ICDT 2021

The Sixteenth International Conference on Digital Telecommunications

ISBN: 978-1-61208-835-8

April 18 - 22, 2021

ICDT 2021 Editors

Stan McClellan, Texas State University, USA

ICDT 2021

Forward

The Sixteenth International Conference on Digital Telecommunications (ICDT 2021) continued a series of events covering topics related to telecommunications aspects in multimedia environments. The scope of the conference was to focus on the lower layers of systems interaction and identify the technical challenges and the most recent achievements.

The conference served as a forum for researchers from both the academia and the industry, professionals, and practitioners to present and discuss the current state-of-the art in research and best practices as well as future trends and needs (both in research and practices) in the areas of multimedia telecommunications, signal processing in telecommunications, data processing, audio transmission and reception systems, voice over packet networks, video, conferencing, telephony, as well as image producing, sending, and mining, speech producing and processing, IP/Mobile TV, Multicast/Broadcast Triple-Quadruple-play, content production and distribution, multimedia protocols, H-series towards SIP, and control and management of multimedia telecommunications.

High quality software is not an accident; it is constructed via a systematic plan that demands familiarity with analytical techniques, architectural design methodologies, implementation polices, and testing techniques. Software architecture plays an important role in the development of today's complex software systems. Furthermore, our ability to model and reason about the architectural properties of a system built from existing components is of great concern to modern system developers.

Performance, scalability and suitability to specific domains raise the challenging efforts for gathering special requirements, capturing temporal constraints, and implementing service-oriented requirements. The complexity of the systems requires an early stage adoption of advanced paradigms for adaptive and self-adaptive features.

Online monitoring applications, in which continuous queries operate in near real-time over rapid and unbounded "streams" of data such as telephone call records, sensor readings, web usage logs, network packet traces, are fundamentally different from traditional data management. The difference is induced by the fact that in applications such as network monitoring, telecommunications data management, manufacturing, sensor networks, and others, data takes the form of continuous data streams rather than finite stored data sets. As a result, clients require long-running continuous queries as opposed to one-time queries. These requirements lead to reconsider data management and processing of complex and numerous continuous queries over data streams, as current database systems and data processing methods are not suitable.

Event stream processing is a new paradigm of computing that supports the processing of multiple streams of event data with the goal of identifying the meaningful events within those streams.

We take here the opportunity to warmly thank all the members of the ICDT 2021 technical program committee, as well as all the reviewers. The creation of such a high quality conference program would not have been possible without their involvement. We also kindly thank all the authors who dedicated much of their time and effort to contribute to ICDT 2021. We truly believe that, thanks to all these efforts, the final conference program consisted of top quality contributions. We also thank the members of the ICDT 2021 organizing committee for their help in handling the logistics of this event.

ICDT 2021 Chairs

ICDT 2021

Committee

Table of Contents

Optimization of Sparse Matrix Arithmetic Operations and Performance Improvement using FPGA

Dinesh Kumar Murthy
Ingram School of Engineering
Texas State University
San Marcos, TX, USA
d_m410 @txstate.edu

Semih Aslan
Ingram School of Engineering
Texas State University
San Marcos, TX, USA
aslan@txstate.edu

Abstract — **The increasing importance of sparse connectivity representing real-world data has been exemplified by recent work in the areas of graph analytics, machine language, and high-performance computing. Sparse matrices are the critical component in many scientific computing applications, where increasing sparse operation efficiency can contribute significantly to improving overall system efficiency. The main challenge lies in efficiently handling the nonzero values by storing them using a specific storage format and then performing matrix operations, taking advantage of the sparsity. This paper proposes an optimized algorithm for performing sparse matrix operations in storage and hardware implementation on Field-Programmable Gate Arrays (FPGAs). The results are obtained from implementing the sparse algorithm on hardware for matrices of different sizes. Sparsity percentages and sparsity patterns achieved low latency and high throughput compared with the standard algorithm. Further, the number of resources utilized was primarily reduced, enabling the FPGAs to focus on larger, more interesting problems.**

Keywords - Sparse matrix; latency; throughput; memory; FPGA; hardware architecture.

I. INTRODUCTION

We live in a "big data" era where graph processing has become increasingly important, because the amount of data generated and collected from many real-world applications such as sensors, social networks, portable devices. Graphs are used to model many systems of interest to engineers and scientists; today, useful information is being extracted. Once entered into a computer, the data no longer looks like a graph. Often, it is in the form of a sparsely populated matrix with mostly zeros compared to nonzeros [1] [2]. When the number of zeros is relatively large, efficient data structures are required. Numerous studies have addressed finding new algorithms for sparsely distributed matrices.

When obtaining information in a graph algorithm with a small number of nonzero entries but millions of rows and columns, memory would be wasted by storing redundant zeros [3][4]. There are two ways one would take advantage of the sparsity of a matrix: one would be to store the nonzero elements of a matrix, and the second is to process only the nonzero elements of a matrix [5]. However, large graphs are hard to deal with as inputs, and outputs limit the state-of-the-art graph processing systems. For the most part, Central Processing Units (CPUs) and Graphics Processing Units (GPUs) compute well on a performance scale. However, there is a small niche where an FPGA has been an attractive platform that can handle the same computation task for acceleration and achieve high performance with low power computation for many applications. Specifically, due to the memory access pattern of graph problems, it is still challenging to develop high throughput and energy-efficient FPGA design [6].

This paper's primary goal is to develop an efficient algorithm for various sparse matrix arithmetic operations like addition, subtraction, multiplication, element by element multiplication, and square root. By utilizing the sparse matrix storage method, storage requirements should be reduced when compared to a standard matrix operation algorithm. The main goal is to improve efficiency in terms of latency and throughput [7][8]. The performance analysis is calculated based on the design that minimizes gate count, area, and reducing the number of multipliers and adders. The architectural design is scalable, simple to implement, and capable of handling matrices of various sizes. This paper is organized as follows. In Section II, the basics of matrix operations are discussed. In Section III, the proposed algorithm and system design are explained. FPGA simulation and mapping are discussed in Sections IV and V, respectively. Sections VI and VII show the detailed performance analysis and the results. This paper concludes in Section VIII.

II. MATRIX OPERATION

The design performs sparse matrix addition operations of two sparse matrices where only the nonzero values are stored, and the required operation is performed. It is performed by using two algorithms:

- A symbolic algorithm, which determines the structure of the resulting matrix.
- A numerical algorithm, which determines the values of nonzero elements using the knowledge of their positions.

$$c_{i,j} = (a_{i,j}) + (b_{i,j}) \tag{1}$$

Each nonzero (nz) element of matrices **A** and **B** needs one floating-point operation, so the total number of floating-point operations to be performed is the number of nz elements. When the computation is completed, the number of nz output operations is written on the external memory.

III. SYSTEM DESIGN

A. Storage Format

The proposed architectural algorithm performs sparse matrix addition in which the number of rows and number of columns of two matrices is equal. A parallel implementation of the addition with enough fast memory algorithm, is proposed. Consider a matrix addition of **A**+**B**, where **A** has a density s_1 percentage with size n×n (a square matrix is considered), and matrix **B** has a density s_2 percentage with size n×n. Density s_x percentage is defined as the number of nonzero elements to the total number of elements in the matrix n^2. The matrix addition performs the operation row-wise and column-wise throughout the matrix only for the nonzero elements present, leaving behind the zeros. When an addition operation must be performed on both input matrices, the number of rows and columns are first compared to determine if they are equal, i.e., both the matrices should be of the same size. An additional operation cannot be performed if the matrices are of different sizes. Then, the matrix elements are checked row-wise and column-wise from top-to-bottom order for nonzero elements, as shown in Fig. 1. Two separate counters, *A_count* and *B_count*, are used to increment the row and column for both the **A** and **B** input matrices. This keeps incrementing from n to n+1 for the size of the matrix. The algorithm for the sparse matrix addition **A**+**B** is presented in Fig. 2.

The most important part of this algorithm is the index comparison, which is represented as *A_index* for matrix **A** and *B_index* for matrix **B**. After first storing the nonzero elements, the row value of matrix **A** is compared with the row value of matrix **B** for each operation. If the index of *A_sr* is equal to the index of *B_sr*, then the next step of comparing the column value of both matrices is performed. If the index of *A_sc* is equal to the index of *B_sc*, then a matrix addition operation is performed. The *VAL* array of the respective row and column, i.e., *A_sv* and *B_sv*, are added to each other as a sum. The assumption is made that the nonzero element is located anywhere in the matrix and is highly sparse. Finally, the nonzero element of input matrix **A** that does not match the row and column of matrix **B** is given directly as the sum in the output matrix.

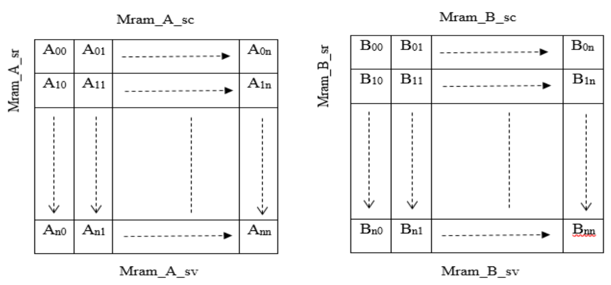


Figure 1. Representing row and column access of matrices

B. Design Algorithm

```
A → n×n sparse matrix
B → n×n sparse matrix
for i → 0 to MAT_SIZE do
        if (A[i]≠ 0) then
                Indexing row and column = i + 1
                A_sv [i] =A [i]
                A_index = A_count + 1
        end
        if (B[i] ≠ 0) then
                Index2rc = i + 1
                B_index = B_count + 1
                B_sv [ i] = B [ i]
        end
        if((A_sr[A_index]==B_sr[B_index])&&
        (A_sc[A_index] ==B_sc[B_index])) do
                Row <= A_sr [A_index]
                Col <= A_sc [A_index]
                Sum <= A_sv [A_index] + B_sv [B_index]
        end
        if (A_sv [A_index] ≠0) then
                Row <= A_sr [A_index]
                Col <= A_sc [A_index]
                Sum <= A_sv [A_index]
        end
        if (B_sv [B_index] ≠ 0) then
                Row <= B_sr[B_index]
                Col <= B_sc[B_index]
                Sum <= B_sv[B_index]
        end
end
```

Figure 2. Algorithm for Sparse Matrix Addition Operation

C. Memory Control

Memory control plays a crucial part in architectural design. The memory control block oversees real enable sign and assigning a memory access address, so accurate data is acquired by the algorithm logic through all stages. The operation is performed at the row level, so throughput is not affected by the latency of data reading while performing the arithmetic operation.

As shown in Fig. 3, the memory control module is designed as a finite state machine. At the beginning of the finite state machine, reset is set to Idle, which resets all the registers to predefined values. After this state, the matrix values are inferred for writing data to the Block RAMs (BRAMs), which triggers the memory control transition from the Idle State to the Read and Write state.

Once the elements are written, it calculates the nonzero values by checking row-wise and column-wise throughout the array by increasing the pointer locations by one. With the nonzero elements located successfully, separate arrays are created for matrix storage format in the order of *ROW*, *COL*, and *VAL*. As the name indicates, the row and column values are stored starting from 0 to the maximum, and the respective integer values are written accordingly. Once the sparse matrix storage format is generated, the arithmetic design algorithm checks the *ROW* and *COL* arrays and performs addition if both are equal. Otherwise, the design sends the values directly to the output, since addition is not required there. When the system performs all arithmetic operations, the finite state goes back to Idle State. By operating this way, only the nonzero elements undergo additional processes, and in the final state, the output is sent back.

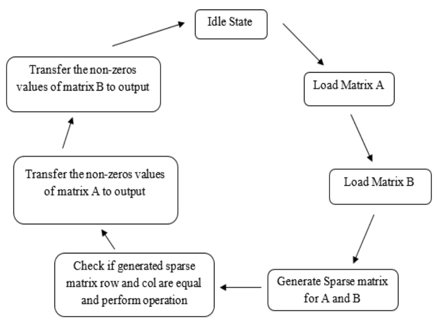
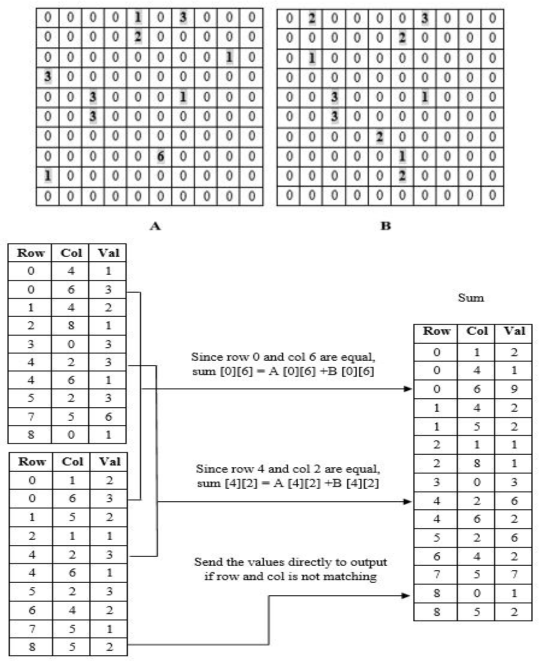

Idle State
Load Matrix A
Load Matrix B
Generate Sparse matrix for A and B
Check if generated sparse matrix row and col are equal and perform operation
Transfer the non-zeros values of matrix A to output
Transfer the non-zeros values of matrix B to output

Figure 3. State transition diagram of the memory control

For example, if there are two matrices **A** and **B** with ten nonzeros each, as shown in Fig. 4. The state machine will read the values and write the nonzero values in the storage format illustrated above. The necessary arithmetic operation is then performed from the Idle state, staying in hold for the state until it receives an end signal from the controller.



A

0	0	0	0	1	0	3	0	0	0
0	0	0	0	2	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0
3	0	0	0	0	0	0	0	0	0
0	0	3	0	0	0	1	0	0	0
0	0	3	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	6	0	0	0	0
1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

B

0	2	0	0	0	0	3	0	0	0
0	0	0	0	0	2	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	3	0	0	0	1	0	0	0
0	0	3	0	0	0	0	0	0	0
0	0	0	0	2	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	2	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Row	Col	Val
0	4	1
0	6	3
1	4	2
2	8	1
3	0	3
4	2	3
4	6	1
5	2	3
7	5	6
8	0	1

Row	Col	Val
0	1	2
0	6	3
1	5	2
2	1	1
4	2	3
4	6	1
5	2	3
6	4	2
7	5	1
8	5	2

Sum

Row	Col	Val
0	1	2
0	4	1
0	6	9
1	4	2
1	5	2
2	1	1
2	8	1
3	0	3
4	2	6
4	6	2
5	2	6
6	4	2
7	5	7
8	0	1
8	5	2

Since row 0 and col 6 are equal, sum [0][6] = A [0][6] +B [0][6]
Since row 4 and col 2 are equal, sum [4][2] = A [4][2] +B [4][2]
Send the values directly to output if row and col is not matching

Figure 4. Operational example for the addition of sparse matrices

IV. SIMULATION

Random matrices of various sizes are generated using MATLAB with variation in sparsity pattern and sparsity percentage. Additionally, two parameters, *MAT_SIZE* (size of the matrix n×n) and *ELEMENT_SIZE* (number of bits of the integer) are included with the design, which is passed to the input as known information.

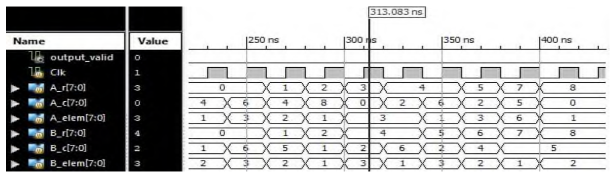


Figure 5. Waveform showing storage of sparse matrices

As shown in Fig. 5, the nonzero elements of the input matrices are stored to BRAMs in the format specified as two-dimensional arrays. The memory controller then reads the BRAMs to perform the required arithmetic operation.

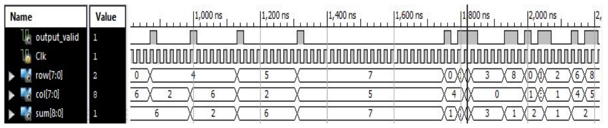


Figure 6. Waveform showing results of arithmetic operation (sum)

Fig. 6 shows the results of the addition operation in a simulation waveform. The algorithm is tested with multiple test values by varying the sparsity percentage and the golden result vectors generated using MATLAB.

V. FPGA MAPPING

Using Xilinx ISE Design Suite, the designed algorithm is implemented on the target device Xilinx Artix7 XC7A100T-1CG324C board, comprising of 15,850 logic slices and a maximum of 4,860 Kbits fast BRAM [9] [10]. The hardware implementation is split into two major top modules. The first module is designed to implement the sparse matrix arithmetic operations, and the second module is to implement a Universal Asynchronous Receiver Transmitter (UART) communication and data exchange between the PC and FPGA. Each of the top modules is subdivided into smaller modules to carry out specific operations with the other modules through internal signals as shown in Fig. 7.

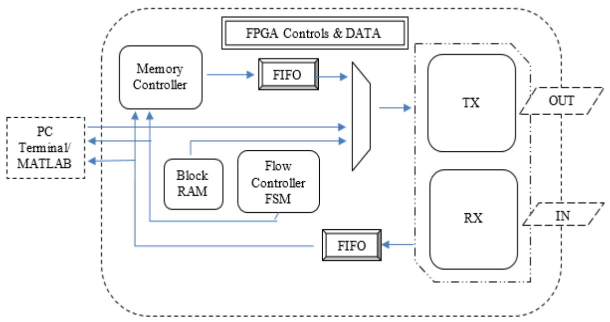


Figure 7. Block Diagram of the TX and RX Module

The transmitter module is used to transfer data over the UART device. It serializes a byte of data and transmits over a Transmit Data (TxD) line. The serialized data has 9600 Baud Rate, 8 data bits (least significant bit first), 1 Stop bit, and no parity. The receiver module double-registers the incoming data. This module makes sure all the bits are sent out. These modules expect the clock generated to be 100 MHz. The Phase-Locked Loop (PLL) is a control system that produces an output signal whose phase is related to an input signal. Keeping the input and output phases in lock steps, the input and output frequencies can be kept the same. These are widely used for synchronization purposes. For our hardware design, which operates at 20 MHz, the phase-locked loop is used to compensate for the required 100 MHz clock frequency. This IP core is generated using the design tool.

VI. PERFORMANCE ANALYSIS

The following metrics were calculated to show the algorithm's efficiency, such as latency, throughput, and resources utilized. Latency is the amount of time it takes to complete an operation, the time between reading the first element of the input matrix and writing the first element of the output matrix. Throughput is the number of such operations executed per unit of time.

The latency for matrix addition operation was significantly reduced, and high throughput was achieved using the proposed algorithm compared with the standard matrix algorithm. Table II illustrates the comparison of different test values with matrix sizes ranging from 10x10 to 100x100 with sparsity ranging from 1% to 10% for both proposed sparse and standard matrix algorithms for different operations.

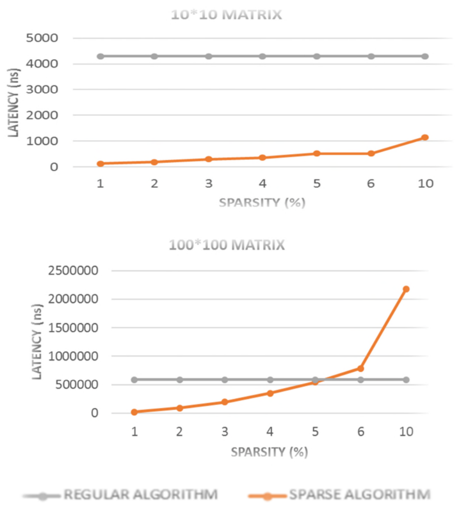


Figure 8. Latency for Sparse Matrix Addition

The comparison of latency calculated is plotted as a graph, which is shown in Fig. 8. The difference between the standard algorithm and the sparse algorithm is shown. Fig. 9. shows the difference in throughput between the two methods and shows that the proposed algorithm achieved high throughput.

After experimentation with different test values, there are improvements in latency and throughput for smaller matrices with high sparsity percentage and larger matrices with low sparsity percentage. Once the mapping of matrices is implemented on the FPGA platform, the resources utilized are shown in Table I.

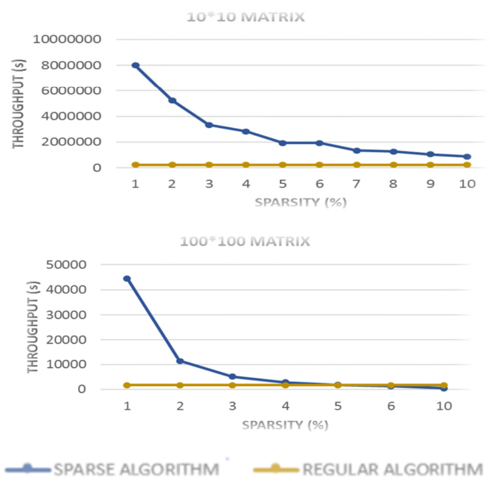


Figure 9. Latency for Sparse Matrix Addition

VII. RESULTS AND DISCUSSION

In most cases, it is evident that latency and throughput are directly dependent on the number of nonzero elements present in the matrix. The efficiency of the design can be further improved by increasing the frequency of the overall design clock. The maximum speedup of the design for any matrix depends on the number of rows and columns being processed. One primary purpose of this paper is to reduce the storage space used in an FPGA when implemented. This is also accomplished when the design is implemented in an Artix 7 FPGA board. The amount of resources utilized for the proposed sparse algorithm is less than the standard algorithm. The comparison is tabulated in Table I. The design uses only 3 percent of the total FPGA resources. Further, pipelining can be implemented to increase the computational speed of the system. For arithmetic operations performed on large matrices or memory-based algorithms and for small matrices, a pipelined algorithm will be quite efficient.

VIII. CONCLUSION

Today's applications require higher computational throughput and a distributed memory approach for real-time applications. This research is primarily focused on designing an optimized architecture for sparse matrix operations, allowing for more efficiency than standard operations. The functionality of the design is verified by different sets of test cases under a specific size. The system contains a memory control which fetches the data from memory and passes it on for various arithmetic operations. Research improvement in this area is needed to increase logic resources by a comparable increase in I/O bandwidth and on-chip memory capacity, especially when the matrix sparsity is unstructured and randomly distributed.

TABLE I. DESIGN RESOURCE UTILIZATION SUMMARY

Slice Logic utilization		
Number of Slice Registers	4,799 out of 126,800	3%
Number of Slice Look-up Tables (LUTs)	6,702 out of 63,400	10%
Slice Logic Distribution		
Number of occupied Slices	2,413 out of 15,850	15%
Input/Output (IO) Utilization		
Number of bonded IO Blocks	3 out of 210	1%
Specific Feature Utilization		
Number of Block RAM/FIFO	2 out of 270	1%

REFERENCES

[1] X. Lin and J. Xu, "Special Issue on Graph Processing: Technique and Applications," Data Sci. Eng., vol. 2, no.1, p. 1, 2017.

[2] A.Ching, S. Edunov, M. Kabiljo, D. Logothetis,and ,S. Muthukrishnan, “One Trillion Edges : Graph Processing at Facebook-Scale," , Proceedings of the VLDB Endownment, vol. 8, no. 12, pp. 1804-1815, 2015.

[3] M. Ryan, "FPGA Hardware Accelerators - Case Study on Design Methodologies and Trade-Offs", 2013. Thesis. Rochester Institute of Technology. Accessed from http://scholarworks.rit.edu/theses/959.

[4] T. Mattson et al., "Standards for graph algorithm primitives," IEEE High Performance Extreme Computing Conference (HPEC), Waltham, MA, USA, 2013, pp. 1-2, doi: 10.1109/HPEC.2013.6670338.

[5] S. Jain, N. Kumar, J. Singh, and M. Tiwari, “FPGA Implementation of Latency, Computational time Improvements in Matrix Multiplication,” International Journal of Computer Applications, 2014, vol.86, no.8, doi:10.5120/15007-3261.

[6] S. Aslan and J. Saniie, "Matrix Operations Design Tool for FPGA and VLSI Systems," 2016, Circuits and Systems, vol. 7, no.2, pp. 43–50, doi: 10.4236/cs.2016.72005.

[7] P. Grigoras, P. Burovskiy, E. Hung and W. Luk, "Accelerating SpMV on FPGAs by Compressing Nonzero Values," 2015 IEEE 23rd Annual International Symposium on Field-Programmable Custom Computing Machines, Vancouver, BC, Canada, 2015, pp. 64-67, doi: 10.1109/FCCM.2015.30.

[8] L. Zhuo and V. Prasanna, “Sparse Matrix-Vector multiplication on FPGAs,” In Proceedings of the 2005 ACM/SIGDA 13th international symposium on Field-programmable gate arrays *(FPGA '05).* ACM, New York, NY, USA, 63-74.

[9] B. Hamraz, N. Caldwell, and P. Clarkson "A Matrix-Calculation-Based Algorithm for Numerical Change Analysis”, IEEE Transaction on Engineering Management, Vol.60, No.1 February 2013.

[10] Nexys 4 DDR board – Reference Manual.

TABLE II. LATENCY AND THROUGHPUT CALCULATION

Matrix Size (n*n)	**Number of nonzero (nnz)**	**Sparsity**	**Sparse Algorithm**		**Standard Algorithm**	
			Latency (ns)	**Throughput**	**Latency (ns)**	**Throughput**
Matrix Addition						
1010	10	0.1	1137.169	879376.7681	4298.0515	232663.5688
20x20	32	0.08	8550.567	116951.3086	18688.1835	53509.74855
40x40	96	0.06	57464.964	17401.90771	93787.3685	10662.41666
60x60	144	0.04	123981.857	8065.696257	214929.95	4652.678698
100x100	100	0.01	22427.802	44587.51687	588369.806	1699.61135
Matrix Subtraction						
1010	9	0.09	911.1375	1097529.187	3205.4335	311970.2842
20x20	28	0.07	6156.6615	162425.6913	19199.5965	52084.42792
40x40	80	0.05	43638.191	22915.70702	38884.05	25717.4857
60x60	108	0.03	84721.1265	11803.43134	214411.526	4663.928375
100x100	100	0.01	70001.865	14285.33368	589749.829	1695.634235
Matrix Multiplication (Element-by-Element)						
1010	10	0.1	1107.282	903112.3056	3205.4335	311970.2842
20x20	36	0.09	9780.263	105482.3057	19199.5965	52084.42792
40x40	80	0.05	42519.648	23518.53901	38884.05	25717.4857
60x60	72	0.02	32073.866	31178.03136	214411.526	4663.9283
100x100	100	0.01	5152.62	18364.9265	589749.82	1695.6342

Optimized Architecture for Sparse LU Decomposition on Matrices with Random Sparsity Patterns

Dinesh Kumar Murthy
Ingram School of Engineering
Texas State University
San Marcos, TX, USA
d_m410 @txstate.edu

Semih Aslan
Ingram School of Engineering
Texas State University
San Marcos, TX, USA
aslan@txstate.edu

***Abstract* — This paper investigates a method for improving the performance of sparse Lower-Upper (LU) decomposition which is widely used to solve sparse linear systems of equations, appearing in many scientific and engineering application models. However, LU decomposition is considered a computationally expensive tool. When dealing with large sparse matrices, numerical decomposition takes more time using normal matrix LU implementation. The problem of interest here is the irregular sparsity pattern which limits performance gain. An efficient architecture for sparse LU decomposition is proposed for both symmetric and asymmetric matrices with random sparsity percentages and patterns. The algorithm spends time in simultaneous localization and mapping of the sparse matrix and then solving the linearized system. The performance of the algorithm with matrices of varying parameters is calculated and compared with a regular LU decomposition algorithm. In most cases, there are performance improvements in terms of speed, area, and power.**

Keywords – Pivoting; latency; linear systems; throughput; LU Decomposition; Field Programmable Gate Arrays (FPGAs).

