Channel Tracking and Prediction with Deep Temporal Convolutional Networks

Arick G. Grootveld, Vlad I. Bugayev, Andrew G. Klein
Department of Engineering and Design, Western Washington University
Bellingham, WA 98225
{grootva, bugayev, andy.klein}@wwu.edu

Kirty P. Vedula, D. Richard Brown III
Department of ECE, Worcester Polytechnic Inst.
Worcester, MA 01609
{kpvedula, drb}@wpi.edu

***Abstract*— This paper proposes a temporal convolutional network (TCN)-based neural structure for channel tracking and prediction and compares the performance of the Kalman filter (KF) and TCN in three settings: (i) a simulated setting where the channel obeys a linear autoregressive (AR) model with additive white Gaussian noise where the Kalman filter is known to be optimal, (ii) a simulated setting where the ideal assumptions required for Kalman filter optimality are not satisfied due to the mismatch between the presumed linear time-varying AR model and the true model and (iii), a simulated setting where the AR channel follows the Gilbert-Elliott model and exhibits stable and unstable state space conditions. We also compare our results in the three scenarios with a least squares approach and the Ricatti equation which provides the asymptotic limits. Numerical mean squared error (MSE) results are presented to demonstrate the accuracy achieved by the machine learning approach and the loss of performance experienced by the KF as a function of the amount of model mismatch. The results demonstrate that the machine learning approach achieves performance close to optimal irrespective of the amount of model mismatch.**

TABLE OF CONTENTS

1. INTRODUCTION

Channel tracking and prediction are critical tasks in communication systems with mobility. The ability to accurately track and predict channels is necessary to enable many aspects of modern communication systems including coherent demodulation, rate and power control, and beamforming [1].

Under an ideal AR channel model with linear dynamics, Gaussian noise and known noise covariance matrices, the classical Kalman filter (KF) is the optimal estimator and predictor [2]. In non-ideal situations, like for example, when the underlying dynamical model does not match the dynamics of the actual channel, the KF is no longer optimal.

We propose a better solution using recent advances in deep learning, specifically using sequence prediction algorithms in the task of channel tracking and prediction. While recurrent neural networks (RNNs) and long short-term memory (LSTM) have historically been the most popular choice for sequence prediction – and, indeed, prior work has demonstrated promising results for their use in channel tracking [3] [4] – recent results in machine learning have shown that a new architecture called temporal convolution networks (TCNs) can often outperform LSTM-based RNNs in sequence prediction problems [5]. In addition to their performance advantage in a wide range of sequence prediction problems, TCNs have a number of implementation advantages over RNNs in that they admit a more parallelizable implementation and have reduced memory requirement for training. TCNs also have several structural advantages including an adjustable receptive field size and avoidance of the problem of exploding/vanishing gradients known to plague RNNs.

To show statistical significance in support of our results, we designed three separate test cases for the TCN: one case where the real channel model matches the assumed, another case where the real model deviates from the assumed channel model and a final case in which the real channel oscillates between stable and borderline unstable states.

2. CHANNEL MODEL

We assume a Gauss-Markov channel model, otherwise known as an auto-regressive (AR) model. Time varying transmission channels are often modeled in this way due to the causal and Gaussian white noise influenced system dynamics. Previous related works have supplied the motivation and justification for us to choose this model [1] [6].

In general, a noisy observation $z[k]$ of an Nth order autoregressive (AR) process $x[k]$ is described by the equations

$$\begin{aligned} x[k] &= \left(\sum_{n=1}^{N} f_n x[k-n]\right) + v[k] \\ z[k] &= x[k] + w[k] \end{aligned}$$

978-1-7281-2734-7/20/$31.00

where $v[k]$ and $w[k]$ are both Gaussian, zero-mean noise with variances σ_v^2 and σ_w^2, assumed to be uncorrelated.

