Analysis of an Open Channel Identifier using Stochastic Gradient Descent and GNU Radio

Ashley Beard ASHLEY.BEARD@SPECTRUMBULLPEN.COM
Spectrum Bullpen, LLC. 6050 Babcock St. SE., Ste. 26, Palm Bay, FL 32909, USA

Steven Sharp STEVEN.SHARP@SPECTRUMBULLPEN.COM
Spectrum Bullpen, LLC. 6050 Babcock St. SE., Ste. 26, Palm Bay, FL 32909, USA

Abstract

In this paper, we address the problem of radio spectrum crowding by using a stochastic gradient descent neural network algorithm on simulated cognitive radio data to identify open and closed channels within a specified RF range. We used GNU Radio 3.8 flowgraphs to simulate cognitive radio data for standard U.S. Wi-Fi channels, and to design both the neural network and classical power estimation algorithms. Our methods include the possibility for leveraged use in many spectrum sensing applications such as channel detection, modulation classification, and radio fingerprinting. We provide analytical insight into the performance of this neural network that goes beyond that of previous work in this immediate field. These analyses will show the stochastic gradient descent algorithm achieves an advantageous accuracy over the traditional channel occupation algorithm.

1. Introduction

1.1. Impact Statement

In Wi-Fi broadcasting, the signal traffic is becoming unwieldy for internet companies even when using the current Orthogonal Frequency Division Multiplexing (OFDM) modulation and several different frequency bands (Mathai & Sagayam, 2013). Cognitive radio technology could be a useful method to alleviate congestion by adaptive reuse of the frequency bands. As more households and businesses gain access to Wi-Fi routing, more signals need to be distributed in a set bandwidth without interfering. Our algorithm can detect open channels given a list of frequencies in any bandwidth in the electromagnetic spectrum. This method can be propagated to many other uses, including radio transmission/receiving, utilizing "white spaces" in the television spectrum noted from (Zeng et al., 2010), or efficiently sorting through "Big Data" in radio astronomy (Lukic et al., 2020) (Alhassan et al., 2020).

1.2. Neural Networks

Neural networks (NNs) use machine learning (ML) algorithms with structures inspired by the theory of neuron structure in biological systems. There can be an infinite number of permutations of neurons and connections for a NN, therefore there can be an infinite number of NN topologies constructed. The advantages of using ML to analyze data are the speed, accuracy, and autonomy improvements compared with human observation. These algorithms can make classifications, sort and categorize data, and find patterns in data that are otherwise elusive to traditional algorithms that don't utilize ML. The performance of a network depends on several factors such as size (number of neurons and/or layers), activation functions, organization, and the type of application one is attempting. For example, research in the field has shown a better performing NN for an image classifier is one of several variations of convolutional NNs (CNNs) because of the large number of neurons that are layered and fully-connected between each layer. This scheme allows for the comparing of pixel groups with their neighbors.

The process of training NNs are categorized into two main categories of ML: supervised and unsupervised. Supervised learning requires inputs from both raw data and associated training data which supplies the expected output of the algorithm. Unsupervised learning uses an input of only raw data and the algorithm will find a pattern without knowing the expected output. The best learning type choice depends on the type and context of data. Unsupervised learning is best used for applications such as clustering data, classification, or finding anomalous data (Barlow, 1999). Supervised learning is used in applications such as facial recognition or spectrum sensing in cognitive radios (Thilina et al., 2013). The latter of which will be our focus for this research.

Proceedings of the 11^{th} GNU Radio Conference, Copyright 2021 by the author(s).

Feedforward Neural Networks: Our algorithm for determining an unoccupied RF channel uses a simple feedforward (FF) NN topology. The network's weights are updated or trained using stochastic gradient descent and backpropagation. The topology of the FF network consists of a single input layer, one or more hidden layers, and a single output layer. In stochastic gradient descent, the product of the input data and its weight value is first fed into a cost function. The gradient of the function is then calculated using backpropagation. Depending on the sign and magnitude of the gradient calculated as well as the predetermined learning rate, the weight values in each layer are recalculated according to their level of contribution when moving the layer values closer to the expected output (Nielson, 2015). At the start of the next training iteration of data, the weights are re-introduced into the calculations and the process continues until an acceptable error has been reached or a specified number of iterations has been exhausted.

1.3. Cognitive Radio and Spectrum Sensing

An advantage of using a NN over a traditional power estimation method is virtually no need for *a priori* known parameters of primary user activities. The primary broadcaster may need to keep proprietary information about their transmit and receive (Tx-Rx) system confidential, making full communication between primary and secondary users difficult or impossible (Duan et al., 2014). NNs can be fed raw transmitted signals and characterize primary user signals without the need of system information. This allows a faster and more efficient dynamic spectrum access (DSA) (Tumuluru et al., 2010) (Zhou et al., 2018).

Our algorithm uses spectrum sensing to detect what frequencies are available to use. Applying this process to a radio, Wi-Fi router, wireless sensor, or base station, these devices will then have the information needed to adjust broadcasting frequency automatically. There are certain standards designated for spectrum sensing given by the IEEE 802.22 (Shellhammer, 2008). The required detection time is two seconds, meaning if two transmitters share a frequency, a secondary transmitter must detect a primary transmitter two seconds after the primary has started broadcasting. The specified bandwidth needs to be vacated, otherwise, there might be strong interference between the receivers (Zeng et al., 2010). Another requirement is to limit the normalized probability of outputting a false signal decision to 0.1 or less. In other words, the spectrum sensing method must not exceed false signal decisions more than 10% of the time. The inputs used by the spectrum sensing algorithm in (Shellhammer, 2008) include the channel number, channel bandwidth, and optionally the input signal type (analog, digital, etc.). Our algorithm accepts an input of a list of ten channel frequencies on which to make signal decisions. The sensing mode that we have used in our algorithm defines two outputs: the signal present decision and the confidence metric. The first output gives a binary value that specifies whether a signal is present in the given channel and the second output is the percent error that the spectrum sensing algorithm possesses when deciding.