I. INTRODUCTION

Numerical solutions of large linear systems are important for scientific and engineering applications like linear programming, circuit simulation, semiconductor device simulations, image processing, and power system modelling. Solving such systems of equations generally involves two methods: the direct method including Cholesky decomposition, LU decomposition, QR decomposition, and iterative methods. The Cholesky decomposition is a special form of LU decomposition which deals with symmetric positive definite matrices. Adapting these parallel architectures to solve large sparse linear system of equations is a main focus of research [1].

A number of software- and hardware-based approaches have been developed to obtain better solutions for LU decomposition. Software implementation includes a Super nodal approach which considers the matrix as sets of continuous columns with the same nonzero structure, and a Multifrontal approach organizing a large sparse matrix into a small dense matrix [2]. Field Programmable Gate Arrays (FPGAs) have unique advantages in solving these problems. Depending on the characteristics of the algorithm, an architecture is designed with reconfigurable computational resources and memory. The consumption of energy is reduced and is a platform for experimentation and verification. Though there are many FPGA-based architectures for dense matrices, only a few are proposed for sparse matrix decomposition [3][4]. The three main direct methods for sparse LU decomposition are left-looking, right-looking and count algorithms. The proposed FPGA-based architecture for sparse LU decomposition can efficiently decompose the sparse matrix with varying sparsity patterns. The architecture first factorizes the columns from the lower triangular part of the matrix in parallel with the rows from the upper triangular part of the matrix. The control structure performs pivoting operations while factorizing the rows and columns of the matrix.

The rest of the paper is organized as follows. Section II introduces the theoretical background of LU decomposition, Section III describes the architectural design with proposed algorithm, Section IV proves the simulation of the design using Xilinx Vivado Design suite with verification of MATLAB results, and Section V provides FPGA mapping of the design and discussion of performance results. This paper concludes with a brief conclusion in Section VI.

II. BACKGROUND

A. Sparse LU Decomposition

LU decomposition or factorization is a popular matrix decomposing method for many numerical analysis and engineering science problems. It decomposes the matrix as a product of the lower triangular matrix (**L**) whose diagonal elements are equal to 1 and all the elements above the diagonal are equal to 0, and an upper triangular matrix (**U**) whose elements below the diagonal are equal to 0. If **A** is a square matrix, LU decomposes **A** with proper row and/or column orderings into two factors, which is shown in Fig. 1.

$$\boldsymbol{A = LU} \tag{1}$$

$$\begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ L_{21} & 1 & 0 \\ L_{31} & L_{32} & 1 \end{pmatrix} \times \begin{pmatrix} U_{11} & U_{12} & U_{13} \\ 0 & U_{22} & U_{23} \\ 0 & 0 & U_{33} \end{pmatrix} \tag{2}$$

LU decomposition is a direct method that can solve large systems of linear equations that arise from important applications such as circuit simulation, power networks, and structural analysis [5]. To ensure stability during LU decomposition, pivoting operations are performed to remove zero elements from the diagonal of matrix **A**. Without proper pivoting, the decomposition may fail to materialize. A proper permutation in rows or columns is sufficient for LU decomposition, which is also known as Partial Pivoting. This approach is suitable for a square matrix, and it is numerically stable in practice.

$$\boldsymbol{PA = LU} \tag{3}$$

On the other hand, Full Pivoting involves both row and column permutations.

$$\boldsymbol{PAQ = LU} \tag{4}$$

where **Q** is a permutation matrix which reorders the columns of **A**.

The forward reduction and backward substitution techniques are more stable compared to matrix inverses to solve systems of linear equations because every nonsingular matrix possesses an LU decomposition. When compared with regular matrices, sparse matrices can benefit from algorithms that reduce the number of operations which are required to calculate **L** and **U**. However, the disadvantage is that sparse methods will suffer from irregular computation patterns as they are dependent on the nonzero structure of the matrix.

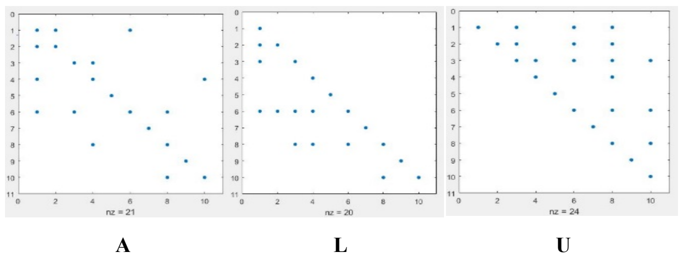


Figure 1. Example of a sparse matrix and its factors L and U

B. Related Work

There have been many architectures proposed for sparse LU decomposition which either target domain-specific sparsity patterns or require a pre-ordered symmetric matrix [6]. Blocking is a useful technique for gaining higher throughput for dense matrices. When decomposing in blocks using a Block Sparse Row (BSR) format for solving linear systems, it is limited to a matrix containing square blocks of a single dimension. When decomposition is executed in parallel, it often tries to avoid pivoting by using threshold pivoting or static pivoting beforehand. The architecture proposed in [7] implements a right looking algorithm and includes a hardware mechanism for pivoting. The performance of this is primarily I/O bandwidth limited.

Another implementation captures the static sparsity pattern and is exploited to distribute the data flow representation of computation for circuit simulation [8]. A more general hardware design is proposed parallelizing a left looking algorithm to support processing symmetric positive definite or diagonally dominant matrices. The factor limiting architecture efficiency is dynamically depending data dependencies. One more algorithm proposes choosing a pivoting strategy, where the matrix is decomposed block-wise. FPGAs have been shown to be effective in accelerating a wide range of matrix operations in recent years [9] [10].

The algorithm with row pivoting yields **LU=PA**, where the matrix overwrites **A** with **LU-I**, and **I** is an identity matrix. The first half of the algorithm will be triangular solving, leaving behind pivoting and scaling. In the case of sparse matrix, it will be inefficient for swapping rows. Due to having a single unreduced row or column, full pivoting is not easily achievable. The control system is implemented as a Finite State Machine (FSM), which tracks the progress of the units for synchronization. The algorithm for sparse matrix LU decomposition is in Fig. 2.

```
Algorithm
A → $n×n$ sparse matrix
P → $n×n$ identity matrix
[n, m] = size(A)
set reset high
$U = A$
$L = P = I_{n*n}$

[Perform pivoting operation]
function pivot (A, P, i)
        P = choose pivot ($A_i$: end, i)
        if ($P \neq k$) then
                SWAP ($A_i$, *, $A_p$, *)
                SWAP ($P_i$, *, $P_p$, *)
        end if
        return (A, P)
end function
[Interchanging rows in matrix]
If m≠ j
  U ([m, j], :) = U ([j, m], :)
  P ([m, j], :) = P ([j, m], :)
  If j<=2
    L ([m, j], 1: j-1) = L ([j, m], 1: j-1)
  end
end
[Update row and column entries]
for i = j+1 to n
  for j = 1 to n
    $L_{i,j} = U_{i,j} / U_{j,j}$
      for k = j+1 to n-1
            U (i, *) = U (i, *) - L (i, j) × U (j, *)
        end
  end
end
```

Figure 2. Pseudo code for Sparse LU Decomposition

III. Sparse LU Decomposition Architecture

The proposed approach for sparse LU decomposition consists of the following operations:

1. Pivoting strategy, when A has nonzero entries which are at fill-up locations.
2. Symbolic decomposition, which estimates the memory requirements for L and U factors.
3. Numerical calculation, which is computed using Gaussian elimination.

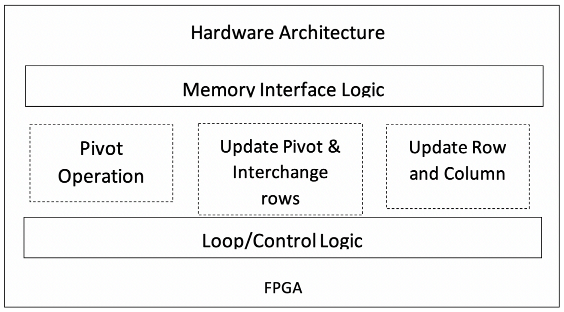


Figure 3. Proposed LU Decomposition Hardware Algorithm

To maximize performance, LU hardware is designed to focus on maintaining a regular computation and memory access pattern. Fig. 3 shows a block diagram of the proposed hardware algorithm. The control and memory access handle the operations performed for decomposing the matrix. The design ensures the memory will have enough space to store the values [11] [12].

A. Pivot Operation

In order to perform a pivoting operation, the design includes usage of lookup tables and memory pointers to keep track of memory mapping. It performs a pivot search for each step of matrix elimination. Index pointers are created for each pivot to store the row and column physical address, accordingly. These physical addresses are then used to fetch the values from memory. These values are sequentially checked as they arrive for the absolute maximum values with the index. Using a register, it is stored as a pivot element. The minimum amount of memory utilized is proportional to the size of the matrix. Once pivoting is complete, an update is sent back to the lookup tables.

B. Update Pivot and Interchange Rows

The "Update Pivot and Interchange Rows" logic block performs normalization prior to elimination for the pivot values of row and column requested from memory. The necessary data such as pivot index, values and column are inferred from the previous state. This process is executed one by one after each pivot value is fetched and read. The updated row and column values and the normalized row and column values are then stored in registers.

C. Update Row and Columns

The remaining computations required are performed during this transition state. First, it indicates if the given row or column should be updated. Second, it manages the addresses of nonzero entries that are to be stored. This unit contains the necessary floating-point multiplier and adder to perform the required arithmetic operations [13]. This unit is operational in parallel to maximize the utilization of all logic units. This will update the number of updated logics that fits in FPGA chip. There are enough resources available in the FPGA that can accommodate all of the units.

IV. Implementation and Verification

Various arbitrary matrices with different sparsity patterns are generated using MATLAB and are tested using the hardware architecture. A parameter n is included along with

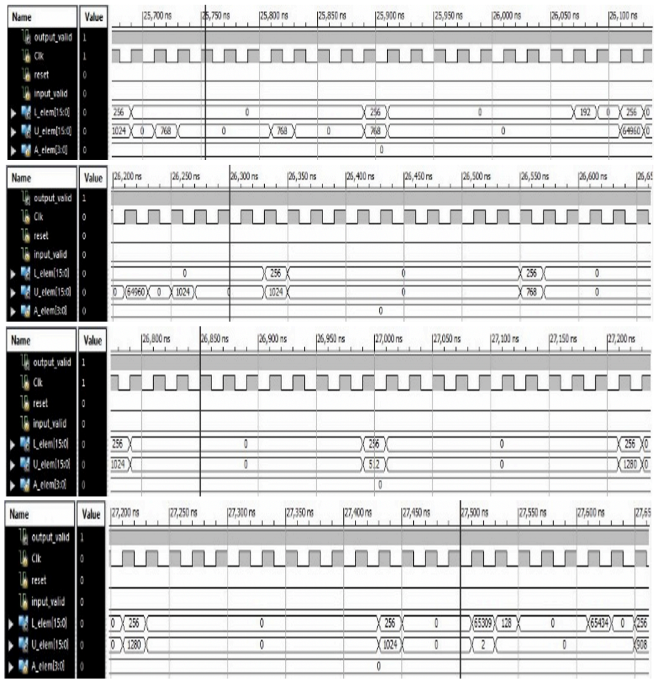


Figure 4. Simulation Waveform for LU Decomposition

The simulated results are stored in an external .txt file and are verified with the results from MATLAB for precision loss. For **L** matrix, the error ranges between -0.0872 to 0.0357 and for **U** matrix it ranges between -0.0108 to 0.0057 as shown in Fig. 5 below.

$$L_{diff} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.001 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.001 & 0.003 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 7.81e^{-04} & -0.001 & -0.002 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -2.60e^{-04} & -0.001 & 0.002 & 0.003 & 0 & 0 & 0 & 0 \\ 0 & 0 & -5.20e^{-04} & -0.001 & 9.44e^{-04} & 0.001 & -0.001 & 0 & 0 & 0 \\ 0 & 0 & -0.001 & -0.002 & 0.0036 & 3.24e^{-04} & -0.003 & 0.009 & 0 & 0 \\ 0 & 0 & 0 & -0.002 & 0.0059 & -0.004 & 0.003 & -0.01 & 0.006 & 0 \end{bmatrix}$$

$$U_{diff} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -0.007 & 0.001 & 0 & 0.004 & -5.20e^{-04} & -0.002 & 0.001 & 0.003 \\ 0 & 0 & -0.007 & -0.023 & -0.02 & -0.005 & -0.01 & -0.029 & -0.026 & 0.011 \\ 0 & 0 & 0.003 & -0.01 & -0.003 & -0.005 & -0.00 & -0.013 & -0.017 & -0.003 \\ 0 & 0 & 0 & 4.44e^{-016} & -0.01 & 0.01 & 0.04 & 0.017 & 0.014 & 0.012 \\ 0 & 0 & 0 & 0.01 & 0.023 & 0.01 & -0.003 & 0.008 & 0.013 & -0.009 \\ 0 & 0 & -0.007 & 0.02 & 0.003 & 0.019 & -0.02 & 0.011 & 0.027 & 5.95e^{-04} \\ 0 & 0 & 0.007 & 0.01 & -0.003 & -0.023 & 0.04 & -0.019 & -0.011 & -0.00 \end{bmatrix}$$

Figure 5. MATLAB Calculated Errors Values

V. PERFORMANCE ANALYSIS

A comparison of LU decomposition of sparse matrix of size 10x10 and 100x100, with a different sparsity range of 10% to 50% is shown in Fig. 6 below. The proposed LU decomposition design was able to achieve lower latency than the regular algorithm LU decomposition. The results are also verified with the MATLAB LU decomposition outputs for precision loss.

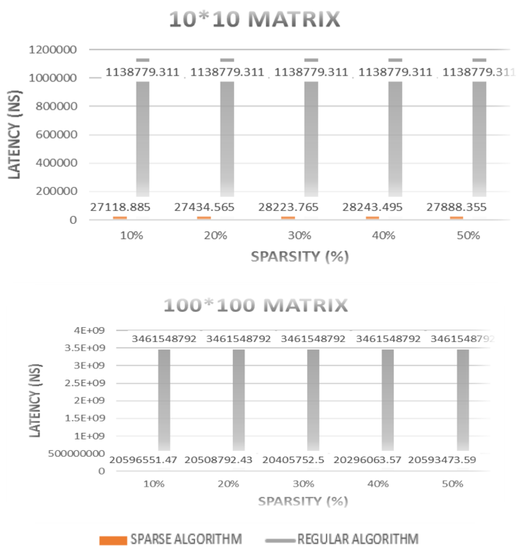


Figure 6. Latency Comparasion

A comparison of the throughput calculated from the sparse matrix algorithm and regular algorithm is plotted in the form of a graph and is represented in Fig. 7. As the throughput needs to be high for better performance, we are able to infer from the graph that high throughput was achieved.

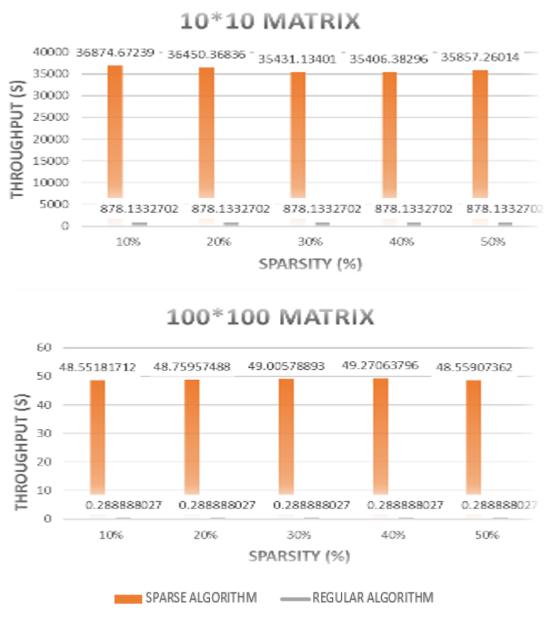


Figure 7. Block Diagram of the TX and RX Module

The data from Table I shows that the matrix storage format proposed in this research was able to achieve minimum resource utilization, as opposed to the traditional regular LU decomposition algorithm. The proposed design was implemented on a Xilinx Artix7 XC7A100T-1CG324C board comprising of 15,850 logic slices and a maximum of 4,860 Kbits fast block RAM. This is achieved with optimization through the implemented design for the LU decomposition. A difference in about one third of the total resources utilized was achieved, as seen in Fig. 8 and 9, respectively.

The performance of the design is based on the architecture and its parameters. As an FPGA has enough computational resources and the design is memory-bound, the performance is totally dependent on memory access time.

TABLE I. RESOURCES UTILIZED FOR PROPOSED ALGORITHM

Device Utilization Summary			
		Proposed Sparse Algorithm	**Regular Algorithm**
Slice Logic Utilization	**Available**	**Used**	
Slice Registers	126,800	3,420	10,863
Slice LUTs	63,400	11,211	16,807
Memory	19,000	8	64
Occupied Slices	15,850	3,504	5,455
IOBs	210	40	40

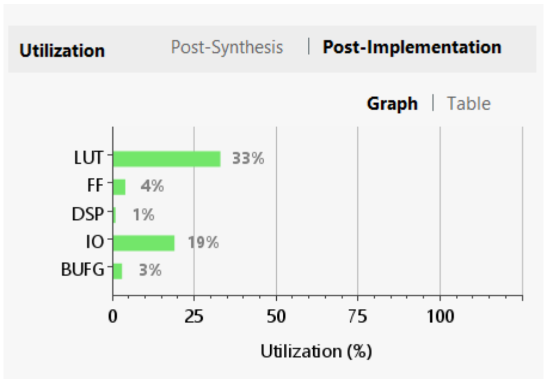
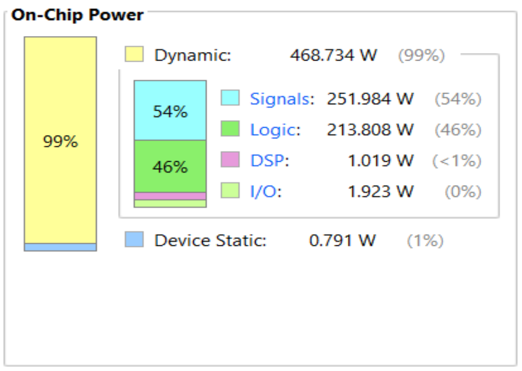


Figure 8. FPGA Design Utilization



On-Chip Power

🟨 Dynamic:	468.734 W	(99%)	
	🟦 Signals:	251.984 W	(54%)
	🟩 Logic:	213.808 W	(46%)
	🟪 DSP:	1.019 W	(<1%)
	🟩 I/O:	1.923 W	(0%)
🟦 Device Static:	0.791 W	(1%)	

Figure 9. Design Power Requirements

VI. CONCLUSION

Numerous engineering and machine learning applications rely largely on solving linear equations using LU decomposition, due to rapid developments in the field of mathematics and computation. Compared with a CPU and GPU, the FPGA does not have an instruction set. Instead, it possesses a number of reconfigurable logic blocks which could perform any digital logic function. In this paper, a computational implementation of the LU decomposition is proposed using an optimized algorithm. The proposed architecture can achieve further improvement by increasing the overall design clock.

REFERENCES

[1] M. Wielgosz, G. Mazur, M. Makowski, E. Jamro, P. Russek, and K. Wiatr "Analysis of the Basic Implementation Aspects of Hardware-Accelerated Density Functional Calculations," OJS Computing and Informatics, vol. 29, no. February, pp. 989–1000, 2010.

[2] A.Ching, S. Edunov, M. Kabiljo, D. Logothetis, and S. Muthukrishnan, "One Trillion Edges : Graph Processing at Facebook-Scale," , Proceedings of the VLDB Endownment, vol. 8, no. 12, pp. 1804-1815, 2015.

[3] A. Pinar and M. T. Heath, "Improving Performance of Sparse Matrix-Vector Multiplication," Proceedings of the 1999 ACM/IEEE Conference on Supercomputing, January 1999 Pages 30–39 doi:10.1145/331532.331562.

[4] T. Mattson et al., "Standards for graph algorithm primitives," IEEE High Performance Extreme Computing Conference (HPEC), Waltham, MA, USA, 2013, pp. 1-2, doi: 10.1109/HPEC.2013.6670338.

[5] S. Jain, N. Kumar, J. Singh, and M. Tiwari, "FPGA Implementation of Latency, Computational time Improvements in Matrix Multiplication," International Journal of Computer Applications, 2014, vol.86, no.8, doi:10.5120/15007-3261.

[6] S. Aslan and J. Saniie, "Matrix Operations Design Tool for FPGA and VLSI Systems," 2016, Circuits and Systems, vol. 7, no.2, pp. 43–50, doi: 10.4236/cs.2016.72005.

[7] P. Greisen, M. Runo, P. Guillet, S. Heinzle, A. Smolic, H. Kaeslin, and M. Gross, "Evaluation and FPGA Implementation of Sparse Linear Solvers for Video Processing Applications", in IEEE Transactions on Circuits and Systems for Video Technology, vol. 23, no. 8, pp. 1402-1407, Aug. 2013, doi: 10.1109/TCSVT.2013.2244797.

[8] W. Liu and B. Vinter, "An Efficient GPU General Sparse Matrix-Matrix Multiplication for Irregular Data", 2014 IEEE 28th International Parallel and Distributed Processing Symposium, Phoenix, AZ, USA, 2014, pp. 370-381, doi: 10.1109/IPDPS.2014.47.

[9] J. Johnson, T. Chagnon, P. Vachranukunkiet, P. Nagvajara, and C. Nwankpa, "Sparse LU Decomposition using FPGA", International Workshop on State-of-the-Art in Scientific and Parallel Computing (PARA), pp. 1-12, 2008.

[10] G. Wu, X. Xie, Y. Dou, J. Sun, D. Wu, Y. Li, and A. S. Matrix, "Parallelizing Sparse LU Decomposition on FPGAs", 2012 International Conference on Field-Programmable Technology, Seoul, Korea (South), 2012, pp. 352-359, doi: 10.1109/FPT.2012.6412160.

[11] L. Polok and P. Smrz, "Pivoting Strategy for Fast LU Decomposition of Sparse Block Matrices", HPC'17: Proceedings of the 25th High Performance Computing Symposium April 2017, no. 14, Pages 1–12.

[12] X. Wang and S. G. Ziavras, "Parallel LU Factorization of Sparse Matrices on FPGA-Based Configurable Computing Engines," Wiley Concurrency Computat.: Pract. Exper.,, vol. 16, no. April, pp. 319-343, 2004.

[13] Siddhartha and N. Kapre, "Breaking Sequential Dependencies in FPGA-Based Sparse LU Factorization," 2014 IEEE 22nd Annual International Symposium on Field-Programmable Custom Computing Machines, Boston, MA, USA, 2014, pp. 60-63, doi: 10.1109/FCCM.2014.26.

IoT Applications with Common Distributed Architecture for Data Acquisition

Kushal Thapa
Ingram School of Engineering
Texas State University
San Marcos, USA
email: k_t260@txstate.edu

Vinay Lokesh
Ingram School of Engineering
Texas State University
San Marcos, USA
email: v_v183@txstate.edu

Kevin Seets
Ingram School of Engineering
Texas State University
San Marcos, USA
email: kms489@txstate.edu

***Abstract*—Internet of Things (IoT) applications have many forms, and leverage many different technologies. However, certain classes of applications have extremely strong similarities in system architecture. This paper discusses several important applications, which leverage a small set of loosely coupled, distributed data acquisition subsystems to effect a centrally coordinated, data intensive function or application. For brevity, we call this structure "Coordinated IoT for Data Acquisition" (CIDAQ). Finally, this paper introduces a novel power line communication research, which employs this CIDAQ architecture for data capture and processing.**

***Keywords*—Internet of Things; IoT; Smart Grid; Active Shooter; Machine Learning; ML; Artificial Intelligence; AI.**

I. Introduction

In some scenarios where Internet of Things (IoT) technologies are used, the application of interest could be useful for training first responders, as it provides a more analytical approach to how first repsonders move and react. In other cases, IoT's use relates more to optimization of an industrial process, where a net of data could increase functionality or longevity. It could also have use in observations of biological processes where data has proven difficult to gather by more traditional means. In all cases, the application benefits from the Coordinated IoT for Data Acquisition (CIDAQ) architecture, where a distributed, loosely-coupled set of IoT devices provides telemetry data to a central repository, and Machine Learning and Artificial Intelligence (ML/AI) algorithms are employed to produce some application-related insights.

In this paper we describe various applications which leverage the CIDAQ architecture as well as some useful technologies. Section II presents a compelling application related to training of first responders in active shooter scenarios. Section III briefly describes several other applications, and presents some of the ML/AI techniques that can be useful in these applications. Section III-D exhibits some businesses and products that are already available for purchase, ranging in size and scope.

We conclude the paper in Section IV with a particularly interesting application of the CIDAQ architecture, which focuses on the electrical grid. In this application, the distributed system "listens" to current disturbances on the electrical distribution grid, "talks" upstream from the outlet to the substation, and "geolocates" electrical devices for system management purposes based on actively and passively gathered telemetry data.

II. Active Shooter Training

A particularly compelling application of IoT systems is to augment the training of first responders. This application leverages the CIDAQ architecture by placing data acquisition devices and sensors on various body parts of first responders who are participating in scenario-based, real-time active shooter training.

