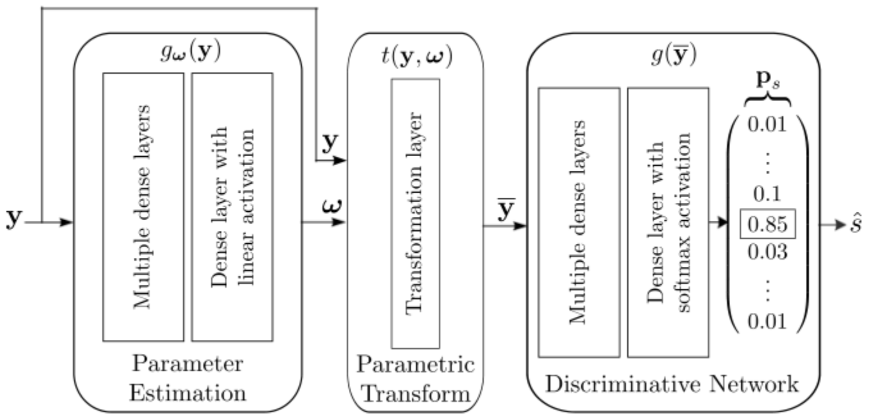
University of Stuttgart
Germany

Institute of Telecommunications
Professor Dr.-Ing. Stephan ten Brink

Seminar Thesis

Radio Transformer Networks

Sai Rahul Kaminwar



Date of hand out:	May 15, 2018
Date of hand in:	July 19, 2018
Supervisor:	Jakob Hoydis
	Stephan ten Brink
	Sebastian Dörner
	Sebastian Cammerer

Abstract

In this study thesis, a fundamental way to interpret communications system as an end-to-end trainable Neural Network (NN) task is defined in the form of Autoencoders. Later on, learned attention models known as Radio Transformer Network (RTN) are introduced into the network for easing the task of signal detection. These models are inspired from spatial transformer networks to perform appropriate radio domain transformations. A specific case of phase offset estimation and correction using such kind of attention models are discussed and the results in block error rate (BLER) vs signal-to-noise ratio (SNR) are compared against the quadrature phase-shift keying (QPSK) and autoencoders with and without expert knowledge for the same information rate of 2 Bit/Symbol. By the end, inference is made regarding the use of RTN and its prominence in reducing the complexity at the receiver.

Title page image: A radio receiver represented as a RTN. The input y runs through a parameter estimation network $g_\omega(y)$, and has a transform t(y, ω) applied to it to generate signal $\bar{y}$, and then is sent into the discriminative network $g(\bar{y})$ to produce the output ŝ.

Contents

Acronyms

QPSK	quadrature phase-shift keying
SNR	signal-to-noise ratio
BLER	block error rate
RTN	Radio Transformer Network
ML	Machine Learning
DL	Deep Learning
NN	Neural Network
MLP	Multi Layer Perceptron
AWGN	additive white Gaussian noise

Notations

x	scalar variables, especially in time domain: italic lower case letters
$a \cdot b$	scalar product of two scalars
$x(t), H(z)$	functions of continuous variables: argument is placed in round parentheses
$p(y\|x)$	conditional probability density function
α	phase rotation, complex number realised as two real valued numbers

1 Introduction

The basic building blocks of communication system consist of a transmitter, channel and a receiver. Although the process seems straightforward, many complications arise especially in the physical layer and there are diminishing results in terms of performance in this layer. The machine learning and deep learning approaches have shown significant improvements in performance in domains such as Computer Vision where it is impossible to arrive at a robust solution due to the inconsistency and unpredictability of inputs (as in case of handwritten digit recognition). This might also be true in some cases in the field of the communications where some complex scenarios are difficult to be described in mathematical models. So, the concepts of Machine Learning (ML) and Deep Learning (DL) can be applied to this domain as long as the communication models do not sufficiently capture the real effects.