Another possible spectrum sensing method is described in (Nasser et al., 2021) as a "listen-and-talk" approach called Full-Duplex Cognitive Radio (FDCR). Self-interference cancellation (SIC) is important in FDCR to allow simultaneous scanning and transmitting without detecting one's own signal. Three paradigms of cognitive radio are underlay access, overlay access, and interweave access. Interweave access and FDCR are the main interests of this paper due to the current challenge of energy and computational efficiency of these implementations. FDCR is most beneficial for the standard in spectrum sharing of secondary users cutting their transmission within the acceptable window as they detect a primary-user transmission (Liao et al., 2017).

1.4. Related Work

- Tumuluru et al. use a multi layer perceptron very similar to the feedforward model we design in this paper, to sense the occupancy status of a spectrum (Tumuluru et al., 2010). They update parameters using batch backpropagation similar to our NN training method. We perform a more detailed analysis on our model using simulated GNU Radio signal data.

- A conference paper by Cao and Gu describes results of neural network experiments on Gaussian and specific non-Gaussian distributions of data – in particular, uniform distributions over a unit sphere, and transelliptical distributions (Cao & Gu, 2019). The work we present in our manuscript uses stochastic gradient descent with a sigmoid activation function similar to one of their tested methods. However, our data differs from their approach as it is a randomized sample distribution of frequencies, confirmed in our analysis in Section 4.

- There has been previous work that utilize ML with GNU Radio. For example, an Out-of-Tree (OOT) module was designed by (Rodriguez & Dassatti, 2020) to use a deep-learning model - specifically a CNN to classify modulation types in signals. They use a PlutoSDR receiver to gather their data instead of our method of simulating data using GNU Radio's Signal Source block. Rodriguez and Dassatti mention taxing computational work in their model, but do not quantitatively describe the computational cost of their model.

- Solanki, Dehalwar, and Choudhary introduce a net-

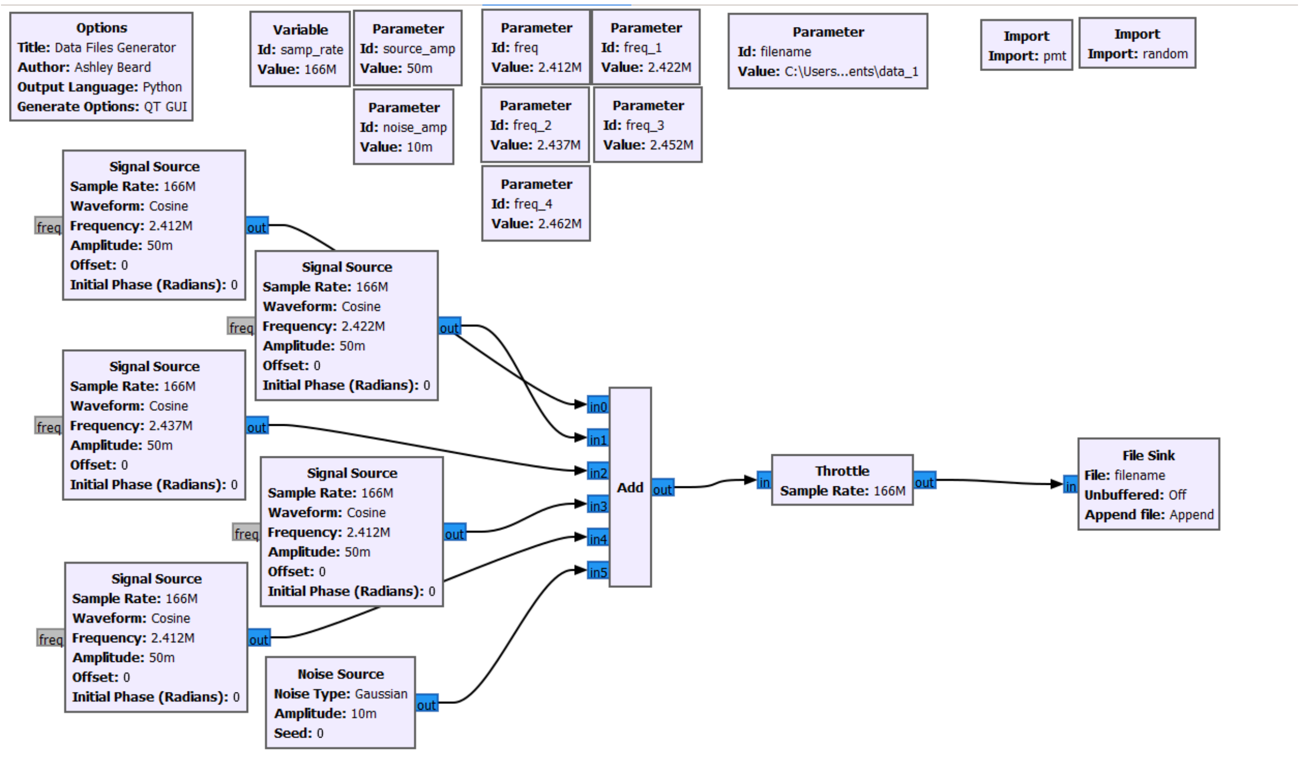

Figure 1. GNU Radio data generation flowgraph

work containing a combination of Long Short-Term Memory and CNN layers to recognize temporal dependencies and spatial dependencies of data respectively. Their model was inspired by three previous models by (Cheng et al., 2019) and (Gao et al., 2019); the authors called it "DLSenseNet" (Solanki et al., 2021). The Inception architecture used in their model provides accurate results for the purposes of Solanki et al, but it demands a high computational cost that is not required for our model. This high accuracy is in part due to the capability of accepting complex data instead of our method of converting complex values to magnitudes and hence stripping some relevant data features in the process.

