RF Fingerprinting of 5G Cellular Devices

Justin Holzer
Electrical & Computer Engineering
Worcester Polytechnic Institute
Worcester,USA
jtholzer@wpi.edu

Isha Chidrawar
Computer Science
Worcester Polytechnic Institute
Worcester,USA
ichidrawar@wpi.edu

Simranjeet Ravinder Saini
Computer Science
Worcester Polytechnic Institute
Worcester,USA
srsaini@wpi.edu

Justin Weintraub
Computer Science
Worcester Polytechnic Institute
Worcester,USA
jweintraub@wpi.edu

Abstract* — 5G Networks are being deployed nationwide after significant investments in network security. Unfortunately, there exists ways to spoof the network. This article suggests a technology to distinguish the different transmitters on a network in an effort to help with network security. This paper explores various Artificial Intelligence (AI) methods to distinguish transmitters based on unique RF impairments from each transmitter. The AI methods use raw I/Q samples to identify one transmitter from another. Since each transmitter has a slightly different analog RF chain, these RF impairments create a unique fingerprint for each transmitter by altering the I/Q data in a slight but distinguishable pattern. The different RF impairments are discussed and modeled. This paper looks at two different synthetic data generation approaches to produce data for training and testing AI methods. This paper discusses the different AI and Machine Learning (ML) methods along with each method's accuracy in transmitter classification using raw I/Q samples. The most effective AI method discussed in this paper, an 18-layer Deep Neural Network (DNN), is presented along with the classification accuracy from the DNN training, validation and testing.

I. INTRODUCTION

For many commercial cellular providers (in the United States, for example, these include Verizon, AT&T, T-Mobile and others), an ardent desire to provide secure communications drives the need to uniquely identify users on their network. Figure 1 illustrates several cellular phones in use in a single cell of a commercial cellular network. While users self-identify as part of the 3GPP protocol, a concern over spoofing (presenting user ID that is not one's own) leads cellular and internet providers to catch the (sophisticated) spoofing attacks on their network. The spoofing attack is illustrated in Figure 1by the Rogue Phone. The Rogue Phone presents itself to the Cellular Base Station as another user and thereby makes it difficult for the Cellular Base Station to reject it from the network. If there were a way to uniquely identify users based solely on the RF transmission properties of the radio, this would give the network a chance to identify any anomalous devices, keep them off the network and maintain network security. If there were a way to discover the RF fingerprint of the wireless device, this would provide the network provider with valuable insight into spoofing attempts and could be used to deny unauthorized network access.
The approach shown in this paper can be applied to the Rogue Phone scenario, although we do not specifically try to find the Rogue Phone separately from the valid phones. Networks and users are equally concerned about "Stingray attacks" or "IMSI grabbers" [4][5][6]. The concern over the ability to be tracked without the user being aware leads to some privacy concerns for cell phone users. The ability to uniquely identify a transmitter will allow cell phone users to maintain their privacy and not give away information unaware.

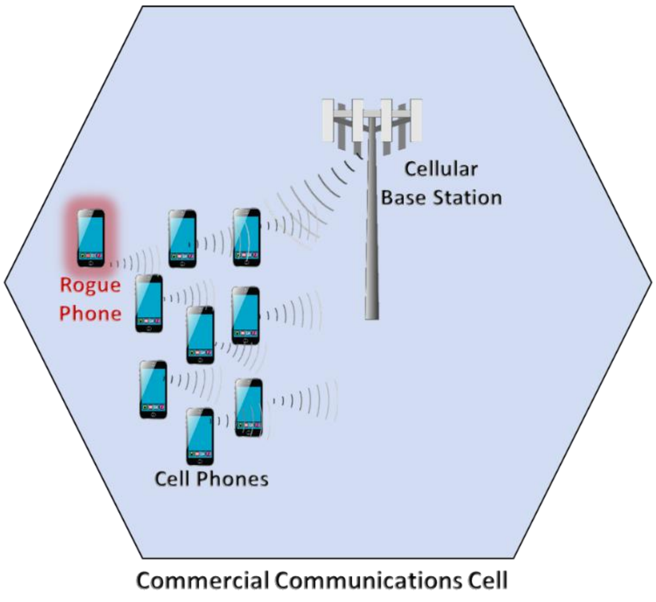


Figure 1 ***Cell Phone Communications Scenario*** *- Commercial communications cell is vulnerable to spoofing attacks from rogue phone*

II. BACKGROUND

The ability to fingerprint wireless devices has become an interesting topic for identifying hardware-specific features on radios. RF fingerprinting can be used to enhance security of networks or wireless connections. The RF impairments and variations on the RF fingerprint from one RF transmitter to another RF transmitter can be ridiculously small and difficult to pick out. The changes to the waveform are too small for a human to identify and difficult to impossible for current approaches (e.g. matched filters) even with large quantities of data. As depicted in [1] and illustrated in Figure 2, the hardware features from the analog front end of the transmitter may come from RF bandpass filters, Frequency mixers, power amplifiers, Digital to Analog converters or other RF components. These components may introduce any number of effects, including non-linear distortions, I/Q imbalance, DC offset, carrier frequency offset or general phase noise. Figure 1 illustrates how some of these identifying features are introduced to the analog signal in hardware. For this project, we will be focusing on 5G waveforms for the transmit signal.

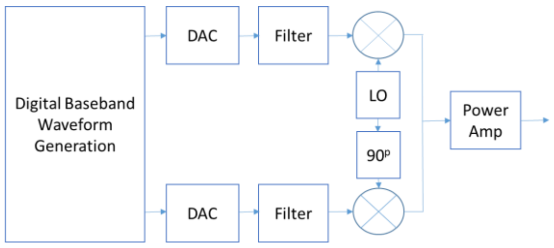


Figure 2: ***RF Chain Model –*** *Simplified model of RF Chain that imposes RF impairments on transmit waveform by transmitter [1]*

Several challenges must be addressed to successfully fingerprint RF transmitters. High Signal to Noise Ratio (SNR) is essential to detect the fingerprint from the noise. When SNR is low, the noise will overpower the fingerprint (and potentially the signal) and will mask the ability to successfully discover the transmitter fingerprint. This research does not investigate the threshold for SNR to satisfy performance requirements. Also, the fingerprinting process may require more data than is available for each transmitter. For this research, we assume the ability to collect ample data on each transmitter.

