2024 IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE) | 979-8-3503-5111-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/WiSEE61249.2024.10850415

A Low-Complexity LSTM Network to Realize Multibeam Beamforming

Hansaka Aluvihare[1], Carina Shanahan[1], Sirani M. Perera[1], Sivakumar Sivasankar[2], Umesha Kumarasiri[2], Arjuna Madanayake[2], Xianqi Li[3]
[1]Department of Mathematics, Embry-Riddle Aeronautical University, Daytona Beach, FL, 32703 USA
Emails: {aluvihah, shanahc1}@my.erau.edu, pereras2@erau.edu
[2]Department of Electrical and Computer Engineering, Florida International University, Miami, FL, 33174 USA
Emails: {ssiva011, ukuma003, amadanay}@fiu.edu
[3]Department of Mathematics and Systems Engineering, Florida Institute of Technology, Melbourne, FL, 32901 USA
Email: xli@fit.edu

***Abstract*—Massive data structures can be embedded in the form of weight matrices, enabling to design of neural networks with low-complexity learning algorithms. These data can be organized in rows of matrices, containing progressive phase shifts for a specific beam, along with input and output vectors consisting of time-domain signals. In our previous work, we have identified that multi-beam beamformers based on true-time-delays (TTDs) can be mathematically formulated as the elements of delay Vandermonde matrices (DVM). Thus, by adopting a frequency domain variable, we can express TTDs of time delay data in terms of elements of the DVM. Learning from prior work, we propose to present a low-complexity neural network to realize multibeam beamforming leveraging a novel LSTM network. The goal of our work is to reduce the complexity of the multibeam beamforming algorithm from $\mathcal{O}(N^2L)$ to $\mathcal{O}(N^sL)$, where $1 < s < 2$, by imposing factorization of the DVM in an LSTM network having L layers.**

***Index Terms*—Sparse Matrices, delay Vandermonde matrices, Complexity and Performance of Algorithms, wireless communication, Multi-beam beamformers, Neural networks, Low-complexity Networks, and LSTM.**

I. Introduction

Beamforming is a signal processing technique that enables the directional focusing of transmitted or received signals, resulting in improved signal strength, interference mitigation, and enhanced spatial resolution. While narrow and wideband beamforming techniques are both available, wideband multi-beam beamforming has gained popularity in emerging mm-wave (mmW) communication systems, wireless networks, ultrasound imaging, and radar. Many beamforming techniques can be efficiently implemented using FFT-based algorithms, which can utilize either time-domain pre-FFT schemes or frequency-domain post-FFT schemes [1]–[3]. Unfortunately, narrowband multi-beam beamformers based on FFT suffer from the long-standing problem of beam squint. Fortunately, the TTDs-based wideband multi-beam beamformers use a Vandermonde structure of delay-based steering vectors to overcome the beam squint problem, resulting in a DVM beamforming matrix [1], [4]–[7].

This work was funded by the National Science Foundation with Award Numbers 2229473 and 2229471.

Beamforming has been integrated into the design of neural networks, such as convolutional neural networks (CNNs), to enable adaptive beamforming with limited signal snapshots for applications such as radar [8] and real-time medical ultrasound imaging [9]. Moreover, recurrent neural networks (RNNs), which are well-suited for time-series analysis, have been utilized to recognize speech in conjunction with beamforming [10], as well as for adaptive beamforming to minimize sidelobe interference [11]. It is evident that RNN can be utilized for analyzing time delays of elements in a linear array or other configurations across a wide frequency range.

Long Short-Term Memory (LSTM) networks are a subclass of RNNs with specific filtering functions for improving sequential modeling. In particular, LSTM networks are adept at capturing long-term dependencies in temporal data [12], [13]. Traditional RNNs often struggle to capture long-range dependencies due to the vanishing or exploding gradient problem [14], [15]. This limitation hampers the learning and retention of relevant information over extended sequences. However, LSTMs address this issue by introducing a memory block that contains "memory cells" and gating mechanisms. These mechanisms enable the network to selectively retain or discard information over multiple time steps [16]. The LSTM architecture consists of five main components: the cell state, the hidden state, the forget gate, the input gate, and the output gate, which are utilized at every time step [17], [18]. LSTM has been successfully implemented in beamforming applications [19]–[26] in the past, and has also been utilized to reduce the complexity of beamformers [27], [28].