- A cognitive radio called MEGANs was trained using power spectral maps as images by Han, Xue, Shao, and Xu in (Han et al., 2020). CNNs were used in this case because they have documented superiority in image classification and reconstruction (Jin et al., 2017). Our FF NN is a form of CNN but we use a different activation function from the commonly used *softmax* function in CNNs (Agarap, 2019). They showed that their MEGAN model outperforms various traditional models in power spectrum map estimation however, they did not explore the comparisons to other learning models such as FF NNs. This manuscript provides that lacking contribution towards power estimation.

These papers are described above to demonstrate our use of methods that have been shown to work in past manuscripts, as well as show the lack in critical analysis or comparison of these methods in the spectrum sensing and cognitive radio fields. Our manuscript aims to use a common NN model for the sake of simplicity and incorporate advanced analysis methods to better describe the performance of ML in a relevant real-world scenario.

The rest of the manuscript is organized as follows. Section 2 describes the creation process and formatting of the simulated data in our work. Section 3 describes the algorithms and data manipulation for our two signal detection methods followed by a description of the formatting of the algorithm outputs. Section 4 contains a collection of analyses on our results, including an analysis of the percent error in our signal detection methods, of the random bias in our algorithms, and of the compared computational complexity of our methods. Our concluding statements and intended future work are included in Section 5, followed by Acknowledgements and References.

2. Data Creation Using GNU Radio

The data used to train and test the NN was generated using GNU Radio, which is a DSP software. GNU Radio has a graphic interface in which the user builds unique flowcharts of digital signal processing components. Once compiled and executed, these flowcharts can produce Python scripts, which can also be modified for more specific intentions such as iterating multiple times through a section of the flowchart and outputting data files.

The basic form of what blocks were needed for the data generation were called and organized using the GNU Radio flowgraph tool. However, in order to automate generation of large numbers of data files, we modified the compiled Python file generated by the flowgraph. The flowgraph in Fig. 1 represents one iteration of a looping, file-generating process.

Table 1: Common Wi-Fi Band Frequencies (IEEE, 2016)
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

We designed a model using five signal sources, all with a clockwise waveform, a sample rate of 166 Msps, and an amplitude of -35.0 dBm. Each signal possessed a randomized selection of frequencies from Table 1 of commonly used 2.4GHz ISM band Wi-Fi frequencies referenced from (IEEE, 2016). The model also included a Gaussian noise source with an amplitude of -79.0 dBm. These signal sources were sent through an addition and throttle block resulting in a single data stream set to the specified sample rate of 166 Msps. We chose this sample rate to fall within the bounds of the Nyquist-Shannon sampling theorem (Shannon, 1949). We generated a set of 800 files of binary I/Q data and 800 accompanying text files listing the open and closed frequencies (as Boolean values) associated with each I/Q data set. The I/Q data from our five signal sources and one noise source is output as alternating I (In phase) and Q (Quadrature) components of output signal. Those files were split into 80% or 640 file sets being fed into the training phase of the NN and 20% or 160 file sets being reserved for the test phase. Those 160 reserved files were introduced as "novel data" to test the accuracy of the algorithm output identifying the correct frequencies as "open" or "in use".

