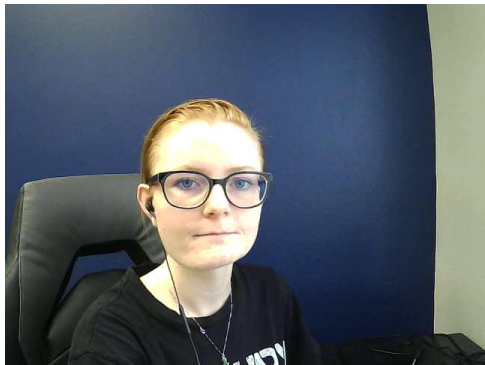
An Open Channel Identifier using GNU Radio

ASHLEY BEARD - PRESENTER

STEVEN SHARP



- *Using:* GNU Radio & Python
- *Models:* Neural network vs. Classical power estimation
- *For:* Radio channel identification
- *Accuracy:* Neural Network > Power Estimation
- *Efficiency:* Neural Network < Power Estimation

OUR CLAIM:

Outline

I. About Spectrum Bullpen, LLC.

II. Motivations

III. Related Work

IV. Experimental Setup

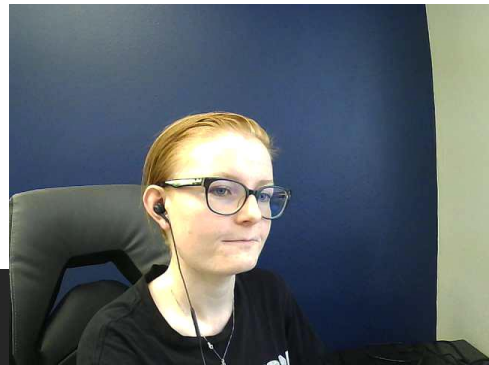
V. Data Generation with GNU Radio

VI. Algorithms

VII. Analysis Testing

VIII. Conclusions and Future Work

IX. Current R&D at SB

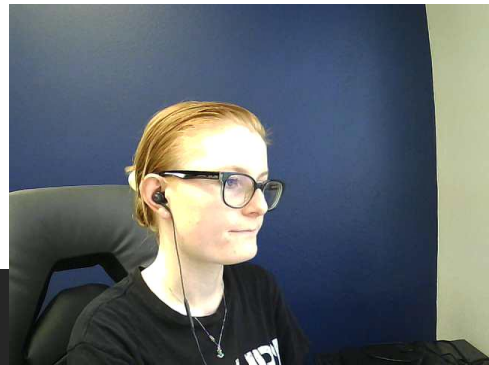


About Spectrum Bullpen, LLC.

- Founded by Raymond Shaw in 2016
- Small business in the Florida Space Coast
- Focus Areas:
  - Spectrum planning and execution
  - Software radio solutions
  - Machine learning applications
  - DSA

Motivations

- Contribute to GNU Radio community
- Applications of Channel Detection
  - Spectrum Sensing
  - Cognitive Radio
  - Interference minimization
  - DSA policy generation



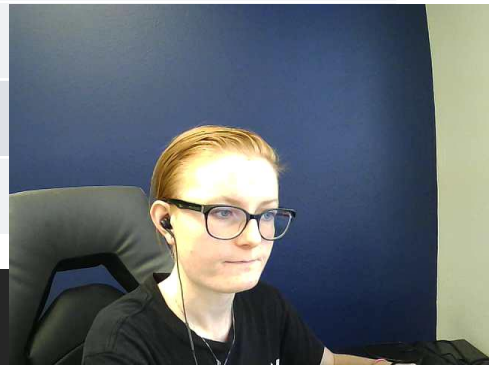
Related Work

Authors	Year	Neural Network Design	GNU Radio
O'Shea, Clancy, & Ebeid	2007	N/A	Yes
O'Shea & West	2016	13-layer CNN	Yes
Rodriguez & Dassatti	2020	N/A	Yes
Tumuluru, Wang, & Niyato	2010	Multilayer perceptron	No
Cao & Gu	2019	3-layer CNN	No
Han et al.	2020	GAN model	No
Solanki et al.	2021	CNN & LSTM layers	No

Experimental Setup

- Two algorithms head-to-head
  - Battle for the title of Superior Channel Identifier Algorithm
- Power Estimation Algorithm
  - Traditional baseline to test efficacy of ML
- Feed-forward Neural Network
  - Using stochastic gradient descent and backpropagation
- Wi-Fi signal data generated using GNU Radio
  - Randomly selected frequencies from Table 1
  - Five “occupied” and five “open”
- Are observed frequencies occupied or open?