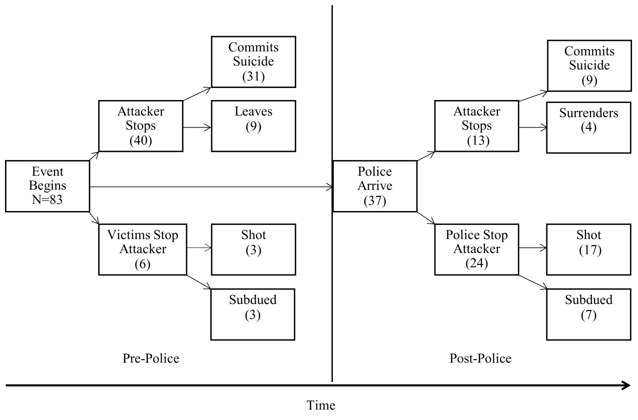


Fig. 1. Comparison of active shooter situations with and without first responder intervention [1].

Having properly trained first responders immediately available for active shooter situations is paramount to ensuring the safety and survival of bystanders. Fig. 1 indicates an increase in victim shootings and death associated with police intervention. In a comparison study of 83 events with and without police intervention, 37 (45%) included police intervention, which accounted for over half of total victim shootings (63%) and deaths (56%) [1]. Though this is a particularly alarming trend, it does give insight on the fact that police officers need to be better trained to deal with active shooter situations.

Traditional training for an active shooter situation can be time consuming and expensive, and so is often not effective in producing measurable outcomes [2]. The prevalence of active shooter situations in the United States reveals a necessity for officer training that is effective in measurable outcomes as well as conventional metrics of time and cost [3]. Students at Texas State University have designed a system with the potential to improve training for first responders which could play a part in revolutionizing first responder training, as shown in Fig. 2. This system leverages the CIDAQ architecture. By placing sensors strategically on participant, data about proper movement and weapon handling can be gathered in real-time from multiple participants, processed, and analyzed in a central location, and leveraged to improve the effectiveness of first-responder training. This information can in turn be used to create augmented reality training programs that could be effective tools in saving law enforcement valuable time and money, and in improving the ability to repeat training remotely [2]. Additionally, the data can be used to precisely compare

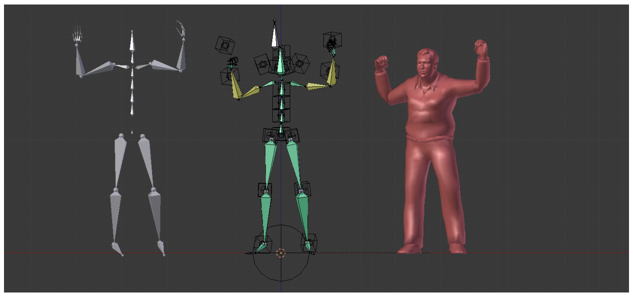


Fig. 2. Augmented motion-tracking of first responders using CIDAQ architecture sensors [4].

and contrast the effectiveness of different training programs, thereby improving the effectiveness of the training program as well as the measurable outcomes for each participant.

The devices used in this application consist of Inertial Measurement Units (IMU) placed on the head, chest, and weapon of the participants. Each IMU contains analog orientation, acceleration, and location sensors whose outputs can be easily acquired, stored, measured, and manipulated. The data collected by these devices creates a 3-dimensional map of the user's movements which can be reconstructed and replayed at-will. Augmented reality devices can create virtual training experiences with these maps that include all of the necessary movements and actions required by a first responder [3]. This training method not only improves time and cost effectiveness, but also affords a new level of access. The ease of distribution of these expert training programs increases accessibility for smaller or more distant municipalities and also helps streamline training. This ensures first responders have access to the exact same training, which will help with joint operations as well as transfers between departments.

The application of CIDAQ architecture for training first responders does not stop at active shooter situations; there are a number of other projects being developed and at least one that is already in use. Dartmouth College's Interactive Media Laboratory and Institute for Security Technology Studies created a virtual program designed to aid in training for terrorism response. The program, called Ops-Plus, utilizes 3D simulators to aid in training against attacks that involve nuclear, radiological, biological, and even chemical warfare. The Los Angeles Police Department currently uses an immersive simulation trainer, called HYDRA, used to train first responders in a series of scenarios, from earthquakes to terrorism response, that would be difficult to recreate. New York City has partnered with the Environment Tectonics Corporation to develop a software, similar to HYDRA, that creates an immersive environment designed to help first responders prepare for citywide disaster management [5]. As these technologies progress, the breadth and depth of IoT application will increase, reducing potential harm to first responders and civilians.

III. Similar Applications and ML/AI

In addition to applications such as training for first responders, the CIDAQ architecture is being actively deployed in various other applications including monitoring and control of industrial processes, monitoring of the habitat for endangered species, and enabling efficient hospital care for bedridden patients. These applications are described briefly along with the general nature of ML/AI algorithms which could be used in processing the resulting telemetry data to create new knowledge, or to improve application-specific outcomes.

A. *Industrial Application*

Heat trace cables or heat tapes are vital in oil and gas, chemical treatment, power generation and many other industries. These cables and their control systems assist in the continuous delivery of gases and liquids, often preventing the contents of pipes or tanks from freezing in extreme environments.

Based on advances in heat sensitive polymer design, many of these cables come with self-regulating ability. In other words, the heat generated by these cables can compensate for environmental temperatures by autonomously adapting their absorption of electrical current [6]. This capability provides operational simplicity in external power control systems, as well as convenience in direct attachment to the electrical power source.

However, the heat cable can be damaged during deployment, or can degrade due to aging or other conditions. Approaches to locating damaged portions of heat cable is a difficult challenge. One approach to this remote monitoring problem is to integrate temperature sensors into the cable, or add IoT-based devices along the cable to make measurements. As a distributed, network-based monitoring architecture, a CIDAQ system is a logical candidate for this application. Via a CIDAQ architecture, deployed heat cables and the systems they monitor can be remotely evaluated using low-rate data transmitted along the heat cable and power lines. This data, transmitted directly via low-frequency power line communications techniques, can also be aggregated, assimilated, and analyzed using the ML/AI algorithms.

B. *Biodiversity Application*

The conservation of endangered species is important for maintaining biodiversity and a well-balanced ecosystem. Several techniques have only recently been applied in the marine environment to detect the presence of marine species [7]. Confirming presence relies on locating the animals, which can prove challenging for species with low population numbers. A variety of methods have been used to determine the presence of rare marine species, including fishing and underwater visual surveys [8]. However, these approaches typically require substantial field-based effort by researchers and data gatherers. Although scientists have been able to achieve a significant amount of success using Environmental DNA (eDNA), not every organism will be readily detected by eDNA and the scale of the water bodies impacts the probability of detection.

Importantly, the abiotic factors of temperature, UV radiation, and amount of DNA present all impact the length of time that the eDNA stays in the environment [9]. A compelling approach which uses the CIDAQ architecture to monitor endangered species is detects and processes animal voice or audio signals. Examples of such endangered species that are being monitored with a CIDAQ architecture are the Houston Toad and Craw Frog [10]. In these applications, an embedded solution detects toad calls automatically with real-time notification transmission capabilities to engage remote researchers. The labelled audio data is filtered and fed to a machine learning model to extract features. The extracted features are then fed to classification algorithms using the processing pipeline shown in Fig. 3.

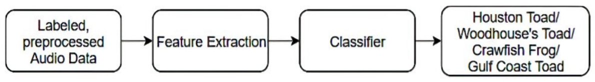

Labeled, preprocessed Audio Data
Feature Extraction
Classifier
Houston Toad/ Woodhouse's Toad/ Crawfish Frog/ Gulf Coast Toad

Fig. 3. Experimental Method.

This application of the CIDAQ architecture leverages deep learning architectures, such as Recurrent Neural Network (RNN), Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), Gated Recurrent Units (GRUs) as well as conventional signal processing such as Mel-frequency Cepstral Coefficients (MFCC), Linear predictive coding (LPC), Perceptual Linear Prediction (PLP), Mel Filter banks, and Spectrograms [11]. These technologies are used to improve the identification of endangered species with reduced false-positive rate [10].

C. Medical Application

Another intuitive application that uses CIDAQ architecture enables efficient hospital care for bedridden patients. The feasibility of using pervasive sensing technology and artificial intelligence for autonomous and granular monitoring in the Intensive Care Unit (ICU) is vital since manual observations can suffer from subjectivity. The use of sensing technologies and network-based telemetry can bring timely intervention to assist in making life-saving decisions while dealing with high levels of uncertainty under strict time constraints [12]. Artificial intelligence in the critical care unit could reduce doctors' workload to allow them to spend time on more critical tasks. The approach used in this application include accelerometer sensors, a light sensor, a sound sensor, and a high-resolution camera to capture data on patients and their environment in the ICU. Various computer vision and deep learning techniques are used to recognize a patient's face, posture, facial expressions, head pose, and extremity movements from video data [13]. For activity recognition, data from wearable accelerometer sensors worn on the wrist, ankle, and arm are analyzed. Additionally, the information uses the room's sound pressure and light intensity levels to examine their effect on patients' sleep quality. This framework employs a cascaded architecture with three stages of deep CNNs to predict face and landmark locations in a coarse-to-fine manner [12].

In general, most embedded applications dealing with IoT, machine learning and artificial intelligence implement a CIDAQ architecture. Many of these applications leverage deep learning algorithms, which are concerned with very large datasets of labelled analog data, such as image, text, audio and video [14]. Machine learning algorithms used in CIDAQ-based systems can be based on supervised, unsupervised or semi-supervised learning. Supervised learning model is based on training data and helps to make predictions, some of the important algorithms under supervised learning are logistic regression and back propagation neural networks. Unsupervised learning model is prepared by deducing structures present in the input data, some of the important algorithms under unsupervised learning are Apriori and K-means algorithm. Semi-supervised learning is a mixture of both labelled and unlabeled data.

D. Market Data

As IoT technologies gain in popularity and scope of capability they will become more available and more widely used. There are a number of businesses–ranging from startups to Fortune 50 companies–using IoT, and implementing CIDAQ architecture in a range of ways, for their products and services. A few smaller startup companies are: Vicotee, MachineMax, and Radio Bridge [15]. Vicotee is a Norwegian company that offers a variety of sensors that can be used in conjunction with each other for myriad 'smart' applications including but not limited to shipping, infrastructure, healthcare, air quality, and land management [16]. MachineMax, based out of the UK, on the other hand offers a software, rather than only offering a full-package, that uses IoT to interconnect 3rd party sensors into a a single platform [17]. Similarly, US based Balena offers IoT 'fleet management' that is designed to push updates across varying platforms using a cloud-based container [18].Instead of simply offering a product line, US based Radio Bridge offers IoT data-as-a-service where they set up devices and monitor the data, allowing the end user to focus on whatever task is at hand [19]. Small startups are not the only businesses interested in IoT technologies, there are also several major players competing in the market as well, specifically IBM, Samsung, AWS, and Microsoft [20]. Each of these Global 500 businesses offers a proprietary monitoring solution for their products.

IV. Electric Distribution Grid

As mentioned in Section I, the electric distribution grid is a particularly compelling application of the CIDAQ architecture. As more compute and sensor devices pervade modern society, reliance on the electrical infrastructure continues to increase. Although "the grid" is one of the unspoken wonders of the modern technological society, the increased burdens of two-way power flow, distributed generation facilities, and complex/dynamic structure are pressing technologists to create better approaches to monitoring, controlling, and leveraging existing grid infrastructure.

An intuitive approach to leverage existing grid infrastructure is to use the grid itself as a communication medium. This would pave a way for the development of a self-regulating power grid, and in addition, obviate the need for deploying other communication resources for grid related applications, thereby saving time and money worth billions [21]. This distributed system concept fits precisely within the CIDAQ architecture.

Using the power grid as a communication medium is not a novel concept. This technology, commonly referred to as Power Line Communication (PLC), has been used since early 1920's for applications like fault detection and automatic meter reading [22]. However, the development in the PLC applications has been severely hindered due to the dynamic and unpredictably noisy nature of this medium. This problem is aggravated by the different electrical devices in the power grid, like transformers, which obstruct and muddle the communication signals even more [23].

One simple approach to solving this problem is the transmission of low-frequency communication signals. Low- frequency signals don't attenuate or distort as much compared to high-frequency signals, even after passing through the transformers [24]. Thus, a low-frequency band, typically in the range of 150 Hz-1350 Hz [24], can be used directly for PLC applications. However, one major disadvantage of using low- frequency bands for communication is the low data rate. Consequently, this solution has been under-researched and

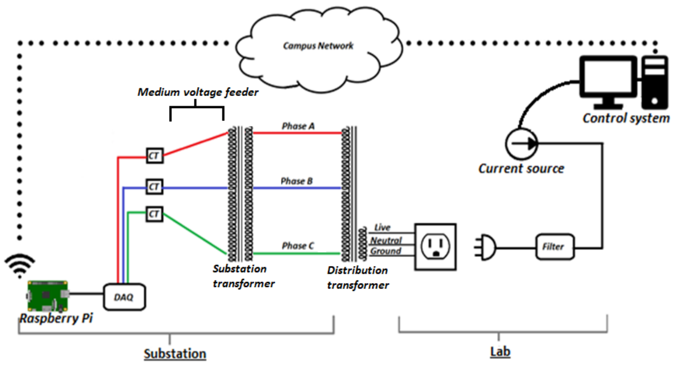


Fig. 4. Experimental setup for sending and capturing communication signals through powerlines.

mostly overlooked.

Nonetheless, such low-frequency PLC has applications in fields that don't require high-speed data transmission but prioritize reliability, simplicity, scalability, and ease of deployment. There is an urgent need for this type of technology in the power sector. As mentioned before, the existing power grids are failing because of the exponential increase in power demand over the last few decades [25]. This problem is exacerbated by the disconnect between the power producers and consumers which leads to a huge waste of already depleted power supply. Low-frequency PLC can bridge this disconnect thereby becoming the backbone communication infrastructure of a continuously sensing and self-monitoring power grid called "smart grid" [26].

Therefore, there is a need for research and study in the field of low-frequency PLC. To that end, our research is going to be focused on employing a CIDAQ network architecture to test low-frequency power line communication for simple digital communication. Fig. 4 shows a simplified design of this network architecture.

As shown in Fig. 4, a programmable current source injects a known signal into the powerline via a stabilizing filter. This signal passes through single-phase and three-phase distribution grid via transformers. When the signal passes through a transformer, the image of the signals are ingrained in the other two-phases of the power line, creating noisy image or echo signals [23]. In the electric substation, the injected signals, which have passed through multiple transformations, are collected by a Data Acquisition Device (DAQ) [27]. The DAQ is controlled by an embedded system such as a Raspberry Pi [28]. The control system and remote DAQ devices are connected to a common wireless network for synchronization. The control system commands the programmable current source to inject communication or disturbance signals, and simultaneously commands the DAQ to acquire signal data. This data is collected centrally for signal processing and machine learning techniques to analyze and reconstruct the original signal, remove extraneous noise or images of signals, and create a global understanding of the signal context on the distribution grid.

The raw data captured at the substation contains the input communication signal mixed with a more dominant power signal and its highly correlated harmonics, plus a highly dynamic and unpredictable environmental noise. Using the CIDAQ architecture, the distributed system extracts the information sent by the input signal from this mixture. Traditional

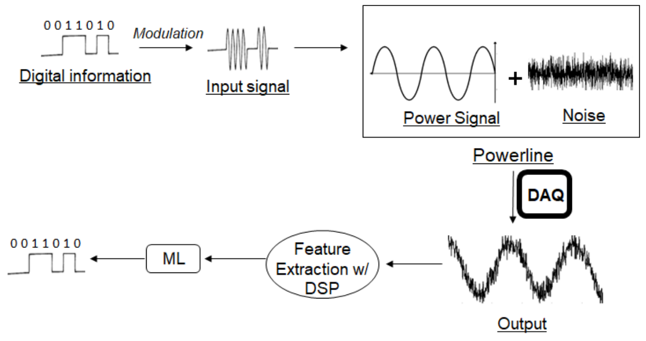


Fig. 5. Flowchart showing the basic signal workflow from input to machine learning output.

Digital Signal Processing (DSP) techniques [29] do this job in other communication media like telephone wires and optical fibers, but are not adequate for highly noisy PLC. As a result, supplementing DSP techniques with ML becomes a critical aspect of this CIDAQ implementation. Machine learning is a data-driven technique that formulates a relationship between the input and the output based on the training data [30]. In the case of the ultra-low-frequency powerline environment, the training data is composed of various features of the raw signal extracted using DSP techniques. This data is fed into ML algorithms to form a model which extracts the transmitted information from the raw noisy data. The overall signal flow from transmitted input to the extracted ML output is shown in Fig. 5.

This way, the CIDAQ architecture effectively captures PLC data from the electrical grid and deciphers the information contained in the data. This CIDAQ architecture can also easily be scaled to employ multiple DAQs, which can be placed at differing locations in the power grid. A similar architecture can also be used to capture non-PLC internal power grid data, which can contain information about the state of the grid and its components. This non-PLC data can similarly be processed and deciphered using signal processing and ML techniques via the same CIDAQ architecture.

V. Conclusion

Using the context of several related applications, this paper has introduced the CIDAQ architecture, which leverages distributed IoT-like devices to acquire relatively high-rate signals and analyze them collectively to produce some application-specific outcome. In cases of first responder training, IoT devices are distributed on a participant's body for analysis of form and function in a high intensity situation. In cases of industrial control and management, IoT devices are distributed on heat-tape which ensures the consistent flow of gas or liquid in an industrial environment. In cases of species management or healthcare management, IoT devices gather data from the environment or from healthcare facilities for processing and analysis, and to create a larger context from which to prioritize societal or personal decisions.

In the electrical grid, which underpins almost all related applications, sensors, and data acquisition systems are distributed throughout a larger, dynamic context in order to acquire signals, gather data, cross-correlate events, and effect changes in system efficiency that will enable future applications which could benefit from the CIDAQ architecture.

ACKNOWLEDGEMENT

The authors would like to thank Bo Hyse, Matt Healea, and Karina Paz for the use of their senior design project as an example in this paper, as well as the Advanced Law Enforcement Rapid Response Training (ALERRT) team at Texas State University for the use of their research involving active shooter training.

REFERENCES

[1] M. H. Martaindale, W. L. Sandel, and J. P. Blair, "Active-shooter events in the workplace: Findings and policy implications," *J. Bus. Contin. Emer. Plan.*, vol. 11, no. 1, p. 6–20, 2017.

[2] P. Glass *et al.*, "Use of computer simulation modeling to reduce the consequences of an active shooter event in a large event venue," in *2018 IEEE International Symposium on Technologies for Homeland Security (HST)*, 2018, pp. 1062–1065.

[3] B. Heyse, M. Healea, and K. Paz, "EE Senior Design project Statement of Work: 3-D Motion Capture and Virtualization for Response Training," Ingram School of Engineering, Texas State Univ., San Marcos, Texas, Senior Design Thesis, 2019.

[4] Miecz. Blender motion capture retargeting tip. 25-March-2021. [Online]. Available: https://www.blendernation.com/2018/08/29/blender-motion-capture- retargeting-tip/

[5] G. Meyers. Simulation software is cost-effective for emergency response training. FyorgeFX Training Simulations. 25-March-2021. [Online]. Available: https://forgefx.com/simulation-software-is-cost-effective-for-emergency-response-training/

[6] USX^{TM} Self-Regulating Heating Cable. Thermon. 25-March-2021. [Online]. Available: http://www.thermon.com/us/products/electric-heating/heat-trace/usx#overview

[7] K. Weltz *et al.*, "Application of environmental DNA to detect an endangered marine skate species in the wild," *PLoS ONE*, vol. 12, pp. 1–16, June 2017.

[8] G. F. Ficetola *et al.*, "Species detection using environmental DNA from water samples," *Biology letters*, vol. 4, pp. 423–5, Sep. 2008.

[9] H. C. Rees *et al.*, "The detection of aquatic animal species using environmental DNA – a review of eDNA as a survey tool in ecology," *Journal of Applied Ecology*, vol. 51, pp. 1450–1459, 2014. [Online]. Available: http://eprints.nottingham.ac.uk/id/eprint/30254

[10] S. Islam and D. Valles, "Houston Toad and other chorusing amphibian species call detection using deep learning architectures," in *Proc. 10th IEEE Computing and Comm. Workshop and Conf.*, Jan. 2020, pp. 0511–0516.

[11] L. Rabiner and R. Schafer, *Theory and Applications of Digital Speech Processing*. Pearson, 2011.

[12] A. Davoudi *et al.*, "Intelligent ICU for Autonomous Patient Monitoring Using Pervasive Sensing and Deep Learning," *Scientific Reports*, vol. 9, pp. 1–13, May 2019.

[13] N. A. Halpern and S. M. Pastores, "Critical care medicine in the united states 2000-2005: an analysis of bed numbers, occupancy rates, payer mix, and costs," *Critical care medicine*, vol. 38, no. 1, p. 65—71, Jan. 2010. [Online]. Available: https://doi.org/10.1097/CCM.0b013e3181b090d0

[14] Deep Learning. Machine Learning. 25-March-2021. [Online]. Available: https://machinelearningmastery.com/what-is-deep-learning/

[15] 5 top wireless sensor solutions impacting the telecom sector. 30-March-2021. [Online]. Available: https://www.startus-insights.com/innovators-guide/5-top-wireless-sensor-solutions-impac ting-the-telecom-sector/

[16] How vicotee's technology can help you move fast forward. 30-March-2021. [Online]. Available: https://https://www.vicotee.com/use-cases/

[17] We're building the new age of construction. 30-March-2021. [Online]. Available: https://machinemax.com/pages/solutions

[18] What is balena? 08-April-2021. [Online]. Available: https://www.balena.io/what-is-balena

[19] Iot data-as-a-service. 30-March-2021. [Online]. Available: https://radiobridge.com/services/iot-data-as-a-service

[20] J. Guth *et al.*, *A Detailed Analysis of IoT Platform Architectures: Concepts, Similarities, and Differences*. Springer, 2018, pp. 81–101.

[21] K. Rabie *et al.*, "IEEE Access Special Section Editorial: Advances in Power Line Communication and its Applications," *IEEE Access*, vol. 7, pp. 133 371–133 374, Sep. 2019.

[22] K. Dostert, "Telecommunications over the power distribution grid - possibilities and limitations," in *Proc. Int. Symp Power Line Comm. and Appl.*, 1997, pp. 1–9.

[23] S. U. Ercan *et al.*, "Power line communication design and implementation over distribution transformers," in *2017 10th International Conference on Electrical and Electronics Engineering (ELECO)*, 2017, pp. 190–194.

[24] D. Dzung, I. Berganza, and A. Sendin, "Evolution of powerline communications for smart distribution: From ripple control to OFDM," in *2011 IEEE International Symposium on Power Line Communications and Its Applications*, 2011, pp. 474–478.

[25] S. McClellan, D. Valles, and G. Koutitas, "Dynamic voltage optimization based on in-band sensors and machine learning," *Appl. Sci.*, vol. 9, no. 14, p. 2902, July 2019.

[26] N. Uribe-Pérez *et al.*, "Smart grid applications for a practical implementation of IP over narrowband power line communications," *Energies*, vol. 10, no. 11, p. 1782, Nov. 2017.

[27] DI-1100 4-channel USB Data Acquisition Starter Kit. DATAQ Instruments. 25-March-2021. [Online]. Available: https://www.dataq.com/products/di-1100/

[28] Raspberry Pi 4 Model B. Raspberry Pi. 25-March-2021. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/

[29] D. Kakati and S. C. Arya, "A full-duplex optical fiber/wireless coherent communication system with digital signal processing at the receiver," *Optik*, vol. 171, pp. 190 – 199, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S003040261830799X

[30] S. Raschka and V. Mirjalili, *Python Machine Learning*, 2nd ed. Packt Publishing, 2017.

Supervised Machine Learning in Digital Power Line Communications

Kushal Thapa[*], Stan McClellan[†], Damian Valles[§]
Ingram School of Engineering
Texas State University
San Marcos, TX, USA
email: [*]k_t260@txstate.edu, [†]stan.mcclellan@txstate.edu, [§]dvalles@txstate.edu

***Abstract*—Power Line Communications (PLC) is a technology that uses power lines to transport communication data alongside the AC electric signals. Due to the highly penetrative pre-existing power grid infrastructure, PLC has a huge networking potential, especially in the implementation of smart grid technologies. However, PLC medium poses a major hindrance in the form of poor signal propagation. Traditional signal processing measures are not enough to demodulate these poor signals at the receiver end. To overcome this challenge, we are investigating Machine Learning (ML) as a supplement to the traditional digital signal processing techniques in this project. Our project focuses on testing and comparing various supervised machine learning and deep learning algorithms for the purpose of digital PLC bit classification.**

Keywords-power line communications; PLC; machine learning; ML; smart grid.

I. INTRODUCTION

The use of electrical wiring and power lines for network communication is not new. Since the early 1920s, this technology has been used to automate meter reading by utility companies [1]. Beyond this application, the potential of Power Line Communications (PLC) was conceptualized as a universal networking solution mainly because of the pre-existing power-grid [2]. This power grid would obviate the need for building other types of dedicated communication infrastructures like phone lines and optical fibers, thereby saving billions in cost [2]. However, over the years, such high expectations of this technology have not been realized due to many factors. One of the primary culprits is signal propagation.

The power grid infrastructures, including the power cables, were not designed for communication purposes. Thus, communication signals face various hindrances in this medium, including highly variant and dynamic noise, radiation leakage, undesired modulation, etc., [3]. All of these problems aggregate to cause poor propagation of the signal. One approach to solving this problem is to devise ways to cancel out these causes and maintain a better quality of signal throughout its communication path. A different approach would be to design a better, more sensitive receiver that could extract information even from the poorly propagated communication signals. The latter approach has an advantage because only the receiver needs modification, while the former might need engineering improvements in the transmitter and the medium.

Traditional communication receivers work primarily by implementing Digital Signal Processing (DSP) techniques, such as demodulation, filtering, digitization, etc., [4]. However, these methods alone are not sensitive enough to extract the information signals in PLC. Machine Learning (ML), which is a technique that probes data for information, might be a good supplement to traditional signal processing in creating more sensitive receivers for PLC. Therefore, in our study, we have designed a PLC network architecture and used ML with signal processing features to extract the transmitted information from the raw PLC signal captured at the receiver.

The signal workflow of our project is shown in Figure 1. Digital information is modulated onto an analog carrier at the transmitter in one of several well-known approaches. This analog signal is injected into the power line where it combines with the dominant power signal plus highly variant noise. The output is collected with a Data Acquisition Device (DAQ) at the receiver, and it consists of a raw signal that resembles a power signal. Various features are extracted from this raw signal using DSP. These features, along with the corresponding digital labels, are arranged into a dataset. This dataset is then fed into ML algorithms, which creates a model. Lastly, we use this ML model to classify and thus, extract the transmitted digital information.

