Using Sequence to Sequence Learning for Digital BPSK and QPSK Demodulation

Sarunas Kalade*, Louise Crockett and Robert W. Stewart
Department of Electronic and Electrical Engineering, University of Strathclyde
204 George Street, Glasgow, G1 1XW, Scotland, UK
Email: *sarunas.kalade@strath.ac.uk

***Abstract*—In the last few years Machine Learning (ML) has seen explosive growth in a wide range of research fields and industries. With the advancements in Software Defined Radio (SDR), which allows more intelligent, adaptive radio systems to be built, the wireless communications field has a number of opportunities to apply ML techniques. In this paper, a novel approach to demodulation using a Sequence to Sequence (Seq2Seq) model is proposed. This type of model is shown to work effectively with PSK data and also has a number of useful properties that are not present in other machine learning algorithms. A basic Seq2Seq implementation for BPSK and QPSK demodulation is presented in this paper, and learned properties such as Automatic Modulation Classification (AMC), and ability to adapt to different length input sequences, are demonstrated. This is an exciting new avenue of research that provides considerable potential for application in next generation 5G networks.**

I. Introduction

The progress made in ML and deep learning has enabled rapid growth in the fields of Computer Vision (CV), Natural Language Processing (NLP), and other disciplines. These fields have seen a shift towards data driven research and engineering, where novel state-of-the-art systems use ML algorithms and are replacing hand-crafted methods. With the advancements in SDR and Cognitive Radio (CR), and the opportunity of high performance processing in 5G cores, and edge computing networks, there is now excellent potential to exploit data driven implementations in the communications field.

Artificial Intelligence (AI) is a hot topic when talking about next generation wireless networks [1], and a reasoned demonstration of Deep Learning (DL) in the radio physical layer has been made in [2]. A great deal of research has already been conducted to bring AI into digital radio communications, particularly in the fields of Cognitive Radio (CR) for spectrum sensing and sharing [3], and Automatic Modulation Classification (AMC), where using ML supervised learning algorithms are well suited [4]. State of the art, AMC systems have historically used expert statistical features, such as higher order moments and cumulants [5],[6]. These methods require the designer of a radio receiver to have an in-depth knowledge of the signals of interest, and adding compatibility with new radio standards to such receivers may be very costly, requiring additional research and development.

DL is in an exciting phase currently, with the availability of powerful Graphical Processing Units (GPUs) and the opportunity to collect and store more data than ever before. The strongest quality of DL models is that they can usually be trained as end-to-end systems, learning the features and necessary transformations required to map inputs X to desired outputs Y. In [7], a deep Convolutional Neural Network (CNN) was trained on raw radio data to recognize modulation schemes at a comparable success rate to that of more traditional methods [4]. Recurrent Neural Networks (RNNs) have seen good success when applied to sequence data such as audio or text. In [8], the authors created an FM (Frequency Modulation) demodulator using an RNN, trained end-to-end, to reconstruct transmitted speech signals – this approach was demonstrated to outperform traditional FM demodulation at low SNR. A neural network of the same architecture has also been applied to wireless transmission traffic detection in [9], showing good results in classifying different transmission protocols.

Another advantage of using DL and deep neural networks for radio receivers is the reconfigurability aspect. Conventional receivers comprise many DSP modules and changing hardware can be costly; however a deployed neural network is, in essence, a set of weights. Models can be trained off-site and offline, and once a better performing algorithm is trained and validated it can be easily applied to an SDR receiver as a firmware update.

This paper presents a novel approach to radio physical layer modulation scheme classification and pulse shaped symbol recovery (which we will refer to as BPSK/QPSK demodulation), as a single module through the use of a Seq2Seq [10] model as illustrate in Figure 1. This type of architecture has been very successful in NLP for translation purposes because of its ability to accumulate context and, unlike the DL architectures mentioned previously, possesses the capability of outputting sequences, rather than single classification decisions. A Seq2Seq model consists of 2 RNNs connected in an encoder-decoder structure, and uses supervised learning to learn to map input sequences to output sequences.

The rest of this paper is laid out as follows: Section II outlines the approach to the design of the model. The details of how the model was trained are presented in Section III, which includes data formatting, and hyperparameters of the training algorithm used. Sections IV and V provide results and discussion, and conclusions, respectively.