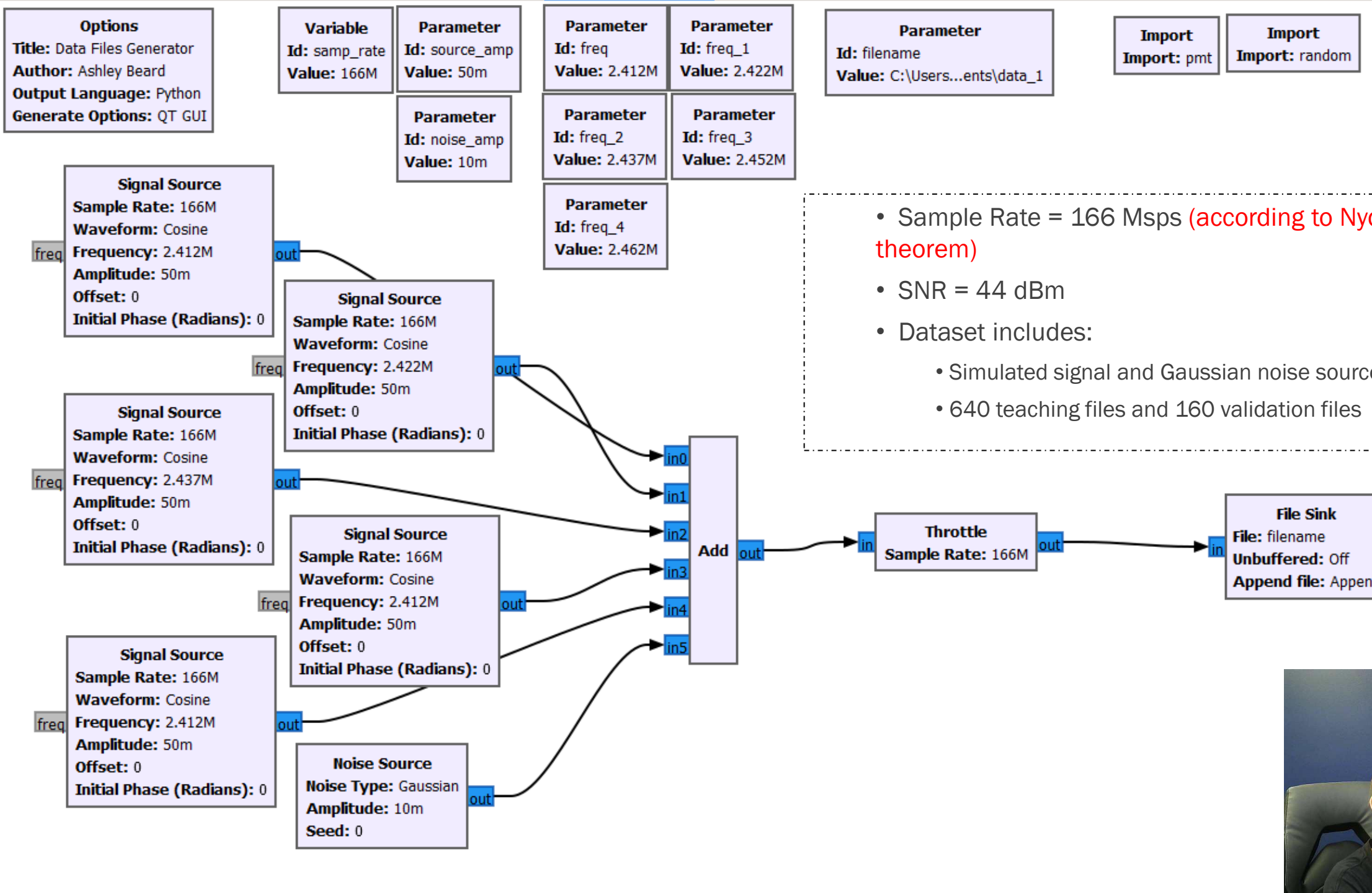
Table 1: Common Wi-Fi Band Frequencies[1]
2.412 GHz
2.417 GHz
2.422 GHz
2.427 GHz
2.432 GHz
2.437 GHz
2.442 GHz
2.447 GHz
2.452 GHz
2.462 GHz



