Motivation, Vol. 24, Elsevier, 1989, pp. 109–165.

[8] Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, Yoshua Bengio, An empirical investigation of catastrophic forgetting in gradient-based neural networks, 2013, arXiv preprint arXiv:1312.6211.

[9] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, Andrew Gordon Wilson, Averaging weights leads to wider optima and better generalization, 2018, arXiv preprint arXiv:1803.05407.

[10] Fayçal Ait Aoudia, Jakob Hoydis, Model-free training of end-to-end communication systems, IEEE J. Sel. Areas Commun. 37 (11) (2019) 2503–2516.

[11] S. Cammerer, F.A. Aoudia, S. Dörner, M. Stark, J. Hoydis, S. ten Brink, Trainable communication systems: Concepts and prototype, IEEE Trans. Commun. 68 (9) (2020) 5489–5503.

[12] N.A. Letizia, A.M. Tonello, Capacity-driven autoencoders for communications, IEEE Open J. Commun. Soc. 2 (2021) 1366–1378.

[13] Rich Caruana, Multitask Learning (Ph.D. thesis), Carnegie Mellon University, Pittsburgh, PA, 1998.

[14] Kevis-Kokitsi Maninis, Ilija Radosavovic, Iasonas Kokkinos, Attentive single-tasking of multiple tasks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 1851–1860.

[15] E. Armanious, D.D. Falconer, H. Yanikomeroglu, Adaptive modulation, adaptive coding, and power control for fixed cellular broadband wireless systems: some new insights, in: 2003 IEEE Wireless Communications and Networking, 2003, WCNC 2003, Vol. 1, 2003, pp. 238–242, vol.1.

[16] Joseph Downey, Dale Mortensen, Michael Evans, Janette Briones, Nicholas Tollis, Adaptive coding and modulation experiment using NASA's space communication and navigation testbed, in: 2016 Communications Satellite Systems Conference, ICSSC, 2016.

[17] Intae Hwang, Taewon Jang, Mingoo Kang, Sangmin No, Jungyoung Son, Daesik Hong, Changeon Kang, Performance analysis of adaptive modulation and coding combined with transmit diversity in next generation mobile communication systems, Future Gener. Comput. Syst. 20 (2) (2004) 189–196.

[18] D. Wu, S. Ci, Cross-layer design for combining adaptive modulation and coding with hybrid ARQ to enhance spectral efficiency, in: 2006 3rd International Conference on Broadband Communications, Networks and Systems, 2006, pp. 1–6.

[19] A.J. Goldsmith, S.G. Chua, Adaptive coded modulation for fading channels, IEEE Trans. Commun. 46 (5) (1998) 595–602.

[20] Shu Lin, Juane Li, Fundamentals of Classical and Modern Error-Correcting Codes, Cambridge University Press, Cambridge, 2021, URL https://www.cambridge.org/core/books/fundamentals-of-classical-and-modern-errorcorrecting-codes/19A81ED5D7E9C6A1EBB9657683B6E39C.

[21] Florence Jessie MacWilliams, Neil James Alexander Sloane, The Theory of Error-Correcting Codes, vol. 16, Elsevier, 1977.

[22] Raj Chandra Bose, Dwijendra K. Ray-Chaudhuri, On a class of error correcting binary group codes, Inf. Control 3 (1) (1960) 68–79.

[23] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2016, pp. 770–778.

[24] Sergey Ioffe, Christian Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: Francis Bach, David Blei (Eds.), Proceedings of the 32nd International Conference on Machine Learning, in: Proceedings of Machine Learning Research, vol. 37, PMLR, 2015, pp. 448–456, URL https://proceedings.mlr.press/v37/ioffe15.html.

[25] Xavier Glorot, Antoine Bordes, Yoshua Bengio, Deep sparse rectifier neural networks, in: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, JMLR Workshop and Conference Proceedings, 2011, pp. 315–323.

[26] Prajit Ramachandran, Barret Zoph, Quoc V. Le, Searching for activation functions, 2017, arXiv preprint arXiv:1710.05941.

[27] L.N. Smith, Cyclical learning rates for training neural networks, in: 2017 IEEE Winter Conference on Applications of Computer Vision, WACV, 2017, pp. 464–472.

[28] Diederik P. Kingma, Jimmy Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.

[29] M. Sadeghi, E.G. Larsson, Physical adversarial attacks against end-to-end autoencoder communication systems, IEEE Commun. Lett. 23 (5) (2019) 847–850.

[30] Shan Ai, Arthur Sandor Voundi Koe, Teng Huang, Adversarial perturbation in remote sensing image recognition, Appl. Soft Comput. 105 (2021) 107252, URL https://www.sciencedirect.com/science/article/pii/S1568494621001757.

[31] S. Kojima, K. Maruta, C.J. Ahn, Adaptive modulation and coding using neural network based SNR estimation, IEEE Access 7 (2019) 183545–183553.