III. METHODS

A. *Data Set #1 – First Approach to Data Synthesis*

To test and exercise the proposed AI methods and investigate which methods work best, two different approaches were used to generate data for this project. Data set #1 was simulated in Matlab by modulating pseudorandom number (PN) sequences with Quadrature Phase Shift Keyed (QPSK) modulation and then the RF impairments were applied in software to the I/Q data (vectors of complex numbers). This data set represents a generic RF transmission from any relatively modern wireless communication device since QPSK is a common modulation scheme.

It is assumed that the RF impairments for each transmitter are constant across the entire data set. The parameters for the simulated data were randomly selected from a uniform distribution over the range of values specified in Table 1.

Table 1 ***RF Impairment Parameters*** *- RF Impairments introduced to data set #1. Values for each transmitter were chosen randomly from the given value range*

RF Impairment	**Units**	**Value Range**
Phase Offset	Degrees	0 to 30
Frequency Offset	Hz	1 to 100
Phase Noise Level	dBc/Hz	-1 to -50
Amplitude Imbalance	dB	1 to 10
Phase Imbalance	Degrees	1 to 10
DC offset applied to real (I)	Linear gain	0.0001 to 0.1
DC offset applied to imaginary (Q)	Linear gain	-0.0001 to -0.1
Amplifier Gain	dB	0 to 50

This first set of data was generated using a Matlab script. The script is given in a supplementary document (see S1).

B. *Data Set #2 – Second Approach to Data Synthesis*

Data set #2 was generated to look a lot like an actual 5G cellular data transmission. The 5G standard [7] sets the physical layer with very specific parameter options so that the cell phone and base station know what waveforms to expect. To maintain compliance with the 5G commercial standard, the MathWorks 5G Toolbox was leveraged to generate several waveforms with various configurations to simulate various aspects of 5G wireless communication. Included in the generated data was Uplink and Downlink waveforms, waveforms with various bandwidth, various subcarrier spacing and other pertinent configuration parameters. We focused our effort on the waveform described by the list of parameters given in Table 2. Figure 3 **FFT of generated 5G Waveform** - 5G Cellular Waveform used to transmit from SDRs illustrates the frequency domain representation of the 5G signal used in our data synthesis.

Table 2 ***5G Parameters*** *- 5G Parameters used for waveform synthesis of data set #2*

5G Parameter	**Value**
MCS	QPSK R=1/3
Subcarrier spacing (kHz)	15
Channel Bandwidth (MHz)	5
Subframes	10
Layers	1
Cell Identity	1
Allocated resource blocks	25

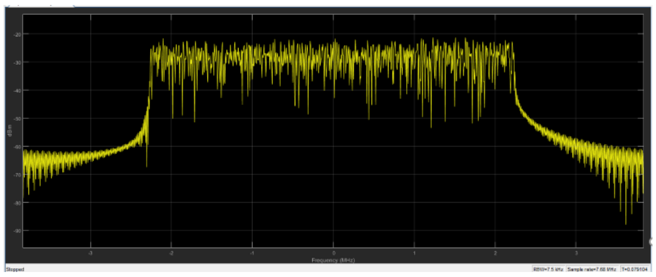


Figure 3 ***FFT of generated 5G Waveform*** *- 5G Cellular Waveform used to transmit from SDRs*

We used software defined radios (SDRs) to transmit the waveform shown in Figure 3. This allows us to get RF impairments on the data from actual analog RF components on the 5G waveform. While the SDRs may not act exactly the same as cell phones, an SDR is a reasonable surrogate transmitter that allows exploration of different AI methods. These methods are explained later in this paper. Figure 4 illustrates how one of the SDRs was connected to a laptop. For our data collection, ADALM-PLUTO SDRs were controlled via Matlab to transmit and receive the data from the laptop.

Figure 4 ***Software Defined Radio (SDR) Data Synthesis*** *– Hardware setup of individual SDR for 5G data transmitter and receiver*

It is important to note that our data generation does not have any channel effects because our transmitter was wired to our receiver with an RF coax cable (with SMA connectors). This allows us to focus on the fingerprint of the transmitter and not get confused by the wireless channel effects, such as fading and multipath. While those effects are important to take into consideration, those experiments are left for a future research effort.

IV. AI METHODS OVERVIEW

Two Approaches to Data Usage for AI Methods

For data usage method #1, after bringing our data from data set #1 to Python, we converted it into a usable train/test format with 80% of the data being allocated to training. Using SKLearn (python library) we tested various machine learning algorithms to explore the capabilities of each method. To format our data for SKLearn, we converted the waveform of complex (real and imaginary or I/Q) into a single array. The array contains real and imaginary interleaved as follows.

```
Waveform = [Real_sample_1,
            Imaginary_sample_1,
            Real_sample_2,
            Imaginary_sample_2,
            …
            Real_sample_N,
            Imaginary_sample_N]
```

The 1d array was split up into selections of 1000 continuous samples. Given the ease of synthesizing data points, over 1e6 samples were used, giving many inputs to train each classifier. The following classifiers were employed in this research: Ada Boost, Random Forest, Logistic Regression, Neural Network, Support Vector Machine with a RBF Kernel, Support Vector Machine with a Polynomial Kernel, Bernoulli Naive Bayes, Gaussian Naive Bayes.

A. AdaBoost

The AdaBoost (AB) (short for Adaptive Boosting) classifier utilizes a method called ensemble learning. Ensemble learning is based off of generating multiple hypotheses and predicting based on their collective result. The type of ensemble learning AB uses is called boosting. With boosting, an initial hypothesis is calculated with a weight of 1 for all examples. The hypothesis is progressively copied and altered as the algorithm updates to better fit the results of the inputs, generating many hypotheses until the maximum is reached. AB consists of a unique property of increasing the accuracy of hypotheses that perform similarly to guessing, increasing the accuracy of the overall prediction. To do this, it replaces said hypothesis with a hypothesis that classifies the training data perfectly [16]. AB's properties related to checking multiple hypotheses have the potential to increase accuracy, making it a useful classifier to test.

