Deep Networks for Equalization in Communications

by

Laura Brink

A thesis submitted in partial satisfaction of the
requirements for the degree of
Master's of Science

in

Electrical Engineering and Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Anant Sahai, Chair
Professor John Wawrzynek

Fall 2018

The thesis of Laura Brink, titled Deep Networks for Equalization in Communications, is approved:

Chair	____________________	Date	__________
	____________________	Date	__________
	____________________	Date	__________

University of California, Berkeley

Deep Networks for Equalization in Communications

Abstract

Deep Networks for Equalization in Communications

by

Laura Brink

Master's of Science in Electrical Engineering and Computer Science

University of California, Berkeley

Professor Anant Sahai, Chair

We apply the techniques from meta-learning to the communications domain. Specifically, we explore how equalization techniques can learn how to handle new environments without training on them.

Contents

List of Figures

Chapter 1

Introduction

Signal processing has long relied on well-defined, structured processes and protocols to function. However, in order to move to more robust and adaptive systems, we will need to overhaul these tightly structured processes. We must design new robust and adaptive communications systems.

From an academic perspective, the rise of machine learning tools and processing has allowed us to tackle problems we have not yet been able to, like in image processing. However, we still do not fully understand the reach or the limitations of this technology. In order to study the limitations of machine learning, we must apply them to spaces that we have studied extensively, like communications, and compare them to the well-known baselines.

Most communications systems have three main processes at the receiver; equalization, demodulation, and error-correction. While we will need to design robust forms of all of these processes, we will focus on equalization for the remainder of this paper.

1.1 Motivation

Meta-learning is the idea that algorithms need to be able to 'learn to learn' in order to generalize to different applications. For example, if a robot is trained to pick up coffee mugs, then we want that robot to be able to quickly learn how to pick up water bottles without having to re-train it. Meta-learning was inspired by humans ability to generalize how to learn [11]. Additionally, researchers in neurology have studied how the brain synapses change over time, suggesting that our algorithms will have to change over time to continue learning [1]. We refer the reader to [12] for a survey of meta-learning technologies and to [7] for recent developments in meta-learning algorithms.

In this paper, we want to understand if our neural network based communication processes can learn to learn. As in, can they handle new environments with new data sequeneces without having to re-train for the new variables. The communications application is arguably an easier application of meta-learning than others have been exploring, like image classification [**khodadeh**]. In order to better understand the reach and limitations of meta-learning, we need to apply it to something simpler, like communications. In this paper, we will explore whether or not we can learn to learn to communicate.

1.2 Background

Inter-symbol Interference and Equalization

Inter-symbol interference occurs when we are transmitting over a channel that has some echos. These echos cause the receiver to hear a garbled signal instead of the original signal from the transmitter. This is called inter-symbol interference because the receiver is hearing a combination of symbols across time.

Let $\vec{x} = [x_0, x_1, \ldots x_n]$ be the set of n complex symbols that the transmitter sends over the channel that connects the transmitter to the receiver. Each channel will have different characteristics. Some channels may have echos, others may have delays, often channels will have both. When a channel has echos, this is called a multipath channel because there are multiple paths to reach the receiver. Each path is called a tap. We can characterize a channel by characterizing the taps.

Let $\vec{a} = [a_0, a_1, \ldots a_\ell]$ be the set of characteristic for a multipath channel that has l taps. When a sequence of symbols like $\vec{x}$ is transmitted over this channel, the channel taps are convolved over the sequence. Additionally, there is noise in the system denoted by v_i.

$$\tilde{x}_m = \sum_{i=0}^{\ell} a_i x_{m-i} + v_i \tag{1.1}$$

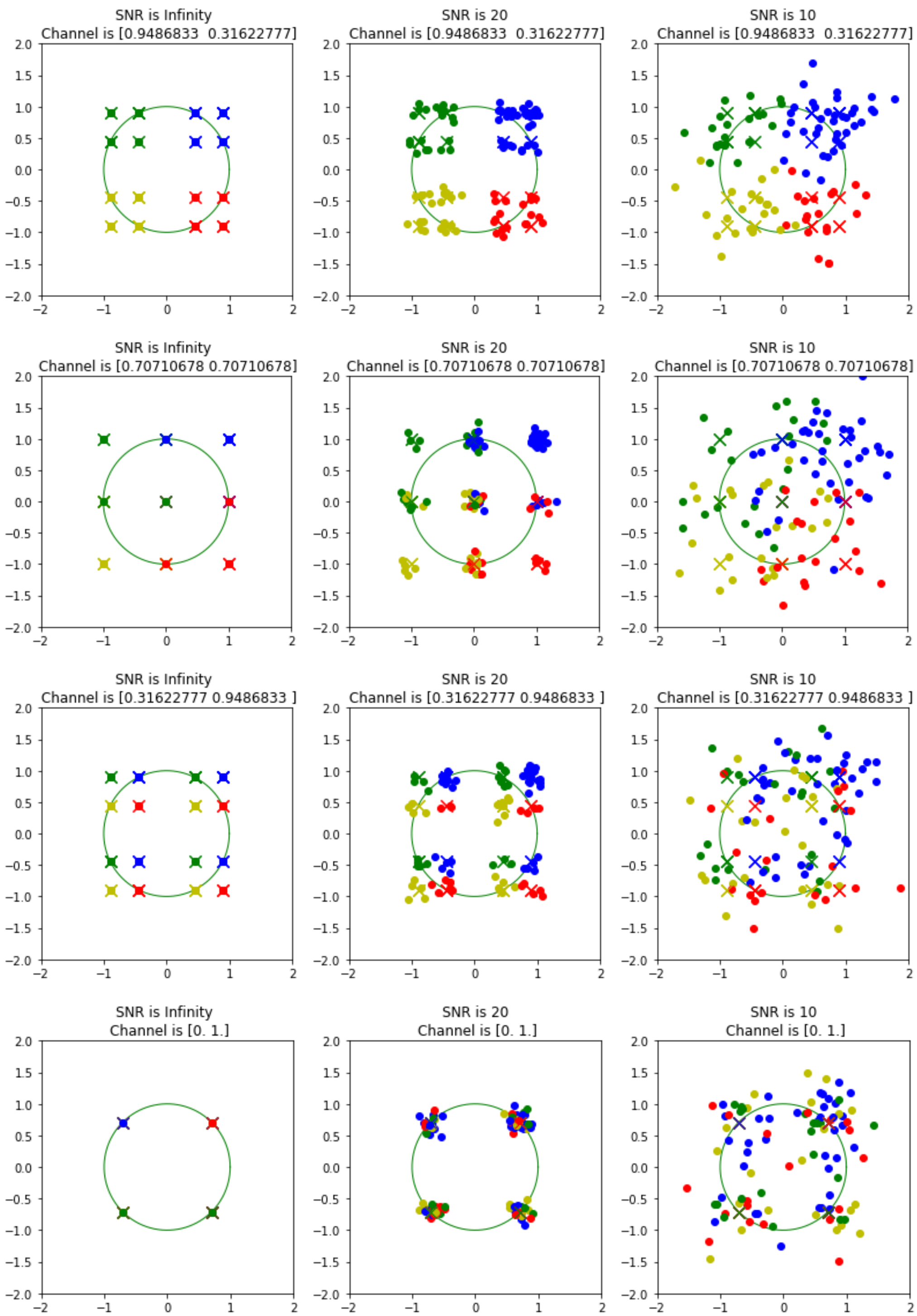
The receiver will hear a signal that is corrupted by inter-symbol interfence and noise; $\vec{\tilde{x}} = [\tilde{x}_0, \tilde{x}_1, \ldots \tilde{x}_{n+\ell}]$. Receivers must be able to handle garbled signals in order to transmit data in the real world. The process of removing the inter-symbol interference is called equalization. The goal of equalization is to take in a garbled signal and output a signal with minimal inter-symbol interference.