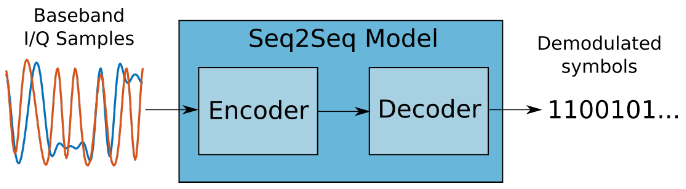

Fig. 1. High Level Overview of Seq2Seq Model

II. Background

A. Signal Model

The BPSK and QPSK symbols for training and testing in this work were all generated from a random distribution using MATLAB, and pulse shaped using a Raised Cosine Filter with a rolloff factor of $\alpha = 0.35$, oversampling factor $sps = 8$ and filter span in symbols $F_{span} = 8$. The resultant waveforms are then split into their real and imaginary components, and therefore each symbol period is represented by a 2x8 matrix. A training example consisting of 6 symbol periods is illustrated in Figure 2.

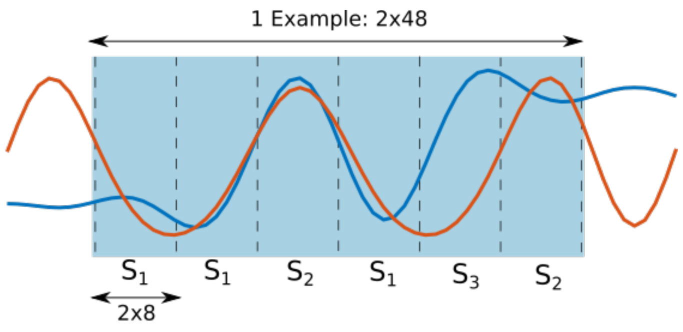

Fig. 2. Single Training Example Overview

In terms of ML, the act of symbol extraction can be considered a classification problem with $M+2$ classes, where M is the number of possible received symbols, and an additional 2 classes are set aside for the EOS (End Of Sequence) and padding (PAD) tokens. For BPSK and QPSK combined, M = 6. Each symbol $\{S_0, S_1, ...S_5\}$ and EOS, PAD tokens can be represented as one-hot encoded vectors such as:

$$\begin{bmatrix} S_0 \\ S_1 \\ ... \\ S_5 \\ EOS \\ PAD \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & ... & ... & ... & ... & & \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

An example sequence of 6 one-hot encoded labels to go along with the waveform shown in Figure 2 would be represented by a 7×8 matrix, if EOS is included.

B. Recurrent Neural Networks

RNNs are mainly used when working with sequential data, and are most prominently used in NLP and Audio processing. What differentiates them from CNNs and Multi Layer Perceptrons (MLPs) is the presence of a hidden state, which is updated after each time step, therefore acting as additional memory for storing context about the input sequence. RNN cells generally follow a basic update rule for hidden state, h, and output y at time step t, as given in (1) and (2) respectively, where the W terms represent the hidden state, input and output weight matrices respectively, and b are the biases.

$$h_t = tanh(W_h h_{t-1} + W_x x_t + b_h) \quad (1)$$
$$y_t = W_y h_t + b_y \quad (2)$$

Because RNNs have memory, the outputs produced at time step *t* do not only depend on the current input, but on all the past inputs as well. An unrolled RNN representation is shown in Figure 3.

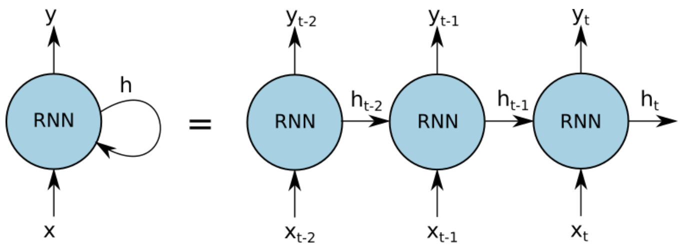

Fig. 3. Overview of a Recurrent Neural Network

Regular RNNs based solely on a *tanh* or sigmoid activation have somewhat fallen out of favor in recent developments, because they have poor memory retention in cases of long sequences due to vanishing/exploding gradients. The introduction of LSTM (Long Short-Term Memory) cells [11], which are another type of RNN, have addressed this problem — they support very long sequences, and allow even more advanced models to be developed. The update rules for an LSTM cell from [12] are defined as