Throughout this paper, we constrain the process order to be AR-2 where the process (or state) $x[k]$ and noisy observation $z[k]$ are given by

$$\begin{aligned} x[k] &= f_1 x[k-1] + f_2 x[k-2] + v[k] \\ z[k] &= x[k] + w[k] \end{aligned}$$

In state-space form, this becomes:

$$\begin{aligned} \boldsymbol{x}[k] &= \boldsymbol{F}\boldsymbol{x}[k-1] + \boldsymbol{v}[k] \\ z[k] &= \boldsymbol{H}\boldsymbol{x}[k] + w[k] \end{aligned}$$

where

$$\begin{aligned} \boldsymbol{x}[k] &= \begin{bmatrix} x[k] \\ x[k-1] \end{bmatrix} \\ \boldsymbol{v}[k] &= \begin{bmatrix} v[k] \\ 0 \end{bmatrix} \\ \boldsymbol{F} &= \begin{bmatrix} f_1 & f_2 \\ 1 & 0 \end{bmatrix} \\ \boldsymbol{H} &= \begin{bmatrix} 1 & 0 \end{bmatrix} \end{aligned}$$

The AR process states are complex numbers which contain information about the phase and gain modulation inherent to the channel itself. Under the assumption that the transmission frequency band is very narrow in spectrum, we can model the channel as a singular complex number which alters both the phase and magnitude of signals passed through it. We use a neural network to extract the nonlinear and weighted relationships between samples such that we can make better predictions and estimations of the state than we could otherwise with linear estimators like the least squares and Kalman filter approaches.

3. Data Generation

We designed a data generation script which outputs m AR process sequences of length n. This data is used for training and testing the least squares (LS), Kalman filter (KF) and TCN state prediction and estimation systems. Separate training and testing subsets of data are created to ensure validity in the numerical results for the non-mismatched and mismatched AR parameter cases as well as the Gilbert-Elliott case.

The generated data follows this format:

$$AR = \begin{bmatrix} P_{10} & P_{11} & \dots & P_{1n} \\ P_{20} & P_{21} & \dots & P_{2n} \\ \vdots & & \ddots & \\ P_{m0} & P_{m1} & \dots & P_{mn} \end{bmatrix}$$

The data within each row varies in type, according to the input specifications of each test case. All of the generated data is shared between the least squares, Kalman filter and TCN so the MSE results are generated by each channel tracking system operating on the same datasets. We chose the length of the data matrix rows (n) to be 15 samples to ensure that the Kalman Filter was able to converge to its steady state MMSE for all scenarios

When generating an AR-n process it is important to take into account the matrix that is formed by the AR coefficients. In our case if the pair of values that form the AR Coefficients cause the F matrix to have eigenvalues that are greater than 1, the system will become unstable. Figure 1 displays the values which provide stability and instability, as well as the centroid which corresponds to the our choosen coefficients, f_1 = 0.5, f_2 = -0.4.

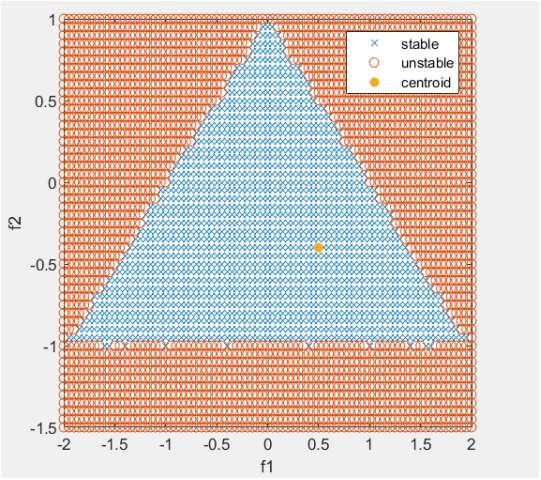

Figure 1. AR Coefficient Stability Region

Case 1: Static AR Processes

In the static AR coefficient case, each row of the

data matrix contains an AR process generated with the same F matrix. Each row begins with a randomly generated sample $z[-1]$ where $z[-1] \sim \mathcal{N}(0,1)$ Each row contains a different AR process but with the same AR coefficients. This format ensures that the TCN and KF both received a wide variety of AR processes that share the same coefficients but differ in starting value. In this case, both the training and testing data formats are the same.

