Low Complexity Autoencoder based End-to-End Learning of Coded Communications Systems

Nuwanthika Rajapaksha, Nandana Rajatheva, and Matti Latva-aho
Centre for Wireless Communications,
University of Oulu,
Finland
E-mail: nuwanthika.rajapaksha@oulu.fi, nandana.rajatheva@oulu.fi, matti.latva-aho@oulu.fi

***Abstract*—End-to-end learning of a communications system using the deep learning-based autoencoder concept has drawn interest in recent research due to its simplicity, flexibility and its potential of adapting to complex channel models and practical system imperfections. In this paper, we have compared the bit error rate (BER) performance of autoencoder based systems and conventional channel coded systems with convolutional coding (CC), in order to understand the potential of deep learning-based systems as alternatives to conventional systems. From the simulations, autoencoder implementation was observed to have a better BER in 0-5 dB E_b/N_0 range than its equivalent half-rate convolutional coded BPSK with hard decision decoding, and to have only less than 1 dB gap at a BER of 10^{-5}. Furthermore, we have also proposed a novel low complexity autoencoder architecture to implement end-to-end learning of coded systems in which we have shown better BER performance than the baseline implementation. The newly proposed low complexity autoencoder was capable of achieving a better BER performance than half-rate 16-QAM with hard decision decoding over the full 0-10 dB E_b/N_0 range and a better BER performance than the soft decision decoding in 0-4 dB E_b/N_0 range.**

***Index Terms*—autoencoder, end-to-end learning, deep learning, neural networks, wireless communications, modulation, channel coding**

I. Introduction

Wireless networks and related services have become essential and basic building blocks in the modern society which have changed the life styles to a great extent. Emergence of many new services and applications is challenging the traditional communication landscapes in terms of reliability, latency, energy efficiency, flexibility and connection density, requiring new architectures, algorithms and novel approaches in each and every layer of a communications system.

Communications is a well established field with expert knowledge based on information theory, statistics and mathematical modelling. Especially for the physical layer, well proven mathematical approaches for modelling channels [1], determining optimal signaling and detection schemes for reliable data transmission compensating for hardware imperfections [2] etc. are there. However, conventional communications theories display several inherent limitations in achieving the large data and ultra-high-rate communication requirements in complex scenarios. Fast and effective signal processing in latency critical applications, modelling the channels in complex environments, and realizing optimum performance in sub-optimal fixed block structured systems are some of the challenges faced by modern communications systems. Applying deep learning concepts to the physical layer has attracted a wider interest in recent history, due to certain advantages of deep learning which could be useful in overcoming above challenges.

Reliable message transmission from a source to destination over a channel with the aid of a transmitter and receiver is the basic requirement of a communications system. In practice, the transmitter and receiver are divided into a series of multiple independent blocks in order to achieve an optimal solution for the communications task. Even though such a block structure allows individual analysis, controlling and optimization of each block, it is not clear that these individually optimized blocks attain the optimum end-to-end performance [3]. A deep learning-based end-to-end communications system on the other hand, follows the original definition of a communications system and tries to optimize transmitter and receiver in an end-to-end manner without having an artificial block structure [3], [4]. Such a straightforward structure which has less energy consumption, lower computational complexity and processing delays seems attracting to be applied in practical systems, especially if such deep learning-based systems may outperform existing conventional systems.

In this study, we have investigated the performance of autoencoder based end-to-end communications systems in order to understand their potential as an alternative to conventional systems [5]. Our main contributions are as follows:

- We have proposed a new autoencoder architecture for end-to-end learning of a communications system in additive white Gaussian noise (AWGN) channel, which has a lower training and processing complexity compared to the autoencoder layout proposed in [3]. The newly proposed autoencoder layout has lower model dimensions compared to the original autoencoder layout proposed in [3] and hence has a smaller number of learnable parameters, thus having a significantly lower training complexity than the originally proposed autoencoder layout. Lower model dimensions of the proposed autoencoder has also resulted in having lower processing complexity, hence low latency processing compared to the original one.

This work was supported by the Academy of Finland 6Genesis Flagship (grant no. 318927).

978-1-7281-5207-3/20/$31.00 ©2020 IEEE

- Autoencoder performance analysis with respect to advanced channel coded systems and higher order modulation schemes has not been covered so far in existing related works. We have evaluated the BER performance of the proposed autoencoder in comparison with conventional channel coded systems which utilize convolutional coding as forward error control mechanism and which operate in higher modulation orders. From the simulations, we have demonstrated that autoencoder has close performance to the conventional systems.
- Furthermore, we have presented a processing complexity analysis of the autoencoder based system and the conventional communications system, which shows that the autoencoder based system can be implemented with a lower processing complexity and a lower processing delay compared to conventional systems.
- We have also demonstrated the BER performance comparison of the original autoencoder architecture [3] with conventional convolutional coded systems with BPSK modulation. Simulation results for different code rates and different autoencoder configuration settings are presented.