$$i_t = \sigma_i(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \quad (3)$$
$$f_t = \sigma_f(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \quad (4)$$
$$y_t = \sigma_y(W_{xy}x_t + W_{hy}h_{t-1} + W_{cy}c_t + b_y) \quad (5)$$
$$c_t = f_t c_{t-1} + i_t tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \quad (6)$$
$$h_t = y_t tanh(c_t) \quad (7)$$

where σ is the sigmoid function, i and f are respectively the input and forget gates, c is the cell state, and all the W terms are the corresponding weight matrices. For the purposes of constructing the Encoder-Decoder network, the LSTM cell can be treated just like a regular RNN cell. DL libraries such as Tensorflow [13] will abstract the complexity such that the interface does not change, regardless of cell type.

III. Implementation

A. Encoder

The encoder is an RNN made up of 2 layers of stacked LSTM cells ($cell\_size = 128$) — this determines how large the network is and its memory capacity. The general structure of the encoder is shown in Fig 4.

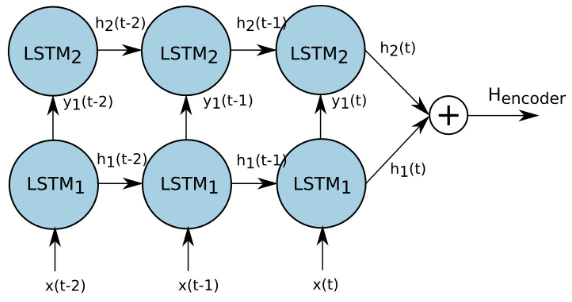

Fig. 4. Encoder Structure

At each time step t the encoder network is fed a sample input $x(t)$, composed of the real and imaginary components of the received baseband radio signal. The outputs of the encoder network are disregarded and left unused, as the only purpose of the network is to accumulate information about the incoming waveform. The hidden state of each stacked LSTM is updated after each step, and once the final sample is processed, the hidden states are concatenated to be passed off to the decoder. The passed hidden state should have all of the information necessary to determine the modulation scheme and the received bits encoded inside.

B. The Decoder

The decoder RNN cells must be of the same cell size as the encoder (to allow hidden state sharing), however the number of output steps need not correspond to the number of iterations required to encode the input waveform. Each predicted output is fed back into the decoder as the input of the next time step to assist in predicting the next symbol as shown in Figure 5.

The GO vector in this case is just the cell sized input set to all zeros. The network will continue outputting symbols until it finally outputs an EOS token.

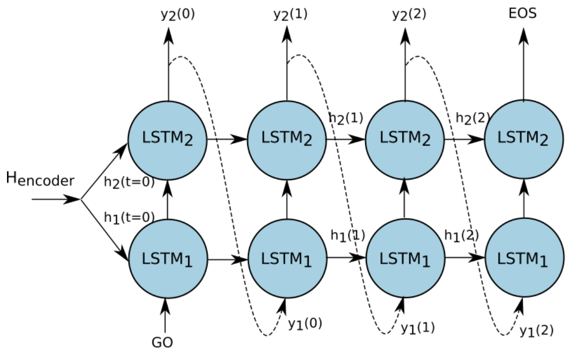

Fig. 5. Decoder Structure

Since this is a classification task, each output y_t goes through a softmax layer where a probability distribution over all possible symbols is generated. These are then used to predict symbols in a sequence.

$$\sigma(Y)_j = \frac{e^{y_j}}{\sum_{k=1}^{K} e^{y_k}} \tag{8}$$

where σ is the normalized output over all possible symbols for that time step. The j index denotes the output neuron number, while K is the number of outputs, and Y refers to the activation values coming from the fully connected layer neurons.

C. Training

For the training set, a total of $N_{bpsk}, N_{qpsk} = 8192$ sequences of pulse shaped symbols were generated for each SNR level at 0dB, 5dB, 10dB and 15dB. The training set is then randomly shuffled and split into mini-batches of 512 sequence-label pairs per batch.