Case 2: Varied AR Processes

In the varied AR process case, each row of the data matrix contains a different AR process with differing coefficients, meaning that each row of the data matrix is associated with a unique F matrix. The processes begin with the same conditions as in the first case: $z[-1] \sim \mathcal{N}(0,1)$. The variations in the F matrices are determined by a normal, Gaussian distribution about the chosen coefficient mean values, such that $f_1 \sim \mathcal{N}(0.5,1)$ and $f_2 \sim \mathcal{N}(-0.4,1)$. The reasoning for introducing this variation is that during the training phase, we wish to expose the model to as many AR processes as possible such that we obtain a network which is better capable of generalizing and extrapolating to other processes.

For the mismatched data generation case, we obtain the heatmap of chosen coefficient values shown in Figure 2. We

discard any values which cause the eigenvalues of the F matrix to be larger than one, therefore the actual mean value of the coefficients deviates from what we use to generate the overall dataset.

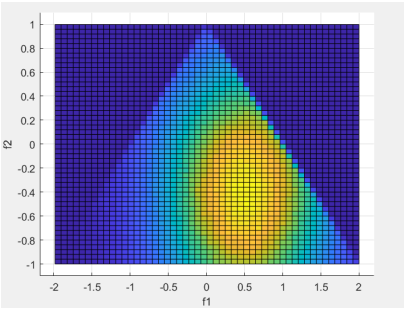

Figure 2. AR Coefficient Heatmap

In testing, we wish to acquire better averages for the resulting MSE values, therefore the AR process data matrices now contain AR processes in each row that share the same F matrix. With each iteration of testing, a new AR process data matrix is generated, with a different, normally distributed F matrix for the whole set of data.

Case 3: Gilbert-Elliott Model

To construct the data for the burst noise Gilbert-Elliott channel of the form shown in Figure 3, we utilized a Markov chain for AR sequence generation. For every new sample generated, the Markov chain has a probability of transitioning from state to state (P_{BG}, P_{GB}) or staying where it is currently (P_{BB}, P_{GG}). The "bad" state here corresponds to a process generation scheme where the AR coefficients produce eigenvalues that are close to one, edging on the border between stability and instability while the "good" state is any other coefficient pair that resides close to the center of the triangle in Figure 1. For the "bad" set of coefficients, we chose the lower right-hand corner of the triangle in Figure 1. For the "good" case, we chose f_1 = 0.5, f_2 = -0.4.

The complete AR sequence will have both transition regions with good and bad coefficients as well as consistent segments. We wish to challenge the linear estimators and neural network to recognize when the changes occur, as the optimum solution to this type of channel model would be two Kalman filters that can be switched between, depending on the current state. Upon each testing and training iteration, the generated sequence is formed into the following Toeplitz matrix structure:

$$\mathbf{X} = \begin{bmatrix} x[k-1] & \dots & x[k-N-1] \\ x[k-2] & \dots & x[k-N-2] \\ \vdots & \ddots & \\ x[k-M-1] & \dots & x[k-M-N-1] \end{bmatrix}$$

Then, this data is processed by each of the channel tracking systems. In the testing phase, we processed the AR sequence generated using the combination of good and bad coefficients and also obtained results for testing on both sets separately.

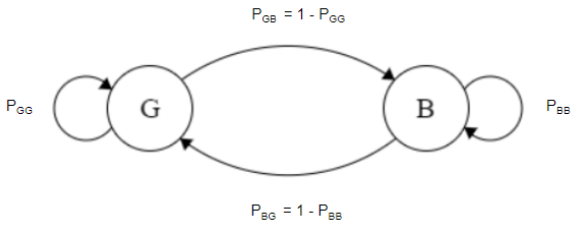

Figure 3. Gilbert-Elliott Probabilistic Markov Chain

4. Linear Estimators

The most common approaches to accurately tracking the channel described in Section 2 use linear estimators such as the least squares solution or the Kalman filter. The LS solution is the most straightforward of the two and simply tries to find a line of best fit to the training data which is then used in further evaluations for test data. The KF on the other hand requires knowledge of the channel statistics such that a probabilistic estimation and prediction can be made of the current and next states.

