A New Emulation Platform for Real-time Machine Learning in Substance Use Data Streams

Stefan A Bruendl, Hua Fang, Hieu Ngo, Edward W. Boyer, Honggang Wang
Computer and Information Science[123]*, Electrical and Computer Engineering*[5]*. University of Massachusetts Dartmouth*
North Dartmouth, MA, USA
Population and Quantitative Health Sciences[2]*, University of Massachusetts Medical School*
Worcester, MA, USA
Emergency Medicine[4]*, Brigham and Women's Hospital, Harvard Medical School*
Boston, MA, USA
sbruendl[1], hfang2[2], hngo1[3], hwang1[5]@umassd.edu, EBOYER@bwh.harvard.edu[4]

Abstract—**With 5G networks on the rise, it becomes more and more important to grant researchers access to tools that allow for development and experimentation in the field of 5G transmission. Healthcare can benefit greatly from these developments. In this paper a real-time transmission technique is described and tested that, if implemented, allows wearable devices to transmit multiple streams of data on various frequencies. These tests will be used to explain how this presented platform works, what drawbacks and benefits exist with the proposed scheme, and how to further develop the solution of real-time transmission of sensitive data, such as substance-use data, at higher frequencies.**

Keywords—Machine Learning, Real-time transmission, signal processing, 5G mobile communication, mmWave Technologies

I. Introduction

Hardware and software are both improving at steady rates with 5G networks on the rise, while faster machine learning algorithms are being developed. This rapid development increases interest in research and therefore the need for testing platforms and software environments. There are hardly enough tools, if any, that combine software and hardware, especially in situations where real-time machine learning algorithms are needed for examining data streams. This paper introduces a new pipeline for real-time machine learning algorithms coupled with emulating the transmission of multiple data streams concerning substance use data. The data streams used in this paper concerns substance use, such as surrogates of sympathetic nervous system excitement, were collected by multiple sensors recording data every 0.5 seconds [10]. Currently, these real-time data streams were collected by a wristband with integrated biosensors, saved on the server, and analyzed offline for detection of substance use events. The proposed pipeline presented in this paper emulates a scenario in which these data streams are collected, transmitted, and analyzed all within real-time. This pipeline uses GNU Radio and Universal Software Radio Peripherals (USRPs) for real time transmission of multiple data streams and real time peak detection algorithms for analysis and detection of events. This pipeline is tested at different frequencies to emulate the benefits of 5G networks and the effects of increasing frequency of transmission on accuracy, speed and safety. Furthermore, this paper explains the setup of this pipeline in order to make it accessible to any researcher, even those unfamiliar with code or transmission schemes. A major part of this pipeline is GNU Radio which is a free, open source tool that is easy to install, easily accessible to anyone, and allows for emulation and simulation of transmission schemes. This tool combined with peak detection algorithms allows researchers to then recover and analyze multiple data streams without errors in real-time at various frequencies and allows for further development in either one of those fields of research, be it to improve existing machine learning algorithms, develop new transmission schemes at different frequencies, or test real-time detection of substance use events in individuals.

II. An Overview of GNU Radio

The development tool GNU Radio is a free open-source program that allows communication with a variety of Software Defined Radios (SDRs) while allowing users to modify and adjust SDR hardware. GNU Radio is based on "flowgraphs" which contain "blocks". The standard structure of any GNU Radio program consists of a flowgraph with sink/source blocks and manipulation blocks in between. As described below, GNU Radio can be used by anyone, even those unfamiliar with computer engineering and science. For medical researchers analyzing data and developing new tools, this software is incredibly useful as it allows for development of new schemes without any prior knowledge of the field. It is therefore an invaluable tool for researchers aiming to create real-time event detection algorithms when considering single/multiple stream data in substance use areas. On top of that, GNU Radio allows for adjusting frequency in order to emulate 5G networks.

A. Basic GNU Radio Blocks

1) Source and Sink Blocks: Source blocks and Sink blocks are the main way of loading and saving data with GNU Radio. Sources are most often signals, waves or files that are read in from a file. Sinks can save, graph, or transmit these files depending on the type of block used. In the case of substance use data, multiple streams may be combined as one .csv file that can be read as a Source File, transmitted, and saved as a Sink file in .csv format.

2) Mod and Demod Blocks: To convert a file to a transmission ready format, a wave, *Mod* blocks are used on to change a file, which is read as a byte stream, to a complex modulated signal. Each *Mod* block in GNU Radio has a corresponding *Demod* block, to demodulate a received signal after a successful transmission to recover the original file.

*3) Variables**:** Variable* Blocks are similar to variables in code; reusable definitions and terms that are implemented in several parts of a program, or in this case flowgraph. *Variable*s have an *ID*, by which they are referenced and a *Value*.

B. Using GNU Radio for Emulation

GNU Radio is highly recommended when simulating with SDRs, but it may also be used as an emulation tool by running scripts virtually. To do this, the *Sink* and *Source* blocks can be substituted by *Virtual Sink* and *Virtual Source* blocks respectively. This will enable emulated communication without purchasing any radio equipment. This is a powerful alternative for researchers that would prefer to obtain hands-on experience with this tool and potentially write scripts for it before purchasing and testing on actual hardware.

III. Set up an environment for transmitting multi-stream substance use data in real time

There are three major components to this emulation study, the GNU Radio transmission simulation on real USRP machines, the file manipulator that splits input files into various smaller input files, and the event detection algorithms.

