Recurrent Neural Networks and Matrix Methods for Cognitive Radio Spectrum Prediction and Security

Alexander M. Glandon
Old Dominion University, aglan001@odu.edu

RECURRENT NEURAL NETWORKS AND MATRIX METHODS FOR COGNITIVE RADIO SPECTRUM PREDICTION AND SECURITY

by

Alexander M. Glandon
B.S. May 2015, Old Dominion University

A Thesis Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

ELECTRICAL AND COMPUTER ENGINEERING

OLD DOMINION UNIVERSITY
May 2017

ABSTRACT

RECURRENT NEURAL NETWORKS AND MATRIX METHODS FOR COGNITIVE RADIO SPECTRUM PREDICTION AND SECURITY

Alexander M. Glandon
Old Dominion University, 2017
Director: Dr. Khan M. Iftekharuddin

In this work, machine learning tools, including recurrent neural networks (RNNs), matrix completion, and non-negative matrix factorization (NMF), are used for cognitive radio problems. Specifically addressed are a missing data problem and a blind signal separation problem. A specialized RNN called Cellular Simultaneous Recurrent Network (CSRN), typically used in image processing applications, has been modified. The CRSN performs well for spatial spectrum prediction of radio signals with missing data. An algorithm called soft-impute for matrix completion used together with an RNN performs well for missing data problems in the radio spectrum time-frequency domain. Estimating missing spectrum data can improve cognitive radio efficiency. An NMF method called tuning pruning is used for blind source separation of radio signals in simulation. An NMF optimization technique using a geometric constraint is proposed to limit the solution space of blind signal separation. Both NMF methods are promising in addressing a security problem known as spectrum sensing data falsification attack.

ACKNOWLEDGEMENTS

I am grateful for Lasitha Vidyaratne, Mahbubul Alam, and other colleagues for their encouragement. I am grateful for my advisor Dr. Iftekharuddin for his supervision. I am grateful for Dr. Xin and his student Sharif Ullah for their guidance throughout the project. I am grateful for Dr. Li for his insightful suggestions. I am grateful for my Mom and Dad and for my wife Marjorie and her Mom and Dad. I am grateful for my brother and other family and friends. This work is partially supported through a grant funding from NSF (Award #1310353).

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Cognitive radio (CR) is a technology that intelligently analyzes a spectrum-environment to provide reliable communication and efficient radio spectrum usage. Government regulations assign bands to licensed users (also called primary users). This static assignment of spectrum is inefficient; typically at a given location and time, spectrum is mostly unoccupied by the primary users. The emergence of software defined radio affords dynamic reconfiguration of devices and enables CR. With CR, a radio can change parameters to operate over available spectrum (also called secondary use of spectrum) [1].

Machine learning is another enabler of CR. Machine learning may be used to determine a placement of radio nodes to allow strong network spectrum coverage with a minimal resource requirement [2]. Machine learning may also help to identify spectrum holes from readings in order to decide a mode of operation that is cooperative, reliable, and efficient [3], and it can be used to identify a malicious radio network node [4]. In this work, previously used techniques from the literature will be used together and extended to provide novel solutions for these tasks.

Artificial neural networks have been applied to spatial spectrum analysis for CR. The novelty here is to use specialized neural networks that operate over 2D grids of image pixels for image processing, modifying those networks for 2D spatial spectrum analysis. They are then used as a more generalized prediction tool that is shown to operate over many degrees of freedom. A tool for finding missing data in 2D grids called matrix completion has been used for frequency-time spectrum prediction in CR, and a special type of neural network for time series processing has been used for spectrum forecasting. This matrix completion is used together with the neural network tool. Together they allow forecasting to be performed using incomplete data. Lastly, a tool called non-negative matrix factorization (NMF) was used for radio source separation and localization in CR. A form of this factorization under specific constraints used in the literature for other problem domains is shown applicable to CR problems. Trilateration is shown to be applicable as a constraint for NMF, which was not seen in the literature. The matrix completion is also used as a preprocessing step for NMF with missing data.

1.1 Spatial Spectrum Prediction

When designing a CR network, the placement of radio nodes in space must be considered. It is not possible to test every configuration of radio nodes for their collective spectrum coverage. Given a few configurations with measurements taken over a grid of receiving locations, an artificial neural network can be trained to predict approximate signal strength for other configurations. The network used here is a Cellular Simultaneous Recurrent Neural Network (CSRN) which is well suited to the form of the problem. Given a limited number of spectrum readings from a given configuration, this network is used to predict readings over an entire spatial grid. Using the spectrum predictions, a set of network nodes' spatial placement can be decided.

1.2 Frequency-Time Spectrum Prediction

Another task is spectrum prediction given spectrum readings in the past. Readings taken at any radio node will be incomplete. This is because spectrum sensing can only be performed over a discrete subset of frequencies and times. The resolution over which a measurement can be performed is limited due to its continuous nature. Hardware costs are also a limiting factor for the availability of measurements.

Spectrum analyzers can sense a wide band of radio spectrum. Their measurement sweeps over this wide band, taking too much time for many CR applications where channel occupancy must be detected with minimal delay. However, spectrum analyzer data can be used for testing machine learning algorithms' capacities to predict radio spectrum power. When operating, each radio can measure a smaller frequency band over a shorter time compared to a spectrum analyzer. Therefore, the matrices are randomly masked to simulate incomplete readings from CRs leaving only blocks of a frequency-time spectrum available. This data prior to masking represents a radio spectrum ground truth. Soft-Impute is a matrix regularization (or completion) technique. See [5] for an early application of matrix completion to the CR problem domain. In section 2.6 certain properties inherent in the measurements are described. These properties allow the matrix completion technique to make good approximations for the missing readings, even considering the presence of noise [6].

1.3 Short Term Temporal Spectrum Forecasting

Given a frequency channel observed to have spectrum holes in the past, a CR may still interfere with primary users as spectrum holes are a dynamic phenomenon. A solution to this problem is to make short term forecasts of signal strength based on past signal strength. These forecasts can then be analyzed for spectrum holes that are likely to occur. In this work, time recurrent neural networks (RNNs) are used to forecast near future values of signal strength. Due to the highly irregular nature of spectrum availability, RNNs are used as a tool because they can predict complicated trends in time series. RNNs can also be trained in real-time, changing their parameters used for prediction as the signal trend characteristics change over time. RNNs are used in [7] for prediction of future radio signal strength. Here, RNNs are used to predict future values of time series that have been found using the matrix completion. It will be shown that future signal values cannot be predicted using matrix completion alone. Therefore, RNNs will then be applied to process these completed matrices. The RNNs are trained to use trends in time series for different frequency bands to make predictions of future data in each band.

1.4 Security Threat Identification

Cognitive radio systems are often described with a special component called a fusion center (initially named as a “designated controller” in [8]). A fusion center can be used for analyzing readings from multiple CRs and then determining the availability of spectrum. Another application of a fusion center is in identification of security threats [4, 9]. One specific threat is a spectrum sensing data falsification attack (SSDF). In this type of attack a radio reports misleading spectrum measurements. The radio’s reports could even be selectively true or false to cause the most harm while also reducing the possibility of being discovered as malicious [10].

Collaborative sensing with a fusion center enables blind signal separation techniques as radio spectrum measurements are known from several locations. A proposed leave-one-out blind signal separation scheme is described in section 4.4 as a defense against SSDF. To perform the blind signal separation, the matrix of power measurements over the receivers, frequencies, and times of measurement is decomposed using a NMF. The first matrix in the factorization represents the inverse square of the path from each radio transmitter source to each CR receiver. This represents signal pathloss. The second matrix in the factorization represents the power emitted by each transmitter at specific frequencies and times. The technique applied is used to find the factorization with an unknown number of transmitters. It is found that, given matrix dimensions significantly larger than the rank, the factorization is unique up to a scaling and a permutation. It will be shown in section 4.4 how this scaled and permuted factorization can be used to separate the transmitted signals.

For each radio transmitting a signal, its position can be determined using least-squares trilateration [11] given the NMF decomposition. This can be applied to determine where a malicious user is operating. This least-squares approach will also generate a geometric constraint that can be used to determine the quality of a NMF decomposition. If the quality of a NMF decomposition is given, this can be used to verify which decompositions derive from a malicious report, and this represents a constraint that can be used to solve the NMF problem under a more general setting, such as when the number of transmitters is unknown.

1.5 Integration of Approaches

Spatial spectrum prediction problem can be formulated as a missing data problem. A neural network given limited number of power measurement samples over a grid of locations, can estimate the missing data over the remainder of the grid. Frequency-time spectrum prediction can also be formulated as a missing data problem. A receiver node fixed in space may have limited measurements of spectrum over variable frequencies and times. Here, the matrix completion method estimate the missing data over the remainder of a frequency-time grid. Forecasting in time is performed using the completed data. The difference in the two problems is the structure of the given measurements. In the spatial spectrum prediction problem, grids of spectrum are completely measured for certain transmitter locations. In the frequency-time prediction problem complete grid measurements are not available, but the matrix to be completed is known to have certain properties (see section 2.6). Blind signal separation is an inherently different type of problem from the missing data problem. Here, signals from separate sources are additively combined. Given the viewpoints of multiple receivers, signal transmission sources can be estimated using NMF.

These two approaches can be combined. Each receiver in a collaborative network will have limited measurements over the frequency-time domain. In a real application when blind signal separation is performed it will be necessary to estimate the missing measurements beforehand. Matrix completion can be used as a preprocessing step to estimate the missing data prior to using non-negative matrix factorization for blind signal separation, and given the signal separation/localization, RNNs may be applied to forecast future spectrum power. In the collaborative sensing context, this allows for more accurate determination of spectrum availability and better assurance of non-interference with other radios' communications.

1.6 Summary of Goals

To summarize the work to be done, 5 goals are identified. First is to use CSRN to solve the missing data problem for spatial spectrum prediction. This can be used in planning the spatial layout of a radio network. Second is to use matrix completion to solve the missing data problem for frequency-time prediction and to process the completed data using RNNs for short term temporal forecasting. This can be useful in identification of spectrum holes that secondary users can utilize for communication. Third is to use NMF for blind source separation. This is shown applicable for identification of SSDF attacks. Fourth is to describe a novel NMF algorithm for blind source separation that uses a geometric constraint. This may be able to identify the latent dimension of a measured matrix by reducing the solution space of factorizations. Fifth is to use matrix completion as a preprocessing tool for NMF. This enables blind source separation when there is missing data in the measured matrix.

CHAPTER 2
BACKGROUND

2.1 Software Defined Radio

As mentioned, software defined radio is an enabler of CR. Radio began as an analog technology. Through the 70s and 80s more components were implemented digitally. This includes the information sent through the radio waves and the hardware that manipulates these signals into media that a person uses like text, sound, or video. Hardware flexibility increased and cost decreased over time. Hardware also became more programmable and therefore more configurable. These developments make the ability to dynamically change radio operating parameters on a single device more practical, and software became a more dominant tool for implementing this configurable operation [12]. As time passed, more components of the radio interface moved from analog to digital hardware and, finally, to software.

On a modern software defined radio, much of the signal processing is performed in software. To describe this, consider the path from the human understandable information to the analog wave. With slight modification, the same applies in the reverse direction. A message today is typically coded in digital form. This first requires encoding. Information stored on a computer is already in binary form. During a phone call, an encoder takes voice and converts it to binary form. A modern video camera records, encodes, and then stores in binary form. Early radios transmitting voice, for example, never encoded the sound into binary but instead converted the sound into another analog signal. This modern standard of digitization is useful for software defined radio. It allows for information transmitted to be digitally coded for the channel from its already available binary representation. There are various channel coding schemes, implementing reliable transmission and security that can be completed in software. The next part of the pipeline converts the digital signal to analog. Depending on the frequency of the carrier wave used, the digital signal may need to be sent very quickly. To complete more tasks in software, the digital to analog converters (and analog to digital converters) must operate at a higher frequency [13]. This presents a challenge to hardware developers that is currently being addressed. Currently, several aspects of radio are still implemented in analog. Upconverters and downconverters modulate and demodulate waves in analog circuits. Power amplifiers, as their name implies, amplify the power of the signal in analog circuits. Filters tune out unwanted frequencies in analog circuits. Even antennas are reconfigured using cues from software for different frequencies. These components are dynamically tuned or selected using software but are still implemented in analog circuitry [14].

Different infrastructures are put in place for different communications implementations. Software defined radios have potential to work across protocol potentially reducing the current diversification necessary in infrastructure [13]. Radio configurability also allows for CR systems to be practically implemented.

2.2 Cognitive Radio

In 2014-2015, the FCC auctioned a portion of the spectrum in their auction “Advanced Wireless Services”. The net take was $41 billion [15]. This illustrates the ineffective utilization and high demand for spectrum. Even in high traffic urban areas there is an abundance of occupancy in some bands, and zero to low occupancy in other bands. This suggests that some licensed bands are not used all the time or in all locations by the users that they are licensed to. CR is a suggestion that by developing more advanced flexible radio nodes we can find more opportunities to use available spectrum. This may be of practical necessity as devices that use wireless spectrum are proliferated. As early as 2002 the FCC had published a report to stimulate this kind of management of the wireless spectrum [3].

CR is a new paradigm for radio. The idea is to use software defined radios with flexible receiver characteristics to learn what the radio environment is like at a particular location and time across several frequencies. The radio can use this information to determine what channel can be used for transmission and how to operate on that channel. Being aware of the available spectrum and operating over spectrum holes, the radio network becomes more reliable and efficient, and reducing interference by operating over spectrum holes improves reliability. More information may be sent at a time and a location improving efficiency. A radio may even send information about its radio scene awareness to other nodes in cooperation. This creates a “feedback” of information into the radio scene from each node that is learning about the scene [3].

2.3 Energy Sensing

There are several methods of spectrum sensing including energy detection, cyclostationarity, matched filtering, etc. This study is conducted on data from different devices measuring radio signal energy. Occupancy of a spectrum band can be determined by comparing a summation of energy over a spectrum band at one time to a threshold based on a noise floor [16]. One technique for determining a noise floor is based on measuring the temperature of a location [17]. If the signal is within a range of the noise floor this indicates that a primary user is not active. If the signal is above the noise floor, a primary user may or may not be present. The secondary users can only operate when the primary user is known not to be active. If the primary user may or may not be present, the secondary user cannot operate.

2.4 Neural Networks for Spatial Spectrum Prediction

One task performed here prediction of spectrum coverage from a set of radio nodes. Because separate radio waves pass through each other without interference, this problem can be decomposed into predicting the spectrum coverage from radio nodes individually. Traditionally this problem is approached using mathematical modeling of pathloss under simple geometric environments. These models are formed using physical laws or empirical evidence, but both types involve explicit mathematical formula [18]. Recently, artificial neural network (hereafter referred to as neural network) modeling of a different type has been applied to the pathloss problem. Neural networks show promise in predicting pathloss in environments with complicated geometries, such as indoors. This approach is taken in [2]; their model is novel and they don't provide the network information about the propagation environment. Instead they give the network a training set of measurements without explicitly defining the environment's geometry. The neural network is tasked with learning from example the path-loss model of an indoor environment. Although others have used neural networks without explicitly defining the environment [19], this is a first example of modeling such a complex geometry of reflections and absorptions. Here, the spatial prediction is again performed environment-blind, but with a more powerful neural network. In [2] the network is trained to use the transmitter and receiver coordinates to infer the pathloss between them. Here, the network is trained to use a set of sample measurements with a transmitter at a fixed location, and to find the power over remaining locations over a grid. This missing data problem is well suited for CSRN that is used in this work. Before describing CSRN, an overview of neural networks is presented.

Table 1: Artificial Neural Network Terminology

statistics	neural networks
model	network
estimation	learning
regression	supervised learning
interpolation	generalization
observations	training set
parameters	synaptic weights
independent variables	inputs
dependent variables	outputs
ridge regression	weight decay

Table 1 found in [20] is an introduction to artificial neural networks using statistical terminology. Neural networks are a highly general purpose tool in pattern recognition. The basic form is a 3 layer feed-forward neural network used for a supervised learning problem. As an example, consider a non-linear regression problem on a surface, i.e. 2 inputs, and 1 output. A corresponding 3 layer network has an input layer with two nodes for the inputs, a hidden layer with a fixed but somewhat arbitrary number of hidden nodes, and an output layer with one node for the output. The values incoming to each node in the hidden layer from the input layer are multiplied by weights and summed. A non-linear function operates each hidden layer node and produces a set of hidden values. Then these hidden values are multiplied by weights and summed to produce an output value. Several algorithms exist that, given examples of inputs and corresponding outputs, will find weight values that setup the network as an appropriate input-output map. These are called supervised learning algorithms as they are supplied example outputs for each input set. Equations 1 and 2 describe the operation of a 3 layer network with *N* inputs and 1 output.

$$h_m = \sigma(b_m + \sum_{n=1}^{N} w_{mn} \cdot i_n) \tag{1}$$

$$o = b + \sum_{m=1}^{M} w_m \cdot h_m \tag{2}$$

The value of the non-linear function *σ*'s output at hidden node *m* is h_m. The output of the entire network is *o*. The weights from the input *n* to hidden node *m* are w_{mn}. There is also a bias 1 term that is weighted by b_m for input to each hidden node. The weights from the hidden node m to output node are w_m. There is also a bias weight *b* input to the output node. For a classification type problem, different ranges of the continuous output are mapped to discrete class labels or from 0 to 1 continuously as probability for different class membership.

2.5 Cellular Simultaneous Recurrent Neural Network

Cellular Simultaneous Recurrent Neural Networks (CSRNs) are a specialized type of neural network. They are first shown as capable tools in solving a maze traversal problem [21]. This task has not been solvable using typical feed-forward neural networks (multi-layer perceptrons). Since initial application for maze traversal, CSRNs have been applied to various other problems such as image processing [22]. A CSRN is a neural network composed of subnetworks arranged in cells to match the topology of a problem domain. A typical example is a 2D grid. For example, in maze traversal, each cell may represent a wall unit or a passageway unit. In image processing or each cell may correspond to a pixel of an image to be processed. Each subnetwork in a cell is a simultaneous recurrent neural network (SRN). The simultaneous characterization is optional (see [23]), but is used in this work since the processing performed has no time component.

The architecture of each SRN is shown in figure 1. There are several options in configuring each SRN. The SRN depicted in figure 1 is Elman style with a fully connected recurrent to hidden layer. Here, a generalized multilayer perceptron configuration is used, with limited connectivity between the recurrent and the hidden layer. Neighbor connections pass between each cell in the grid, and in this network 4 connections are incoming to each SRN from its North, East, South, and West neighbor cells. The choice of which hidden nodes to use for neighbor connection and which hidden node to use for output is also optional. The number of neighbor connections is optional. For example, diagonal compass directions such as Northeast could also have been used l. This would result in more network complexity, which may make training more difficult. Figure 2 shows an example architecture mapped to a grid of power readings. Each location on the grid has an associated SRN. In section 3.1 CSRN is shown to map to the problem of spatial spectrum prediction and the specific architecture used is described.