Studies have shown that it is possible to learn full transmitter and receiver implementations for a given channel model, which are optimized for a chosen loss function (e.g., minimizing BLER). The transmitter, channel, and receiver are represented as one deep NN which can be trained as an Autoencoder.

The performance of such a system is comparable to the current state-of-the-art transmitter receiver systems which mean that the same performance can be obtained using the neural networks and they can be used to replace the highly complicated architectures of the present-day systems.

To further reduce the complexity of the training of such an autoencoder, expert knowledge can be integrated into the model using RTN. Before explaining the RTN's it is desirable to know how the concept of autoencoders has been applied to the communication systems. The following sections will deal with the potential of DL networks in the communication system followed by Autoencoders, Radio Transformer Networks which were used carry out predefined correction algorithms ("Transformers") at the receiver end and the approach used to implement such a network in the already present autoencoder setup.

1.1 Potential Of Deep Learning In Communication Systems

To further strengthen the argument of the usage of Artificial Neural Networks in communications ,let us consider the present day communication systems, where most signal processing algorithms have solid foundations in information theory and statistics, the models were designed mathematically ,but a practical system consists of many imperfections arising due to non linearities of Power amplifiers, or due to synchronization mismatch of the transmission and receivers. There might be cases where the characteristics of the channel are not known and since it is difficult to model these channels mathematically, the results of such a mathematical based system might not be optimal. Moreover, the communication blocks are optimized independently (for example source coding and channel coding are performed one after the other). The joint optimization of such components(eg. factor graphs[1]) may provide gains but lead to computationally complex systems.

Keeping in view of all these issues a Deep Learning based Communication system which does not require a mathematically tractable model, and which can be optimized for specific channel and hardware configuration can be used to minimize the imperfections caused in the former systems..

2 Autoencoders

2.1 Autoencoders In General

The autoencoders are type of NN whose main aim is to learn a representation of data for the purpose of dimensionality reduction. It learns to compress the data from the input layer into a short code which is then uncompressed into something that closely resembles the original input data. In short, the compression forces the autoencoder to engage in reducing the dimensionality by ignoring the noise. This can be clearly explained by the structure as in the figure 2.1

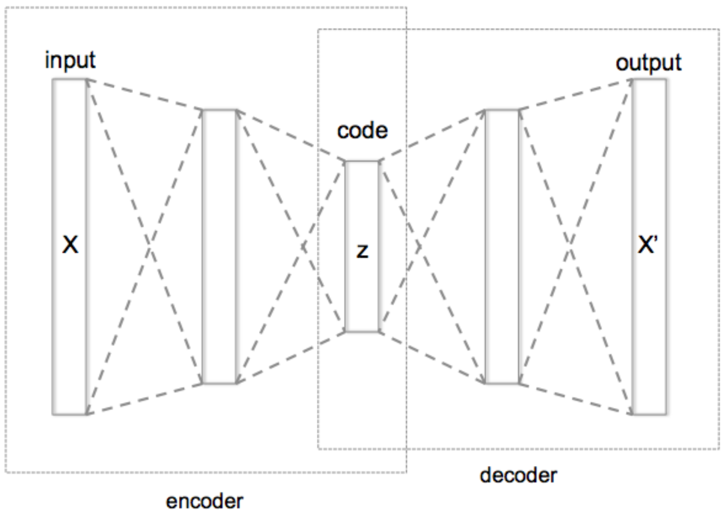


Figure 2.1: Structure of Autoencoder

The figure shows the architecture of the autoencoder in its simplest form which resembles a Multi Layer Perceptron (MLP) having an input and output layer along with multiple hidden layers. It is interesting to note that the number of nodes in the input and output layers is the same for the purpose of reconstructing its own inputs making them unsupervised learning models.

2.2 Autoencoders For Communication Systems

In simple terms a basic communication system consists of a transmitter which will communicate one out of M possible messages s using N discrete channel uses. Generally, energy, power and amplitude constraints are applied by the hardware of the transmitter on the outgoing signal x carrying the message. The channel is described by the conditional probability density function $p(y|x)$, where y denotes the received signal. Upon reception of the signal y, the receiver applies a reverse transformation to reconstruct the transmitted message s.