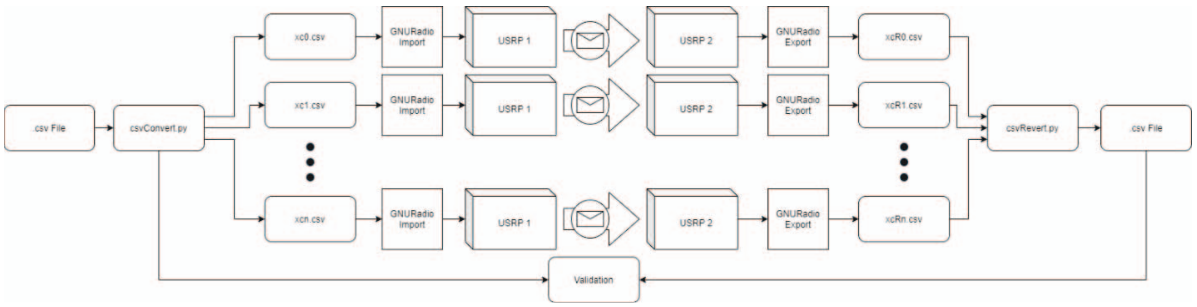
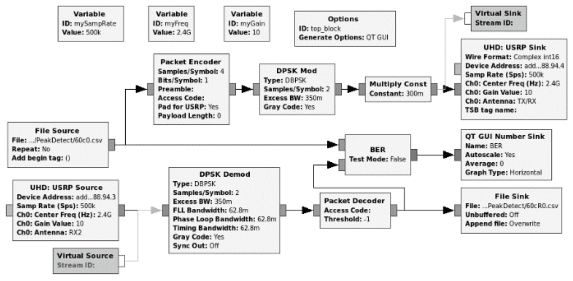


Fig. 1. A diagram showcasing the path a CSV file undertakes in this workflow.

The second component is run prior to transmission in order to split a file into chunks that are small enough to be transmitted all at once on USRP machines. Once all chunks of the original file are transmitted in series, another script is run to check for errors and merge all pieces back together into a copy of the original file. These scripts allow for errorless transmission of multi-stream data. This workflow is depicted in Figure 1 above. The following sections will describe the contents of each script and the GNU Radio flowgraph, alongside information on how to set up the USRP machines correctly.

A. GNU Radio Environment for multi-stream data

1) GNU Radio Flowgraph for errorless real-time transmission of multi-stream data: Using USRP Hardware Drivers (UHD), the Variable Blocks at the top of Figure 2 are used to modify the *UHD: USRP Source* and *UHD: USRP Sink* blocks. They modify the sample rate, frequency and gain on these blocks. Stream chunks are read in the *File Source* block in .csv format. The *Packet Encoder* and *Packet Decoder* Blocks then split and merge the file into smaller chunks splitting one byte stream into multiple. These are followed and preceded by the Differential Phase Shift Keying (DPSK) blocks, *DPSK Mod* and *DPSK Demod,* that work to modulate these chunks into and from complex waves that can be transmitted. If no USRP devices are available, utilizing *Virtual Sink* and *Virtual Source* blocks for transmission (dark grey in Figure 2) allows for software-based simulation. Once packets are transmitted, received, demoded, and decoded they are sent to a *File Sink* for saving and analysis and compared with the original *File Source* using a *BER* block to allow Bit error rate analysis per transmission.



Variable
ID: mySampRate
Value: 500k

Variable
ID: myFreq
Value: 2.4G

Variable
ID: myGain
Value: 10

Options
ID: top_block
Generate Options: QT GUI

Virtual Sink
Stream ID:

Packet Encoder
Samples/Symbol: 4
Bits/Symbol: 1
Preamble:
Access Code:
Pad for USRP: Yes
Payload Length: 0

DPSK Mod
Type: DBPSK
Samples/Symbol: 2
Excess BW: 350m
Gray Code: Yes

Multiply Const
Constant: 300m

UHD: USRP Sink
Wire Format: Complex int16
Device Address: add...88.94.4
Samp Rate (Sps): 500k
Ch0: Center Freq (Hz): 2.4G
Ch0: Gain Value: 10
Ch0: Antenna: TX/RX
TSB tag name:

File Source
File: .../PeakDetect/60c0.csv
Repeat: No
Add begin tag: ()

BER
Test Mode: False

QT GUI Number Sink
Name: BER
Autoscale: Yes
Average: 0
Graph Type: Horizontal

UHD: USRP Source
Device Address: add...88.94.3
Samp Rate (Sps): 500k
Ch0: Center Freq (Hz): 2.4G
Ch0: Gain Value: 10
Ch0: Antenna: RX2

DPSK Demod
Type: DBPSK
Samples/Symbol: 2
Excess BW: 350m
FLL Bandwidth: 62.8m
Phase Loop Bandwidth: 62.8m
Timing Bandwidth: 62.8m
Gray Code: Yes
Sync Out: Off

Packet Decoder
Access Code:
Threshold: -1

File Sink
File: ...PeakDetect/60cR0.csv
Unbuffered: Off
Append file: Overwrite

Virtual Source
Stream ID:

Fig. 2. The GNU Radio flowgraph diagram used for this paper

2) USRP setup to emulate transmission of multi-stream data from sensors to a processing unit: For the purposes of this paper 2 USRP N210 machines were used with one SBX daughterboard onboard of each. which allows the USRP devices to transmit within a frequency range of 400MHz – 4.4GHz. Coupled with one log periodic antenna per USRP with a frequency range of 850MHz – 6.5GHz, the devices can transmit and receive at a range of 850MHz – 4.4GHz. Each USRP is connected to a 1Gbps Gigabit ethernet switch to communicate packets with a computer Packet flow is described in Figure 3. Diagrams for this dataflow and physical setup can be found in Figure 3 and Figure 4 below.