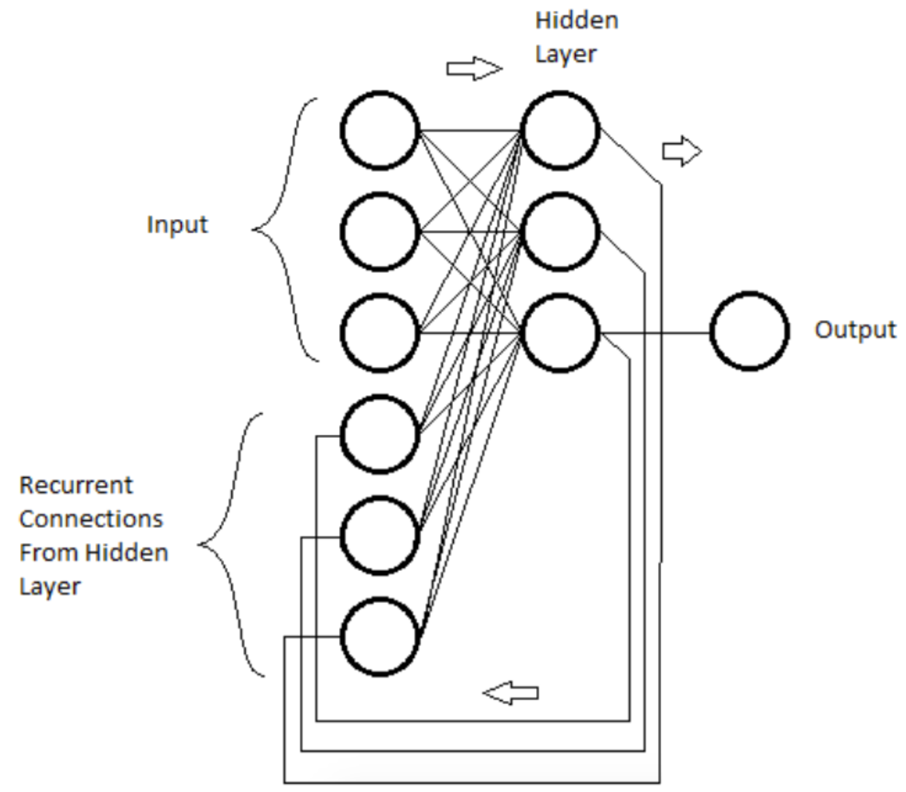


Figure 1: Simultaneous Recurrent Network Architecture

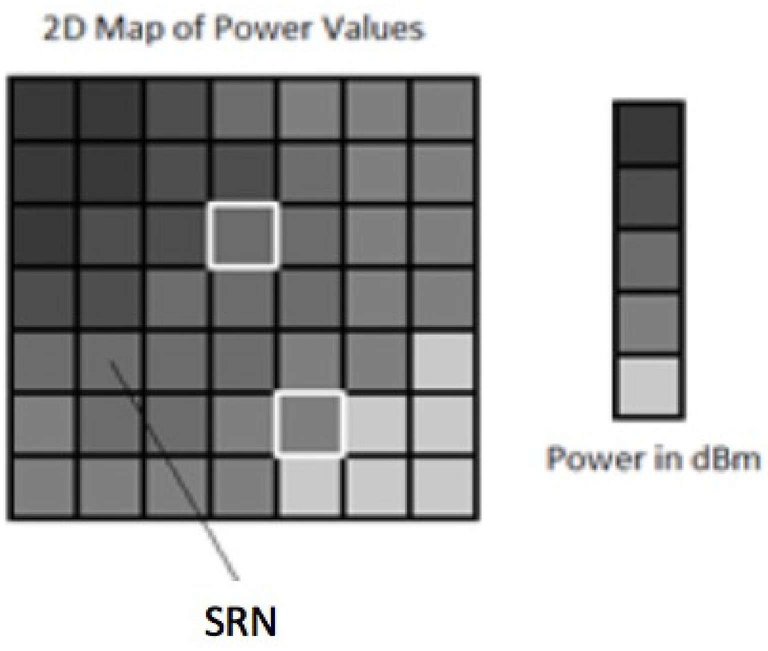


Figure 2: CSRN mapped to radio domain problem

2.6 Missing Data Problem

Another side of spectrum prediction involves using a radio node to measure spectrum power at one location in space. The measurements collected will vary not in spatial location but in frequency and time. As mentioned, a software defined radio can dynamically change its operating frequency. But it doesn't operate on all bands simultaneously. Therefore, the measurements from a radio node are incomplete over frequency and time. A popular and related problem is the Netflix problem. In the Netflix problem an incomplete matrix is available that represents different ratings of movies. Each row is a different person who gave the rating, and each column is a different movie that was rated. This matrix can be modeled as a low rank matrix with noise. This is because peoples' tastes in movies are believed to fall into groups or personality types. Then groups of rows from the matrix should have similar ratings, as they correspond to these personality groups. These may be higher or lower for individuals as some people are more easily pleased or less easily pleased. This implies that the rows are linearly dependent with added noise [24], meaning similar up to a scaling factor within each group. This is also called low rank. In the radio problem, the low rank assumption is found because the power measured is found to have correlation across different frequencies and across different times. Following [5, 6], matrix completion is used here to estimate missing power measurements over frequency and time. The algorithm is described in the next section.

The dataset used in this work [17] consists of spectrum analyzer power measurements in units of decibel-milliwatts (dBm). Figure 3 shows large correlation values away from the main diagonal in the frequency correlation matrix. Large correlation values away from the main diagonal are also present in the time correlation matrix. See [6] for similar figures of frequency and time correlation matrices for this data set. They also give matrix completion performance on a band of frequencies known for television broadcasting as this band has strong correlations in frequency and time. Here, the TV band is also used for testing the performance of the matrix completion. It may be more plausible to complete the matrix using a ratio scale (e.g. watts) instead of a log scale (e.g. decibel-milliwatts). However, it is found with this data that performance is better using the log scale originally given by the dataset. An explanation for this was not yet found.

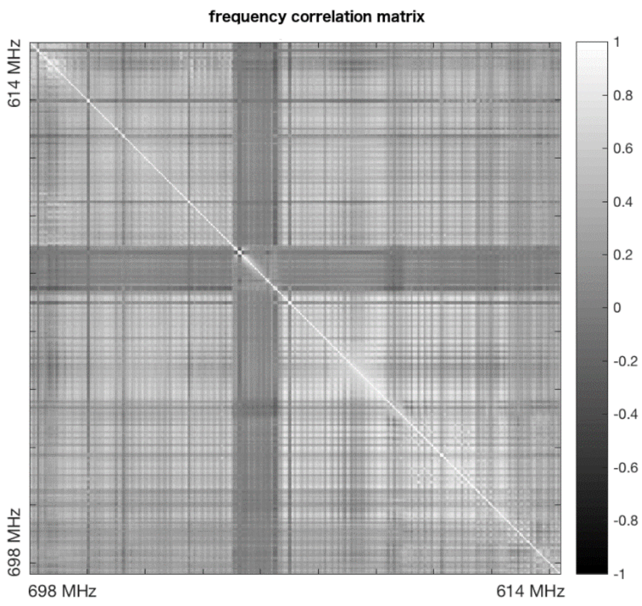


Figure 3: Correlation Matrix over Frequencies 614-698 MHz (TV Bands)

As is shown mathematically in section 3.5, it is insufficient for prediction purposes to just complete the matrix. Forecasting using time RNNs for CR was performed in [7]. There an entire time series of power readings for a given frequency is assumed to be known a priori. Here, we apply RNNs as post processing to the matrix completion for short term forecasting using the estimated time series for different frequency bands.

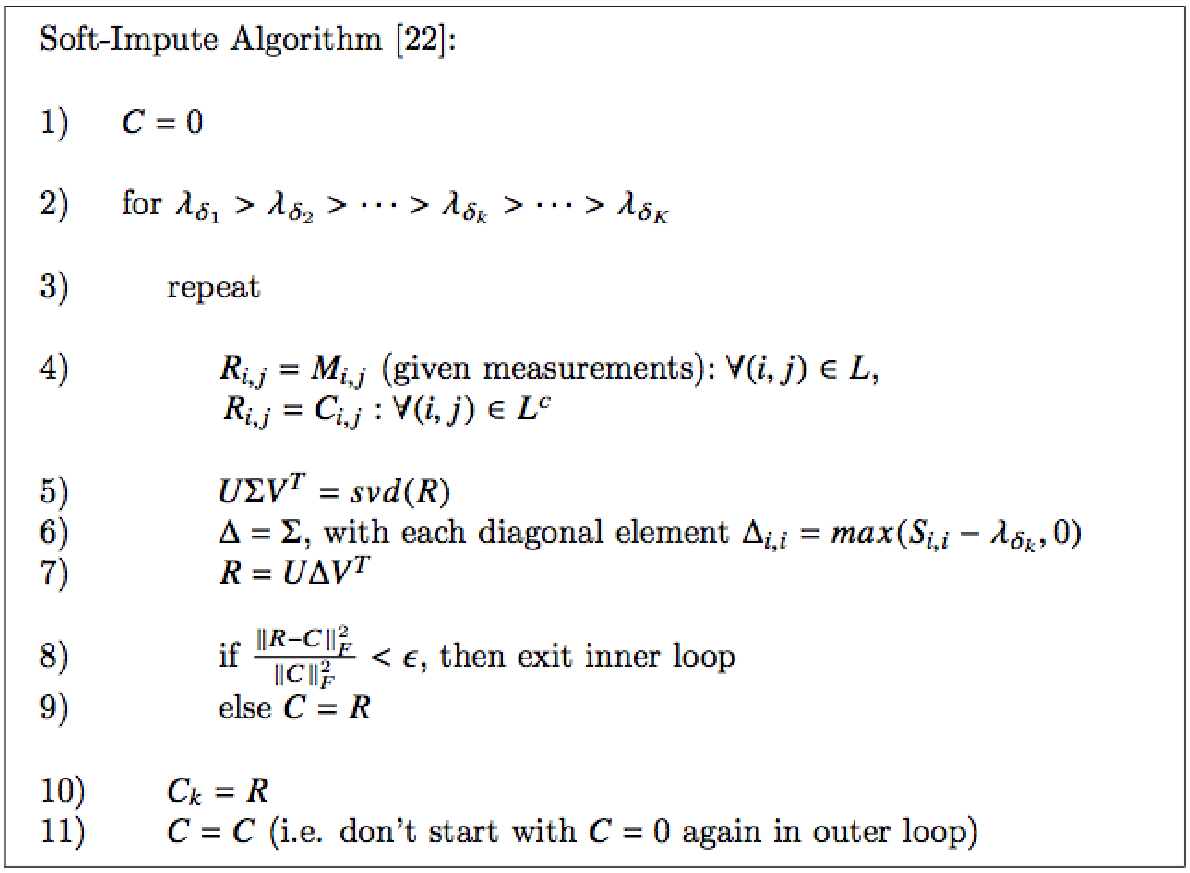
2.7 Soft-Impute Algorithm

Given a partially measured matrix of data, soft-impute is an algorithm for determining the missing data. As the correlation in frequency and time is shown to be high in our spectrum analyzer data, it can be modeled as a low rank matrix with noise.

Soft-impute [24] solves the optimization problem posed in equation 3. M is the incompletely observed matrix. C is the completed matrix that will be inferred. L is the set of locations in the incomplete matrix where values are measured. L_c, the complement of L, is the set of locations where the data is to be completed and will be referred to in the algorithm presented in figure 4. The assumption of low rank can be applied in a relaxed form with the constraint that $||C||_*$ (the nuclear norm of C) is bounded by a small constant δ.

$$\min_{C} \sum_{(i,j)\in L} (C_{i,j} - M_{i,j})^2, \|C\|_* \leq \delta \tag{3}$$

Soft-impute finds a solution to this optimization problem using the algorithm in figure 4. The intuition behind the algorithm is that by removing a portion of the SVD diagonal, the matrix becomes lower rank but less reflective of the measured matrix. To restore the new matrices' compatibility with the measured data, the known measurements are reset. Thus, in an alternating manner, the diagonal is reduced and the known measurements are set to their measured values. A series of solutions is found for the regularization problem posed with different values of δ from equation 13. The decreasing values of λ_δ are related to different values for δ (giving solutions with decreasing nuclear norms). Determination of parameters λ_δ and ε is described in section 4.2.



```
Soft-Impute Algorithm [22]:

1)   $C = 0$
2)   for $\lambda_{\delta_1} > \lambda_{\delta_2} > \cdots > \lambda_{\delta_k} > \cdots > \lambda_{\delta_K}$
3)       repeat
4)           $R_{i,j} = M_{i,j}$ (given measurements): $\forall(i,j) \in L$,
             $R_{i,j} = C_{i,j} : \forall(i,j) \in L^c$
5)           $U\Sigma V^T = svd(R)$
6)           $\Delta = \Sigma$, with each diagonal element $\Delta_{i,i} = max(S_{i,i} - \lambda_{\delta_k}, 0)$
7)           $R = U\Delta V^T$
8)           if $\frac{\|R-C\|_F^2}{\|C\|_F^2} < \epsilon$, then exit inner loop
9)           else $C = R$
10)      $C_k = R$
11)      $C = C$ (i.e. don't start with $C = 0$ again in outer loop)
```

Figure 4: Soft-Impute Algorithm

2.8 Time Recurrent Neural Networks

Given the predictions for the missing data computed using soft-impute, the next step is to predict the future values of the spectrum power using a time recurrent neural network (RNN). RNNs can detect trends in a signal using feedback processing. In a scenario with missing data in a signal the processing can become distorted. Therefore, it is useful to complete the matrix that contains these time varying signals. The matrix completion is a preprocessing step that gives an estimate of the missing signal values so that the RNN can predict using these estimates where the data is not available.

In this work, a single hidden layer Elman type RNN [25] is used. The Elman RNN used is like the SRN in the CSRN. There is an input layer, a hidden layer, and a recurrent input. The illustration in figure 1 is useful for describing the Elman RNN. The only difference is that in the SRN, the input to each cell is not varying in time. The processing occurring was "simultaneous" in the sense that the external input was invariant (the neighbor input considered internal). In the Elman RNN for time series prediction, the external input is variable in time. Consider a 3 input network. This network is generally used by passing a window over a time series as illustrated in figure 5. To get an output for time *T*, the input to the network is the signal at time *T*−3, *T*−2, and *T*−1.

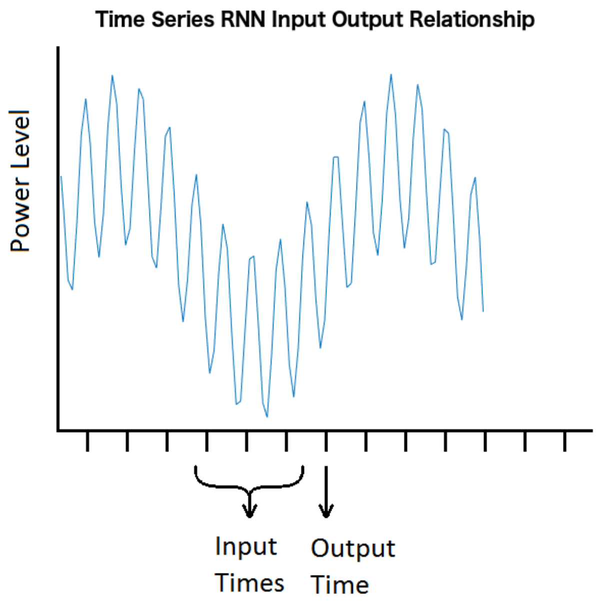


Figure 5: RNN Input/Output Sliding Window

A recurrent input comes from the hidden layer for a previous time window the network was run on. To train a network, known output is used enabling supervised learning. To use the network to forecast, some history of the time series must exist. The network is run over the time series history, and short-term context is used via feedback in the network's estimation of future data. To predict the output one time step into the future, the network is run using inputs up to the end of the known history. To predict the output more than one step ahead, the previous forecasts are also used as inputs.

2.9 Blind Source Separation

Another application of machine learning that has been performed for CR involves separating signals received from different transmitters. There has also been work to localize the transmitters in space given received signals. In [26, 27] the problem is posed as a matrix factorization $X = UV$. X represents the powers received. In [26], for X, each row represents a receiver and each column represents the frequency that is being measured. Elements of U are proportional to the power transmitted times $\left(\frac{c}{4\pi f}\right)^2$ where c is the speed of light, and f is the frequency band of transmission. Elements of V are inversely proportional to d^2, where d is the distance from transmitter to receiver. NMF typically does not give a unique decomposition. To reduce the possible decomposition space assumptions are made. It is assumed that there are a known number of transmitters. It is also assumed that some rows of U are known corresponding to knowing the power transmitted over every frequency for some transmitters. This is a helpful constraint that will be applied here in a similar way. After the decomposition is found a weighted centroid method is used to localize the transmitters.

In [27] the decomposition is more realistic. U represents the power from each transmitter at each time. V represents $\frac{Gc^2}{4\pi f d^\alpha}$ where again c is speed of light, f is frequency band, G is channel gain, and d is transmitter to receiver distance. This d^α term makes localization difficult, and will be formulated differently in our model as a separate pathloss and fading. They assume that the number of transmitters is known. Instead of assuming some transmitter powers are known, they use the constraint that transmitted power is piece-wise constant over time.

In section 3.7 a similar formulation is constructed. It is solved with the number of transmitters as an unknown. This model is used for two different NMF techniques. One method with used sparsity of one factor to help find the number of transmitters and a unique decomposition. The second method will be a proposed use of geometric properties of the problem to find the decomposition and for localization of the transmitters. In [28] the problem of signal separation is approached by applying power spectrum sensing. Their model is like the previous two models discussed, but uses power spectrum instead of energy sensing. They assume that the number of transmitters is known in their model. Here, we will perform the factorization when the number of transmitters is not available, which makes the problem more challenging and realistic.