A similar process can be done in DL with the help of autoencoders. The general characteristics of autoencoders were discussed in the previous section, but in our case, the autoencoder serves a different purpose. It is used to learn the representations x of the message s which are robust to channel impairments such as fading, distortion and can be recovered at the receiver with the smallest probability of error. An example of such an autoencoder is shown in the figure 2.2

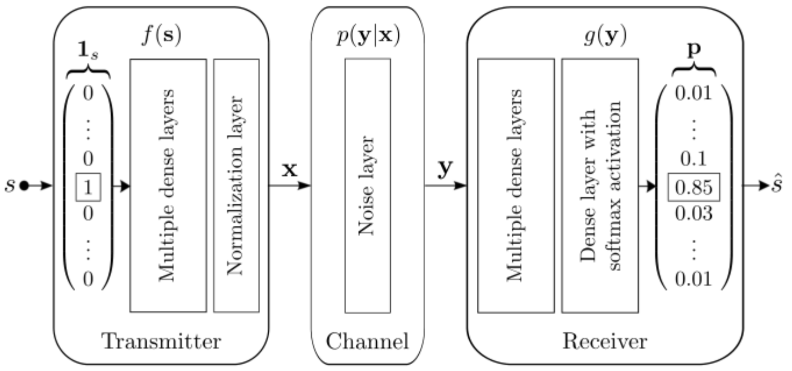


Figure 2.2: *Structure* of Autoencoder for a Communication System over an AWGN Channel

Here, the transmitter consists of a feedforward NN with multiple dense layers followed by a normalization layer to ensure that physical constraints on x are met. The input s to the transmitter is encoded as a one-hot vector I.e.; an M-dimensional vector, the s^{th}element of which is equal to one and zero otherwise. The channel here is represented by an additive noise layer with a fixed variance $\beta = (2RE_b/N_0)^{-1}$,where E_b/N_0 denotes the energy per bit (E_b) to noise power spectral density (N_0) ratio. The receiver is also implemented as a feedforward NN with the last layer using a softmax activation whose output $p \in (0,1)^M$ is a probability vector over all possible messages. The decoded message corresponds to the index of the element of p with the highest probability which is highlighted in the figure with a box

3 Radio Transformer Networks

In communication receivers, synchronization is performed on the received signals before further processing. As discussed in the introduction, this is essential because of the imperfections of the hardware systems and channel characteristics resulting in phase, and sample timing offsets. This synchronization can be thought as an attention patch which can estimate the offsets at the receiver and correct them before sending the signal to the later stages[2] . Unfortunately, the process of estimating these offsets is a highly specialized task and expert knowledge is required to perform the task, so it is desirable to build a system which can learn the generalization and adapt without needing such kind of expert knowledge for parameter estimation.

In Computer Vision Spatial Transformer Networks (STNs)[3] were introduced to solve a similar kind of problem where a differentiable end to end feed-forward attention model is introduced into existing convolution architectures .This network can be trained directly from the loss of training examples and can be used to evaluate on new sample giving the neural networks the ability to transform the feature maps spatially without any additional training supervision to the optimization process .The STN resulted in models which learn in-variance to translation, rotation and more generic warping which made them achieve state-of-the-art performance on several benchmarks.

Radio Transformer Networks are introduced in a similar fashion as a generalization to the STN architecture but introduces radio/domain specific parametric transformations. The DL models are augmented with expert propagation domain knowledge which is not signal specific through the use of this Radio Transformer Network. These networks can be used wherever parametric transformations seeded by estimated parameters are needed. An RTN for one specific case where it is placed at receiver is discussed here

An RTN [4]as shown in the figure 3.1 consists of three parts:

- A learned parameter estimation Layer: $g_\omega(y)$, consisting of multiple dense NN layers which compute a parameter vector from its input y.
- A parametric transform Layer: This applies a deterministic (and differentiable) function to y which is parameterized by ω and suited to the propagation phenomena.
- A learned discriminative network: This is the same receiver part as shown for autoencoder in the previous section which produces the estimate $\hat{s}$ of the transmitted message (or other label information) from the canonicalized input $\bar{y}$.