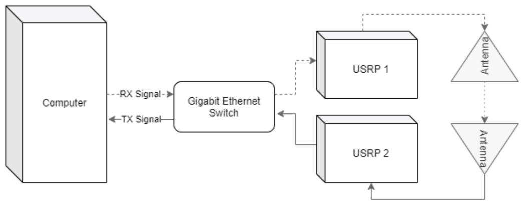


Fig. 3. A diagram of the physical USRP setup and visualization of the dataflow of a signal.

Fig. 4. A picture of the physical USRP setup

B. CSV Packet Manipulation – Re-organizing multiple data-streams into chunks for transmission

Due to GNU Radio packet size constraints, sending files larger than a certain threshold *w* is nearly impossible to do at once. For this paper, a packet splitting conversion and reverse conversion workflow is introduced and explained below.

1) Conversion and Multiplication of data: Prior to executing the GNU Radio programs for transmission, a script is run to take an input file *S*, repeat every observation *D* times and create chunks *T* for transmission. Increasing the rate of repetition *D* increases the file size, but also increases accuracy as there are more data to consider for validation and merging after transmission. The algorithm takes *mL-2p* number of rows (where *mL* is defined in Eq. (2) of all duplicates and orders them in various chunks *T* which is padded with *p* amount of 0s at the start and end of each chunk. These 0s are added to pad the file as USRP machines may potentially omit certain lines at the start or end of files during simulated and real transmission. Each chunk *T* is sent to GNU Radio individually, where it is transmitted by one USRP and received by another.

$$S \to T \mid T_{i_j} = \begin{cases} S_{\left\lfloor \frac{mL*i+j-p}{D} \right\rfloor}, & if\ j \geq p\ and\ j < (mL-p) \\ 0, & otherwise \end{cases}$$
$$\forall i \in \left[0 \ldots \left\lfloor \frac{D*oL}{mL} \right\rfloor\right], j \in [0 \ldots mL] \quad (1)$$

Symbol	Description
S	Source file
T	Fragment/Chunk file
R	Recovered file
mL	Maximum length per Fragment File T (rows)
oL	Length of the original file S (seconds of data/2)
p	Padding at the start and end of each fragment T
D	Number of Duplicate datapoints of S in T
w	Upper limit of total datapoints per fragment T
c	Amount of data streams in file S
i	Index of Fragment File T
j	Column/observation within Fragment File T
b	Time required per script per observation
y	Gaussian filter result per datapoint t
x	The value of the observation at time t
σ	Standard deviation of Gaussian distribution
t	Time of observation/column (similar to j)
ϖ	Gaussian function for one dimension
g	Gaussian function to compute Smooth Vectors
n	Smooth Width for peak detection algorithm
v	Preliminary set of Smooth Vectors
m	Final set of Smooth Vectors
h	Height of observation after applying smoothing
A	Amplitude threshold for peak Detection
d	Derivative of smoothed observation
a	Amplitude threshold for Derivative analysis

TABLE I
VARIABLES USED IN CONVERSION, REVERSE CONVERSION, PEAK DETECTION AND ANALYSIS

Eq. (1) ensures that *j* files of *T* are filled with information from *S*. The first and last *p* numbers within file *T* are used as padding. There shall be $\left\lfloor \frac{D*oL}{mL} \right\rfloor$ amount of chunks *T* created. This depends on how many chunks of size *mL* can fit into the total amount of datapoints (*oL*) from *S* after multiplying the amount by *D*. Ignoring the padding p and considering the chunk of File *i* the equation is currently on, *D* identical copies from *S* are placed in *D* different spots of T_j until the second padding half is reached (*mL-p*). The number of chunks *T* increases as more recordings are taken. The parameter *p* allows for scaling of padding; *mL* allows for the maximum size of each chunk, including *2p*; *oL* is defined by the size of the original file *S*; *D* determines the number of duplicates, if *D* is adjusted so should *mL*.

In the case of the above mentioned flowgraph and USRP setup, a limit of ~2475 data entries (*w*) was observed which was obtained by multiplying the amount of data streams (*c*) with the number of recordings in time prior to the terminating character ("X" for USRP transmission). With this limit *w* an optimal *mL* may be found using Eq. (2)

$$mL = \frac{\frac{w}{c} - 2p}{D} \quad (2)$$

$\left\lfloor \frac{D*oL}{mL} \right\rfloor$ then signifies how many fragments are created depending on the *D, oL,* and *mL* per fragment. The graph depicted in Figure 5 demonstrates how *mL* is affected by changing input parameters *D* and *c* while considering a word limit *w* of 2475 and a static padding *p* of 50 rows at the start and end of each file.

