Data-Driven Covariance Estimation

John T. Rogers II
Department of Electrical and Computer Engineering
Mississippi State University
Starkville, MS 39762
Email: jtr242@msstate.edu

John E. Ball
Department of Electrical and Computer Engineering
Mississippi State University
Starkville, MS 39762
Email: jeball@ece.msstate.edu

Ali C. Gurbuz
Department of Electrical and Computer Engineering
Mississippi State University
Starkville, MS 39762
Email: gurbuz@ece.msstate.edu

2022 IEEE International Symposium on Phased Array Systems & Technology (PAST) | 978-1-6654-4166-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/PAST49659.2022.9975068

***Abstract*—Array processing algorithms commonly use covariance matrices as inputs. Improved performance compared to the sample covariance matrix can be achieved using knowledge of the array configuration. We present an analysis of Convolutional Neural Network (CNN) architectures for covariance matrix estimation for a simulated Uniform Linear Array (ULA). Additionally, a novel neural network configuration is presented which demonstrates reduced error and more resilience to lower Signal-to-Noise Ratio (SNR) levels when compared to the sample covariance matrix.**

***Index Terms*—Covariance Matrix, Data-Driven, Convolutional Neural Network**

I. Introduction

Covariance matrices have many desirable properties which has led to their use as inputs for many signal processing algorithms. A covariance matrix consists of the pairwise covariances between two sets of random variables. An auto-covariance matrix is the covariance matrix of a set of random variables with itself and it is highly used in many communications, radar, and array signal processing applications. Herein, all covariance matrices used are auto-correlation matrices and the terms will be used interchangeably. The auto-covariance matrix is defined in (1), where $\boldsymbol{\mu}_x$ is a vector of the means for each random variable in the set of random variables $\mathbf{X}$. When the random variables are zero mean, this equation simplifies to equation for the auto-correlation matrix.

$$\mathbf{K} = E[\mathbf{X} - \boldsymbol{\mu}_x]E[\mathbf{X} - \boldsymbol{\mu}_x]^H = E[\mathbf{X}\mathbf{X}^H] - \boldsymbol{\mu}_x\boldsymbol{\mu}_x \quad (1)$$

As the true covariance matrix is often not known in practice, the sample covariance matrix is generally used as an approximation.

The sample auto-covariance matrix is commonly used in Direction of Arrival (DOA) estimation and related array processing algorithms. Beamforming techniques such as the Minimum Variance Distortionless Response (MVDR) [1] apply steering vectors to the covariance matrix of a signal to project the signal power across an angular spectrum. Other algorithms exploit the desirable properties of the eigen-decomposition of the covariance matrix as the signals and noise are separated into orthogonal subspaces. The MUltiple SIgnal Classification (MUSIC) algorithm [2] applies beamforming to eigen-vectors corresponding to the noise sub-space. Number of Sources estimation, which is required for algorithms like MUSIC, commonly investigate the eigen-values of the covariance matrix. For a signal with multiple signal sources, the signal power of each source is constrained to a single eigen-value whereas the remaining eigen-values are equal to the noise power. As the sample covariance matrix is an approximation, tests such as the Akaike Information Criterion [3] and Minimum Description Length [4] are applied to the eigen-values to obtain an estimate of the number of sources.

Improved estimation of the auto-covariance matrix is an open area in signal processing with many recent developments. Knowledge of the application scenario, such as the array configuration, can be used to achieve improved covariance matrix estimation. In [5], interference steering vectors are used to reconstruct the interference plus noise covariance matrix. Upadhya and Vorobyov [6] presented an algorithm for covariance matrix estimation for Multiple-Input Multiple-Output (MIMO) systems. Deep neural networks are used in [7] for covariance matrix estimation in a computer vision application. In [8], covariance matrix estimation is modeled as an optimization problem using geometric considerations.

In this paper, we explore the use of Convolutional Neural Networks (CNNs) for covariance estimation. We present a comparison of four simple CNN architectures representing common network topologies. Additionally, we present a novel neural network configuration to estimate the covariance matrix from the array snapshot observations. The proposed learning-based covariance matrix estimation approach has the following properties:

- Significantly improved estimation performance compared to the sample covariance matrix.
- A constrained network size allowing for efficient computation.

A. Nomenclature

A summary of variables used herein and their definitions is shown in table I.

II. Methodology

Herein, we define the signal model and present multiple Artificial Neural Network (ANN) topologies and configurations that are trained and tested to estimate the covariance matrices of signals from a simulated Uniform Linear Array (ULA).

