Learning to Communicate: Channel Auto-encoders, Domain Specific Regularizers, and Attention

Timothy J. O'Shea
Virginia Tech, Arlington, VA
Email: oshea@vt.edu

Kiran Karra
Virginia Tech, Arlington, VA
Email: kiran.karra@vt.edu

T. Charles Clancy
Virginia Tech, Arlington, VA
Email: tcc@vt.edu

***Abstract*—We address the problem of learning an efficient and adaptive physical layer encoding to communicate binary information over an impaired channel. In contrast to traditional work, we treat the problem an unsupervised machine learning problem focusing on optimizing reconstruction loss through artificial impairment layers in an autoencoder (we term this a channel autoencoder) and introduce several new regularizing layers which emulate common wireless channel impairments. We also discuss the role of attention models in the form of the radio transformer network for helping to recover canonical signal representations before decoding. We demonstrate some promising initial capacity results from this approach and address remaining challenges before such a system could become practical.**

I. Introduction

Radio communication theory has long sought to find algorithms to attain efficient transfer of information over a variety of communications channels. These range from thermal noise limited channels exhibiting Gaussian noise behaviors to more complex channels exhibiting multi-path fading, impulse noise, spurious or continuous jamming, and numerous other complex impairments. Information theory [1] and analysis of modulation schemes gives us bounds on achievable capacity and information density vs bit error rates for given modulation schemes, bandwidths and signal to noise ratios (SNRs), but does not tell us generally how to achieve them.

Throughout years we have achieved numerous discrete operating steps near these bounds allowing efficient operation at specific SNR levels. Rate matching and code adaptation have allowed us to operate close to this capacity curve at numerous discrete SNR levels, but many of the techniques to get there are computationally complex in practice and require expensive hardware and DSP software tuning to implement in mobile radios.

By taking the approach of unsupervised learning of an end-to-end communications system by optimization of reconstruction loss in a channel auto-encoder with a set of domain appropriate representative channel regularizers, we provide a method to learn new modulation schemes and methods of modulation which blur the lines between modulation and error correction, providing similar capacity, bit error rate performance (BER), and lower computational complexity requirements at runtime than current systems while removing the need for low level expert design. This approach will lead to a new class of communications systems with a greater ability to generalized and adapt to impairments which traditional communication systems can not easily, often due to decision baked into expert design. The impact of this approach to wireless system development, deployment and capabilities holds enormous potential.

II. Channel Auto-encoders

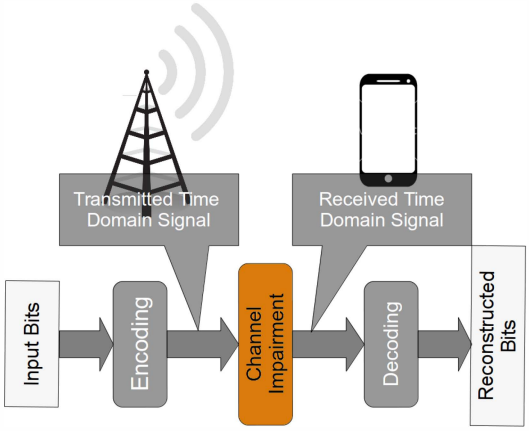


Fig. 1: The channel auto-encoder scenario

We introduce the channel auto-encoder as our primary method for learning end-to-end radio communications systems. In its simplest form, the channel auto-encoder includes an encoder, a channel regularizer, and a decoder. In this paper we limit our scope to binary channel auto-encoders in which the input values are binary and our output goal is to reconstruct these input bits, but the same architecture could apply to the encoding of real valued signals as well. Figure 1 provides a simple high level illustration of this model.

Auto-encoders [3] provide a powerful method for performing unsupervised learning. They optimize reconstruction loss through a series of representations typically using a mean squared error (MSE) objective and a stochastic gradient descent (SGD) solver to find network weights achieving an effective regression. This has been shown to work well in numerous domains, especially in computer vision, we previously demonstrated the viability on radio signals in [12].