2.10 Tuning Pruning

Non-negative matrix factorization is a method that can be applied to blind signal separation. A measured matrix can be factored into a product of two matrices signifying constituent elements of a problem. If applicable, a set of constraints is used for a given problem to ensure a unique factorization up to a permutation of the rows of the first matrix and columns of the second matrix in the product. One popular constraint is to use prior knowledge of the latent dimension of the matrix product. Equation 4 shows a matrix product of a rank *r* matrix *M*, where *r* is the latent dimension of the decomposition *AB*.

$$\begin{gathered} M = AB \\ M \in \mathbb{R}^{m\times n}, \quad A \in \mathbb{R}^{m\times r}, \quad B \in \mathbb{R}^{r\times n} \end{gathered} \tag{4}$$

This work applies a technique for automatic relevance determination, where sparseness gives a unique decomposition [29]. In this technique, the latent dimension is automatically determined. It builds upon a gradient descent technique for finding decompositions described in [30]. In gradient descent, a cost function is given. This can be the sum of squares of the difference in the elements of the original matrix and the elements of the product of the factors in the matrix decomposition. Gradient descent involves taking the gradient of the cost function with respect to the solution found. The negative of the gradient of a given solution will point to a lower cost solution. Iteratively this can be used to find a minimal cost solution. In NMF problems, it is typical to alternate taking the gradient with respect to each factor, and to update according to those gradients one at a time. Projected gradient as described in [30], uses second order derivative information as a condition to find the optimal number of times in a row to update with respect to each factor. The automatic relevance determination work takes this second order projected gradient technique and builds on it. They include a penalty in their cost function for deviation in sparsity in one of the factors. The decomposition begins with a large latent dimension. As the algorithm follows the gradient descent, when a latent dimension becomes very irrelevant it is removed. This means that if most entries are far smaller than in the other latent dimensions they are considered not relevant. At the same time the penalty promotes sparsity, or zero entries. Eventually the model is reduced to a form that can no longer have elements removed, without increasing the cost. This factorization should be the most parsimonious.

Here, the tuning pruning algorithm is exposited. The matrix to be factored is *M*, and its factorization is *AB*, where *B* is assumed to be a sparse matrix. The cost function shown in equation 5 is a sum of squares of the difference in the matrix to be decomposed *M* and its factorization *AB* plus a penalty for sparse rows of *B* (see β defined in figure 6). Σ^2 represents the variance of the noise in the measured matrix M, so, intuitively, if the noise is high, the cost is reduced for poor factorizations.

$$C = \frac{\|M - AB\|_F^2}{2\sigma^2} + \sum_k \beta_k \sum_m H_{k,m} \tag{5}$$

The algorithm for tuning pruning is given in figure 6. First *K* which represents the latent dimension is set to a large number. This number will be reduced until the most parsimonious model is found. *B* reflects the importance of each latent dimension to the model. Initializing β to 0 means that each column of *A* or row of *B* is considered important initially. Rows of *A* and columns of *B* are gradually pruned. To find the update of *A* and *B*, projected gradient is used. Projected gradient was applied using first and second derivative of the cost function with respect to *A* and *B* individually. These derivatives are given in equations 6 through 9. The second derivate matrices are block diagonal. B_k is updated by determining the sparsity of the rows of *B*. When a row becomes irrelevant because most entries are close to 0, it is removed.

```
Tuning-Pruning Algorithm [12]:

1)    set $K$ to a large number
2)    set $\sigma^2$ to the noise variance assumed in $M$
3)    set $\beta_k$ to 0 for all $k \in \{1, 2, \ldots, K\}$
4)    set $A$ and $B$ to random matrices

5)        repeat

6)            $A = projected\_gradient(A, B)$
7)            $B = projected\_gradient(A, B, \sigma^2, \beta)$
8)            $\beta_k = \frac{M}{\sum_m H_{k,m}}$
9)            for all $\beta_k > 10^9 \|M\|_F$,
              remove column $k$ from $A$ and remove row $k$ from $B$

10)       until convergence
```

Figure 6: Tuning-Pruning Algorithm

$$\nabla_A C = \frac{(AB - M)B^T}{\sigma^2} \tag{6}$$

$$\nabla_A^2 C = \frac{\begin{bmatrix} BB^T & & \\ & \ddots & \\ & & BB^T \end{bmatrix}}{\sigma^2} \tag{7}$$

$$\nabla_B C = \frac{A^T(AB - M)}{\sigma^2} + \begin{bmatrix} \beta_1 & \cdots & \beta_K \\ \beta_1 & \cdots & \beta_K \\ & \vdots & \\ \beta_1 & \cdots & \beta_K \end{bmatrix} \tag{8}$$

$$\nabla_B^2 C = \frac{\begin{bmatrix} A^T A & & \\ & \ddots & \\ & & A^T A \end{bmatrix}}{\sigma^2} \tag{9}$$

The tuning pruning paper suggests performing the projected gradient using an alternating technique, where *A* is updated by subtracting the derivate of the cost with respect to *A* and then *B* is updated by subtracting the derivative of the cost with respect to *B*. In [30] an algorithm is given to use the second derivate information to find the optimal number of alternating updates before continuing to the next step of the algorithm. In [29] they also suggest a more direct approach to continue after the alternating updates produce less than 10^{-6} change in the cost *C* or when 25 alternating steps have been performed.

2.11 Least-Squares Trilateration

If a collaborative network can decompose a matrix of power readings, then the factor giving the distances from each receiver node to each transmitter node can be used to localize transmitter nodes. Each receiver's location may be known using GPS coordinates. Given at least 3 receiver nodes, a trilateration process can perform the localization of the transmitters. There are *M* receiver locations (r_{1x}, r_{1y}), (r_{2x}, r_{2y}), ... , (r_{Mx},r_{My}). There are *K* transmitter nodes with unknown locations. Then there exists $K \cdot M$ equations:

$$d_{mk}^2 = (r_{mx} - t_{kx})^2 + (r_{my} - t_{ky})^2 \tag{10}$$

The distance estimated from transmitter *k* to receiver *m* is d_{mk}. These formulas can be rewritten as a least squares problem that is overdetermined if there are 3 or more receivers. Selecting receiver 1 as a reference node, the formulas are manipulated as follows [11]:

$$d_{mk}^2 = (r_{mx} - r_{1x} + r_{1x} - t_{kx})^2 + (r_{my} - r_{1y} + r_{1y} - t_{ky})^2 \tag{11}$$

$$\begin{aligned} d_{mk}^2 =&(r_{mx} - r_{1x})^2 + (r_{my} - r_{1y})^2 + 2(r_{1x} - t_{kx})(r_{mx} - r_{1x}) + \\ &2(r_{1y} - t_{ky})(r_{my} - r_{1y}) + (r_{1x} - t_{kx})^2 + (r_{1y} - t_{ky})^2 \end{aligned} \tag{12}$$

$$d_{mk}^2 = d_{m*}^2 + 2(r_{1x} - t_{kx})(r_{mx} - r_{1x}) + 2(r_{1y} - t_{ky})(r_{my} - r_{1y}) + d_{1k}^2 \tag{13}$$

$$(d_{mk}^2 - d_{m*}^2 - d_{1k}^2)/2 = (r_{1x} - t_{kx})(r_{mx} - r_{1x}) + (r_{1y} - t_{ky})(r_{my} - r_{1y}) \tag{14}$$

where d_{m*} is the distance from receiver m to reference receiver 1.

Linear System "*k*":

$$\begin{bmatrix} (r_{2x} - r_{1x}) & (r_{2y} - r_{1y}) \\ (r_{3x} - r_{1x}) & (r_{3y} - r_{1y}) \\ \vdots & \vdots \\ (r_{Mx} - r_{1x}) & (r_{My} - r_{1y}) \end{bmatrix} \cdot \begin{bmatrix} (r_{1x} - t_{kx}) \\ (r_{1x} - t_{ky}) \end{bmatrix} = \begin{bmatrix} (d_{2k}^2 - d_{2*}^2 - d_{1k}^2)/2 \\ (d_{3k}^2 - d_{3*}^2 - d_{1k}^2)/2 \\ \vdots \\ (d_{Mk}^2 - d_{M*}^2 - d_{1k}^2)/2 \end{bmatrix} \tag{15}$$

or

$$A \cdot x_k = b_k \tag{16}$$

Now the system of $K{\cdot}M$ equations has been rewritten as a set of K linear systems, each with $M-1$ equations. A least squares solution to each $A{\cdot}x_k = b_k$ can be found.

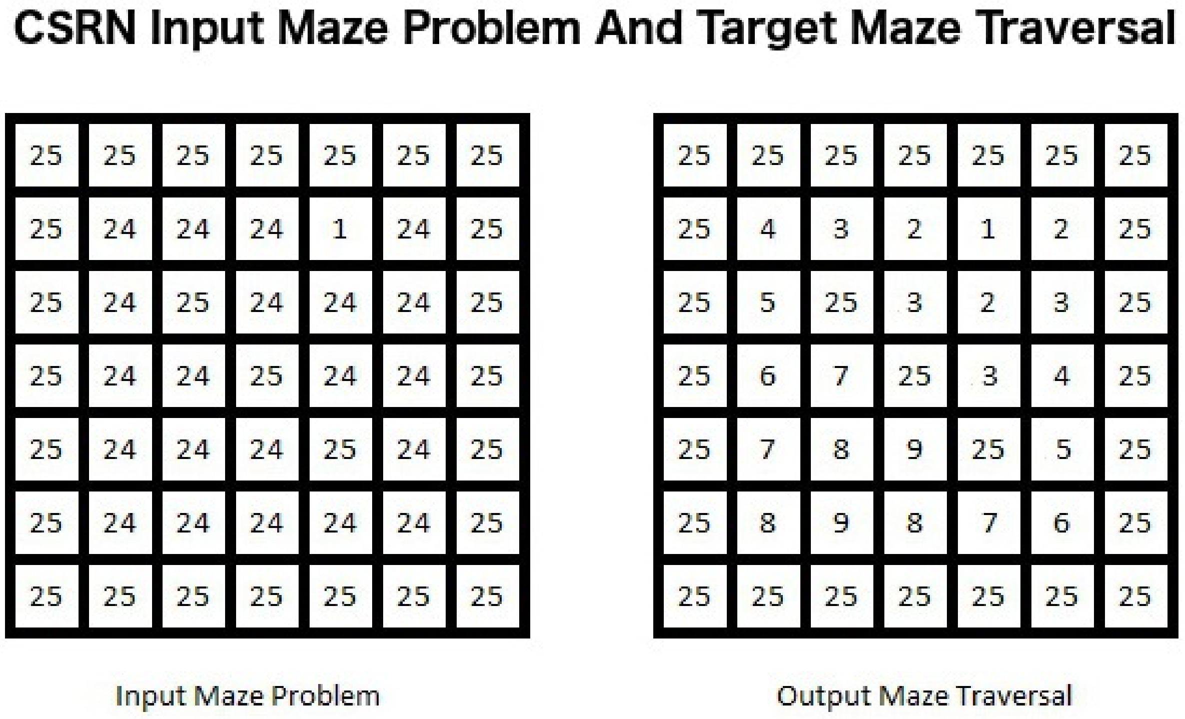
$$\hat{x_k} = (A^T A)^{-1} A^T b_k \tag{17}$$

Other methods for trilateration involve gradient descent or higher order derivative techniques such as Newton-Raphson which are also described in [11].

CHAPTER 3
METHODS

3.1 CSRN for Spatial Spectrum Prediction

In the original CSRN work [21] each SRN has one input reflecting the input maze architecture. For example, a higher input represents a wall, a medium valued input represents a path, and a lower value input represents a goal. The output of the network is a path from any location to the goal, indicated by decreasing values in cells along the path. Figure 7 gives an example input and output to the CSRN.


CSRN Input Maze Problem And Target Maze Traversal
Input Maze Problem
Output Maze Traversal

Input Maze Problem

25	25	25	25	25	25	25
25	24	24	24	1	24	25
25	24	25	24	24	24	25
25	24	24	25	24	24	25
25	24	24	24	25	24	25
25	24	24	24	24	24	25
25	25	25	25	25	25	25

Output Maze Traversal

25	25	25	25	25	25	25
25	4	3	2	1	2	25
25	5	25	3	2	3	25
25	6	7	25	3	4	25
25	7	8	9	25	5	25
25	8	9	8	7	6	25
25	25	25	25	25	25	25

Figure 7: Maze Example Inputs and Outputs

The problem of using a neural network to estimate missing spatial spectrum information was described in [2]. The idea there is to train an MLP to take the position of a transmitter and the position of a receiver as input to the network and to output the power received. The network is trained in a certain indoor environment, and the transmitter and receiver are moved to different pairs of locations. The network is tested in the same room, with the transmitter and receiver at different pairs of locations.

Here, a related but more challenging problem is considered. Given a limited set of measurements of the spatial spectrum in a room with the transmitter at a given location, can the received power be estimated for all receiver locations in the room. This problem is made difficult by moving the transmitter for each prediction task. As in [2] the network is trained with the transmitter in one set of locations, and testing with the transmitter in a completely different set of locations. This may be seen as similar to the maze problem [21]. The network has information about the goal and the walls, but elsewhere the path is left as a missing data problem. There is a pattern to be found, namely the direction of traversal to the goal from any path. In the spatial spectrum prediction problem given measurement samples are like having the goal given. The missing data to be completed is the intermediate readings. In both problems there is a constraint. In the maze, the walls are the constraint. In the spectrum prediction, the given measurements should constrain the solution.

In [22] a CSRN is used to perform affine transformations on images. Two inputs are given to each cell, one to parameterize the transformation and the other as location information for each pixel. The output gives the location information of the transformed pixel. This multiple input model was used in this work to solve the spatial spectrum prediction problem.

Figure 2 from chapter 2 shows the problem mapped to a CSRN. The readings collected are shown as highlighted grid locations. As described each cell where a power value is input or output contains an SRN, and all SRNs are interconnected. One input is a 0 or 1 corresponding to whether the cell corresponds to a location where power is measured. The second input is varied from 0 to 1 as the power at a cell is sampled between the minimum and maximum power measured. If this location is not where a measurement was taken, a default 0.5 was used as input. This was to give the least effect for the second input at unmeasured areas. Also, as logistic sigmoid is used for the nonlinearities, this 0.5 level input produces zero activation in this first layer. The output is representative of power over the entire grid. To train the model, measurements are taken over an entire spatial grid for some transmitter locations. The model is tested for prediction of power with different transmitter locations and input power measured at a few sample locations. The network predicts the remaining power over a grid for different transmitter locations.

3.2 Spatial Spectrum Data Collection

Spatial spectrum data has been collected for this experiment. Our laboratory room was taped in a grid with 5’ between grid points in one direction and 8’ between grid points in the other direction. The room is a rectangle and resolves to a 5x5 grid of sampling locations. The radio environment was observed over the Wi-Fi band using a Mac Wi-Fi software tool. IEEE 802.11b/g/n channel 2, being observed as empty in the lab was the channel that the data was collected on. A laptop running Linux was configured to transmit a constant power signal. When the laptops were placed on top of each other the signal strength measured approximately -27dBm. A script was written to wait for a user keystroke, which was given when the receiver was in the correct location on the grid. After the user signal, the laptop waited for a stable signal measurement, then recorded the signal power after stabilization. This was implemented by waiting for identical readings from successive measurements of signal power. The transmitter was moved to 5 different locations, and the receiver traversed the grid for each transmitter location. Figure 8 shows an example grid. The transmitter coincided with the peak signal power. Figure 9 is a contour plot of the same data.

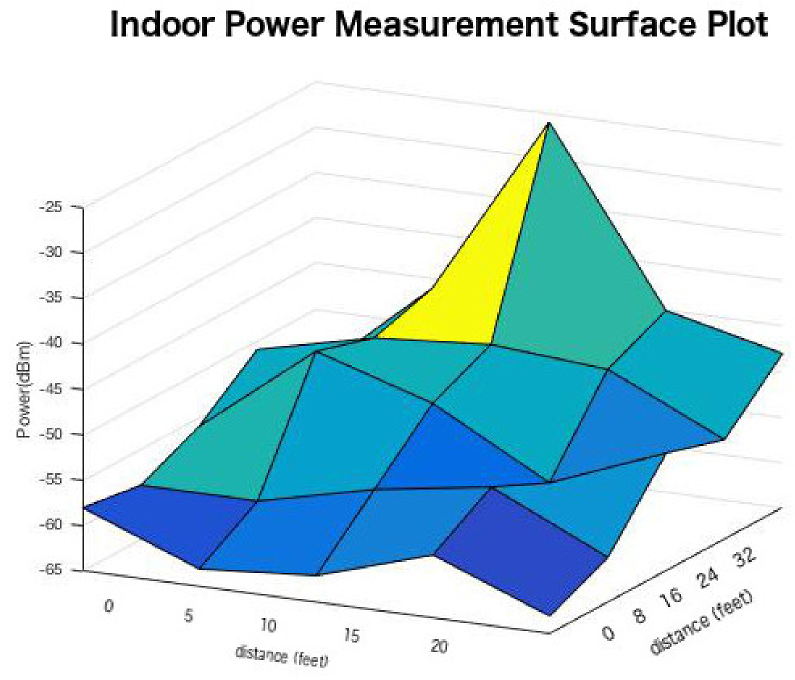


Figure 8: Example of Power Measurement Indoors

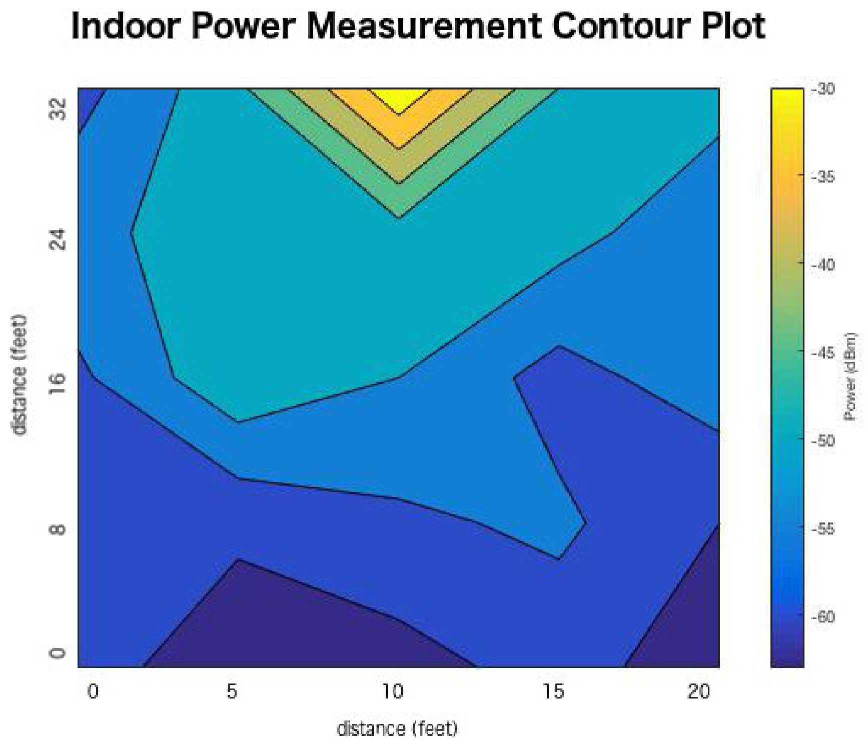


Figure 9: Contour Plot of Power Measurement Indoors

A trick is used to take a small dataset and get good training results. Because only a small set of sample locations are used as input to the network, these sample locations can be varied. For example, using 5 sample locations on a 5x5 grid gives 53130 possible sample schemes. Given this size training set, the CSRN network can learn a signal fading model that is generalized over arbitrary sampling schemes.