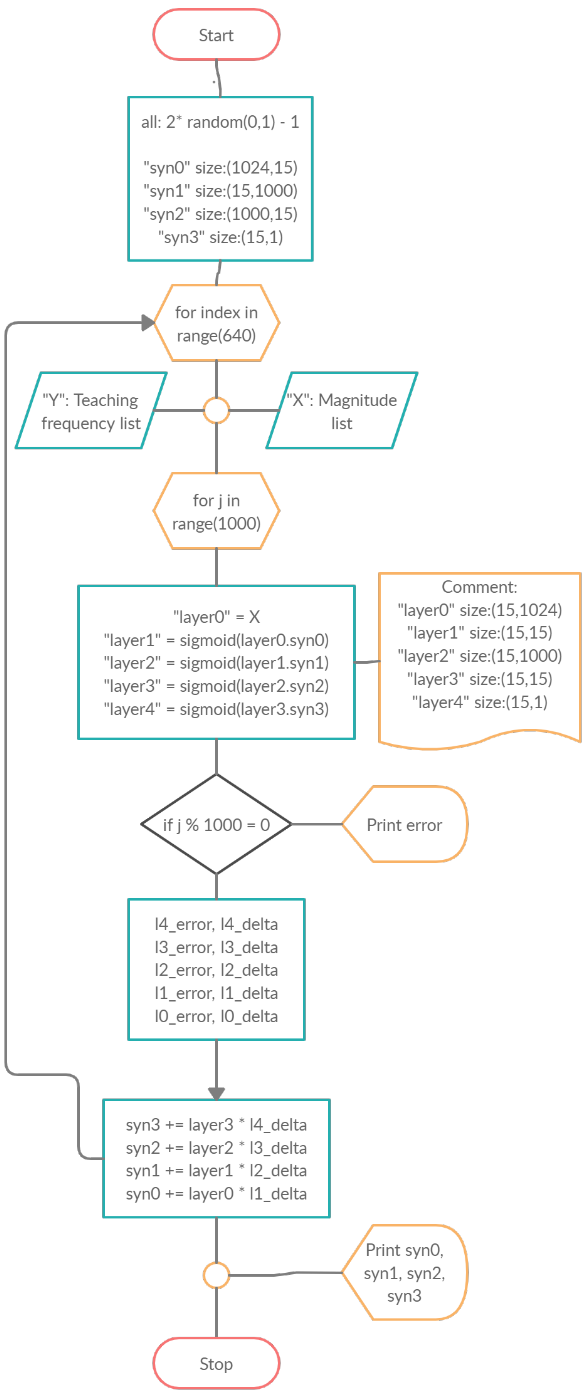

Figure 2. Algorithm flowchart of Neural Network feedforward and backpropagation calculations

3. Structure of Algorithms

3.1. Structure of the Neural Network

We used supervised ML as opposed to unsupervised ML to allow us to train the algorithm using our known inputs and expected outputs. A simple FF network trained using backpropagation was selected because our input and output data were both binary vector streams. We used stochastic gradient descent in the NN algorithm because it was the most computationally efficient choice for the size of our input data as opposed to a batch gradient descent algorithm or something similar. Our NN contained a sigmoid cost function (1), three hidden layers (with one input and one output layer), and four weight variables. The array sizes and relationships of these layers and weights can be seen in Fig. 2. A learning rate (ϵ_0) of 0.0001 was incorporated into the network as well. We use a standard value of $10^{-6} < \epsilon_0 < 1$ for the learning rate that works best for gradient descent learning (Bengio, 2012a).

$$\sigma(x) = \frac{1}{1+e^{-x}} \in (0,1),\ x \in \mathbb{R} \tag{1}$$

Preprocessing the Data: To increase the accuracy of the NN learning step, five consecutive sets of 1024 I/Q data were taken from each data stream file. These sets were combined into an array of 1024 averaged values. This process was conducted ten times for a total of ten samples of averaged sets of 1024 I/Q data. Averaging the complex values in this way allowed the NN to better read and catalog the features of the input data. The data generated by GNU Radio consisted of complex I/Q data while our NN model is designed for real-valued data, so modifications needed to be made to our input. The FF framework was compatible with float input, not complex values, so we converted each input sample into magnitudes. We achieved this by computing the norm of each sample of I/Q data before feeding the input matrix into the learning loop of the NN. The option of keeping the complex-valued data by treating the real and imaginary components as separate features was considered. We chose to keep the data paired together to maintain the 1024-sized input.

Each input sample contained a size 1024 vector of I/Q data, making a 10-by-1024 size array for the ten samples. Four common data features were concatenated to the end of the NN data input array. The mean, variance, standard deviation, and maximum of each size-10 column of I/Q data were calculated and appended to the input. A final column of zeros was appended to the input array as sequence padding. Using this method has shown to provide the NN a better probability of learning the features of our data input and improve the accuracy of a FF NN (del Rio et al., 2020). This results in a final input size of 15-by-1024 after incorporating the appended features.

3.2. Structure of the Traditional Algorithm

We created a power estimation algorithm to be used as a competitor algorithm in both accuracy and computational efficiency to the NN. It uses a single GNU Radio script to produce signals with the same parameters as the data generation for the NN script, identify their frequencies, and finally produce the same output as the NN script. Below describes first the signal generation of the script, then the analysis that outputs the used and open frequencies. This method works similarly to the signal detection block used in (O'Shea et al., 2007). However we are not concerned about multi-threading and do not run our blocks directly through a Power Processing Element.

Five signals with the same parameters and a Gaussian noise source are first added to a single total signal. They are then throttled to a sample rate of 166 Msps like the NN script, and a fast Fourier transform (FFT) is applied to the added signals. The complex output of the FFT is separated into eighteen frequency bins, the values of which are converted to squared magnitudes and measured. The first ten bins are intended to each contain a frequency channel shown in Table 1, so in the script, the last eight bin values are no longer needed and dropped. The flowgraph used to generate and collect the data for the power estimation algorithm is shown in Fig. 3. The squared magnitudes are sorted from smallest to highest and the five lowest values are identified as the empty or open channels. The five largest bin values are identified as the channels in-use and are cross-referenced with the frequency list to show the in-use frequencies corresponding to its respective bin. This process is repeated 160 times - the same number of unique channel data files that are fed into the NN at the testing phase.