TABLE I
VARIABLE DEFINITIONS

Name	Definitions
$\mathbf{X}$	Simulated RADAR Measurement containing P snapshots
x	Single element of $\mathbf{X}$
$\mathbf{s}$	Noiseless Simulated RADAR Measurement
$\mathbf{K}$	True Covariance Matrix
$\mathbf{K}_s$	Sample Covariance Matrix
$\hat{\mathbf{K}}$	Estimated Covariance Matrix
$\boldsymbol{\lambda}$	True Covariance Matrix Sorted Eigenvalue Vector
$\hat{\boldsymbol{\lambda}}$	Estimated Covariance Matrix Sorted Eigenvalue Vector
M, m	Number of Receiver Array Elements and Indexing Variable
P, p	Number of Snapshots and Indexing Variable
T, k	Number of Signal Sources and Indexing Variable
a	Amplitude of Signal Source
f	RADAR Operating Frequency
c	Speed of Light in a Vacuum, $3 \cdot 10^8$
Δx	Receiver Array Element Spacing
ϕ	Angle of Signal Source
n	Complex i.i.d. White Gaussian Noise Sample
λ, α	SeLU Constants

A. Signal Model

The synthetic data set used simulates the measurements of a ULA using the following equation:

$$x_m = \left[\sum_{k=1}^{T} a_k \exp\left(-j2\pi \frac{f}{c} \Delta x\, m \sin\left(\phi_k\right)\right)\right] + n_m \quad (2)$$

where x_m is the power measured by the m-th receive array element, a_k is the amplitude of the k-th signal source, T is the true number of sources, $j = \sqrt{-1}$, f is the operating frequency, c is the speed of the wave, Δx is the spacing between array elements, m is the array element index, ϕ_k is the angle of the $k-th$ signal source, and n_m is independent and identically distributed (i.i.d) complex white Gaussian noise observed by the $m-th$ array element. A single array measurement is simulated by arranging x_m measurements for all M receive elements as a vector with randomly generated complex noise. The final signal is constructed by arranging P snapshots of the array measurement into a $[M \times P]$ matrix $\mathbf{X}$.

These simulated signal matrices are further processed to obtain the sample and true covariance matrices. As the signal is sinusoidal and the noise is white, the simulated signal matrices have a true mean of 0, thus the covariance matrix is equal to the correlation matrix. Therefore, we can calculate the sample covariance matrix, $\mathbf{K}_s$ using the approximated correlation matrix defined as follows:

$$\mathbf{K}_s = \frac{1}{P}\mathbf{X}\mathbf{X}^H \quad (3)$$

Likewise the true covariance matrix is equal to the true correlation matrix given by:

$$\mathbf{K} = E[\mathbf{X}\mathbf{X}^H] \quad (4)$$

As the noise is independent for each receiver array element, the true covariance between different elements is unaffected by noise. As the covariance matrix diagonal contains the variance for each element, these values are effected by noise . The diagonal terms are equal to the diagonal terms of the corresponding noiseless covariance matrix plus the noise variance. Therefore, the true covariance matrix can be found using (5) where $\mathbf{s}$ is a vector containing a noiseless measurement of the signal constructed using only the bracketed portion of (2), σ_n^2 is the noise power, and $\mathbf{I}$ is the identity matrix.

$$\mathbf{K}_s = \mathbf{s}\mathbf{s}^H + \sigma_n^2\mathbf{I} \quad (5)$$

B. Simulation Data

The data used herein simulates an 11 element ULA RADAR operating at 5GHz and 32 snapshots are simulated for each set of signal sources. The receive array element spacing is set to one-half of the wavelength. Signal sources are randomly distributed within a 180° Field of View (FOV) with a minimum spacing of 20°. The simulated signals contain 0 to 4 signal sources with per source SNR values ranging from $-10dB$ to $20dB$ measured at the receiver. The training dataset contains $10,000$ signals for each number of sources providing a total of $50,000$ signals. Likewise, the testing dataset contains $1,000$ signals for each number of sources. Additional testing datasets are used for assessing performance relative to SNR and the number of snapshots. The dataset for the former contains 100 signals for each number of sources and each integer SNR value between $-10dB$ and $20dB$ resulting in a total of $15,500$ signals. The range of snapshots tested includes the first 8 powers of 2, ranging from 1 to 128. 1000 signals are generated for each value of the number of sources and snapshots resulting in $40,000$ signals.