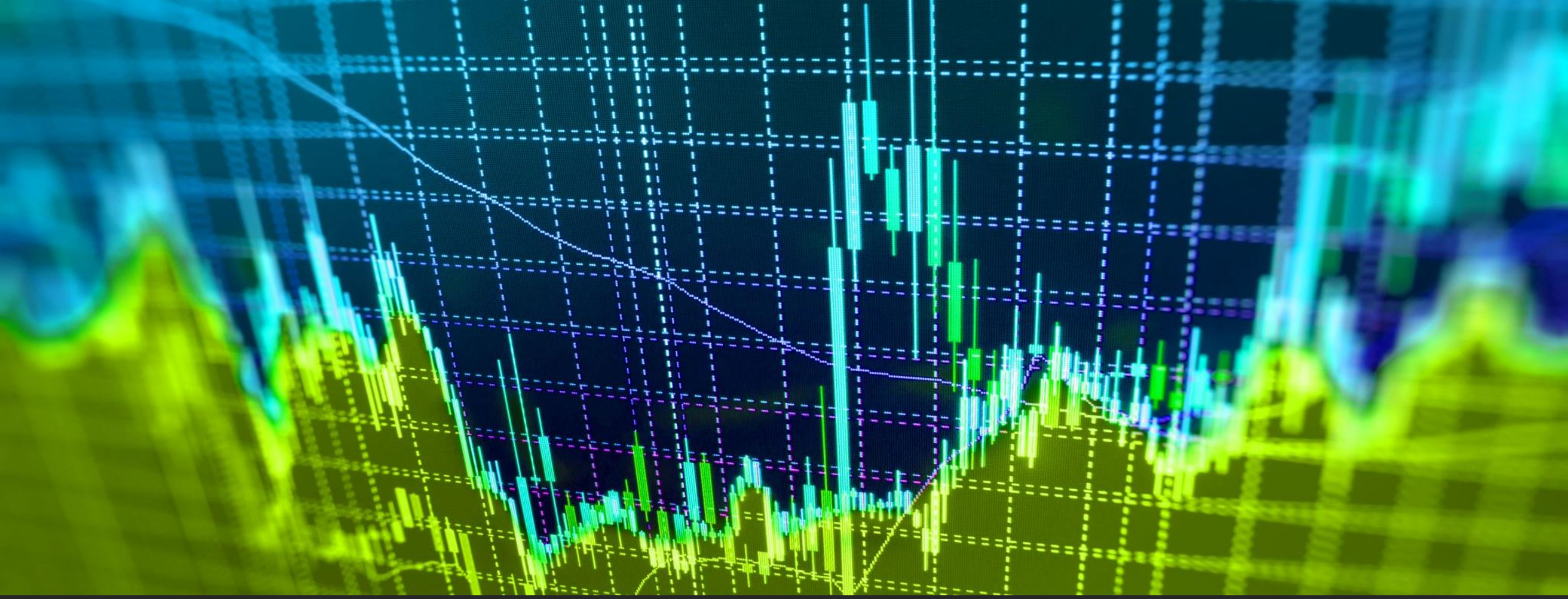
1) IEEE 802.11-2016: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications, December 2016. Table 15-6.