By allowing the parameter estimator g_ω to take the form of an NN, the system can be trained end-to-end to optimize for a given loss function. It is important to note that the training process of such an RTN does not seek to improve the parameter estimation itself but it optimizes the

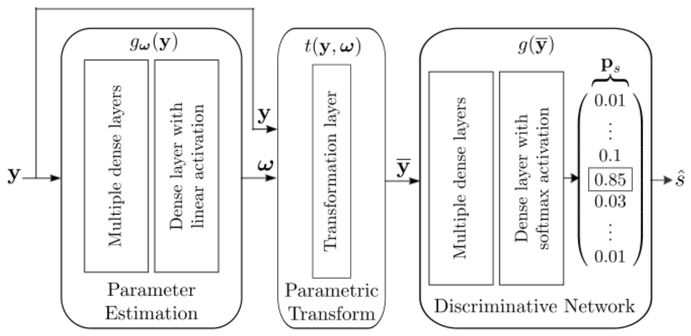


Figure 3.1: Structure of RTN

way they are estimated to obtain best end-to-end performance (e.g., BLER). A more detailed description of RTN can be found in [4].

3.1 Phase Offset Correction Using RTN

Let us consider the task of phase offset estimation and correction for signal detection using the RTN in the Autoencoder setup discussed earlier, Let $y_c = e^{j\varphi}\tilde{y_c} \in C^n$ be a vector of IQ samples that have undergone a phase rotation by the phase offset φ, the goal of g_ω is to estimate a scalar $\hat{\varphi} = \omega = g_\omega(y)$ that is close to the phase offset φ,which is further used to compensate the phase offset by the parametric transform t to compute $\bar{y_c} = e^{-j\hat{\varphi}}y_c$. This signal is now sent into the discriminative network for detection or any other application such as classification of signals. In this way, the RTN makes the signal less complicated simplifying the task of the discriminative network. This parametric estimation layer is implemented via a neural network whose architecture which will be discussed in the next section.

3.2 RTN In Autoencoders

The initial setup consisted of basic end-to-end learned autoencoder over an AWGN channel. The performance measured was BLER for different SNR. Each message/block is a combination of 4 complex channel uses(symbols) where each of them carry 2 Bit/Symbol of information. This constitutes a message alphabet of $2^{4.2} = 256$ different messages s which is sent to the autoencoder to encode and further decode at the receiver. The messages are compared with a block of 4 consecutive QPSK symbols which also has the same information rate i.e. 2 Bit/symbol. The performance is shown in the figure 3.2

So, in order to increase the performance, the phase offset is corrected by employing an RTN before the receiver. The original incoming signal is routed to the parameter estimation layer