Numerous enhancements have been introduced which expand on the basic autoencoder by using layers such as convolutional layers [7] to force shift invariance and reduce the number of network parameters, and the use of regularizers such as L1 and L2 regularization, input noise [6] and dropout [9]. These regularizers are general purpose as they help prevent over-fitting in any network. We also supplement them with

978-1-5090-5844-0/16/$31.00 ©2016 IEEE

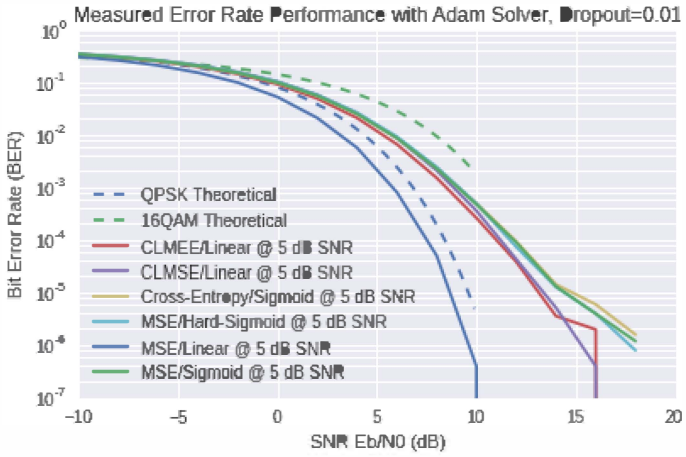


Fig. 2: AWGN BER Performance vs SNR of for Different Loss Functions training at 5dB SNR

domain specific regularizing effects, which introduce variations due to specific channel impairments during every training pass, preventing over-fitting to specific channel conditions in the training data set.

For each measurement we use a dataset of 100,000 examples, each 128 random bits, this is a minuscule fraction of the complete input code space (2^{128} codes), but sufficiently allows our network to converge in most cases. We partition this dataset into 80% training and 20% test/validation.

III. Loss in Channel Auto-encoders

A. Mean Square Error Based Loss

Mean squared error is a commonly used loss function in auto-encoders for regression and we apply it here on discrete bits in $0, 1$. We compare sigmoid or hard sigmoid (which saturate to 0 or 1) and linear output activations. In the case of the linear activation, we consider several loss functions.

For linear output activation, MSE is not obviously the best choice because a large amplitude less than 0 or greater than 1, may be treated as a high likelihood, and may not be something we wish to penalize. This leads to the idea of several clipped error functions considered below, where t is our target bit and p is our network output.

$$\mathcal{L}_{MSE} = (t-p)^2 \tag{1}$$

$$\mathcal{L}_{CLMSE} = \begin{cases} (t-p)^2 \times \mathbb{1}(p>0) & \text{if } t=0 \\ (t-p)^2 \times \mathbb{1}(p<1) & \text{if } t=1 \end{cases} \tag{2}$$

$$\mathcal{L}_{CLMEE} = \begin{cases} e^{t-p} & \text{if } t=0 \\ e^{p-t} & \text{if } t=1 \end{cases} \tag{3}$$

$$\mathcal{L}_{CLMLE} = \begin{cases} (t-p) \times \mathbb{1}(p>0) & \text{if } t=0 \\ (p-t) \times \mathbb{1}(p<1) & \text{if } t=1 \end{cases} \tag{4}$$

where $\mathbb{I}(\cdot)$ is the indicator function.

The $\mathcal{L}_{MSE}$ is the classic MSE function, which provides an equal penalty proportional to the squared distance between predicted and target values. $\mathcal{L}_{CLMSE}$ penalizes predicted values proportional to the squared distance to the target only on the wrong side of the threshold γ. $\mathcal{L}_{CLMLE}$ is similar but linear instead of squared. And finally $\mathcal{L}_{CLMEE}$ penalizes the loss exponentially in a one-sided way. The crucial difference between this and the clipped linear and squared loss functions are that it never allows the loss to go to zero. Our hypothesis behind this is that it would encourage the solver to encourage high likelihoods, pushing the predicted values as far away as possible from the threshold.

B. Cross-Entropy Loss

Cross entropy loss between input and output bits is another potential choice which is quite appealing in the case of predicting binary outputs from binary inputs. We may constrain the output to the range of $(0, 1)$ by using a sigmoid output layer activation and then consider the cross-entropy loss given in the equation below where $\{x_i\}$ consists of the input bits and $\{\hat{x}_i\}$ consists of the output bits.

$$\mathcal{L}_{CE} = \sum_i x_i log\hat{x}_i + (1-x_i)log(1-\hat{x}_i) \tag{5}$$

C. Evaluation

We compare these loss functions by measuring the bit error rate (BER) performance of the channel auto-encoder after training at a fixed SNR. Results are shown in Figure 2.