Figure 1.1 demonstrates the effects of multi-tap channels on a QPSK modulation constellation. We show what the received signal symbols constellations are from a sequence of 100 symbols modulated in QPSK through different two tap channels. We assume that the channel taps are constant during the transmission of the signal. We see that under certain channel conditions, like when the two taps are equal, it is very difficult to distinguish between the four constellations. Engineers have built processes to remove inter-symbol interference. First, let's go into the case when the channel characteristics are known.

Equalization for a known channel

If the receiver knows the channel characteristics, $\vec{a}$, perfectly, then there are a few different methods that can be used. The zero-forcing equalizer applies the inverse of the channel

Figure 1.1: The effects of a two tap channel on the QPSK constellation.



response to the received signal. It is called zero-forcing because there will be zero inter-symbol interference if there is no noise. There are some limitations of the zero-forcing equalizer. First, the impulse response of the equalizer needs to be infinitely long. Second, if there is a weak signal at a frequency, then the inverse gain is going to be very large. This will amplify any noise in the system. Third, if there are any zeros in frequency response, these cannot be inverted.

Another equalizer, the minimum mean squared error (MMSE) equalizer, handles noise much better than the zero-forcing equalizer and is explained in the next section. While it's important to consider how well a receiver can equalize with a known channel, this is rarely the case. Usually, we do not know the channel characteristics.

Equalization for an unknown channel

When the receiver does not know the channel characteristics, the process of equalization essentially has two jobs; first, identify the channel, second, remove the inter-symbol interference. If the receiver did not identify the channel first, there would be no way to remove the affects of it on the received signal.

In order to do channel estimation, most systems require that packets begin with a known sequence called a preamble. The signal sent will be broken into two parts; $\vec{x} = [\vec{x}_{pre}, \vec{x}_{data}]$. The signal received on the transmitter is $\vec{\tilde{x}} = [\vec{\tilde{x}}_{pre}, \vec{\tilde{x}}_{data}]$. The receiver knows what the orginal preamble sequence was, $\vec{x}_{pre}$, and can use the received preamble sequence, $\vec{\tilde{x}}_{pre}$, to estimate the behavior of the channel. Once the channel is estimated, the receiver then equalizes the data, $\vec{\tilde{x}}_{data}$.

One common method to estimate the channel is to use the least squares optimization framework. Let H be the estimate of the channel. The least squares channel estimator wants to find H that minimizes the squared error between the received signal, $\vec{\tilde{x}}_{pre}$, and what the predicted received signal would be if the the channel H was applied to the original preamble, $\vec{x}_{pre}$.

$$\min_{H} ||\vec{\tilde{x}}_{pre} - H\vec{x}_{pre}||^2 \tag{1.2}$$

The receiver needs to choose the length of H, representing how many taps (or echos) there might be in the channel. The length of H will be set based on the environment that the transmitter and receiver are communicating in.

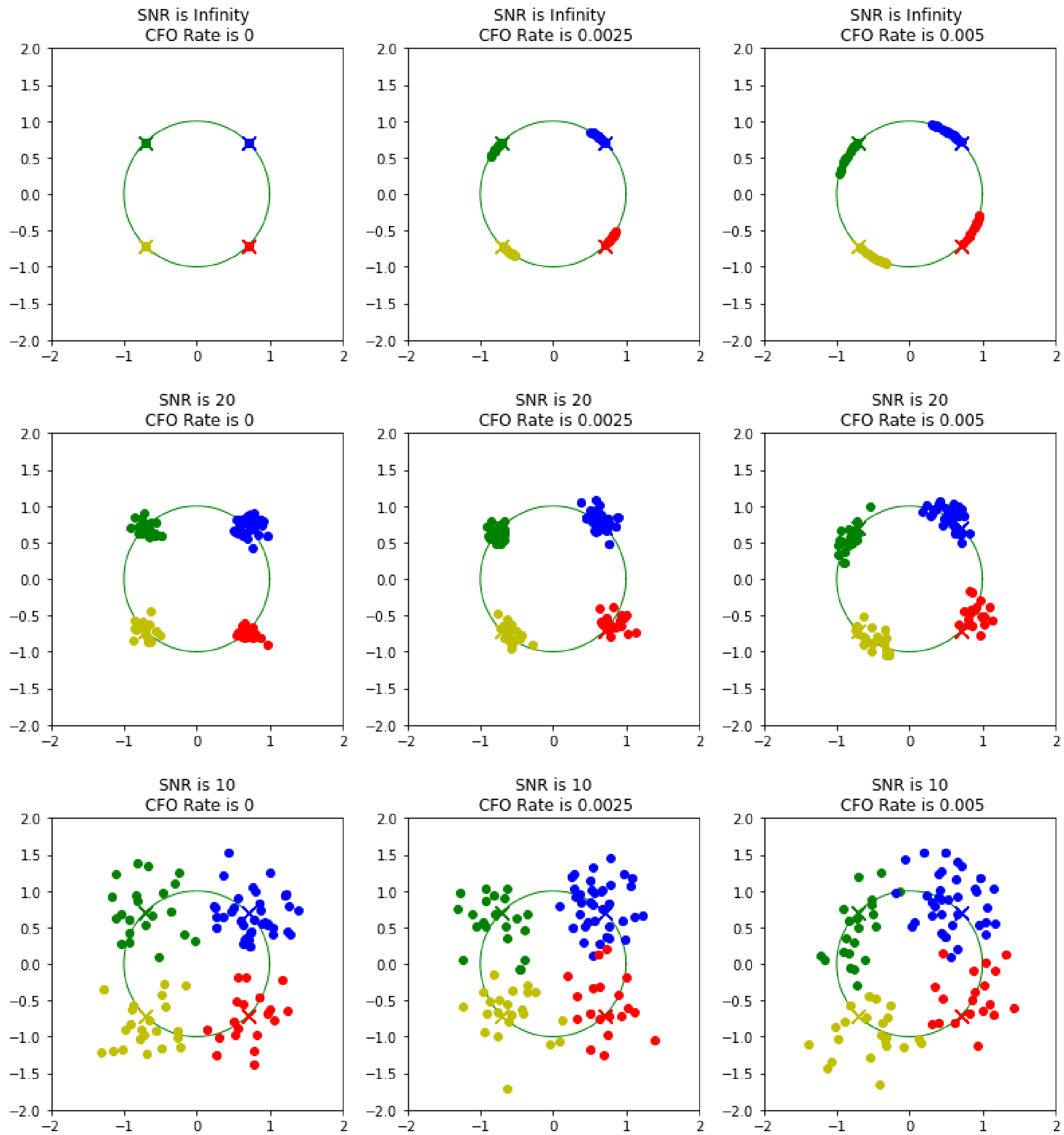
Once the channel response has been estimated, an equalizer has to use that information to remove the inter-symbol interference from the received signal. The MMSE equalizer minimizes the error between the equalized preamble and the original known preamble by choosing the optimal inverse of the channel response, W.