In the LSTM, the forget gate plays a significant role in determining which information is to be kept or discarded from the previous time step. It accomplishes this by considering the previous hidden state, h_{t-1}, and the current input, x_t, and passing them through an activation function. The input gate controls the inflow of new information to be stored in the long-term memory of the network. First, it generates a candidate value, $\tilde{C}_t$, This candidate value is calculated by applying two distinct weight matrices and biases that act upon the previous hidden state, h_{t-1}, and the current input vector, x_t. The computation for the candidate value $\tilde{C}_t$ is as follows:

$$\tilde{C}_t = a_t(W_{c,h} \cdot [h_{t-1}] + W_{c,x_t} \cdot [x_t] + b_c) \quad (1)$$

where $W_{c,h}$ refers to the weight matrix applied to the hidden state, a_t is the activation function, and W_{c,x_t} refers to the weight matrix applied to input x_t.

The mathematical structure of the input gate is similar to that of the forget gate. The final stage of the LSTM is the output gate, which plays a crucial role in updating the short-term memory of the network. With these well-designed components, the LSTM model effectively manages both the incoming information and the memory storage, resulting in a powerful and dynamic short-term memory system.

A. Our Previous Work [29]

We proposed a structured neural network to realize multi-beam beamforming using structure-imposed weight matrices and submatrices. We have shown that these structured weight matrices greatly reduce the complexity of the network from $\mathcal{O}(M^2L)$ to $\mathcal{O}(Mp^2L)$, where L is the layers of network, M is the number of nodes in the input and output layers, p is the number of submatrices per layer, and $M >> L, p$. We also showed that the network architecture could be utilized to realize multi-beam beamforming without sacrificing accuracy.

B. The Proposed Work

To reduce arithmetic complexity from $\mathcal{O}(N^2L)$ to $\mathcal{O}(N^sL)$, we aim to reduce the number of weight parameters that need to be trained in an LSTM network. In order to accomplish this, we impose the factorization of the DVM in [1] as weight matrices inside the LSTM, representing TTD multi-beam beamformers based on the elements of the DVM. Imposing this structure onto the weight matrices does come at the expense of accuracy, to some extent, which we explore in detail in Section IV.

The remainder of this paper is structured as follows. In Section II, we discuss the methodology for using the LSTM network to realize multi-beam beamforming-based TTDs, including the details of forward propagation and backpropagation. In Section III, we present our experimental setup for the data collection. In Section IV, we give details on the process of training the proposed structured LSTM (S-LSTM) network, as well as the simulated beamformers. Finally, we will conclude the paper in Section V.

II. Method

Seen LSTM as a subclass of RNN, we introduce a structure-imposed LSTM network to realize multi-beam beamforming-based TTDs. Our approach introduces the S-LSTM by regularizing the weight matrices of neural networks utilizing the structure of the DVM, followed by sparse factorization. By utilizing the DVM-vector product, we can realize multi-beam beamforms while ensuring robust and trained structured neural networks to reduce the complexity in comparison to conventional LSTMs. This approach not only reduces the arithmetic complexity but also the space complexities based on the classical DVM factorization having many sparse matrices, especially for large problem sizes. The high-level design of the proposed approach is outlined in Fig 1. The S-LSTM cell processes input data from each antenna array element to predict the DVM-vector product in real time.

A. S-LSTM Neural Network for Beamforming

The weight matrices inside each LSTM cell can be represented as sparse matrices having the network architecture shown in Fig. 1. At every time step t, we input N incoming signals to produce N output signals where N represents the number of elements in the antenna array, while the hidden layers are passed through the dense weight matrices. We developed an LSTM network to manage a substantial volume of input data. We adopt the framework of the DVM, sequentially followed by the factorization presented in [1], to propose a structural LSTM network for the implementation of multibeam beamformers. Consequently, the weight matrices derive from the factorization of the DVM and are integrated within the S-LSTM framework. The S-LSTM network consists of 1 LSTM cell layer and fully connected layers. The inputs and outputs of the LSTM cells contain $2N$ nodes representing N real and N imaginary values.

B. Forward propagation

We replaced the traditional LSTM cell with sparse matrices in the DVM to reduce network complexity. In the LSTM cell, we implemented two types of matrices derived from DVM factorization, substituting the traditional fully connected weight matrices. These include a diagonal matrix (D_{2N}) and a Fourier matrix (F_{2N}). In a conventional LSTM cell [30], the forward propagation process is directed by three essential gates: the forget gate, the input gate, and the output gate. These gates, in conjunction with the cell state, enable the LSTM to adeptly handle long-term dependencies by selectively preserving or discarding information [30].