The rest of the paper is organized as follows. In Section 2, we provide a description of the data-capture methodology, feature extraction process used in the raw-data, and the setup of the ML models. In Section 3, we present the outcomes of ML model optimization, performance of these models, and validation of the results. Finally, we summarize the paper and provide conclusions in Section 4.

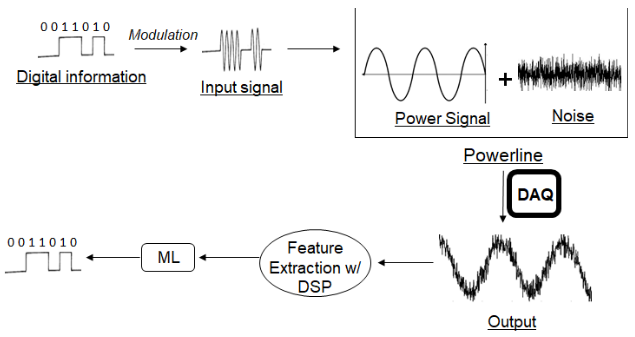


Figure 1. The flow of signals from digital input to ML output.

II. Procedure

A. Data Capture

The experimental setup used to capture the PLC data is shown in Figure 2. As shown in the figure, we used a current source to inject a low amplitude signal with a frequency of 1595Hz into the power line via a current-modulator and a stabilizing filter. This signal first passes into a single-phase lab wiring, then into a three-phase distribution power grid via transformers. Some signature of the signal gets ingrained on all three-phases during this transition [3]. In the substation, the power signal and the injected signal go through more transformation, primarily due to Current Transformers (CT). These transformed signals were then collected at the substation using DAQ. The communication signal originated at the low voltage region (the lab) and traveled towards the high voltage region (substation) of the distribution grid, thereby making the PLC path upstream.

B. Raw Data

The raw data, captured using DAQ, was a three-phase time-series data consisting of a power signal at around 60Hz, communication signal at around 1595Hz, and time-variant noise at all frequencies. The power signal dominated the time-domain plot of this raw data because of its relatively high amplitude. Thus, the time-domain plot did not show any trace of our communication signal. The power signal and its harmonics also dominated the frequency-domain spectrum plot. However, a small peak was present at 1595Hz that showed the presence of our transmitted signal. However, the spectrum plot cannot show the time-varying nature of the signal, and thus, did not provide us information about the digital data that was transmitted. A spectrogram, which is a plot of signal energies in a time vs. frequency graph, helps acquire this information. Figure 3 shows the spectrogram of the Phase A raw data. We can see the dominant power signal and its harmonics at low frequencies. More significantly, there is a clear dotted band above 1500Hz, which is our communication signal. In this frequency band, the short bright dashes represent the 1s, and the gap between these dashes represents the 0s. These discrete amplitude (energy) shifts correspond to the data (bits) modulated and transmitted by the current source.

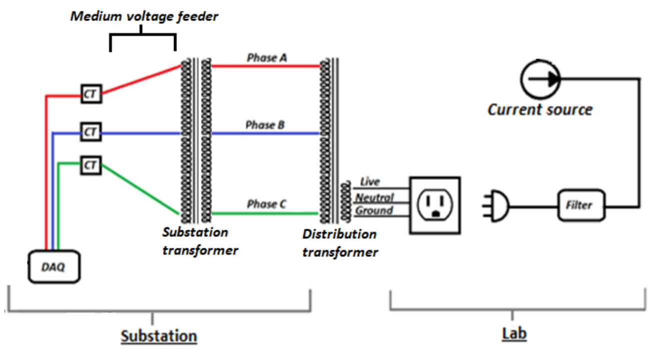


Figure 2. Experimental setup for sending and receiving a current signal through power lines in a distribution power grid.

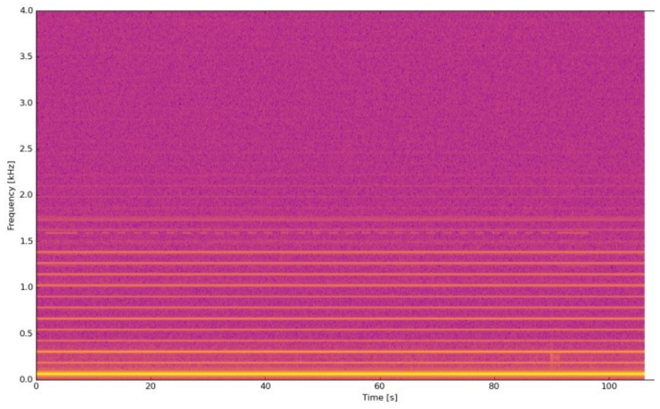


Figure 3. Spectrogram of Phase A of captured raw data.

C. Feature Extraction

After the raw data was collected, this data needed to be converted to ML-ready format. First, we divided the raw data into numerous frames corresponding to the resolution frame of the labels (sample length of a single bit in the transmitted analog signal). Each of these frames would be a sample row in our final dataset. Then, from each of these frames, we extracted various features as described below:

1) Amplitude Envelope

This feature gives the change in the amplitude of the signal over time [5]. It effectively traces the outline of the signal in the time-domain. In our case, the raw signal's amplitude envelope, as is, would not provide any meaningful information as the 60Hz power signal dominates all other superimposed sinusoidal signals. Therefore, we filtered the raw frames with band pass filters of 100Hz bandwidth starting from 1Hz and up to 2000Hz with no overlap (1Hz-100Hz, 101Hz-200Hz,...,1901-2000Hz). Hence, we divided each frame into twenty frequency-separated signals and calculated each of these signals' amplitude envelope. Our expectation was that the amplitude envelope of one of these signals which contains our communication frequency (1595Hz in our case) would provide information about the bit that was transmitted in that frame.

2) RMS (Root Mean Square) Energy

The energy of a signal is the measure of the "strength" of the signal. A signal's energy is defined as the sum of the square of its magnitude [6]. Thus, RMS Energy (RMSE) is the square root of the mean energy of a signal. Equation (1) [7] shows the formula for RMSE where x_i is the ith sample of signal x and N is the total number of samples.

$$RMSE = \sqrt{\frac{(x_1^2 + x_2^2 + \cdots + x_N^2)}{N}} \quad (1)$$

In our case, the raw signal's energy (or each frame) would again be dominated by the power signal. Hence, we frequency separated the frames as before and calculated RMSE for each of the twenty bandpass filtered signals of each frame. Like the amplitude envelope, we were expecting variations in the RMSE of 1,501-1,600Hz signals of different frames corresponding to the bit these frames were carrying.

3) Spectral Centroid

Amplitude envelope and RMSE are time-domain features, and thus, they were extracted from the time-series data. We decided to use spectral centroid to probe the frequency-domain of the raw data for important signal characteristics. The spectral centroid compares the center of mass of the signal's spectrum [8]. Our raw signal's spectrum had a primary peak at around 60Hz and secondary harmonic peaks at multiples of 60Hz because of the dominant power signal. Whenever the communication signal was present in the raw signal, there should also be a peak at 1595Hz (our communication frequency). We assumed that the presence and absence of the communication signal (corresponding to 1 and 0, respectively) would noticeably shift the center of the spectrum's mass, thereby providing a classification measure of the transmitted bit. Therefore, we included the spectral centroid of each frame as one of the features.

D. Machine Learning

After the dataset was formed by compiling the features from the raw data and labels were recorded, it was used in machine learning models with a 70% training split. To form the models with various supervised algorithms, Python Sci-kit learn used for Logistic Regression (LR) [9], Support Vector Machines (SVM) [10], and Decision Tree (TREE) [11]. The hyperparameters for these algorithms were optimized using the grid search [12] method. A majority voting model [13] was also created from the optimized LR, SVM, and TREE to check if such ensemble model would outperform the individual models. ROC AUC scores [14], precision [15], recall [16], and f1 scores [17] were computed to evaluate and compare these various models, training, and testing accuracy scores. Learning curves [18] were plotted and evaluated to ensure the models were not overfitting or underfitting. Confusion matrices [19] were also plotted to visualize the accurate label versus the predicted label.

Besides these basic "one neuron" ML models, multi-neuron, multilayer Artificial Neural Network /Deep Neural Network (ANN/DNN) model [20] was also tested using python's Tensor Flow and Keras. The various hyperparameters of these ANN/DNN models were optimized by manual trial and error method. Accuracy scores, loss and validation curves, and confusion matrix were generated to evaluate this ANN/DNN model's performance and this performance was compared with the other ML models.

ML was performed on the full dataset (with combined phase A, B, and C data). However, the accuracy and other performance metrics were low for this full dataset. Hence, the same ML techniques were applied for the phase A data only as well. The comparisons on the various metrics between these two datasets and other significant results are presented in Section 3.

III. Results and Discussions

A. Grid Search

A grid search was performed on the LR, SVM, and TREE algorithms to optimize the models' hyperparameters. Tables I and II show the optimized parameters and their corresponding values for each of these algorithms. These tables also show the training and testing accuracy values for respective algorithms. Table I is for the phase A data only, while Table II is for the combined phase A, B, and C data (full dataset).

As shown in Table I, all three algorithms, after grid search optimization, had similar performance in terms of training and testing accuracy for phase A data. The accuracy values were in the mid ninety percent, which indicates that the ML was successful in learning and classifying the samples into binary digital bits.

On the other hand, Table II shows that the ML models were not as relatively successful in the same regard for the full dataset. This might be because the phase B and C dataset did not have the same amount of information on the communication signal as phase A, or the features that we extracted did not work as well for phase B and C data. As shown in Figure 2, the communication signal is injected into a single phase, and the image of this signal gets ingrained into the other two phases when the signal transitions through a distribution transformer. From our accuracy result, we can infer that the signal was injected directly into phase A, and the images were produced in phase B and C later in the PLC path.

B. Feature Selection

Next, to examine the most impactful features and to plot a 2D graph with decision regions for each model, we used the Sequential Backward Selection (SBS) [21] method to filter out the two most essential features from a total of 41 (20 each of amplitude envelope and RMSE plus one spectral centroid). The results are presented in Tables III and IV.

As shown in these tables, one of the two best features for every algorithm in both datasets was 'RMSE 1501-1600'. This is the RMS energy feature of the samples after being filtered with a 1501Hz-1600Hz band pass filter. This frequency range is significant because our input communication signal is at 1595Hz. This result shows that the ML models can correctly identify the frequency band location of our communication signal. Further, the tables also show that RMSE was consistently the best feature in all cases. This is expected because the main difference between the 1s and 0s in our input signal is the signal strength, and RMSE is the measure of this signal strength.

The tables also show the accuracy values of the models with just the two best features. Comparing these values to the values in Tables I and II, we can see that reducing the dataset features from 41 to 2 did not have a significant impact on the accuracy of the models.

A 2D plot of labels with decision regions was produced using the two best features for each model. Figure 4 shows such a 2D plot of the Phase A training and testing set with SVM decision regions.

TABLE I. GRID SEARCH RESULTS FOR PHASE A DATA

Classifiers	Optimized parameters	Training accuracy	Testing accuracy
Logistic Regression	C=1.0, solver=lbfgs	94.06	93.99
SVM	C=1000, gamma=0.001	94.45	95.19
Decision Tree	Max_depth=1, Min_samples_split=1.0	94.19	95.19

TABLE II. GRID SEARCH RESULTS FOR FULL DATASET

Classifiers	Optimized parameters	Training accuracy	Testing accuracy
Logistic Regression	C=1.0, solver=lbfgs	77.27	76.73
SVM	C=10, gamma=0.1	77.15	75.52
Decision Tree	Max_depth=5, Min_samples_split=7	73.84	72.22

TABLE III. FEATURE SELECTION RESULTS FOR PHASE A DATA

Classifiers	Two best features	Training accuracy	Testing accuracy
Logistic Regression	RMSE 201-300 and RMSE 1501-1600	93.29	94.58
SVM	RMSE 1301-1400 and RMSE 1501-1600	95.53	96.78
Decision Tree	RMSE 1-100 and RMSE 1501-1600	95.7	95.19

TABLE IV. FEATURE SELECTION RESULTS FOR FULL DATASET

Classifiers	Two best features	Training accuracy	Testing accuracy
Logistic Regression	RMSE 501-600 and RMSE 1501-1600	73.43	72.21
SVM	RMSE 701-800 and RMSE 1501-1600	76.6	74.92
Decision Tree	RMSE 501-600 and RMSE 1501-1600	78.86	74.51

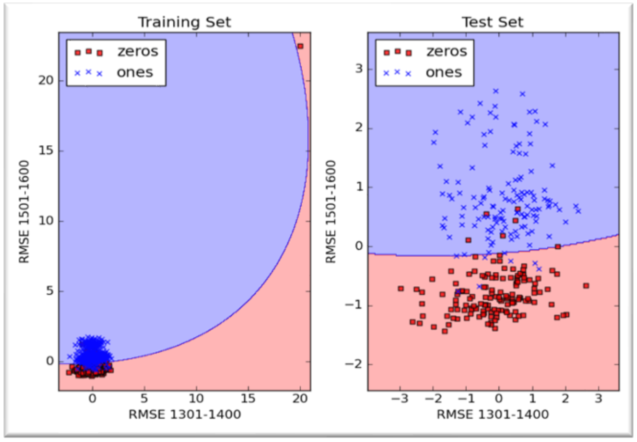


Figure 4. 2D feature plot showing labels and decision boundary of the SVM model for phase A data.

C. *Learning Curve*

To check if the ML models were overfitting or underfitting, we produced learning curves for each model. Overfitting is caused by high variance when models train with the noise and the appropriate data and produce a disproportionate result in the training and testing set [20]. In learning curves, overfitting can be implied if the training and validation accuracy curves do not converge and are far apart. On the other hand, underfitting is caused by high bias when the models do not consider all relevant data with appropriate weight. Underfitting can be implied in learning curves if the training and validation accuracy is consistently low [20].

Figure 5 shows the learning curve of the LR model for the phase A dataset. This shows that the model was not overfitted or underfitted. The two other models for the phase A dataset also had similar learning curves showing no overfitting or underfitting.

D. *Ensemble model*

After the LR, SVM, and TREE models were optimized, they were assembled into one classifier by soft (with probabilities) majority voting. The results of the individual classifier along with the ensemble model for phase A dataset and full dataset are shown in Table V. As shown in this table, both the phase A and full dataset had a slight decrease in the accuracy of their corresponding majority voting model compared to the best individual model.

E. *Confusion Matrix*

For all the models, including the ANN/DNN, confusion matrices were produced. The confusion matrix of the decision tree model for the phase A dataset is shown in Figure 6. It shows the number of True Positive (TP) on the top left quadrant, False Negative (FN) on the top right, False Positive (FP) on the bottom left, and True Negative (TN) on the bottom right. From these values, other metrics, including accuracy, can be calculated. In Figure 6, Precision =[TP/(TP+FP)], Recall =[TP/(TP+FN)] and F1 score =[2*(Precision*Recall)/ (Precision + Recall] [22] are calculated and shown on the plot title.

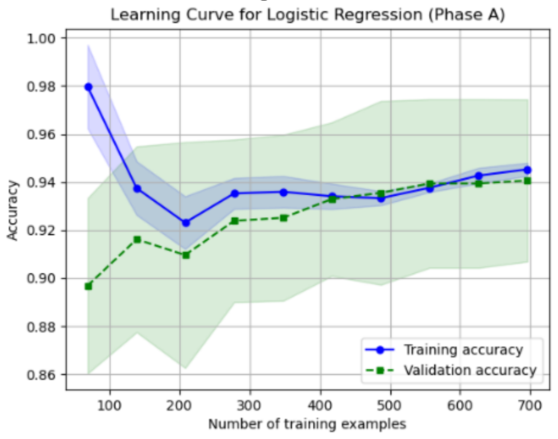


Figure 5. Learning curve for logistic regression model of phase A dataset.

As shown in this figure, the decision tree model correctly classified most of the labels (shown in blue quadrants) while it wrongly classified eight samples of each of the two labels (shown in white quadrants). SVM was the best model for the phase A dataset, while for the full dataset, LR was the best.

F. *ROC AUC Curve*

The Receiver Operating Characteristics (ROC) curve is a graph of model probabilities of False Positive Rate (FPR) versus True Positive Rate (TPR). FPR is the ratio of the number of False Positives (FP) to the total number of negatives (FP+TN), while TPR is the ratio of True Positive (TP) to the total number of positives (TP+FN) [23]. ROC curve shows a model's performance at all classification thresholds, and the Area Under this Curve (AUC) provides a metric for this performance measure [23]. Figure 7 shows the ROC curve and ROC AUC scores of LR, SVM, TREE, and Majority voting models for the phase A dataset. Please note that the straight diagonal line in the middle of the plot is a hypothetical model which cannot distinguish between the two classes and is equivalent to "guessing" the classification. Therefore, its AUC is 0.5. This diagonal line represents a threshold, and if a model falls below this threshold, it is performing worse than guesswork. As seen in Figure 7, our LR, SVM, TREE and Majority voting models' respective curves are close to AUC of 1. The performance of these models in this metric is very similar.

G. *ANN/DNN model and its Loss Curve*

After testing the three basic ML algorithms, we created an ANN/DNN model with the phase A dataset. The number of hidden layers in this model, number of nodes in each layer, activation functions for each layer, optimizer, and hyperparameters for the model were all tuned and optimized by trial and error. The results of this optimization are shown in Table VI.

With these parameters, the ANN/DNN model was trained with phase A data, and it produced a final training accuracy of 98.19% and testing accuracy of 94.29%. These accuracy values are slightly better than the corresponding accuracy values of LR, SVM, TREE, or Majority voting models. Figure 8 shows the training versus test (validation) curve of this ANN model. As shown in this figure, the model's loss decreased and stabilized as the model trained for more epochs.

TABLE V. ACCURACY VALUES FOR INDIVIDUAL AND ENSEMBLE MODEL IN PHASE A AND FULL DATASET

Classifiers	Phase A Dataset Accuracy	Full Dataset Accuracy
Logistic Regression	0.94	0.77
SVM	0.93	0.76
Decision Tree	0.92	0.74
Majority Voting	0.93	0.76

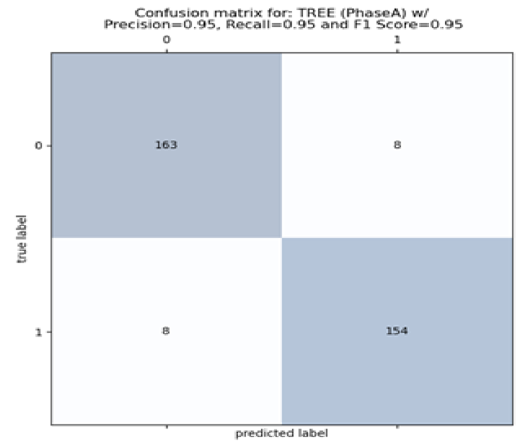


Figure 6. Confusion matrix of the decision tree model for phase A dataset.

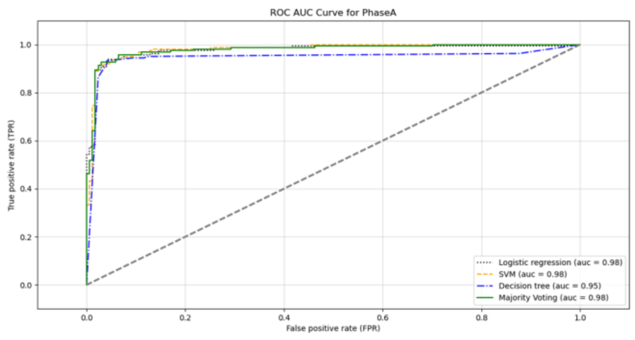


Figure 7. ROC AUC curve of LR, SVM, TREE and majority voting model.

TABLE VI. ANN/DNN OPTIMIZED HYPERPARAMETER VALUES

Number of hidden layers	2
Number of nodes in each hidden layer	50, 50
Activation function for each layer	tanh, tanh, sigmoid (for output layer)
Optimizer	Adam
Optimizer parameters	Learning rate of 0.01 and beta decay (beta_1) of 1e-5
Number of epochs	100
Validation ratio	0.01

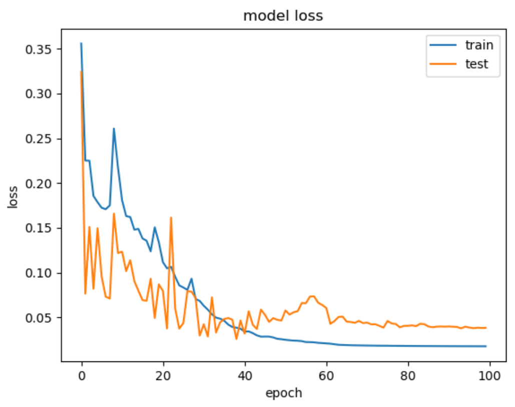


Figure 8. Training and testing (validation) loss curve for the ANN model.

IV. CONCLUSION

This research study extracted time-domain (amplitude envelope and RMS energy) and frequency-domain (spectral centroid) feature from raw PLC data to generate an ML-ready dataset. Then, we used the dataset in three supervised machine learning algorithms: logistic regression, support vector machine, and decision tree, to generate classification models. We optimized these models using the grid search method, investigated impactful features in each model using sequential backward analysis, checked for model's overfitting and underfitting using learning curves, and used accuracy, ROC AUC scores, precision, recall, and f1 score metrics to evaluate and compare the performance of the models. Using these performance metrics, we found out that all three models (and an ensemble model made by majority voting of the three) performed similarly, with SVM being slightly better than the rest because of its non-linear classification.

Then, we used an artificial neural network/deep neural network model with two hidden layers to perform the same classification task on the PLC dataset. This ANN model performed slightly better than the aforementioned basic ML models.

We also observed that all the models performed significantly better with the standalone phase A dataset than the full dataset containing data from all three phases. This is most likely because the input signal was initially transmitted through the phase A power line, and the phases B and C only got images of this signal along the PLC path. Hence, the deteriorated signal data in phases B and C diluted the full dataset and caused the model to be less accurate. In future works, the signal reception from the secondary phases can be improved, for example, by using a Rake receiver. This could result in a better performance from the full dataset.

REFERENCES

[1] K. Dostert, "Telecommunications over the power distribution grid–possibilities and limitations," IIR-Powerline, vol. 6, no. 97, 1997.

[2] A. M. Tonello, N. A. Letizia, D. Righini, and F. Marcuzzi, "Machine Learning Tips and Tricks for Power Line Communications," IEEE Access, vol. 7, pp. 82434–82452, 2019, doi: 10.1109/ACCESS.2019.2923321.

[3] S. U. Ercan, O. Ozgonenel, Y. E. Haj, C. Christopoulos, and D. W. P. Thomas, "Power line communication design and implementation over distribution transformers," in 2017 10th International Conference on Electrical and Electronics Engineering (ELECO), Nov. 2017, pp. 190–194.

[4] D. Kakati and S. C. Arya, "A full-duplex optical fiber/wireless coherent communication system with digital signal processing at the receiver," Optik, vol. 171, pp. 190–199, Oct. 2018, doi: 10.1016/j.ijleo.2018.05.140.

[5] T. Smyth, "Amplitude Envelopes", Department of Music, UCSD, 2019. [Online]. Available from: http://musicweb.ucsd.edu/~trsmyth/sinusoids171/Amplitude_Envelopes.html [retrieved: April, 2021]

[6] B. Boashash, "Chapter 4 - Advanced Time-Frequency Signal and System Analysis," in Time-Frequency Signal Analysis and Processing (Second Edition), Oxford: Academic Press, 2016, pp. 141–236.

[7] Energy and RMSE. musicinforetrieval.com. [Online]. Available from: https://musicinformationretrieval.com/energy.html#:~:text=The%20root%2Dmean%2Dsquare%20energy,x%2C%20sr%20%3D%20librosa [retrieved: April, 2021]

[8] J. M. Grey and J. W. Gordon, "Perceptual effects of spectral modifications on musical timbres.," Journal of the Acoustical Society of America, vol. 63, pp. 1493–1500, May 1978.

[9] sklearn.linear_model.LogisticRegression. Scikit-learn. [Online]. Available from: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html [retrieved: April, 2021]

[10] sklearn.svm.SVC. Scikit-learn. [Online]. Available from: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html [retrieved: April, 2021]

[11] sklearn.tree.DecisionTreeClassifier. Scikit-learn. [Online]. Available from: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html [retrieved: April, 2021]

[12] sklearn.model_selection.GridSearchCV. Scikit-learn. [Online]. Available from: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html [retrieved: April, 2021]

[13] sklearn.ensemble.VotingClassifier. Scikit-learn. [Online]. Available from: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html [retrieved: April, 2021]

[14] sklearn.metrics.roc_auc_score. Scikit-learn. [Online]. Available from: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html [retrieved: April, 2021]

[15] sklearn.metrics.precision_score. Scikit-learn. [Online]. Available from: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html [retrieved: April, 2021]

[16] sklearn.metrics.recall_score. Scikit-learn. [Online]. Available from: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html [retrieved: April, 2021]

[17] sklearn.metrics.f1_score. Scikit-learn. [Online]. Available from: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html [retrieved: April, 2021]

[18] Plotting Learning Curves. Scikit-learn. [Online]. Available from: https://scikit-learn.org/stable/auto_examples/model_selection/plot_learning_curve.html [retrieved: April, 2021]

[19] sklearn.metrics.confusion_matrix. Scikit-learn. [Online]. Available from: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html [retrieved: April, 2021]

[20] S. Raschka, Python machine learning. Packt publishing ltd, 2015.

[21] sklearn.feature_selection.SequentialFeatureSelector. Scikit-learn. [Online]. Available from: https://scikit-learn.org/dev/modules/generated/sklearn.feature_selection.SequentialFeatureSelector.html [retrieved: April, 2021]

[22] K.P. Shung. Accuracy, Precision, Recall or F1? Towards data science. Mar. 15, 2018. [Online]. Available from: https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9 [retrieved: April, 2021]

[23] D. M. Powers, "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation," arXiv preprint arXiv:2010.16061, 2020.