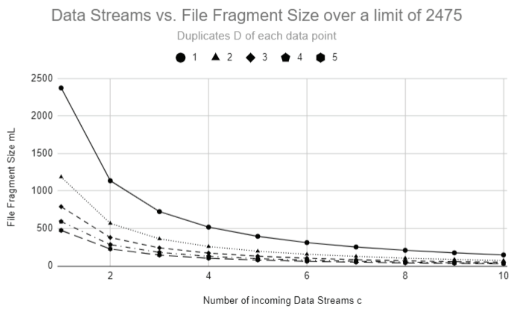


Fig. 5. A graph showcasing the reduction of observations per fragment as data streams increase per duplicate

2) Reverse-conversion and Mode of data: After transmitting and receiving each fragment T of the original file, they are merged back together using a separate script. If there is a substantial amount of errors in a chunk, the script will alert users and prompt a re-transmission of that chunk. If instead there are only numerical errors caused by random noise, taking the Mode of all D transmitted values will still eliminate the error and result in the true value of that reading.

$$T \rightarrow R \mid R_{\frac{mL*i+j-p}{D}} = Mode\left(T_{i_j} \ldots T_{i_{j+D-1}}\right)$$
$$\forall i \in \left[0 \ldots \left\lfloor \frac{D*oL}{mL} \right\rfloor\right], j \in [0D+p, 1D+p, 2D+p \ldots mL-p]$$
(3)

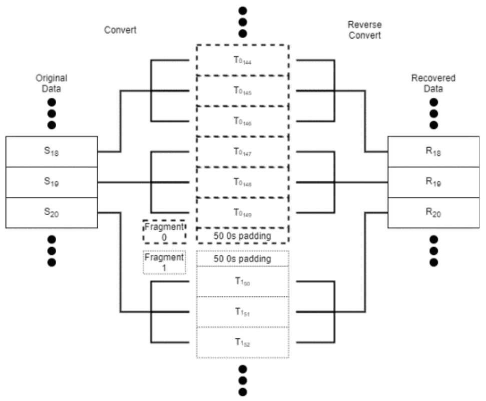


Fig. 6. A diagram showcasing the distribution of observations across files from the source file *S* to the individual fragments *T* to the recovered file *R*

In the case of the reverse-conversion function, the files use an operator that takes D amount of numbers to take a mode of. Padding may be cut off irregularly and must be removed on a chunk-by-chunk basis in order to only consider real values. The resulting file R will have the same size and content as the original file S, allowing for real-time, 1-to-1 recovery of multi-stream sensor data. Figure 6 sums up this process from converting all data streams from source S to multiple chunks T with D=3 and back to, creating a full circle of data processing. Algorithm 1 below showcases the entire transmission and recovery workflow.

Algorithm 1 Transmission workflow

Input: Source file S, length of source file S: oL, amount of input streams c, duplication factor D, file padding p, word limit w.

```
1: Use Eq. (2) to obtain a value for mL
2: Use Eq. (1) to map S→T
3: do
4:     Use Eq. (3) to map {$T_0$ ... $T_{\lfloor \frac{D*oL}{mL} \rfloor}$}→R
5:     if returns no errors
6:         return R
7:     else
8:         Request re-transmission of erroneous file
9:         Overwrite erroneous file with re-transmission
10:    end if
11: while Eq. (3) returns errors
```

IV. Testing Methods

A. Transmitting larger files and multiple streams of data

There are two ways to adjust for larger file sizes: Either allow less rows mL to be included in a fragment T or decrease D to lower the number of duplicates sent per entry at the potential loss of accuracy. To find the best values of each, Eq. (2) and its outcome, Figure 5 are used given certain input data streams and a cumulative time period of recordings to find the best mL as c and D increase.

B. Interference and Frequency Testing

Signal to Noise Ratio (SNR) and Bit Error Rate (BER) of transmissions are common metrics in wireless communication testing. GNU Radio includes a BER block which is implemented in the flowgraph in Figure 2. BER is noted and averaged across all transmissions of chunks to result in average BER per frequency (Figure 7). BER averages will be computed in Non-line of sight transmission (NLoS) and Line of Sight (LoS) environments respectively at each frequency. In order to obtain a percentage based BER, the obtained log 10 form from GNU Radio is taken as an exponent to 10 (10^BER = %BER). BER is recorded at various frequencies. The necessity to switch to higher frequencies for the emerging age of 5G is real and will require testing for usable up-to-date research results. It is not directly possible to reach 5G frequencies such as 60GHz with the SBX daughterboards used in this paper. There are methods and equipment described in [1] that allow for this to occur, though it was not possible to acquire these resources for the purpose of these tests. From results in BER gathered at various frequencies, 5G results may, however, be interpolated to show how higher frequencies may affect the results of transmission.

C. Validation Testing of Substance use Data

File S and R may be compared to validate the claim of this paper of 1-to1 transmission. There are two forms of validation and processing chosen for this project, both of which are detailed further below.

1) Excel Validation of recovered sensor data: Validation with this method includes loading *S* and *R* into the same excel file and comparing each entry *j* on a row by row basis for each *c*. Performing this validation is done by listing each row side by side and executing the *"=EXACT(COL1, COL2)"* command to observe identical entries. Comparing the amount of *TRUE* vs. *FALSE* values results in a precise depiction of the accuracy and errors of the used transmission scheme and script modifiers.

2) Algorithm Validation Example of Cocaine Data: Once the data is validated to be identical, or in the case in which it is not possible to do so, it is useful to observe whether the results with the original file correspond with the results of the file after transmission. Peak detection is used to check whether results in *S* correspond to results in *R*.