The following matrix structure is used to construct the S-LSTM neural network. At each time step t, the forget gate f_t can be computed using the previous cell hidden state h_{t-1} as

$$f_t = \sigma(D_{f_1}x_t + D_{f_2}h_{t-1} + b_f)$$

where, $x_t \in \mathbb{R}^{2N}$ represents the input, while $h_{t-1} \in \mathbb{R}^{2N}$ represents the previous hidden state. D_{f_1}, $D_{f_2} \in \mathbb{R}^{2N \times 2N}$ denote the diagonal weight matrices, and b_f refers to the bias term. The function $\sigma(.)$ represents the activation function. During the training of the S-LSTM network, only the weights along the diagonal of the weight matrices (i.e. D_{f_1}, D_{f_2}) were trained, while the remaining weights were kept at zero.

The input gate i_t determines which new information should be added to the cell state [30], while the cell candidate $\tilde{C}_t$ represents the potential updates. These are given by:

$$i_t = \sigma(F_{i_1}x_t + F_{i_2}h_{t-1} + b_i)$$
$$\tilde{C}_t = tanh(F_{c_1}x_t + F_{c_2}h_{t-1} + b_c)$$

979-8-3503-5111-8/24/$31.00 © 2024 IEEE

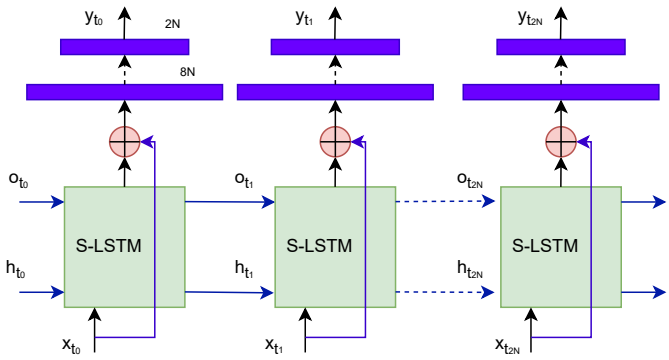

Fig. 1: The diagram illustrates the S-LSTM architecture, where consecutive S-LSTM units are linked by a sequence of data across time stamps $t_0, t_1, \cdots, t_{2N}$. Inputs $x_{t_0}, x_{t_1}, \ldots, x_{t_{2N}}$ are supplied to the corresponding S-LSTM cells. The hidden states $h_{t_0}, h_{t_1}, \ldots, h_{t_{2N}}$ move through each cell to the next, while the output states $o_{t_0}, o_{t_1}, \ldots, o_{t_{2N}}$ are transmitted forward. Moreover, skip connections allow direct information flow between alternate layers, improving gradient flow during backpropagation. This approach ensures more comprehensive feature extraction and better information retention across layers, with the final prediction at each state via $y_{t_0}, y_{t_1}, \ldots, y_{t_{2N}}$

Where F_{i_1}, $F_{i_2} \in \mathbb{R}^{2N\times 2N}$ are the Fourier matrices that use radix factorization to reduce complexity. The updated cell state C_t can be computed as:

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

where $\odot$ denotes element-wise multiplication. Finally, the output gate o_t determines the hidden state h_t, which is a filtered version of the current cell state passed through a tanh activation:

$$o_t = \sigma(D_{o_1} \cdot x_t + D_{o_2} \cdot h_{t-1} + b_o)$$
$$h_t = o_t \odot \tanh(C_t)$$

Where, D_{o_1} and D_{o_2} denote the diagonal weight matrices. The high level design of the network architecture is shown in the Fig. 1.

We introduce a skip link connecting the inputs to the S-LSTM output after the S-LSTM layer. The primary goal of the S-LSTM layer is to identify the time series characteristics necessary for computing the output. To compute the output following the skip connection, we incorporate several additional dense layers. Additionally, we decompose the weight matrices into diagonal and Fourier matrices between the final layers. The forward propagation equations for dense layers can be derived as follows.

Let $o_t \in \mathbb{R}^{2N}$ be the output vector of the S-LSTM at time step t. The first dense layer has $8N$ units, and the last dense layer has $2N$ units. Let's assume $W_1 \in \mathbb{R}^{8N\times 2N}$and $W_2 \in \mathbb{R}^{2N\times 8N}$ are the weight matrices of the dense layers, and $b_1 \in \mathbb{R}^{8N}$and $b_2 \in \mathbb{R}^{2N}$ are the bias terms in each layer. The skip connection $s_t = x_t + o_t \in \mathbb{R}^{2N}$ is the addition of the input and the S-LSTM output o_t. Thus, the forward propagation equation for the dense layers can be shown via:

$$a_t = \sigma(W_1 s_t + b_1) \qquad y_t = W_2 a_t + b_2,$$

where $a_t \in \mathbb{R}^{8N}$ is the output of the first dense layer after applying the activation function. To reduce the complexity of the dense layers, we utilize the DVM factorization in [1] for weight matrices, as explained in our previous work [29] s.t. $W_1 = \begin{bmatrix} w_1^{(1)} \\ w_1^{(2)} \end{bmatrix}$, $W_2 = \begin{bmatrix} w_2^{(1)}, w_2^{(2)} \end{bmatrix}$. Where $w_1^{(p)} \in \mathbb{R}^{4N\times 2N}$ and $w_2^{(p)} \in \mathbb{R}^{2N\times 4N}$ represent submatrices in which p can be either 1 or 2. Each submatrix can be factorized into the following matrices [29].