The structure of the paper is as follows. The following sub section discusses some of the related work. Section II explains the autoencoder concept and autoencoder based systems as alternatives for coded systems with BPSK modulation and higher order modulations. Section III presents the obtained results for the implemented autoencoder based systems. Section IV concludes the paper with a summary of findings of the study.

A. Related Work

Using the autoencoder concept for end-to-end learning of communications systems was first proposed in [3], [4]. A novel way of communications system design based on autoencoder is presented in [3], where the communication task is considered as an end-to-end reconstruction task which jointly optimizes the transmitter and receiver in a single process. The concept of radio transformer networks (RTNs) is presented as a method to incorporate expert domain knowledge in the machine learning model. Extension of the autoencoder model to multiple transmitter and receiver pairs is also presented. In [6], authors have introduced a novel physical layer scheme for the multiple input multiple output (MIMO) communications, extending the autoencoder based end-to-end learning approach to multi-antenna case. Autoencoder based communications for the single input single output (SISO) and MIMO interference channel in flat-fading conditions is introduced in [7].

In [8], the channel autoencoder model is improved enabling end-to-end learning in instances where the channel response is unknown or difficult to model in a closed form analytical solution. By adopting an adversarial approach for channel response approximation and information encoding, jointly optimum solution is learned for both tasks over a wide range of channel environments. They have presented the results of the proposed system with training and validation done for an over-the-air system. In [9], an autoencoder based over-the-air transmission system is presented where they have built, trained and executed a communications system fully composed of neural networks (NNs) using unsynchronized off-the-shelf software-defined radios. They have also introduced mechanisms for continuous data transmission and receiver synchronization, proving the feasibility of over-the-air implementation of a fully deep learning-based communications system.

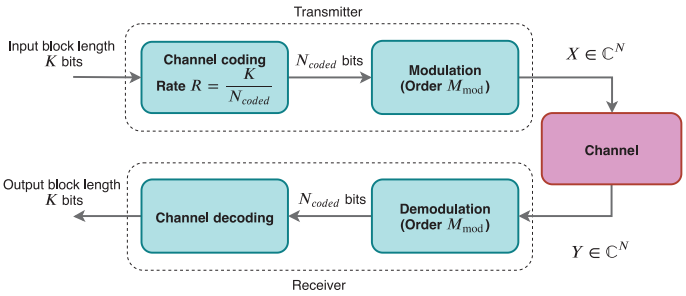

Fig. 1. System model of a conventional communications system.

More recently, the concept of end-to-end learning is also being applied in optical communications and molecular communications domains, where autoencoder based frameworks are proposed and implemented with comparable performance, showing the potential of deep learning-based end-to-end communications in complex channel conditions and operating environments [10], [11].

However, according to our best knowledge, autoencoder performance comparison with respect to advanced channel coded systems and higher order modulation schemes has not been covered in the existing literature so far. In order to address this gap, in this paper, we have investigated the performance of the autoencoder based end-to-end learning, comparing it against conventional communications systems which use CC. Also, the novel autoencoder layout which we have proposed in the paper is shown to have better performance compared to convolutional coded systems with higher order modulations.

II. Autoencoder based End-to-End Learning of Coded Systems

A. System Model

As shown in Fig. 1, a standard communications system consists of several blocks. The channel encoding block with code rate R converts the information block of size K bits to a block of size N_{coded} bits. Modulator of order M_{mod} then converts each $k_{mod} = log_2(M_{mod})$ bits of the coded block into symbols, forming a set of N transmit symbols according to the modulation scheme. At demodulation/detection and channel decoding blocks at the receiver, the reverse process is performed in order to recover the transmitted information block.

When a communications system is interpreted as an autoencoder based system, autoencoder tries to optimize the overall system in an end-to-end manner, instead of having an

explicit block structure as in a conventional communications system. Autoencoder models are implemented equivalent to conventional communications systems based on several system parameters such as the input message size, number of channel uses per message and signal power constraints, enabling performance comparison of the two approaches.

Formulation of the autoencoder model for the end-to-end communications system is as follows. The autoencoder processes information of k bits at a time, which we will define as a sub-block, where k bits ($k = log_2(M)$) long binary vector is considered as the message s out of M possible messages $s \in \mathbb{M} = \{1, 2, ..., M\}$ to be transmitted. Therefore, the K bit information blocks used in conventional system are to be divided into k bits long sub-blocks and fed iteratively into the autoencoder. Let n_{coded} is the number of coded bits per each message s. For a code rate of R, $n_{coded} = k/R$. Then, $n = n_{coded}/k_{mod}$ symbols are required to transmit the encoded n_{coded} bits where $k_{mod} = log_2(M_{mod})$ and M_{mod} is the given modulation order. Thus, the autoencoder transmits k bits long message s using the signal $\mathbf{x} \in \mathbb{C}^n$. The received signal $\mathbf{y} \in \mathbb{C}^n$ is

$$\mathbf{y} = \mathbf{x} + \mathbf{w}, \tag{1}$$

where $\mathbf{w} \sim \mathcal{CN}(\mathbf{0}, \beta\mathbf{I}_n)$ is Gaussian noise. At the receiver side, autoencoder outputs a k bits long sub-block which is the estimated message $\hat{s}$. The full K bit information block is obtained by concatenating the estimated messages $\hat{s}$.