For peak detection algorithms, we use two different algorithms. The first one, Local Maximum Search (LMS), is an algorithm that looks for a point that is larger than its adjacent points. Specifically, in this case, we use the Gaussian filter to smooth the data, linear interpolation for baseline correction of the data, and finally use local maximum for finding the peak points. The Gaussian filter is calculated as

$$y(t) = x(t) \times \mathfrak{w}(t) \tag{4}$$

Where $x(t)$ is the signal at time t and

$$\mathfrak{w}(t) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{t^2}{2\sigma^2}} \tag{5}$$

For linear interpolation, we correct the baseline by dividing the data into small segments and using the median of each segment as the baseline point. Finally, the peak is determined if it is a local maximum of N neighboring points.
The second algorithm is The Smoothed Z-score Peak Detection Algorithm (SPDA) [6], based on the theoretical normal distribution of the acceleration. The signal of a peak is determined when the new data point is greater than a given x number of standard deviations away from a given moving average. For this case, we decide to use median instead of the mean for better robustness.
Using the results from the peak detection, we can validate the results of recovered data automatically by using the number of peaks and their corresponding time in the data.

D. Detecting Cocaine Use in Real-time

Another method for peak detection provided by [11] smooths data as it arrives from a data stream and detects any peaks above a user-set amplitude threshold parameter *A*. In the case of cocaine use, observers may most likely be interested in changes in three surrogates of sympathetic nervous system, i.e., valleys of temperature and peaks of sharp increase in Locomotion and electrodermal activity (EDA). The algorithm may run in parallel to detect peaks in both data streams at the same time. After finding such a peak, additional information is computed and displayed to the user including location of the peak and width derived from a least-squares curve-fit of a Gaussian function. For the purposes of this experiment, simply knowing the height and location of a peak is enough.

$$g = e^{\left(-\frac{j-\frac{n}{2}}{0.6005612049323 * \frac{n}{2}}\right)^2} \tag{6}$$

$$V_j = g_j, \forall j \in [1 \ldots n] \tag{7}$$

$$m_j = \frac{V_j}{\sum_{k=1}^{n} V_k}, \forall j \in [1 \ldots n] \tag{8}$$

$$h_j = \left(\sum_{k=j}^{j+n} m_{k-j} * R_k\right) \forall j \in [0 \ldots oL - n] \tag{9}$$

$$d_j = \left|\frac{oL*(h_j - h_{j-1})}{10}\right|, \forall j \in [n \ldots oL] \tag{10}$$

In order to find these peaks, the algorithm hinges on the gaussian function provided in [11] and transcribed in Eq. (6) to manipulate all observations within a range of a given smooth width *n*. Preliminary Smooth Vectors *V* are calculated for all numbers in the Smooth width range *[1 ... n]* (Eq. (7))which are then all divided by the sum of all Smooth Vectors to give a finalized gaussian scale of smooth vectors *m* (Eq. (8)). This scale is applied to every observation from *n* to *oL* of file *R* by taking the previous *n* observations, applying the smooth vectors *m* to each corresponding observation and adding all vectors to obtain the final corresponding smoothed point *h* that may be plotted and analyzed by peak detection algorithms. This algorithm therefore loses the first *n* observations in order to obtain smoothed points. In order to run Eq. (9) in real-time, only one datapoint *j* and its *n* predecessors are required. Instead of looking ahead as Eq. (9) suggests, the datapoint would consider prior datapoints instead of successive ones. These smoothed data points may also be used to determine significance of the detected peak by observing the rate of change of data within a data stream. Eq. (10), taken from [11] does this with derivatives which have their own defined amplitude threshold *a* to describe significance. With these equations, after smoothing, datapoints may be considered for peak detection using the algorithm below.

Algorithm 2 Real Time Peak Detection

```
Input: Result file R, length of result file R: oL, amount of
input streams c, amplitude threshold A, derivative amplitude
threshold a, smooth width n

1: Use Algorithm 1 to obtain a result file R
2: Use Eq. (6), Eq. (7) and Eq. (8) to obtain n amount of
smooth vectors m
3: do
4:     for each c
5:         if current datapoint j in c < n
6:             Skip
```

```
7:      Else
8:          Use Eq. (9) to map j → $h_j$
9:          Use Eq. (10) to map $h_j$ → $d_j$
10:         if $h_{j-2} < h_{j-1}$ and $h_{j-1} > h_j$ and $h_{j-1} >$ A and
            $d_j >$ a
11:             Report a significant peak for c at j-1
12:         end if
13:     end if
14: end for
15: if all c report significant peaks within n observations
16:     return j
17: end if
18: while Algorithm 1 returns a new file R
```

Cocaine use has been found to cause a sharp increase in EDA and body motion while causing a decrease in body temperature over time. Using these metrics concurrently allows us to predict when cocaine was administered to a user. The data streams used in this study contains labelled events of cocaine use which can be detected using Algorithm 2 above. This is completed by detecting changes in three surrogates of sympathetic nervous system, a significant peak in EDA, locomotion (Z-axis measuring xxx), and a valley in temperature (obtained by using the absolute values of the real time temperature data to detect peaks, though decreased after cocaine use). Significance here is defined as a high peak of the first derivative of a stream followed by a valley. This is interpreted as a sudden increase or decrease (since Eq. (10) considers absolute values of derivatives) in the original data. These peaks are detected within 3 minutes of their initial occurrence which is the maximum time frame it takes for cocaine side effects to be observable, thus giving an almost real-time evaluation of cocaine use.