$$w_1^{(p)} = \breve{D}_{2M} F_{2M} J_{2M\times 2N} \hat{D}_{2N}$$
$$w_2^{(p)} = \hat{D}_{2N} [J_{2M\times 2N}]^T F^*_{2M},$$

where $M = 2N$, $J_{M\times N} = \begin{bmatrix} I_N \\ 0_N \end{bmatrix}$, I_N is the identity matrix, and 0_N is the zero matrix. Consequently, in the backpropagation process, we only train the weights along the diagonal of the diagonal weight matrices. For Fourier matrices, we employ the well-known FFT factorization [31], over $\lambda \in [1, 2, ..., log(2N)]$ steps.

C. Backpropagation

In the context of the Backpropagation algorithm, we utilize TensorFlow's automatic differentiation capabilities [32] to compute gradients and perform weight updates within the proposed S-LSTM. This process utilizes TensorFlow's *autodiff* functions, which track operations and calculate gradients during the forward pass. Subsequently, these gradients are employed in the backpropagation in time-stamped through batch learning [33] in order to update the model's weights.

III. Experimental Setup for Data Acquisition

Fig. 2 shows the experimental system for data acquisition, highlighting the overall design based on a 32-element antenna array [34]. Key subsystems include the antenna array, RF receiver chain, and digital hardware unit which is field programmable gate array (FPGA) based. The RF front-end integrates a 32-element ULA at 5.8 GHz with 32 direct conversion RF receivers. A centralized local oscillator (LO) supplies signals via a 32-output power divider connected to a low-phase-noise oscillator. Each receiver features a 16 dB low-noise amplifier (LNA) with a 2.4 dB noise figure, followed by a 4.7 GHz to 6 GHz bandpass filter. The band-limited signal is downconverted to low-IF and further amplified by 30 dB, and digitized using two ADC16x250-8 ADC cards (16 channels, 8-bit, 250 MHz).

A. Re-configurable Open Architecture Computing Hardware

ROACH-2 [35] has been used in our systems to sample the intermediate frequency (IF) signal and data acquisition. The ROACH board, equipped with a Xilinx Virtex-6 sx475t

979-8-3503-5111-8/24/$31.00 © 2024 IEEE

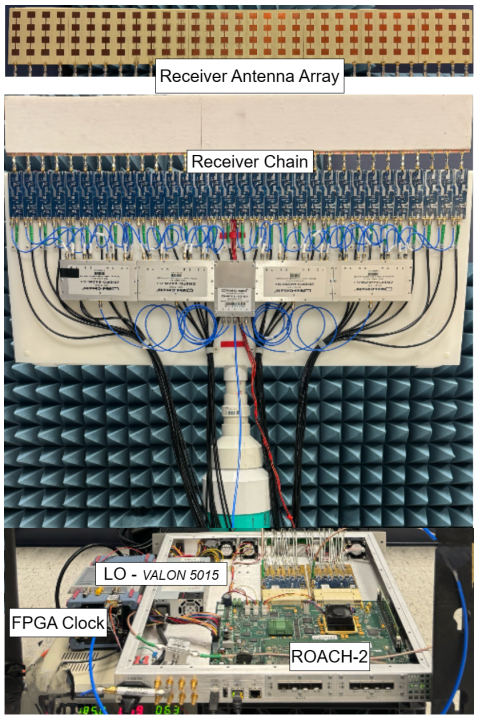

Fig. 2: Multi-channel 5.8 GHz receiver with fully digital back-end processing using ROACH-2 FPGA board. The system was custom-designed and built by Madanayake's group at FIU.