The next section gives an overview of the autoencoder implementation using a feedforward NN. In Section II-C and II-D, we have discussed the implementation details of two different autoencoder architectures where the models are designed using the above discussed system parameters. Section II-C follows the original autoencoder architecture proposed by [3], and in the Section II-D we have presented the newly proposed low complexity autoencoder.

B. Autoencoder Basics

An autoencoder is a type of artificial neural network that tries to reconstruct its input at the output in an unsupervised manner [12]. It has a hidden layer $\mathbf{h}$ that describes a code used to represent the input $\mathbf{x}$. The network can be viewed as consisting of two parts: an encoder function $\mathbf{h} = f(\mathbf{x})$ and a decoder that produces a reconstruction $\hat{\mathbf{x}} = g(\mathbf{h})$. The learning process of the autoencoder can be described as minimizing the reconstruction loss:

$$\mathbf{L}(\mathbf{x}, \hat{\mathbf{x}}) = \mathbf{L}(\mathbf{x}, g(f(\mathbf{x}))), \tag{2}$$

where $\mathbf{L}$ is a loss function such as cross-entropy or mean square error (MSE) which penalizes $g(f(\mathbf{x}))$ for being different to the input $\mathbf{x}$.

Autoencoder can be implemented as a feedforward NN with L layers which describes a mapping $f(\mathbf{x}_0; \boldsymbol{\theta}) : \mathbb{R}^{N_0} \mapsto \mathbb{R}^{N_L}$ of an input vector $\mathbf{x}_0 \in \mathbb{R}^{N_0}$ to an output vector $\mathbf{x}_L \in \mathbb{R}^{N_L}$, through L iterative processing steps

$$\mathbf{x}_l = f_l(\mathbf{x}_{l-1}; \theta_l), \qquad l = 1, ..., L, \tag{3}$$

where $f_l(\mathbf{x}_{l-1}; \theta_l) : \mathbb{R}^{N_{l-1}} \mapsto \mathbb{R}^{N_l}$ is the mapping performed by the l^{th} layer which depends on the output vector $\mathbf{x}_{l-1}$ from the previous layer and the set of learnable parameters θ_l. $\boldsymbol{\theta} = \{\theta_1, \theta_2, .., \theta_L\}$ denotes the set of all parameters of the network which are learnt through model training. In the autoencoder models that we have implemented, we have mainly used *Dense* layers in which $f_l(\mathbf{x}_{l-1}; \theta_l)$ has the form

$$f_l(\mathbf{x}_{l-1}; \theta_l) = \sigma(\mathbf{W}_l\mathbf{x}_{l-1} + \mathbf{b}_l), \tag{4}$$

where $\mathbf{W}_l \in \mathbb{R}^{N_l \times N_{l-1}}$, $\mathbf{b}_l \in \mathbb{R}^{N_l}$ and set the of parameters is $\theta_l = \{\mathbf{W}_l, \mathbf{b}_l\}$. $\sigma(\cdot)$ is a called an *activation function* such as *ReLU*, *Linear*, *Softmax* etc. which introduces non-linearity to the network. Apart from the *Dense* layers, the *Normalization* layer defined in (5) is used at the output of the transmitter side of the network to guarantee the transmit energy constraints. Furthermore, the *Noise* layer defined in (6) is used in-between the transmitter and receiver side of the network to denote the AWGN channel for model training.

$$f_l(\mathbf{x}_{l-1}; \theta_l) = \frac{\sqrt{E}\,\mathbf{x}_{l-1}}{\|\mathbf{x}_{l-1}\|_2} \tag{5}$$

$$f_l(\mathbf{x}_{l-1}; \theta_l) = \mathbf{x}_{l-1} + \mathbf{w}, \quad \mathbf{w} \sim \mathcal{N}(0, \beta\mathbf{I}_{N_{l-1}}) \tag{6}$$

In (5), E is the total transmit energy for the message s. In (6), a Gaussian noise vector with zero mean and covariance matrix $\beta\mathbf{I}_{N_{l-1}}$ is added to the input, generating a different output for the same input at each execution instance, which simulates the AWGN channel behaviour [3].