$$\min_{W} ||\vec{x}_{pre} - \vec{\hat{x}}_{pre}||^2 \tag{1.3}$$

$$\vec{\hat{x}}_{pre} = W(H\vec{x}_{pre} + \vec{v}) \tag{1.4}$$

$$W^* = \vec{x}_{pre}\vec{x}_{pre}^T H^T (H\vec{x}_{pre}\vec{x}_{pre}^T H^T + \sigma_v^2 I)^{-1} \tag{1.5}$$

Figure 1.2: The effects of a carrier frequency offset on the QPSK constellation.



The MMSE equalizer works well for known and unknown channels and does not amplify noise like the zero-forcing equalizer. This is why the MMSE equalizer paired with a least squares channel estimator are widely used.

Carrier Frequency Offset and Correction

Now, if we were to implement our minimum mean squared error equalizer on a physical receiver, we would find some problems with our equalization process. Our equalizer will equalize the first symbols very well. However, as we equalize end parts of our sequence, we will encounter a physical phenomonen called carrier frequency offset, CFO. Carrier frequency offset occurs when the transmitter and receiver are at slightly different frequencies. It also occurs when the transmitter and receiver are moving, causing a sort of Doppler effect.

When there is a significant CFO present, the symbols will gradually rotate. The received signals will be the original signals rotated at a rate ω.

$$\tilde{x}_i = x_i e^{ij\omega} + v_i \tag{1.6}$$

Figure 1.2 demonstrates the effects of CFO on a QPSK modulation constellation (no multipath channels). We show what the received signal symbols constellations are from a sequence of 100 symbols modulated in QPSK with different CFO rates. We assume that the CFO rate, ω, is constant during the transmission of the signal.

There are a few ways to handle CFO, some are more elegant than others. The first solution is to try to side-step the problem entirely. Since the effects of CFO depend on the length of a packet, one solution is to make packets so short that the symbols only move a little bit. In this case, the effects of CFO can be ignored.

Another solution is using a phase-locked loop, which is a control system that outputs a signal with a phase related to the input signal. A Costas loop is a circuit that implements a phase-locked loop for CFO correction for continuous time signals by adapting the sampling rate [3].

What happens when there is both CFO and inter-symbol interference? The received signal will have the effects of the channel and CFO as well as the noise.

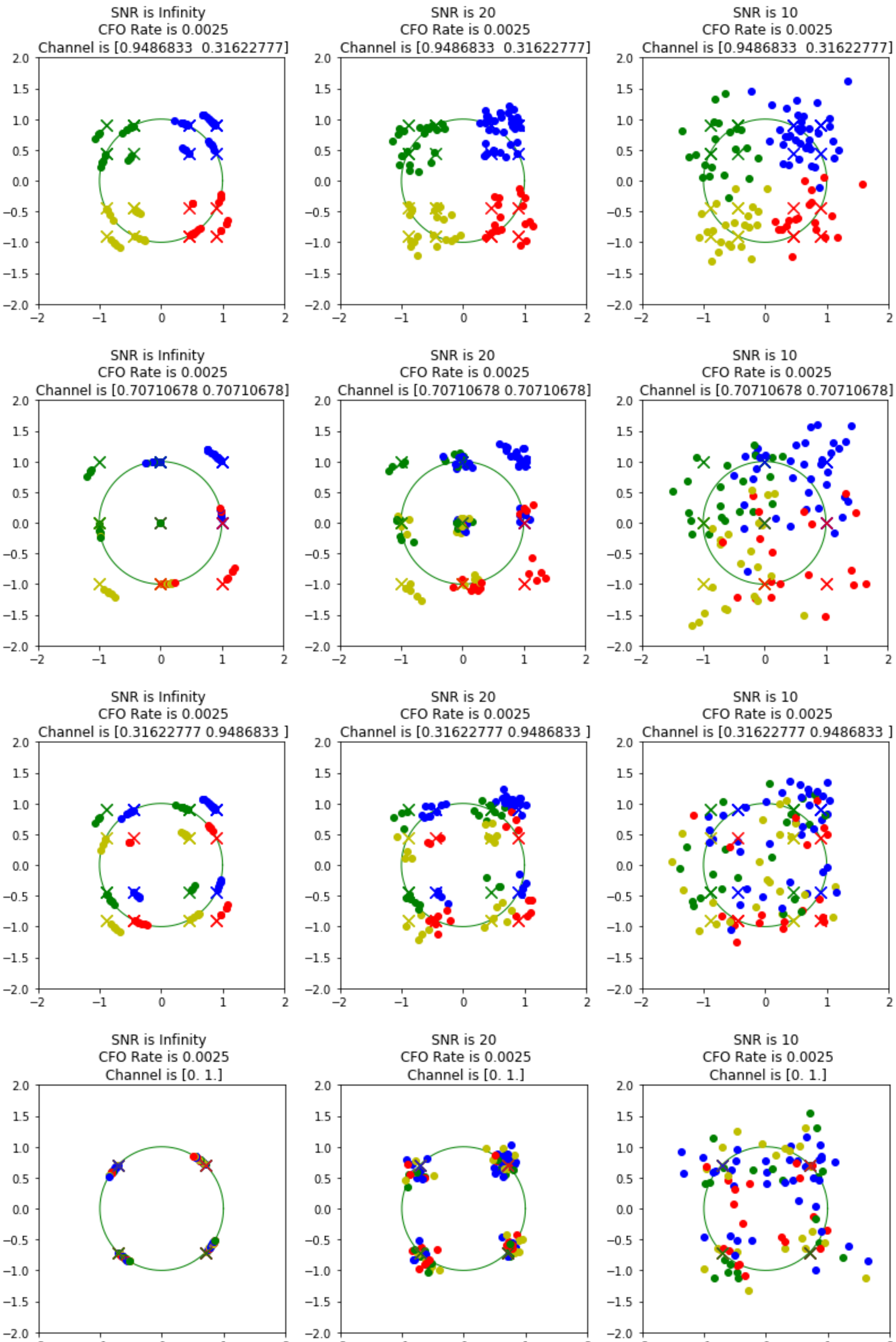
$$\tilde{x}_m = (\sum_{i=0}^{l} a_i x_{m-i}) e^{mj\omega} + v_i \tag{1.7}$$