Riccati Equation

The Riccati equation allows us to have a lower bound for the best possible performance in channel estimation we could possibly have with a model mismatch, assuming access to an infinite dataset. In this specific case, this equation will always converge to a lower MSE than any other channel estimation method we are using (LS, KF, TCN). In the matched model case, the Riccati will simply verify that the KF is working properly, as they both converge to and provide the same solution. The Riccati in the discrete time domain is implemented with the following equation:

$$X = A^{\mathrm{T}} X F - (F^{\mathrm{T}} X H)(R + H^{\mathrm{T}} X H)^{-1}(H^{\mathrm{T}} X F) + Q$$

Least Squares

Utilizing the least squares algorithm, we obtained filter coefficients which transform the measured state matrix ($\mathbf{Z}$) into the best prediction or estimation ($\hat{x}$) that we can make given a sequence history of N and a filter order of M. The prediction/estimation and least squares coefficient solutions (a_{ls}, b_{ls}) are given by the following equations:

$$\begin{aligned} \hat{x_{est}} &= \mathbf{Z} a_{ls} \\ \hat{x_{pred}} &= \mathbf{Z} b_{ls} \\ a_{ls} &= (\mathbf{Z}^{\mathrm{T}}\mathbf{Z})^{-1}\mathbf{Z}^{\mathrm{T}} x_{est} \\ b_{ls} &= (\mathbf{Z}^{\mathrm{T}}\mathbf{Z})^{-1}\mathbf{Z}^{\mathrm{T}} x_{pred} \end{aligned}$$

$$\begin{aligned} x &= [x[k] \dots x[k-M]]^{\top} \\ \hat{x_{pred}} &= [\hat{x}[k-1] \dots \hat{x}[k-M-1]]^{\top} \\ \hat{x_{est}} &= [\hat{x}[k] \quad \hat{x}[k-1] \quad \dots \quad \hat{x}[k-M]]^{\top} \\ a_{ls} &= [a_0 \quad a_1 \quad \dots \quad a_N]^{\top} \\ b_{ls} &= [b_0 \quad b_1 \quad \dots \quad b_N]^{\top} \end{aligned}$$

$$\mathbf{Z} = \begin{bmatrix} z[k-1] & \dots & z[k-N-1] \\ z[k-2] & \dots & z[k-N-2] \\ \vdots & \ddots & \\ z[k-M-1] & \dots & z[k-M-N-1] \end{bmatrix}$$

The least squares filter coefficients are computed once using a training data sequence consisting of many segments of either one AR process or many, depending on the case being examined. Both the mismatched and matched model cases are evaluated with training occurring on large collections of AR processes and testing on individual, longer sequences. The Gilbert-Elliott case requires the LS to be trained on a Toeplitz matrix input data structure, as is mentioned in Section 3.

Kalman Filter

We wish to estimate the current state as well as predict the future state of the AR process using the Kalman filter to compare performance of the TCN to a standard channel tracking system. To realize the KF in this situation, we used the following state update equations [2]:

$$\begin{aligned} \hat{x}[n|n-1] &= \mathbf{F}\hat{x}[n-1|n-1] \\ \mathbf{M}[n|n-1] &= \mathbf{F}\mathbf{M}[n-1|n-1]\mathbf{F}^{\mathrm{T}} + \mathbf{Q} \\ \mathbf{K}[n] &= \frac{\mathbf{M}[n|n-1]\mathbf{H}}{\sigma_w^2 + \mathbf{H}^{\mathrm{T}}\mathbf{M}[n|n-1]\mathbf{H}} \\ \hat{x}[n|n] &= \hat{x}[n|n-1] + \mathbf{K}[n](z[n] - \mathbf{H}^{\mathrm{T}}\hat{x}[n|n-1]) \\ \mathbf{M}[n|n] &= (\mathbf{I} - \mathbf{K}[n]\mathbf{H}^{\mathrm{T}})\mathbf{M}[n|n-1] \end{aligned}$$