We evaluate BER performance with both the RMSprop [10] and Adam [8] solvers at several learning and dropout rates and obtain slightly better performance with the Adam solver. The results show that the best performance seems to occur when using $\mathcal{L}_{MSE}$ with a linear output. This is somewhat counter intuitive as it needlessly penalizes higher likelihoods, but still warrants future investigation. It should be noted that we are still constraining the network to linear intermediate activations and only 2 convolutional filters in the case of the CNN, further constraining performance for now.

IV. Domain Specific Regularization

Radio signal propagation is a heavily characterized phenomenon which we can model reasonably well. We select several common impairments that occur during over the air transmission and build corresponding channel regularization layers to evaluate reconstruction learning capacity under each. Initially considered are the effects of:

- R_{noise}: Additive Gaussian thermal noise
- R_{toa}: Unknown time and rate of arrival
- R_{foa}: Carrier frequency and phase offset
- R_h: Delay spread in the received signal

Gaussian noise and dropout are commonly used regularizers [9] [4], however their typical use is at training time only, removing them at evaluation time. In our case we are actually seeking to train representations which can cope with similar noise at evaluation time, emulating the expected channel impairment effects as we measure the network's performance. In radio communications, additive noise like this is widely used to model thermal noise. In this layer we add a real Gaussian random

variable N to each in-phase (I) and quadrature (Q) sample passed through the channel where $N \sim N(0, \frac{10^{(-SNR_{dB}/10.0)}}{\sqrt{2.0}})$. We normalize input activations to an average power of 1 prior to this operation.

Unknown time and rate of arrival has a direct equivalence to the computer vision domain, here our regularizer applies an unknown shift and scaling in the time domain which our encoding must cope with. In radio systems, this occurs as radio propagation times vary and clocks on transmit and receive systems are typically not synchronized to each other. We choose a time shift of $\theta_t \sim N(0, \sigma_t)$ and a time-dilation rate $\theta'_t \sim N(1, \sigma'_t)$ and apply it to activations.

Carrier frequency and phase offset does not have an obvious equivalent in the vision domain, but would be equivalent to some form of mixing between RGB channels around a random spherical rotation. Since radio typically uses complex baseband sampling, and in this representation, unknown offsets in center frequency, phase and Doppler shift result in static or linear mixing of the the two complex base-band components (I and Q). To simulate a real system, this layer randomly picks a phase $\theta_f \sim U(0, 2\pi)$ and a frequency offset $\theta'_f \sim N(0, \sigma_f)$, where σ_f is our expected center frequency offset error and applies this rotation to activations.

Lastly, Delay spread in the received signal simulates the arrival of numerous delayed and phase shifted copies of a signal at the receiver. Since this is simulated as a linear system and we assume stability over a single sample time window, we can choose a random non-impulsive channel delay spread filter and convolve it with the input signal to obtain an output which has been spread in time linearly according to a random channel response.

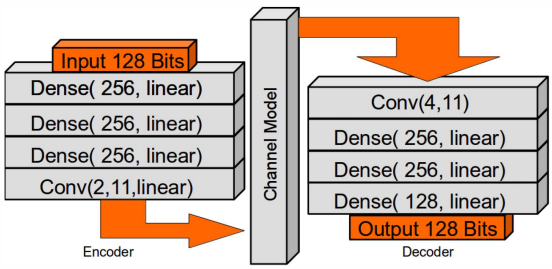
We implement each of these regularizers as tensor-operations and layers in Keras such that they can be easily applied to the encoded signal representation of the auto-encoder.