- Windows 10 System
- Version: 3.8.0.0
- Python version: 2.7.10
- Packages required: time, random, pmt, numpy

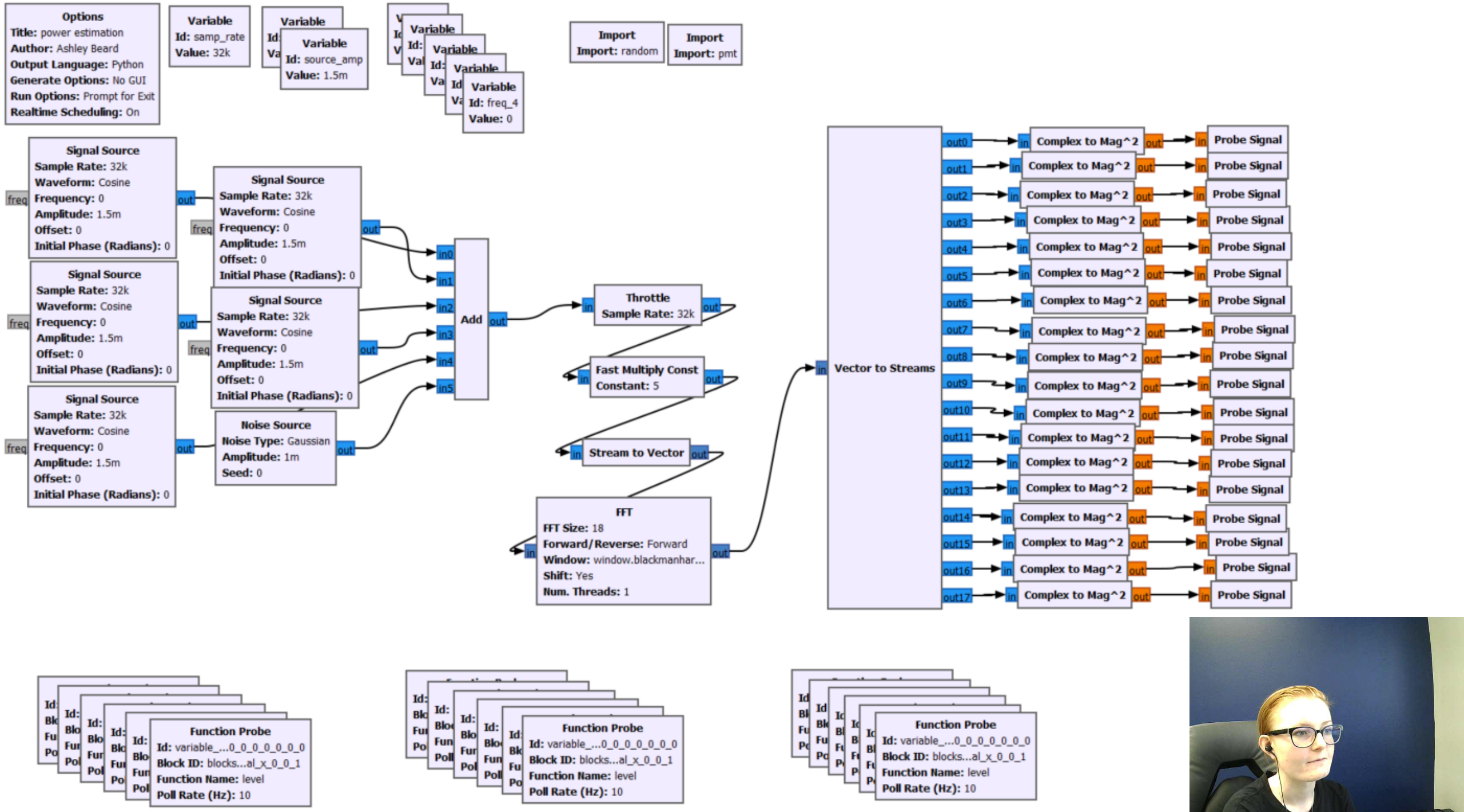
GNU Radio Data Generation



- Sample Rate = 166 Msps (according to Nyquist-Shannon theorem)
- SNR = 44 dBm
- Dataset includes:
  - Simulated signal and Gaussian noise sources
  - 640 teaching files and 160 validation files