B. Random Forest

Random Forest (RF) is used for classification as well as regression problems. It also takes the approach of ensemble learning. In the case of RF, it combines results from multiple decision trees and the majority voting is considered for the classification result [17]. A Random Forest method is illustrated in Figure 5

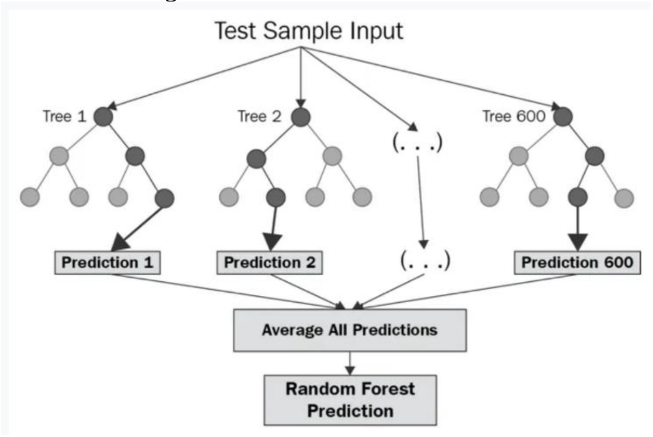


Figure 5 ***Random Forest Structure*** *- The structure for how the Random Forest takes in a sample and converts it into a prediction [17]*

RF works well with categorical as well as continuous data. Also, unlike many other classifiers it is not affected much by noise in the data and is able to handle missing values effectively. All these characteristics contribute to the likelihood of a high level of accuracy when employing this method.

C. Logistic Regression

Logistic Regression (LR) is a supervised learning classification algorithm used for classification. It is quite simple in nature, using a logarithmic based equation to determine if an input belongs to the result or not. Outputs to one side of the equation return 0/false while to other returns 1/true. The weights for said equation were updated with each input, being regularized so the weights wouldn't change a considerable amount. For LR to work with our data, a one vs. rest approach was used, with one transmitter being compared to the rest for each transmitter. Due to LR's relative simplicity, it was used as the first classifier to test data set #1

D. Naive Bayes

Naive Bayes (NB) is a type of classifier based on probability. It applies Bayes' theorem, the theorem that calculates the conditional probability of an event based off its relationship with another event. The classifier finds the conditional probability of each result (in our research, for each transmitter) based on the given input and selects the result with the highest probability. NB is known to be fast and scalable when it comes to training and is unique in the methods we explored because it is a probability-based classifier. We looked at two different approaches of NB, namely Gaussian NB and Bernoulli NB. Multinomial was not used because of its reliance on all positive data points. Bernoulli NB has the drawback of expecting features to be represented in binary. Our collected data had points in a range, so it was unlikely that this classifier would succeed. Gaussian NB works with continuous data, making it more likely to succeed. Gaussian assumes the data is normally distributed, so the higher the accuracy the higher the chance of our data points being normal, as stated by the central limit theorem.

E. Support Vector Machine

The "Support Vector Machine" (SVM) is a supervised machine learning technique that may be used to solve classification and regression problems. SVM is one of the most robust and accurate classification algorithms. The objective of SVM is to generate a hyperplane that splits a data set into two or more classes as optimally as possible. Each data item is plotted as a point in n-dimensional space (where n is the number of features you have), with the value of each feature being the value of a certain coordinate in the SVM algorithm. SVM then accomplishes classification by locating the hyperplane that clearly distinguishes the two classes. The support vector is the line that goes through the nearest data points from the hyperplane. And the distance between the line and the support vectors is called the margin. To get the optimal hyperplane, we need to select the one with the maximum margin.
For a space of n dimensions, we get a n-1 dimensional hyperplane. When the data set is nonlinear, we do not have to implement it manually but can be done with the kernel trick. The kernel is a collection of mathematical functions used by SVM algorithms that takes the non-separable problem and converts it to a separable problem as shown in the Figure 6 that takes non-linear data points with 2-dimensional space and converts to a 3-dimension space to get a hyperplane or a decision surface that divides the data more clearly.

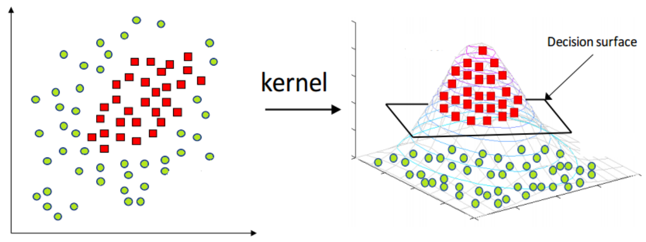


*Figure 6. **Support Vector Machine Dimension Conversion –** SVM's kernel converting a 2-dimensional space to 3-dimensional space to find a clear decision surface*

In our implementation we used two different kernels, RBF (Gaussian radial basis function) and polynomial. In the polynomial kernel, to estimate similarity, it examines not only the provided attributes of input samples, but also combinations of these features. Polynomial has proven to be popular in natural language processing and image processing. For non-linear data, RBF is frequently a good option. When there is no prior information on data, it assists in efficient separation. RBF requires only one parameter and there are good heuristics to find it. Hence it is more likely to give better results than a polynomial kernel.