3.3 Unscented Kalman Filtering

Following the style of [22] UKF is used to train the CSRN. UKF is a tool that uses the forward propagation formula iteratively to tune the parameters of the network. UKF is an extension of Kalman filtering that is applicable to the non-linear mapping that CSRN performs. To adjust the weights, they are considered as a random vector. This random vector represents the state of a state space system as would be found in the Kalman filter type problem. The process equation is the identity transformation (i.e. the weight vector is unmodified), and the measurement equation is the CSRN network itself. In training a Kalman filter, the state of the system, or the weights for this model, are considered a random variable with an estimated distribution. For each training input or set of training inputs, a new distribution for the weights is determined by a balancing act. On one side the current weight estimate is used to find an output. On the other side is the training target.

To estimate distributions in UKF, a tool called unscented transformation is used. Given the mean and covariance of a random variable, a good estimate of the random variable transformed by a non-linear function can be made by using the unscented transformation. This involves using the mean and covariance to guess samples from the distribution. These samples are transformed by the nonlinear function, and the new samples are used to estimate the mean and covariance of the transformed variable. Additionally, the cross-covariance of the original random variable and the transformed random variable can be estimated.

Using this transform, the mean and covariance of the estimated output and cross covariance between the weight vector and the estimated output are found. The mean and covariance of the output represents how well the current weights predict the target. The cross covariance between the weight vector and the estimated output shows how the output changes as the weights are changed. In each iteration a random variable is determined that is the most likely distribution of the weights given the output and the target. This distribution of the weights is given by an updated mean and a covariance [31]. As is noted when training the RNNs, UKF does not always train as well as backpropagation. However, UKF and backpropagation performance are both good as will be discussed in the results section. Backpropagation is another method used for training and is made into an adaptable training tool.

3.4 Backpropagation

Backpropagation is another technique also used to train the CSRN. Since CSRN is a recurrent network, the backpropagation is commonly referred to as backpropagation through time. Backpropagation is a technique for assigning a set of parameters to a neural network so that the network maps input to output correctly. The premise is that a network represents a differentiable formula with respect to the parameters called weights, and a cost function can be composed using the network function. Input-output pairs that are examples of how the network map should behave are themselves input to the cost function. The output of the cost function is only low when the network map is correctly describing the input-output relationship. For example, if x_i and y_i are input-output pair *i*, and *f*(*x*) is the neural network map, then a typical cost function could be sum of residual squares as given in equation 18.

$$J = \sum_i (f(x_i) - y_i)^2 \tag{18}$$

The goal is to minimize *J* given the known input-output examples x_i, y_i. For the given cost function, if f(x) is differentiable with respect to the network weights, then so is J. The reason that it is useful to find the derivative of the network with respect to the weights is that it allows the network to be modified to reduce the value of J. This is because the gradient of J with respect to the weights shows the direction of change in the weights that results in the fastest increase in J and the negative of the gradient shows the direction of the change in the weights that results in the fastest decrease in J. Because this gradient is valid only locally or at a given value of the weights, to find the lowest value of the cost function J, the negative of the gradient can be used to modify the weights slightly. Small modifications should reduce the cost as this gradient is valid in a small region of the cost function. The process when repeated iteratively using the negative of the new gradient after the weights are perturbed is called backpropagation.

A method for training arbitrary networks with backpropagation avoiding the analytical Jacobian calculation is described here. As described in [32], “Backpropagation can be applied to any system with a well-defined order of calculations, even if those calculations depend on past calculations within the network itself.” In that work, equation 19 is presented that allows the gradient of the network with respect to the weights to be calculated in the order that the network function itself is evaluated. This implies that in very convoluted or recurrent neural network, that the gradient of the cost with respect to the weights can be determined in the order of the flow of information through the network.

$$\frac{\partial^+ J}{\partial z_i} = \frac{\partial J}{\partial z_i} + \sum_{j>i} \frac{\partial^+ J}{\partial z_j} * \frac{\partial z_j}{\partial z_i} \tag{19}$$

This equation is aptly named “ordered-derivative”. The equation is best motivated with an example. Say the network calculates the following function $f(x) = w_1 * g(x)$ and $g(x) = w_1 * x$. All z_i are the values needed to calculate the network output. First $z_1 = w_1$ and $z_2 = x$ are known. Notice that the first values known have the smallest indices on z (1 and 2). Next $z_3 = \mathrm{g(x)}$ is calculated, and so has the next index 3. Then $z_4 = f(x)$. Finally, $z_5 = J$. Thus, passing from input to output, first *TARGET* is $z_3 = z_1 * z_2$.

$$\begin{aligned}
\frac{\partial^+ z_3}{\partial z_1} &= \frac{\partial z_3}{\partial z_1} + \sum_{j>i} \frac{\partial^+ z_3}{\partial z_j} * \frac{\partial z_j}{\partial z_1} \\
\frac{\partial^+ z_3}{\partial z_1} &= \frac{\partial z_3}{\partial z_1} + \frac{\partial^+ z_3}{\partial z_2} * \frac{\partial z_2}{\partial z_1} \\
\frac{\partial^+ z_3}{\partial z_1} &= \frac{\partial z_3}{\partial z_1} + \frac{\partial^+ z_3}{\partial z_2} * 0 \\
\frac{\partial^+ z_3}{\partial z_1} &= \frac{\partial z_3}{\partial z_1} \\
\frac{\partial^+ z_3}{\partial z_1} &= z_2 \\
\frac{\partial^+ g}{\partial w_1} &= x
\end{aligned} \tag{20}$$

Second, *TARGET* is $z_4 = z_1 * z_3$.

$$\begin{aligned}
\frac{\partial^+ z_4}{\partial z_1} &= \frac{\partial z_4}{\partial z_1} + \sum_{j>i} \frac{\partial^+ z_4}{\partial z_j} * \frac{\partial z_j}{\partial z_1} \\
\frac{\partial^+ z_4}{\partial z_1} &= \frac{\partial z_4}{\partial z_1} + \frac{\partial^+ z_4}{\partial z_2} * \frac{\partial z_2}{\partial z_1} + \frac{\partial^+ z_4}{\partial z_3} * \frac{\partial z_3}{\partial z_1} \\
\frac{\partial^+ f}{\partial w_1} &= \frac{\partial f}{\partial w_1} + \frac{\partial^+ f}{\partial x} * \frac{\partial x}{\partial w_1} + \frac{\partial^+ f}{\partial g} * \frac{\partial g}{\partial w_1} \\
\frac{\partial^+ f}{\partial w_1} &= g + w_1 * 0 + \frac{\partial g}{\partial w_1} \\
\frac{\partial^+ f}{\partial w_1} &= g + \frac{\partial g}{\partial w_1}
\end{aligned} \tag{21}$$

Lastly, *TARGET* is $z_5 = (f - \hat{y})^2$ where $\hat{y}$ is the output corresponding to input x.

$$\begin{aligned}
\frac{\partial^+ z_5}{\partial z_i} &= \frac{\partial z_5}{\partial z_1} + \sum_{j>i} \frac{\partial^+ z_5}{\partial z_j} * \frac{\partial z_j}{\partial z_1} \\
\frac{\partial^+ z_5}{\partial z_i} &= \frac{\partial z_5}{\partial z_1} + \frac{\partial^+ z_5}{\partial z_2} * \frac{\partial z_2}{\partial z_1} + \frac{\partial^+ z_5}{\partial z_3} * \frac{\partial z_3}{\partial z_1} + \frac{\partial^+ z_5}{\partial z_4} * \frac{\partial z_4}{\partial z_1} \\
\frac{\partial^+ J}{\partial w_1} &= 0 + 0 * 0 + 0 * \frac{\partial z_3}{\partial z_1} + 2 * (f - \hat{y}) * \frac{\partial z_4}{\partial z_1} \\
\frac{\partial^+ J}{\partial w_1} &= 4 * (f - \hat{y}) * \frac{\partial f}{\partial w_1}
\end{aligned} \tag{22}$$

The point is that the derivative can be calculated in the same order that the network calculates the function. This is explicit in the above equations as $\frac{\partial^+ g}{\partial w_1}$ only depends on the input to *g* which is *x*. $\frac{\partial^+ f}{\partial w_1}$ only depends on the input to *f* which is *g* and the derivative previously calculated for its input g. And $\frac{\partial^+ J}{\partial w_1}$ only depends on the input to *J* which is *f* and the derivative previously calculated for its input *f*.

In finding a gradient, the ordered derivatives of the cost with respect to each weight are found. To find this programmatically, for every input to the network a structure corresponding to every weight is set to zero. For every calculation, this structure is updated by adding the relevant partial derivative information. If two or more paths are taken in a network for a given input or intermediate value, the structure is cloned for each path. This can be applied in a general setting. Multiple branching paths, recurrency, weight sharing, and different activation functions (as long as they are differentiable) can be handled. The backpropagation is reduced to passing the structure appended with the signal through a forward propagation network.

This method is shown to work on a highly complex network solving a non-trivial problem involving several degrees of freedom considering that the transmitter is non-fixed and the grid is two dimensional. It demonstrates that the gradient of a network cost function can be calculated in the order of the forward computation of output. This technique is also useful for any method that uses Jacobian such as Levenberg-Marquardt.

3.5 Matrix Completion and Spectrum Forecasting

Missing data is a general problem in engineering. Either due to uncertainty or cost in taking measurements or noise, limitations in data collection exist. A CR node needs to determine what portions of radio spectrum are available at what times. In this domain missing data presents itself. CRs enabled by software defined radio can only take measurements over limited ranges of frequency at the speed necessary to ensure channels are available over fine resolution windows of time. In [5] and [6], matrix completion is used to address this missing data problem in cognitive radio. Here, matrix completion is used for the frequency-time domain problem of estimating the entire spectrum given only a limited set of measurements. These estimates can be used to guide selection of what frequencies to monitor so that a radio can avoid interfering with primary users.

To extend this idea, RNNs are used to make short-term forecasts given the completed data. The RNN is given an estimate of an entire time series based on the information that the low rank condition provides via matrix completion. Then using the most complete picture, the RNN gives a prediction of future spectrum use. This can help identify spectrum holes that may be used by the cognitive radio network.

Spectrum analyzers perform slow sweeps over a large band of spectrum. Because these sweeps are slow, wide band spectrum analyzers are not used as cognitive radio hardware. The data is useful as a test of frequency-time missing data estimation, as over a given frequency and time range, and at a certain resolution the data is complete. Frequency-time domain matrices of spectrum power measurements from [17] are used. As radios change their operating parameters they collect data from different frequencies at different times. This is considered by masking the data in a block-wise manner to simulate incomplete readings from CRs. The performance of the data matrix completion tool soft-impute and the forecasting tool RNNs together are tested.

We avoiding using the matrix completion as a future prediction tool as in [6]. Assume the rows of a matrix represent different frequencies and the columns of a matrix represent different times. If a column for a given time is partially measured matrix completion should be useful in predicting the missing measurements of that column, given additional incomplete columns. However, if a column for a given time is completely unknown, e.g. for a future time, matrix completion is an inappropriate tool. The intuitive explanation is that a matrix, upon swapping columns, does not change rank or even nuclear norm*, so upon completing a matrix with an unmeasured column, the completed column will not necessarily be related to nearby columns. In other words, temporal dependence cannot exhibit itself in the completion of this unknown column.

A more rigorous way to show this is to consider the singular value decomposition of a matrix that occurs in the soft-impute algorithm. If the last column of the measurement matrix is fully unknown, in the initial iteration the R matrix after step 4 of the algorithm is of the form shown in equation 23. The next step of the algorithm performs singular value decomposition, finding a form of R shown in equation 24.

$$\begin{bmatrix} R_{11} & R_{12} & \dots & R_{1m-1} & 0 \\ R_{21} & R_{22} & \dots & R_{2m-1} & 0 \\ \vdots & \vdots & \ddots & \vdots & 0 \\ R_{l1} & R_{l2} & \dots & \dots & 0 \end{bmatrix} \tag{23}$$

$$\begin{bmatrix} U_{11} & U_{12} & \dots & U_{1l} \\ U_{21} & U_{22} & \dots & U_{2l} \\ \vdots & \vdots & \ddots & \vdots \\ U_{l1} & U_{l2} & \dots & U_{ll} \end{bmatrix} \begin{bmatrix} \Sigma_1 & 0 & 0 & & & & & \\ 0 & \Sigma_2 & 0 & & & & & \\ 0 & 0 & \Sigma_3 & & & & & \\ & & & \ddots & & & & \\ & & & & \Sigma_{r-1} & 0 & 0 & \\ & & & & 0 & \Sigma_r & 0 & \\ & & & & 0 & 0 & 0 & \\ & & & & & & & \ddots \end{bmatrix} \begin{bmatrix} V_{11} & V_{21} & \dots & V_{m1} \\ V_{12} & V_{22} & \dots & V_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ V_{1m} & V_{2m} & \dots & V_{mm} \end{bmatrix} \tag{24}$$

Matrix *U* being orthogonal, after being post multiplying by Σ will form a new matrix, *Y*, with the first columns each being scalar multiples of a unique orthonormal column from *U*. The remaining *m*−*r* columns will be all zero columns.

* It is obvious that column swapping preserves rank. Column swapping also preserves the relaxed nuclear-norm constraint. This is because Schatten-p norms with $p \geq 1$ (nuclear norm being p = 1), are invariant to unitary transformations [33]. A permutation matrix post multiplying a matrix, which results in a column swap, is a special case of a unitary transformation.

$$U\Sigma V^T = YV^T = \begin{bmatrix} Y_{11} & Y_{12} & \dots & Y_{1r} & 0 & 0 \\ Y_{21} & Y_{22} & \dots & Y_{2r} & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & 0 & 0 \\ Y_{l1} & Y_{l2} & \dots & Y_{lr} & 0 & 0 \end{bmatrix} \begin{bmatrix} V_{11} & V_{21} & \dots & V_{m1} \\ V_{12} & V_{22} & \dots & V_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ V_{1m} & V_{2m} & \dots & V_{mm} \end{bmatrix} \tag{25}$$

Setting equations 23 and 25 equal, and considering the last column of both sides, equation 26 follows. Remembering that the first columns of *Y* are scalar multiples of unique orthonormal columns, it follows that they are linearly independent. Since these first columns are linearly independent, the first r elements of the V^T column vector in equation 26 must be zero as shown in equation 27.

$$Y \begin{bmatrix} V_{m1} \\ V_{m2} \\ \vdots \\ V_{mm} \end{bmatrix} = 0 \tag{26}$$

$$V_{mi} = 0 : \forall i \in \{0, 1, \dots r\} \tag{27}$$

This is a key result. Step 6 of the algorithm reduces the absolute value of the diagonal elements of Σ. Then two portions of the "YV^T " decomposition (eq. 25) are sure to remain unchanged. First the *V* matrix is unmodified. Second, at least the last *m*−*r* columns of *Y* will remain all zero columns. This means that the last column of the R^* matrix in step 7 of the algorithm must have a last column that is all non-zeros just as the *R* matrix had. Thus the algorithm can never return (aside from possible computer rounding errors) a C_k for any *k* that has a non-zero final column. Making future predictions in this problem context with the soft-impute algorithm will not work. The proposed supplementary RNN method is used instead for making predictions.

3.6 Spectrum Analyzer Data

The spectrum analyzer data available in this study is in the form of radio spectrum power readings across a wide band of frequencies (in 1500 MHz segments) over a long time span (days). See [17] for a description of the dataset. This data is used here to test a missing data estimation algorithm called Soft-Impute for use in CR spectrum prediction. The data is analyzed in portions as matrices (rows of discrete times, and columns of discrete frequencies). Each matrix represents a history of spectrum power data for a CR system.

3.7 Radio Network Tensor Model

An important model used in this work describes a network of radio nodes transmitting and receiving signals. The transmitters may be primary and/or secondary users. The received signals of interest are the set of signals received in the CR network of secondary users. These are the only signals that a fusion center can utilize to understand the radio environment.

Given one transmitter and receiver pair (*i*, *j*), *i* indicating one of *I* primary or secondary user radios transmitting and *j* indicating one of *J* secondary user radios receiving in a space. An inverse square law [34] governs the power decay with distance as in equation 28, where *k* is a specific time, and *l* is a specific frequency. Φ_{ijkl} is the power at receiver *j* from transmitter *i*, and θ_{ikl} is the power emitted from transmitter *i*. λ_l is the wavelength at frequency *l*.

$$\varphi_{ijkl} = \theta_{ikl}\left(\frac{\lambda_l}{d_{ij}}\right)^2 \tag{28}$$

As the signal is traveling in air and is traveling along multiple paths, fading exists dependent on the time *k*, frequency *l*, and propagation environment in the space containing transmitter *i* and receiver *j*. Gaussian additive white noise also exists [35]. This Johnson-Nyquist noise has a signal power proportional to the temperature and the bandwidth range being considered [36, 37]. This is shown in equation 29 where *T* is temperature, ΔB is bandwidth range, and K_b is Boltzmann's constant.

$$P_{noise} = K_b T \Delta B \tag{29}$$

For example, consider a 22MHz wide band at 2.4GHz (802.11b/g/n) Wi-Fi signal with ΔB being 22MHz in the above equation, the thermal noise at room temperature of 300K is 9.11e-4 Watts or -100.4 dBm. To model this noise signal, the noise is considered white noise over the bandwidth of interest. Therefore, the periodogram* reduces to the modulus squared of a sum of

* See [38] for a description of periodograms. See [39] for a description of χ^2 distribution and its relation to Gaussian distribution.

independent complex Gaussian terms. To simplify this, approximate the sum of independent complex Gaussians as itself a complex Gaussian, and approximate the modulus squared of this complex Gaussian as a χ^2 distribution. This more realistic model is represented in equation 30 with a fading coefficient f_{ijkl} and χ^2 noise term ε_{ijkl}[*].

$$\varphi_{ijkl} = f_{ijkl}\theta_{ikl}\left(\frac{\lambda_l}{d_{ij}}\right)^2 + \epsilon_{ijkl} \tag{30}$$

Lastly, power gains exist at the transmitter antenna and the receiver antenna. These may vary from radio to radio and the completed model is shown in equation 31 with φ_{ijkl} the power measured after antenna gain g_{jl} for frequency l at receiver j, and θ_{ikl} the power emitted before antenna gain g_{il} for frequency l at transmitter i.

$$\phi_{ijkl} = g_{jl}f_{ijkl}(g_{il}\theta_{ikl})\left(\frac{\lambda_l}{d_{ij}}\right)^2 + \epsilon_{ijkl} \tag{31}$$

Equation 4 will be rewritten as equation 32 since only the power in the environment after the transmitter antennas is useful to know in a CR network. Θ_{ikl} represents the power after at each transmitter after the antenna gain.

$$\phi_{ijkl} = g_{jl}f_{ijkl}\Theta_{ikl}\left(\frac{\lambda_l}{d_{ij}}\right)^2 + \epsilon_{ijkl} \tag{32}$$

Substituting equation 3 into equation 32 can be rewritten as equation 33, with ε_{ijkl} scaled noise.

$$\varphi_{ijkl} = \frac{\phi_{ijkl}}{g_{jl}} = f_{ijkl}\Theta_{ikl}\left(\frac{\lambda}{d_{ij}}\right)^2 + \varepsilon_{ijkl} \tag{33}$$

The power Φ_{jkl} received from the sum of each transmitted power over all I is given in equation 34 with the noise variable E_{jkl} remaining as approximately χ^2 of one degree of freedom (as this reflects noise power at one receiver).