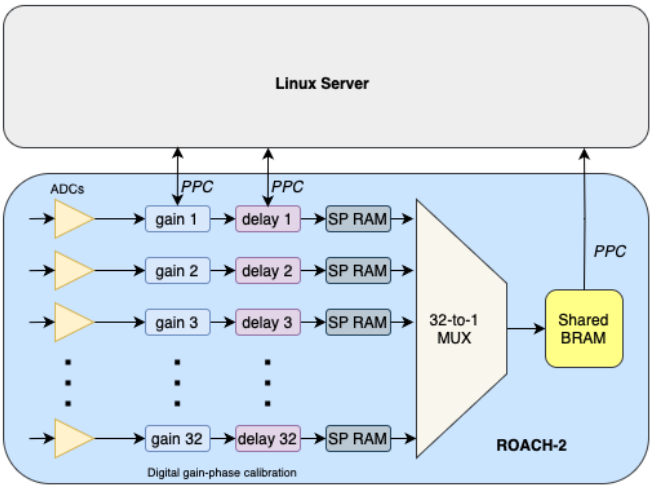

Fig. 3: Digital hardware architecture for data acquisition, featuring digital gain calibration, storage in single-port RAM, data routing through a MUX, and channel data accessed via BRAM software.

FPGA, was integrated with two ADC cards, each supporting 16 channels. A double ridge guide horn antenna (A.H Systems SAS-571) transmitted tones at 5.705, 5.71, and 5.715 GHz, while the LO was set to 5.70 GHz, producing IF signals at 5, 10, and 15 MHz. These downconverted signals were sampled by the ADC at 200 MHz to digitize the data. A VALON 5015 frequency synthesizer (10 MHz to 15 GHz) provided the LO signal, while a NOISE XT low-jitter clock synthesizer (2 MHz to 7 GHz) generated the clock signals for the FPGA and ADC. Measurements were conducted with approximately 20 m separation between the transmitter and receiver. The digital acquisition system, as depicted in Fig. 3, comprises two key subsystems: A) digital gain and phase calibration and B) data acquisition and sampling.

B. Digital Gain and Phase Calibration

In order to compensate for the discrepancies in between the 32 direct conversion RF receiver chains, the sampled 32 signal streams were digitally calibrated. The idea behind the digital gain-phase calibration is to retrieve 32 equal sampled signal streams when the antenna array perceives the same wave at its 32 elements. A double-ridge guided-horn antenna was placed at a distance of 14m at the direct broadside angle. This ensures that the receiver antenna array receives the same planar wave at its all 32 elements. Whilst the horn antenna is transmitting, each 32 sampled stream is multiplied by a number between 0-255 to receive normalized amplitude waveforms. After such gain calibration, the digital phase calibration utilizes integer delay filters to match the phases of the 32 streams which are capable of delaying each element sample stream by 0-32 clock cycles.

C. Data Acquisition in 32 Channel ADC

Retrieval of the data utilized the 1 Gbps PPC (Power PC) link of the ROACH-2 system which provides common registers between the FPGA fabric and the ROACH-2 control fabric (software registers). Once the data is written into software registers, the PPC control logic sends the written data to the host Linux server using Ethernet. The ADCs generate 8-bit samples in a rate of 200 MSps. A total of 1024 samples from each element were captured and stored in 1024 distinct memory addresses within each of the 32 Xilinx single-port RAM blocks. The data stored in each RAM block were subsequently multiplexed into software registers sequentially, allowing for transmission via the PPC (PowerPC) link. This was repeated 200 times for the targeted 4 IF frequencies. The antenna array was rotated from an angle of $-90°$ to $90°$ with $1°$ increments relative to the $0°$ broadside direction, while the transmitting horn antenna was maintained in a fixed broadside position($0°$), simulating signal reception from various angles which resulted in a 5-D array of data.

IV. S-LSTM Training and Simulations for Beamformers

The training and testing of the S-LSTM model for beamforming were conducted using a dataset composed of simulated beamforming patterns and corresponding desired beam

979-8-3503-5111-8/24/$31.00 © 2024 IEEE

outputs as stated in Section III. The training data was obtained as time series data, and 64-time steps were created per batch. We first transformed the raw input data into a fixed-point representation to ensure numerical stability and compatibility with hardware implementations. This transformation converts the data into a fixed-point format, dividing the input values by 2^9. This ensures that the data falls within a smaller, normalized range suitable for efficient computation during training. The transformed data becomes the new input for the training process. Since the received signal contains only real-valued components, we applied the Hilbert transform to obtain the corresponding analytic signal, consisting of real and imaginary parts. The Hilbert transform was performed along the time axis.

From the original dataset, which has a dimension of $3 \times 180 \times 200 \times 1024 \times 64$, where 3 corresponds to the number of different frequencies, 180 to the angle values, 200 to the number of samples, 1024 to the time steps, and 64 to the antenna element features, a smaller subset was extracted for model training. The extracted dataset has a reduced dimension of $3 \times 180 \times 1 \times 64 \times 64$, where the number of samples was reduced from 200 to 1, and the time steps were shortened from 1024 to 64.