Power Estimation

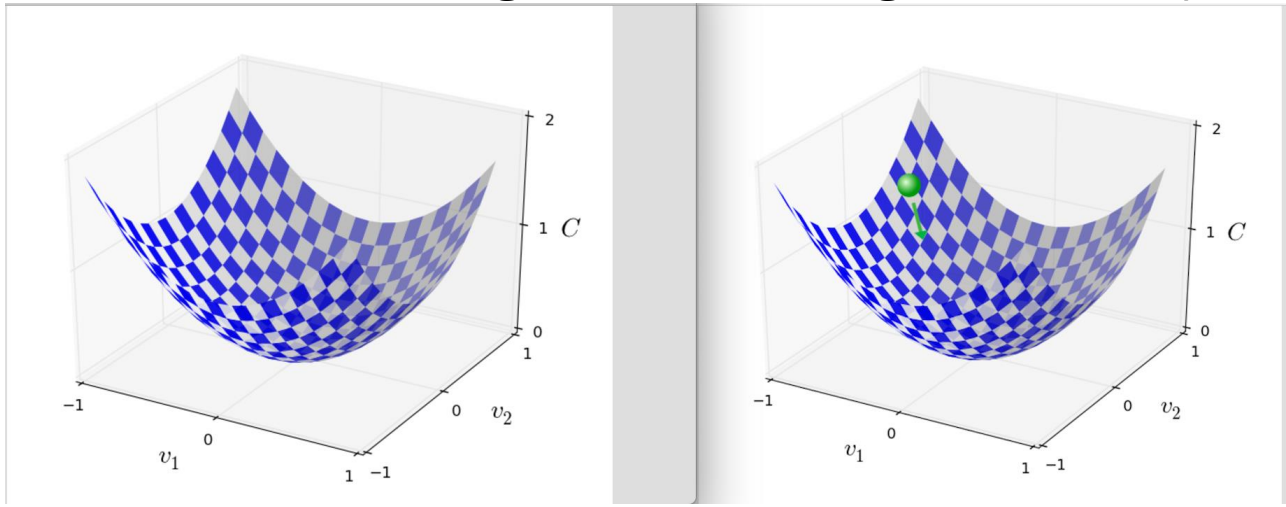

Options
Title: power estimation
Author: Ashley Beard
Output Language: Python
Generate Options: No GUI
Run Options: Prompt for Exit
Realtime Scheduling: On
Variable
Id: samp_rate
Value: 32k
Variable
Id: source_amp
Value: 1.5m
Variable
Id: freq_4
Value: 0
Import
Import: random
Import
Import: pmt
Signal Source
Sample Rate: 32k
Waveform: Cosine
Frequency: 0
Amplitude: 1.5m
Offset: 0
Initial Phase (Radians): 0
Signal Source
Sample Rate: 32k
Waveform: Cosine
Frequency: 0
Amplitude: 1.5m
Offset: 0
Initial Phase (Radians): 0
Signal Source
Sample Rate: 32k
Waveform: Cosine
Frequency: 0
Amplitude: 1.5m
Offset: 0
Initial Phase (Radians): 0
Signal Source
Sample Rate: 32k
Waveform: Cosine
Frequency: 0
Amplitude: 1.5m
Offset: 0
Initial Phase (Radians): 0
Noise Source
Noise Type: Gaussian
Amplitude: 1m
Seed: 0
Add
Throttle
Sample Rate: 32k
Fast Multiply Const
Constant: 5
Stream to Vector
FFT
FFT Size: 18
Forward/Reverse: Forward
Window: window.blackmanhar...
Shift: Yes
Num. Threads: 1
Vector to Streams
Complex to Mag^2
Probe Signal
Function Probe
Id: variable_...0_0_0_0_0_0_0
Block ID: blocks...al_x_0_0_1
Function Name: level
Poll Rate (Hz): 10