Within these datasets the signals are processed into input-label pairs. The inputs are the sample covariance matrices of the signals computed according to equation 3. Likewise, the labels are the true covariance matrix computed according to equation 5. To mitigate the effect of the SNR range the network inputs and labels are normalized. As the norm of the true covariance matrix would not be available in any practical application, the Frobenius norm of the sample covariance matrix is used for all normalization. As shown in Figure 1, the input sample covariance matrix is normalized using its Frobenius norm for each input. The same factor is used to normalize the true covariance matrix, which serves as the data label. Finally this factor is multiplied by the network output to return it to the appropriate scale. The normalized form of the input and labels are pre-computed before training the networks.

As support for complex-value neural networks is not yet widespread, the real and imaginary components of these matrices are separated to form two $[M \times P \times 2]$ multidimensional arrays.

C. Utilized Neural Network Structures

The evaluated network architectures are all Convolutional Neural Networks (CNNs) utilizing of two dimensional convolution layers. These layers can vary in number of convolutional

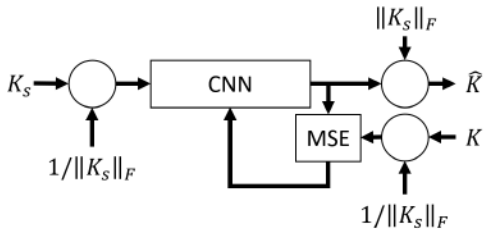

Fig. 1. Normalization Flowchart

masks, mask size, and mask stride. Additionally, a convolutional layer can employ zero-padding. Convolutional transpose layers also utilize a set of masks with the same parameters to expand the input size by multiplying all values of the mask by a single sample from the input.

Fully Connected layers are also utilized. These layers consists of an arrangement of multiple linear perceptrons. These layers can vary in the number of perceptrons used.

The Scaled Exponential Linear Unit (SELU) [9] is the sole activation function used in these networks. The SELU is defined in (6) where λ and α are predefined constants set to 1.05070098 and 1.67326324 respectively. This activation function is self-normalizing eliminating the need for batch normalization layers. Additionally, the SELU, unlike the more commonly used Rectified Linear Unit (ReLU), can return negative values which are necessitated by the Hermitian nature of the estimated matrices. A SELU function follows all convolutional layers and fully connect layers in the evaluated networks except for the output layers.

$$SELU(x) = \lambda \begin{cases} x & x \geq 0 \\ \alpha e^x - \alpha & x < 0 \end{cases}, \tag{6}$$

All networks are trained using batch processing. The Mean Square Error (MSE) with a learning rate of 0.001 is used as the loss function. All networks are trained for 100 epochs with a batch size of 100.

D. Error Metrics

Each network topology is assessed using the Normalized Mean Square Error (NMSE). Because the scale of the covariance matrix is influenced by the signal power, noise power, and number of signal sources, normalized error metrics are required. The NMSE utilizes the Frobenius Norm of the true covariance matrix to normalize the error as shown in (7) where K is the true covariance matrix, $\hat{K}$ is an estimated covariance matrix, and $||\cdot||_F$ is the Frobenius norm.

$$NMSE[\hat{\mathbf{K}}] = \frac{||\hat{\mathbf{K}} - \mathbf{K}||_F}{||\mathbf{K}||_F} \tag{7}$$

III. Results

All network's performance were assessed by taking the average NMSE for 5 training instances. Outlying error values for individual networks were observed, but only for larger-than-average errors which indicates inconsistent training convergence. These outliers were not observed for results presented in this section.

A. Evaluated Architectures

Four major networks architectures were evaluated. The first network utilized two convolutional layers followed by two matching convolutional transpose layers to produce a symmetric structure which constricts the size of the intermediate data structures. The second network uses four convolutional layers with zero-padding to maintain the same size for first two dimensions for each intermediate data structure. The third network expands the size of the intermediate data structures by utilize two convolutional transpose layers, followed by two symmetric convolutional layers. The final network uses a traditional CNN structure with two convolutional layers and terminates with two fully connected layers.

Table II shows the layers and relevant parameters for each network. Layers are indicated using the following notation: Convolutional layers are denoted as Conv(Number of Masks, Mask Size), Convolutional Transpose as ConvT(Number of Masks, Mask Size), Fully Connected as FC(Number of Perceptrons), Batch Normalization as BN, and Scaled Linear Unit as SeLU. The layer parameters presented in table II were selected to allow a fair comparison between the 4 proposed architectures.