F. Neural Network

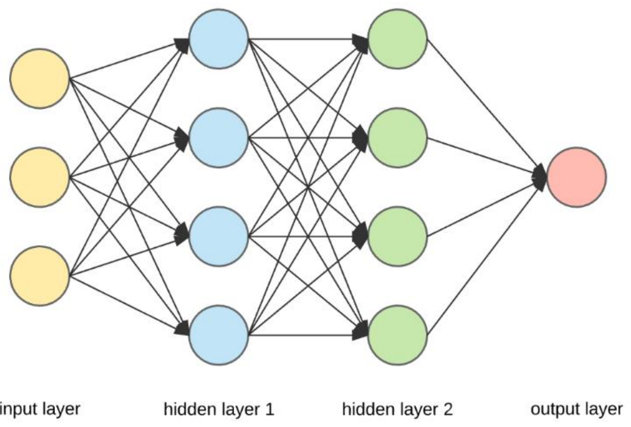


*Figure 7 **Neural Network -** An example of a Neural Network's layers and nodes. [10]*

A Neural Network (NN) is meant to simulate the electrical activity of our brain and nervous system, being composed of nodes that perceive a given input differently. Inputs are passed from one layer of nodes to another, until it eventually reaches the result layer that determines which class to assign the input data. In between the result and input layers are variable hidden layers. These nodes and layers are illustrated in the example shown in Figure 7. After the NN learns whether its decision was right, each node's weights are updated so it could more accurately predict a transmitter in the future. NNs are known for being slow to converge and requiring a large input set to accurately predict results. However, they have the potential to be an accurate classifier, with its ability to detect complex nonlinear relationships between the input and output. For data usage method #1, we implemented a (non-deep) NN. Our simple NN was created using SKLearn, being passed in as input one hidden layer of size transmitters squared. This was done so the network would gradually decrease in size and because it was shown to have the highest accuracy out of any simple methods used for the hidden layers.

G. Deep Neural Network

Deep Neural Networks (DNNs) and other Deep Learning architectures or deep structured learning architectures are "part of a broader family of machine learning methods based on artificial neural networks with representation learning" [8]. A DNN is a network with multiple layers that can model complex non-linear relationships.
The DNN we began with leverages an example DNN from MathWorks [9] as well as learning from a DNN published by Sankhe et al. [1]. Our DNN has 18 layers, as illustrated in Figure 8. The first Convolution NN layer (shown in dark blue) allows us to input a frame of sampled data with complex numbers (real and imaginary) as one unit, much like one would input an image to a convolution NN. This is different from the other methods described above (e.g. NB or LR) because those look at each sample individually. There is information that is lost when individual samples are not processed relative to the previous sample. Likewise, there is information lost when I and Q are processing independently. This leads to a strong advantage that the DNN has over the previously explained methods.
The illustration in Figure 8 of the 18-layer DNN is color-coded to show how multiple layers are repeated in this DNN. The NN architecture used here consists of two convolutional layers (first in dark blue, then in light blue) and three fully connected layers (in green). The idea behind this design is the convolutional NN will learn features independently from the real (I) and imaginary (Q) data. The next CNN combines the real and imaginary parts of the complex data together to extract features that are dependent on the relationship between I and Q – such features include phase information. The fully connected layers (illustrated in Figure 8 as green blocks) classify the data based on the learned features from the convolutional NN layers.

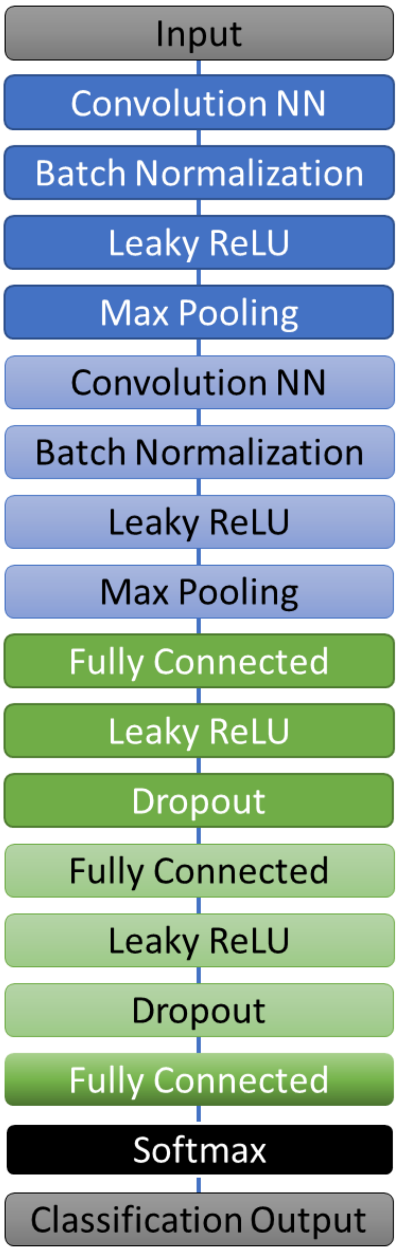


*Figure 8. **DNN Configuration** – We used an 18-Layer DNN for classifying 5G RF transmitters*

Description of DNN Layers

The convolutional layer works by applying a filter on its inputs, transforming the input into a feature map. Doing so allows for the detection of features inside the given input, increasing our network's accuracy [11].
A batch normalization layer individually normalizes a mini batch of data across all observations for each channel. Batch normalization layers are used between convolutional layers and nonlinearities, such as ReLU layers, to accelerate convolutional neural network training and minimize sensitivity to network initialization. [12]
The Leaky ReLU scales the inputs by .01 when they are less than zero, helping to speed up training by scaling inputs closer to 0 [9].

A 2-D max pooling layer performs down sampling by dividing the input into rectangular pooling regions, then computing the maximum of each region [13]. Max pooling reduces computational load and reduces overfitting.
A fully connected layer connects itself to all activations in the previous layer, with the fully connected layer's size being variable [14]. This is primarily used to decrease the size of the inputs.
The SoftMax layer finds the probability of each input being a specific class, with all the probabilities adding up to 1 inside the layer. In our case, each class is associated with a transmitter [15].

I. *Conditioning the Data for the DNN*

The input layer of the DNN requires that the RF sampled data be formatted in a way that the convolutional NN can read in the data. The data is formatted in a 4-dimensional matrix of size M x N x P x Q. M is the number of samples, also called the frame size. For our experiments, M was set to 160, although more research can be done to optimize this number. N is size two, one column for real and one column for imaginary data. P is set to 1. While one could technically eliminate this dimension from the matrix without loss of information, P is maintained for compatibility with the CNN. Q can be set such that all available data is used in the DNN. The trade-off from larger Q is the need for larger processing times. It can also be difficult to obtain more samples.