Neural Network

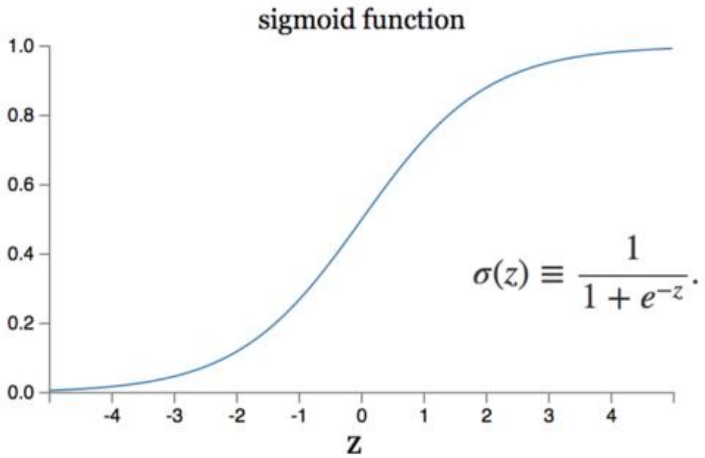
Feedforward Neural Network

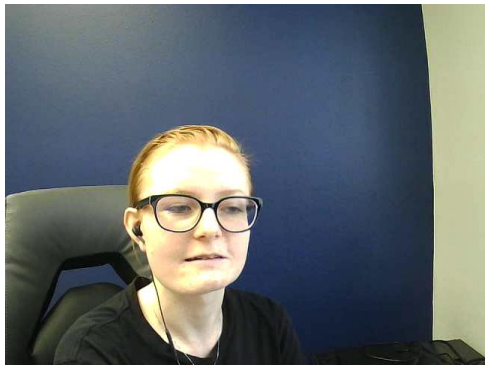
- Five layers: One input, three hidden, one output
- Sigmoid activation function
- Stochastic gradient descent:
  - Descend to lowest estimate error
  - Choosing randomized initial weights
- Backpropagation training:
  - Recalculate weights, feed back into first layer, repeat

Gradient Descent: Analogous to a ball rolling to the lowest point



Images courtesy of: https://towardsdatascience.com/deep-learning-feedforward-neural-network-26a6705dbdc7





Analysis

I. Percent Error
II. Randomness Test
III. Computational Complexity

Percent Error

- Feedforward NN: **66.17% accuracy**
- Power Estimation: **49.12% accuracy**
- Reasons:
  - Converting to magnitudes
  - High frequency → High sampling rate[1]
  - Optimizing learning rate (Hyperparameter)