3.3. Format of Outputs

The output from the NN is in the form of a binary array that classified the channel list as open or occupied. It outputs an array of ten "0's" and "1's", with "1" identifying if the algorithm detects a signal in the specified frequency and "0" identifying the channel as open. The backpropagation loop used to train the NN was iterated 1000 times. Every ten of these iterations, a training error was calculated. These training errors describe the discrepancy between the training data and the output array and ideally, as the NN "learns" the data features fed to it, this error will decrease with respect to increased iterations. At the end of the training phase, we saved the set of final weights to use in the testing phase. This phase used the same script as the training phase, but with only the feedforward steps (excluding the backpropagation steps). The testing phase of the NN yields the percent error showing how often the algorithm incorrectly identifies an open or closed channel, as well as the array of "0's" and "1's" and the training output

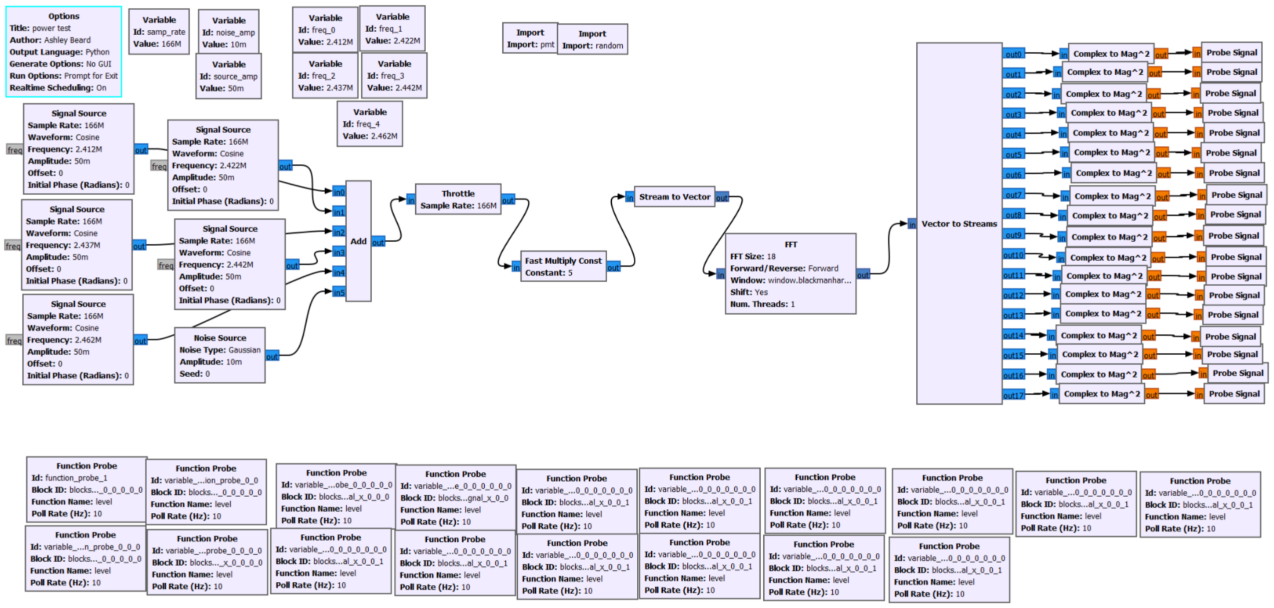

Figure 3. GNU Radio flowgraph of traditional algorithm data generation and collection

expected.

The power estimation algorithm explained in the previous subsection is not separated into a training and testing phase because it does not utilize any ML algorithms. However, to better compare its accuracy to the NN script, it outputs the same format of channel identifier and the average percent error of its estimation is calculated.

4. Analysis

4.1. Analysis of Percent Errors

When comparing the percent errors that are given by the validated NN and the power estimation (non-NN) algorithm, we show that the NN calculates a smaller percent error than the non-NN at a signal-to-noise ratio (SNR) = 44 dBm. Typical wireless networks need an SNR of at least 18 dBm to provide even a low data rate of 6-18 Mb/s (Jevremovic); with greater than 40 dBm being considered "excellent signal". This SNR is the limit at which these two algorithms can differentiate the simulated signal and noise sources from GNU Radio, however, the NN is more accurate at this limit than the non-NN, making it a better choice for channel identification software. The NN learning at an SNR within the range of what is ideal for most wireless networks shows that the algorithm succeeds within what is expected in real-world applications of a channel identifier.

The NN outputs a training error for its final training epoch of 0.1814 and a final test phase error of 0.3383 or 33.83%. The power estimation algorithm output an error of 0.5088 or 50.88%. Our accuracy goal for the stochastic gradient descent algorithm is a correct identification at least 90% of the time or an incorrect identification 10% of the time. The testing phase of the stochastic gradient descent algorithm achieved an accuracy of 66.17%, falling short of the goal by 23.83%. The secondary goal of the stochastic gradient descent algorithm is to accurately identify channel use more often than the power estimation algorithm. The power estimation algorithm achieved a percent accuracy of 49.12%, demonstrating that using a stochastic gradient descent algorithm with ML in a channel identifier offers a more advantageous accuracy than a simple power estimation algorithm without ML.

The accuracy of the NN can most likely be improved to our initial goal of at least 90% by optimization of the learning rate hyperparameter. The learning rate value is a hyperparameter of gradient descent learning and presents a strong impact on the convergence or divergence in optimizing the learning algorithm (Bottou & LeCun, 2003)(Darken et al., 1992). To increase the chances of resulting in a global optimum, the learning rate must usually have an accuracy to within a factor of two (Bengio, 2012b). This will allow a global minimum in error calculations during the training phase as opposed to finding only a local minimum. It is worth investigating in the future whether our stochastic gradient descent algorithm converged to a local minimum, resulting in lower accuracy.