Remote Filesystem Event Notification and Processing for Distributed Systems

Kushal Thapa[†‡], Vinay Lokesh[*#], Stan McClellan[†§]
[†]*Ingram School of Engineering*
**Dept. of Computer Science*
Texas State University
San Marcos, TX, USA
e-mail: [‡]k_t260@txstate.edu, [#]v_v183@txstate.edu, [§]stan.mcclellan@txstate.edu

***Abstract*— Monitoring and safeguarding the integrity of files in local filesystems is imperative to computer systems for many purposes, including system security, data acquisition, and other processing requirements. However, distributed systems may have difficulty in monitoring remote filesystem events even though asynchronous notification of filesystem events on a remote, resource-constrained device can be very useful. This paper discusses several aspects of monitoring remote filesystem events in a loosely-coupled and distributed architecture. This paper investigates a simple and scalable technique to enable secure remote file system monitoring using existing Operating System resident tools with minimum overhead.**

Keywords*— Secure Remote Filesystem Monitoring; Firewall; Distributed Architecture; Secure Network Communication; SSH; Secure Shell Protocol; Filesystem.

I. INTRODUCTION

Most modern computer systems incorporate local storage containing files associated with user data, application data, and other important data, such as trade secrets and passwords [1]. The vast majority of attacks to such classified files are well known issues in day-to-day operations so it is vital to ensure the integrity of the file system. There are two general approaches to monitoring filesystems for unauthorized access: (a) Hash-based file integrity (b) Real-time file integrity. The hash-based approach is to scan critical files on systems on a regular schedule, detecting changes by comparing the current file hash to the previous version. In a real-time approach, the information is provided on not just file changes, but also on all the file read, write, and create events so determining a violation becomes much simpler [2].

Although most modern computer systems have several tools that are capable of tracking local file system events, it becomes much more complex to monitor file system events remotely from a central location [3].

Each system in distributed architecture is typically capable of monitoring filesystem events, such as creation, deletion, and changes in local files. This can be performed by tools like inotify [4], kqueue [5], FSEvent [6], direvent [7] etc. However, these tools inherently lack the ability to monitor remote filesystems. Tools like Secure Shell Protocol Filesystem (SSHFS) [8][9] allow a user to mount remote directories into the local system; however, file monitoring is not possible with SSHFS. Asynchronous notification of filesystem events on a remote, resource-constrained devices can be very useful, particularly in distributed acquisition architectures and other scenarios where data is processed asynchronously.

In distributed architecture, one of the complexities introduced by Internet-based (IP) networking is a firewall [10][11]. Firewalls are vital for network security; however, the presence of an intervening firewall can make communication between distributed systems much more complex. Many networking solutions and architectures allow the users to circumvent certain firewall restrictions, thus increasing complexity while introducing security risks. Here, we leverage the well-known network architecture where an Internet-reachable system acts as a middleman to establish a secure, bidirectional network connection between firewalled devices. This approach is not new, however, comprehensive analysis of various parameters is difficult to obtain, so we provide some results and discussion regarding the various configuration options and performance of this architecture.

In Section II of this paper, we describe various tools that are generally used to monitor local filesystem events. We also briefly discuss about Secure Shell Protocol Filesystem (SSHFS) [9] and Secure Shell Protocol (SSH) [12]. Section III presents our approach to an experiment, which presents different network architectures and usage of those architectures with SSHFS and SSH. In Section IV we evaluate local filesystem monitoring and network communications using metrics for (a) complexity, (b) portability, and (c) efficiency/speed. We conclude the paper in Section V describing the overall summary of our experiment.

II. BACKGROUND AND RELATED WORK

Inotify [4] is a filesystem event notification tool for Linux operating systems. This tool allows user to add an automated watch to a file or directory which can monitor certain filesystem event(s) (for example: open, write, modify, close, etc.). When those events occur on the file or the folders being watched, this tool provides asynchronous, event-driven notification to the user for user interaction. Inotify-tools provides command-line interface to inotify [13][14]. Inotifywait is a shell utility included in inotify-tools that waits for changes to files or folders, and outputs the description of these changes when made. There are many options available for this command [15] through which the user can specify the target for the watch, the nature of the watch and the format of the output. These options, along with the fact that multiple watches can be made simultaneously, makes this tool very easy-to-configure, user-friendly and scalable. However, inotify is a kernel feature, which only monitors local file system events, and thus, remote filesystem events, not implemented in the local kernel, are not registered by inotify [13][15].

iWatch [16] is a kernel feature written as a Perl wrapper for inotify to monitor changes in specific directories or files, sending alarms to system administrator in real-time. iWatch can run as a daemon, as well as via the command-line. The daemon mode uses an Extensible Markup Language (XML) configuration file to register a list of directories and files to monitor. The command line mode will run without a configuration file. In the XML configuration file, each target can have an individual email contact point. This contact point allows an email notification for any modification in the monitored targets.

kqueue [3][5] is an event notification interface in FreeBSD, supported by other operating systems such as NetBSD, OpenBSD, DragonflyBSD, and macOS. kqueue monitor demands a file descriptor to be opened for every file which is being watched hence restricting its application to very large file systems [5]. kqueue does not provide direct support for generic events such as ‘create’ for files and its Application Programming Interface (API) is designed with higher dependency on the local kernel, limiting the ability to work asynchronously with remote file system monitoring.

Filesystem Notification Events (FSEvents) [6] is an event notification API designed for macOS. FSEvents is a kernel feature and has a device file called /dev/fsevents. It follows a simple process where all the primal event notifications are passed to the userspace through this device file. The event stream it is then filtered by a daemon to publish notifications. The macOS version 10.7(lion) added the capability to watch filesystem [6]. The FSEvents monitor is not constrained by requiring unique watchers and thus scales well for large systems with huge number of directories. Although FSEvents can monitor a directory that is within a remote mounted volume and provides a callback for local changes, it cannot detect changes made by users on other machines.

FileSystemWatcher [17] is a specific class in the System.IO namespace, which is used to monitor and detect file system changes in Windows. It triggers events for every change that appear in file or directory which is being watched. It generates a new instance for FileSystemWatcher with arguments required to specify the directory and type of files which needs to be monitored, and a buffer into which all the file changes are written. The kernel then reports file changes by writing to that buffer. This suffers event loss when large number of changes are pushed into the buffer. One of the drawbacks of filesystem watcher is that it can only establish a watch to monitor directories, not files. To monitor a file, its parent directory must be watched in order to receive change events for all of the directory’s children.

Tripwire is one of well-known file integrity program[18][19]. Tripwire was essentially built as a strong file integrity checker for Unix systems. The original Tripwire was termed as Academic Source Release (ASR) which has features such as strong set of supported hash functions, the power to examine file attributes, and a good configuration. It was a freely available program with reporting capabilities limited to results displayed only on the terminal screen. It also lacks database protection and verification capability.

The **Python Watchdog** module [20] is used to monitor file system events. Python Watchdog has a standard API for developers to select and deploy a monitor. Facebook’s Watchman [21] is similar to Python’s Watchdog module which also provides a similar interface for initiating different monitors. However, both these tools are operating system dependent making it infeasible for remote file monitoring and processing.

Direvent, like inotify, is a filesystem monitoring tool. However, direvent works in GNU/Linux, BSD and Darwin (Mac OS X) systems [7]. This allows for uniformity, and possibly integration of file monitoring processes, across diverse systems in a network. The files and directories to be watched, along with their corresponding target event, are specified in the direvent configuration file with ‘watcher’ statements. Filesystem events can be divided into two major groups. (a) system-dependent events that are specific for each kernel interface (b) generic events that do not depend on the underlying system. They provide a higher level of abstraction and make it possible to port configurations between various systems and architectures. However, direvent relies on the local event monitoring Application Programming Interface (API) provided by kernel [22]. As a result, it is not natively compatible with remote file system monitoring. When compiling with Berkeley Software Distribution (BSD) systems direvent uses kqueue – another kernel event notification mechanism.

Secure Shell Protocol Filesystem (SSHFS) [9] is a file system in user space (FUSE) that uses the SSH File Transfer Protocol (SFTP) to locally mount a remote file system. The mounted file systems can be accessed and used the same way a local file system is, both from the command line or using other tools. Unfortunately, inotify is not aware of filesystem changes on an SSHFS mount which are initiated from the remote end of the link.

Secure Shell Protocol (SSH) [12][23] is a secure network communication paradigm that operates at the Open System Interconnection (OSI) session level. All session data transferred through an SSH connection is transparently encrypted. This encryption is transparent, which means it gets decrypted by SSH client daemon at the specified destination, and thus, users do not have to deal with decrypted data, because of its utility and security features, SSH is widely used for remote system management tasks and can incorporate multiple use-cases, including forwarding graphical sessions, automating “jump” behavior to access systems behind firewalls, and so on.

a) Multiplexing

SSH has the ability to carry multiple sessions over single TCP connection via “multiplexing”. One of the benefits of multiplexing is that it speeds up certain operations that utilize an SSH session.

b) Reverse Port Forwarding

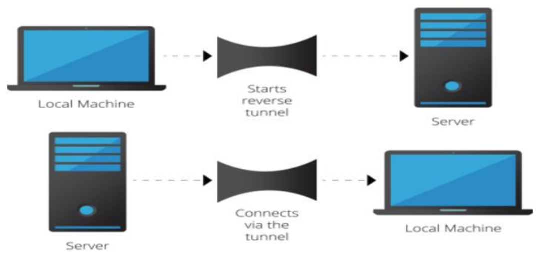


Figure 1. Working of SSH Reverse Port Forwarding [24].

Figure 1 shows the working of Reverse SSH port forwarding. It is a technique through which systems that are behind a firewall can be accessed from the outside world. With this technique, a port on a remote machine can be forwarded to the local machine while still initiating the tunnel from the local machine. This works by listening to the port on the remote side, and whenever a connection is made to this port, the connection is forwarded over the secure channel to the host port from the local machine.

III. APPROACH

Our evaluation of these several alternatives consists of two parts: (a) building a simple and secure network architecture to communicate between devices, and (b) using that architecture to test remote, asynchronous filesystem monitoring. The network architecture as well as file monitoring test setup that is designed and described in this paper is simplified to only two devices, however, this system is scalable to include any number of devices.

A. *Network Architecture*

Network communication is one of the primary challenges when creating a system where interconnectivity between devices is required. For this research, we have chosen an SSH based network architecture which leverages an IP reachable system acting as a “jump server” to establish communication between devices behind their individual firewall. Figure 2 shows the basic components of this three-prong architecture.

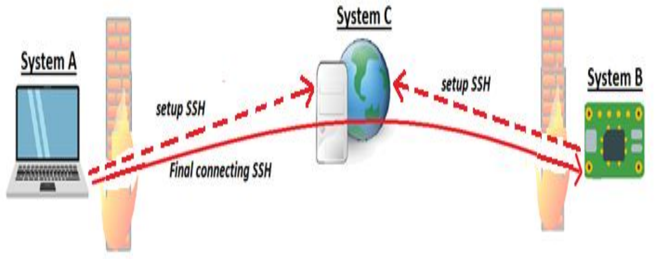


Figure 2. Representation of three-prong architecture.

As shown in Figure 2, system A is used to control remote devices behind a firewall, represented by System B. Both A and B are behind network firewalls, so a direct SSH connection cannot be made from A to B or vice versa. Thus, the need of system C, which is an open IP reachable server. Both A and B can communicate with C because firewalls allow outgoing connections. Using port forwarding, an SSH tunnel can be made from A to B via C. In this architecture, system C acts merely as SSH reflector, and no special configuration is necessary.

One of the main advantages of this architecture over other architectures and network solutions that circumvent the firewall restrictions is its simple configuration. The configuration for establishing this network begins when the first host (A) which creates an SSH connection to C. Similarly another host (B) also creates an SSH connection to C. Finally, to create connection between A and B ports from A and B are forwarded to a common port in C, thereby creating a continuous tunnel from A to B. Two other simpler network architectures (referenced as Arch-0 and Arch-2 in this paper) were built using the components of our primary architecture (Arch-1) to compare the results. The design of these two architectures are shown in Figure 3 and Figure 4, respectively.

i. *Arch-0*

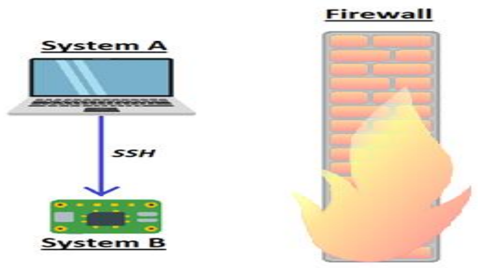


Figure 3. Representation of systems and network connection behind a common firewall.

As shown in Figure 3, there is a common architecture between systems in the same network where there is no firewall between hosts. System A can SSH directly into system B. This architecture is used as a control and as a baseline in this study.

ii. *Arch-2*

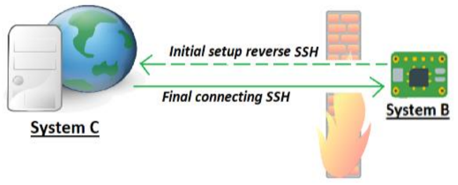


Figure 4. Representation of components, security zones and connections in a client-server architecture.

Figure 4 shows the network architecture that is used more commonly as an alternative for our three-prong architecture. Here, the peripheral devices indicated by system B are controlled directly by an open IP reachable server. The system C cannot directly form an SSH connection to B because of the presence of a firewall. Thus, B needs to initiate a special SSH tunnel which is used by C to form a reverse SSH channel back to B.

In this architecture, C acts as the control center. Thus, the main difference between our three-prong architecture and this architecture lies in the role of the server. In this architecture, the server’s resources are heavily utilized. This can be advantageous if the server is powerful. However, since the server is internet reachable, it can pose security risks if its configuration is too complex. The main advantage of the three-prong architecture is in “hidden complexity” because the control unit is protected by firewall, so the exposed attack surface is minimized.

To evaluate these architectures, we used multiplexed SSH combined with a simple timing test which is shown using a vertical line diagram in Figure 5. This test program sends a simple UNIX command via regular and multiplexed SSH tunnels, and records the time taken to send, execute and receive the output of this command. To account for the variability of network speed at different times, this program was run every hour of every day for about a month. The results of this experiment are compiled in Section IV.

Figure 5 shows the three network architectures of interest, Arch-0, Arch-1 and Arch-2. In this case, Arch-1 indicated by red is a “three prong” network architecture, whereas Arch-0 shown in Figure 3 and Arch-2 shown in Figure 4. indicated blue and green respectively, are two other simpler architectures built using the primary Arch-1 architecture. The bold lines represent multiplexed connections of these three architectures while the thin lines represent non-multiplexed connections. The dashed lines represent connections necessary for their corresponding architecture. For our timing test, both system A and B are on the same network. In case of Arch-1 the connections goes through System C hence the firewalls can be separate.

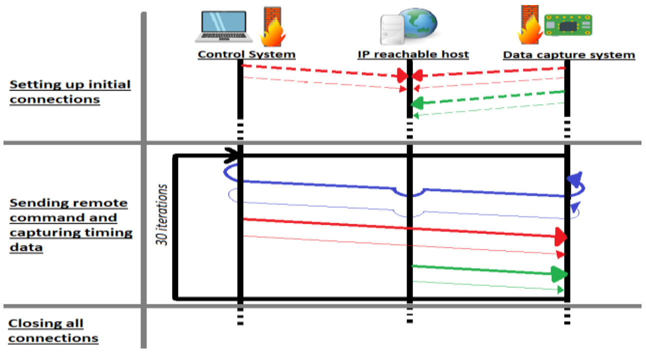


Figure 5. Vertical line diagram to represent three network architecture, Arch-0, Arch-1 and Arch 2.

B. Remote Filesystem Monitoring

Application specific file monitoring is useful in detecting certain changes and responding to those changes. However, tools such as inotify and direvent lack capability to monitor files and directories of other devices in the network. To overcome this challenge, we took two approaches:

1) Using SSHFS

Secure Shell Protocol Filesystem SSHFS [11] enables the user to mount remote filesystem in the local filesystem. The user can access and monitor the mounted files and directories manually. To automate this monitoring process, we tried coupling SSHFS with inotify by mounting a remote filesystem and monitoring changes on it.

In general terms, inotify is not aware of filesystem changes on an SSHFS mount which are initiated from the remote end of the link. This is because SSHFS is built on top of SFTP; hence, it is a client view of the remote filesystem and does not export filesystem events from the remote system.

2) Using SSH

Using Secure Shell Protocol (SSH) [16], we devised a two-step method for remote filesystem monitoring which is simple, intuitive and scalable, and utilizes the light pre-existing OS-resident tools. The target file or directory in the local system is monitored using a tool such as inotifywait.

Then, using the event registration of the monitored target as a trigger, a command is sent to the other end of the channel using SSH. Such SSH appended commands can be a simple OS command or another trigger to a script, and thus can be easily modified according to application requirements.

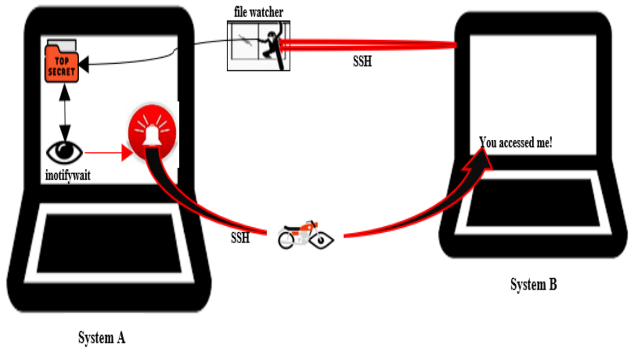


Figure 6. Remote Filesystem monitoring using SSH and inotify tools.

As shown in Figure 6, an inotifywait is issued on a top secret directory in System A, so whenever a filesystem event occurs in that watched section of the filesystem, the event is transmitted to the remote monitoring configuration on System B along with a timestamp of the event.

IV. Results

We evaluate different network architectures mentioned in Section III along with a timing data as shown in Figure 7 and Figure 8. Multiplexed SSH connections significantly reduce connection time because the TCP handshake and keying interaction to set up the SSH session is already performed, and is being re-used and efficiently. From Figure 7 and Figure 8, it is clear that Arch-0 exhibits the fastest communication time in both multiplexed and non-multiplexed architectures, which is reasonable since both systems are on the same network, with no firewall.

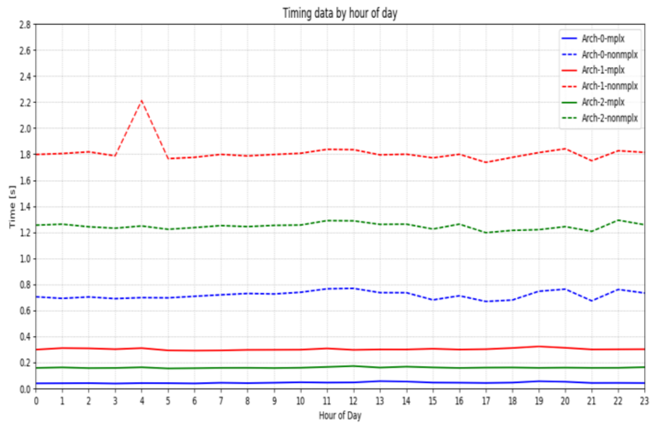


Figure 7. Timing data by hour of the day.

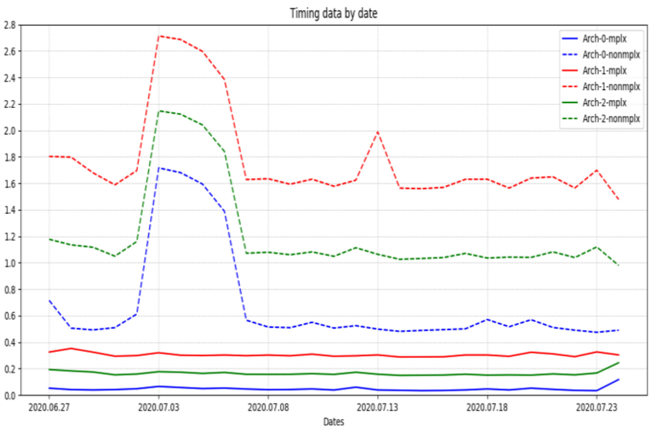


Figure 8. Timing data by date.

Also from Figure 7 and Figure 8, the multiplexed connection of Arch-1 recorded lower time than non-multiplexed version of Arch-0, which is interesting because in Arch-1 a firewall exists between systems, so TCP/keying lags are substantial, even for systems on the same network. The non-multiplexed connections exhibit substantial random latencies due to multiple network transits of TCP handshakes and keying interchanges, whereas corresponding points in the multiplexed configuration do not exhibit the same issues due to the more efficient re-use of multiplexed connections. The performance of multiplexed connections is much more consistent.

V. Conclusion

This paper discusses the use of various filesystem monitoring tools which support local file system monitoring, but inherently lack ability to monitor remote filesystems. In distributed and loosely coupled architectures, monitoring of filesystem events on remote systems, possibly behind firewalls, can have important application-layer benefits and utility. To examine the performance of various system configurations, we evaluate network architectures with both multiplexing and non-multiplexing techniques, concluding that a simple and scalable technique using multiplexed SSH connections and inotify tools enables secure remote file system monitoring with minimum overhead. By recording timing of filesystem events on each of these network architectures, we note that multiplexed SSH connections are consistent, and much more efficient than other methods, even with complex distributed architectures involving exposed systems, multiple firewalls, and "three prong" SSH tunnels.

References

[1] K. P. Suresh, U. Himanshu, and L. Leonel, “File integrity monitoring tools: issues, challenges, and solutions,” [Online]. Available from: https://onlinelibrary.wiley.com/doi/epdf/10.1002/cpe.5825 [retrieved April 2021]

[2] S. Evangelou, “How to verify File Integrity using hash algorithms in Powershell,” [Online]. Available from: https://stefanos.cloud/blog/kb/how-to-verify-file-integrity-using-hash-algorithms-in-powershell/ [retrieved April 2021]

[3] A. K. Paul et al. "FSMonitor: Scalable File System Monitoring for Arbitary Storage Systems," in IEEE International Conference on Cluster Computing (CLUSTER) Cluster Computing (CLUSTER), Albuquerque, 2019, pp. 1-11.

[4] R. Love, "Kernel Korner - Intro to Inotify," Linux Journal, 2005.

[5] J. Lemon, "Kqueue – A generic and scalable event notification facility," in Proceedings of the FREENIX Track: USENIX Annual Technical Conference, Boston, 2001, pp. 1-14

[6] Apple, "File System Events Programming Guide," 13 December 2012.[Online]. Available from: https://developer.apple.com/library/archive/documentation/Darwin/Conceptual/FSEvents_ProgGuide/Introduction/Introduction.html#//apple_ref/doc/uid/TP40005289-CH1-SW1. [retrieved April 2021].

[7] S. Poznyakoff, "GNU Direvent," 13 July 2019. [Online]. Available from: https://www.gnu.org.ua/software/direvent/manual/direvent.html. [retrieved 06 August 2020].

[8] N. Rath, "libfuse/ sshfs," [Online]. Available from: https://github.com/libfuse/sshfs. [retrieved March 2021].

[9] M. E. Hoskins, "SSHFS: Super Easy File Access over SSH," Linux Journal, 2006, pp. 1-6.

[10] J. R. Vacca and S. Ellis, Firewalls : jumpstart for network and systems administrators, Elsevier Digital, 2005.

[11] W. Noonan and I. Dubrawsky, Firewall fundamentals, Indianapolis, Cisco, 2006.

[12] D. J. Barrett, R. G. Byrnes, and R. E. Silverman, "Introduction to SSH," in SSH, The Secure Shell: The Definitive Guide : The Definitive Guide, O’Reilly Media, Inc., 2011, pp. 1-15.

[13] A. Schwab, "inotify-tools," [Online]. Available from: https://github.com/inotify-tools/inotify-tools. [retrieved April 2021].

[14] C. Fischer, "Linux Filesystem Events with inotify," Linux Journal, 2018. pp. 1-17.

[15] R. McGovern, "inotifywait(1) - Linux man page," [Online]. Available from: https://linux.die.net/man/1/inotifywait. [retrieved April 2021].

[16] C. Wirawan, “iwatch - realtime filesystem monitoring program using inotify,” Ubuntu man page [retrieved 25 March 2021]

[17] Microsoft. FileSystemWatcher. https://docs.microsoft.com/en-us/dotnet/api/system.io.filesystemwatcher?redirectedfrom=MSDN&view=netframework-4.7.2, 2010. [retrieved April 2021].

[18] D. Armstrong, “An introduction to file integrity checking on unixsystems,” [Online] Available from: https://www.giac.org/paper/gcux/188/introduction-file-integrity-checking-unix-systems/104739. [retrieved April 2021].

[19] G. Kim and E. H. Spafford, “The Design and Implementation of Tripwire A File System Integrity Checker,” In:2nd ACM Conference on Computer and Communications Security, pp. 18–29. ACM, Fairfax, VA, USA (1994)

[20] Python Watchdog. [Online]. Available from: https://pypi.org/project/watchdog/, 2010. [retrieved April 2021].

[21] Facebook Watchman. “A file watching service. https://facebook.github.io/watchman/, 2015”. [retrieved April 2021].

[22] M. Kerrisk, "Filesystem notification, part 2: A deeper investigationof inotify," 14 July 2014. [Online]. Available from: https://lwn.net/Articles/605128/. [retrieved April 2021].

[23] T. Ylonen and C. Lonvick, "The Secure Shell (SSH) Protocol Architecture," RFC 4251, January 2006. [Online]. Available from: https://www.rfc-editor.org/info/rfc4251. [retrieved April 2021].

[24] J. Knafo, “What is reverse SSH Porting” Available from: https://blog.devolutions.net/2017/3/what-is-reverse-ssh-port-forwarding. [retrieved March 2021]

An Evaluation of Neural Network Performance Using Complex-Valued Input Data

Kushal Thapa*, Stan McClellan†
Ingram School of Engineering
Texas State University
San Marcos, TX, USA
email: *k_t260@txstate.edu, †stan.mcclellan@txstate.edu

Abstract—**Complex-valued data is ubiquitous in many scientific fields. However, machine learning for complex-valued input is still in the developmental stage. Alternatively, complex data can be transformed to real data in a few different ways to fit the traditional machine learning framework. In this research, we compare the performance of two such ways - combining real and imaginary components or stacking them - on a simple neural network. To compare these two methods, we create magnitude (combined) and rectangular (stacked) spectrograms from artificial time-series data. Then, we feed the raw 1D time-series dataset, 2D magnitude spectrogram dataset, and 3D rectangular spectrogram dataset to a neural network for training and validation. As a measure of performance, we track the accuracy of each dataset model. From our experimentation, we found out that the rectangular dataset outperforms the magnitude spectrogram in most cases.**

Keywords-complex-valued data; machine learning; neural network; real spectrogram; imaginary spectrogram.

I. Introduction

Machine Learning (ML) is a computational technique of building models for complex systems using experiential data. Data is at the heart of machine learning. Therefore, the quality, quantity, format, and other characteristics of data have huge impact on the efficacy and effectiveness of the ML models. The quality and quantity aspect of data in ML, in a generalized sense or for a specific domain/application, are well documented in archive literature, such as [1]–[3]. In this paper, we focus on the performance of ML with input data in complex-valued format.