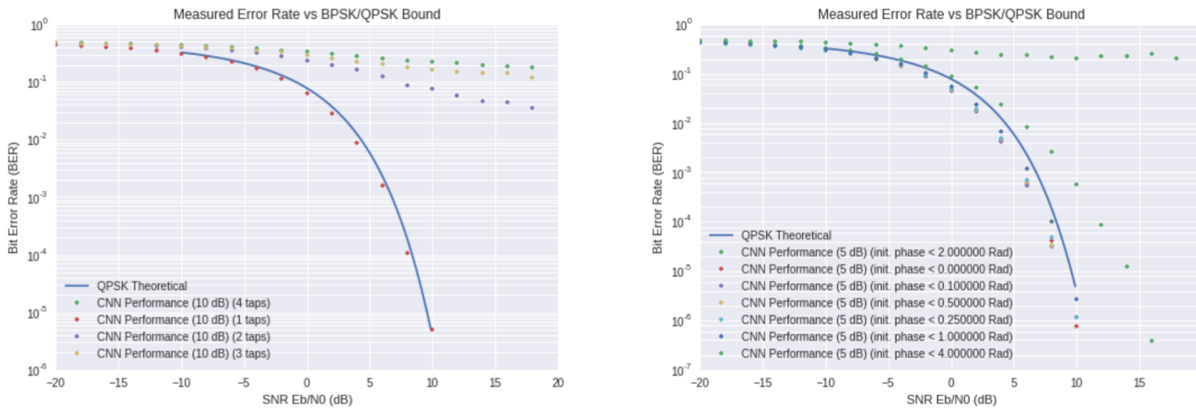
Figure 1.3 demonstrates the effects of CFO and multi-tap channels on a QPSK modulation constellation. We show what the received signal symbols constellations are from a sequence of 100 symbols modulated in QPSK with different CFO rates and two tap channels. We assume that the CFO rate, ω, and the channel taps are constant during the transmission of the signal. Modern day receivers have to combine CFO correction, channel estimation, and equalization processes in order to handle these received signals.

1.3 Related Works

Recently, many researchers have become more interested in applying deep learning techniques to communications systems. We refer the reader to [2][4][20][8][17] for surveys and

Figure 1.3: The effects of a two tap channel and a carrier frequency offset on the QPSK constellation.





(a) Impact of Delay Spread

(b) Impact of Random Initial Phase

Figure 1.4: The impacts of multipath channels and initial phase offsets on bit error performance [13].

motivations of these kind of works. Most of the works that we have found typically only deal with additive white gaussian noise (AWGN) channels or Rayleigh fading channels.

In [5], Dörner et. al. implemented an end to end transmitter and reciever with neural networks that allowed gradients to flow all the way back from the receiver during training. However, they restricted themselves to only sending a certain set of messages, only seeing AWGN channels, and they did not address CFO.

Other groups have been focusing on decoding with recurrent neural networks (RNNs) but also only deal with AWGN channels [9][10]. Others have been working with generative adversarial networks (GANs) to train an end-to-end communication system [21]. However, they also only consider AWGN channels or Rayleigh fading channels.

For those that do consider more complex channels, most re-train their models for each new channel seen. Ye et. al. consider OFDM systems where their feedforward neural networks estimate the channel state information then train offline for that specific channel [22]. [19] considers nonlinear channels but re-trains their network for each new channel. Note, this work does not go into detail about the architecture used or how the networks are trained.

Goldsmith and Farsad train a detector for optical and moleculular channels [6]. They assume a Poisson model for the channel and attempt to predict the probability mass function. However, they assume that they retrain for each Poisson parameter and they do not address CFO.

Timothy O'Shea's group has been doing some excellent work in this area. O'Shea's first work in this area jointly optimized a transmitter and receiver for a given channel model (AWGN and Rayleigh fading) but the work does not go into detail about the perforamnce for various channels [17].

In a subsequent paper, O'Shea et. al. explore how multipath channels and random inital phase affects the system performance [13]. Figure 1.4 shows how their performance drastically decreased with multipath channels and for non-zero phase offset.

In [15], they added an attention model to perform synchronization for time, frequnecy, phase and sample timing offset. However, their work showed that the synchronization was still quite noisy and did not perform much better than without the attention model [13]. Additionally, this work did not consider channel fading.

O'Shea et. al. has also attempted to use GANs to approxiate the channel response model for AWGN channels with and without phase noise[14]. The GANs were unable to find accurate probability density functions to represent the channel response.

In [18], they use convolutional neural networks (CNNs) to estimate, but not correct, carrier frequency offset and timing offset. However, they only consider AWGN channels and Rayleigh fading channels. In [16], they explore how unsupervised learning can train autoencoders for multiple antenna communications. They do re-train for each new channel and use a Rayleigh fading channel model.

Chapter 2

Deep Networks for Equalization

2.1 Channel Estimation

- compare least squares and how KNN did with pure deep nn based architecture
- NN did better than least squares
- hyperparam search over general 1-layer to 4-layer dense layers, and number of nodes and activations
- plots: how error changed with respect to data points, as number of data points increased, NN outperformed LS
- preamble: 100
- want: QPSK, plot of preamble length
- want? how to visualize that NN does better than LS

2.2 Channel Equalization

Learning an inverse

We explore whether a neural network can learn to do division.