$$M=\sqrt{I^2+Q^2}$$

$$\phi=\tan^{-1}\left[\frac{Q}{I}\right]$$

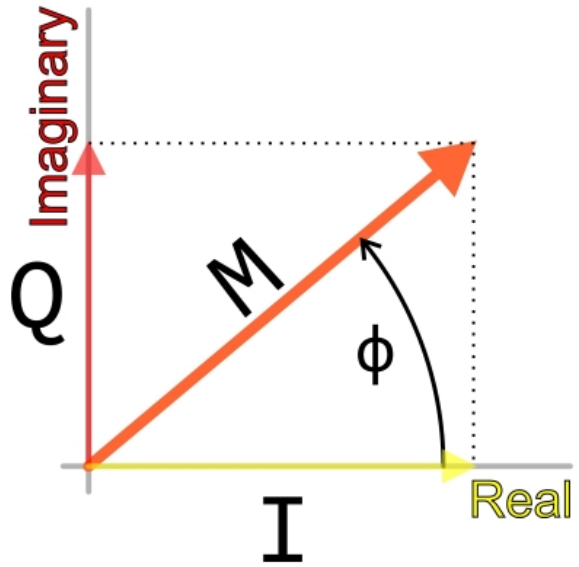
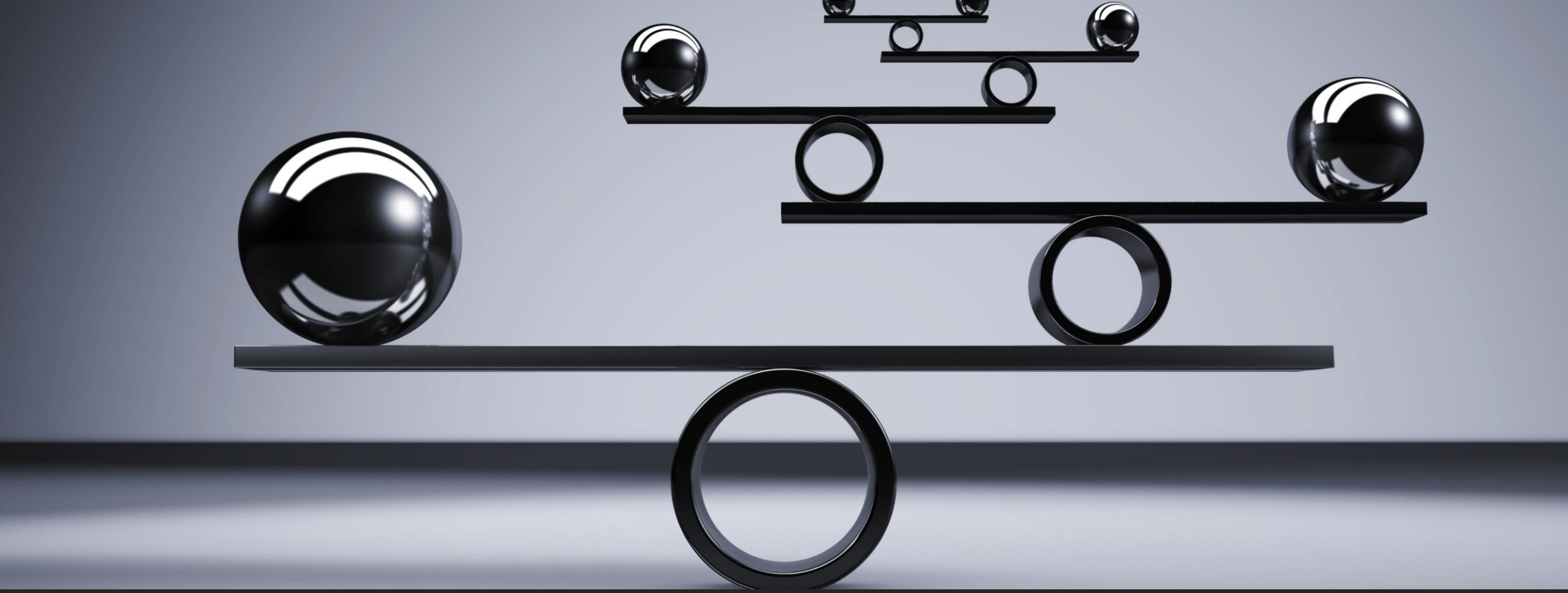


Image courtesy of https://www.allaboutcircuits.com/technical-articles/how-to-process-iq-signals-software-defined-rf-receiver-dsp-digital-signal/

1) Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F. A., Bengio, Y., and Courville, A. On the spectral bias of neural networks. In *36th International Conference on Machine Learning*, pp. 5301–5310, Long Beach, CA, USA, 2019.