V. Network Structure Selection and Evaluation

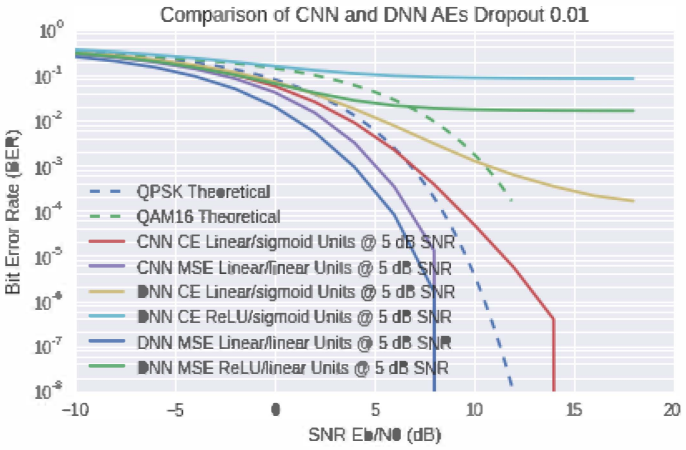
We compare several network structures for our auto-encoder, including a deep fully-connected (dense) neural network (DNN) shown in 3a and a convolutional neural network (CNN). In both cases we consider several activation functions for our hidden units, and a linear layer on the output of the encoder and decoder units, or a sigmoid output activation in the case of cross-entropy loss.

To evaluate learned representations, we focus on bit error rate (BER) as a function of signal-to-noise ratio (SNR), an important metric in communications which characterizes how reliably a system can communicate as the channel degrades. We compare results with curves for the commonly used QPSK and QAM-16 modulation schemes [2] whose performance in terms of SNR vs BER can be expressed analytically using the expressions from equations 6 and 7. These benchmarks capacities are shown as dotted lines to provide a comparison to widely used modern day expert modulation performance.

$$\textbf{QPSK:} P_b = \frac{1}{2}\,\mathrm{erfc}\left(\sqrt{\frac{E_b}{N_0}}\right) \tag{6}$$



(a) Auto-encoder Construction



(b) Candidate Performance

Fig. 3: [a] Encoder and Decoder structure used; and [b] Comparison of CNN and DNN architectures on AWGN only

$$\textbf{QAM16:} P_b = \frac{3}{8}\,\mathrm{erfc}\left(\sqrt{\frac{4E_b}{10N_0}}\right) \tag{7}$$

It is critical to note these are **not** Shannon Capacity, we are comparing against expert modulation scheme error performance without forward error correction. We can observe the actual information theoretic performance of these and other modulations in figure 4, so we are **not** claiming to be beating the Shannon bound! Coded modulations would also move left on this graph.

Our information rate is fixed at the number of hidden units our bits are represented by. For AWGN we obtain best performance using a DNN with linear activations, however performance relies on impulsive delay spread providing independence activation in time. A comparison between CNN and DNN performance is shown in figure 3b.

A. *Effects of Training SNR*

The SNR of the channel model used in training has a significant effect on the ultimate performance of the resulting channel encoding scheme. In figure 5a we show performance plots resulting from training the CNN at a range of different training SNRs. We tend to obtain best system performance when training at low positive SNR values such as 5dB. This is likely because here we are stressing the hidden units to represent

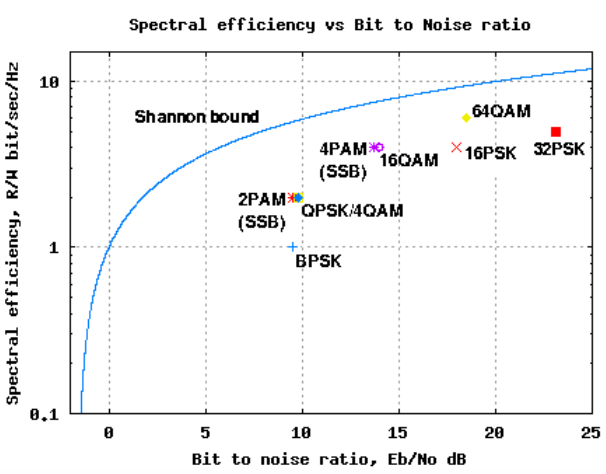
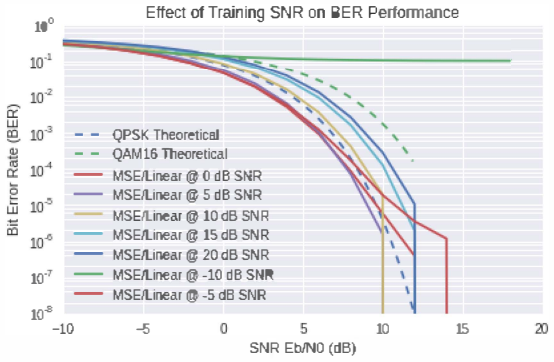
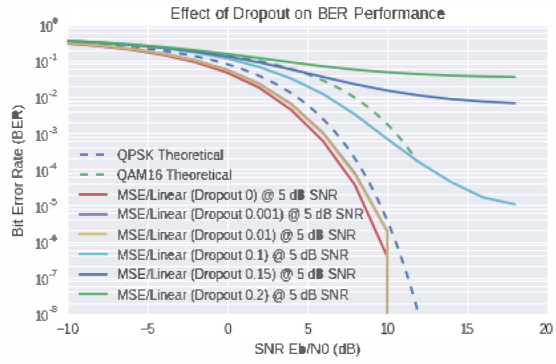