During the model training, a set of input and output vector pairs, $(\mathbf{x}_{0,i}, \mathbf{x}^{\star}_{L,i}), i = 1, ..., S$, is used to minimize the reconstruction loss

$$\mathbf{L}(\boldsymbol{\theta}) = \frac{1}{S}\sum_{i=1}^{S} l(\mathbf{x}^{\star}_{L,i}, \mathbf{x}_{L,i}), \tag{7}$$

by adjusting the parameter set $\boldsymbol{\theta}$, where $l(\mathbf{u}, \mathbf{v}) : \mathbb{R}^{N_L} \times \mathbb{R}^{N_L} \mapsto \mathbb{R}$ is the loss function. Here, $\mathbf{x}^{\star}_{L,i}$ is the expected output when $\mathbf{x}_{0,i}$ is given as input. For an autoencoder, $\mathbf{x}^{\star}_{L,i} = \mathbf{x}_{0,i}$ since the objective of an autoencoder is reconstruction of the inputs at its output. $\mathbf{x}_{L,i}$ is the actual output from the NN for input $\mathbf{x}_{0,i}$. Stochastic gradient descent (SGD) algorithm is one of the most widely applied algorithms to obtain optimum set of θ which uses a random *mini-batch* approach for gradient computation of an approximation of $\mathbf{L}(\boldsymbol{\theta})$ when updating parameters θ during training [3].

C. End-to-End Learning of Coded Systems with BPSK

Here we consider the original autoencoder proposed by by [3] with slight variations, in order to do a BER performance evaluation of autoencoder based systems in comparison to conventional channel coded communications system with

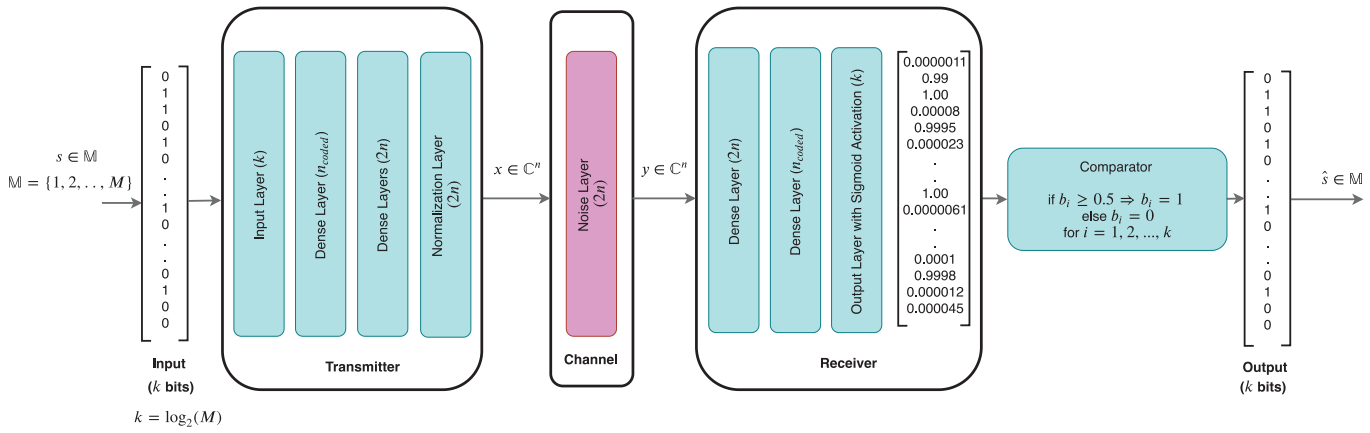

Fig. 2. New autoencoder implementation with lower training and processing complexity.

BPSK modulation over the AWGN channel. Given the task of communicating one out of M possible messages $s \in \mathbb{M} = \{1, 2, ..., M\}$ using n complex channel uses with a minimum reconstruction error, the autoencoder model is formulated as a feedforward NN constructed by sequentially combining the layers {*Input, Dense-ReLU, Dense-ReLU, Dense-Linear, Normalization, Noise, Dense-ReLU, Dense-ReLU, Dense-Softmax*} which have $\{M, M, M, 2n, 2n, 2n, M, M, M\}$ output dimensions respectively. Layers 1-5 compose the transmitter side of the system where the energy constraint of the transmit signals is guaranteed by the *Normalization* layer at the end. Layers 7-9 compose the receiver side of the system where estimated message can be obtained from the output of the *Softmax* layer. At the output of the transmitter side and the input of the receiver side we have real-valued vectors of size $2n$ which correspond to the n complex channel uses. The model is trained end-to-end using SGD using the categorical cross-entropy loss function on the set of all possible messages $s \in \mathbb{M}$.

However, as also pointed out in [3], it is evident that representation of message s by an M-dimensional vector becomes impractical for large M values due to huge memory and processing requirements. Model dimensions significantly increase with M and large number of learnable parameters make it difficult to efficiently train the model. In order to overcome this challenge we have developed a new autoencoder layout which uses a binary vector to represent message s which has only $log_2(M)$ dimensions. Details of the implementation and its performance evaluation are given in the following sections.