V. Results

A. *Results from Data Usage Method #1*

To measure the accuracy of our classifiers used with data usage method #1 for both datasets, we took the average percent accuracy of 6 random training/testing instances of each classifier. These random instances consisted of using a transmitter out of our given pool (Ex: 2 random transmitters out of the 15 is chosen each iteration). Without this last step, the accuracy shown would've only been fixed to the chosen transmitters, causing an inaccurate estimate of the accuracy. Some transmitters were shown to be harder to distinguish than others. The number of transmitters utilized varied so we could determine at what number of transmitters we could do accurate classifications to characterize each classifier with different classifiers. These results can be seen in the attached graphs, recording the results of each classifier at 2, 3, 5, 10, and 15 transmitters. We used 80% of our data for training and 20% for testing.

Table 3 Accuracy of classifiers in data usage method #1 with dataset #1– results shown for variable numbers of transmitters. Gray highlights results that are as good as guessing, while red highlights strong results.

Transmitters	Ada	Forest	Neural	RBF SVM	Poly SVM	Naive Bernoulli	Naive Gaussian
2	74%	86%	50%	44%	50%	40%	90%
3	62%	82%	43%	64%	25%	38%	94%
5	38%	68%	25%	49%	20%	24%	91%
10	18%	55%	18%	34%	16%	19%	80%
15	10%	45%	15%	25%	12%	13%	71%

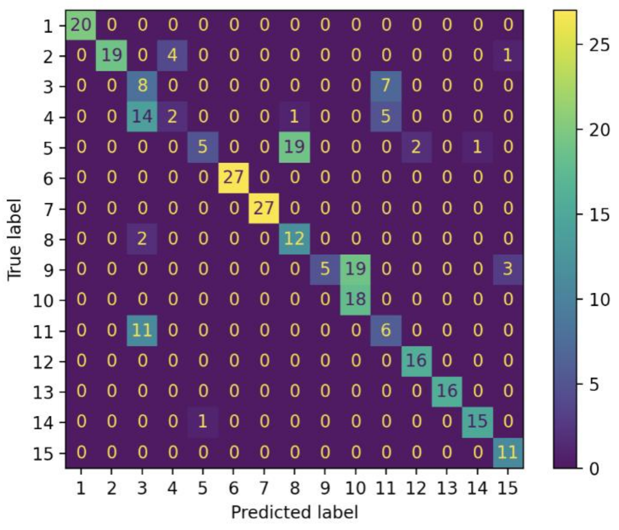


Figure 9 ***Gaussian Naïve Bayes Confusion Matrix*** *- Example confusion matrix of testing accuracy of an instance of Gaussian Naive Bayes with 15 transmitters.*

Table 3 highlights our results for dataset #1 with this method. The Gaussian Naive Bayes (GNB) classifier was shown to have the highest accuracy overall, not dropping below 0.7. The confusion matrix in Figure 9 displays an example of its performance. On 7 out of 15 transmitters, it was able to perfectly classify its fingerprint. Random Forest (RF) had the second highest accuracy overall, with its accuracy not dropping below 0.5 in each graph.
Logistic Regression (LR), Neural Network (NN), Support Vector Machine with a poly kernel (PSVM), and Bernoulli Naive Bayes (BNB) all had results that were as good as guessing for lower number of transmitters, but when the transmitters increased to a substantial amount, their results were shown to be better than guessing. Despite them being better than guessing, only RF, Support Vector Machine with a RBF kernel (RSVM), and GNB produced results greater than 20% at 15 transmitters. And only GNB produced results greater than 50% at 15 transmitters.
When it comes to general trends, most classifiers tended to decrease in accuracy as the number of transmitters increased. An exception to this trend being GNB jumping from 90% to 94.17% accuracy when the transmitters increased from 2 to 3.

Table 4 Accuracy of classifiers in data usage method #1 with dataset #2– results shown for variable numbers of transmitters. Gray highlights results that are as good as guessing, while red highlights strong results.

Transmitters	Ada	Forest	Neural	RBF SVM	Poly SVM	Naive Bernoulli	Naive Gaussian
2	59%	63%	51%	86%	47%	48%	84%
3	41%	44%	34%	84%	34%	32%	81%
4	33%	31%	25%	73%	28%	24%	68%

For dataset #2 the results can be seen in Table 4. In this case, RSVM was the top performer with an accuracy of 72.5% at 4 transmitters, and GNB was the second highest performer with an accuracy of 68.13% for the same amount. LR, NN, PSVM, and BNB have results that are as good as guessing no matter the transmitter amount. We only went up to 4 transmitters here, so to see if these trends would continue the same at a higher number would require a larger amount of data to be gathered.

Datasets #1 and #2 had similar top performers, with RF RSVM and GNB performing adequately in both. However, the top percentages were significantly lower in dataset #2, with RSVM producing an accuracy of 83.89% at 3 transmitters on dataset #2 and GNB having an accuracy of 94.17% with the same number of transmitters for dataset #1. The low performers were similar for the datasets, with LR PSVM and BNB having results as good as guessing for 2 and 3 transmitters in both instances. Adaptive Boosting (ADA) had neither strong nor weak results across each table. ADA and NN were shown to be the slowest algorithms.

B. DNN Results from Data Set #1

Data set #1 (QPSK-modulated data with software-simulated RF impairments) was used to exercise the Deep Neural Network (DNN). The data was split into training data (80% of the data), validation data (10% of the data) and test data (10% of the data). The training and validation data accuracy and loss is plotted in **Error! Reference source not found.**. Figure 11 and Figure 12 illustrate the ability of the DNN to classify the data from Data Set #1. The accuracy is impressive given that 14 transmitters were simulated for this data set.