- can a NN learn to do division?
- given a one tap channel and data sequence, output is equalized data sequence
- with and without log feature scaling
- without log errors = 10^-6

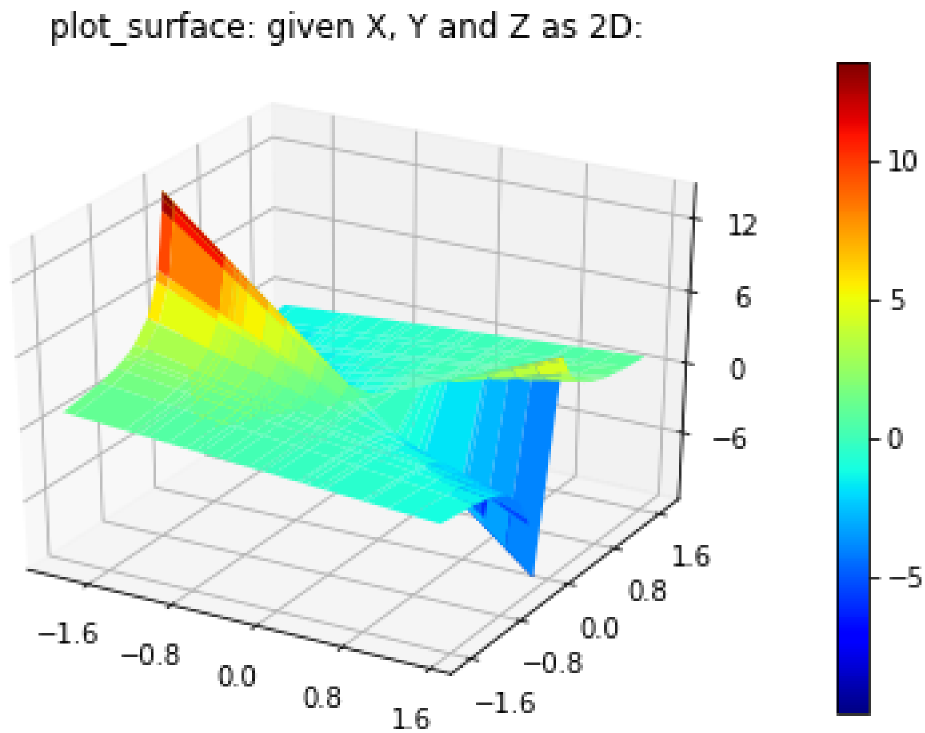


Figure 2.1: Topographical surface representation of the division function; $z = \frac{x}{y}$.

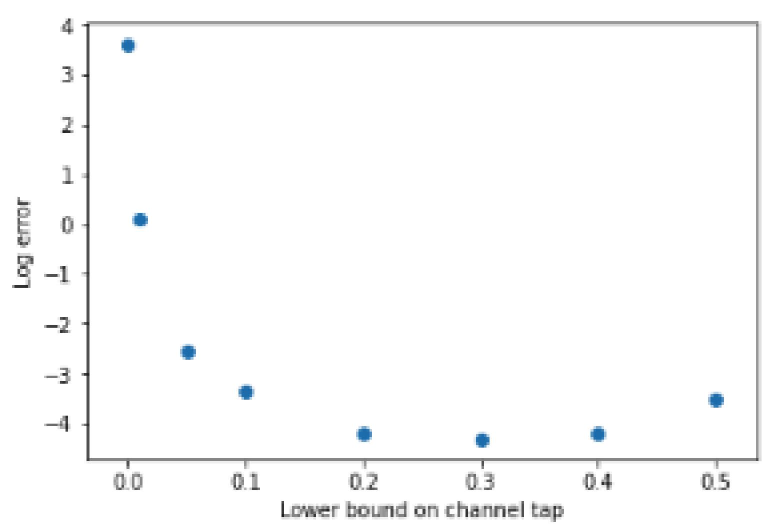


Figure 2.2: One tap channel inversion with respect to range.

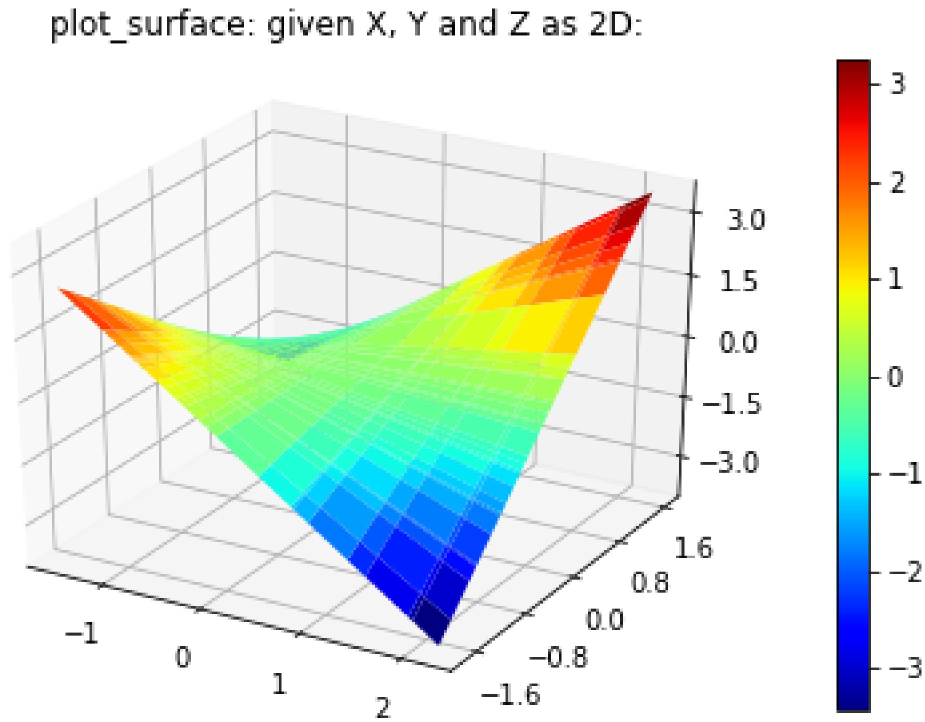


Figure 2.3: Topographical surface representation of the multiplication function; $z = xy$.

- MMSE gets error = 10^-32
- with log errors =
- straight inversion, without log feature scaling 10^-7 - with dense layers
- inversion with log feature scaling with error of 10^-14
- plots: inversion, but as a function of beta for both non-log and log

Learning to multiply two inputs

NN given x, y - output x*y

RNN for Channel Equalization

- compare to MMSE
- re run with the new RNN architecture
- backprop length of 3. crude search from 1-10. 2 tap channel

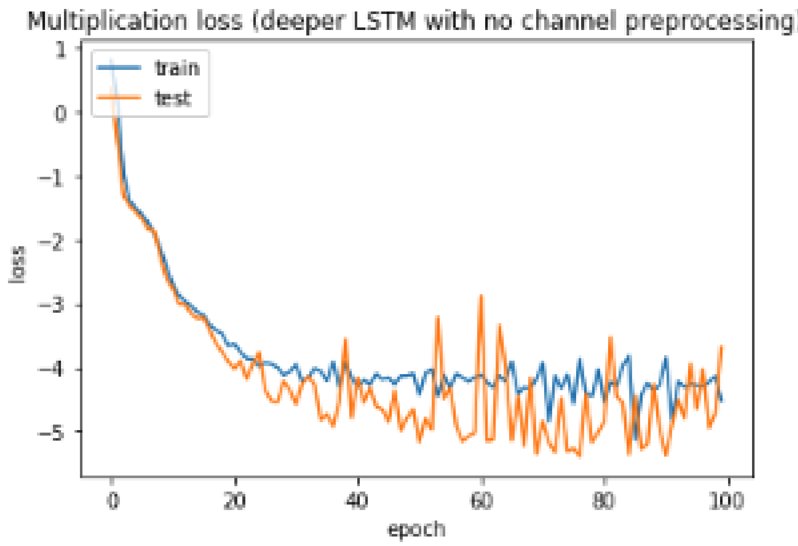


Figure 2.4: LSTM loss trying to learn the Multiplication function; $z = xy$.