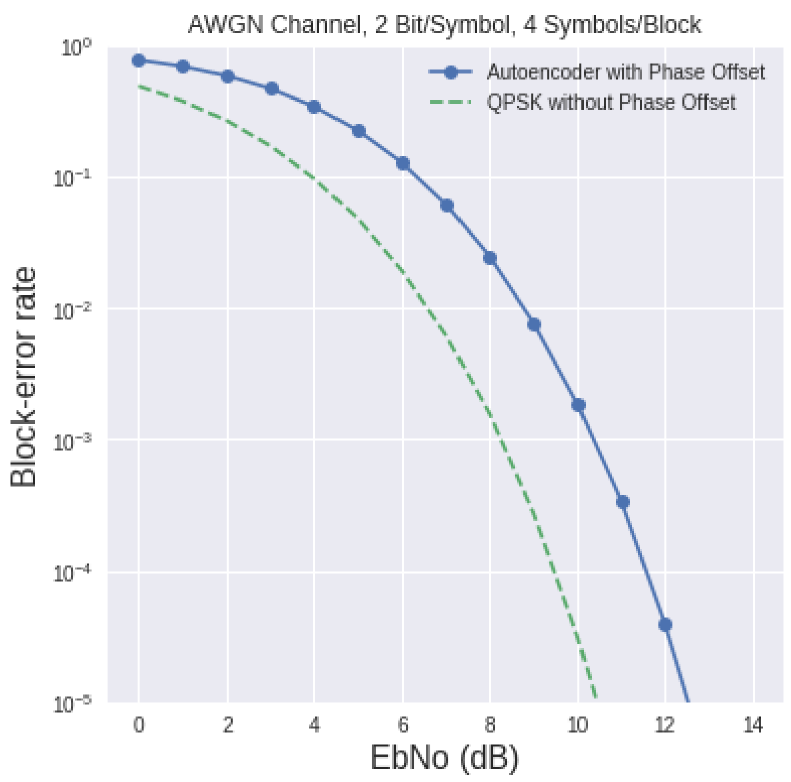


Figure 3.2: Performance of Autoencoder vs QPSK

which consists of a neural network performing a regression task and outputting two real numbers which can be realized as a complex number. This complex number can be thought as a phase offset which is then multiplied by the original incoming signal (parametric transformation). The obtained transformed signal is then passed to the discriminative network which is already present in the autoencoder setup.

Layers	Details
Input	y
Dense + ReLU	No.of neurons :64
Dense + ReLU	No.of neurons :64
Dense + Linear	No.of neurons :2

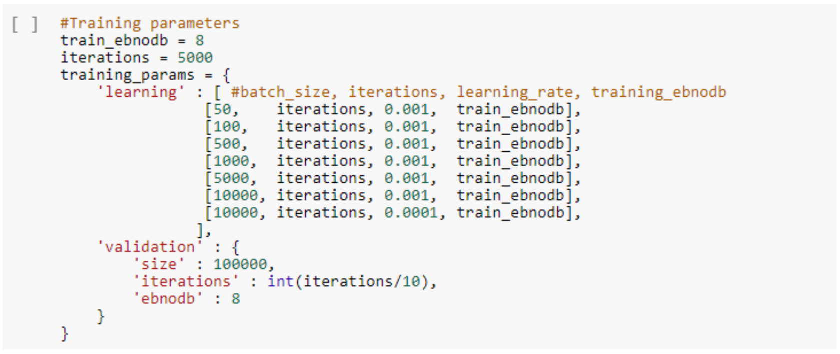
Table 3.1: RTN Architecture for Phase Offset Estimation

This can be clearly explained as follows: Let us consider the incoming signal to the receiver be y , which is routed and sent to the parameter estimation layer which outputs a parameter α (here: the phase rotation, which is a single complex number, realized as two real-valued numbers).This α is used for a specific parametric transformation say $y = \bar{}\alpha \cdot y$. Finally, the discriminative layer now sees $\bar{y}$ as input instead of y and performs the signal detection. The architecture of the parametric estimation layer is shown in table 3.1

4 Task Solving

4.1 Training And Validation

The setup is trained at SNR = 8dB, keeping all the other parameters same for both with and without RTN for a reasonable comparison. The training, validation parameters along with epochs and learning rate details are shown in figure 4.1



```
[ ]  #Training parameters
     train_ebnodb = 8
     iterations = 5000
     training_params = {
         'learning' : [ #batch_size, iterations, learning_rate, training_ebnodb
                       [50,    iterations, 0.001,  train_ebnodb],
                       [100,   iterations, 0.001,  train_ebnodb],
                       [500,   iterations, 0.001,  train_ebnodb],
                       [1000,  iterations, 0.001,  train_ebnodb],
                       [5000,  iterations, 0.001,  train_ebnodb],
                       [10000, iterations, 0.001,  train_ebnodb],
                       [10000, iterations, 0.0001, train_ebnodb],
                      ],
         'validation' : {
             'size' : 100000,
             'iterations' : int(iterations/10),
             'ebnodb' : 8
         }
     }
```