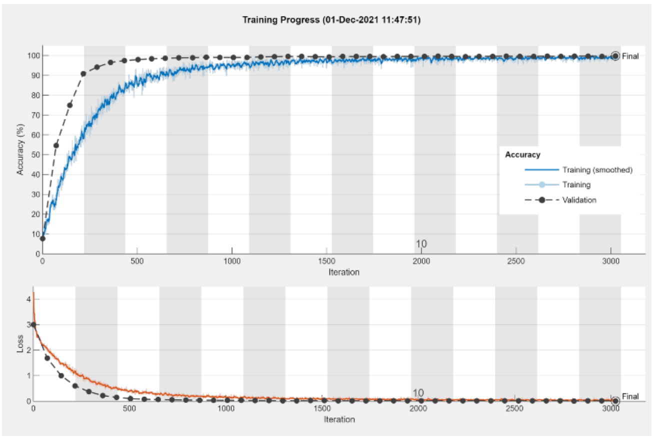
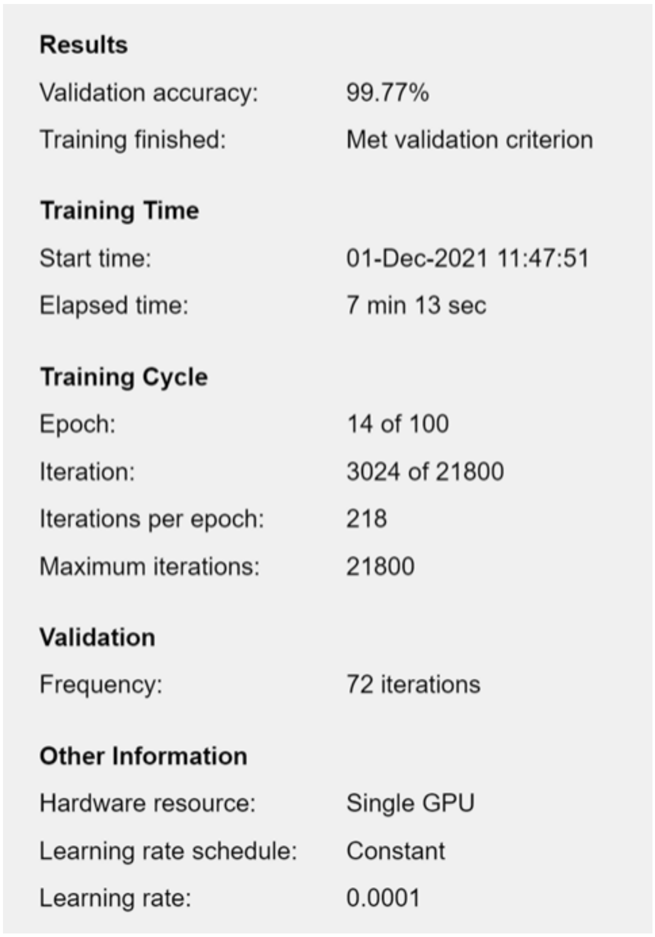


Figure 10 ***Deep Neural Network Training Accuracy and Loss with Data Set #1*** *- Graph visualizing accuracy (blue) and loss (red) in training/testing/validation over iterations*



Results

Validation accuracy:	99.77%
Training finished:	Met validation criterion

Training Time

Start time:	01-Dec-2021 11:47:51
Elapsed time:	7 min 13 sec

Training Cycle

Epoch:	14 of 100
Iteration:	3024 of 21800
Iterations per epoch:	218
Maximum iterations:	21800

Validation

Frequency:	72 iterations

Other Information

Hardware resource:	Single GPU
Learning rate schedule:	Constant
Learning rate:	0.0001

Figure 11 ***Training and Validation Parameters and Results with Data Set #1*** *- Statistics for our results for our DNN with data set #1*

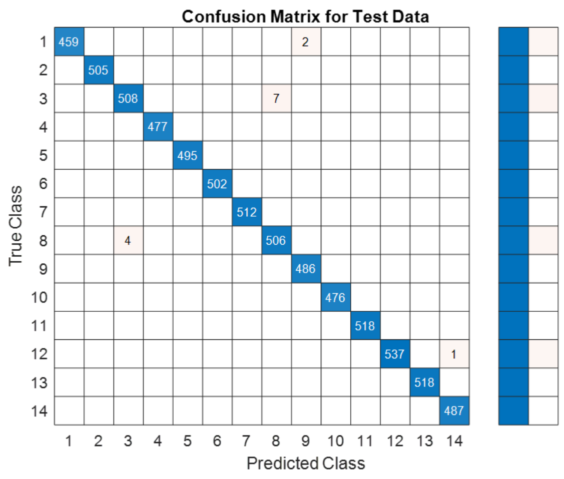


Figure 12 ***DNN Confusion Matrix for Data Set #1*** *- Testing accuracy of data set #1 visualized in a confusion matrix*

For the Test Data from Data Set #1, the test accuracy achieved 99.8%. These results lead us to believe that it is worthwhile to explore the DNN accuracy for Data Set #2 and other more challenging data sets.

C. Results for DNN from Data Set #2

Data Set #2 was used for a number of different experimental runs. As explained above, Data Set #2 contains a simulated 5G uplink (UL) waveform. For one experiment, the Data Set #2 waveform was played through a set of SDRs in order to impose the RF Impairments from each SDR on the waveform. These data sets were then used to test the ability to perform RF fingerprinting of each SDR based on the SDR (analog) RF Impairments. It should be noted that the transmitter and receiver in these SDRs were not syncronized, so the data frames used for each collection, were essentially random draws from the transmit waveform, explained above in the Methods section.
To illustrate the challenge in performing RF fingerprinting, it is instructive to see the results from a data collection where the number of samples was limited. As seen with results presented below, this was mostly an issue of limited training samples. Table 5provides some parameters on the set up of the data for DNN RF Fingerprinting. It can be seen that while the DNN better than a random guess might do for 10 fingerprints, it still leaves a lot to be desired for a reliable fingerprinting capability.