V. Testing Results

A. Transmitting larger files and multiple streams of data

GNU Radio was found to have a file size limit described earlier in this paper as *w*. It was found that files larger than this limit experienced premature cutoffs (terminating character X was placed prior to end of file) and incomplete data streams amongst other displacements of observations within the file. This limit also needs to be downsized further when considering more data streams for transmission as the recording period increases. For example, utilizing 2 streams of data *c* allows for a chunk-size *mL* of up to 200 rows whereas 6 columns/streams of data may only allow for chunk-sizes of 80 rows as there are more bits needed to represent all streams. These numbers are examples of file size when considering *D*=5. Reducing this number will significantly increase the number of rows available, but, as mentioned before, may reduce accuracy overall (Figure 5).

B. Interference and Frequency Testing

5 frequencies of 1.8GHz, 2.4GHz, 3.0GHz, 3.8GHz, and 4.4GHz were evaluated on both a LoS and NLoS basis at 40 cm distance between antennae. All eleven fragments of the original file were transmitted twice per frequency, once for LoS and once for NLoS; if an error occurred, they were re-transmitted. BER was observed in all cases. Figure 7 below shows the output average BER of all eleven files including re-transmissions per frequency. As frequency increases, BER decreases in a mostly linear fashion. BER also remains higher in NLoS scenarios than LoS scenarios for the most part. An interesting piece to note about this data is that in some cases, BER was the exact same in LoS and NLoS, up to the precision of 10^-6. It is likely that all BER values are skewed to be higher than their actual values due to the padding cutoff problem avoided by the conversion and reverse conversion scripts; as noted before, transmitted files are often cut-off prematurely by GNU Radio transmission and started after around 8 values. These errors occur consistently and may cause a large initial bit difference per transmission, resulting in a skewed BER. Even in this small sample, an overall decrease in BER is still observed as frequency increases in both LoS and NLoS situations.

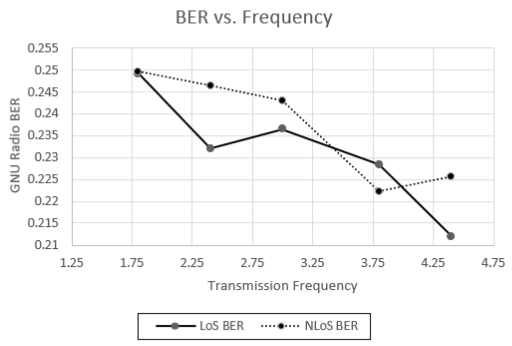


Fig. 7. BER vs. Frequency conducted at the 5 mentioned frequencies

C. Validation Testing

1) Excel Validation of recovered data: Excel validation following the steps outlined above was performed after validating each resulting fragment in the reverse conversion script. All NLoS and LoS transmissions resulted in accurate 1-to-1 results. Some fragments needed to be repeated as requested by the reverse conversion script. This ensured that *R* was indeed accurate and representative of the original data.

2). Algorithm Validation of Cocaine Data: In Figure 8, we are using the LMS and the SPDA for validation of the data. From this figure, we observed that the peaks detected using both algorithms are identical in the ground-truth data and the recovered data. Thus, we can conclude that the data has the same shape before and after transmission giving us 1-to-1 results.

Additionally, as an emulation tool for real-time machine learning testing in substance use data streams, we demonstrate the usability of peak detection algorithms on the transmitted data. We simulate the real-time application of the data by passing signals into the algorithm as they would in a practical situation. From Figure 9, we can see that there is an extended period of negative peak event in the subject's temperature, which suggests that this is the period of cocaine's usage due to the effect of cocaine on body temperature. Another peak detection scheme is described in more detail in the next section.

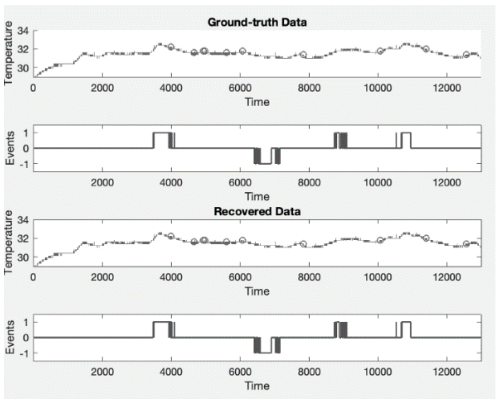


Fig. 8. Validation of recovered data using peak detection

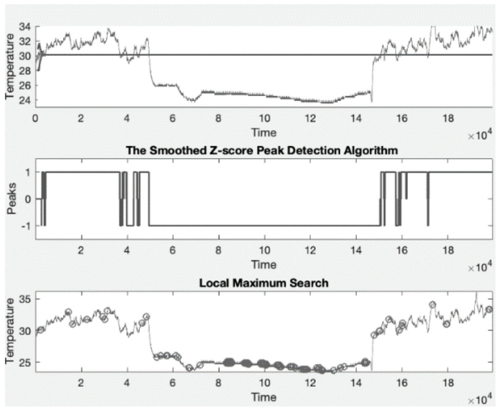


Fig. 9. Real-time peak detection

D. Detecting Cocaine Use in Real-time

The smooth width n set for this algorithm is 351 (where 360 is equivalent to 3 minutes) to detect peaks within at most 3 minutes of the initial occurrence. The labeled peaks in Figure 10 are confirmed cocaine administration to compare results of Algorithm 2 to. The amplitude threshold A to detect these peaks was 30*C for temperature, 1.5 for EDA, and 0.5 for Z. These amplitude thresholds were tuned after considering noise reduction and outliers within the three data streams. For derivative amplitude thresholds, a=1 was used for all variables. These limits are used for testing and validation purposes.