[*] Matlab simulation comparing simulated periodogram on white noise and χ^2 found nearly identical PDFs.

$$\Phi_{jkl} = \sum_{I} f_{ijkl}\Theta_{ikl}\left(\frac{\lambda_l}{d_{ij}}\right)^2 + E_{ijkl} \tag{34}$$

This means that given measurements at *J* receivers, the signals arriving at the antennas before receiver gain Φ_{jkl} may be found by dividing by the gains g_{jl} at each antenna. This could be done if the frequency response of each receiver's antenna is known to the fusion center. Perhaps even more feasible is that each receiver *j* only reports Φ_{jkl}, which could be performed in a radio's hardware or software.

Further development of the model is done for blind signal separation. F_{ijkl} can be relatively constant over small ranges of frequency and for short periods of time. Let (K_1, L_1), (K_2, L_2), ... denote a partitioning of the frequency×time domain where each $\Gamma = (K\gamma, L\gamma)$ in the partition is a region with f_{ijkl} constant over *k* and *l*. The model can now be written in matrix form by squeezing the tensors (where grouping of indices in parenthesis indicates preferred interpretation as a single dimension). Equation 35 shows the model over the range of a particular Γ.

$$\Phi_{j(kl)} = \sum_{I} f_{ij\Gamma}\Theta_{i(kl)}\left(\frac{\lambda_l}{d_{ij}}\right)^2 + E_{ij(kl)} \tag{35}$$

Blind signal separation is described in section 2.10 for a set of readings $\Phi_{j(kl)}$ (with receiver gains already divided). The latent dimension in the separation is the transmitters i. A decomposition is found as given in equation 36 and in matrix form in 37 where *P* is the measured power with each row *j* already divided by the gain at receiver *j* and with each group of columns corresponding to each *l* already divided by λ_l^2.

$$\frac{\Phi_{j(kl)}}{\lambda_l^2} = \sum_{I} a_{ji} b_{i(kl)} \tag{36}$$

$$P_\Gamma = A_\Gamma B_\Gamma \tag{37}$$

In addition to the above model decomposing nicely as a NMF, the matrices of received power are also in a low rank form. This is true when the number of channels with active transmitters are low relative to a given number of frequencies and times collected by a receiver for analysis. This allows for the missing data problem to be solved with matrix completion algorithms for frequency-time spectrum completion. As tuning pruning requires one matrix to be sparse, this model can be factored using that method. The second matrix factor is assumed to be sparse meaning that spectrum is not transmitted over most frequencies and times. Section 4.4 describes how the factorization is found using tuning pruning.

3.8 Radio Network Simulation

A simulation was used for blind signal separation testing. This simulation is derived from the equation 36 that concludes the formulation of the tensor model in section 3.7. The formula considers the radio scene observed by the receiver CR as a matrix. Each element of the measured matrix $M_{j\gamma}$ is the power at receiver j at frequency and time γ. The value of λ gives a factor on columns of M that can be removed since each receiver is aware of each frequency it is measuring. Fading is added as beta distributed noise. Setting the β parameter as 2 and varying the α parameter between 10 to 80 gives fading close to 1, with stronger fading given with lower α. This distribution is of course a heuristic choice, but it is a simple description that is easily manipulated with one degree of freedom meeting the need by only modifying the β parameter and keeping α constant. Figure 10 gives the shapes of the range of distributions applied.

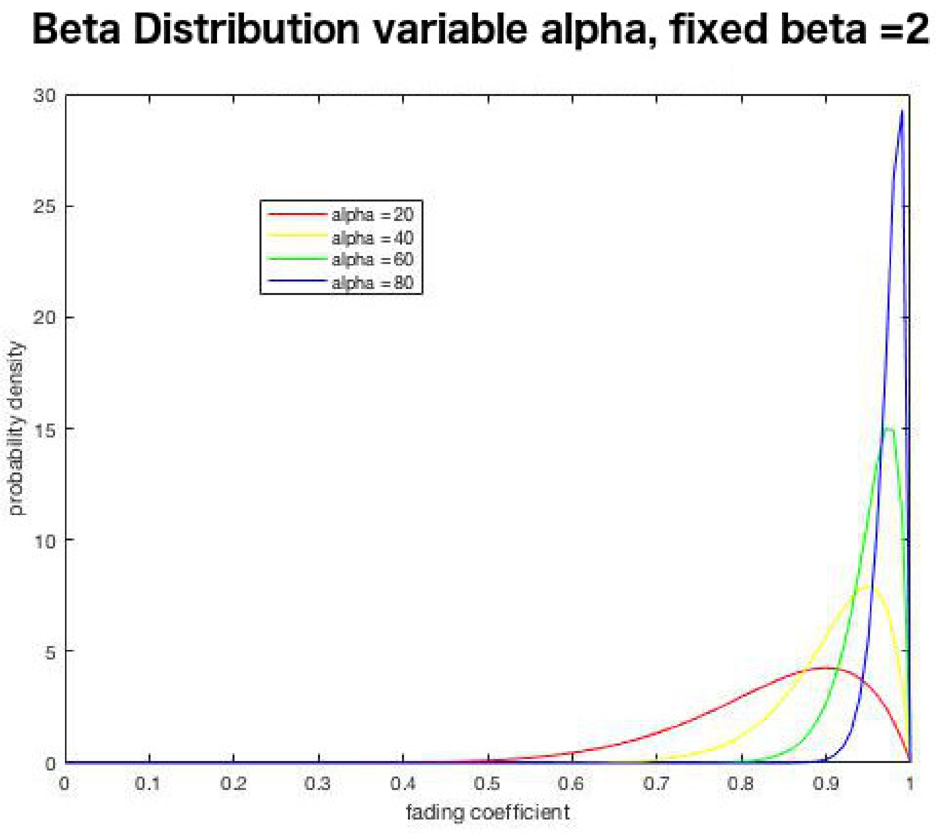


Figure 10: Various Beta Distributions modifying 1 Degree of Freedom (α)

The element A_{ji} is the inverse of the distance squared between receiver j and transmitter i with a fading factor. The element $B_{i\gamma}$ is the transmitted power from transmitter i at frequency and time γ. The transmitted power was modeled with gains varying from 10 to 40 milliwatts. The channel activity is sparse with activity likelihood at 25% per frequency/time slot. Transmission gains vary over time and frequency to generalize the simulation. The spatial area is 1000x1000 meters. There are 5 transmitters and 100 receivers. There is a set of 100 transmission frequency/time pairs.

CHAPTER 4
RESULTS

4.1 Spatial Spectrum Prediction Results

The CSRN structure used in this experiment is based on experiments on previous problems such as maze traversal and image processing. As described earlier there are two external inputs and four neighbor inputs. These are fed into a set of 5 recurrent neurons. As these neurons are recurrent, they take input from their values calculated on previous iterations. The neighbor values are passed from one recurrent neuron in each cell. The external output of the network is taken directly from another recurrent neuron in each cell. The weights used in each cell are shared. This is possible given the symmetrical topology of the problem, where each cell is equivalent in the sense that no spatial locations are preferred and the geometrical grid is homogeneous. This is helpful in training as a reduced number of parameters are more easily trained. The number of input samples per grid is 10. This means that if a user wants to use the network with a different transmitter location, they would want to measure the signal received at 10 locations.

The CSRN is trained for the spatial spectrum prediction problem using backpropagation and UKF for comparison. Figure 11 shows the training error as a function of training time for both methods. Here, the x-axis is time in seconds for both to get a good comparison. Both algorithms are parallelized. CSRN runs multiple inputs in parallel to generate a Jacobian. UKF runs matrix operations in parallel. This is just one example training run. Tables 2 any 3 show testing error for 3 separate training runs. For each trial number in tables 2 and 3 the same training and testing sampling is used for backpropagation and UKF to get a fair comparison between methods. The error in the figure and the tables is mean absolute error. This is an average of the absolute value of the difference over all training targets and predictions. UKF is run for 50 epochs and backpropagation is run for 500 epochs. The time to train is comparable. Given the range of the data from about -60 dBm to -30 dBm the error of the prediction is small. This may be due to high sampling of 10 out of 25 locations used as input to the network.

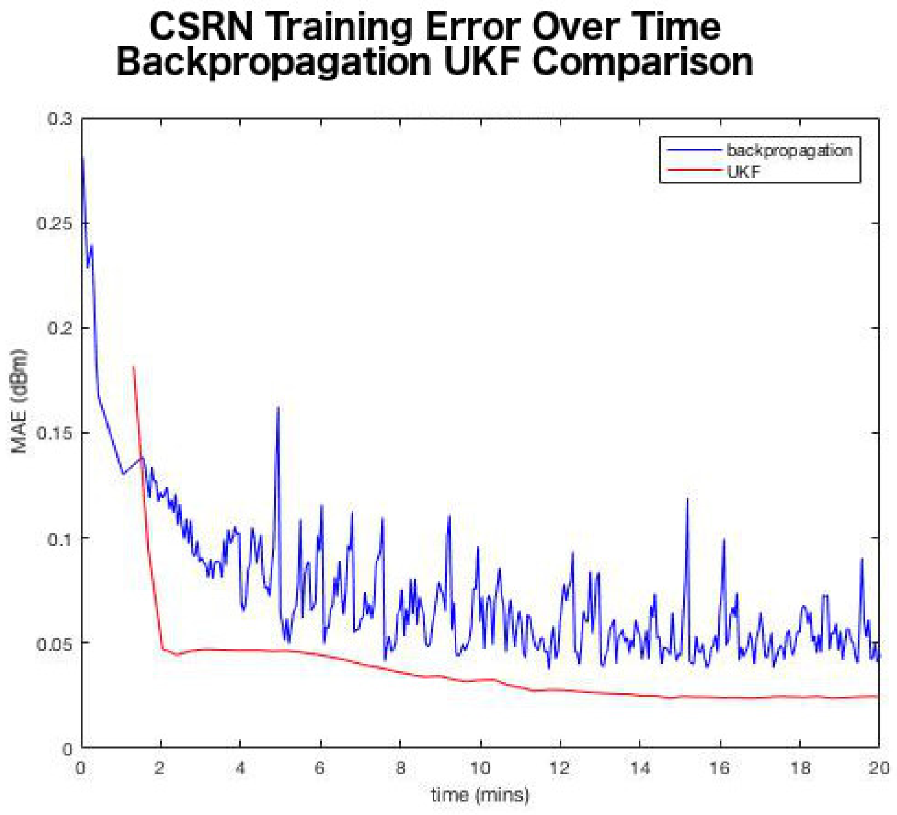


Figure 11: UKF and Backpropagation Training Error over Time

Table 2: CSRN UKF Testing Results

	transmitter location 1	transmitter location 2	time (secs)
Trial 1	0.0839 dBm	0.0703 dBm	1012
Trial 2	0.0419 dBm	0.0385 dBm	1194
Trial 3	0.0373 dBm	0.0712 dBm	1077

Table 3: CSRN Backpropagation Testing Results

	transmitter location 1	transmitter location 2	time (secs)
Trial 1	0.0758 dBm	0.1049 dBm	1077
Trial 2	0.0654 dBm	0.0463 dBm	1081
Trial 3	0.0777 dBm	0.0509 dBm	1012

4.2 Frequency-Time Spectrum Completion Results

In this study a subset of the real data matrices was used to determine the completion average absolute error as given in equation 38 as a function of the value of λ_1 (see fig. 4).

$$\text{mean absolute error} = \frac{\sum_{(i,j)\in L^c} |C_{i,j} - M_{i,j}|}{|L^c|} \tag{38}$$

Figure 12 shows that when running the soft-impute algorithm, decreasing λ_1 likewise decreases the minimum that error eventually converges to. However, as λ_1 becomes very large, the performance gain becomes insignificant relative to the time cost of running the algorithm. Notice this is plotted on a log scale implying the small gain for continuing to increase λ_1. Therefore, for subsequent results in this work, a value of 1/32 σ (the first singular value of the matrix to be completed) is used as λ_1.

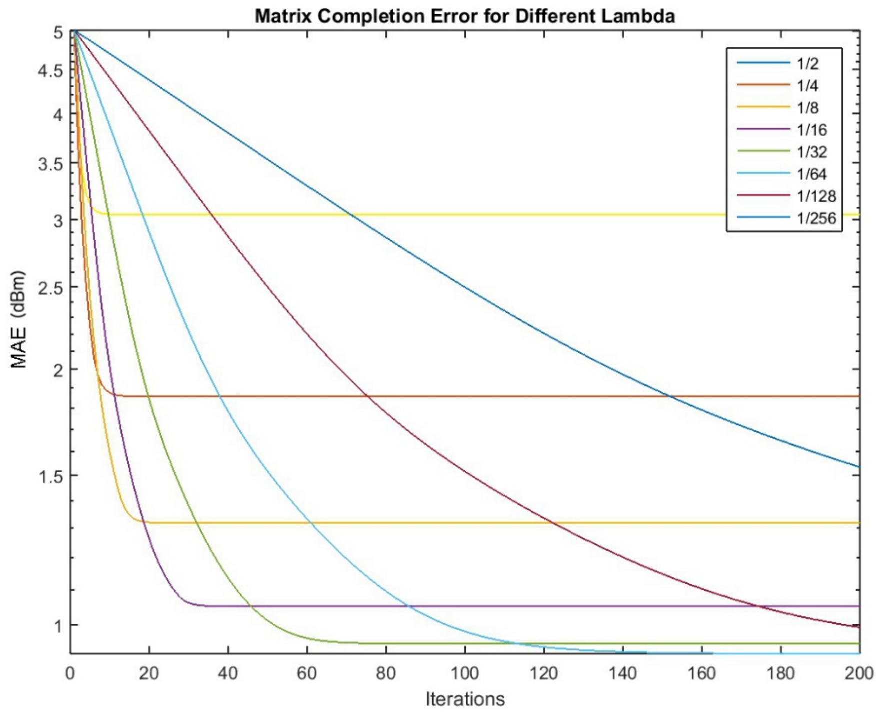


Figure 12: Parameterization of Matrix Completion

Inspection was then used to determine what value of ε causes the algorithm to run until the error converges to a minimum ($\varepsilon = 10^{-6}$). No reduction in error was observed for repeated trials, so k is set to 1. This method was followed for parameterization of the matrix completion used for the simulation data with multiple radios. In this work, the final matrix to be used for input to the RNN is given by using the values of C where the matrix is to be completed, and otherwise using the known values from M where the locations have measurements given (see equation 38).

The correlation matrix shows that there are dependences between even separated frequencies; however, these are less significant that between similar frequencies. An experiment was conducted based on this premise. The matrix of real data to be completed was partitioned into blocks of different frequency spans. Then matrix completion was performed on the individual blocks, to determine the optimal partitioning. This technique was not found to give any improvement in matrix completion performance, and so further elaboration is omitted.

The soft-impute algorithm is tested on the 770 MHz band radio data. Television bands are isolated since there exists strong frequency correlation among these bands. Figure 13 shows the CDF of the average absolute error for completing the matrix. The range of errors is relatively small compared to their value, where the vertical region of the graph indicates that most error is between 1 and 1.2 dBm. The error is much lower than the standard deviation of the data which is around 7dBm.

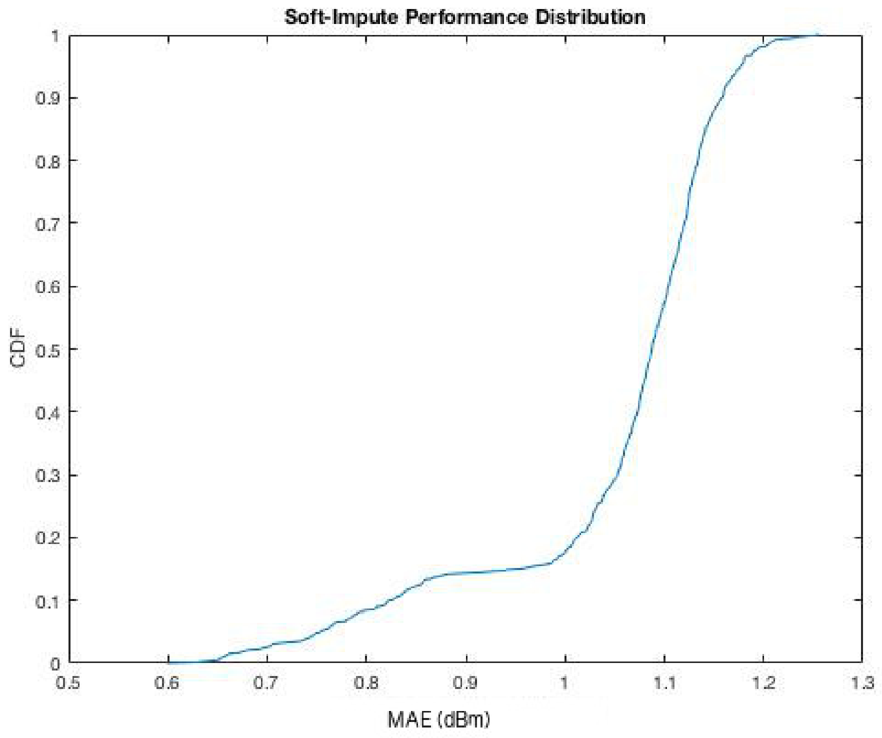
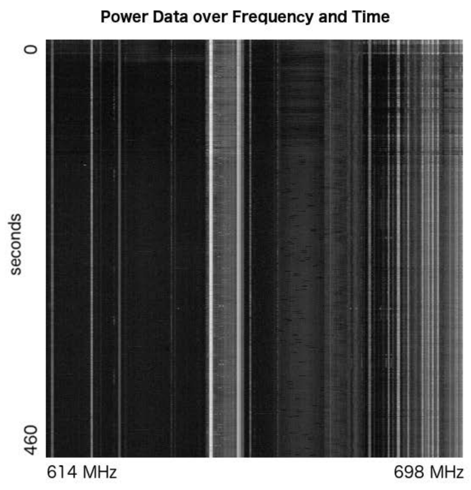


Figure 13: Soft-Impute Performance

Figure 14 visualizes the data and the masking applied to make missing data. Figure 15 visualizes the data after the soft-impute algorithm has estimated the missing data. The dark and light regions are lower and higher power measurements respectively.