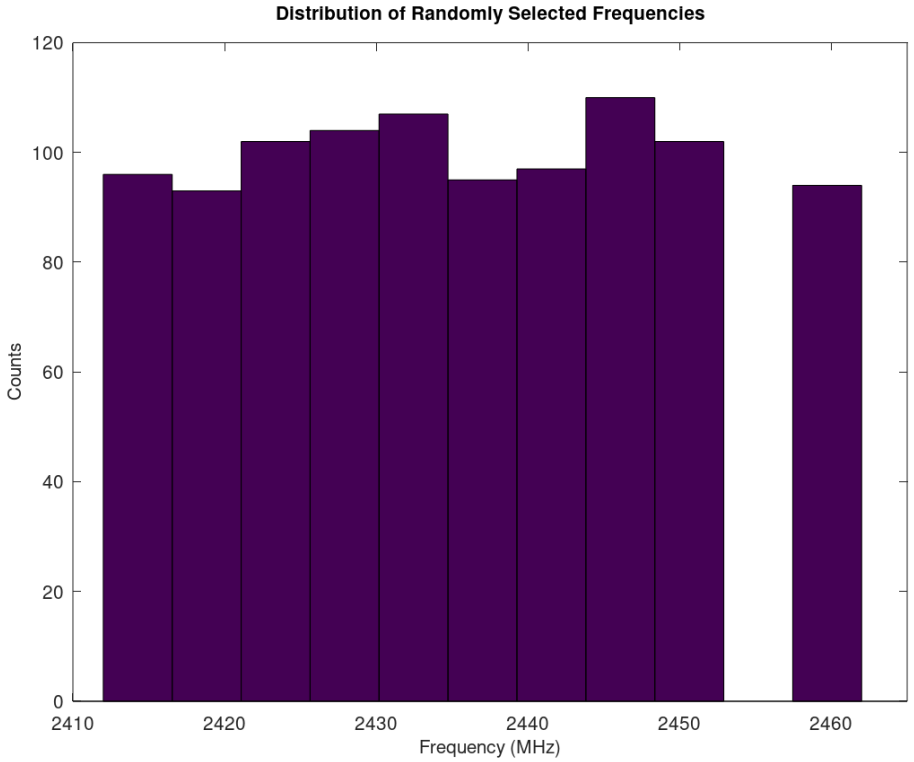


Randomness Testing

Randomness Testing

- Importance of pseudo-random algorithms
- Causes bias to generated data
  - NN could train incorrectly

- **Right:** Selection of signal frequencies shows a mostly uniform distribution (aside from gap with no available frequency)



Computational Complexity

Computational Complexity

POWER ESTIMATION ALGORITHM

Fast Fourier Transform: $O(Nlog_2N)$

Python Timsort: $O(Nlog_2N)$

Mag^2: $O(N)$

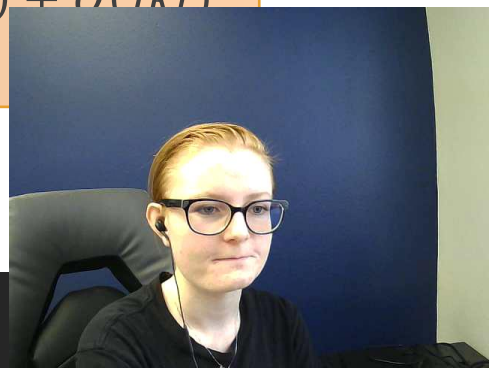
Full: $O(Nlog_2N) + O(Nlog_2N) + O(N)$

FEEDFORWARD NEURAL NETWORK

Matrix multiplication: $O(abc)$

- Where matrices are not square: $AB = C$, $A \in \mathbb{R}_{a\times b}$ and $B \in \mathbb{R}_{b\times c}$

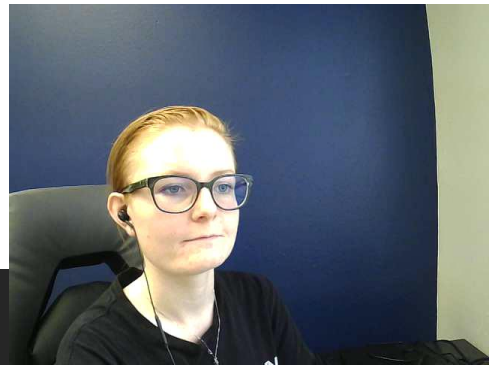
Full: $O(abc) + O(def) + O(ghi) + O(jkl)$



Conclusions & Future Work

- GNU Radio implementation of:
  - Bandwidth data generation and storage
  - Signal Power Estimation
  - Simple Feedforward neural network
- Results:
  - Neural network > traditional power estimation
    - Greater accuracy, but higher complexity
- Future Improvements:
  - Complex ($\mathbb{C}$) input
  - Hyperparameter optimization
  - Deep layer interactions

	Power Estimation	Neural Net
Accuracy	49.12%	66.17%
Complexity	$O(Nlog_2N) + O(Nlog_2N) + O(N)$	$O(abc) + O(def) + O(ghi) + O(jkl)$
Easily modified for more applications?	Not Easily	Yes!



Current R&D at SB:

RF Propagation (RFP): development and testing involves comparing the Longley-Rice radio propagation model for Irregular Terrain and a predictive model that uses NN's to emulate that model in a superior manner

RF Fingerprinting (RFF) : using a similar NN model from this paper to not only detect signals but to further identify the type and intentions of signal transmitter

Thank You

Data and code can be found at a github repository here:
https://github.com/AABeardSB/channelEst

View our company website: https://www.spectrumbullpen.com/