The confirmed drug use cases at Peak 1 and Peak 2 were detected by the smoothed data streams generated by Eq. (9) in Algorithm 2. Temperature experienced valleys which were deemed significant while EDA and Z experienced significant peaks within a time period of 3 minutes after the events (Figure 10). While there was a deep valley detected in temperature at Peak 1 (26*C), Peak 2 saw diminishing effects since it was the second one within an hour. Temperature was not affected as strongly as it was during the first use. However, EDA and Z saw significant changes at Peak 2, making it less subtle and easier to detect.

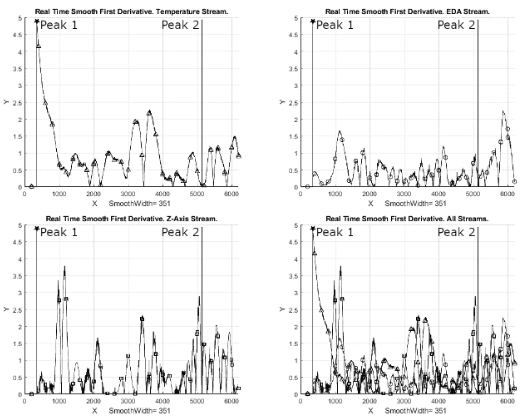


Fig. 10. Graph of the absolute value of the first derivative of negative temperature (top left, triangles), EDA (top right, circles), absolute value of Z-axis locomotion (bottom left, squares) and all three streams combined over time (bottom right) for peak detection. Confirmed peaks are labeled. These graphs are derived from Eq. (10)

While Peak 1 wasn't as easily detected due to delays in effects, Peak 2 is much more promising despite diminished temperature effects. Both Peak 1 and Peak 2 were, however, found and validated in real time using Algorithm 2, showcasing the strength and speed of the proposed detection scheme.

VI. Conclusion

Considering only a short distance of 40 cm, the 5 aforementioned frequencies were tested in LoS and NLoS environments to simulate a common environment between on-body temperature sensors and receivers. Results from Figure 7 suggest that data sent from sensors will be transmitted with higher accuracy when considering higher frequencies.

The dataset used was of real-time, biomedical nature, concerning substance use detection, specifically cocaine, based on skin temperature, locomotion, and EDA change, based on [10]. Data is collected 2 times each second and needs to be transmitted at a faster or equal rate to constitute real-time requirements and allow for accurate detection of drug use. Running Algorithm 2 allows for an estimate of time required to transmit and analyze one datapoint on average. Adding the time needed to convert (b_c), transmit (b_t), reverse (b_r), and run detection algorithms (b_d) per column c gives us $b_{avg} = \frac{0.43+23.2+0.476+3*0.00017831}{2310} \approx$ 0.01s, or 10.44ms per datapoint on average out of oL datapoints within one of the files used for testing (see Eq. 11 for details). Since data is collected twice each second, one second of sensor data can be sent, recovered and analyzed in roughly 20.871 ms on average, constituting real time recovery.

$$b_{avg} = \frac{b_c+b_t+b_r+c*b_d}{oL} \tag{11}$$

The context of cocaine use detection may be expanded to other drugs as well. The detection algorithms aren't limited to datasets and transmissions aren't limited by frequencies. There is a lot of potential for new research within substance use data, signal processing and 5G networks with this new, easily accessible and free tool, allowing for rapid development in sensor hardware and software for medical or other purposes.

Further developments in this area may aim to implement the proposed transmission scheme in 5G capable USRP equipment proposed in [1]. Another future improvement may use Out of Tree Modules (OOT Modules) for the scripts described in this paper. OOT Modules are scripts imported into GNU Radio as blocks that function as other GNU Radio packages would.

Acknowledgments

This research was partly supported by NIH grant RO1DA033323-01A1, and UMassD office of provost to Dr. Fang, NIH R01DA033769-01 to Dr. Boyer, NSF Nets 1744272 to Drs. Fang and Wang.

References

[1] A. Quadric, H. Zeng and Y. T. Hou, "A Real-Time mmWave Communication Testbed with Phase Noise Cancellation," IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Paris, France, 2019, pp. 455-460. doi: 10.1109/INFCOMW.2019.8845251

[2] R. Yared, C. A. Jaoude and J. Demerjian, "Smart-phone based system to monitor walking activity: MHealth solution," 2018 IEEE Middle East and North Africa Communications Conference (MENACOMM), Jounieh, 2018, pp. 1-5. doi: 10.1109/MENACOMM.2018.8371021

[3] S. A. Bruendl and H. Fang, "Making the Switch to 5G and 60 GHz in mHealth Applications Using USRP Hardware," in IEEE Internet Computing, vol. 24, no. 2, pp. 57-64, 1 March-April 2020, doi: 10.1109/MIC.2019.2962797.