Figure 4.1: Training and Validation Parameters

4.2 Approach And Results

- Initially as seen in the basic Autoencoder system in figure 3.2 the performance is way beyond the performance of QPSK, and the goal is to bring the performance on par with QPSK by introducing RTN.
- The first approach taken was to implement the above parameter estimation layers with tanh activations as mentioned in [4].
- The tanh layers were replaced with ReLU and a slight change in performance was observed.
- Changing the number of neurons from 64 to 128 or 256 and adding a new layer did not improve the performance either.
- The initial attempt showed the performance in figure 4.2

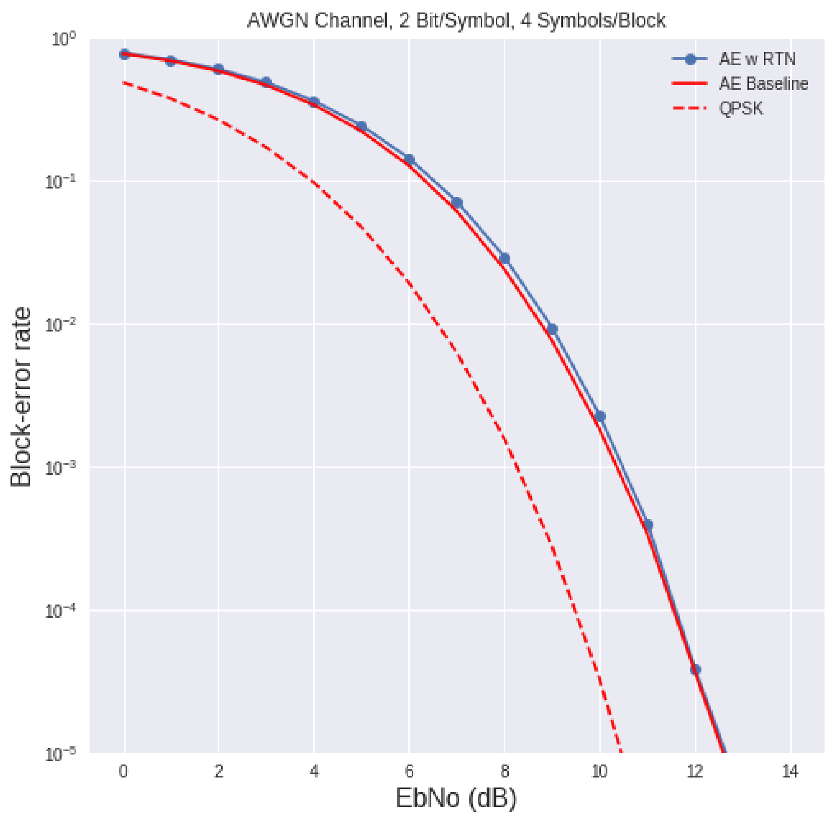


Figure 4.2: Autoencoder Initial Performance for a Simple Network

- As seen there is no improvement even after introducing the RTN layer which made me infer that by introducing RTN for solving simple tasks does not improve the performance.
- So after a discussion with Mr.Dörner, he has enhanced the system by providing the decoder part of the Auto-encoder a sequence of additional messages. By providing such additional sequence of messages, the problem of decoding for the normal NN became more complex and since the input to the receiver part had increased the RTN started to help and enhancement in performance was observed. The performance was compared for various lengths of additional sequences given to the receiver part.
- To further improve the performance the complex parameter obtained from the estimation layer is normalized to force it to be on a unit circle and is then multiplied by the original incoming signal. By doing this the RTN is forced to correct only the phase offset and not the gain.
- After multiple attempts trying with different code lengths, the performance of the RTN was almost the same as the performance of QPSK in terms of BLER for a particular codelength of 11. The performance is plotted with the QPSK, Autoencoder without and with RTN and is shown in figure 4.3