Table 5 ***DNN Parameters and Result using 1 Million Complex Samples from ten SDRs -***
Setup for DNN with data set #2 and 10 SDRs

Parameter	Value
Data Set	Data Set #2
Number of SDRs (Fingerprints)	10
Data Samples from each SDR	1 Million
Training Samples	800k (80%)
Validation Samples	100k (10%)
Test Samples	100k (10%)
AI Method	DNN
Test Accuracy	19%

Some changes were made to improve the DNN for RF Fingerprinting. As shown in Table 6, the number of samples was increased by a factor of 10 and the number of transmitters (SDRs) was reduced to 4. This was done in order to give the model many more data points to train. It also simplifies the classification problem dramatically. Given that the accuracy results increased to 90%, this DNN shows promise and merits further investigation.

Table 6 ***DNN Parameters and Results using 10 Million Complex Samples from 4 SDRs - Statistics*** *for our DNN with data set #2 and 4 SDRs*

Parameter	Value
Data Set	Data Set #2
Number of SDRs (Fingerprints)	4
Data Samples from each SDR	10 Million
Training Samples	8M (80%)
Validation Samples	1M (10%)
Test Samples	1M (10%)
Size of individual data frame	160 samples
AI Method	DNN
Max Epochs	100
Test Accuracy	90%

The training and validation accuracy are illustrated in Figure 13 and Figure 14. By examining the training progress of the DNN with 8 Million complex samples (for training), we see that the DNN does much better. It could also be noted that the training may benefit from more epochs or more training data.

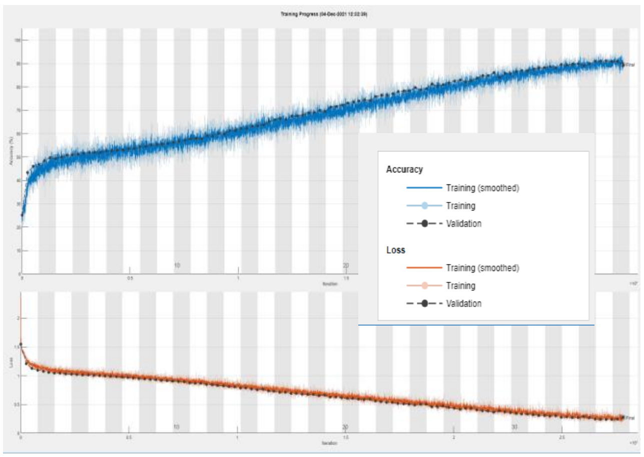
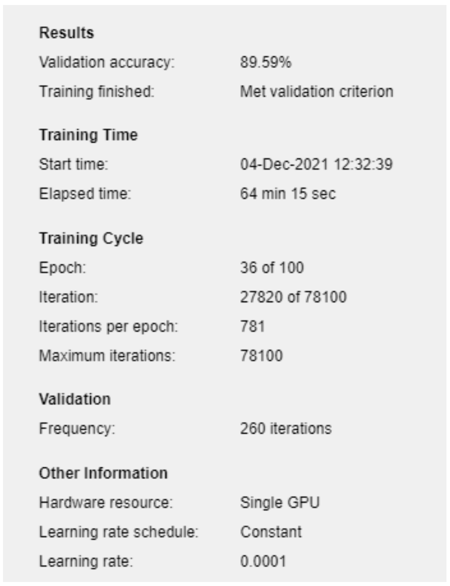


Figure 13 ***Deep Neural Network Training Accuracy and Loss with Data Set #2 and 4 SDRs -*** *Graph visualizing accuracy (blue) and loss (red) in training/testing/validation over iterations*



Results	
Validation accuracy:	89.59%
Training finished:	Met validation criterion
Training Time	
Start time:	04-Dec-2021 12:32:39
Elapsed time:	64 min 15 sec
Training Cycle	
Epoch:	36 of 100
Iteration:	27820 of 78100
Iterations per epoch:	781
Maximum iterations:	78100
Validation	
Frequency:	260 iterations
Other Information	
Hardware resource:	Single GPU
Learning rate schedule:	Constant
Learning rate:	0.0001

Figure 14 ***DNN Results from Training and Validation using 10M complex samples and 4 SDRs*** *- - Statistics for our results for our DNN with data set #2 and 4 SDRs*

Finally, the confusion matrix (see Figure 15) for the DNN with 10M samples and 4 SDRs is a great illustration of how accurate the DNN is for RF Fingerprinting. As mentioned in Table 6, the test accuracy for this DNN run was 90%.

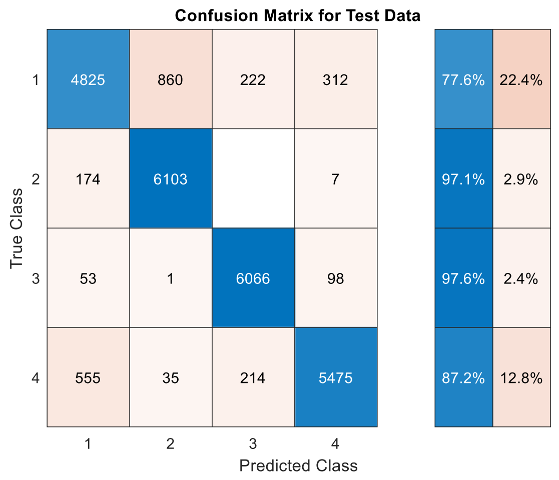


Figure 15 ***Confusion Matrix for DNN of 4 SDRs using 10M Complex Samples*** *- Testing accuracy of data set #2 with 4 SDRs visualized in a confusion matrix*

VI. CONCLUSION

The results seen with the classifiers utilizing data usage method #1 (i.e., Ada Boost, Random Forest, Logistic Regression, Neural Network, SVM and Naive Bayes) were prone to always have imperfect accuracy. With the highest recorded accuracy being Gaussian Naive Bayes (GNB) 94.13% accuracy with 3 transmitters for dataset #1 as seen in Table 3. For both datasets, the low accuracy was in part because of the way the data had to be converted to be inputted to SK-Learn's version of the classifiers. The data samples were taken as individual samples, with the real and imaginary parts considered separately. This means that the phase information is lost from the complex RF samples. The Deep Neural Network (DNN) provided stronger accuracy for both datasets, partly alluding to this point as it maintained the phase information. DNN of course was also a more complex classifier than the rest used, so its higher accuracy was predictable on multiple accounts.