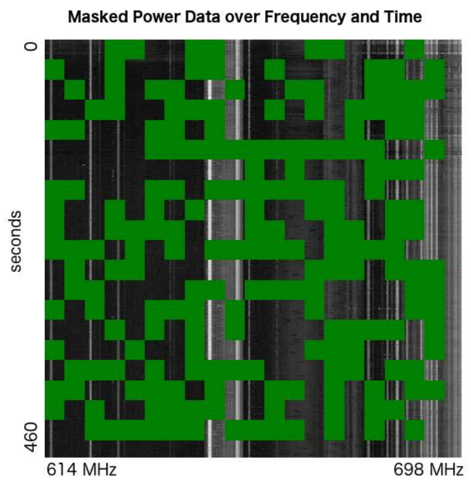


Figure 14a: Original Power Data

Figure 14b: Masked Power Data

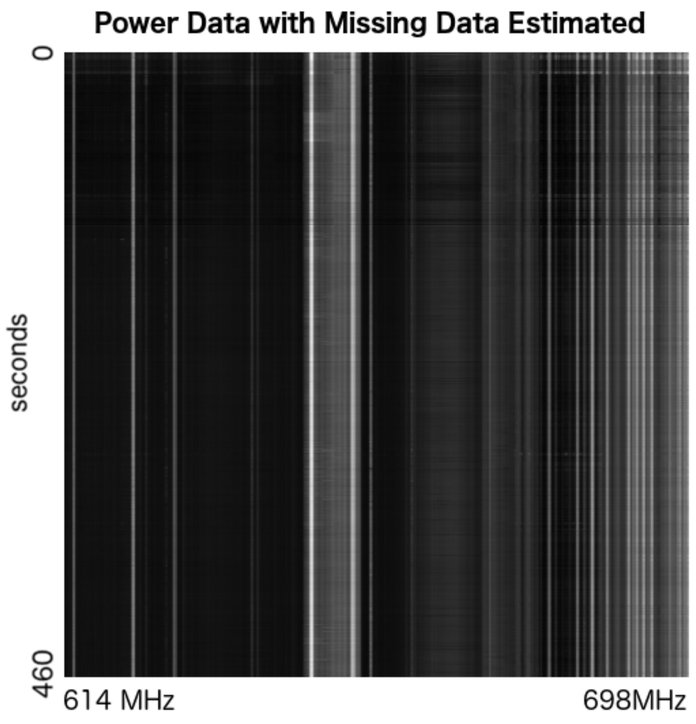


Figure 15: Completed Frequency-Time Data

4.3 Short Term Spectrum Forecasting Results

For the short-term spectrum forecast given the completed matrix, an unscented-Kalman filter training was applied to train the Elman RNN. The signal is noisy, and its prediction does not admit to this training technique. The backpropagation algorithm is able to train the Elman RNN on this data set. The network is trained and validated on each band of interest to determine the optimal number of hidden neurons.

For forecast performance two metrics are used. Mean absolute error (MAE) is given in eq. 39 and is used to measure absolute error in terms of the units of the signal. Mean relative error (MRE) is given in equation 40 and is useful as a percentage error relative to the signal magnitude.

$$MAE = \frac{\sum_{i=1}^{n} |\hat{y}_i - y_i|}{n} \tag{39}$$

$$MRE = \frac{\sum_{i=1}^{n} |\frac{\hat{y}_i - y_i}{y_i}|}{n} \tag{40}$$

Figure 16 shows the validation on 100 and 1000 training epochs in MAE. It is shown that the validation error is converging and that the optimal choice of hidden layer size is 10 units. This is likely due to the ability of a smaller network to generalize better on the validation data.

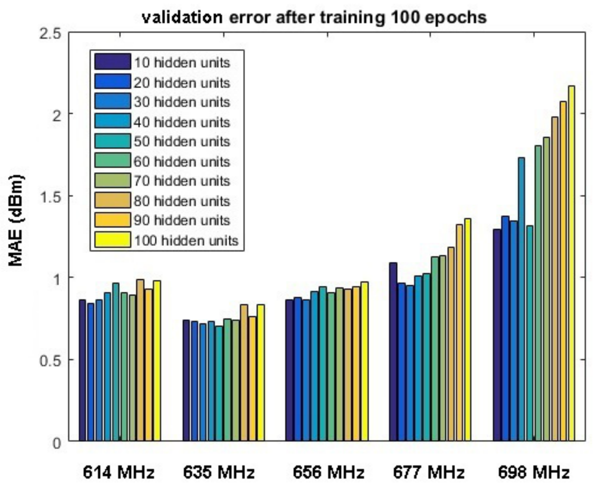


Figure 16a: Validation Error 100 epochs

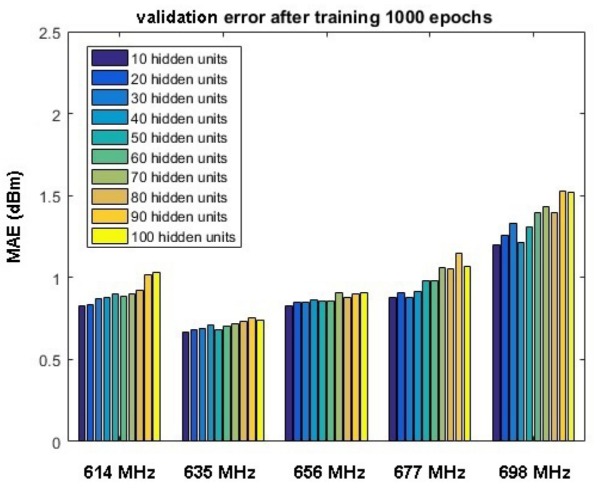


Figure 16b: Validation Error 1000 epochs

RNN forecasts accuracy is shown in figure 17 and figure 18. The former is a CDF for the errors in band given in MAE. The latter is forecast MAE as a function of the length of time into the future that the forecast is being made. The prediction error reaches a maximum as the forecast length becomes longer. This is because the forecast approaches a random guess as it becomes more difficult to predict further into the future. The error here reflects the variance in the data, with some changes having higher or lower variability. It is also clear that the error for some short term forecasts is quite low considering the variance of the signal.

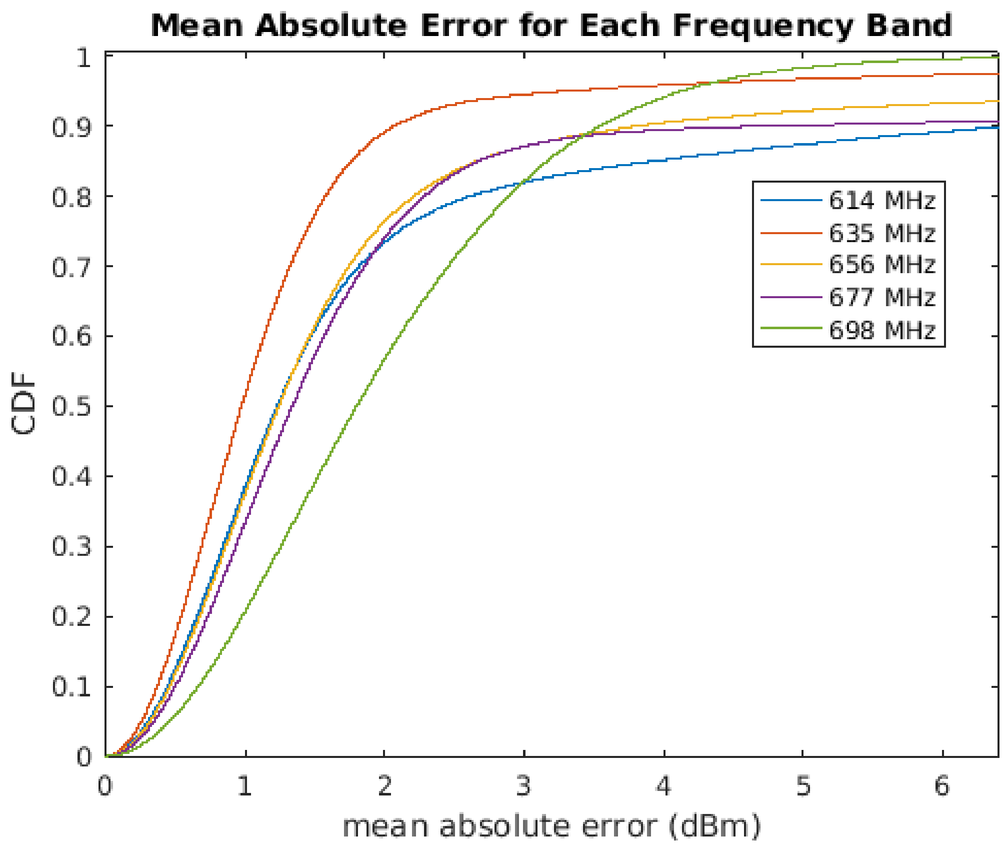


Figure 17: CDFs for RNN Forecasts in Each Band

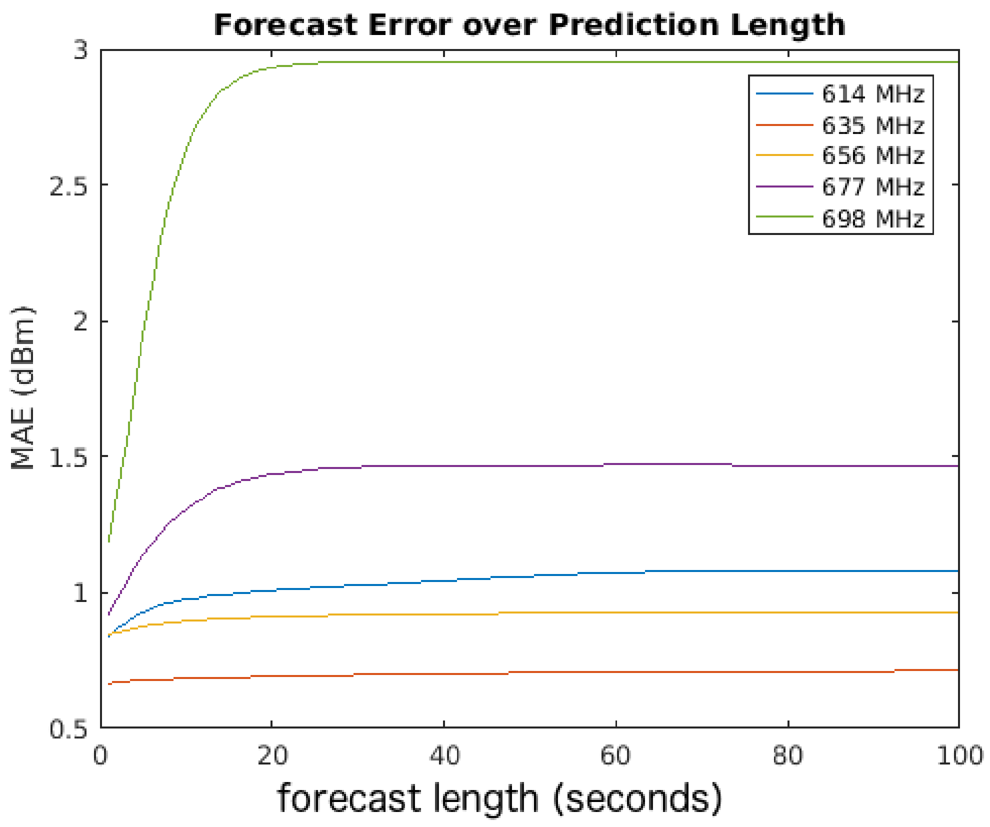


Figure 18: RNN Forecasts in Each Band over Forecast Length

Thermal baseline noise is a function of the temperature [36, 37]. However, in this study it is assumed constant in each band over time. This is because in the studied TV bands, the man-made signal fluctuations are slow, and it is difficult to separate them from small thermal noise that also fluctuates slowly. K-means clustering with $k = 2$ was used to separate the ground truth readings into 2 clusters representative of noise floor and channel occupancy. Channel status is predicted by assigning each prediction from the RNN to either cluster (a CR could easily determine the noise floor from temperature, or from measurement history). The receiver operating characteristic curves (ROC) are given for sensitivity and specificity in figure 19. The sensitivity is true positive to total positive ratio and the specificity is true negative to total negative ratio. Additional measures of fitness of the time series prediction include mean absolute error and mean relative error; these are given in table 4.

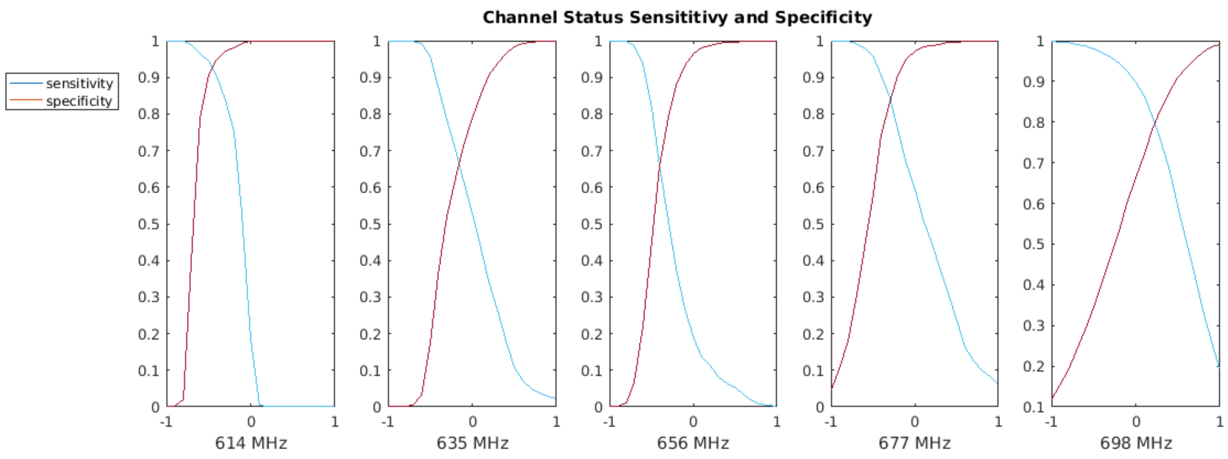


Figure 19: RNN Sensitivity And Specificity ROC Curves for Each Band

Table 4: RNN Forecast Statistics

Band	MRE(1/5)	MAE(1/5)
614 MHz	0.00747/0.00829	0.837/0.929 dBm
635 MHz	0.00588/0.00599	0.929/0.677 dBm
656 MHz	0.00755/0.00782	0.677/0.876 dBm
677 MHz	0.00864/0.01074	0.917/1.140 dBm
698 MHz	0.01323/0.02148	1.140/1.949 dBm

MRE - mean relative error for forecasts 1 or 5 time steps ahead
MAE - mean absolute error for forecasts 1 or 5 time steps ahead

4.4 Leave-One-Out Blind Signal Separation for Security

Equations 35, 36, and 37 are rewritten here:

$$\Phi_{j(kl)} = \sum_I f_{ij}\Gamma\Theta_{i(kl)}\left(\frac{\lambda_l}{d_{ij}}\right)^2 + E_{ij(kl)} \tag{35}$$

$$\frac{\Phi_{j(kl)}}{\lambda_l^2} = \sum_I a_{ji} b_{i(kl)} \tag{36}$$

$$P_\Gamma = A_\Gamma B_\Gamma \tag{37}$$

Because of the sparsity of $\Theta_{i(kl)}$ in simulation, the algorithm is able to determine A_Γ and B_Γ uniquely up to permutation of columns and rows respectively (corresponding to the latent variable ordering) and relative scaling by a constant. A_Γ is proportional to the ratio $f_{ij\Gamma}/d_{ij}^2$. B_Γ is proportional to $\Theta_{i(kl)}$. For each secondary user that is transmitting in addition to receiving, a source will be known a priori before the blind signal separation. They may report their transmission powers after antenna gain $\Theta_{i(kl)}$. These reports can be used after the blind signal separation to validate the results, detect malicious users, and to determine the actual scaling of A_Γ and B_Γ, so that A_Γ is equal to $f_{ij\Gamma}/d_{ij}^2$ and B_Γ is equal to $\Theta_{i(kl)}$.

The final problem in the decomposition is to find $f_{ij\Gamma}$ and d^2 given $f_{ij\Gamma}/d_{ij}^2$. This is accomplished by observing A over different partition domains Γ. F_{ijkl} may be near one for many values of (i, j, k, l) corresponding to absence of fading for a certain path, frequency, and time instant. If many observations are taken of $f_{ij\Gamma}/d_{ij}^2$, the maximum value for each (i, j) pair over the set of observations can be used as the value of $1/d_{ij}^2$. This will be useful for localization of radios. Less involved formulations of the matrix decomposability of the radio problem were described from [26, 27]. In both approaches the derived model assumes that the latent dimension of the decomposition is known. This is not assumed when the decomposition is recovered in this work.

In this work, the following leave-one-out scheme is proposed to counter the SSDF attack. Reports from N radio receivers are collected at the fusion center. A blind signal separation is iteratively performed on $N-1$ of the reports. Tuning-pruning can be used for signal separation. If the algorithm can find a solution with small error, then the $N-1$ radios are giving valid readings. If for other sets of radio reports the NMF error is large, then the radio that is common to all other sets is flagged as corrupt. If all sets of reports give high NMF error, 2 malicious nodes may be present and the process is repeated with all $N-2$ radio reports. If a malicious user is transmitting it can also be localized using the trilateration approach from the NMF decomposition. Let the locations of each node in the CR network be known as may be collected using GPS. This is a mild assumption and is used in trilateration. In the next section, this proposed security idea is tested using the simulation described in section 3.8. The decomposition step of the algorithm is performed there.

4.5 Blind Signal Separation Results

Two implementation changes are made for the tuning pruning algorithm. For the gradient descent step the update of 25 times per matrix before continuing to update the other matrix was not found to perform well. Instead, the update was only run once, and then the update was run on the other matrix, a single step alternation. Also, to simplify the convergence criterion, and to get consistent latent dimension (number of transmitters) selection, the algorithm was run until the latent dimension found stabilized.