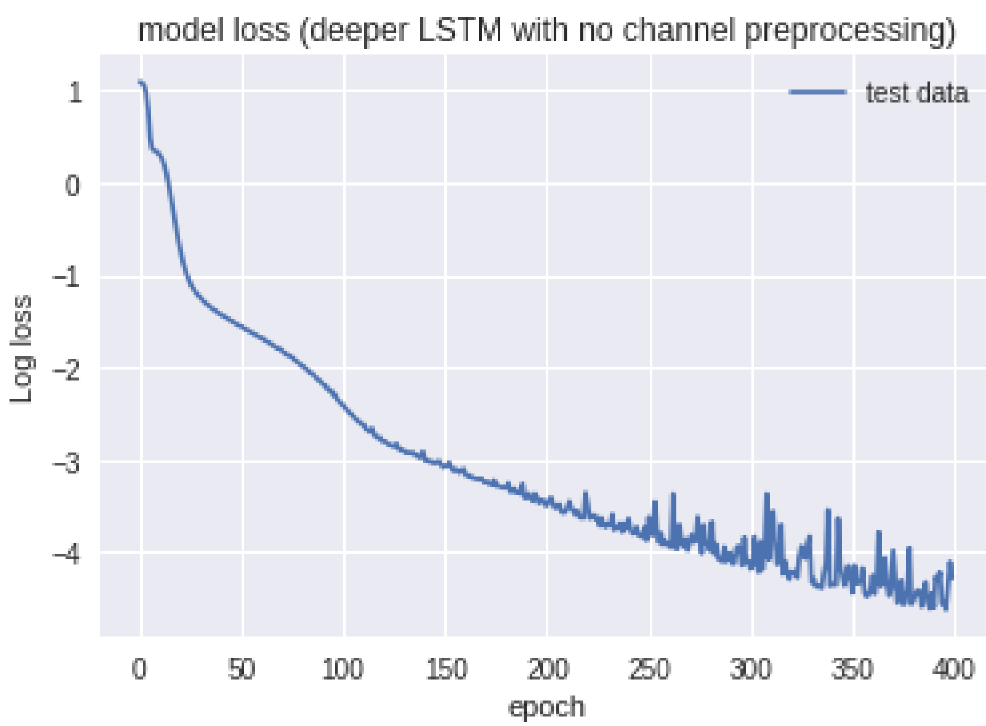


Figure 2.5: LSTM loss trying to learn the Multiplication function; $z = xy$.

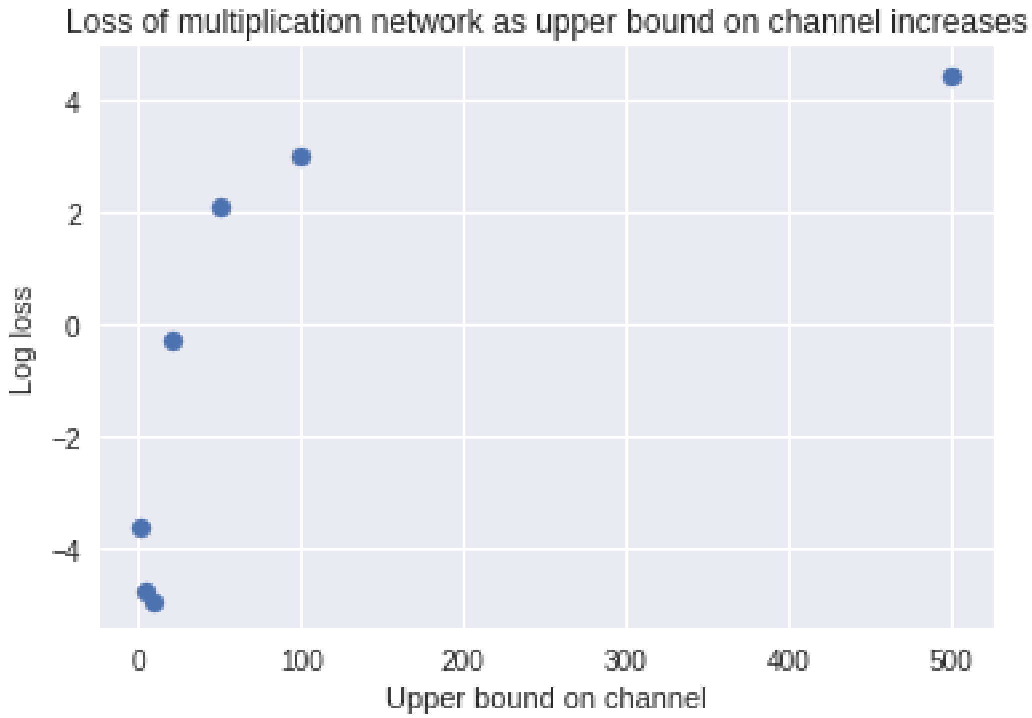


Figure 2.6: LSTM loss trying to learn the Multiplication function; $z = xy$.

- added channel preprocessing: didn't seem to make too much of a difference
- plot log/ log scale to find converging in error

Figure 2.7 shows which channels the RNN equalizer and classic demodulator got any bits wrong. There were a total of $50k$ random channels in the test set. Of that set, only 43 channels resulted in incorrect bits after the RNN equalizer and classic demodulator. From the figure, it is clear that these difficult channels are clustered into four regions. All of the four regions are when the first and second tap of the channel are equal in magnitude. The mean squared error between the equalized data and the true data among just the bad channels was 0.7306. The mean squared error among the good channels was 0.000264.

Tap 1	Tap 2	Counts of bad estimates	Mean Squared Error
0.707	0.707	14	0.7162
−0.707	0.707	1	0.5236
0.707	−0.707	16	0.8172
−0.707	−0.707	12	0.6492

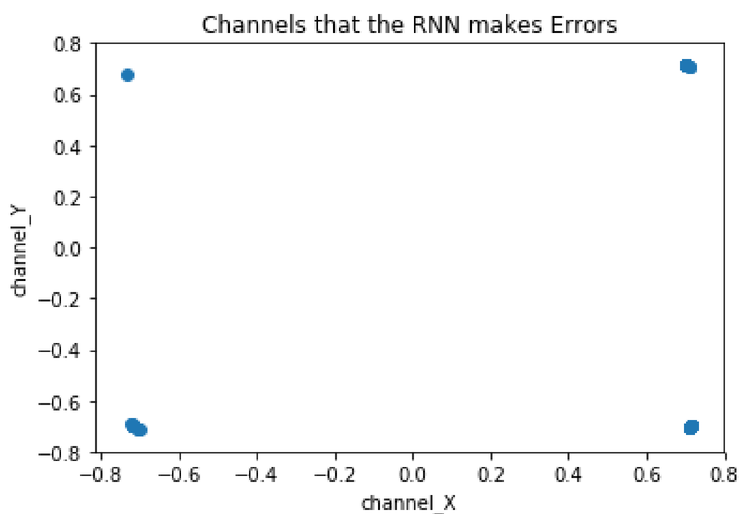


Figure 2.7: What two tap channels does the equalizer get wrong?