Complex-valued data contains information from both real and imaginary axes. This kind of data is present in many scientific applications and areas, such as signal processing [4] in communication systems, Magnetic Resonance Imaging (MRI) [5] in biomedical imaging, seismic monitoring [6] in geosciences, etc. ML can be a great tool in research and technological development in these areas; however, ML algorithms typically do not handle complex numbers well [7]. Thus, complex data can be pre-processed for ML in these applications by either a) taking only the real component and ignoring the imaginary component or b) combining the real and imaginary components in some way to produce real-valued data or c) separating the real and imaginary components and feeding them simultaneously to the same ML model. Approach a) is generally not desirable because of the loss of information caused by ignoring the imaginary component completely. Interestingly, for approaches b) and c), we are unable to find general guidelines in the literature which describe performance differences or areas of optimality. Hence, the objective of this paper is to make a comparison of these two complex-valued data pre-processing methods for ML with the aim of setting a general guideline when dealing with complex datasets in training ML models.

There is a fourth approach as well, which uses novel Neural Network (NN) models like Complex-Valued Neural Network (CVNN) that can handle the complex dataset. In this approach, the complex-valued data does not need any format change during pre-processing. However, CVNN and the general use of complex numbers in “deep learning” seems to be an active research area, with a lot of different concepts [6]–[12]. Interestingly, the predominant use of real-valued weights in neural networks seems to derive from the focus on real-valued optimization problems. As shown in [8], the use of complex-valued networks (CVNN) on datasets with phase characteristics results in better performance. However, contemporary CVNN are very complicated and sensitive, and the added complexity might not be worth the disruption to the toolchain for most applications. Here, we focus on the use of conventional technology and tools in a way that does not involve a complete re-structuring of the toolset. Hence, our study is limited to the comparison of pre-processing methods of complex-valued dataset for use in real-valued NN.

This paper is organized as follows. Section 2 describes the experimental setup we used to leverage two pre-processing methods for complex-valued data. Section 3 provides the result of the performance comparison between these pre-processing methods when used in a simple NN. Section 4 provides the conclusion of this experimentation.

II. Experimental Setup

To be able to control, finetune and vary the different data parameters for our comparison, we leverage an artificial dataset. This artificial dataset is based on simple single-bit detection/classification, which is a fundamental component of digital communication. The general findings from the experiments in this dataset should be applicable beyond this domain as well.

Figure 1 shows the flow of our experiment, divided into three main steps. The first step is the creation of artificial time-series data. From a randomly generated binary class digital information {length=10,000 bits}, analog time-series signals were created using Amplitude Shift Keying (ASK), Frequency Shift Keying (FSK), and Phase Shift Keying (PSK) {sampling rate=10,000; number of bits per second=100}.

Then, varying levels of AWGN {from SNR= -21dB to 21dB in increments of 3dB} were added to the clean signals to make our raw time-series data (Test 1). In the second step, this 1D raw time-series data was transformed into a 2D complex-valued spectrogram using Short-Time Fourier Transform (STFT) {frame length=50, frame overlap=50%, window=Hanning}. Then, the complex-valued data was transformed into magnitude-only spectrograms (described as Approach 'b' in Section I: 'Introduction'), as well as real/imaginary spectrograms. The real / imaginary spectrograms were stacked to make a 3D dataset as described in [13] (Approach 'c'). Finally, the 1D time-series dataset, the 2D magnitude spectrogram dataset and the 3D real-imaginary dataset (referred henceforth as rectangular dataset) were flattened and fed into a fully connected NN with one-hidden layer. Although our main objective was to compare the magnitude spectrogram and rectangular spectrogram in a NN, we used time-series dataset as a control input to evaluate and compare the complexity and performance of NN for the other two datasets. In all cases, the hyper-parameters of this NN, listed in Table I, were kept identical. For 'Test 2', an ideal power signal {60Hz, 120V RMS} was added to the noisy time-series signal at the end of 'Step 1', and steps 2 & 3 were repeated. The purpose of 'Test 2' was to simulate the presence of a dominant interfering signal in the raw data. The performance of the NN models from both tests were evaluated primarily using the accuracy metric. This is because accuracy in our experimental context characterizes the Bit Error Ratio (BER), which is an important metric in digital communication. BER is the ratio of wrongly classified bits (or error bits) to the total number of transmitted or evaluated bits. Thus, BER is the "unit complement" of accuracy, i.e., BER + accuracy =100%. Other performance metrics, such as precision and recall, were also measured (see [14] for the data file containing these metrics) but are not evaluated in this paper.

III. RESULTS AND DISCUSSIONS

A. *Modulation Intensity*

While converting the binary information to an analog signal using either of the three modulation schemes (ASK, FSK or PSK), the differentiating parameter between bit values or states can impact detection. Here, we define the modulation intensity as the difference between these parameter values. For example, if the amplitude of the 'high' is set to '1' and 'low' to '0.4' in ASK, the modulation intensity is 60%. In practice, this modulation intensity is dependent on various external factors, which are not the focus of this study. We chose the intensity parameter based on a subjective 'inflection point' in a NN model-accuracy versus 'low values' graph as shown in Figure 2. Based on these plots (see [14] for similar FSK and PSK figures), we set the intensity values as shown in Table II. Clearly, as intensity decreases, differentiation between bit values or states becomes more difficult, and thus, the NN model classification accuracy drops.

B. *Training Time Comparison*

The architecture of the NN for all signal types and datasets was the same. However, the size of the datasets was different as shown in Table III. Since the size of each sample was different, the training time was also bound to be different. Figure 3 shows the distribution of the total training time for each type of dataset (ASK, FSK and PSK). The time-series dataset had the quickest training time due to small size and single dimensionality. The rectangular spectrogram dataset was the slowest, with the training time almost twice as much as magnitude spectrogram dataset. Depending on the application, the training time of a NN or ML model can have vital consideration.

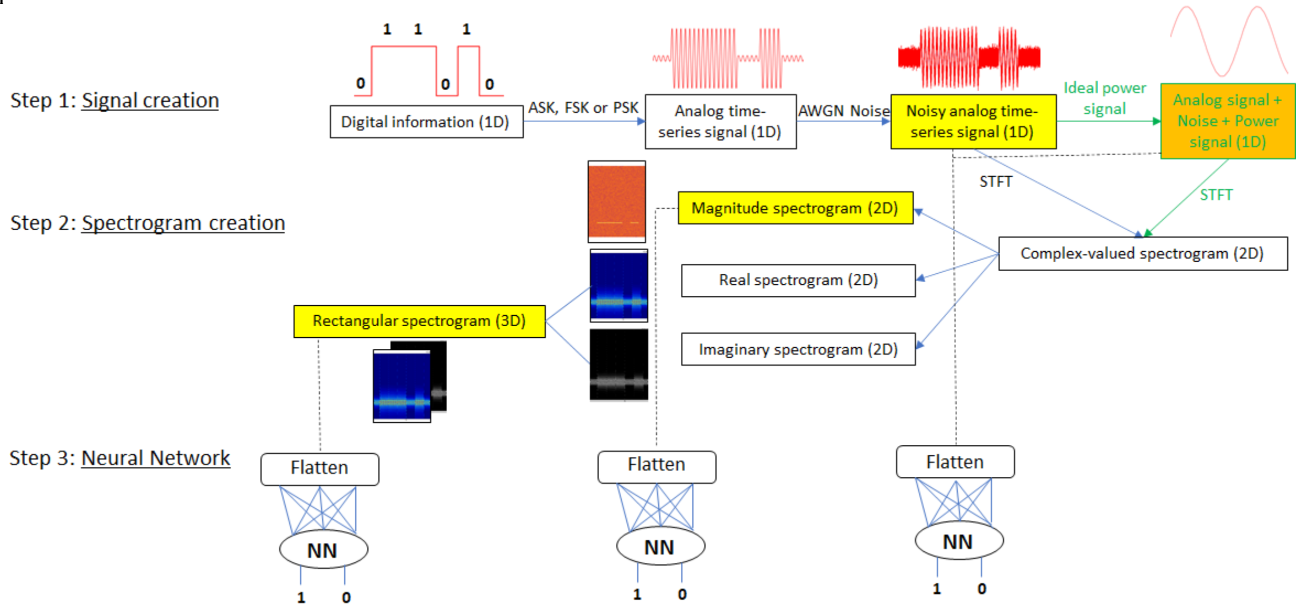

Step 1: Signal creation
1 1 1
0 0 0
Digital information (1D)
ASK, FSK or PSK
Analog time-series signal (1D)
AWGN Noise
Noisy analog time-series signal (1D)
Ideal power signal
Analog signal + Noise + Power signal (1D)
Step 2: Spectrogram creation
STFT
STFT
Magnitude spectrogram (2D)
Complex-valued spectrogram (2D)
Real spectrogram (2D)
Imaginary spectrogram (2D)
Rectangular spectrogram (3D)
Step 3: Neural Network
Flatten
NN
1 0
Flatten
NN
1 0
Flatten
NN
1 0

Figure 1. Flow of experiment showing creation of raw time-series modulated signal, transformation to various spectrograms and the use of the three datasets (highlighted) in NN. The extra sub-step of addition of power signal for 'Test 2' is shown in green.

TABLE I. HYPER-PARAMETERS OF THE NEURAL NETWORK

Total number of samples	10,000
Training to Test ratio	70:30
No. of hidden layers	1
No. of nodes in the hidden layer	64
No. of nodes in the output layer	2
Activation function for the hidden layer	Relu
Activation function for the output layer	Softmax
Optimizer	RMSProp
Loss function	Categorical Entropy
No. of training epochs	10
Batch size for training	16

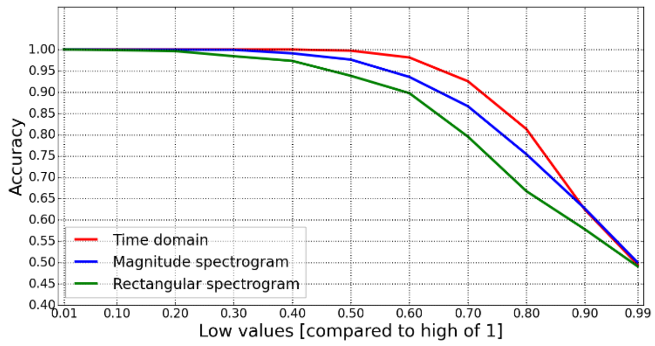


Figure 2. The NN model's test accuracy for a range of 'low values' (compared to a high of '1') for ASK signal with SNR=0dB. The plot shows general decrease in accuracy as 'low-values' get closer to the high-value, i.e., as modulation intensity decreases.

TABLE II. MODULATION INTENSITY VALUES

	High	Low
ASK	1 V	0.7 V
FSK	1000 Hz	950 Hz
PSK	0°	25°

TABLE III. DATASET SIZE

Dataset	Size (each sample)
Time-series (1D)	100
Magnitude spectrogram (2D)	3 x 1024
Rectangular spectrogram (3D)	3 x 1024 x 2

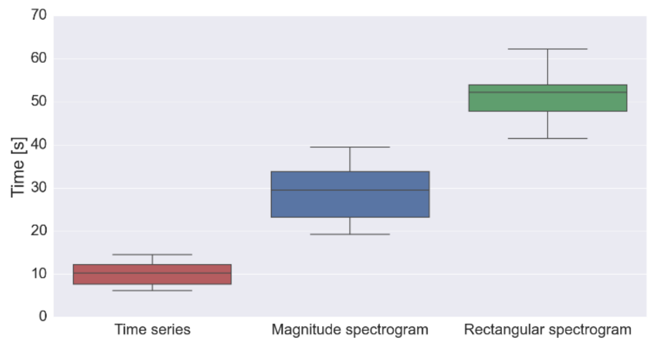


Figure 3. Boxplot showing the total training time distribution for the time-series (red), magnitude spectrogram (blue) and rectangular spectrogram (green) NN models.

C. *Test accuracy*

1) Test 1

a) ASK signal

Figure 4 shows the accuracy of the NN models for the time-series (1D), magnitude spectrogram (2D) and rectangular spectrogram (3D) datasets containing ASK signals with SNR ranging from -21dB to 21dB. For all models, there is a general trend of increase in testing accuracy when the SNR increases. This is, again, a fairly intuitive behavior since higher SNR means the dataset is 'cleaner' and the NN models can better differentiate between the 'highs' and 'lows' of the core signal.

The comparison between the three datasets is more interesting. In low SNR conditions (less than -15dB), the rectangular spectrogram model seems to perform slightly better than the magnitude spectrogram model. However, as the SNR increases, the performance of the magnitude spectrogram model improves rapidly and overtakes the rectangular spectrogram model after about -6dB. This can be explained by the type and quantity of information each dataset contains. Magnitude spectrogram, by definition, contains the magnitude or energy information of the signal, which is directly proportional to the signal amplitude. So, for ASK signals, the magnitude spectrogram more clearly represents modulation transitions in higher SNR conditions, thus simplifying the task of the NN as compared to the rectangular spectrogram. In contrast, the rectangular spectrogram holds more information about the signal, which is a boon in low SNR conditions but also the cause of data dilution resulting in lower performance compared to magnitude spectrogram in high SNR condition.

b) FSK signal

Figure 5 shows a similar NN model accuracy versus SNR plot as Figure 4, but for FSK signals. In contrast with the ASK signals of Figure 4, the comparison between the time-series, rectangular spectrogram and magnitude spectrogram models is more consistent across all SNR levels. The time-series and rectangular spectrogram models are evenly matched, and better than magnitude spectrogram models (until convergence occurs) in terms of accuracy.

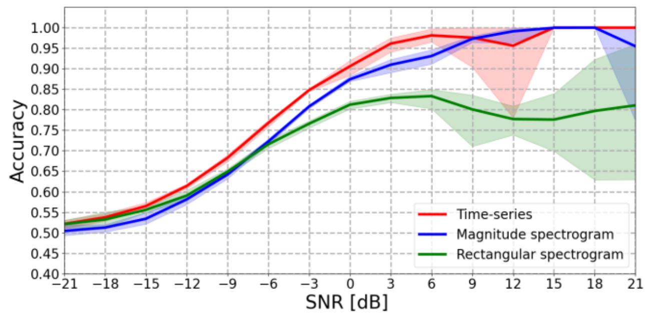


Figure 4. Test accuracy of NN models trained with time domain, magnitude spectrogram and rectangular spectrogram datasets containing ASK signals (high=1, low=0.7) with SNR ranging from -21dB to 21dB. Time-series model had generally highest accuracy while the rectangular spectrogram model shows better performance than magnitude spectrogram model only in low SNR conditions.

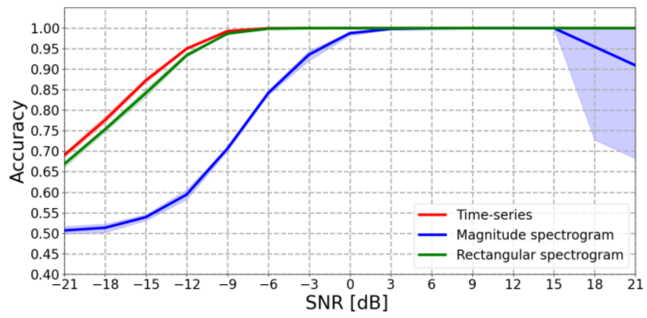


Figure 5. Accuracy versus SNR plot for NN models of FSK signals (high=1000 Hz, low=950 Hz) showing similar performance of time-series and rectangular spectrogram models while the magnitude spectrogram model performed worst across all SNR levels.

The higher accuracy of rectangular spectrogram compared to magnitude spectrogram can again be explained by the quality and quantity of information represented by the magnitude and rectangular spectrograms. The real, imaginary and magnitude spectrograms all contain the frequency shift information. By stacking the real and imaginary parts together, the quantity of information is doubled in rectangular spectrogram. However, unlike ASK, this does not dilute the dataset because the quality of information in relation to frequency is the same in all three sets. Therefore, the 3D rectangular spectrogram performs better than the 2D magnitude spectrogram over all SNR values.

c) PSK signal

Figure 6 shows a similar NN model accuracy versus SNR plot as Figures 4 and 5 but for PSK signals. As with the FSK models, the time-series and rectangular spectrogram model accuracies are evenly matched across all SNR levels. However, the magnitude spectrogram models were stuck at around 50% accuracy regardless of the SNR level. By definition, magnitude spectrogram completely ignores the phase information of the complex-valued spectrogram, and thus, the corresponding model can't distinguish between the different phases of the PSK signals. In contrast, the rectangular data retains this phase information indirectly, as indicated by the improved rectangular spectrogram accuracy curve in Figure 6.

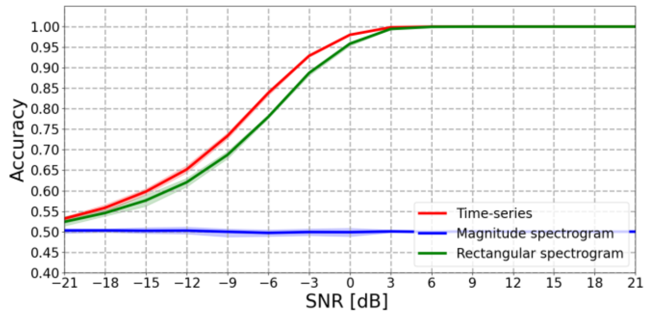


Figure 6. Accuracy versus SNR plot for NN models of PSK signals (high=0°, low=25°) showing the similar performace of time-series and rectangular spectrogram models. The magnitude spectrogram models' accuracy was approximately 50% for all SNR levels because of its inherent inability to retain phase information.

2) Test 2

From Figure 4-6, we observe that time-series NN model performed better than the spectrogram models in all three cases. This can be attributed to the unprocessed information that this raw time-series signal contains. Spectrograms need pre-processing, and each pre-processing step results in some information loss. Hence, the pre-processed spectrograms contained less information than the unprocessed time-series signal. However, the information contained in the time-series signal can be confused in the presence of fake or interfering signals. In such cases, we expect the classification performance of the time-series NN model to suffer. To test this hypothesis, we conducted 'Test 2'.

As explained in Section II, Test 2 is similar to Test 1 except that an ideal power signal is included to the AWGN added modulated signal as a strong out-of-band interferer. In practical terms, this power signal simulates a dominant interfering signal that makes the identification and classification of the desired signal more difficult and can confound the ML training process. This test case directly corresponds to an application scenario of a power line communication medium where the power signal is much stronger than the communication signal and affects the reactive channel in various other ways.

Figure 7 shows the accuracy of the NN models for time-series, magnitude spectrogram and rectangular spectrogram datasets each with ASK, FSK or PSK signals. This figure shows that the time-series and magnitude spectrogram NN models fail to produce any notable result regardless of modulation type. On the other hand, the rectangular spectrogram models are able to compensate for the dominant, out-of-band interfering signal and produce a good classification result. This figure also shows that this plot is not merely a far-left extension (very low SNR) of the plots in Figures 4, 5 and 6, at least not for rectangular spectrogram models, as the accuracies approach 100%. The sharp increase in accuracies with increasing SNR indicates that the rectangular spectrogram model is affected more by the less energetic AWGN than the highly energetic out-of-band interferer.

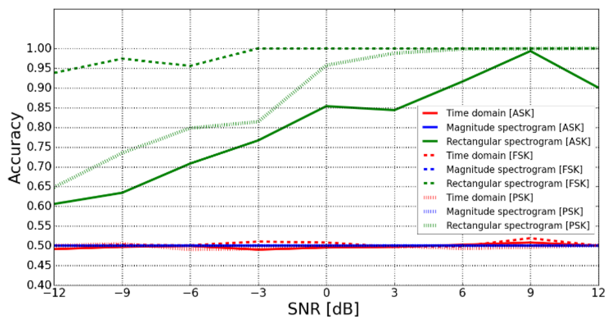


Figure 7. Accuracies of the time-series, magnitude spectrogram and rectangular spectrogram NN models for ASK, FSK and PSK signals with added ideal power signal. The SNR levels in the X-axis of the plot is discounting the power signal (i.e., this SNR=Energy of the core modulated signal/Energy of the AWGN).

IV. CONCLUSION

In this study, we compared the performance of time-series, magnitude spectrogram and rectangular spectrogram datasets in training a simple, fully connected NN. We observed that time-series and rectangular spectrogram training data performed better than magnitude spectrogram data for FSK, PSK and low-SNR ASK signals. We also observed that rectangular spectrogram training data performs significantly better than other formats when there are dominant out-of-band interferers present.

REFERENCES

[1] J. Zhou, R. Cao, J. Kang, K. Guo, and Y. Xu, "An Efficient High-Quality Medical Lesion Image Data Labeling Method Based on Active Learning," IEEE Access, vol. 8, pp. 144331–144342, 2020, doi: 10.1109/ACCESS.2020.3014355.

[2] S. Raschka, "Chapter 4: Building good training dataset," in Python machine learning : machine learning and deep learning with Python, scikit-learn, and TensorFlow., Second edition, Fully revised and Updated., Packt Publishing, 2017, pp. 109–144.

[3] M. C. Sorkun, J. M. V. A. Koelman, and S. Er, "Pushing the limits of solubility prediction via quality-oriented data selection," iScience, vol. 24, no. 1, p. 101961, Jan. 2021, doi: 10.1016/j.isci.2020.101961.

[4] J. Krzyston, R. Bhattacharjea, and A. Stark, "Complex-Valued Convolutions for Modulation Recognition using Deep Learning," in 2020 IEEE International Conference on Communications Workshops (ICC Workshops), Jun. 2020, pp. 1–6, doi: 10.1109/ICCWorkshops49005.2020.9145469.

[5] W. He, Y. Zhang, J. Ding, and L. Zhao, "A Modified Phase Cycling Method for Complex-Valued MRI Reconstruction.," Int. J. Biomed. Imaging, pp. 1–7, Nov. 2020, doi: 10.1155/2020/8846220.

[6] J. S. Dramsch, M. Lüthje, and A. N. Christensen, "Complex-valued neural networks for machine learning on non-stationary physical data," Comput. Geosci., vol. 146, p. 104643, Jan. 2021, doi: 10.1016/j.cageo.2020.104643.

[7] S. Scardapane, S. V. Vaerenbergh, A. Hussain, and A. Uncini, "Complex-Valued Neural Networks With Nonparametric Activation Functions," IEEE Trans. Emerg. Top. Comput. Intell., vol. 4, no. 2, pp. 140–150, Apr. 2020, doi: 10.1109/TETCI.2018.2872600.

[8] H. G. Zimmermann, A. Minin, and V. Kusherbaeva, "Comparison of the complex valued and real valued neural networks trained with gradient descent and random search algorithms," 2010, pp. 213–218.

[9] R. Chakraborty, Y. Xing, and S. X. Yu, "SurReal: Complex-Valued Learning as Principled Transformations on a Scaling and Rotation Manifold," IEEE Trans. Neural Netw. Learn. Syst., pp. 1–12, 2020, doi: 10.1109/TNNLS.2020.3030565.

[10] M. Amin, "Complex-Valued Neural Networks: Learning Algorithms and Applications," Ph.D. Dissertation, University of Fukui, Japan, 2012.

[11] J. A. Barrachina, C. Ren, C. Morisseau, G. Vieillard, and J.-P. Ovarlez, "Complex-Valued vs. Real-Valued Neural Networks for Classification Perspectives: An Example on Non-Circular Data," ArXiv200908340 Cs Stat, Sep. 2020. [Online]. Available from: http://arxiv.org/abs/2009.08340 [retrieved: April, 2021]

[12] C. Trabelsi et al., "Deep Complex Networks," in arXiv:1705.09792 [cs], Feb. 2018, pp. 1–19. [Online]. Available from: http://arxiv.org/abs/1705.09792 [retrieved: April, 2021]

[13] J. Shima, "Weak signal processing systems and methods," United States Patent 10879946, Dec. 29, 2020.

[14] K. Thapa, kushal-thapa/NN_complex-valued-data. 2021. https://github.com/kushal-thapa/NN_complex-valued-data.

A non-Linear MIMO-OFDM Preprocessor for non-Gaussian Channels

Danilo Pena*, Thais Areias†, Luan Pena‡, Juliano Bazzo§

Sidia Institute of Science and Technology, Manaus, Brazil
E-mail: *danilo.pena@sidia.com, †thais.areias@sidia.com, ‡luan.pena@sidia.com, §juliano.bazzo@sidia.com

***Abstract*—Multiple-Input and Multiple-Output (MIMO) and Orthogonal Frequency-Division Multiplexing (OFDM) are crucial technologies inside the 5G mobile communication systems and beyond. Design and evaluation of detector techniques over realistic channel conditions are essential in order to transmit signals at high rates and with high reliability in such technologies. In this paper, we present the evaluation of MIMO-OFDM preprocessors over non-Gaussian impulsive noise. Also, we propose a non-linear sigmoid preprocessor without a threshold parameter as an alternative to the traditional preprocessors. The simulation results show that the Symbol Error Rate (SER) performance depends on not only the preprocessors used and their thresholds but also the impulsiveness level in the noise.**

Keywords*—*Impulsive noise; sigmoid function; non-linear preprocessors; non-linear MIMO.

I. Introduction

Multiple-Input and Multiple-Output (MIMO) systems have received much attention in recent years due to the increasing demand of high transmission rates and high quality of service for wireless communications. MIMO-OFDM techniques are applied to many applications and contexts, such as 4G and 5G networks, 802.11ac, and vehicular environments [1]. With the increasing number of mobile users in the same time-frequency resource, the array efficiency of MIMO allows us to reduce the transmit power and improve energy efficiency [2]. In addition, MIMO is robust in the face of hardware imperfections, such as multiplicative phase drifts and additive distortion noise [2].

One of the greatest challenges of MIMO-OFDM systems is to detect signals with high performance and relatively low computational complexity. However, information signals are degraded by many different undesirable wireless channel effects in which noise assumptions have been demonstrated as one of the greatest challenges faced by MIMO systems. Thus, the design conception of such technology must consider realistic channel and noise models in order to represent well the current applications. On the other hand, various wireless channels have been demonstrated to suffer from impulsive noise which is more accurately characterized as non-Gaussian processes [3]. Those effects are commonly caused by man-made sources, electrical devices, ignition noise in vehicles, and bursty radio frequency emissions typical in urban environments [1].

In severe impulsive noise scenarios, the effects of the MIMO performance may be misread as low Signal-to-Noise Ratio (SNR), when in fact there is a certain impulsiveness level degrading overall system performance. Especially for classical MIMO detectors that rely on second-order statistics noise models, the Gaussian model assumption of wireless noise behavior leads to meaningful degradation or does not work well. Thus, one way to improve the performance of MIMO-OFDM systems is minimizing undesirable effects of channel and noise by designing detectors considering non-Gaussian noise. Notably, detectors have been proposed over non-Gaussian noise models with considerable improvements compared to traditional ones in those scenarios.