D. Low Complexity End-to-End Learning of Coded Systems with Higher Order Modulations

This section presents the design and development of the new autoencoder based end-to-end communications system which has a low training and processing complexity compared to the previous autoencoder model. Instead of taking the autoencoder input corresponding to message s as an M-dimensional *one-hot* vector, here we directly give the k bits long sub-block as the input to the autoencoder which corresponds to each message s. At the output, the autoencoder is trained to output an estimated message $\hat{s}$ which is also a k bits long binary vector.

TABLE I
LAYOUT OF THE NEWLY PROPOSED LOW COMPLEXITY AUTOENCODER

Layer	Output dimensions
Input	k
Dense-ReLU	n_{coded}
Dense-ReLU	$2n$
Dense-ReLU	$2n$
Dense-Linear	$2n$
Normalization	$2n$
Noise	$2n$
Dense-ReLU	$2n$
Dense-Linear	n_{coded}
Dense-Sigmoid	k

In addition to having binary inputs to reduce the training complexity, we have also improved the original autoencoder layout proposed by [3], changing the layer dimensions of the model in order to absorb different parameter settings in a conventional communications system. Internal layers of the proposed autoencoder architecture are designed with parameters relating to the channel encoder, modulator functions at the transmitter side and demodulator, channel decoder functions at the receiver side. Thus, the design parameters k, R, M_{mod} are used to determine the dimensions of each layer of the autoencoder model as shown in Table I. Fig. 2 illustrates the layout of the model. Here also, at the output of the transmitter side and the input of the receiver side we have real-valued vectors of size $2n$ since we assume n complex channel uses.

A *Sigmoid* layer is used as the output layer along with binary-cross entropy loss function since we deal with binary inputs and outputs. After the model training, autoencoder output is applied to a comparator module which returns the binary outputs as shown in Fig. 2. More details about selecting the model layout, layer types and activation functions are available in [5].

III. Simulations and Results

A. Coded Systems with BPSK Modulation

Autoencoder BER performances were evaluated and compared with their equivalent conventional channel coded BPSK systems using the autoencoder model described in Section II-C. For BPSK, $M_{mod} = 2$ and hence $n_{coded} = n$. Thus, different rates $R = log_2(M)/n$ were obtained by changing the message size $M = \{2, 4, 16, 256\}$ and the number of n channel uses, resulting in autoencoder models equivalent to baseline systems with BPSK modulation and code rates $R = \{1/2, 1/3, 1/4\}$. Noise variance of the AWGN channel is $\beta = (2RE_b/N_0)^{-1}$ where E_b/N_0 is the energy per bit (E_b) to noise power spectral density (N_0) ratio.

Model implementation, training and testing was done in Keras [13] with TensorFlow [14] backend. End-to-end model training over the stochastic channel model was done using SGD with Adam optimizer with a learning rate of 0.001. Equal message transmit energy constraints were kept in each autoencoder model and its corresponding baseline system during simulations for fair comparison. For each model, 1,000,000 random messages were used as the training set and model training was done over 50 epochs with batch size of 2000. E_b/N_0 = 5 dB was used for model training. Model testing was performed with 1,000,000 different random messages over 0 dB to 10 dB E_b/N_0 range. BER performances of the autoencoder models and corresponding baselines were compared for each configuration setting.

Convolutional codes with Viterbi decoder with hard and soft decision decoding was used in the channel encoder/decoder blocks in the baseline system. Information block length for baseline system is taken as $K = 800$ and CC with constraint length 7 is used.

BER plots in Fig. 3 and Fig. 4, show that the autoencoder BER performance improves with increasing message size. In each figure, BER curve of $M = 2$ model overlaps with that of uncoded BPSK where as $M = 256$ model has much improved BER curve. Having more degrees of freedom and flexibility in the model due to the increased message size which helps a better end-to-end optimization is the reason for this improvement. M and R values determine the dimensions of the autoencoder model. Not much coding gain is achieved for low M values since the layer dimensions limit the non-linearities introduced by the model during the learning process. It is well explained from having 279,840 learnable parameters in the $M = 256$ model and only 46 learnable parameters in the $M = 2$ model. Layer dimensions increase with M, and for same code rate R model has higher degrees of freedom with more learnable parameters that can be optimized to minimize the end-to-end reconstruction error, enabling it to learn transmit symbols with a coding gain.