Notation	Description
$\hat{x}[n\|n-1]$	State prediction
$\hat{x}[n-1\|n-1]$	Correction value
$\mathbf{M}[n\|n-1]$	Minimum prediction MSE
$\mathbf{M}[n-1\|n-1]$	Minimum estimation MSE
Q	Process noise
$\mathbf{K}[n]$	Kalman gain
σ_w^2	Measurement noise covariance

Generally, if the Kalman filter has perfect knowledge of the real AR coefficients, the MSE of this system will approach the Riccati equation MMSE. If the Kalman filter has an error in the assumed channel parameters however, it will result in a worse MSE than the Riccati. In our case, we tested both circumstances; when the Kalman filter has perfect knowledge of the coefficients and also when there is a slight deviation.

5. Temporal Convolutional Network

The temporal convolutional network (TCN) is the neural network architecture we chose to design and test against the least squares and Kalman filter solutions. The TCN has proven to be more successful in sequence identification problems than recurrent neural networks (RNN) and long-short term memory neural networks (LSTM). The structure and general design of this neural network was provided by [5], which we used and altered for channel estimation specific inputs and outputs.

The TCN structure is essentially a one dimensional, fully connected network with causal convolutions. The main differences between a traditional convolutional neural network and the TCN are the diluted causal convolutions between network layers and the ability to specify a desired output sequence length. To accomplish this, the network contains hidden layers which are all of the same length and uses zero padding to keep the dilutions feasible and maintain the output size specification. Diluted convolutions increase the receptive field of each convolutional layer, providing the network the ability to account for sequence elements that are further back in the input history. Figure 4 demonstrates diluted convolution with a kernel size (K) of 3 and two stages of dilution (d).

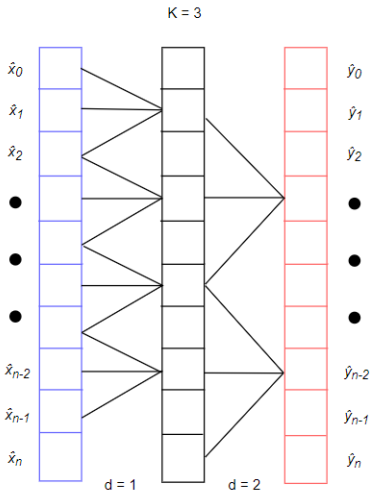

Figure 4. Diluted Convolution Example

The TCN consists of multiple "residual blocks" which act as fundamental construction elements in this architecture. One instance of this network may consist of many residual blocks, all daisy-chaining in one long sequence. These residual blocks consist of two layers of diluted, causal convolutions and a non-linearity (ReLU) on the output of each layer. The residual blocks contain an identity mapping which aids in training for slight variations around the input sequence by explicitly weighting the output with a scaled version of the input before any convolutions took place.

To optimize the TCN for channel prediction and estimation tasks, we implemented some common functions that many neural networks share, namely: learning rate scheduling, early stopping and dropout. Adding a learning rate scheduling system to the network improved the prediction and estimation accuracy while early stopping and dropout were included as countermeasures for model over-fitting.