TABLE II
Network Architectures

Network 1	Network 2	Network 3	Network 4
Conv(32,5)	Conv(32,5)	ConvT(32,3)	Conv(32,5)
SeLU	SeLU	SeLU	SeLU
Conv(32,3)	Conv(32,5)	ConvT(32,5)	Conv(32,3)
SeLU	SeLU	SeLU	SeLU
ConvT(32,3)	Conv(32,3)	Conv(32,5)	FC(242)
SeLU	SeLU	SeLU	SeLU
ConvT(2,5)	Conv(2,3)	Conv(2,3)	FC(242)
Number of Learned Parameters			
21922	37282	52642	264140

The results for the four tested architectures are given in table III as well as the performance using the sample covariance matrix. All four networks demonstrated improved performance compared to the sample covariance matrix with network 4 demonstrating the best overall performance.

TABLE III
Network Architectures Results

	Average NMSE	
	Training	Testing
$\mathbf{K_s}$	0.1048	0.1041
Network 1	0.0238	0.0237
Network 2	0.0119	0.0119
Network 3	0.0140	0.0140
Network 4	0.0060	0.0062

As the fourth network contained significantly more learned parameters than the other three, a second experiment was

performed after balancing the number of learned parameters. The modified parameters include the number of neurons in perceptron layers and the number of masks in the convolutional and convolutional transpose layers. The balanced networks are shown in table IV.

TABLE IV
BALANCED NETWORK ARCHITECTURES

Network 1	Network 2	Network 3	Network 4
Conv(64,5)	Conv(48,5)	ConvT(40,3)	Conv(64,5)
SeLU	SeLU	SeLU	SeLU
Conv(64,3)	Conv(48,5)	ConvT(40,5)	Conv(64,3)
SeLU	SeLU	SeLU	SeLU
ConvT(64,3)	Conv(48,3)	Conv(40,5)	FC(22)
SeLU	SeLU	SeLU	SeLU
ConvT(2,5)	Conv(2,3)	Conv(2,3)	FC(242)
Number of Learned Parameters			
80706	82034	81802	81280

The results for the adjusted networks are shown in table V. Network 4 still exhibits the best performance and was selected for further improvement.

TABLE V
BALANCED NETWORK ARCHITECTURES RESULTS

	Average NMSE	
	Training	Testing
K_s	0.1048	0.1041
Network 1	0.0167	0.0167
Network 2	0.0108	0.0108
Network 3	0.0120	0.0120
Network 4	0.0071	0.0073

B. Final Network Results

The final network architecture is shown in Table VI. After selecting from the four proposed networks, each parameter was modified and tested one at a time to improve the networks performance. The network utilizes zero-padding as this allows for an increased variety in convolutional mask arrangements as well as a decrease in error. The remaining parameters were modified in the following order: convolutional mask size, number of convolutional layers, number of convolutional masks, number of perceptrons in the third layer, number of fully connected layers, batch size, and learning rate.

TABLE VI
FINALIZED NETWORK ARCHITECTURE

Conv(2,3)
SeLU
Conv(2,3)
SeLU
FC(88)
SeLU
FC(242)

The final results this network are shown in table VII. The final network's error is approximately $1/20th$ of the error when using the sample covariance matrix.

The performance of the network relative to SNR is shown in figure 2. The final network demonstrates robust performance

TABLE VII
FINAL RESULTS

	Average NMSE	
	Training	Testing
K_s	0.1048	0.1041
Mean	0.0055	0.0056
Minimum	0.0053	0.0053
Maximum	0.0058	0.0059

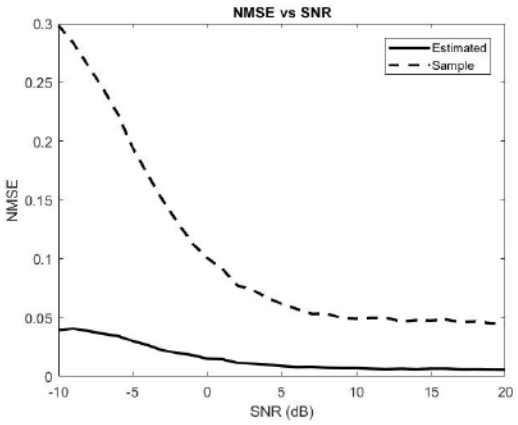

Fig. 2. Performance vs SNR for Network Results and Sample Covariance Matrix

relative to SNR as the performance for all tested SNR levels is superior to the sample covariance matrix at high SNR levels.

The performance relative to the number of snapshots used when constructing the input sample covariance matrices is shown in figure 3 with the exact values given in table VIII. For all number of snapshots tested, the proposed method demonstrates lower error than the sample covariance matrix. Additionally, the error decreases monotonically as the number of snapshots increases demonstrating the networks ability to generalize despite being trained on a dataset with inputs containing 32 snapshots exclusively.