[32] P.V.R. Ferreira, R. Paffenroth, A.M. Wyglinski, T.M. Hackett, S.G. Bilen, R.C. Reinhart, D.J. Mortensen, Reinforcement learning for satellite communications: From LEO to deep space operations, IEEE Commun. Mag. 57 (5) (2019) 70–75.

[33] Sarogini Grace Pease, Russell Trueman, Callum Davies, Jude Grosberg, Kai Hin Yau, Navjot Kaur, Paul Conway, Andrew West, An intelligent real-time cyber-physical toolset for energy and process prediction and optimisation in the future industrial internet of things, Future Gener. Comput. Syst. 79 (2018) 815–829, URL https://www.sciencedirect.com/science/article/pii/S0167739X1630382X.

[34] Jiwei Huang, Han Gao, Shaohua Wan, Ying Chen, Aoi-aware energy control and computation offloading for industrial IoT, Future Gener. Comput. Syst. 139 (2023) 29–37, URL https://www.sciencedirect.com/science/article/pii/S0167739X22002916.

Christopher P. Davey has a Master of Information Technology degree from the Queensland University of Technology (QUT, Australia) in 2007 and completed a Master of Science majoring in mathematics and statistics from The University of Southern Queensland (UniSQ, Australia) in 2020. Chris has over a decade of professional experience in software development and systems integration. He is currently progressing work on a PhD program at UniSQ with the focus of his research being on "Deep Learning for Wireless Communications". He has worked on "Artificial Intelligence for Decision-Making (AI4DM)" and "AI-enabled communicating systems" research project funded by the Australian Government's Department of Defence.

Ismail Shakeel received a Ph.D. degree in Telecommunications in 2007 and a BEng (Hons) degree in Electronic Engineering in 1997 from the University of South Australia. He has completed two master's degrees at the University of Canterbury (NZ) and Monash University in 2001 and 2002 respectively. Ismail joined Defence Science and Technology Group (DSTG) in 2011 and is currently with the Information Sciences Division at DSTG. Ismail is also an Adjunct Professor at the University of Southern Queensland. Before joining DSTG, he has worked in both academia and industry and holds a patent and generated more than 40 technical reports and publications in the field of telecommunications. Ismail's current research interests include signal detection and classification techniques, artificial intelligence-enabled wireless communication, interference-resistant signalling, chaotic communication, and cooperative wireless communication.

Ravinesh C. Deo is a Highly Cited Author, 2021 Clarivate) leads UniSQ's Advanced Data Analytics Lab as Professor at the University of Southern Queensland (UniSQ), Australia. He is a Clarivate Highly Cited Researcher with publications ranking in top 1% by citations for field and publication year in the Web of Science citation index and is among scientists and social scientists who have demonstrated significant broad influence, reflected in the publication of multiple papers frequently cited by their peers. He leads cross-disciplinary research in deep learning and artificial intelligence, supervising 20+ Ph.D./M.Sc. Degrees. He has received Employee Excellence Awards, Elsevier Highly Cited Paper Awards, and Publication Excellence and Teaching Commendations. He has published more than 270 articles, 150 journals, and seven books with a cumulative citation that exceeds 11,600.

Ekta Sharma holds a Ph.D. degree in Artificial Intelligence from University of Southern Queensland, Australia. She completed her M.Phil., M.Sc. (Operations Research), and B.Sc. (Mathematical Sciences), from the University of Delhi, India. She has over a decade of experience in both Academia and Industry across varied roles from Area Manager to Learning Advisor, Lecturer, and Researcher at Universities in Europe, India, and Australia. Her research work has received funding from the Australian Government, the Australian Defence Science and Technology Group, The Australian Mathematical Sciences Institute, and the Australian Tropical Agriculture Institute. She is working on varied cross-disciplinary research projects in artificial intelligence and Wireless Communications.

Sancho Salcedo-Sanz was born in Madrid, Spain, in 1974. He received the B.S degree in Physics from Universidad Complutense de Madrid, Spain, in 1998, the Ph.D. degree in Telecommunications Engineering from the Universidad Carlos III de Madrid, Spain, in 2002, and the Ph.D. degree in Physics from Universidad Complutense de Madrid in 2019. He spent one year in the School of Computer Science, The University of Birmingham, U.K, as postdoctoral Research Fellow. Currently, he is a Full Professor at the department of Signal Processing and Communications, Universidad de Alcalá, Spain. He has co-authored more than 240 international journal papers in the field of Machine Learning and Soft-Computing and its applications. His current interests deal with Soft-computing techniques, hybrid algorithms and neural networks in different problems of Science and Technology.

Jeffrey Soar is Personal Chair in Human-Centered Technology at the School of Business, University of Southern Queensland. His research is in AI, e-business, e-health, technology and development, and social and organisational change. He came to academic research from a long and distinguished career in industry including as chief information officer in government agencies in Australia and New Zealand.