To determine the output values, it is necessary to conduct a multiplication of the input values by the scaled DVM matrix $\tilde{A}_N = [\alpha^{kl}]_{k,l=0}^{N-1}$, where $\alpha = e^{-j\omega\tau}$, ω and τ denote the temporal frequency and delay, respectively [1], [4]. In the case of each incoming vector x_t, the output y_t can be derived as $\tilde{A}_N x_t$. In the given equation, we assume that $\tau = \frac{1}{f_{\max} N}$, where $f_{\max} = 100$ MHz and $\omega = 2\pi f$, with f being variable based on the frequency values (i.e., 5, 10, 15 MHz). Accordingly, the calculation of separate $\tilde{A}_N$ matrices will be required for each frequency. Upon obtaining output data that matches the input's dimensions, we proceed to construct the final dataset. This involves reshaping the data into a $540 \times 64 \times 64$ matrix. Furthermore, we have created a validation/testing dataset with dimensions of $27 \times 64 \times 64$ to verify and evaluate the trained model. The input and output matrix will be used for training the S-LSTM model.

The proposed architecture consists of a sequential S-LSTM network designed to process input data of shape $m_b \times 64 \times 64$, where m_b is the batch size. S-LSTM comprises 64×64 diagonal and Fourier matrices, followed by an intermediate hidden layer with 256 units and an output layer with 64 units. The Leaky ReLU activation function, utilizing a minimal scaling factor of 0.02, is employed to characterize activations within the layers of the learning process, all while maintaining the linear nature of the problem without introducing excessive nonlinearity. In our simulation, we considered $\lambda = 1$ as described in the forward propagation within Section II. The simulations are conducted using *Python version 3.10.12* Google Compute Engine backend(*Colab*) and *Tensorflow version 2.17.0 with Keras* as the backend. The model is compiled using the *Adam optimizer* with a learning rate of 0.01, and the mean squared error(MSE) defined by equation 2 is used as the loss function to measure the error between the predicted($\hat{y}_i$) and actual values(y_i). The training was performed for 400 epochs with $m_b = 27$.

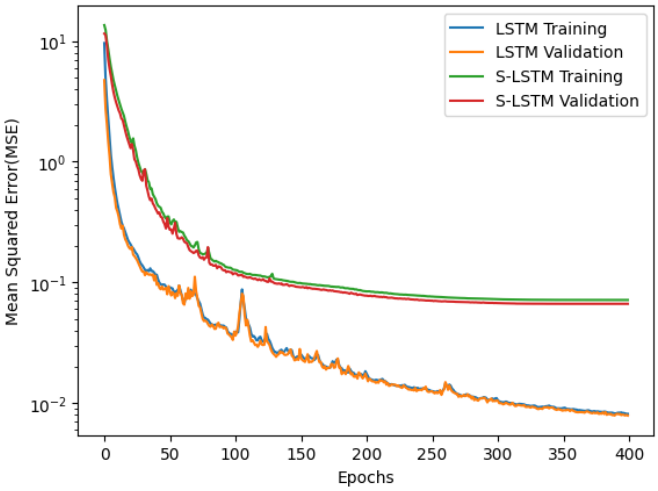

Fig. 4: Training and Validation MSE comparison between LSTM and S-LSTM models over 400 epochs. The LSTM model (blue and orange) demonstrates a higher initial MSE, while S-LSTM (green and red) indicates efficient convergence with fewer variations, particularly in the early epochs.

$$MSE = \frac{1}{m_b} \sum_{i=1}^{m_b} (y_i - \hat{y}_i)^2 \tag{2}$$

The proposed S-LSTM model's performance was compared to a conventional LSTM network with fully connected weight matrices, both following the same overall architecture and structure as shown in Fig.4. In the conventional network, the fully connected weight matrices were applied at each layer, without the structured weight constraints that are inherent to the S-LSTM architecture. The training and validation MSE of both models can be shown in Fig.4

Model	Training MSE	Validation MSE	No. of Weights
LSTM	8.0×10^{-3}	7.9×10^{-3}	66112
S-LSTM	7.8×10^{-2}	7.9×10^{-2}	42688

TABLE I: Final MSE for LSTM and S-LSTM models

This structured approach in the S-LSTM model allows it to predict beamformed output effectively while reducing computational complexity. The introduction of structured weight matrices in the S-LSTM enables more efficient representation learning, leading to a significant performance requiring both temporal and spatial feature extraction.