[4] P. Zetterberg and R. Fardi, "Open Source SDR Frontend and Measurements for 60-GHz Wireless Experimentation," in IEEE Access, vol. 3, pp. 445-456, 2015, doi: 10.1109/ACCESS.2015.2414815.

[5] B. Mohanta, P. Das and S. Patnaik, "Healthcare 5.0: A Paradigm Shift in Digital Healthcare System Using Artificial Intelligence, IOT and 5G Communication," 2019 International Conference on Applied Machine Learning (ICAML), Bhubaneswar, India, 2019, pp. 191-196, doi: 10.1109/ICAML48257.2019.00044.

[6] Brakel, J. Smoothed Z-Score Algorithm. 2016. Available online: **http://stackoverflow.com/questions/22583391/peak-signal-detection-in-realtime-timeseries-data** (accessed on 10 March 2020).

[7] Yang, C., He, Z. & Yu, W. Comparison of public peak detection algorithms for MALDI mass spectrometry data analysis. *BMC Bioinformatics* **10,** 4 (2009). https://doi.org/10.1186/1471-2105-10-4

[8] J. Rumbut et al., "Harmonizing Wearable Biosensor Data Streams to Test Polysubstance Detection," 2020 International Conference on Computing, Networking and Communications (ICNC), Big Island, HI, USA, 2020, pp. 445-449, doi: 10.1109/ICNC47757.2020.9049684.

[9] J. Rumbut, D. Singh, H. Fang, H. Wang, S. Carreiro and E. Boyer, "Poster Abstract: Detecting Kratom Intoxication in Wearable Biosensor Data," 2019 IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE), Arlington, VA, USA, 2019, pp. 71-72, doi: 10.1109/CHASE48038.2019.00028.

[10] Carreiro S, Fang H, Zhang J, et al. iMStrong: Deployment of a Biosensor System to Detect Cocaine Use. *J Med Syst*. 2015;39(12):186. doi:10.1007/s10916-015-0337-9

[11] O'Haver, T., 2020. *Intro. To Signal Processing:Case Studies*. [online] Terpconnect.umd.edu. Available at: <https://terpconnect.umd.edu/~toh/spectrum/CaseStudies.html#realtime> [Accessed 22 June 2020].

[12] H. Wang, D. Peng, W. Wang, H. Sharif, H. Chen and A. Khoynezhad, "Resource-aware secure ECG healthcare monitoring through body sensor networks," in IEEE Wireless Communications, vol. 17, no. 1, pp. 12-19, February 2010, doi: 10.1109/MWC.2010.5416345.

[13] H. Wang, D. Peng, W. Wang, H. Sharif and H. Chen, "Cross-layer routing optimization in multirate wireless sensor networks for distributed source coding based applications," in IEEE Transactions on Wireless Communications, vol. 7, no. 10, pp. 3999-4009, October 2008, doi: 10.1109/T-WC.2008.070516.

[14] H. Wang, M. Hempel, D. Peng, W. Wang, H. Sharif and H. Chen, "Index-Based Selective Audio Encryption for Wireless Multimedia Sensor Networks," in IEEE Transactions on Multimedia, vol. 12, no. 3, pp. 215-223, April 2010, doi: 10.1109/TMM.2010.2041102.

[15] W. Wang, D. Peng, H. Wang, H. Sharif and H. Chen, "Cross-layer multirate interaction with Distributed Source Coding in Wireless Sensor Networks," in IEEE Transactions on Wireless Communications, vol. 8, no. 2, pp. 787-795, Feb. 2009, doi: 10.1109/TWC.2009.071009.

[16] M. S. Mahmud, H. Wang, A. M. Esfar-E-Alam and H. Fang, "A Wireless Health Monitoring System Using Mobile Phone Accessories," in IEEE Internet of Things Journal, vol. 4, no. 6, pp. 2009-2018, Dec. 2017, doi: 10.1109/JIOT.2016.2645125.

[17] R. Wang, H. Yang, H. Wang and D. Wu, "Social overlapping community-aware neighbor discovery for D2D communications," in IEEE Wireless Communications, vol. 23, no. 4, pp. 28-34, August 2016, doi: 10.1109/MWC.2016.7553023.

[18] Fang, H, Wittbold, M., Zhang, J., Weng, S, Carreiro, S, Mullins, R, Boyer, E. Describing Real-time Substance-use Detection from Big Biosensor Data: A Case study of Cocaine Users, Health Behavior Methods and Measures paper session, Society of Behavioral Medicine, San Antonio, TX.

[19] Carreiro S, Wittbold K, Indic P, Fang H, Zhang J, Boyer EW. Wearable Biosensors to Detect Physiologic Change During Opioid Use. J Med Toxicol. 2016 Sep; 12(3):255-62. PMID: 27334894.

[20] Wang CJ,* Fang H$^+$, Carreiro S, Wang H., Boyer E. A New Mining Method to Detect Real Time Substance Use Events from Wearable Biosensor Data Stream. Proceedings of International Conference on Computing, Networking and Communications: Social Computing and Semantic Data Mining, pp. 465-470. ICNC2017. NIHMSID: 907527

[21] Wearables technology for drug abuse detection: A survey of recent advancement. MS Mahmud, H Fang, S Carreiro, H Wang, EW Boyer, Smart Health 13, 100062