Another possible cause of a lower accuracy could be ex-

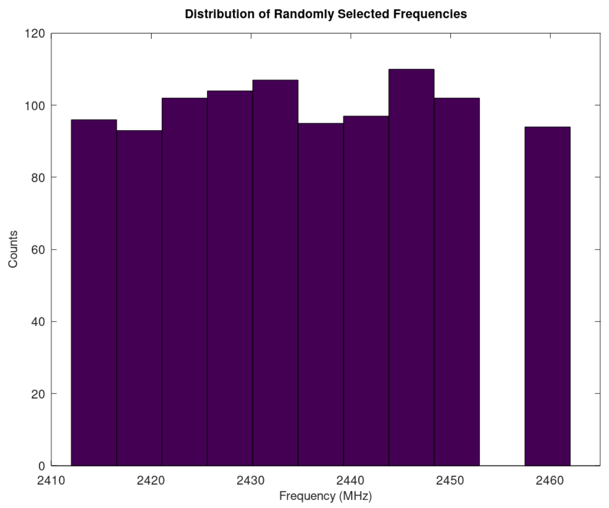

Figure 4. Bin Counts as a function of frequency in MHz. There are 11 bins between a range of 2410 MHz and 2465 MHz. The plot includes data from 200 files each with 5 frequency choices for a total of 1000 frequency choices plotted. The bin values only vary from a median value of 100 counts by ± 10 counts, providing a relatively precise discrete uniform distribution, and also providing an account of unbiased random selection of signal frequencies.

plained by spectral bias. Lower frequencies in a NN data feed are more robust to random weight changes. Therefore, lower frequencies are learned first and higher frequencies are learned late in the optimization process (Rahaman et al., 2019). Since we are using signals in the GHz range, learning may be less efficient than if the network was fed frequencies from a lower frequency bandwidth such as the Medium Frequency (MF) band in the 300 kHz to 3,000 kHz range.

4.2. Testing for Randomness

The randomization functions we used for selecting the frequencies of the radio signals were tested for "true randomness" by plotting the chosen frequencies into a histogram (Fig. 4). The internal Python *random* selection functions we used are not truly random, but they need to be a high enough degree pseudo-random generator so there is less likely to be any selection bias in the data creation process. A selection bias would cause the NN algorithm to give inaccurate responses to input with varying types of features (Lebiere et al., 2013) (Bird et al., 2020).

A set of 200 generated data files were used in this analysis with each file containing a random selection for a subset of five frequencies out of the North American channel list in the 2.4GHz ISM band (3_f, 2016). The histogram shows a discrete uniform distribution of frequencies or amplitudes. There is a gap in the 2453 MHz to 2457 MHz bin because our list of frequencies to choose from did not include anything between 2452 MHz and 2462 MHz. The range of frequencies in this gap could be removed from the histogram and would not detract from the uniform distribution shape. All the bin counts fall within 100 ± 10 counts and subsequently, we can say that the *random* Python function chose frequencies without showing any detrimental selection bias. An advantage of having randomly selected frequencies is the prevention of the NN overfitting the data features and therefore the prevention of the necessary incorporation of regularization techniques such as dropout or L2 and L1 regularization in the NN algorithm.

4.3. Analysis of Computational Cost

The following is an analysis of the computational costs of both the traditional algorithm and the NN algorithm. In minimizing the computational cost of an open channel identifier, one is also helping decrease the energy expenditure of the spectrum sensing mechanism (Zhou et al., 2018). This paper endeavors to analyze the theoretical computational cost of the NN model proposed and compare it to that of the classical power estimation algorithm. However, analyzing the system specific energy costs of these is beyond the scope of this paper. For the analysis of the time complexity, we are not including the calculations performed to preprocess the input data before being fed into the NN. Again, a simple demonstration of the flow of the NN and traditional algorithms in terms of Python calculations can be seen in Figs. 2 and 3.

The traditional power estimation algorithm includes a *numpy sorted* function, GNU Radio's FFT algorithm, and a conversion of complex to magnitude squared floats.

The *sorted* function in Python uses the Timsort algorithm which has a time complexity $O\left(Nlog_2N\right)$ (Python, 2018). The FFT algorithm in GNU Radio has a time complexity of $O\left(Nlog_2N\right)$ (Loan, 1992). Finding the magnitude squared of the complex input includes calculating the complex conjugate shown in (2) and multiplying by the complex components. This leaves the computational complexity to be $O\left(N\right)$.

$$||Real + Imag(i)|| = \sqrt{I^2 + Q^2} \tag{2}$$

Substituting the dimensions of the input data array for the values of N, we arrive at the following total computational complexity by adding each algorithm's complexity together:

$$\begin{aligned} O\left(Nlog_2N\right)+O\left(Nlog_2N\right)+O\left(N\right)= \\ O\left(1024log_2\left(1024\right)\right)+O\left(1024log_2\left(1024\right)\right) \\ +O\left(1024\right)=O\left(21,504\right) \end{aligned} \tag{3}$$

The NN computational complexity primarily consists of matrix multiplication and element-wise scalar operations. The time complexity is calculated for each layer of the NN as a matrix multiplication except the first layer, which is only defined as the input array. The second through fifth layers are calculated as the matrix product of the previous layer and weight, for example, the second layer is calculated as the product of the first layer "layer0" and first weight "syn0". Typically, the complexity is calculated for the multiplication of square matrices as $O\left(n^3\right)$ with matrix dimensions of $n \times n$ (Alon et al., 2012) (Cormen et al., 2009). However, the layers in our FF network involve multiplication of rectangular matrices which do not possess dimensions of n and n^α. Therefore the total time complexity for the five-layered FF network will be $O\left(abc\right)+O(def)+O\left(ghi\right)+O\left(jkl\right)$ where a and b are the dimensions of the first layer, "layer0", b and c are the dimensions of the first weight "syn0", d through f are the dimensions of the second layer and weight "layer1" and "syn1" respectively, and so on to the fourth layer and weight represented by j, k, and l (Cormen et al., 2009).