Fig. 4: Un-coded Modulation Information Capacity at a BER of 10^{-5} from [5]



(a) Performance vs Training SNR



(b) Performance vs Training Dropout

Fig. 5: [a] Effects of training set regularisation SNR on resulting encoding error rate performance; and [b] Effect of dropout rate regularization on

as much information as possible to perform reconstruction, where as at negative SNR that information is too distorted to learn effectively, and at high SNR significant resilience is not required to recover from channel effects.

B. *Effects of Dropout*

The effect of dropout when training such a system varies from typical ML systems in that our hidden units over the radio channel are constrained in time. It also varies in that we have a lot of binary inputs which are uncorrelated independent bits and a lot of information which we must learn to represent and maintain through. We train with hidden units of twice the width of the input number of bits to allow for numerous competing models to form with dropout. However, since we seek to preserve and reconstruct all of the bits independently at the output, any substantial amount of dropout seriously degrades the information conveyed through the network. We find the best training performance to be with low levels of dropout. Figure 5b shows this effect of dropout on the ultimate channel reconstruction performance of the network after training. In each of these cases MSE is not a particularly good indicator of the ability to generalize across different SNRs. We expect dropout would be more useful when including more non-linear activations.

C. *Effects of Channel Delay Spread*

Delay spread varies in communication systems, from wire-line and space based wireless systems which can sometimes have very short impulsive channel responses to HF and dense multi-path wireless systems which can have long delay spreads. We model this by introducing a regularization layer which takes the form:

$$out = conv(in, h) : h \sim U(-1, +1, shape = [n_{taps}]) \quad (8)$$

That is, for each example through the wireless channel, we convolve a new random filter with given length: n_{taps}. This is a pessimistic view for a real system since h really has a more defined probabilistic envelope, but we use this simplified expression for the scope of this work.

D. *Effect of Channel Frequency and Phase Variance*

Frequency and absolute phase of arrival of a waveform at the waveform are equivalent to a transform of the following form. We have a transmitted input signal of shape x = [2,$n_{\textbf{time samples}}$], where I and Q are on separate channels. A random phase results in the effect as follows.

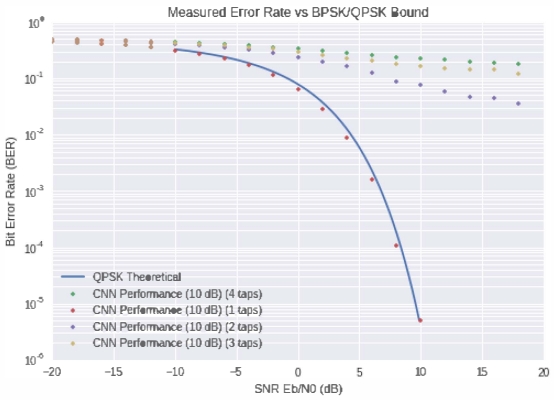
$$y = exp(j\theta) * x \quad (9)$$

Or kept in real terms which we can easily implement in real tensor based systems such as Keras, we have

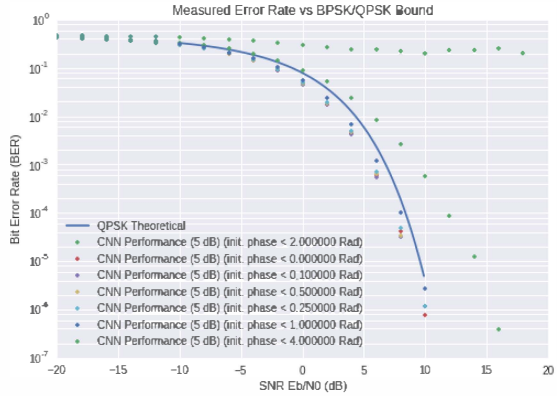
$$y = \begin{vmatrix} x[0,:]cos(\theta) - x[1,:]sin(j\theta) \\ x[0,:]sin(\theta) + x[1,:]cos(j\theta) \end{vmatrix} \quad (10)$$