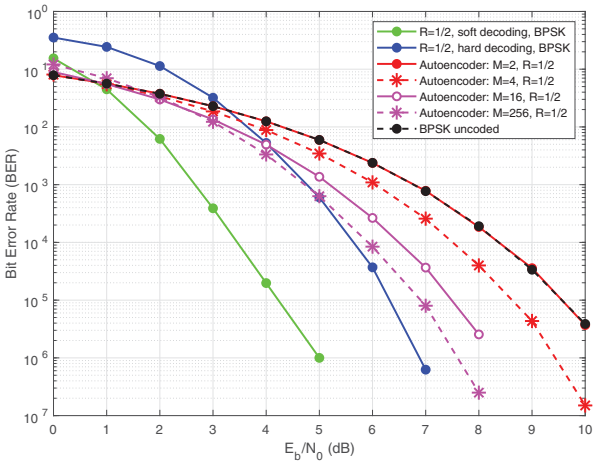

Fig. 3. $R = 1/2$ system BER performance comparison of different autoencoder models with $M = \{2, 4, 16, 256\}$.

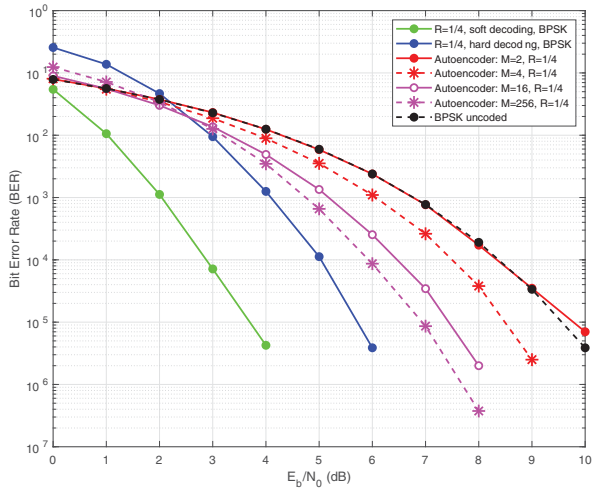

Fig. 4. $R = 1/4$ system BER performance comparison of different autoencoder models with $M = \{2, 4, 16, 256\}$.

From the BER plots, we can observe that autoencoder has a close BER performance to the hard decision CC. In $R = 1/2$ system, $M = 256$ autoencoder outperforms hard decision CC in 0 dB to 5 dB E_b/N_0 range and it is only around 1 dB worse at 10^{-5} BER. This BER gap may be further reduced by carefully adjusting the model training parameters such as the batch size, epochs, learning rate and the training E_b/N_0 value. It should be noted that in each simulation setting, a single model trained at E_b/N_0 = 5 dB gives this BER performance over the full the E_b/N_0 range.

B. Coded Systems with Higher Order Modulations

The AWGN channel BER performance of the new autoencoder proposed in Section II-D was evaluated in comparison

TABLE II
SYSTEM PARAMETERS FOR AUTOENCODER AND BASELINE SYSTEMS

Autoencoder configurations				Baseline system parameters	
M	$k = log_2(M)$	n_{coded}	n	R	M_{mod}
16	4	8	4	1/2	4
64	6	12	6		(QPSK)
256	8	16	8		
256	8	16	4	1/2	16
4096	12	24	6		(16-QAM)

with conventional channel coded systems with higher order modulations. Different system parameters used for autoencoder implementation are summarized in Table II. Noise variance of the AWGN channel is $\beta = (2Rk_{mod}E_b/N_0)^{-1}$. All the other training and testing parameters are same as in Section III-A other than the batch size being 1000 and the number of epochs being 100.

Fig. 5 shows the BER comparison between the autoencoder and baseline for $R = 1/2$ and $M_{mod} = 16$ configuration. Autoencoder outperforms hard decision CC across the full E_b/N_0 range considered. It is interesting to see that the autoencoder is better than both soft decision CC and uncoded 16-QAM in low E_b/N_0 range from 0 dB to 4 dB, where soft decision CC has an inferior BER performance than the uncoded 16-QAM. Here also, a single autoencoder model trained at E_b/N_0 = 5 dB gives this BER performance over the full the E_b/N_0 range. Thus, training the model at a given E_b/N_0 has been capable of learning transmit mechanisms across the full E_b/N_0 range showing the learning capability and adaptability of the deep learning model.

Fig. 6 shows the block error rate (BLER) comparison between the autoencoder and baseline. Autoencoder has an inferior BLER than the baseline which can be explained as the autoencoder is optimized for a minimum message error rate (MER) or BER of each transmit message instead of a minimum BLER. When considering the MER, autoencoder has an acceptable MER performance as shown in Fig. 7, having less than 10^{-4} error at 10 dB.