The model was trained in Tensorflow using the Adam [14] optimizer, with a learning rate of $\alpha = 0.5 \times 10^{-3}$. The loss function selected for this task was Categorical Cross-Entropy, a popular choice for classification problems, defined as:

$$L_s(y, \hat{y}) = -\sum_i y_i \log \hat{y}_i \tag{9}$$

Where L_s is the loss per classified symbol in a sequence, while y and $\hat{y}$ are the ground truth labels and predictions created by the model, respectively. The further the predicted output is from the desired outcome, the higher the loss value. To obtain the total loss of a single classified sequence, an average is taken of all the individual losses.

IV. Results

The neural network was trained with input sequence lengths of N = 3, 5, 7 and 10 symbols in order to ensure that increasing the input window length still allowed it to reasonably fit the data. As seen in Figure 6, sequence lengths of increasing number of symbols can fit the training set, albeit at the expense of longer training times to convergence as sequence length increases.

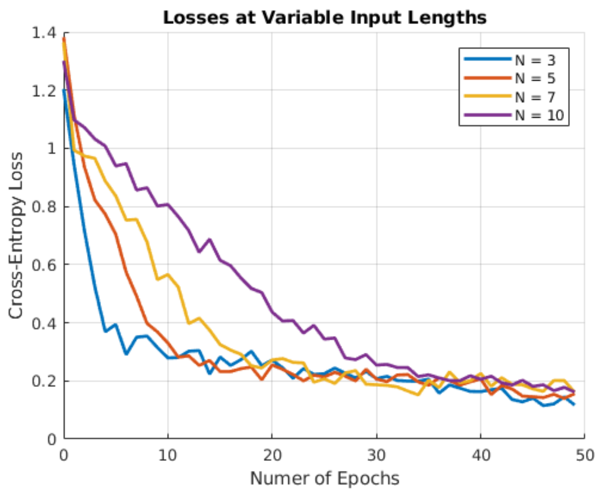

Fig. 6. Losses at Variable Input Lengths

TABLE I
Symbol Prediction Accuracies

Mod	N	Average Acc	Mod	N	Average Acc
BPSK	3	93.8%	QPSK	3	82.1%
	5	94.7%		5	86.2%
	7	95.7%		7	87.0%
	10	96.3%		10	87.5%

TABLE II
Modulation Classification Accuracies

Mod	N	SNR = 0dB	SNR = 5dB	SNR = 10dB	Average
BPSK	3	80.3%	95.6%	99.9%	95.2%
	5	78.9%	98.1%	99.9%	96.1%
	7	83.9%	98.9%	100%	97.3%
	10	87.9%	99.4%	100%	98.0%
QPSK	3	57.9%	86.9%	98.6%	88.2%
	5	74.6%	92.3%	99.1%	93.3%
	7	75.4%	94.9%	99.9%	94.4%
	10	79.4%	97.5%	100%	96.2%

The trained model is then evaluated for SNR values between 0 and 15 dB using a held out test dataset, with 16384 BPSK and QPSK examples per SNR level. The resultant symbol accuracies for each input sequence size can be seen in Table I. Another useful metric for this type of model is the actual modulation classification accuracy. This was obtained by disregarding individual symbol errors, and instead checking whether the predicted symbol belongs to that modulation scheme class. The modulation scheme classification accuracies are summarized in Table II.

Curves for input sequence lengths of 10 symbols compared to an ideally matched filter receiver can be seen in Figure 7.

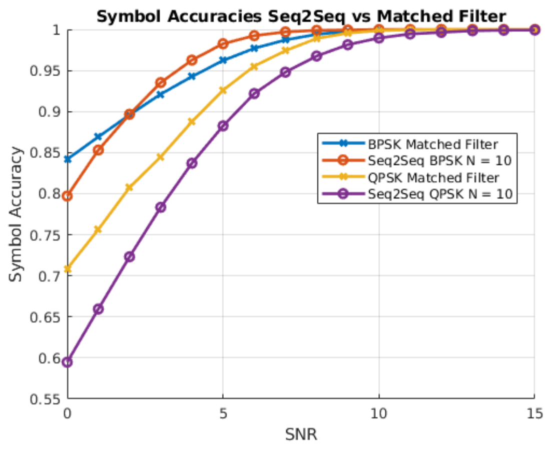

Fig. 7. Accuracy for BPSK and QPSK with 10 Symbol Sequences

As the graph suggests, at low SNR levels the Seq2Seq model does not match the performance of a matched filter, which can be explained by low overall modulation scheme accuracy rates at low SNR in Table II.