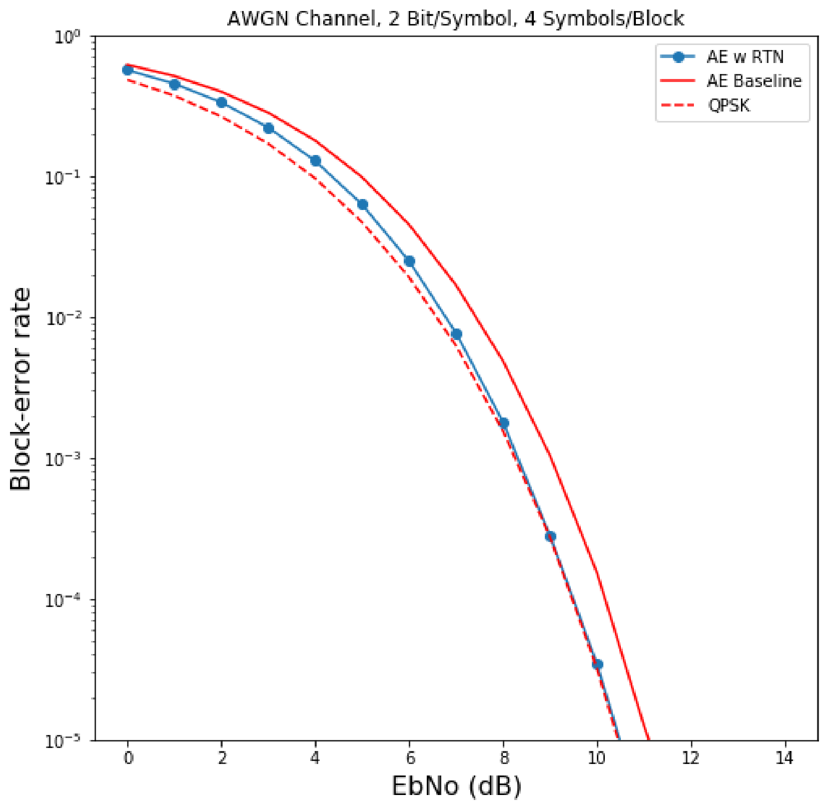


Figure 4.3: Autoencoder Performance with and without RTN compared to QPSK

4.3 Inference

From the above results, the following points can be inferred

- RTN can increase the performance of complex signal processing tasks and do not show any significant improvement for simple tasks.
- As observed from the training, the rate of convergence with RTN is faster than the autoencoder without RTN(can be verified from the ipynb file attached). This means the task of the receiver to decode is reduced as the offsets were corrected by the RTN itself before giving it to the receiver.
- RTN's can be used to integrate expert knowledge into the system to remove the random offsets introduced by the channel or by imperfections in the hardware and achieve comparable performance with state-of-the-art communication systems.

Bibliography

[1] H. Wymeersch, *Iterative receiver design.* Cambridge University Press, 2007, vol. 234. (p. 2)

[2] T. J. O'Shea, L. Pemula, D. Batra, and T. C. Clancy, "Radio transformer networks: Attention models for learning to synchronize in wireless systems," *CoRR*, vol. abs/1605.00716, 2016. [Online]. Available: http://arxiv.org/abs/1605.00716 (p. 5)

[3] M. Jaderberg, K. Simonyan, A. Zisserman, and k. kavukcuoglu, "Spatial transformer networks," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 2017–2025. [Online]. Available: http://papers.nips.cc/paper/5854-spatial-transformer-networks.pdf (p. 5)

[4] T. J. O'Shea and J. Hoydis, "An introduction to machine learning communications systems," *CoRR*, vol. abs/1702.00832, 2017. [Online]. Available: http://arxiv.org/abs/1702.00832 (p. 5), (p. 6), (p. 8)

Erklärung

Hiermit erkläre ich, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie alle wörtlich oder sinngemäß aus anderen Werken übernommenen Aussagen als solche gekennzeichnet habe.

Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht. Das elektronische Exemplar stimmt mit den gedruckten Exemplaren überein.

Stuttgart, July 19, 2018

Sai Rahul Kaminwar