In the simulation results as shown in Table I and Fig. 4, the S-LSTM model achieved MSE of 10^{-2} showing a progressive decline in error rates across epochs. Furthermore, for $\lambda = 1$, the S-LSTM model can save up to 35% in model weights compared to conventional LSTM architecture. Increasing the value of λ enables us to save more weight, resulting in a reduced-complexity network structure., but it sacrifices the MSE. Further analysis suggests that improving the diversity

979-8-3503-5111-8/24/$31.00 © 2024 IEEE

of training data set and fine-tuning model parameters could enhance performance on more challenging test cases composing multiple frequencies and delays.

V. Conclusion

We have proposed an S-LSTM network to realize multi-beam beamformers while imposing the DVM structure along with its factorization to reduce the complexity of the network from $\mathcal{O}(N^2L)$ to $\mathcal{O}(N^sL)$, where $1 < s < 2$ and L is the layers of the network. Our approach allowed the model to effectively extract patterns and reduce complexity as compared to the conventional LSTM. The error rates of the proposed S-LSTM reached an order of 10^{-2}, saving 35% of the weights compared to the conventional LSTM network. In our future work, we plan to retrain the S-LSTM using a diverse dataset, i.e., multiple frequencies and delays, to realize time-advanced multibeam beamformers.

References

[1] S. M. Perera, L. Lingsch, A. Madanayake, S. Mandal, and N. Mastronardi, "Fast dvm algorithm for wideband time-delay multi-beam beamformers," *the IEEE Transactions on Signal Processing*, vol. 70, no. 5913-5925, 2022.

[2] K. W. Masui, J. R. Shaw, C. Ng, K. M. Smith, K. Vanderlinde, and A. Paradise, "Algorithms for FFT beamforming radio interferometers," *The Astrophysical Journal*, vol. 879, no. 16, 2019.

[3] S. R. Seydnejad and S. Akhzari, "Performance evaluation of pre- and post-FFT beamforming methods in pilot-assisted SIMO-OFDM systems," *Telecommunication Systems*, vol. 61, no. 3, pp. 471–487, 2016.

[4] S. Perera, V. Ariyarathna, N. Udayanga, A. Madanayake, G. Wu, L. Belostotski, R. Cintra, and T. Rappaport, "Wideband n-beam arrays with low-complexity algorithms and mixed-signal integrated circuits," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 2, pp. 368–382, 2018.

[5] S. M. Perera, A. Madanayake, and R. Cintra, "Efficient and self-recursive delay vandermonde algorithm for multi-beam antenna arrays," *IEEE Open Journal of Signal Processing*, vol. 1, no. 1, pp. 64–76, 2020.

[6] ——, "Radix-2 self-recursive algorithms for vandermonde-type matrices and true-time-delay multi-beam antenna arrays," *IEEE Access*, vol. 8, pp. 25 498–25 508, 2020.

[7] A. E. A. Blomberg, A. Austeng, and R. E. Hansen, "Adaptive beamforming applied to a cylindrical sonary array using and interpolated array transformation," *IEEE Open Journal or Oceanic Engineering*, vol. 37, no. 1, pp. 25–34, 2012.

[8] X. Wu, J. Luo, G. Li, S. Zhang, and W. Sheng, "Fast wideband beamforming using convolutional neural network," *Remote sensing (Basel, Switzerland)*, vol. 15, no. 3, pp. 712–, 2023.

[9] H. J. Sharahi, C. N. Acconcia, M. Li, A. Martel, and K. Hynynen, "A convolutional neural network for beamforming and image reconstruction in passive cavitation imaging," *Sensors*, vol. 23, no. 21, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/21/8760

[10] T. Hori, Z. Chen, H. Erdogan, J. R. Hershey, J. Le Roux, V. Mitra, and S. Watanabe, "Multi-microphone speech recognition integrating beamforming, robust feature extraction, and advanced dnn/rnn backend," *Computer speech & language*, vol. 46, pp. 401–418, 2017.

[11] H. Che, C. Li, X. He, and T. Huang, "A recurrent neural network for adaptive beamforming and array correction," *Neural Networks*, vol. 80, pp. 110–117, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608016300405

[12] K. P. Deorukhkar and S. Ket, "Image captioning using hybrid lstm-rnn with deep features," *Sensing and imaging*, vol. 23, no. 1, 2022.

[13] B. Cauchi, K. Siedenburg, J. Santos, T. Falk, S. Doclo, and S. Goetze, "Non-intrusive speech quality prediction using modulation energies and lstm-network," *IEEE/ACM transactions on audio, speech, and language processing*, vol. 27, no. 7, pp. 1151–1163, 2019.