The effect for frequency offset is similar except that phase becomes linearly time varying as shown below.

$$y = exp(j(\theta + \theta' t)) * x \quad (11)$$



(a) Impact of Delay Spread



(b) Impact of Random Initial Phase

Fig. 6: [a] Effect of delay spread on the system without attention; and [b] The effect of random initial phase on system performance

Unfortunately invariance to this transform is not readily learned in our experimentation. In figure 6b we see that once our initial phase is distributed between 0 and 2 π, we obtain extremely poor performance. To help our system learn the additional necessary invariance to cope with these channel effects, we introduce a model for attention.

VI. Attention Models for Receiver Synchronization

Performance decays rapidly with delay spread, as we lose the independent information within each hidden unit passing over the channel. To address this we introduce the notion of a Radio Transformer Network for attention. We leverage the end-to-end localization and discriminative network introduced in [11] and adapted to the radio domain in [13]. The localization network estimates parameters such as time of arrival, frequency offset, phase of arrival, and channel response taps, while transforming layers then apply it to the received data.

In the cases of phase and frequency offset, as well as time offsets, and non-impulsive channel responses, we implement a specific radio transformer algorithm as a layer in Keras for each. Our localization network then becomes a parameter estimation and regression network for these transforms followed by a decoder. This network then presents an entire comprehensive

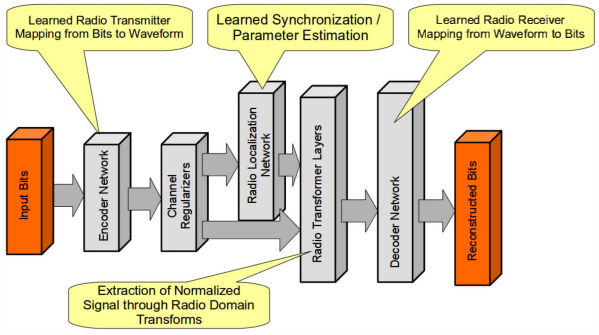


Fig. 7: Our end-to-end unsupervised Radio Transformer Model. Joint learning of transmit, receive and synchronization-attention approaches to minimize bit error rate

scheme for learning end to end encoding, decoding, and synchronization networks for naively learned communications encodings. It can be used completely unsupervised and adapt the channel encoding scheme to any specific radio regularization layer or configuration which might be appropriate for the intended radio use environment.

VII. Visualization of Learned Modulations

It is interesting to look at the learned convolutional features which become the representation basis for the signal over the channel. In figure 8 we show these at 1,2,3 and 4 sample delay spread widths and they seem to take a form reminiscent of a time-frequency style wavelet basis with varying compactness in time depending on the channel delay spread.

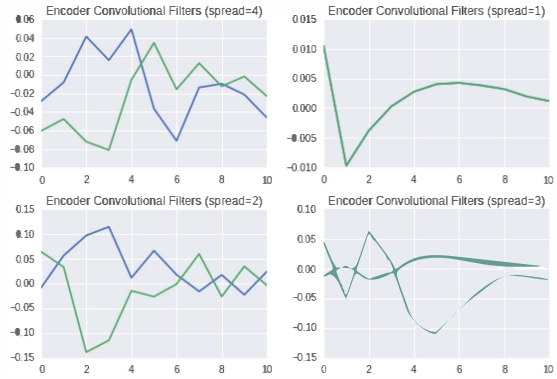


Fig. 8: Encoder Convolutional Basis functions vs Delay Spread

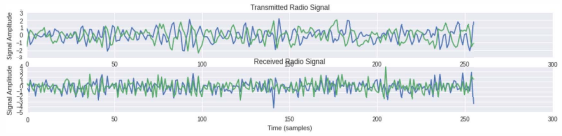


Fig. 9: Example transmit and receive signal at 0dB SNR for 128 bits of information. In-phase and Quadrature components are shown.

In figure 9 we show what the same transmit and receive signal looks like throughout our system at 0dB after channel

regularization. Its interesting to note that the modulation basis here is not clearly recognizable as any existing modulation we use widely now, it seems to use at least 3 common discrete levels, but potentially encodes information in some mixture of time and frequency bins across the sample space.

