An Open Channel Identifier using GNU Radio

ASHLEY BEARD - PRESENTER

STEVEN SHARP
OUR CLAIM:

- **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
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

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

<table>
<thead>
<tr>
<th>Frequency (GHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.412</td>
</tr>
<tr>
<td>2.417</td>
</tr>
<tr>
<td>2.422</td>
</tr>
<tr>
<td>2.427</td>
</tr>
<tr>
<td>2.432</td>
</tr>
<tr>
<td>2.437</td>
</tr>
<tr>
<td>2.442</td>
</tr>
<tr>
<td>2.447</td>
</tr>
<tr>
<td>2.452</td>
</tr>
<tr>
<td>2.462</td>
</tr>
</tbody>
</table>

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
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
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
  - Optimizing learning rate (Hyperparameter)

\[
M = \sqrt{I^2 + Q^2} \\
\phi = \tan^{-1} \left( \frac{Q}{I} \right)
\]


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

POWER ESTIMATION ALGORITHM

Fast Fourier Transform: $O(N \log_2 N)$
Python Timsort: $O(N \log_2 N)$
Mag^2: $O(N)$

Full: $O(N \log_2 N) + O(N \log_2 N) + O(N)$

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: $O(abc) + O(def) + O(ghi) + O(ikl)$
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 (ℂ) input
  - Hyperparameter optimization
  - Deep layer interactions

<table>
<thead>
<tr>
<th></th>
<th>Power Estimation</th>
<th>Neural Net</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>49.12%</td>
<td>66.17%</td>
</tr>
<tr>
<td>Complexity</td>
<td>$O(N \log_2 N) + O(N \log_2 N) + O(N)$</td>
<td>$O(abc) + O(def) + O(ghi) + O(jkl)$</td>
</tr>
<tr>
<td>Easily modified for more applications?</td>
<td>Not Easily</td>
<td>Yes!</td>
</tr>
</tbody>
</table>
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/