We used epochs in the training phase to drastically reduce

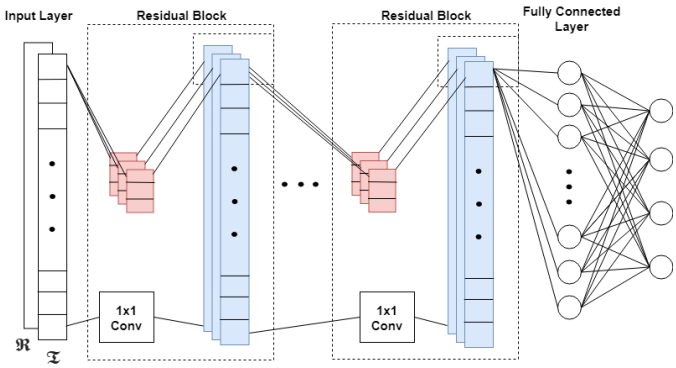

Figure 5. Temporal Convolutional Network Architecture

the amount of time it takes to generate the sequence data and therefore, train the model. Although we can generate an unlimited amount of data to train the network, data generation takes a very long time and significantly cuts into the time available for the training process. By using epochs and overfitting counter measures, we can attain similar results as we would with longer datasets.

Due to the complex nature of the channel values, the imaginary and real components had to be split into separate numbers in order to effectively train the network. This splitting into real and imaginary components has already been done previously with favorable results [7]. The loss function for this network is the mean-squared error (MSE), and for complex values this is calculated in the following way:

$$MSE = E(|x - \hat{x}|^2)$$

Where x and $\hat{x}$ are both complex valued states (actual and computed states, respectively).

6. Results

Case 1: Static AR Processes

	MMSE	Least Squares	Kalman	TCN
Estimation	ay	doe	ay	doe
Prediction	aye	doe	ayez	doez

Case 2: Varied AR Processes

Figures 6 and 7 display the MSE results of all three channel tracking methods in the varied AR processes case. Figure 7 is a magnified version of Figure 6, specifically in the region where the Riccati MSE ranges from 0.1 to 0.13.

The MMSE line in the both figures corresponds to the Riccati equation solution, which provides the asymptotic lower bound for the best possible performance. As can be seen from both figures, all channel tracking methods result in a MSE value that resides above this line.