VIII. Computational Complexity

Computational complexity of training deep neural networks can be quite high, however the feed forward execution of a trained system can be quite low since it takes the form of dense matrix operations. In contrast many error correction codes improving error rate performance in modern day systems rely on sparse or iterative operations potentially requiring more power. The tensor based processing associated with our approach is growing increasingly popular for making efficient use of concurrent compute architectures. Our approach holds the potential to greatly reduce power requirements by lowering clock rates, increasing algorithm concurrency, and adapting.

IX. Realistic Deployment Considerations

In the real world, there are a number of important considerations which differ from simulation assumptions discussed here.

A learned communications scheme could be used in several ways. Systems trained on channel models can be easily deployed in a offline training scenario. In this case learned representations are optimized offline only. Gradients of the channel model can be used directly during training and given sufficiently accurate stable analytic models of the channel, we can learn an efficient set of representations for transport across a similar real the channel without any on-line adaptation.

Since channels may vary in the real world depending on deployment location, conditions, or nearby effects, such a system could also hold potential to be able to perform on-line adaptation and on-line learning of specialized representations which perform well for the given deployment scenario. In this case there are two major considerations which much be addressed in the future for such as system. First is that we do not have analytic channel gradient expressions and must rely on gradients approximation methods, and second is that we are constrained by error feedback bandwidth and latency among nodes collaborating on system model improvement while learning. Methods exist to address both of these issues but will require significant additional work.

X. Conclusion

From a wireless communications perspective, the potential impact of learning how to communicate information optimally based on the environment without expertly designed modulation schemes is enormous. Although Shannon's seminal work defines the bounds of information communication over wireless channels, it does not explain how to reach those bounds. Resultingly, billions of dollars have been spent in communications research to try to achieve these bounds, and yet no general solution yet exists in all scenarios. In some cases algorithmic complexity in expert techniques to operate near capacity are still quite high.

We have demonstrated that this architecture **is** viable for the design and implementation of end-to-end learned communications systems with potential for performance rivaling modern day systems, nearly achieving Shannon capacity while maintaining generality and low complexity.

We have also proposed how radio transformer networks could in conjunction be used to learn synchronization methods for extracting canonical recovered signals, but have not yet demonstrated this. We have shown how choices such as training SNR and Dropout rate have significant impact on model capacity, but need further work to refine best practices surrounding them. We believe this investigation not only enables a powerful new application of unsupervised learning in the radio communication domain, but offers insight into the information capacity of similar deep neural networks.

References

[1] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 3–55, Jun. 1948, ISSN: 1559-1662.

[2] J. Oetting, "A comparison of modulation techniques for digital radio," *IEEE Transactions on communications*, vol. 27, no. 12, pp. 1752–1762, 1979.

[3] G. E. Hinton and R. S. Zemel, "Autoencoders, minimum description length, and helmholtz free energy," *Advances in neural information processing systems*, pp. 3–3, 1994.

[4] Y. Raviv and N. Intrator, "Bootstrapping with noise: An effective regularization technique," *Connection Science*, vol. 8, no. 3-4, pp. 355–372, 1996.

[5] K. Pillai, *Modulation roundup: Error rates, noise, and capacity*, http://www.embedded.com/print/4017668, 2008.

[6] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*, ACM, 2008, pp. 1096–1103.

[7] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *Artificial Neural Networks and Machine Learning–ICANN 2011*, Springer, 2011, pp. 52–59.

[8] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *ArXiv preprint arXiv:1412.6980*, 2014.

[9] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[10] Y. N. Dauphin, H. de Vries, J. Chung, and Y. Bengio, "Rmsprop and equilibrated adaptive learning rates for non-convex optimization," *ArXiv preprint arXiv:1502.04390*, 2015.

[11] M. Jaderberg, K. Simonyan, A. Zisserman, *et al.*, "Spatial transformer networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 2008–2016.

[12] T. J. O'Shea, J. Corgan, and T. C. Clancy, "Unsupervised representation learning of structured radio communication signals," *ArXiv preprint arXiv:1604.07078*, 2016.

[13] T. J. O'Shea, L. Pemula, D. Batra, and T. C. Clancy, "Radio transformer networks: Attention models for learning to synchronize in wireless systems," *ArXiv preprint arXiv:1605.00716*, 2016.