By substituting the dimensions of the layer arrays with their associated variable placeholders: as described in Fig. 2, we arrive at the total computational complexity for the FF NN algorithm to be shown in the calculation in (4).

$$\begin{aligned} O\left(abc\right)+O\left(def\right)+O\left(ghi\right)+O\left(jkl\right)= \\ O\left(15\left(1024\right)\left(15\right)\right)+O\left(15\left(15\right)\left(1000\right)\right) \\ +O\left(15\left(1000\right)\left(15\right)\right)+O\left(15\left(15\right)\left(1\right)\right) \\ =O\left(905,625\right) \end{aligned} \tag{4}$$

Each layer calculates a matrix multiplication and an activation function which acts as an element-wise operation. Compared to models in (O'Shea & West, 2016) and (Solanki et al., 2021), our 5-layer FF network is less computationally intense. In particular, the Keras model from O'Shea and West has 13 layers, making it much more computationally intense than ours. For a machine limited in computational power, one needs to balance their desired NN accuracy and length of time needed to train the model. The expected operational platform is a battery powered RF network transceiver with limited resources. The backpropagation incorporated in our calculations is a large portion of the computational cost of the network. We feel a network would be trained externally and then weights would be sent to the cognitive radio, to prevent over-taxing the system with training.

5. Conclusions

Both algorithms did not obtain the goal for percent error of 10% or less. There are several possibilities for the lower-than-expected accuracy of our NN. These may include but are not necessarily limited to spectral bias, a loss of information when taking the norm of the input, or unfortunate random initial definitions of the weight values which lead to diverging or only settling at a local minimum during calculating error in backpropagation.

For the purposes of this paper, an extremely simple model was created to simplify the debugging, training, and testing phases of the NN algorithm. To ensure the data generation model is applicable to real-world scenarios, parameters such as free space path (f.s.p.) loss and penetration loss coefficients can be used for cooperative sensing schemes, described in (Zeng et al., 2010), where loss coefficients are provided for three different locations of receivers and where each received signal is run through the algorithm. The outputs are then combined to become more accurate in signal identification with varying levels of attenuation.

An investigation in the advantages of modifying this NN algorithm to be designed for calculations in the complex domain is in order. Designing a complex NN would require an activation function that maps the input data not to the range $[0, 1]$ in the set of real numbers $(\mathbb{R})$ like the sigmoid function used in this example, but to a set of complex numbers $(\mathbb{C})$ (Zimmermann et al., 2011). A hypothesized advantage of continuing calculations using complex numbers until the end result is the regained information of the phases of the wave sources and data features that were lost when the complex numbers were initially computed into magnitudes at the start of our algorithm. This could potentially lead to either a quicker convergence to a local minimum error during training, a higher accuracy during training and validating, or both.

5.1. Future Work

This project was designed as a NN puzzle-piece to a larger project. This channel identifier can be used as an approach to interference avoidance in DSA broadcasting. As a use case scenario, one wants to broadcast Wi-Fi at a certain channel but our algorithm shows this channel is occupied. The script then lists to them which channel frequencies are open to pick from. A DSA policy generator can have this NN script consume raw, real-time signal data and output frequencies as free or occupied. This answer from the NN will be fed in as a parameter into the policy generator which will then tell the user or radio how to operate within a particular bandwidth.

In the future, we intend to continue the digital signal model with a closer real-world relationship and to improve computational efficiency. One area of continued research is optimization of the NN through linear activation functions. This method would decrease computational expense through the elimination of exponential functions.

Acknowledgements

We want to thank Raymond Shaw, Gary Coffey, Alejandro Fuste and Dr. Analee Miranda whose comments and suggestions helped clarify and improve this manuscript.

This manuscript reflects independently funded research and development by Spectrum Bullpen, LLC.

The Python and GNU Radio flowgraphs as well as the dataset used for this project are openly accessible at this link: https://github.com/AABeardSB/channelEst.

References

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

Agarap, A. F. Deep learning using rectified linear units (ReLU). 2019. URL https://arxiv.org/abs/1803.08375.

Alhassan, W., Taylor, A. R., and Vaccari, M. The FIRST classifier: Compact and extended radio galaxy classification using deep convolutional neural networks. *Monthly Notices of the Royal Astronomical Society*, 480:2085–2093, October 2020.

Alon, N., Shpilka, A., and Umans, C. On sunflowers and matrix multiplication. In *IEEE 27th Conference on Computational Complexity*, pp. 214–223, Porto, Portugal, 2012.

Barlow, H. B. Unsupervised learning. In Hinton, G. and Sejnowski, T. J. (eds.), *Unsupervised Learning: Foundations of Neural Computation*, pp. 1–17. MIT, Cambridge, MA, USA, 1999.

Bengio, Y. Practical recommendations for gradient-based training of deep architectures. In Montavon, G., Orr, G. B., and Müller, KR. (eds.), *Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science*, volume 7700, pp. 437–478. Springer, Berlin, Heidelberg, 2012a. ISBN 978-3-642-35288-1.

Bengio, Y. Deep learning of representations for unsupervised and transfer learning. In *JMLR: Workshop on Unsupervised and Transfer Learning*, pp. 17–37, 2012b.

Bird, J. J., Ekart, A., and Faria, D. R. On the effects of pseudorandom and quantum-random number generators in soft computing. *Springer: Soft Computing*, 24:9243–9256, 2020.

Bottou, L. and LeCun, Y. Large scale online learning. In *Advances in Neural Information Processing Systems 16*, Vancouver, BC, Canada, 2003.