Chapter 3

Deep Networks for CFO

3.1 Recurrent Neural Network Follows a Circle

We first explore what kind of architectures are necessary for a recurrent neural network (RNN) to learn how to follow a circle. We show that a simple, small linear RNN can learn to act as the rotation matrix for a single rate of rotation around a circle. We also show that in order to have an RNN follow a circle for different rates of rotation, we need a large non-linear layer.

Single Rate

Figure 3.1 shows a recurrent neural network (RNN) that follows a circle for a single rate of rotation. The network's input is the starting point, $x[0]$, and it must predict the next 100 points, $x[1] \ldots x[99]$. Point $x[i]$ is rotated by $e^{ij\omega}$ where ω is the rate of rotation. The RNN architecture is one linear layer that takes the current point as input, $x[i]$, and outputs the next point, $x[i+1]$. We are forcing the state of the RNN to be the next point; state $= x[i+1]$.

Figure 3.1: Linear neural network: follow a circle for a constant CFO rate.

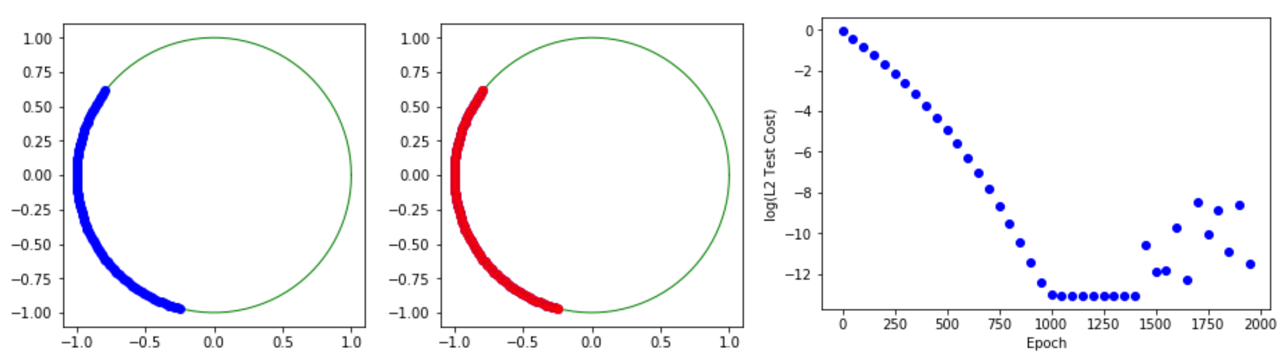
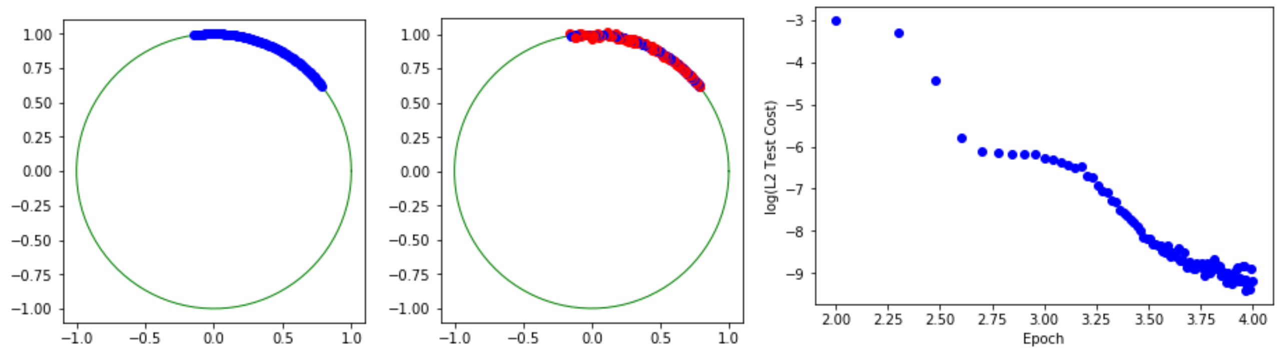


Figure 3.2: Nonlinear neural network: follow a circle for a different CFO rates.



We use a linear layer here because the network essentially has to learn how to become the rotation matrix which is linear with respect to the input.

$$R = \begin{bmatrix} \cos(\omega) & -\sin(\omega) \\ \sin(\omega) & \cos(\omega) \end{bmatrix} \tag{3.1}$$

The RNN was trained for a constant rate of rotation, ω, applied to sequences of 100 points. The network trained for $2k$ epochs, each with a batch size of $1k$ data sequences and with a learning rate of 0.006. The initial starting point, $x[0]$ is a uniform random variable drawn from on the unit circle. The network is tested on $1k$ new data sequences but with the same ω. Figure 3.1 shows the results of an RNN trained and tested for $\omega = 0.02$. The network achieves a loss on the order of 10^{-13}.

Different Rates

Figure 3.2 shows a recurrent neural network (RNN) that follows a circle for a given rate of rotation. The network's input is the starting point, x_0 and the rate of rotation, ω. The RNN must predict the next 100 points, $x_1 \dots x_{99}$. Point x_i is rotated by $e^{ij\omega}$.

We cannot use just linear layers in this case because the RNN must learn to find $\cos(\omega)$ and $\sin(\omega)$ for different ω. Essentially, this RNN needs to approximate the $\sin, \cos$ functions an then apply them to the data points.

The RNN architecture consists of one non-linear layer, with 100 nodes and Relu activation functions, and a linear layer at the output with 2 nodes. The input to the RNN is the estimated current point, $\hat{x}_i$, and the rate of rotation, ω. The RNN outputs the estimate of the next point, $\hat{x}_{i+1}$. We are forcing the state of the RNN to be the estimate of the next point concatenated with the rate of rotation; state $= [\hat{x}_{i+1}, \omega]$ so that state then becomes the input to the next run of the RNN.