To demonstrate the benefit of improved estimation of the

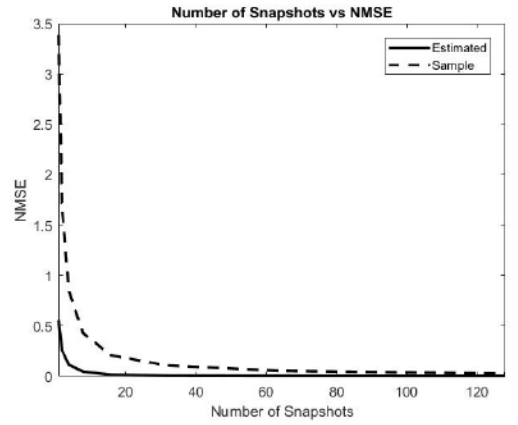

Fig. 3. Performance vs Number of Snapshots for Network Results and Sample Covariance Matrix

TABLE VIII
RESULTS WITH VARIED NUMBER OF SNAPSHOTS USING PROPOSED NETWORK ESTIMATES AND SAMPLE COVARIANCE MATRICES

Snapshots	Estimate	Sample
1	0.5547	3.3872
2	0.2553	1.6747
4	0.1104	0.8388
8	0.0451	0.4254
16	0.0151	0.2057
32	0.0063	0.1045
64	0.0043	0.0524
128	0.0036	0.0262

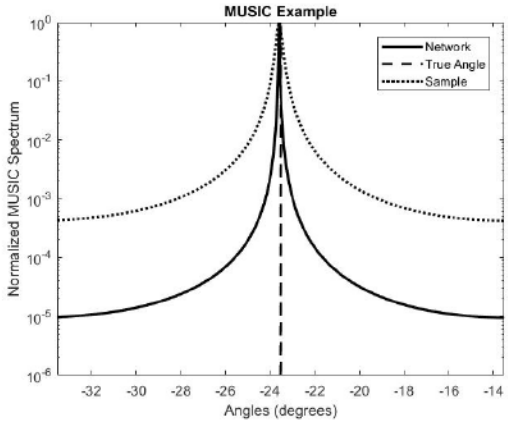

Fig. 4. Normalized MUSIC Spectrum using the Network Results and Sample Covariance Matrix

signal covariance matrix, the MUSIC spectrum of a single target case is shown in figure 4. This case was randomly selected from the testing dataset. The target is located at $-23.5°$ with an SNR of $15dB$.

IV. CONCLUSION

Herein, we show a comparison of simple neural networks representative of common CNN configurations. The traditional architecture of using convolutional layers followed by fully connected perceptron layers demonstrated the universal best performance. This network was further optimized and demonstrated improved performance compared to the sample covariance matrix optimization. The proposed network exhibited reduced error and significantly improved resilience to lower SNR levels.

Future work in this area includes: using the proposed network as the foundation for deep learning networks created using hierarchical network design and the investigation of custom optimization functions.

ACKNOWLEDGEMENTS

This work was sponsored by National Science Foundation under Grant No. 2047771.

REFERENCES

[1] J. Capon, "High-resolution frequency-wavenumber spectrum analysis," *Proceedings of the IEEE*, vol. 57, no. 8, pp. 1408–1418, 1969.

[2] R. Schmidt, "Multiple emitter location and signal parameter estimation," *IEEE transactions on antennas and propagation*, vol. 34, no. 3, pp. 276–280, 1986.

[3] H. Akaike, "A new look at the statistical model identification," *IEEE transactions on automatic control*, vol. 19, no. 6, pp. 716–723, 1974.

[4] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, no. 5, pp. 465–471, 1978.

[5] Z. Zheng, Y. Zheng, W.-Q. Wang, and H. Zhang, "Covariance matrix reconstruction with interference steering vector and power estimation for robust adaptive beamforming," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 9, pp. 8495–8503, 2018.

[6] K. Upadhya and S. A. Vorobyov, "Covariance matrix estimation for massive mimo," *IEEE Signal Processing Letters*, vol. 25, no. 4, pp. 546–550, 2018.

[7] K. Liu, K. Ok, W. Vega-Brown, and N. Roy, "Deep inference for covariance estimation: Learning gaussian noise models for state estimation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 1436–1443.

[8] A. Aubry, A. De Maio, and L. Pallotta, "A geometric approach to covariance matrix estimation and its applications to radar problems," *IEEE Transactions on Signal Processing*, vol. 66, no. 4, pp. 907–922, 2018.

[9] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 972–981.