Cao, Y. and Gu, Q. Tight sample complexity of learning one-hidden-layer convolutional neural networks. In *33rd Conference on Neural Information Processing Systems*, Vancouver, BC, Canada, 2019.

Cheng, Q., Shi, Z., Nguyen, D. N., and Dutkiewicz, E. Sensing OFDM signal: A deep learning approach. *IEEE Transactions on Communications*, 67(11):7785–7798, November 2019.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. *Introduction to Algorithms*, chapter 28, pp. 813–842. The MIT Press, Cambridge, MA, USA, 3 edition, 2009.

Darken, C., Chang, J., and Moody, J. Learning rate schedules for faster stochastic gradient search. In *Proceedings of the 1992 IEEE Workshop*, 1992.

del Rio, A. Lopez, Martin, M., Perera-Lluna, A., and Saidi, R. Effect of sequence padding on the performance of deep learning models in archaeal protein functional prediction. *Nature: Scientific Reports*, 10, September 2020.

Duan, L., Gao, L., and Huang, J. Cooperative spectrum sharing: A contract-based approach. *IEEE Transactions on Mobile Computing*, 13:174–187, 2014.

Gao, J., Yi, X., Zhong, C., Chen, X., and Zhang, Z. Deep learning for spectrum sensing. *IEEE Wireless Communications Letters*, 8(6):1727–1730, December 2019.

Han, X., Xue, L., Shao, F., and Xu, Y. A power spectrum maps estimation algorithm based on generative adversarial networks for underlay cognitive radio networks. *MDPI Sensors*, 20(311), January 2020.

Jevremovic, V. 7 key factors to consider when designing Wi-Fi networks. Technical report, iBWave.

Jin, K. H., McCann, M. T., Froustey, E., and Unser, M. Deep convolutional neural network for inverse problems in imaging. *IEEE Transactions on Image Processing*, 26 (9):4509–4522, September 2017.

Lebiere, C., Pirolli, P., Thomson, R., Paik, J., Rutledge-Taylor, M., Staszewski, J., and Anderson, J. R. A functional model of sensemaking in a neurocognitive architecture. *Computational Intelligence and Neuroscience*, July 2013.

Liao, Y., Wang, T., Song, L., and Han, Z. Listen-and-talk: Protocol design and analysis for full-duplex cognitive radio networks. *IEEE Transactions on Vehicular Technology*, 66(1):656–667, January 2017.

Loan, C. Van. *Computational Frameworks for the Fast Fourier Transform*. SIAM, Philadelphia, PA, USA, 1992.

Lukic, V., de Gasperin, F., and Brüggen, M. ConvoSource: Radio-astronomical source-finding with convolutional neural networks. *MDPI Galaxies*, 8(3), 2020.

Mathai, V. and Sagayam, K. M. Comparison and analysis of channel estimation algorithms in OFDM systems. *International Journal of Scientific & Technology Research*, 2(3):76–80, March 2013.

Nasser, A., Hassan, H. A. H., Chaaya, J. A., Mansour, A., and Yao, K-A. Spectrum sensing for cognitive radio: Recent advances and future challenge. *MDPI Sensors*, 21, March 2021.

Nielson, M. A. *Neural Networks and Deep Learning*, chapter 2. Determination Press, 2015.

O'Shea, T. J., Clancy, T. C., and Ebeid, H. J. Practical signal detection and classification in GNU Radio. In *Proceeding of the SDR 07 Technical Conference and Product Exposition*, 2007.

O'Shea, T. and West, N. Radio machine learning dataset generation with GNU Radio. In *Proceedings of the 6th GNU Radio Conference*, Boulder, CO, USA, 2016.

Python. Time complexity, August 2018. URL https://wiki.python.org/moin/TimeComplexity.

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.

Rodriguez, O. and Dassatti, A. Deep learning inference in GNU Radio with ONNX. In *10th Annual GNU Radio Conference*, 2020.

Shannon, C.E. Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21, 1949. doi: 10.1109/JRPROC.1949.232969.

Shellhammer, S. J. Spectrum sensing in IEEE 802.22. In *1st IAPR Workshop on Cognitive Information Processing*, Santorini, Greece, 2008. Qualcomm Inc.

Solanki, S., Dehalwar, V., and Choudhary, J. Deep learning for spectrum sensing in cognitive radio. *MDPI Symmetry*, 13(147), January 2021.

Thilina, K. M., Choi, K. W., Saquib, N., and Hossain, E. Machine learning techniques for cooperative spectrum sensing in cognitive radio networks. *IEEE Journal on Selected Areas in Communications*, 31(11):2209–2221, November 2013.

Tumuluru, V., Wang, P., and Niyato, D. A neural network based spectrum prediction scheme for cognitive radio. In *2010 IEEE International Conference on Communications*, Cape Town, South Africa, 2010.

Zeng, Y., Liang, Y., Hoang, A. T., and Zhang, R. A review on spectrum sensing for cognitive radio: Challenges and solutions. *EURASIP Journal on Advances in Signal Processing*, 2010.

Zhou, X., Sun, M., Li, G. Y., and Juang, B-H. Intelligent wireless communications enabled by cognitive radio and machine learning. *China Communications*, 15:16–48, December 2018.

Zimmermann, H. G., Minin, A., and Kusherbaeva, V. Comparison of the complex valued and real valued neural networks trained with gradient descent and random search algorithms. In *19th European Symposium on Artificial Neural Networks*, pp. 213–218, Bruges, Belgium, April 2011.