An interesting observation is that the autoencoder achieves acceptable BER and MER with very short input message lengths ($k = 8$ in this case). In conventional channel coding schemes such as the CC that we have used, the block size K needs to be very large in order to achieve a considerable coding gain in the BER. In standard communications systems, the block sizes are in the range of 1000s of bits and in more advanced coding schemes such as polar codes, low-density parity-check (LDPC) codes etc., the full block of K bits is needed to perform the encoding and/or decoding operations which introduce encoder/decoder delays. Furthermore, they have higher processing complexities which generally require iterative processing. Considering the above, autoencoder based systems which can operate in very small input block sizes and which have a lower processing complexity and processing delay compared to conventional systems would be advantageous for low latency, low throughput communications where short message transmission is recommended.

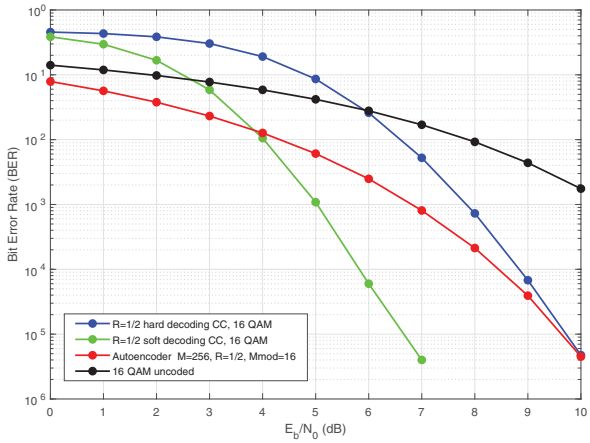

Fig. 5. Autoencoder and baseline BER for $R = 1/2$ and $M_{mod} = 16$.

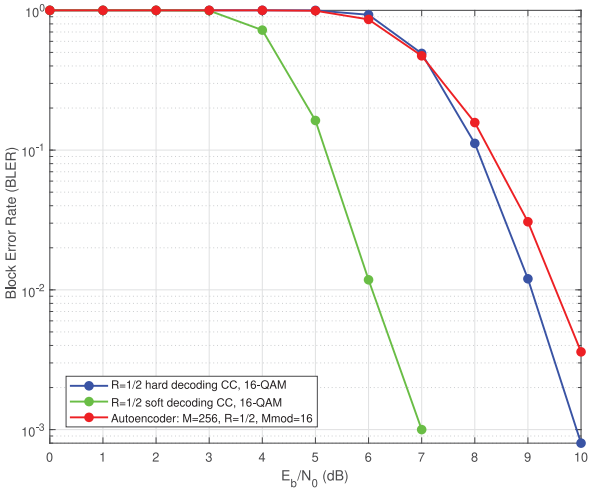

Fig. 6. Autoencoder and baseline BLER for $R = 1/2$ and $M_{mod} = 16$.

C. Training and Processing Complexity

First we compare the model training complexity of the two autoencoder architectures to understand the efficiency of the proposed autoencoder layout. For a model with message size $M = 256$, code rate $R = 1/2$ and modulation order $M_{mod} = 16$, original autoencoder has 267,528 learnable parameters and input dimensionality of the model is 256. Where as in the new autoencoder, only 776 learnable parameters are there and input vector dimension is only 8 bits. During our simulations, both these models were observed to have a similar BER performance. Therefore we can state that the latter model is more efficient to implement, especially for systems with higher modulation orders where larger M values are required.

Next we compare the computational complexity of the autoencoder proposed in Section II-D and the conventional communications system. Here we only consider the processing complexity of the Viterbi decoder when considering the

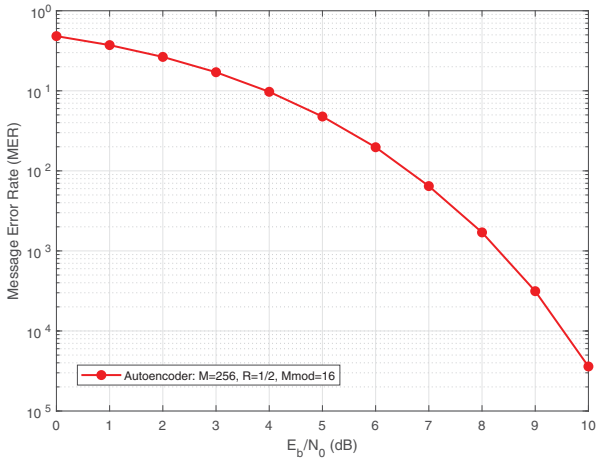

Fig. 7. Autoencoder MER for $R = 1/2$ and $M_{mod} = 16$.

conventional system as it has a higher processing complexity compared to other blocks in the system. Viterbi algorithm with a memory order m has a computational complexity of $4.R.N.2^m$ [15]. Computational complexity in each layer of the autoencoder is given in Table III. These numbers are further reduced with a parallel processing implementation which is common in neural networks [5]. Calculations show that the autoencoder and Viterbi decoder has same range of processing complexity without considering parallel implementation of the autoencoder. Therefore, considering parallel implementation as well, autoencoder has a very low computational complexity compared to the conventional system, given that autoencoder consists of end-to-end processing including both transmitter and receiver sides.