[14] Q. Yuan, Y. Dai, and G. Li, "Exploration of english speech translation recognition based on the lstm rnn algorithm," *Neural computing & applications*, vol. 35, no. 36, pp. 24 961–24 970, 2023.

[15] S. C. Kolen, John F.; Kremar, *A Field Guide to Dynamical Recurrent Networks*. IEEE Press, 2001.

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

[17] D. H. Hopfe, K. Lee, and C. Yu, "Short-term forecasting airport passenger flow during periods of volatility: Comparative investigation of time series vs. neural network models," *Journal of air transport management*, vol. 115, pp. 102 525–, 2024.

[18] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to Forget: Continual Prediction with LSTM," *Neural Computation*, vol. 12, no. 10, pp. 2451–2471, 10 2000. [Online]. Available: https://doi.org/10.1162/089976600300015015

[19] Z. Meng, S. Watanabe, J. R. Hershey, and H. Erdogan, "Deep long short-term memory adaptive beamforming networks for multichannel robust speech recognition," *arXiv.org*, 2017.

[20] P. Bhadauria, R. Kumar, and S. Sharma, "Beamforming in vehicle to infrastructure scenario with respect to lstm and nar method," *Journal of electrical engineering & technology*, vol. 19, no. 1, pp. 641–654, 2024.

[21] L. Yaohui, X. Yuan, T. Shengyu, Z. Yanmin, and Z. Yucheng, "Virtual array element beamforming algorithm based on lstm neural network," *Journal of Physics: Conference Series*, 2024.

[22] H. Xiang, B. Chen, M. Yang, S. Xu, and Z. Li, "Improved direction-of-arrival estimation method based on lstm neural networks with robustness to array imperfections," *Applied intelligence (Dordrecht, Netherlands)*, vol. 51, no. 7, pp. 4420–4433, 2021.

[23] R. Kumar and H. Singh, "Performance dependency of lstm and nar beamformers with respect to sensor array properties in millimeter-wave v2i scenario," *Microwave and optical technology letters*, vol. 65, no. 3, pp. 859–865, 2023.

[24] R. U. Murshed, Z. B. Ashraf, A. H. Hridhon, K. Munasinghe, A. Jamalipour, and F. Hossain, "A cnn-lstm-based fusion separation deep neural network for 6g ultra-massive mimo hybrid beamforming," *arXiv (Cornell University)*, 2022.

[25] I. Rasheed, M. Asif, A. Ihsan, W. U. Khan, M. Ahmed, and K. Rabie, "Lstm-based distributed conditional generative adversarial network for data-driven 5g-enabled maritime uav communications," *arXiv (Cornell University)*, 2022.

[26] V. Elangovan, W. Xiang, and S. Liu, "A multiagency long short-term model beamforming prediction model for cellular vehicle to everything," *SAE international journal of connected and automated vehicles (Print)*, vol. 6, no. 4, pp. 459–472, 2023.

[27] S. R. Pavel and Y. D. Zhang, "Optimization of the compressive measurement matrix in a massive mimo system exploiting lstm networks," *Algorithms*, vol. 16, no. 6, pp. 261–, 2023.

[28] N. Han, I.-M. Kim, and J. So, "Lightweight lstm-based adaptive cqi feedback scheme for iot devices," *Sensors (Basel, Switzerland)*, vol. 23, no. 10, pp. 4929–, 2023.

[29] H. Aluvihare, S. M. Perera, A. MAdanayake, and X. Li, "A low-complexity structure-imposed neural network to realize multi-beam beamforming," *in review, IEEE Transactions on Aerospace and Electronic Systems*, 2024.

[30] S. Hochreiter, "Long short-term memory," *Neural Computation MIT-Press*, 1997.

[31] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex fourier series," *Math. Comp.*, vol. 19, pp. 297–301, 1965.

[32] T. Developers, "Automatic differentiation in tensorflow," https://www.tensorflow.org/guide/autodiff, 2023, accessed: 2024-09-13.

[33] 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.

[34] A. Madanayake, K. Lawrance, B. U. Kumarasiri, S. Sivasankar, T. Gunaratne, C. U. S. Edussooriya, and R. J. Cintra, "Design of multichannel spectrum intelligence systems using approximate discrete fourier transform algorithm for antenna array-based spectrum perception applications," *Algorithms*, vol. 17, no. 8, 2024. [Online]. Available: https://www.mdpi.com/1999-4893/17/8/338

[35] "The Collaboration for Astronomy Signal Processing and Electronics Research," 2024, [Accessed 23-05-2024]. [Online]. Available: https://casper.berkeley.edu/

979-8-3503-5111-8/24/$31.00 © 2024 IEEE