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 TCN-based (temporal convolutional network) structure for channel tracking and prediction and compares channel tracking performance of the Kalman filter and TCN-based channel tracking approach in three settings: (i) a simulated setting where the Kalman filter is known to be optimal, (ii) a simulated setting where the ideal assumptions required for Kalman filter optimality are broken and (iii) a simulated setting where the channel follows the Gilbert-Elliott model. First, under the assumption that the channel obeys a simple linear, autoregressive (AR) model with additive white Gaussian noise, the performance of the proposed TCN approach is compared with a classical Kalman filter, and the performance gap in using the necessarily suboptimal TCN approach is quantified. Next, the idealized assumptions of the channel model are relaxed so that there is mismatch between the presumed linear time-varying AR model and the true model, and results are again compared. Finally, a Gilbert-Elliott channel model is assumed where the channel oscillates between good and bad states and tracking performance is again evaluated. Results from using the least squares approach in all three scenarios as well as the Riccati equation solution are also compared.

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 where, for example, the dynamics of the time-varying channel are linear, the noise is

Gaussian, and the noise covariance matrices are known, the classical Kalman filter is known to be the optimal estimator and predictor. When these idealized assumptions break down, however, or when the underlying dynamical model does not match the dynamics of the actual channel, the Kalman filter is no longer optimal and channel estimation performance can suffer significantly.

This paper applies very recent advances in machine learning for so-called "sequence prediction" problems to 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 [2] [3] – 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 [4]. 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; in addition, TCNs have several structural advantages including an adjustable receptive field size and avoidance of the problem of exploding/vanishing gradients known to plague RNNs.

2. CHANNEL MODEL

There are many different channel models, one of the most common being the Gauss-Markov channel model, otherwise known as the auto-regressive (AR) model. Due to the inherent Gaussian nature of the noise present in the space and air we are surrounded by as well as the causal state dependency on current states of the transmission medium from previous states, it makes sense to assume that the channel can be modeled this way. This is exactly what an AR process is, and previous bodies of work have supplied the motivation and justification for us to choose this model [1] [5].

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

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

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. The idea is to 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

Using Python computational packages, we designed a data generation script which will output a variable number of AR-n process sequences of a defined length. 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 both the mismatched and non-mismatched AR parameter cases.

The format of the data that is generated is as follows:

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

Where the first index for each element corresponds to the process ID and the second index is the sample number in that specific AR process. Each row may contain a different

process generated from different AR coefficients or from the same pair of AR coefficients, depending on if that data is being used to test or train the TCN model in the mismatched case (in the matched case, the AR coefficients are static). The reasoning for 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. In testing, we wish to run the network one by one on longer processes so that we can compare to the KF and LS solutions when they converge on the same datasets.

An important consideration to take when generating an AR-n process are the values of the AR coefficients themselves. If the pair of these values cause the F matrix to have eigenvalues that are greater than 1, the system will become unstable. Figure 4 displays the values which provide stability and instability, as well as the centroid which corresponds to the currently chosen coefficients.
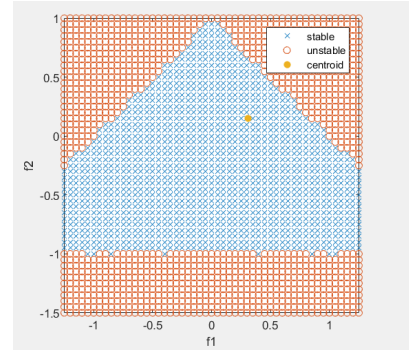

Figure 1. AR Coefficient Stability Region

For the mismatched data generation case where each row of the AR process matrix contains a different AR process generated by a normally distributed deviation from the chosen AR coefficient means, 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.
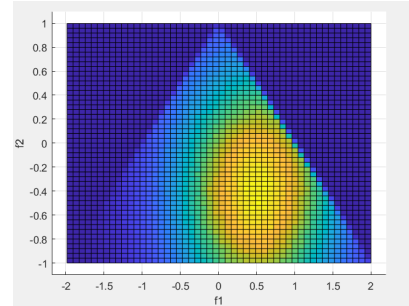

Figure 2. AR Coefficient Heatmap

To construct the data for the Gilbert-Elliott channel of the form shown in Figure 3, we designed a probability based state machine for AR sequence generation. For every new sample generated, the state machine 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. The complete AR sequence will have both transition regions with good and bad coefficients as well as consistent segments. The idea here is 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.
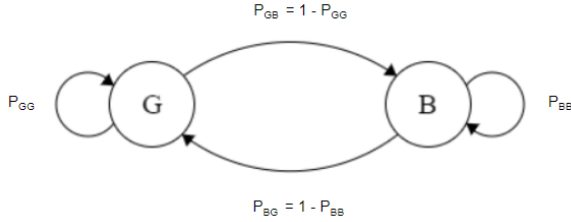

Figure 3. Gilbert-Elliott Probabilistic State Machine

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

The least squares solution is perhaps the simplest and most straight-forward approach to channel tracking. Using this methodology, we obtained least squares filter coefficients which transform the measured state matrix (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:

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

$$
x = \begin{bmatrix} x[k] \\ x[k-1] \\ \vdots \\ x[k-M] \end{bmatrix}
$$

$$
\hat{x_{pred}} = \begin{bmatrix} \hat{x}[k-1] \\ \hat{x}[k-2] \\ \vdots \\ \hat{x}[k-M-1] \end{bmatrix}
$$

$$
\hat{x_{est}} = \begin{bmatrix} \hat{x}[k] \\ \hat{x}[k-1] \\ \vdots \\ \hat{x}[k-M] \end{bmatrix}
$$

$$
a_{ls} = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_N \end{bmatrix}
$$

$$
b_{ls} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_N \end{bmatrix}
$$

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

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 model being used. Both the mismatched and matched model cases are evaluated, with training occurring on large collections of AR processes and testing on individual, longer sequences.

Kalman Filter

Given knowledge of the channel parameters and statistics via data collection, we can implement the Kalman filter approach to channel tracking. We wish to estimate the current state as well as predict the future state of the AR process. To realize the KF in this situation, we used the following state update equations:

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

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 with longer datasets corresponding to closer matching between the two values. 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 previously created by another research team [4] 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. Convolution dilution happens in this network to 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.

The TCN consists of multiple "residual blocks" which act as fundamental construction elements in this architecture. One
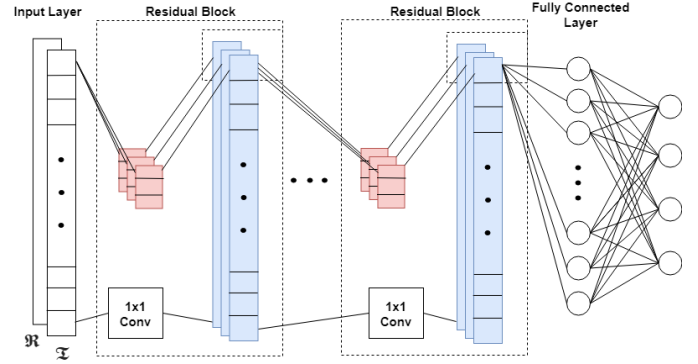

Figure 4. Temporal Convolutional Network Architecture

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.

Due to the complex nature of the channel values and lack of PyTorch support for complex valued neural networks, the imaginary and real components had to be split into separate floating point numbers in order to effectively train the network. This splitting into real and imaginary components has already been done by previous research teams with favorable results [6]. 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

Static AR Processes

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

Varied AR Processes

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

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

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

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

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

[5] 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.

[6] 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.