It is interesting to see the performance of the GNB classifier, which performs better than every other classifier in dataset #1 and all but one in data set #2, including Bernoulli Bayes in both instances. This is likely in part because of GNB's use of continuous data and Naive Bayes' ability to make predictions. With RBF Support Vector Machine (RSVM) and Polynomial Support Vector Machine (PSVM) we can see that RSVM outperforms the polynomial PSVM with both datasets. As discussed in this paper, RSVM is more compatible with the complex-sampled RF data than other, simpler classifiers. RSVM was the top performer in dataset #2, potentially because of our data fitting the exponential kernel it provides well. However, RSVM is not as effective as Random Forest (RF) or GNB for dataset #1. This could be because when the target classes overlap (i.e., has noise or similar features) then SVM accuracy tends to decrease. Also, with larger data collections, the model training time is disproportionally higher with SVM methods. It is interesting that RF was not a top performer in dataset #2, considering this was the dataset that was present with more noise, a characteristic RF is able to handle well and RSVM not as well. Random Forest gives better results than most of the classifiers implemented as it has advantages that support our application, as discussed earlier in this paper.

DNN accuracy significantly outperformed the accuracy of the simpler classifiers tested in this research. This makes intuitive sense for several reasons. First, the DNN maintains more information in the data by organizing the data into "images" or "frames". The sampled data was sliced into frames of 160 complex numbers, which holds a lot of information that is lost when the samples are taken individually. For example, the phase information is maintained with the data frame/image, as well as other RF impairment information, including the frequency offset, phase offset and I/Q imbalance. We anticipate that the advantages of the DNN over simpler classifiers would be further magnified if presented with lower SNR data.

For data set #1, DNN obtained 99.8% accuracy. The high SNR and lack of wireless channel is partly to thank for the extremely accurate results.

While the accuracy for data set #2 was limited to 19% when only 1M data samples were used for 10 transmitters, the results were much better with larger samples and fewer transmitters. The trial results shown by the confusion matrix in Figure 15 show that a 90% accuracy was obtained when 10M complex samples are used for each of 4 transmitters. While there are plenty of future research opportunities to explore, including the number of transmitters and optimal data size for a given scenario, this result gives some hope for the ability to distinguish individual transmitters based on RF impairments.

VII. FUTURE RESEARCH OPPORTUNITIES

SNR vs accuracy: It should be noted that the Signal to Noise Ratio (SNR) for Data Set #1 and #2 are quite high. This

may not be the case for all data sets, especially in complex channel environments, where line of site is not always possible. Future research might focus on characterizing accuracy across the spectrum of SNR values.
Cell Phones: While the data in Data Sets #1 and #2 was very instructive in learning which methods work best, it will be very interesting to see how well the DNN works with data collected from cell phones.
Wireless Channels: It will also be interesting to see how accurate a DNN might be with varying channel responses. One area of research might include channel estimation and cancelation from the data in order to obtain the true transmitter RF fingerprint without the channel effects which can smudge the fingerprint.
Number of cell phones: While this research included varied numbers of (simulated) cell phones, it is left for future research to determine how many cell phones can be fingerprinted successfully.

VIII. Bibliography

[1] K. Sankhe, M. Belgiovine, F. Zhou, S. Riyaz, S. Ioannidis, and K. Chowdhury, "ORACLE: Optimized Radio classification through Convolutional neuraL nEtworks," in Proceedings of IEEE INFOCOM 2019, 2019
[2] K. Sankhe et al., "No Radio Left Behind: Radio Fingerprinting Through Deep Learning of Physical-Layer Hardware Impairments," in IEEE Transactions on Cognitive Communications and Networking, vol. 6, no. 1, pp. 165-178, March 2020, doi: 10.1109/TCCN.2019.2949308.
[3] Wu, Qingyang & Feres, Carlos & Kuzmenko, Daniel & Ding, Zhi & Yu, Zhou & Liu, Xin & Liu, Xiaoguang. (2018). Deep Learning Based RF Fingerprinting for Device Identification and Wireless Security. Electronics Letters. 54. 10.1049/el.2018.6404.
[4] Bonifacic, I. (2021, June 17). US lawmakers want to restrict police use of 'Stingray' cell tower simulators. Engadget.
[5] Wikimedia Foundation. (2021, November 24). Stingray Phone Tracker. Wikipedia.
[6] Whittaker, Z. (2020, August 5). A new technique can detect newer 4G 'stingray' cell phone snooping. TechCrunch.
[7] 3GPP.org
[8] Wikimedia Foundation. (2021, November 26). Deep Learning. Wikipedia.
[9] Design a deep neural network with simulated data to detect WLAN router impersonation. MathWorks.
[10] Ognjanovski, G. (2019, January 14). Everything you need to know about neural networks and backpropagation‑machine learning made easy and fun. Towards Data Science.
[11] Brownlee, J. (2020, April 17). How do convolutional layers work in Deep Learning Neural Networks? Machine Learning Mastery.
[12] batchNormalizationLayer. MathWorks.
[13] Specific Layers of Convolutional Neural Network. MathWorks.
[14] Fully-connected layer. CS231N convolutional neural networks for visual recognition. (n.d.).
[15] Google. (2020, March 17). Multi-class neural networks: Softmax. Machine Learning Crash Course.
[16] Russell, S. J., & Norvig, P. (2010). *Artificial Intelligence: A modern approach* (3rd ed.). Pearson.
[17] Zhang, G. (2018, November 11). What is the kernel trick? Why is it important? Medium.

Supplement-1: Matlab Code for generating simulated data with RF Impairments
Supplement-2: Neural Network code
Supplement-3: Classifiers used for data usage method #1