The RNN is trained for different rates of rotation, ω, applied to sequences of 100 points. The rate of rotation is a uniform random variable drawn from $0 - \frac{1}{50}$. We use mean squared

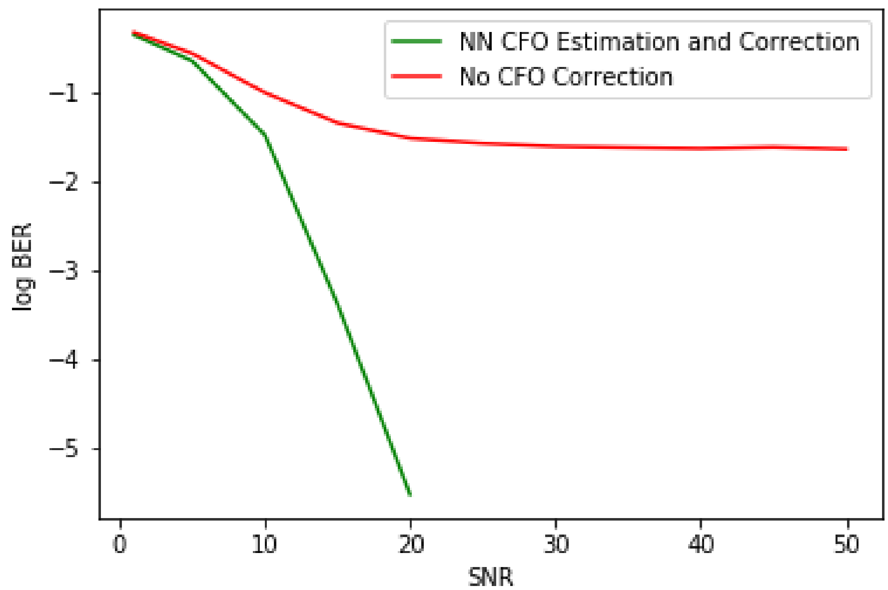


Figure 3.3: The log BER of received signals passed through a neural network CFO estimator with a rotation matrix CFO correction and classic demodulator. The log BER of the same signals passed through just the classic demodulator.

error of the true data sequence and the estimated data sequence for the loss function; $||\vec{x} - \vec{\hat{x}}||^2$. The network is trained for $10k$ epochs, each with a batch size of $1k$ data sequences and with a learning rate of 0.01. The initial starting point, x_0 is a uniform random variable drawn from on the unit circle. The network is tested on $100k$ new data sequences with random ω. Figure 3.1 shows the results of an RNN trained and tested for $10k$ epochs. The network achieves a loss on the order of 10^{-5}.

3.2 Deep Network Carrier Frequency Offset Estimation

- complex gradients problems
- act like the real and imaginary parts are separate
- plots: one tap channel plots, without equalization problems
- plots: two tap channel plots, with equalization problems

Figure 3.3 shows how...

3.3 Deep Network Carrier Frequency Offset Correction

Program a Costas loop for comparison

Chapter 4

Conclusion

conclucsion

Seria Electronică Şi Telecommuncaţii, Transactions on Electronics and Communications

Bibliography

[1] Yoshua Bengio, Samy Bengio, and Jocelyn Cloutier. "Learning a synaptic learning rule". In: *Technical Report 751, Départment d'Informatique et de Recherche Opérationelle, Université de Montréal, Canada* (1990).

[2] Corina Botoca, Georgeta Budura, and Miclau Nicolae. "Nonlinear Channel Equalization Using Complex Neural Networks". In: *Seria Electronică Şi Telecommuncaţii, Transactions on Electronics and Communications* (2004), pp. 227–231.

[3] John P. Costas. "Synchronous Communications". In: *Proceedings of the IRE* 44.12 (1956), pp. 1713–1718.

[4] Theo Diamandis. "Survey on Deep Learning Techniques for Wireless Communications". In: *Departement of Electrical Engineering, Stanford University* (2017).

[5] Sebastian Dörner et al. "Deep Learning-Based Communication Over the Air". In: *arXiv preprint arxiv:1707.03384* (2017).

[6] Nariman Farsad and Andrea Goldsmith. "Neural Network Detection of Data Sequences in Communication Systems". In: *IEEE Transactions on Signal Processing* PP (2018).

[7] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-agnostic meta-learning for fast adaptation of deep networks". In: *International Conference on Machine Learning* (2017).

[8] Hengtao He et al. "Model-Driven Deep Learning for Physical Layer Communications". In: *arXiv preprint arxiv:1809.06059* (2018).

[9] Hyeji Kim et al. "Communication Algorithms via Deep Learning". In: *International Conference on Learning Representations* (2018).

[10] Hyeji Kim et al. "Deepcode: Feedback Codes via Deep Learning". In: 2018.

[11] Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. "Human-level concept learning through probabilistic program induction". In: *Science*. Vol. 350. 6266. 2015, pp. 1332–1338.

[12] Christiane Lemke, Marcin Budka, and Bogdan Gabrys. "Metalearning: a survey of trends and technologies". In: *Artificial Intelligence Review*. Springer, 2015, pp. 117–130.

[13] Timothy J. O'Shea, Kiran Karra, and T. Charles Clancy. "Learning to Communicate: Channel Auto-encoders, Domain Specific Regularizers, and Attention". In: *IEEE International Symposium on Signal Processing and Information Technology*. 2016, pp. 223–228.

[14] Timothy J. O'Shea, Tamoghna Roy, and Nathan West. "Approximating the Void: Learning Stochastic Channel Models from Observation with Variational Generative Adversarial Networks". In: *arXiv preprint arxiv:1805.06350* (2018).

[15] Timothy J. O'Shea et al. "Radio Transformer Networks: Attention Models for Learning to Synchronize in Wireless Systems". In: *arXiv preprint arxiv:1605.00716* (2016).

[16] Timothy O'Shea, Tugba Erpek, and T. Charles Clancy. "Deep Learning-Based MIMO Communications". In: *arXiv preprint arXiv:1707.07980* (2017).

[17] Timothy O'Shea and Jakob Hoydis. "An Introduction to Deep Learning for the Physical Layer". In: *IEEE Transactions on Cognitive Communications and Networking*. Vol. 3. 4. 2017, pp. 563–575.

[18] Timothy O'Shea, Kiran Karra, and T. Charles Clancy. "Learning Approximate Neural Estimators for Wireless Channel State Information". In: *arXiv preprint arXiv:1707.06260* (2017).

[19] K. T. Raghavendra and Amiya K. Tripathy. "An Efficient Channel Equalizer Using Artificial Neural Networks". In: *Neural Network World* 16 (2006), pp. 357–368.

[20] Tianqi Wang et al. "Deep Learning for Wireless Physical Layer: Opportunities and Challenges". In: *China Communications* 14.11 (2017), pp. 92–111.

[21] Hao Ye, Geoffrey Ye Li, and Biing-Hwang Juang. "Channel Agnostic End-to-End Learning based Communication Systems with Conditional GAN". In: *IEEE Global Communications Conference*. 2018.

[22] Hao Ye, Geoffrey Ye Li, and Biing-Hwang Juang. "Power of Deep Learning for Channel Estimation and Signal Detection in OFDM Systems". In: *IEEE Wireless Communications Letters*. 2018, pp. 114–117.