It is noticed in performing the simulation for blind signal separation that there are restrictions on the possible locations of the receivers. Given a relatively small range of distances from transmitters to receivers, the power received may vary widely. This is due to the inverse square term present in equation 35 and is rewritten below in equation 38, where d_{ij} is the distance between a transmitter i and receiver j.

$$\frac{1}{d_{ij}^2} \tag{38}$$

For transmitters that are far from any receivers, the noise is a more dominant effect, and these transmitters are not discovered using the receivers. The other case of transmitters too close to receivers is forced to be absent from the simulation. If this was allowed, the algorithm sees other sources as absent (or zero) due to the dominant signal introduced because the algorithm attempts to present a sparse solution. This restriction is considered appropriate, as a radio network optimally has nodes highly separated in space.

At first, the tuning pruning algorithm did not work when the simulation was run in the original domain. This is because distance squared function generates some strong outliers relative to the majority of the data when the transmitters are close to receivers. The log of the data is taken and then negated to retain the non-negative property. This log domain data is shown useable by the tuning pruning algorithm. Figure 20 shows the tuning pruning algorithm finding the correct latent dimension of the model over training time. The top graph is the error as the algorithm converges. The bottom graph shows the latent dimension of the decomposition at each epoch. The algorithm consistently stabilizes error when the correct latent dimension is found.

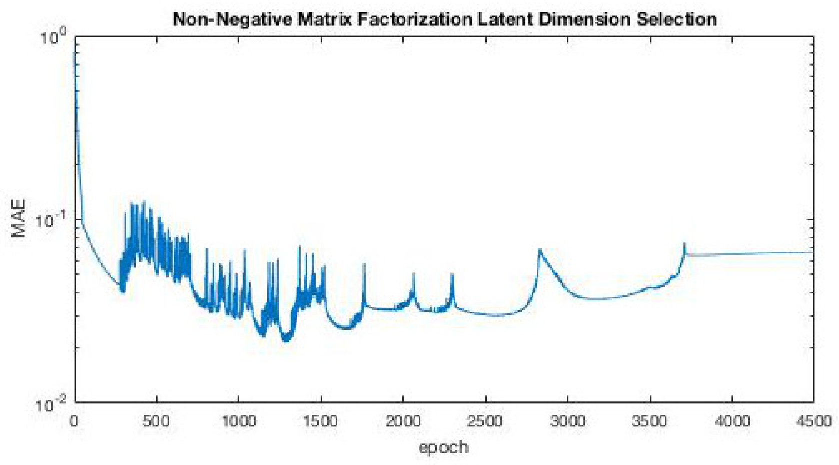


Figure 20a: Log Domain Non-Negative Matrix Factorization Error

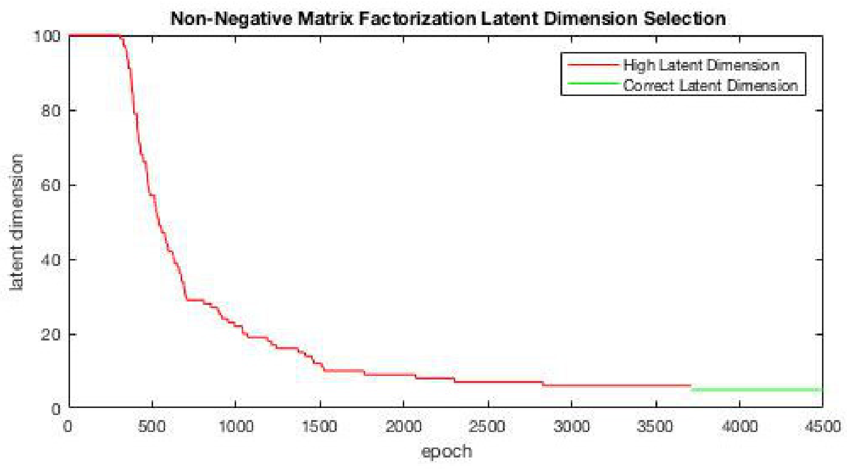


Figure 20b: Log Domain Non-Negative Matrix Latent Dimension

Because having receivers very close to transmitters is not a realistic scenario, this case can be discarded. This is performed in the simulation by filtering out the randomly generated radio network spatial configurations having this quality. After this is done, the factorization works for the original domain. Figure 21 shows the error over training time. The algorithm error doesn't stabilize; instead, it is at a minimum for the correct latent dimension. The correct latent dimension in this original domain is decided as no lower latent dimensions are output after the correct latent dimension is found.

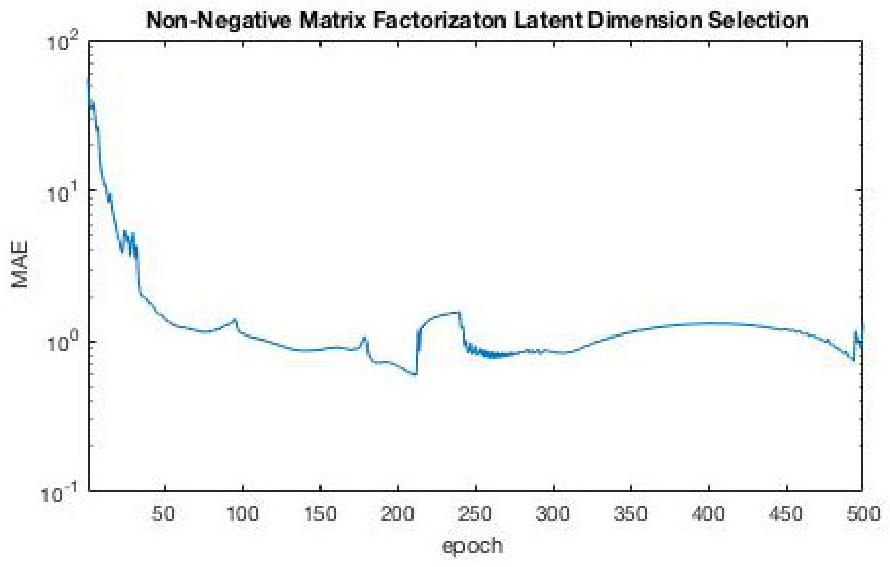


Figure 21a: Original Domain Non-Negative Matrix Factorization Error

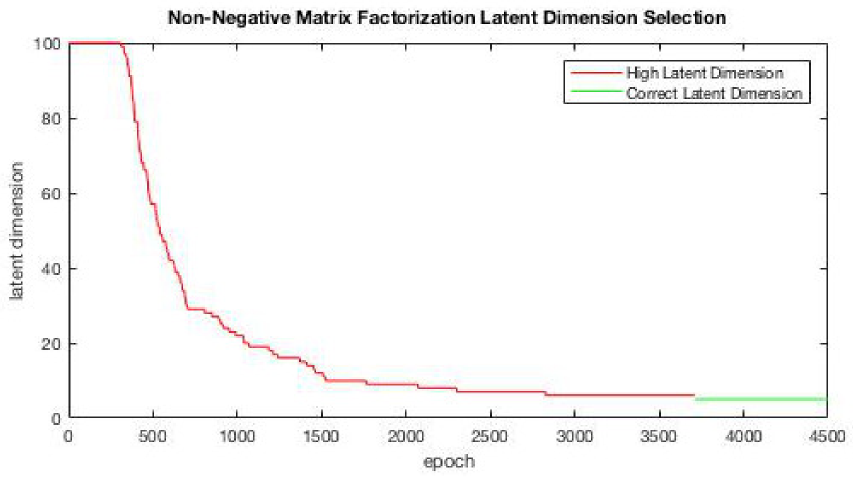


Figure 21b: Original Domain Non-Negative Matrix Factorization Latent Dimension

To compare decomposition with the simulated matrix product the following approach is taken. Normalize the *A* decomposition matrix to the same range as the *A* simulation matrix. Then sort the rows of the decomposition *A* by lowest Euclidian distance from the rows of the simulation *A*. The columns of the *B* decomposition are sorted correspondingly. The *B* decomposition is also scaled to the same range as the *B* simulation. Three metrics are used to compare the decomposition to the simulation. One is the Frobenius norm of the difference in the *A* decomposition and simulation. Another is the Frobenius norm of the difference in the *B* decomposition and simulation. These two metrics may give different results as B is the sparse factor. The third metric is the Frobenius norm of the difference in the simulation matrix product and the product of decomposed *A* and *B*. Tables 5-7 show the three metrics averaged over a series of 100 simulations. Two sources of noise are then added to the simulation to test the robustness of the algorithm. The beta noise varied and the χ^2 transmission noise varied. Figure 22 shows a qualitative representation of the decomposition accuracy with neither noise added to the simulation. Figure 22a is a row from the sparse transmitter power matrix. Notice that many values are 0 where no transmission occurred. Figure 22b is a column from the pathloss matrix. Notice that the values vary widely as the pathloss is inverse quadratic. In a few cases, the number of transmitters is incorrect, and these trials were disregarded when the performance metrics were evaluated. These were typically involving transmitter positions that were too close to each other. This would make the matrix appear lower rank, so in this case the lower latent dimension is expected. Note the close transmitter situation is not typical in real scenarios. There is one additional situation observed. Sometimes the algorithm would overestimate the rank by 1. Again this may be due to radios being placed to closely.

Table 5: Matrix Factorization Performance Decomposition Product Error

β / χ^2	no fading	$\alpha = 80$	$\alpha = 40$	$\alpha = 20$	$\alpha = 10$
no thermal noise	17.024($\times 10^{-3}$ for all)	25.757	22.682	36.551	63.258
mean = 5% power	32.687	33.649	39.234	46.582	70.885
mean = 10% power	62.061	58.873	61.193	70.920	81.279
mean = 15% power	90.180	87.136	86.803	95.544	102.317
mean = 20% power	125.287	114.632	118.544	112.089	127.289

Table 6: Matrix Factorization Performance *A* Decomposition Factor Error

β / χ^2	no fading	$\alpha = 80$	$\alpha = 40$	$\alpha = 20$	$\alpha = 10$
no thermal noise	4.080($\times 10^{-2}$ for all)	4.399	3.876	3.828	5.311
mean = 5% power	3.262	3.273	3.304	3.651	5.419
mean = 10% power	3.640	3.542	3.667	4.529	5.548
mean = 15% power	4.117	3.784	4.680	5.751	6.258
mean = 20% power	5.949	5.240	5.624	5.206	7.642

Table 7: Matrix Factorization Performance B Decomposition Sparse Factor Error

χ^2 \ β	no fading	$\alpha = 80$	$\alpha = 40$	$\alpha = 20$	$\alpha = 10$
no thermal noise	1.789($\times 10^{-1}$ for all)	1.940	1.789	1.656	2.141
mean = 5% power	1.479	1.543	1.460	1.625	2.108
mean = 10% power	1.495	1.564	1.497	1.802	2.257
mean = 15% power	1.647	1.564	1.819	2.150	2.305
mean = 20% power	2.052	2.070	1.953	2.667	2.153

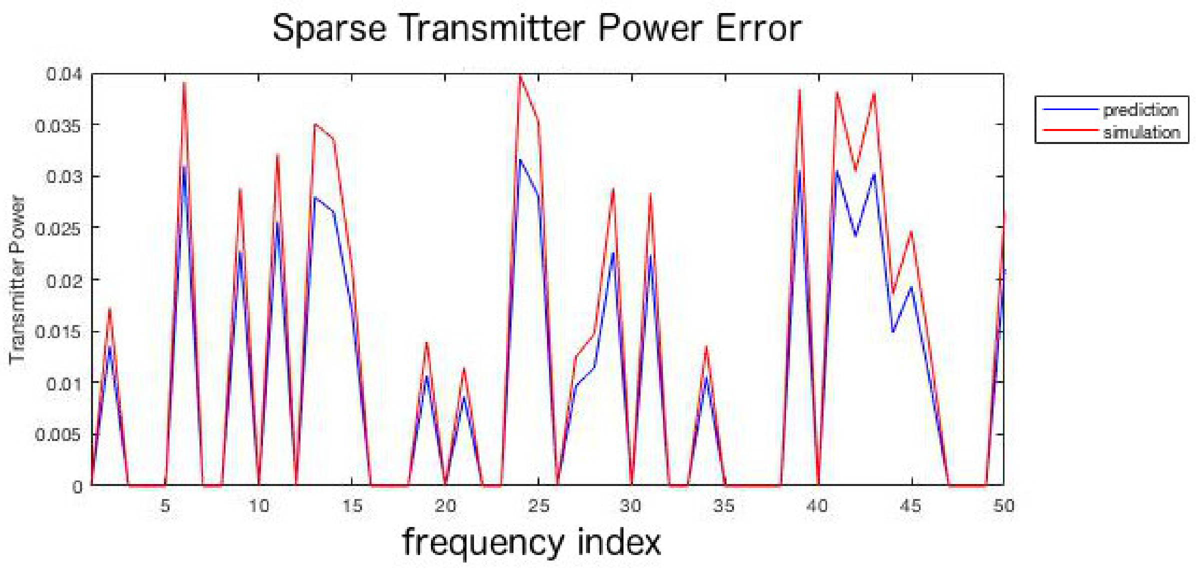


Figure 22a: Visualization of Non-Negative Matrix Factorization A Result

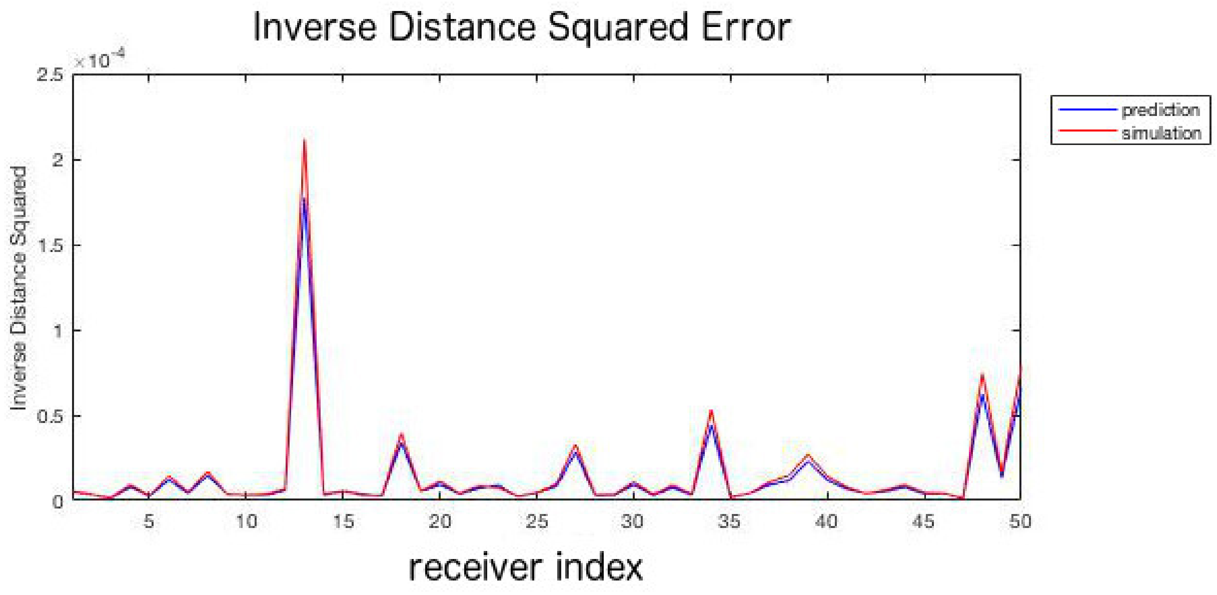


Figure 22b: Visualization of Non-Negative Matrix Factorization B Result

4.6 Blind Signal Separation Algorithm Generalization

As an alternative solution to the tuning pruning algorithm, a geometric constraint is applied to find a unique solution to the blind signal separation problem. In previous works, see [26] for example, localization is posed as a problem to follow any matrix factorization. In other words the matrix completion is performed to give information about the distances between transmitters and receivers. The matrix completion does not, however, consider the constraint that only certain distance combinations are realizable. This naturally constrains the matrix completion and helps produce a unique decomposition. A sum of squares of the difference in the given matrix and its decomposition is a cost function for the problem. To encourage a unique solution a constraint can be added as follows. The reference node trick was used in section 2.11 to find a linear solution to the location of each transmitter given a decomposition. The least squares residuals reflect the plausibility of the solutions to each linear system. Equation 39 shows the form of the residual.

$$||A\hat{x}_k - b_k|| \tag{39}$$

By substituting the form of the solution, equation 40 is found.

$$||A(A^T A)^{-1} A^T b_k - b_k|| \tag{40}$$

This can be rewritten as 41.

$$||(A(A^T A)^{-1} A^T - I)b_k||_2^2 \tag{41}$$

The sum of each of these "*k*" equations can be used as a penalty term added to the cost function as shown in 42, where *X* is the measured matrix, and elements of *U* are inversely proportional to distances squared from transmitter to receiver, and elements of *V* are power emitted from transmitters at different times.

$$J = ||X - UV||_F^2 + \lambda \sum_k ||(A(A^T A)^{-1} A^T - I)b_k||_2^2 \tag{42}$$

Gradient descent is applicable and the derivative of this penalty is straightforward since terms involving *A* are constant with respect to the decomposition *UV*. Equations 43 and 44 show the partial of the cost function with respect to elements of U and V. Gradient notation is omitted for 43 as it would be considerably less compact.

$$\frac{\partial J}{\partial U_{mk}} = \sum_{n=1}^{N} V_{nk} \sum_{\kappa=1}^{K} (U_{m\kappa} V_{\kappa n} - X_{mn}) + \frac{\partial}{\partial U_{mk}} \left(\lambda \cdot \sum_{k} \|(A(A^T A)^{-1} A^T - I) b_k\|_2^2 \right) \tag{43}$$

$$\frac{\partial J}{\partial V_{kn}} = \sum_{m=1}^{M} U_{mk} \sum_{\kappa=1}^{K} (U_{m\kappa} V_{\kappa n} - X_{mn})$$
$$\nabla_V J = U^T (UV - X) \tag{44}$$

Derivation of the partial not evaluated in equation 43 is in equation 45 with $(A(A^TA)^{-1}A^T - I)$ written as A^* to condense the expression.