Several papers show non-linear preprocessors with threshold level in order to mitigate the impulsive noise in receivers [4], [5]. Performance evaluation has been done with those non-linear techniques in OFDM receivers [5], reducing adverse effects of impulsive noise. Recently, an adaptive MIMO receiver was proposed using an impulsive noise level detector. Other adaptive techniques were also presented based on Recursive Least Mean Square (RLS), adaptive Normalized Least Mean Square (NLMS), and Variable Step-size adaptive Normalized Least Mean (VSNLMS), thereby mitigating the impulsive noise effects [6]. The Support Vector Machine (SVM) has been investigated with non-linear complex Multiple Support Vector Machine regression (M-SVM) in this environment. Furthermore, a MIMO detector was proposed based on the maximum complex correntropy criterion using channel estimation to fit a parameter of its technique [7].

Those methods present improvements in detection over impulsive non-Gaussian noise as compared to traditional detectors. However, they usually have too high computational complexity making them often infeasible, due to the adding of an adaptive step or the making of a channel estimation to fit a parameter of the detector. Moreover, many detector solutions require a parameter usually based on a prior noise information. In this context, this work introduces a non-linear preprocessor based on sigmoid functions without free parameter for MIMO detector over non-Gaussian channels.

This paper is organized as follows. In Section II, we describe the MIMO-OFDM system, presenting the channel and noise model. MIMO-OFDM preprocessors are presented in Section III. In Section IV, we propose a MIMO preprocessor based on sigmoid function. In Section V, we evaluate its performance over non-Gaussian scenarios by simulations. In Section VI, we present our final remarks.

II. MIMO-OFDM System

Consider a MIMO system with N_R antennas at the receiver and N_T antennas at the transmitter, illustrated in Figure 1. The transmitter consists of MIMO-OFDM modulation over N subcarriers [5]. In the Orthogonal Frequency-Division Multiplexing (OFDM) transmitter, the bits are mapped into baseband symbols S_k using Phase Shift Key (PSK) or Quadrature Amplitude Modulation (QAM) scheme. Then, the complex baseband OFDM signal is computed by means of inverse Discrete Fourier Transform (iDFT) as:

$$s_n(t) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} S_k e^{j\frac{2\pi kt}{T_s}} \tag{1}$$

where N is the number of subcarriers, and T_s is the active symbol interval.

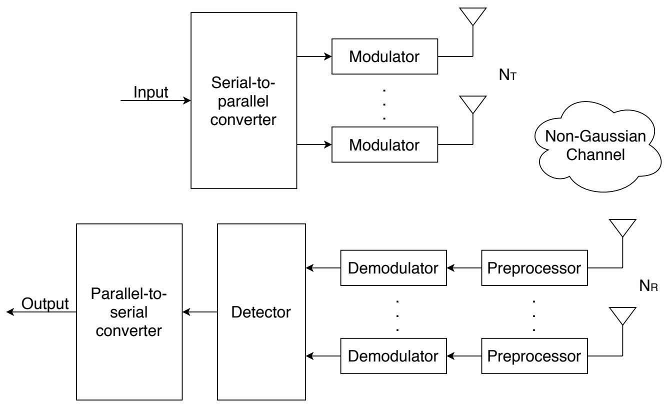


Figure 1. MIMO system architecture.

The N_R antennas are spaced such that the received signals may be considered independent of each other. The k-th symbol received by the m-th antennas is given by:

$$y_m(t) = \sum_{n=1}^{N_T} s_n(t) h_{mn}(t) p(t) + w_m(t), \tag{2}$$

where $s_n(t)$ represents the transmitted symbol from the n-th antenna, $h_{mn}(t)$ represents the channel coefficient between the n-th transmitting antenna and m-th receiving antenna, $w_m(t)$ corresponds to the channel noise, and $p(t)$ is a rectangular pulse.

The channel may be described as:

$$h_{mn}(t) = h_{mn,r}(t) + jh_{mn,q}(t), \tag{3}$$

where $h_{mn,r}(t)$ and $h_{mn,q}(t)$ are Gaussian processes with mean zero and variance equal to $1/2$. We also assume that the differences in propagation times of the signals from the transmitters to the receivers are small relative to the symbol duration.

A. Noise Model

We assume that the noise is uncorrelated, and its distribution can be represented by α-stable distributions, which are based on crucial properties such as generalized central limit theorem and stability. According to the generalized central limit theorem, if the sum of independent and identically distributed random variables with or without finite variance converges, then the limit distribution must be α-stable. Another relevant property, known as stability property, states that the sum of two independent random variables with the same characteristic exponent (α value) is also α-stable.

There are different parametrizations of α-stable distribution for different specifications of the characteristic function. We assume the parameters $\boldsymbol{\theta}_\alpha = (\alpha, \beta, \gamma, \delta)$ and the following characteristic function [8]:

$$\varphi(\omega; \boldsymbol{\theta}_\alpha) = \exp(-\gamma^\alpha |\omega|^\alpha [1 - j\Theta(\omega; \alpha, \beta)] + j\delta\omega), \tag{4}$$

with

$$\Theta = \begin{cases} \beta(\tan\frac{\pi\alpha}{2})(\text{sign}\,\omega), & \alpha \neq 1 \\ -\beta\frac{2}{\pi}(\ln|\omega|), & \alpha = 1, \end{cases} \tag{5}$$

where
α is the *characteristic exponent* such that $0 < \alpha < 2$,
β is the symmetry parameter such that $-1 \leq \beta \leq 1$,
γ is the dispersion or scale parameter such that $\gamma > 0$,
δ is the location parameter such that $-\infty < \delta < \infty$.
ω is the independent variable of the characteristic function.

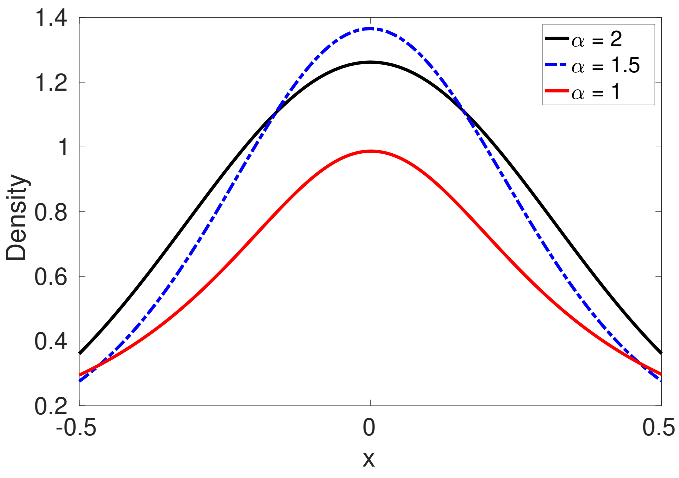


Figure 2. Probability distribution function of symmetrical α-stable with $\beta = \delta = 0$ and $\gamma = 1$.

We also assume a Symmetric α-Stable (SαS) class because it has proved to be very useful in modeling impulsive noise [3]. For such distribution class, $\beta = 0$ and $\delta = 0$ [9]. Figure 2 shows the α value variation versus the random variable representing the impulsiveness level of the distribution, where a low value of α suggests high impulsiveness and a non-Gaussian behavior, and a high value of α means that the distribution is close to the Gaussian behavior, where $\alpha = 2$ is the Gaussian case.

III. Non-Linear Preprocessors

In order to mitigate impulsive noise effects, non-linear preprocessors are applied at the receiver as illustrated in Figure 1. Those memoryless preprocessors are non-linear transformations over the signal amplitude. The most common non-linear preprocessors are blanking and clipping based on thresholds.

A. Blanking

The blanking non-linear mapping can be described as:

$$y_k = \begin{cases} r_k, & |r_k| \leq T \\ 0, & |r_k| > T \end{cases}, \quad k = 0, 1, \ldots M - 1 \tag{6}$$

where T is the blanking threshold and M is the signal length.

B. Clipping

Similar to blanking, the clipping technique maintains the amplitude when the signal is below a threshold. However, when the signal is above the threshold, then the amplitude is saturated by the threshold keeping its phase. This function can be described as:

$$y_k = \begin{cases} r_k, & |r_k| \leq T \\ T\mathrm{e}^{j\arg(r_k)}, & |r_k| > T \end{cases}, \quad k = 0, 1, \ldots M - 1 \tag{7}$$

where T is the clipping threshold and M is the signal length.

IV. Proposed Preprocessor

In this proposed technique, we compute the MIMO-OFDM using a non-linear preprocessor function based on a class of functions called sigmoid. These functions have essential characteristics as non-linear functions, such as monotonically increasing and anti-symmetry.

The non-linear functions aim to ensure the existence of higher-order statistics. The most common functions in the sigmoid family are the hyperbolic tangent functions, described as follows.

$$y_k = \tanh(r_k) = \frac{\mathrm{e}^{r_k} - \mathrm{e}^{-r_k}}{\mathrm{e}^{r_k} + \mathrm{e}^{-r_k}}. \tag{8}$$

These functions are commonly used to compute covariance by using non-linear data transformation, allowing to access information from the signal, even when it is contaminated by non-Gaussian noise [10].

V. Results and Discussions

This section presents numerical simulation results for the performance evaluation of the MIMO-OFDM system using different preprocessors. We examined the Symbol Error Rates (SER) versus the quality of signal metrics in a 2x2 MIMO system. The simulations assess the results using the Monte Carlo method with curves computed with at least 50 errors in the estimation, using 104 subcarriers and 1000 frames. All simulations consider baseband signal using Quadrature Phase Shift Keying (QPSK) modulation and unity energy with the antennas statistically independent of each other. Also, Rayleigh flat fading was assumed as the multipath propagation model in the wireless channel.

The performance metrics are usually computed versus the Signal-to-Noise Ratio (SNR). However, the infinite variance of non-Gaussian SαS processes prevents to compute the signal-to-noise ratio as a measurement of signal quality. In this work, we use the Geometric Signal-to-Noise Ratio (GSNR) [11] instead of the SNR. The GSNR is given by

$$\mathrm{GSNR} = \frac{1}{2C_g}\left(\frac{A}{S_0}\right)^2, \tag{9}$$

where the normalization constant $C_g = e^{C_e} \approx 1.78$ is the exponential of the Euler constant (C_e), used to ensure that GSNR corresponds to SNR when the channel is Gaussian ($\alpha = 2$); S_0 is the geometric power of a SαS random variable; and A is the root-mean-square value of the signal.

Figure 3 shows the performance of MIMO-OFDM receivers over non-Gaussian SαS noise with impulsiveness level of $\alpha = 1.3$ and threshold $T = 2$ for blanking and clipping preprocessors. This scenario represents an environment with high impulsiveness noise where the performance of the MIMO-OFDM system is very low compared to the Gaussian case. However, one can see the preprocessors deliver better performance than the case without the preprocessor, mainly for high GSNR values. Thus, although all preprocessors increase the performance of the MIMO-OFDM system, their performance depends on the impulsiveness level of the noise.

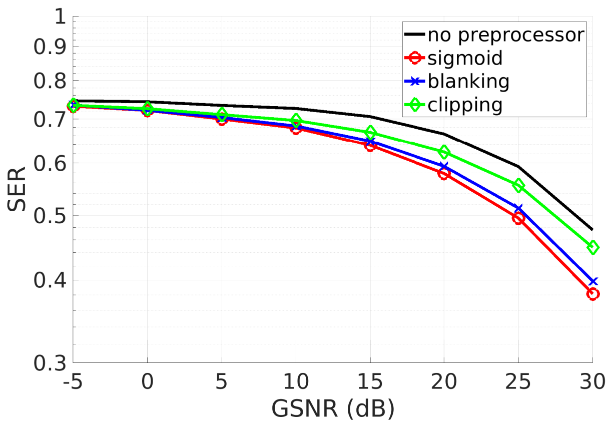


Figure 3. Performance comparison among the preprocessor techniques over SαS noise with $\alpha = 1.3$ and $T = 2$.

A. Impulsiveness Analysis

Figure 4 presents the performance of preprocessors over SαS noise model over different impulsiveness levels, i.e., many different values of α. More impulsiveness level is close to the Gaussian case ($\alpha = 2$), less is the increase in performance due to preprocessors nonlinearity over impulsive noise. This behavior occurs because in this case, the noise is less impulsive than with low values of α.

Although all preprocessors increase performance over high impulsive noise, that also depends on the threshold level used for the blanking and clipping methods.

B. Threshold Analysis

Figure 5 presents the performance of blanking and clipping preprocessors over different threshold levels. For an intermediate range of threshold values, these techniques have better

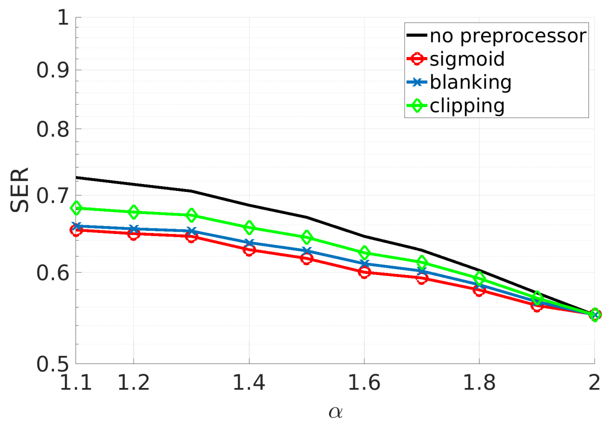


Figure 4. Performance of preprocessors over SαS noise with many different values of α, GSNR = 15 dB and $T = 2$.

performance than the sigmoid preprocessor. However, this range changes with impulsiveness level and GSNR, making this region of values difficult to be set.

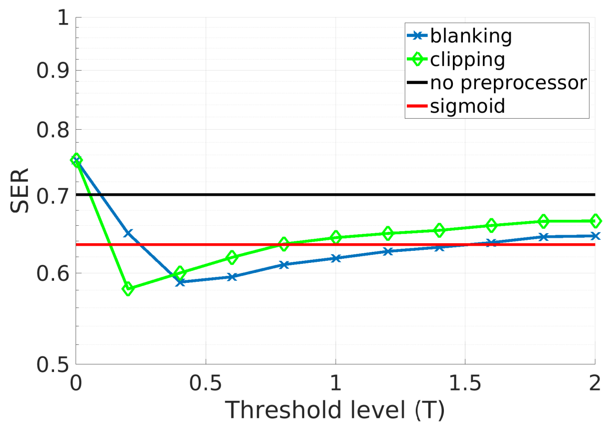


Figure 5. Performance varying threshold level of blanking and clipping preprocessors over SαS noise with $\alpha = 1.3$ and GSNR = 15 dB.

VI. Conclusions

In this paper, we evaluated traditional preprocessors in the detection of signals in MIMO-OFDM systems over non-Gaussian channels. We analyzed those preprocessors by different aspects, such as the impulsiveness level of the noise, the threshold level of the preprocessors, and the quality of the signal. Also, we presented a non-linear sigmoid function as an alternative to the classical preprocessors, comparing their performance over all aspects mentioned before. The traditional blanking and clipping preprocessors depend on the threshold level, which, in turn, also depends on the impulsiveness level in the environment. On the other hand, the sigmoid function does not have any parameters, being an alternative in the trade-off of preprocessors in the MIMO-OFDM detection systems.

Future works may use those results to investigate adaptive and machine learning solutions based on non-Gaussian noise parameters such as GSNR and α values. Thus, the preprocessors using such techniques must present a different performance, thereby being an alternative to the traditional ones.

Acknowledgment

This work was partially supported by Samsung Eletronica da Amazonia Ltda., under the auspice of the informatic law no 8.387/91.

References

[1] S. Liu, F. Yang, X. Wang, J. Song, and Z. Han, "Structured-compressed-sensing-based impulsive noise cancelation for MIMO systems," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 8, pp. 6921–6931, Aug 2017.

[2] X. Cheng, J. Sun, and S. Li, "Channel estimation for FDD multi-user massive MIMO: A variational bayesian inference-based approach," *IEEE Transactions on Wireless Communications*, vol. 16, no. 11, pp. 7590–7602, nov 2017.

[3] C. L. Nikias and M. Shao, *Signal Processing with Alpha-stable Distributions and Applications*. New York, NY, USA: Wiley-Interscience, 1995.

[4] H. A. Suraweera, C. Chai, J. Shentu, and J. Armstrong, "Analysis of impulse noise mitigation techniques for digital television systems," in *Proc. 8th International OFDM Workshop*, 2003, pp. 172–176.

[5] S. Zhidkov, "Analysis and comparison of several simple impulsive noise mitigation schemes for OFDM receivers," *IEEE Transactions on Communications*, vol. 56, no. 1, pp. 5–9, jan 2008.

[6] A. Hakam, N. A. Aly, M. Khalid, S. Jimaa, and S. Al-Araji, "Impulsive noise reduction in MIMO-OFDM systems using adaptive receiver structures," in *2013 IEEE 20th International Conference on Electronics, Circuits, and Systems (ICECS)*. IEEE, dec 2013, pp. 674–677.

[7] P. T. V. De Souza, A. I. R. Fontes, V. S. V. De Souza, and L. F. Silveira, "A novel signal detector in MIMO systems based on complex correntropy," *IEEE Access*, vol. 7, pp. 137 517–137 527, 2019.

[8] M. Shao and C. L. C. Nikias, "Signal processing with fractional lower order moments: Stable processes and their applications," *Proceedings of the IEEE*, vol. 81, no. 7, pp. 986–1010, jul 1993.

[9] G. Samorodnitsky and M. S. Taqqu, *Stable Non-Gaussian Random Processes: Stochastic Models with Infinite Variance*, ser. Stochastic Modeling Series. Taylor & Francis, 1994.

[10] D. Pena, A. Lima, V. de Sousa Jr, L. Silveira, and A. Martins, "Robust time delay estimation based on non-gaussian impulsive acoustic channel," *Journal of Communication and Information Systems*, vol. 35, no. 1, pp. 86–89, 2020.

[11] J. G. Gonzalez, J. L. Paredes, and G. R. Arce, "Zero-order statistics: A mathematical framework for the processing and characterization of very impulsive signals," *IEEE Transactions on Signal Processing*, vol. 54, no. 10, pp. 3839–3851, 2006.

Wireless Frequency Data Manipulation for Embedded Databases Used in Cybersecurity Applications

Page Heller
Endpoint Security Inc
College Station, TX, USA
email: heller@endpointsecurityinc.com

***Abstract*— A unique fingerprint in radio frequency signals provides a natural authentication for wireless edge devices in a cybersecurity application based on frequency analysis. Such fingerprints can be improved if extraneous frequency data is removed from the Fourier Transform prior to authentication, but the data manipulation must be done in real time systems with embedded databases designed to store such fingerprints. These embedded systems require a simple and fast process. A method is proposed to manipulate frequency-domain data captured from wireless signals for use in cybersecurity applications to remove unwanted features and ensure the retention of important attributes in embedded databases. Experimental measurements and field studies are presented which lead to modifications in the methodology to address unexpected features encountered. Computational efficiency is taken into account.**

Keywords-physical layer cybersecurity; wireless security; Fast Fourier Transform; radio frequency waveforms.

I. INTRODUCTION

Under consideration in this paper is a cybersecurity application which relies on analysis of the frequency content of wireless signals sent from sensors, cameras and actuators that make up what is colloquially referred to as the Internet of Things. This particular application is based on authenticating fixed wireless devices by recognizing a fingerprint in wireless signals unique to each device. The fingerprint is based, in part, on polarization mode dispersion resulting from reflections in a multipath environment [1]. This was previously considered an undesirable trait of wireless communications has become a boon to secure identification of individual transmitting devices [2]. Such dispersion is found to be stable for fixed edge devices and relatively impervious to interference and motion within the multipath. The process may be deployed in applications which use channel-hopping, as well, indicating a broad application area [3][4].

Improvements in the fingerprint may be obtained by removing frequency data that is not specific to the calculation of polarization mode dispersion, specifically, data that is outside the bandwidth of the transmitting device. This often includes side lobe data and data near zero resulting from a transformation from the time domain to frequency domain. Because this authentication must be made as received signals are being demodulated in an access point, time is of the essence to minimize latency in the data transmission. Therefore, a method is suggested to trim frequency data for such applications in a manner suitable for real time systems employing embedded databases for retaining such fingerprint data. The method is simple and efficient.

To properly address the subject matter of this paper, the following sections are offered. Section II covers background in the area of common implementations of side lobe reduction for applications that are not necessarily real time and embedded systems, so it is possible to envision how the suggested method compares. Section III discusses the theory behind the suggested method for context on why certain decisions were made. Section IV describes the precise signal processing that takes place in the method in a step-by-step manner. Section V shows results of laboratory measurements and modifications made to the method as a result. Section VI shows data from field measurements and further modifications made to the method taking these tests into account. Section VII draws conclusions from observations made in the previous sections.

II. BACKGROUND

Obtaining Radio Frequency (RF) data for the proposed fingerprinting analysis requires receiving a wireless signal, digitizing it and transforming it to the frequency domain. Here, a Discrete Fourier Transform (DFT) is being used for the transformation.

Inherent to the digitization are certain artificial artifacts of the transformation process that may affect future calculations. Primarily targeted in this paper are side lobes in the frequency response data. To remove these, many methods have been proposed for use in applications of radar, radio and ultrasound, dating back to pivotal publications in the 1960s, like Blackman’s Data Smoothing and Prediction [5]. The most common of these methods are addressed here.

A. Windowing

Windowing covers a broad area of research in RF waveform manipulation. In this process, a function described mathematically within a fixed window and is multiplied with the signal of interest while being incrementally moved, sample by sample, from one end of the signal to the other, that is, in a sliding window. More precisely, the two functions representing the signal and the window, respectively, are convolved; that is, the integral is taken of the multiplication of the two functions as in the definition of convolution given in equation (1), where f is the signal function and g is the window function which advances relative to f by frequency bin steps, τ. In this case t refers, not to time, but instead to frequency.

$$(f{*}g)(t) := \int f(\tau)g(t-\tau)\,\delta\tau \tag{1}$$

Window functions may be as simple as having a rectangular shape or as complex as a Parzen, Welch, or sine wave. One of the most common is a parabolic shape with three different versions introduced each by Hann, Hamming and Blackman [6][5]. Each has a slightly different effect on modifying the transform, particularly in the area of interest: the side lobes.

Even a simple triangular function can greatly reduce the amplitude of side lobes in RF waveforms. Applications engineers in industry have done extensive studies comparing such techniques in both the time and frequency domains [7]. These methods prove useful in operations that are not time critical and have been found appropriate for displaying the results, for instance. It is even conceivable to use such functions to precondition the waveform in the application under consideration. However, the side lobe data is still present after these convolutions and still troublesome for securely identifying RF fingerprints.

B. Discrete Wavelet Transform

A wireless signal of interest in the application of wireless cybersecurity often travels from transmitter to receiver in an industrial, commercial or residential setting. In these environments, the signal reflects off of many walls, ceilings, floors and objects on its way to the receiver. Such rich multipath channels have caused researchers to investigate other means of transforming received signals into frequency domain data. It is desired to find a transform that is perhaps less sensitive to the distortion and dispersion caused by the multipath. One such method is the Discrete Wavelet Transform, which has been shown to improve the Bit Error Rate (BER) over receivers using the DFT [8][9]. In the case under consideration, however, the dispersion caused by the multipath is of particular interest. There is, therefore, a need to preserve it across the main lobe of the resulting DFT.

C. Subcarrier Weighting

Commonly used in the popular Orthogonal Frequency Division Multiplexing (OFDM) protocols is a concept of subcarrier weighting [10]. In this process of transmitting a wireless signal, each OFDM subcarrier is weighted, that is, multiplied by a fixed or dynamic value used to reduce its impact on adjacent channels. This is commonly done using complex numbers to account for polarization of the signal.

Another similar method has been proposed, which takes computational time into consideration to produce a real time method called advanced subcarrier weighting [11]. Designed primarily for the transmission of signals rather than receiving them, the method reduces side lobe interference with signals in adjacent frequency channels. However, since it is designed for the transmission side of the data communications, it is not directly applicable to this application.

D. Ultrasound beam summation

Some of the more exotic concepts of dealing with side lobes have come from the medical industry. These application areas are typically centered upon medical imaging, where the frequency domain data is useful in detecting abnormalities or enhancing features of biological images. Methods of beamforming, again on the transmission side, have provided a rich research area for new methods and improvements. One interesting method, which could have mathematical equivalency in received signals, is interference cancellation. For instance, ultrasound beam summation employs pseudo-inverse foci with a second focus located a distance from the initial focus such that the constructive interference between the two signals cancels the side lobes [12]. Although such methods are complex and again centered on transmission, they may lead to interesting developments in received signal manipulation in the future.

E. Other Fingerprinting Methods

Many research approaches to wireless cybersecurity have centered on fingerprinting source devices as a form of authentication. Convolutional Neural Networks (CNNs) have been proposed to fingerprint radios though deep learning of certain inherent hardware features and responses [13][14]. Ongoing research has indicated, however, that wireless multipath channels introduce distortion that negatively affects the reliability of such methods [15].

The new method of cybersecurity authentication with fingerprinting, which is the method under consideration in this paper, employs such distortion for fingerprinting rather than trying to eliminate it. This new method, however, is impacted somewhat negatively by side lobes, which are formed due to the discrete nature of the transformation into the frequency domain. A new approach is sought to remove the undesired features while not creating a burden on real time system computation time.

III. Theory

In real time embedded system design for radio frequency applications, the primary criterion of concern is execution time. The time required to prepare and store data is a cost subtracted from the total time available to make decisions on actions that must be taken. Thus, the more exotic solutions, although they may produce better results, must give way to the simplest to allow time for the more critical decision-making algorithm.

In the case considered in this paper, execution time is constrained by the requirement to make a decision on the authentication of a received signal prior to the receipt of the next viable signal. A DFT transforms the signal to the frequency domain and further analysis of the dispersion leads to a correlation of the received signal with a known signal. As an example of available time to do this function, using WiFi 802.11n signals transmitted as beacons to search for other devices yields a duration for a signal of around 300 microseconds.