IV. Conclusion

In this study, we have extended the research done in [3] in autoencoder based end-to-end learning of the physical layer doing a further investigation in order to understand the capabilities of autoencoder based communications systems. Coded BPSK with hard decision CC and equivalent autoencoder implementations have a less than 1 dB gap in BER across the 0-10 dB E_b/N_0 range. Autoencoder is observed to have a closer performance to the baseline for higher code rates. Newly proposed light weight autoencoder is shown competent of learning better communication mechanisms compared to the conventional systems. Simulations show that the autoencoder equivalent of half-rate 16-QAM system has a better BER performance than hard decision CC over the full 0-10 dB E_b/N_0 range and soft decision CC in 0-4 dB E_b/N_0.

Autoencoder based end-to-end communications systems, having acceptable BER performance, flexible structure and higher learning capacity, low latency and low processing complexity, show their potential and feasibility as an alternative to conventional communications systems. We have analysed the AWGN channel performance in this study and it should also be extended for other fading channels as well. Also, we have assumed an ideal communications system with perfect timing and carrier-phase and frequency synchronization. Further research needs to be done in order to analyse the performance in non-ideal scenarios. Also, it is essential to investigate the autoencoder performance in comparison to 5G channel codes such as LDPC and polar codes.

TABLE III
Number of Mathematical Operations in Autoencoder Model

Layer (output dimensions)	Multiplications	Additions	Transfer function
Input (k)	-	-	-
Dense-ReLU (n_{coded})	$k.n_{coded}$	$n_{coded}(k+1)$	n_{coded}
Dense-ReLU ($2n$)	$n_{coded}.2n$	$2n(n_{coded}+1)$	$2n$
Dense-ReLU ($2n$)	$2n.2n$	$2n(2n+1)$	$2n$
Dense-Linear ($2n$)	$2n.2n$	$2n(2n+1)$	$2n$
Noise ($2n$)	-	-	-
Dense-ReLU ($2n$)	$2n.2n$	$2n(2n+1)$	$2n$
Dense-Linear (n_{coded})	$2n.n_{coded}$	$n_{coded}(2n+1)$	n_{coded}
Dense-Sigmoid (k)	$n_{coded}.k$	$k(n_{coded}+1)$	k

References

[1] T. S. Rappaport, *Wireless Communications: Principles and Practice*, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2002.

[2] J. Proakis and M. Salehi, *Digital Communications*, 5th ed. Boston, MA, USA: McGraw-Hill Educ., 2007.

[3] 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.

[4] T. J. O'Shea, K. Karra, and T. C. Clancy, "Learning to communicate: Channel auto-encoders, domain specific regularizers, and attention," in *IEEE Int. Symp. Signal Process. Inf. Technol. (ISSPIT)*, Dec 2016, pp. 223–228.

[5] R. N. S. Rajapaksha, "Potential deep learning approaches for the physical layer," Master's thesis, University of Oulu, Finland, 2019. [Online]. Available: http://urn.fi/URN:NBN:fi:oulu-201908142760

[6] T. J. O'Shea, T. Erpek, and T. C. Clancy, "Physical layer deep learning of encodings for the mimo fading channel," in *2017 55th Annu. Allerton Conf. Commun. Control Comput. (Allerton)*, Oct 2017, pp. 76–80.

[7] T. Erpek, T. J. O'Shea, and T. C. Clancy, "Learning a physical layer scheme for the mimo interference channel," in *2018 IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–5.

[8] T. J. O'Shea, T. Roy, N. West, and B. C. Hilburn, "Physical layer communications system design over-the-air using adversarial networks," in *2018 26th Eur. Signal Process. Conf. (EUSIPCO)*, Sep. 2018, pp. 529–532.

[9] S. Dörner, S. Cammerer, J. Hoydis, and S. t. Brink, "Deep learning based communication over the air," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 132–143, Feb 2018.

[10] H. Lee, S. H. Lee, T. Q. S. Quek, and I. Lee, "Deep learning framework for wireless systems: Applications to optical wireless communications," *IEEE Commun. Mag.*, vol. 57, no. 3, pp. 35–41, March 2019.

[11] S. Mohamed, J. Dong, A. R. Junejo, and D. C. Zuo, "Model-based: End-to-end molecular communication system through deep reinforcement learning auto encoder," *IEEE Access*, vol. 7, pp. 70 279–70 286, 2019.

[12] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. The MIT Press, 2016.

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

[14] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[15] M. Sybis, K. Wesolowski, K. Jayasinghe, V. Venkatasubramanian, and V. Vukadinovic, "Channel coding for ultra-reliable low-latency communication in 5g systems," in *2016 IEEE 84th Vehicular Technology Conference (VTC-Fall)*, Sep. 2016, pp. 1–5.