V. Conclusions

An implementation of a Seq2Seq model has been applied to baseband BPSK and QPSK modulation schemes and tested at various sequence lengths with an AWGN channel. Our preliminary results show that this network can perform incredibly well at SNR = 12dB and above, and that demodulation can indeed be accomplished as an end-to-end learning solution capable of outputting sequences of bits.

It was also shown that, as the input sequence length was increased, the ability of the Seq2Seq model to predict modulation scheme steadily improved. This result is consistent with previous AMC research, where the availability of more samples results in better classification rates.

Our future work will include architectural improvements, scaling up and adding more modulation schemes, as well as introducing other perturbations such as phase and timing errors. Demonstrating that this type of model can work with communications data is an exciting step towards future DL-based wireless receiver technologies.

VI. Acknowledgement

This work was co-funded by MathWorks DCRG grant on 'Dynamic Spectrum Access for 5G Communications 2016 - 2018.

References

[1] C. Jiang, H. Zhang, Y. Ren, Z. Han, K. C. Chen and L. Hanzo, "Machine Learning Paradigms for Next-Generation Wireless Networks," in IEEE Wireless Communications, vol. 24, no. 2, pp. 98-105, April 2017
[2] O'Shea, T. J., & Hoydis, J, "An Introduction to Deep Learning for the Physical Layer" 2017, arXiv:1702.00832 [Online]. Available: http://arxiv.org/abs/1702.00832
[3] M. Bkassiny, Y. Li and S. K. Jayaweera, "A Survey on Machine-Learning Techniques in Cognitive Radios," in IEEE Communications Surveys & Tutorials, vol. 15, no. 3, pp. 1136-1159, Third Quarter 2013.
[4] O. A. Dobre, A. Abdi, Y. Bar-Ness and W. Su, "Survey of automatic modulation classification techniques: classical approaches and new trends," in IET Communications, vol. 1, no. 2, pp. 137-156, April 2007. doi: 10.1049/iet-com:20050176
[5] E. E. Azzouz and A. K. Nandi, "Procedure for automatic recognition of analogue and digital modulations," in IEE Proceedings - Communications, vol. 143, no. 5, pp. 259-266, Oct 1996. doi: 10.1049/ip-com:19960752
[6] A. Swami and B. M. Sadler, "Hierarchical digital modulation classification using cumulants," in IEEE Transactions on Communications, vol. 48, no. 3, pp. 416-429, Mar 2000. doi: 10.1109/26.837045
[7] T. J. O'Shea and J. Corgan. "Convolutional Radio Modulation Recognition Networks," CoRR, vol. abs/1602.04105, 2016. [Online]. Available: http://arxiv.org/abs/1602.04105
[8] Elbaz, D., & Zibulevsky, M., "End to End Deep Neural Network Frequency Demodulation of Speech Signals", 2017, arXiv:1704.02046 [Online]. Available: http://arxiv.org/abs/1704.02046
[9] T. J. O'Shea, S. Hitefield and J. Corgan, "End-to-end radio traffic sequence recognition with recurrent neural networks," 2016 IEEE Global Conference on Signal and Information Processing (GlobalSIP), Washington, DC, 2016, pp. 277-281.
[10] Sutskever, I., Vinyals, O., & Le, Q. V., "Sequence to Sequence Learning with Neural Networks" 2014, arXiv:1409.3215 [Online]. Available: http://arxiv.org/abs/1409.3215
[11] Sepp Hochreiter and Jrgen Schmidhuber. 1997. "Long Short-Term Memory", Neural Comput. 9, 8 (November 1997), 1735-1780. DOI=http://dx.doi.org/10.1162/neco.1997.9.8.1735
[12] Graves, A. "Generating Sequences With Recurrent Neural Networks", 2013, arXiv:1308.0850 [Online]. Available: http://arxiv.org/abs/1308.0850
[13] M. Abadi, A. Agarwal, et al., "Tensorflow: large-scale machine learning on heterogeneous systems," 2015. Software from tensorflow.org.
[14] Diederik P. Kingma, Jimmy Ba, "Adam: A Method for Stochastic Optimization," CoRR, vol. abs/1412.6980, 2014. [Online]. Available: https://arxiv.org/abs/1412.6980