To find the best value of K, gradient descent can be performed for several iterations over the cost function with each $K = 1, 2, ...,$. When a solution has a small value for both $||X - UV||_F^2$, and $\sum_k ||(A(A^T A)^{-1} A^T - I) b_k||_2^2$, identify it as the solution. This method will also be useful for heuristic methods such as particle swarm optimization, where the more determined cost function can guide the optimization. An advantage of this technique (using either gradient descent or heuristic approaches) is that the relative scaling of U and V should be discovered, as the trilateration penalty is only small with the correct scaling. This approach also applies to 3D spatial problems.

$$\frac{\partial}{\partial U_{mk}}\left(\lambda \cdot \sum_k \|(A(A^TA)^{-1}A^T - I)b_k\|_2^2\right) =$$

$$\frac{\partial}{\partial U_{mk}}\left(\lambda \cdot \sum_k \|A^* b_k\|_2^2\right) =$$

$$\lambda \frac{\partial}{\partial U_{mk}}\left(\|A^* b_k\|_2^2\right) =$$

$$\lambda \frac{\partial}{\partial U_{mk}} \sum_{\mu=1}^{M-1} (A^* b_k)_\mu^2 =$$

$$\lambda \frac{\partial}{\partial U_{mk}} \sum_{\mu=1}^{M-1} \Big(\sum_{\eta=1}^{M-1} A_{\mu\eta}^* b_{k_\eta}\Big)^2 =$$

$$2\lambda \sum_{\mu=1}^{M-1} \Big[\sum_{\eta=1}^{M-1} A_{\mu\eta}^* b_{k_\eta} \frac{\partial}{\partial U_{mk}} \sum_{\eta=1}^{M-1} A_{\mu\eta}^* b_{k_\eta} \Big] =$$

$$2\lambda \sum_{\mu=1}^{M-1} \Big[\sum_{\eta=1}^{M-1} A_{\mu\eta}^* b_{k_\eta} \frac{\partial}{\partial U_{mk}} \sum_{\eta=1}^{M-1} A_{\mu\eta}^* (d_{\eta k}^2 - d_{\eta *}^2 - d_{1k}^2) \Big]$$

$$= \begin{cases} 2\lambda \sum_{\mu=1}^{M-1} \Big[\sum_{\eta=1}^{M-1} A_{\mu\eta}^* b_{k_\eta} \sum_{\eta=1}^{M-1} A_{\mu\eta}^* (d_{\eta k}^2 - d_{\eta *}^2 - \frac{\partial}{\partial U_{1k}} d_{1k}^2) \Big], & m = 1 \\ 2\lambda \sum_{\mu=1}^{M-1} \Big[\sum_{\eta=1}^{M-1} A_{\mu\eta}^* b_{k_\eta} A_{\mu m}^* \frac{\partial}{\partial U_{mk}} d_{mk}^2 \Big], & m \neq 1 \end{cases}$$

$$= \begin{cases} 2\lambda \sum_{\mu=1}^{M-1} \Big[\sum_{\eta=1}^{M-1} A_{\mu\eta}^* b_{k_\eta} \sum_{\eta=1}^{M-1} A_{\mu\eta}^* (d_{\eta k}^2 - d_{\eta *}^2 - \frac{\partial}{\partial U_{1k}} \frac{1}{U_{1k}}) \Big], & m = 1 \\ 2\lambda \sum_{\mu=1}^{M-1} \Big[\sum_{\eta=1}^{M-1} A_{\mu\eta}^* b_{k_\eta} A_{\mu m}^* \frac{\partial}{\partial U_{mk}} \frac{1}{U_{mk}^2} \Big], & m \neq 1 \end{cases}$$

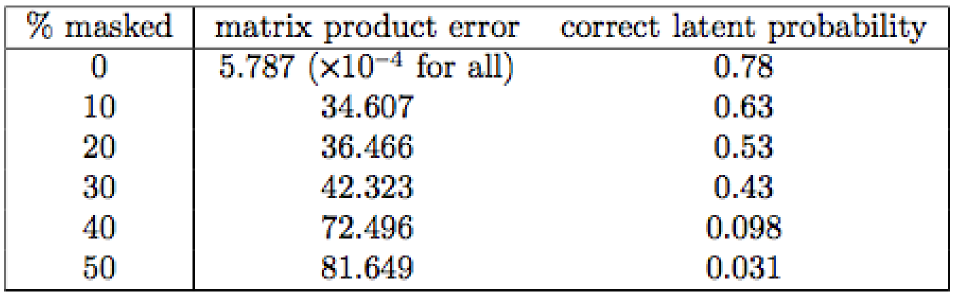
$$= \begin{cases} 2\lambda \sum_{\mu=1}^{M-1} \Big[\sum_{\eta=1}^{M-1} A_{\mu\eta}^* b_{k_\eta} \sum_{\eta=1}^{M-1} A_{\mu\eta}^* (d_{\eta k}^2 - d_{\eta *}^2 + \frac{1}{U_{1k}^2}) \Big], & m = 1 \\ -2\lambda \sum_{\mu=1}^{M-1} \Big[\sum_{\eta=1}^{M-1} A_{\mu\eta}^* b_{k_\eta} A_{\mu m}^* \frac{1}{U_{mk}^2} \Big], & m \neq 1 \end{cases} \tag{45}$$

4.7 Blind Signal Separation with Missing Data Results

Blind signal separation and matrix completion are both based on a low rank conditioned problem. If a given blind signal separation problem is missing data, the matrix completion technique will be appropriate for estimating the missing data as a preprocessing before blind signal separation. To test these algorithms' compatibility, the simulation described in section 3.8

was additionally masked to leave missing data for the blind signal separation problem. The mask applied is like that applied to the 770 MHz television data. Again it is in blocks to render a more challenging problem. The amount of data masked was varied with 0% meaning no masking or all data present, and 100% meaning no data available. Soft-impute is used to complete the matrix. It is then decomposed by tuning pruning. Table 8 shows the average error of 10 trials for each amount of masking. The trials shown are for the correct latent dimension found by the tuning-pruning algorithm. As the level of data missing increased, the algorithm is less likely to correctly identify the number of transmitters. This is given as a probability of correct number of transmitters (latent dimension) described in the last column of the table. The error reflects the difference in the product of factors given by the simulation, and the decomposition matrices product. A trend was not seen in the decomposition errors directly. When the matrix finds the correct latent dimension with more attempts and more missing data, the decomposition is consistently accurate. The 78% accuracy in no noise is likely due to some transmitters being too close to receivers or each other.

Table 8: Matrix Factorization Performance with Missing Data



% masked	matrix product error	correct latent probability
0	5.787 ($\times 10^{-4}$ for all)	0.78
10	34.607	0.63
20	36.466	0.53
30	42.323	0.43
40	72.496	0.098
50	81.649	0.031

Matrix Error of Product Matrix and Percentage of Correct Factorizations

CHAPTER 5
CONCLUSION

5.1 Summary of Work

In this work, machine learning tools are used to address CR problems. A missing data problem and a blind signal separation problem are considered for radio spectrum. CSRN performs well for spatial spectrum prediction of radio signals with missing data. Soft-impute for matrix completion used together with an RNN performs well for missing data problems in the radio spectrum time-frequency domain. An NMF method called tuning pruning is used for blind signal separation of radio signals in simulation. A geometric constraint is proposed to solve NMF problems under more general conditions.

Missing data problems and blind signal separation problems occur in a variety of contexts. It is a new idea to use CSRN in the missing data paradigm. Because of the adaptability of this network to different problem domains, this work may find many other applications where a symmetric topology exists that a network can be mapped to. Given the proposed NMF optimization technique using a geometric constraint, it is possible to solve blind signal separation problems in a new way. The solution constraint may also allow for the correct latent dimension to be identified in a situation when this is not known a priori.

CSRN is shown to be useful for spatial spectrum prediction with missing data. It is shown here to operate on a problem with 3 degrees of freedom, where the transmitter location moves and the locations to predict are over a 2D space. Matrix completion is shown to be useful as another missing data tool in radio. Furthermore, it can be used to preprocess data that will be used for RNNs or NMF. Blind signal separation is discussed in the CR literature with constraints that allow the factorization to be correct. Here, a different sparsity constraint is used, and it allows the blind signal separation to find the latent dimension and to give correct factorization. Also proposed is the leave one out technique for malicious node identification (and localization). Another constraint that naturally follows from the radio spatial problem setup is geometric. This may also prove to be a method for latent dimension identification but under a more natural or general problem constraint.

5.2 Future Work

The CSRN is able to generalize over different unseen transmitter locations. This means that when the CSRN is trained, it doesn’t know any data for the transmitter at certain locations. The CSRN is able to learn the pattern from given transmitter locations and generalize to others. For each given transmitter location in training, the sampling pattern is varied. An experiment that should be performed is to vary the number of sampling patterns that are used to build the training set. For this work, 100 sampling patterns are generated per training grid measured with a specific transmitter location. This could easily be increased to 1000 or more, and given the increased types of training patterns, with adequate training time the CSRN should give better performance on the test data. As more training data is available, the CSRN should be able to detect better the general pattern. It may, however, take longer to reach optimal training error as there are more patterns for correctly finding an input-output relationship. However, if a validation dataset is given, the training may be stopped at the minimum validation error. If more training patterns are given, the number of samples needed per grid may be reduced. Another degree of freedom can also be added to the spatial prediction problem by moving the transmitter location anywhere on the grid instead of only varying this in one dimension of the grid while keeping the other dimension fixed.

One of the training algorithms used for the CSRN is backpropagation. A generalizable algorithm for organizing the derivative calculation is proposed. A possible downside to this organization strategy could be an increase in complexity, especially in terms of computational time. It would be interesting to perform an analysis of the time complexity of this algorithm versus a brute force approach. This analysis may also show ways to improve the algorithm’s speed. The algorithm does appear to run quickly as inspected during initial use here.

In this work, the RNN is run using individual frequency channels separately. Multichannel RNN is another possibility that could be explored. It was shown that correlations exist between different frequency channels. In [40], a CR application of multichannel RNNs is explored.

The tuning pruning NMF technique, as currently implemented, is too slow for practice in real time applications. It will be a challenge to implement this faster. Perhaps GPU acceleration could increase the speed of execution. Two proposed algorithms for blind signal separation need to be tested. First is the leave one out malicious user identification. Here, a malicious user will need to be simulated. Second is the algorithm described at the end of section 4.4 based on partitioning the frequency-time domain. This will require running the simulation or collecting real data over a large frequency-time space. The matrix completion can also be extended to tensor completion for space/frequency/time domain analysis. There is already some work in this field that may be investigated (see e.g. [41]).

References

[1] A. Khattab, D. Perkins, and M. Bayoumi, *Cognitive radio networks: from theory to practice*. Springer Science & Business Media, 2012.

[2] I. Vilovic, N. Burum, and Z. Sipus, "Design of an indoor wireless network with neural prediction model," in *Antennas and Propagation, 2007. EuCAP 2007. The Second European Conference on*, 2007, pp. 1-5: IET.

[3] S. Haykin, "Cognitive radio: brain-empowered wireless communications," *IEEE journal on selected areas in communications,* vol. 23, no. 2, pp. 201-220, 2005.

[4] A. G. Fragkiadakis, E. Z. Tragos, and I. G. Askoxylakis, "A survey on security threats and detection techniques in cognitive radio networks," *IEEE communications surveys and tutorials,* vol. 15, no. 1, pp. 428-445, 2013.

[5] J. Meng, W. Yin, H. Li, E. Houssain, and Z. Han, "Collaborative spectrum sensing from sparse observations using matrix completion for cognitive radio networks," in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on,* 2010, pp. 3114-3117: IEEE.

[6] G. Ding, J. Wang, Q. Wu, L. Yu, Y. Jiao, and X. Gao, "Joint spectral-temporal spectrum prediction from incomplete historical observations," in *Signal and Information Processing (GlobalSIP), 2014 IEEE Global Conference on*, 2014, pp. 1325-1329: IEEE.

[7] K. Tsagkaris, A. Katidiotis, and P. Demestichas, "Neural network-based learning schemes for cognitive radio systems," *Computer Communications,* vol. 31, no. 14, pp. 3394-3404, 2008.

[8] S. M. Mishra, A. Sahai, and R. W. Brodersen, "Cooperative sensing among cognitive radios," in *Communications, 2006. ICC'06. IEEE International conference on*, 2006, vol. 4, pp. 1658-1663: IEEE.

[9] M. J. Saber and S. M. S. Sadough, "On optimal spectrum sensing strategy for cognitive radio systems under primary user emulation attack," in *Telecommunications (IST), 2014 7th International Symposium on*, 2014, pp. 1113-1116: IEEE.

[10] A. Attar, H. Tang, A. V. Vasilakos, F. R. Yu, and V. C. Leung, "A survey of security challenges in cognitive radio networks: Solutions and future research directions," *Proceedings of the IEEE,* vol. 100, no. 12, pp. 3172-3186, 2012.

[11] W. Murphy and W. Hereman, "Determination of a position in three dimensions using trilateration and approximate distances," *Department of Mathematical and Computer Sciences, Colorado School of Mines, Golden, Colorado, MCS-95,* vol. 7, p. 19, 1995.

[12] J. Mitola, "The software radio architecture," *IEEE Communications magazine,* vol. 33, no. 5, pp. 26-38, 1995.

[13] R. G. Machado and A. M. Wyglinski, "Software-defined radio: Bridging the analog–digital divide," *Proceedings of the IEEE,* vol. 103, no. 3, pp. 409-423, 2015.

[14] H. Arslan, *Cognitive radio, software defined radio, and adaptive wireless systems*. Springer, 2007.

[15] (2015). *FCC Auctions - Auction 97 Summary*.

[16] T. Yucek and H. Arslan, "A survey of spectrum sensing algorithms for cognitive radio applications," *IEEE communications surveys & tutorials,* vol. 11, no. 1, pp. 116-130, 2009.

[17] M. Wellens and P. Mähönen, "Lessons learned from an extensive spectrum occupancy measurement campaign and a stochastic duty cycle model," *Mobile networks and applications,* vol. 15, no. 3, pp. 461-474, 2010.

[18] A. Neskovic, N. Neskovic, and G. Paunovic, "Modern approaches in modeling of mobile radio systems propagation environment," *IEEE Communications Surveys & Tutorials,* vol. 3, no. 3, pp. 2-12, 2000.

[19] F. D. Alotaibi, A. Abdennour, and A. A. Ali, "A robust prediction model using ANFIS based on recent TETRA outdoor RF measurements conducted in Riyadh city–Saudi Arabia," *AEU-International Journal of Electronics and Communications,* vol. 62, no. 9, pp. 674-682, 2008.

[20] M. J. Orr, "Introduction to radial basis function networks," ed: Technical Report, Center for Cognitive Science, University of Edinburgh, 1996.

[21] X. Pang and P. Werbos, "Neural network design for J function approximation in dynamic programming," *arXiv preprint adap-org/9806001,* 1998.

[22] L. Vidyaratne, M. Alam, J. Anderson, and K. Iftekharuddin, "Improved training of cellular SRN using Unscented Kalman Filtering for ADP," in *Neural Networks (IJCNN), 2014 International Joint Conference on*, 2014, pp. 993-1000: IEEE.

[23] L. Vidyaratne, A. Glandon, M. Alam, and K. Iftekharuddin, "Deep recurrent neural network for seizure detection," in *Neural Networks (IJCNN), 2016 International Joint Conference on*, 2016, pp. 1202-1207: IEEE.

[24] R. Mazumder, T. Hastie, and R. Tibshirani, "Spectral regularization algorithms for learning large incomplete matrices," *Journal of machine learning research,* vol. 11, no. Aug, pp. 2287-2322, 2010.

[25] J. L. Elman, "Finding structure in time," *Cognitive science,* vol. 14, no. 2, pp. 179-211, 1990.

[26] Z. Hu *et al.*, "Robust non-negative matrix factorization for joint spectrum sensing and primary user localization in cognitive radio networks," in *Waveform Diversity & Design Conference (WDD), 2012 International*, 2012, pp. 303-307: IEEE.

[27] A. Zaeemzadeh, M. Joneidi, B. Shahrasbi, and N. Rahnavard, "Missing spectrum-data recovery in cognitive radio networks using piecewise constant nonnegative matrix factorization," in *Military Communications Conference, MILCOM 2015-2015 IEEE*, 2015, pp. 238-243: IEEE.

[28] X. Fu, N. D. Sidiropoulos, and W.-K. Ma, "Power spectra separation via structured matrix factorization," *IEEE J. Sel. Topics Signal Process,* 2015.

[29] M. Mørup and L. K. Hansen, "Tuning pruning in sparse non-negative matrix factorization," in *Signal Processing Conference, 2009 17th European*, 2009, pp. 1923-1927: IEEE.

[30] C.-J. Lin, "Projected gradient methods for nonnegative matrix factorization," *Neural computation,* vol. 19, no. 10, pp. 2756-2779, 2007.

[31] S. S. Haykin, *Kalman filtering and neural networks*. Wiley Online Library, 2001.

[32] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE,* vol. 78, no. 10, pp. 1550-1560, 1990.

[33] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge university press, 2012.

[34] H. T. Friis, "A note on a simple transmission formula," *Proceedings of the IRE,* vol. 34, no. 5, pp. 254-256, 1946.

[35] C. E. Shannon, "Communication in the presence of noise," *Proceedings of the IRE,* vol. 37, no. 1, pp. 10-21, 1949.

[36] J. B. Johnson, "Thermal agitation of electricity in conductors," *Physical review,* vol. 32, no. 1, p. 97, 1928.

[37] H. Nyquist, "Thermal agitation of electric charge in conductors," *Physical review,* vol. 32, no. 1, p. 110, 1928.

[38] A. V. Oppenheim and R. W. Schafer, "Digital signal processing. 1975," *Englewood Cliffs, New York.*

[39] H. O. Lancaster and E. Seneta, *Chi-Square Distribution*. Wiley Online Library, 1969.

[40] M. I. Taj and M. Akil, "Cognitive Radio Spectrum E volution Prediction using A rtificial Neural Networks based Multivariate T ime Series Modelling," in *Wireless Conference 2011-Sustainable Wireless Technologies (European Wireless), 11th European*, 2011, pp. 1-6: VDE.

[41] X. Fu, N. D. Sidiropoulos, J. H. Tranter, and W.-K. Ma, "A factor analysis framework for power spectra separation and multiple emitter localization," *IEEE Transactions on Signal Processing*, vol. 63, no. 24, pp. 6581-6594, 2015.

VITA

Alexander M. Glandon

Department of Electrical and Computer Engineering
Old Dominion University Frank Batten College of Engineering
231 Kaufman Hall
Norfolk, Virginia 23529

Alexander M. Glandon works in the Old Dominion University Computational Intelligence and Machine Vision Laboratory. His research interests include radio systems, machine learning, signal processing, and computer vision. Projects he is involved in include machine learning for cognitive radio under an NSF grant, seizure detection with EEG, and human action recognition with LIDAR. He plans to continue studying as a doctoral student at Old Dominion University in the Electrical and Computer Engineering Department.