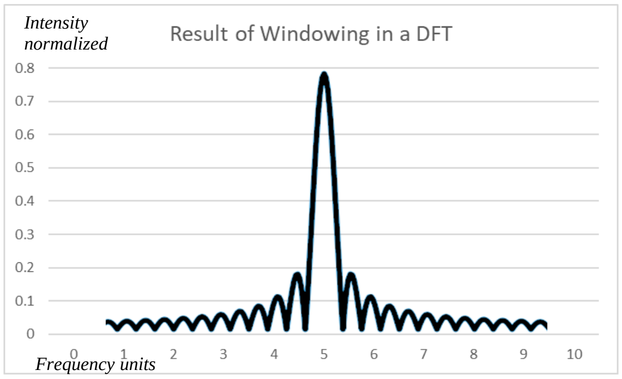


Fig. 1. Windowing data produces side lobes in the results of a DFT on sinusoidal data

By restricting the execution time for data preparation and storage, more time is available for proper authentication. Thus, it is desired to find a way to eliminate unwanted data from the frequency domain data in a short amount of time.

The shape of an ideal DFT resulting from a strong signal, for example one with a Signal-to-Noise Ratio (SNR) of 40 dB, consists of a main lobe centered on the transmitted frequency and side lobes to each side with incrementally lower intensities, such as the one depicted in Figure 1 above. The task, therefore, is to identify and store only the main lobe in an embedded database designed for such signals.

One may note that the ideal case is symmetrical. Often this fact is relied upon to store only half the amount of data, considering that it is mirrored around the center frequency. In the case presented here, however, the non-symmetry of actual frequency data in the main lobe is important to the cybersecurity application under consideration and, thus, the entire main lobe is to be saved.

While many hardware and software solutions have been proposed for reducing side lobes in wireless signal DFTs, a simpler solution would be to trim the resulting DFT to include only the frequency bins of interest. This can also be referred to as cropping, although that term comes from image analysis with the data being removed being image data. Based on trimming as a proposed simplification applied to RF signals, the question then arises of whether or not it is feasible to identify the proper points at which the DFT should be trimmed.

The feature of a DFT being discrete is inherent in the need to trim data. The transform assumes that a selected window for the DFT calculation is the exact width of one or more full cycles of the waveform of interest. However, in reality, the window often cuts a particular waveform short of a full cycle (or whole number multiple of a full cycle). The result is an introduction of side lobes in the transform, representing frequency components that are artifacts of the math rather than frequencies that actually exist in the signals of interest. This is referred to as spectral leakage.

IV. Signal Processing

To identify the main lobe with the least computational effort, one might consider using the maximum intensity of a DFT as an indication of a location near its center. From there, the edges can be sought and the width calculated. The main lobe data must then be trimmed to be placed into a database, along with like entries from other signals received.

A. Step 1: Selecting the Main Lobe

To begin, a point is identified with a high probability of being located as a frequency bin within the main lobe. That would be a point with a high intensity. Thus, a function invoked to find the maximum value in an array of floating point numbers would be appropriate. If the DFT output is an array, using a maximum value function may remove the dimension of the array of interest, replacing it with a scalar value. Look for a function argument, like ‘keepdims,’ in cases where it is desired that the resulting dimension remain consistent with the original dimensions.

In practice, abnormalities exist in the DFTs measured in applications involving WiFi signals; abnormalities which will affect the selection of this point. These will be addressed in Section IV Laboratory Measurements, as well as in Section V Field Measurements. They will require altering this step. Until then, it is sufficient to say that the initial step is to identify a point of high intensity.

B. Step 2: Finding the Edges

Using a point with a high probability of residing in the main lobe as the starting point, it is now possible to examine lower, and then higher, frequency bins to find indications of the edges of the main lobe. In this portion of the process, low intensity is of more interest than high intensity. Finding the edges of the main lobe involves finding the points on either side of the point thought to be in the main lobe where the resulting value is first near zero; that is, near zero at the point closest to the maximum.

Actual measurements are never as clean as the ideal example, however. The main lobe, in fact, is often sprinkled with many points that are near zero. In fact, “zero” can mean some value close to the noise floor of the signal.

A moving-average acts as a low pass filter to smooth the normally jagged transform and provides a more stable value with which to compare some threshold value that is set near zero. A basic form of a moving average is the uniform moving average where the current frequency bin and the prior N-1 bins are summed and divided by N. This is equivalent to multiplying each frequency bin by 1/N and summing.

Yet, a moving average is not necessary. A moving summation accomplishes the same task without requiring a division operation each time a signal is received, nor does it require handling the special case where there could be an undesirable division by zero. Using a summation only requires adjusting the threshold by the number of samples in the moving summation.

In the proposed process, where generalization is desired, the threshold is set as a multiple of the center of the first quartile of sorted and ordered frequency bin intensities. For example, when collecting 12-bit data at 20M samples per second, one might receive a signal for which the DFT covers the full bandwidth of the receiver. However, it might also be far less than the full bandwidth. In either case, the center of the first quartile of sorted intensities most often carries some value close to the noise floor. By adjusting the multiplication factor of this value, a threshold may be chosen to appropriately mark the points at which the main lobe approaches zero (or really the noise floor).

This simple algorithm compares a moving summation incrementally moving away from the maximum, first in the lower frequencies, then in the higher frequencies. It increments until it reaches the limits of the data or falls below the threshold, at which point the bin in the summation window closest to the maximum is used to mark the low and high side of the main lobe; variable names fftlo and ffthi. The “fft” refers to the fundamental function, a Fast Fourier Transform [16].

C. *Step 3: Matching the Database*

To enter the selection into a database, one must consider the size allotted for each entry. If the simple difference, ffthi-fftlo, is either greater than or less than the entry size for the database, then the data must be manipulated to fit the database. In cases where the difference is less than the size allotment in the database, the DFT data is zero-padded on the side of the highest frequency bins.

It is important to note here that future comparisons of the padded data should take into consideration that the padding is not reflective of there being no high frequency components in the stored signal, but rather that the width of the main lobe is smaller than the allotted space. Thus, the values of the fftlo and ffthi should also be stored so the data retrieved may be viewed in proper context.

It is also important to note that padding the frequency domain data is not mathematically the same as padding the time domain data for a signal. In the case of the latter, resulting side lobes in a DFT can be accentuated as the window of data analyzed is artificially shortened.

On the other hand, the DFT of the received signal may be larger than that of the stored data. If the high-to-low difference is greater than the allotted space in the database, then the data must be trimmed again. In performing this operation, the center line between the fftlo variable and the ffthi variable is used trim each the low and high ends of the available frequency bins such that the center remains the center of the trimmed data. This is done to preserve the most valuable portion of the data for later analysis or comparison. Thus, the allotted size is halved and points marked lower and higher by the length of each half are marked relative to the center line. This results in a main lobe that is of lesser width than the original.

V. Laboratory Measurements

Initially, a test was created to establish a baseline for future testing. In this case, the baseline would be defined as one with minimal multipath travel and a single, strong wireless signal to study.

A. *Setup*

An Ettus B210 Universal Software Radio Peripheral (USRP) is employed to receive two channels of synchronized data capture. This device features an Analog Devices two-channel 14-bit Analog-to-Digital Converter. Two RF Elements OARDSBX244 4 dBi Omni antennas are attached to the receive channels and placed in horizontal and vertical positions, with an angle of 90 degrees. A Netgear N600 wireless dual band router generates a periodic beacon broadcast on channel 10 (centered on 2.462GHz) to announce its presence to any listening devices.

In an outdoor setting with no measurable WiFi signals, the router is placed 10m from the antennas of the receiver. A recording is made of the electromagnetic signal with settings of 20M samples per second at a gain of 20.

The recording is used as input data to a pulse detection algorithm which extracts a single beacon in the form of a multidimensional array of complex numbers representing each sample pair for the horizontal and vertical inputs. That beacon is processed using Python’s library NumPy function fft.fftn() in an orthonormal mode, placing resulting vectors on a unit sphere. An FFT shift function is performed to place the frequency bins in order of lowest to highest along the x-axis.

B. *Results*

The resulting DFT, shown in Figure 2, produces an interesting result. The center point is not the maximum value. In fact, the main lobe dips in intensity at the center. This attribute is consistent over many tests. Thus, the algorithm was modified in Step 1 “Selecting the Main Lobe” to find the two highest points and take the center point between them to better reflect a point near the center of the main lobe. With the center point selected, the moving sums increment first forward and then backward from it to find the points at which the intensity is below the selected threshold.

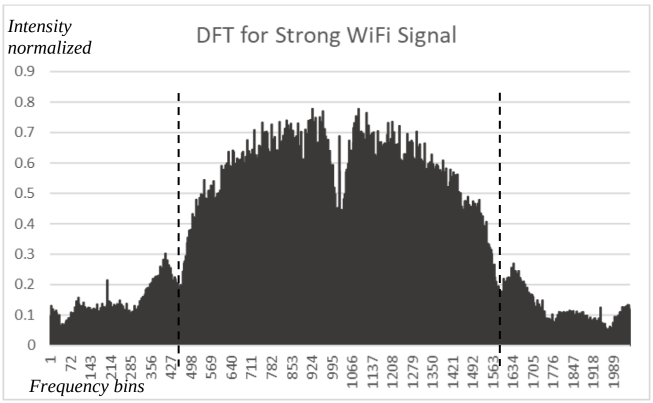


Fig. 2. A Unitized DFT for an example WiFi signal shown across frequency bins centered on 2.462GHz marking main lobe

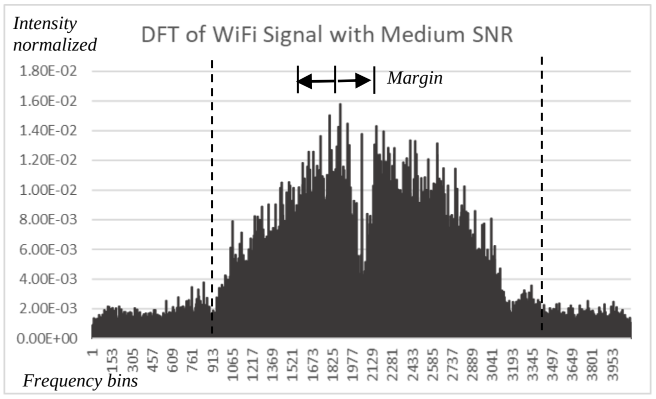


Fig. 3. A signal with lower SNR has a dip in the center approaching zero

The results of the selection process are shown in the same figure as dashed vertical lines demarking the low side and high side of the main lobe.

Moving the router farther away reduces the signal to noise ratio. The impact lessens the prominence of the main lobe making it slightly harder to identify. Nonetheless, the modification of using the two highest points continues to work. This is not obvious since the two highest points are now both to the left of center. But, since we are trying to identify a point with high probability of being in the main lobe, this point will suffice. One near the center would be more certain, but in this case, we are still well within the main lobe. The iteration to the higher frequencies will simply take longer than the iteration to the lower frequencies.

Now, however, it may be seen that the dip in the center is approaching zero. This could cause the method to select the center as one of the edges of the main lobe.

To compensate, the moving summations above and below the selected point begin on either side of a fixed margin around the center, in which we do nothing. Now, when incrementing to higher frequencies, the iterations will begin, not at the selected point, but at a fixed margin away from it, spanning the low center of the DFT.

This is the second modification made to Step 1. Figure 3 displays a signal with lower SNR and the resulting low and high points selected. Note in this image that a side lobe on the high frequency side of the main lobe was included in the selection erroneously. A minor adjustment lowering the multiplication factor on the threshold would resolve this.

VI. FIELD MEASUREMENTS

A. Setup

A site was selected with an indoor space approximating an industrial setting. A barn home was used with cross beams, fixtures in the space and miscellaneous objects which would produce a rich multipath channel for the RF signals to traverse. A barn home is one that is shaped like a barn, featuring a very large living space that has a 12x18 meter floor space and 12 meters ceiling. A USRP was placed along one wall of the facility.

The same router was placed at a point 9 meters in front of the receiving antennas and four Raspberry Pi 3B microcontrollers were placed on an arc 12 meters away from the receiving antennas and on positions relative to the plain of the two orthogonal antennas at angles of 60, 80, 100 and 120 degrees. The Pi's were programmed to connect with the router. Data was captured using the same parameters as the laboratory measurements.

All electrical devices in the house were shut down and interference studies were conducted with electrical fans and a microwave. The tests which follow were shown to have no electrical interference and no stray transmitters.

B. Results

In this environment, the SNR of some of the signals was much lower than those of the previous tests. The shape of the DFTs were also found to vary. The relative signal strength of received signals was not always as strong as those in the laboratory measurements. Their shape may be even less predictable than the lowest laboratory measurements, as may be seen in Figure 4, depicting the DFT for a WiFi signal with low SNR. It is much more difficult to see the main lobe, which extends over what seem to be two lobes rather than one.

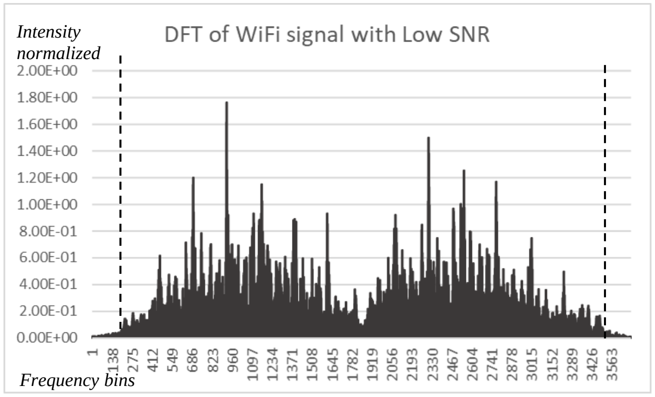


Fig. 4. DFT with low SNR causes the main lobe to be harder to determine

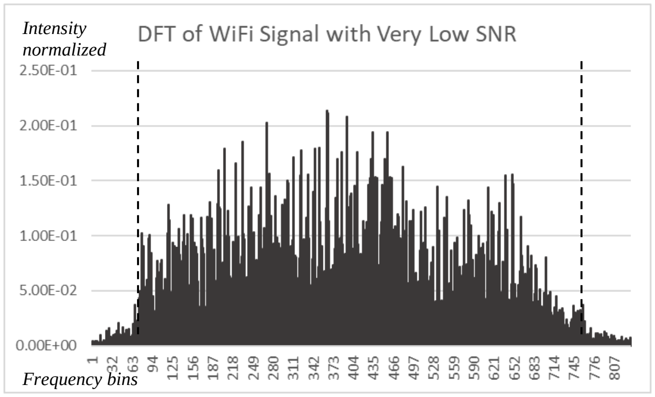


Fig. 5. When the SNR is low the dip in the center can be missing

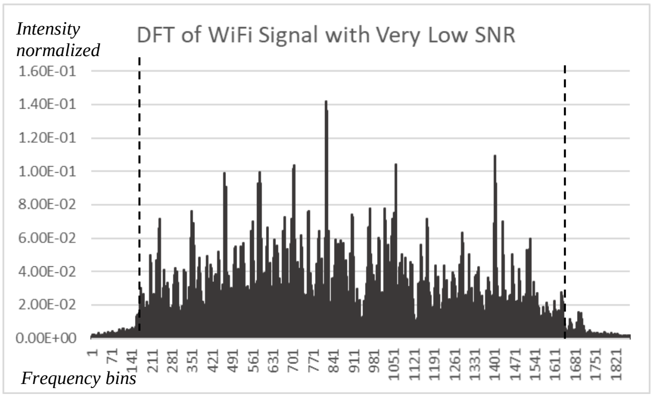


Fig. 6. Even lower SNR lessens the definition of the main lobe

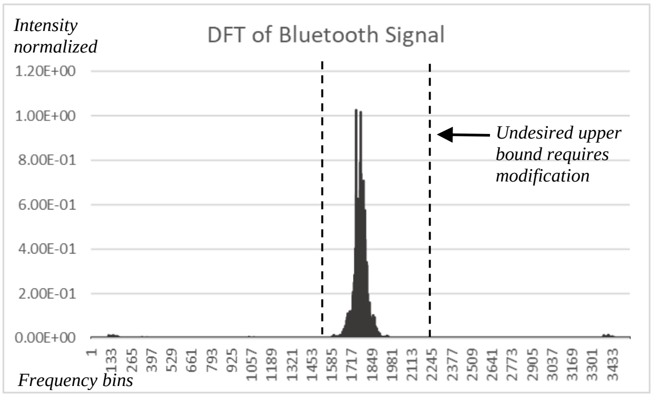


Fig. 7. A DFT of a Bluetooth signal is narrow with sidelobes farther away from the main lobe (before correction)

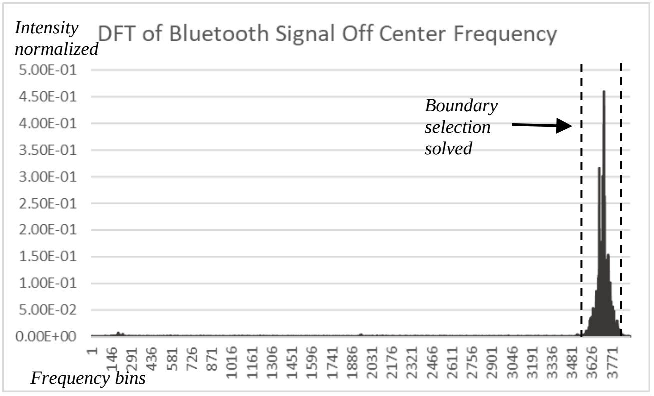


Fig. 8. A DFT of a Bluetooth signal centered on a different channel

On occasion, the dip seen in the previous tests was not present, as may be seen in Figures 5 and 6. Nevertheless, the algorithm for identifying the main lobe worked as modified.

A Bluetooth signal from one of the Pi's was recorded and analyzed. This signal is much more narrow then the previous WiFi signals and thus, the modification to have a fixed margin around the center that is not part of the calculations, along with the width of the window for a moving sum, each failed to accurately mark the low and high sides of this main lobe, as may be seen in Figure 7.

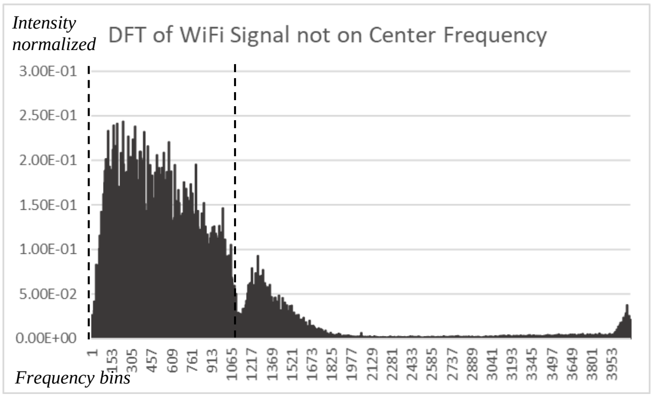


Fig.9. A DFT showing a WiFi signal captured from an adjacent channel

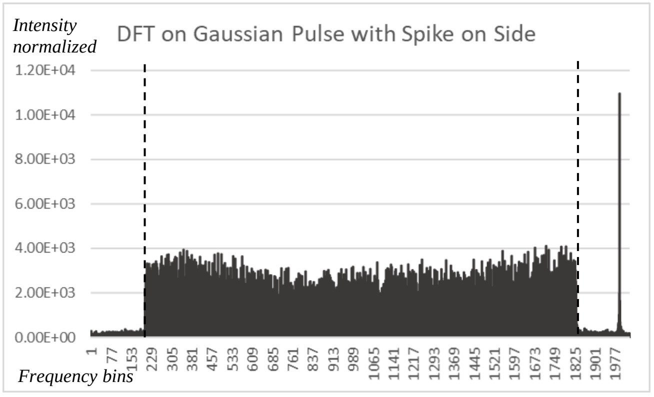


Fig. 10. DFT of a Gaussian pulse has a high intensity artifact that must not be counted as a peak intensity

Bluetooth signals use a channel-hopping protocol and, thus, may appear in several locations across a fixed 20MHz receiver bandwidth. Figure 8 is an example of one that is not centered on the same channel.

In the particular cybersecurity application under consideration for use with this proposed process, the inclusion of data near zero causes problems in later analysis and, thus, is undesirable. As a result, narrow signals, like that depicted in Figures 7 and 8, are made exempt from these two interfering constraints. That is, the fixed margin to begin the sliding summation is not enforced and the window for the moving summation itself is removed, as seen in Figure 8. This is the third modification to Step 1 "Selecting the Main Lobe." The binary parameter marking a narrow band signal is stored along with the fftlo and ffthi parameters.

It should be noted that some signals may not be centered on the appropriate channel frequency, but may still be desirable to process. Such is a case depicted in Figure 9. Here, the signal is centered near the limit of the bandwidth of the receiver, which is approximately 20 MHz. The algorithm continues to perform appropriately in such cases with one boundary being selected as either zero or the maximum number of frequency bins, depending on which side of the desired channel it falls.

Lastly, the DFTs of some WiFi signals were seen to contain one point away from the main lobe with a very high intensity. This typically occurred near an extreme of the receiver's bandwidth, likely from a dominant frequency in the noise floor, or as an artifact of the transform calculations. In either event, the errant point should not be used in the selection of the main lobe, as it is not in the main lobe. Figure 10 is an example of a case where there is a single point with a very high intensity. This is not a signal, per se, but rather a pulse of Gaussian energy like one may see emitting from a radar.

The fourth and final modification to Step 1, as a result, is to not use the first and second highest peaks, but instead use the second and third highest peaks to find a point with high certainty of being a part of the main lobe. Figure 8 depicts such a case, where the second and third peaks were used to find the boundaries of the main lobe efficiently.

With these four modifications made, approximately three thousand signals were studied and marked with selection of the main lobe for storage and retrieval from an embedded database. Only three were improperly marked and all three were signals with SNR less than 3 dB. The markings on all three truncated the main lobe on one side of the center point.

VII. Conclusion

Real time devices requiring use of an embedded database in RF signal applications must be efficient in terms of computing time. In cases where the frequency content is of particular interest, efficiency may be gained by carefully selecting only the main lobe resulting from discrete Fourier transforms and eliminating the side lobes and extraneous data on high and low sides.

Selecting the main lobe, however, can be hindered by several common abnormalities seen in the transforms of wireless signals; abnormalities like a dip in the center of the main lobe, the dip nearing the noise floor, and a single frequency outside the lobe with a very high intensity. In addition, some signals may be narrow band, requiring alternate handling.

Methods presented in this paper have been tested with a large number of varying cases and have shown to produce good results in selecting the proper main lobe information to offer efficient data storage and retrieval for further computation.

Acknowledgments

The author extends his thanks to Dr. Thomas G. Pratt of the University of Notre Dame for his encouragement and experience in the use of the experimental settings and measurement equipment and also to Jay Labhart, Chief Technology Officer for Endpoint Security, for his knowledge of field conditions and assistance in the actual tests.

References

[1] T. G. Pratt and R. D. Kossler, "Input-to-Output Instantaneous Poalrizaton Characterization," IEEE Transactions on Antennas and Propogation, Vol. 67, No. 3, pp. 1804-1818, March 2019.

[2] R. P. Heller, T. G. Pratt, J. Loof and E. Jesse, "RF Biometric for Wireless Devices," In: Arai K., Bhatia R., Kapoor S. (eds) Proceedings of the Future Technologies Conference (FTC) 2018. FTC 2018. Advances in Intelligent Systems and Computing, vol 881. Springer Nature Switzerland AG, Cham. https://doi.org/10.1007/978-3-030-02683-7_65, October 2018.

[3] J. Loof and T. G. Pratt, "Frequency-Hopped Signal Source Identification in Frequency-Selective Channels," IEEE Transactions on Aerospace and Electronic Systems, Vol. 55, Issue 6, pp. 3316-3329, December 2019.

[4] J. Loof and T. G. Pratt, "Unsupervised Classification of Frequency-Hopped Signals in Frequency-Selective Channels," Resilience Week, Denver Colorado, Best Cybersecurity Technology Paper Award, pp. 108-113, IEEE Cat. No. CFP18B24-POD, ISBN 978-1-5386-6914-3, August 2018.

[5] R. B. Blackman, "Linear Data-Smoothing and Prediction in Theory and Practice," January 1, 1965, Addison Wesley, 1st Edition, ISBN-10: 0201006103.

[6] F. J. Harris, "On the use of Windows for Harmonic Analysis with the Discrete Fourier Transform," Proceedings of the IEEE 66(1): 51-83 doi/10.1109/PROC.1978.10837, January 1978.

[7] J. Carnes, "Windowing High-Resolution ADC Data-PartI," EETimes, Designlines, Frbruary 4, 2009

[8] S. V. Moholkar, "BER Performance for FFT and Wavelet Based OFDM Systems over AWGN Channel," International Journal of Research and Scientific Innovation, Vol II, Issue VIII, pp. 52-54, ISSN 2321-2705, August 2015.

[9] A. N. Akansu and X. Lin, "A comparative performance evaluation of DMT (OFDM) and DWMT (DSBMT) based DSL communications systems for single and multitone interference," Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP '98 (Cat. No.98CH36181), Seattle, WA, USA, pp. 3269-3272 vol.6, doi: 10.1109/ICASSP.1998.679562, 1998.

[10] I. Cosovic, S. Brandes and M. Schnell, "Subcarrier Weighting-A Method for Sidelobe Suppression in OFDM Systems, IEEE Communications Letters 10(6); pp. 444-446, DOI: 10.1109/LCOMM 2006.1638610, June 2006.

[11] A. Selim and L. Doyle, "Real-time sidelobe suppression for OFDM systems using advanced subcarrier weighting," IEEE Wireless Communications and Networking Conference, pp. 4043-47. 10.1109/WCNC.2013.6555224, 2013.

[12] A. Ilovitsh, T. Ilovitsh and K.W. Ferrara, "Multiplexed Ultrasound Beam Summation for Side Lobe Reduction," Nature, Scientific Reports 9, article 13961, https://doi.org/10.1038/s41598-019-50317-7, September 27 2019.

[13] K. Sankhe et al., "ORACLE: Optimized Radio clAssification through Convolutional neuraL nEtworks," in IEEE INFOCOM 2019-IEEE Conference on Computer Communications. IEEE, pp. 370–378, 2019.

[14] S. Riyaz, K. Sankhe, S. Ioannidis, and K. Chowdhury, "Deep Learning Convolutional Neural Networks for Radio Identification," IEEE Communications Magazine, vol. 56, no. 9, pp. 146–152, Sept 2018.

[15] A. Al-Shawabka et al., "Exposing the Fingerprint: Dissecting the Impact of the Wireless Channel on Radio Fingerprinting," IEEE INFOCOM 2020 - IEEE Conference on Computer Communications, Toronto, ON, Canada, pp. 646-655, doi: 10.1109/INFOCOM41043.2020.9155259, 2020.

[16] M. T. Heidman, D. H. Burrus and C. S. Sidney, "Gauss and the History of the Fast Fourier Transform," IEEE ASSP Magazine, Vol 1, Issue 4, pp. 14-21, October 1984.