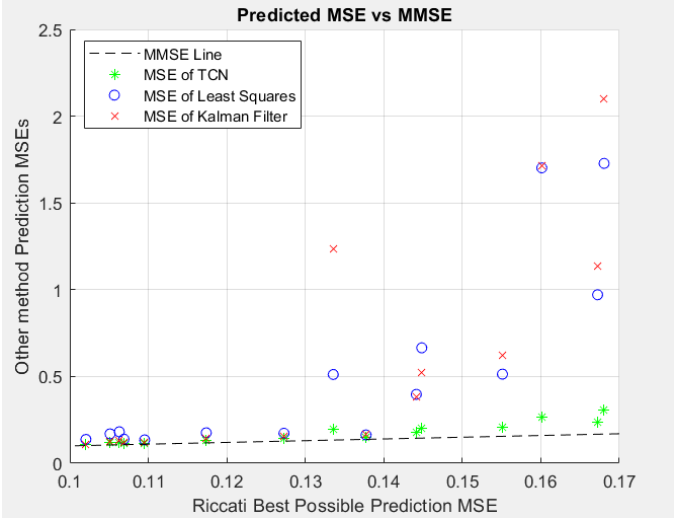

Figure 6. Mismatch Scenario Prediction MSE Graph

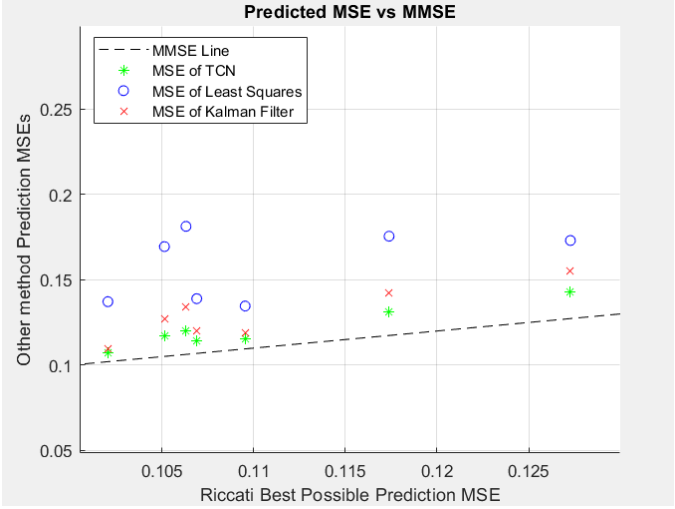

Figure 7. Mismatch Scenario Prediction MSE Graph

The results show that in all cases, the TCN outperforms both the least squares and KF in the task of channel prediction for an AR-2 channel model. Irrespective of the amount of model mismatch, the network has little trouble in compensating and adjusting while the least squares and KF solutions become much worse as the Riccati MSE increases (as the amount of model mismatch increases).

Case 3: Gilbert-Elliott Model

	MMSE	Least Squares	Kalman	TCN
Estimation	ay	doe	ay	doe
Prediction	aye	doe	ayez	doez

7. Conclusion

This is the conclusion.

All source code for this experiment is available at — `https://github.com/aspectlab/deepchannel`

Acknowledgments

This work was supported by a Research Experience for Undergraduates (REU) Supplement to National Science Foundation award CNS-1836695.

References

[1] J. Kang, C. Chun, I. Kim, and D. I. Kim, "Channel tracking for wireless energy transfer: A deep recurrent neural network approach," *arXiv:1812.02986*, 2018.

[2] S. M. Kay, *Fundamentals of statistical signal processing*. Prentice Hall PTR, 1993.

[3] K. Greff, R. K. Srivastava, J. Koutnk, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *IEEE Trans. Neural Netw.*, vol. 28, no. 10, pp. 2222–2232, Oct. 2017.

[4] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[5] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv:1803.01271*, 2018.

[6] K. E. Baddour and N. C. Beaulieu, "Autoregressive models for fading channel simulation," in *GLOBECOM'01. IEEE Global Telecommunications Conference (Cat. No. 01CH37270)*, vol. 2. IEEE, 2001, pp. 1187–1192.

[7] N. Taşpinar and M. N. Seyman, "Back propagation neural network approach for channel estimation in ofdm system," pp. 265–268, 2010.

BIOGRAPHY

Arick Grootveld *is currently pursuing a B.S. in electrical engineering from WWU and expects to graduate in June 2020. His current research activities include... Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.*

Vlad Bugayev *is currently pursuing a B.S. in electrical engineering from WWU and expects to graduate in June 2020. His current research activities include ... Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.*

Andrew G. Klein *received the B.S. degree from Cornell University, Ithaca, NY, USA, the M.S. degree from the University of California, Berkeley, CA, USA, and the Ph.D. degree from Cornell University, all in electrical engineering. Previously, he was an Assistant Professor with the Worcester Polytechnic Institute, Worcester, MA, USA, from 2007 to 2014, and he was a Post-Doctoral Researcher with Supélec/LSS, Paris, France, from 2006 to 2007. He joined the Department of Engineering and Design, Western Washington University, Bellingham, WA, USA, in 2014, where he is currently a Professor.*

D. Richard Brown III *received the B.S. and M.S. degrees from the University of Connecticut in 1992 and 1996, respectively, and the Ph.D. degree from Cornell University in 2000, all in electrical engineering. From 1992 to 1997, he was with General Electric Electrical Distribution and Control. He was a Faculty Member with the Worcester Polytechnic Institute, Worcester, MA, USA, in 2000. He was a Visiting Associate Professor with Princeton University from 2007 to 2008. From 2016 through 2018, he was with the Computing and Communication Foundations Division, National Science Foundation, as a Program Director.*

Kirty Vedula *Rutgers University, MS in Electrical and Computer Engineering, 2014 Amrita University, B Tech in Electronics and Communications Engineering, 2012. I joined the PhD in Electrical and Computer Engineering program in 2015. Prior to this, I completed my Master's degree at Rutgers University. I worked at Mathworks (Framingham, MA) and Visible Energy(Palo Alto, CA) earlier. My research interests are in signal processing, wireless communication systems, wireless power transfer and hardware security.*