Deep Learning Based Identification of Wireless Protocols in the PHY layer

Alex Berian, Irmak Aykin, Marwan Krunz, and Tamal Bose
Department of Electrical and Computer Engineering
The University of Arizona
Tucson, Arizona, USA
{berian, aykin, krunz, tbose}@email.ariozona.edu

***Abstract*—The need for Artificial Intelligence algorithms for future Cognitive Radio (CR) systems is unavoidable. For a CR to operate as best as possible it must identify who is present in spectrum of interest, and what they are doing (jamming, communicating, rogue transmission, etc.). Using this information, a CR can accordingly decide what to do next. Furthermore, being able to determine which wireless protocols are occupying spectrum is an important ability in heterogeneous wireless networks. In this work, we investigate the robustness of various Neural Network (NN) algorithms for classification of wireless protocols when looking at base-band In-phase/Quadrature (IQ) data without needing to decode. We propose a spectrum sensing algorithm based on NNs or other similarly behaved classification algorithms for identifying wireless technologies occupying spectrum. In previous literature, using base-band IQ data, researchers have shown that NN models can classify different modulation formats with promising accuracy. This work explores the potentials, usage, and limitations of using base-band IQ data for classifying various wireless network protocols that employ the same modulation format.**

***Index Terms*—Cognitive Radio, Neural Networks, Signal Classification, Wireless Protocols**

I. Introduction

Signal classification is an essential addition for more effective cognitive radio (CR) technology as spectrum becomes increasingly scarce. It is common practice in modern CRs to use spectrum sensing as means to avoid occupied spectrum [1]. Modern wireless protocols operate on unlicensed shared spectrum, and being able to identify other existing technologies on the spectrum can improve friendly coexistence. To further improve the performance of a CR, it must be able to identify what is present in spectrum of interest, and act accordingly. When identifying what's in the spectrum, it can be complex and often infeasible to decode what's being observed, so a CR will have to settle for simply identifying the signal. A significant number of wireless protocols employ a form of orthogonal frequency division multiplexing (OFDM) which makes it difficult for existing modulation classification algorithms to differentiate between protocols.

This work explores the use of various neural network (NN) architectures for classifying wireless protocols of similar modulation formats. We analyse two main categories of NN: feed forward NN (FNN), and convolutional NN (CNN). We found that a specific kind of CNN known as a residual NN (ResNet) [2] was overall the most effective in classifying wireless protocols.

There are two main approaches of signal classification: likelihood based, and feature based. Many likelihood based methods make optimal decisions, however they are often too computationally complex for real time implementation.

Protocol	Parameter	Possible Values
LTE	RC	R.0, R.1, R.2, R.3, R.4, R.5, R.6, R.7, R.8, R.9, R.10, R.11, R.12, R.13, R.14, R.25, R.26, R.27, R.28, R.31-3A, R.31-4, R.43, R.44, R.45, R.45-1, R.48, R.50, R.51, R.6-27RB, R.12-9RB, R.11-45RB
	CellRefP	1
	PDSCH # Layers	1
	CFI	1, 2, 3
	Ng	Sixth, Half, One, Two
	PHICHDuration	Normal, Extended
	SSC	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
5G	Cell ID	0, 1, 2, …, 99, 100
	# of Subframes	1
	SSB Block Pattern	Case A, Case B
	SSB Transmitted	Random Binary Vector of Length 4
	SSB Periodicity	5, 10, 16, 40, 80
	Cyclic Prefix	Normal, Extended
	BWP Size	25, 50
	BWP Separation	10, 50
	PDSCH Modulation	QPSK, 16QAM, 64QAM, 256QAM
	PDSCH RV Sequence	Random Ternary Vector of Length 4
	PDSCH Mapping Type	A, B
	DM-RS First Symbol Position	2, 3
	# Front Loaded DM-RS Symbols	1,2
	Other DMRS Symbol Positions	0, 1, 2, 3
	PDSCH Scrambling Identity	0, 1, 2, …, 65535
	PDSCH Scrambling Initialization	0, 1
Wi-Fi	MCS	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
	APEP Length	2^9, 2^10, 2^11
	Guard Interval	Short, Long
	Group ID	0, 63
	Partial AID	0, 1, 2, …, 511
	Channel Bandwidth	CBW20, CBW40, CBW80, CBW160

TABLE I
Protocol Parameter Randomization Options

978-1-7281-4905-9/20/$31.00 ©2020 IEEE

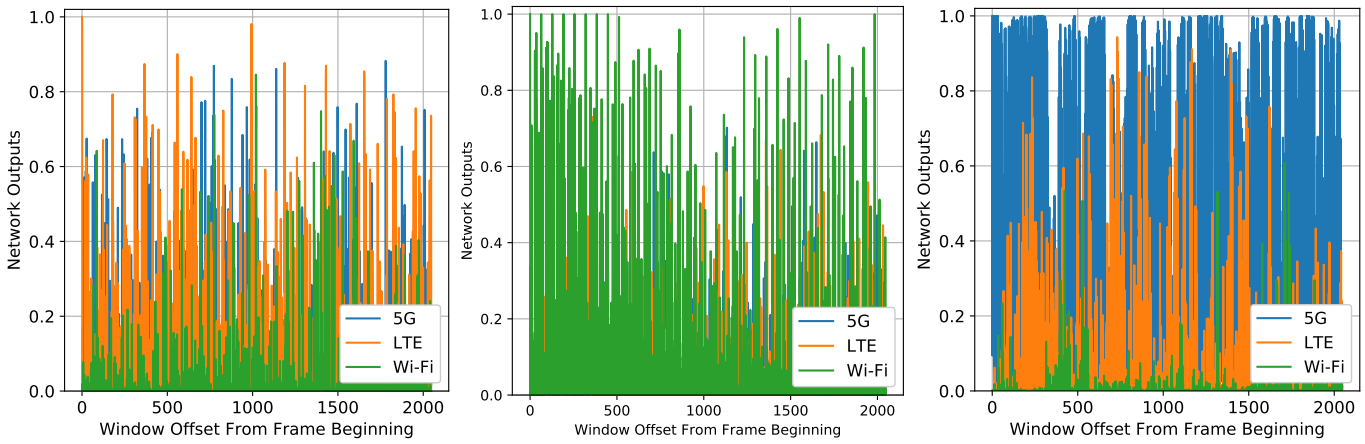

Fig. 1. FNN outputs as the sliding window shifts across a couple of frames. From left to right, LTE, Wi-Fi, ang 5G are analyzed shown respectively.

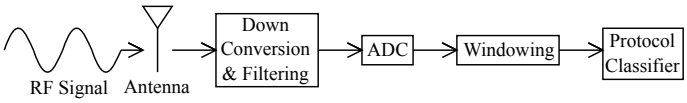

Fig. 2. Receive chain with indication of where to place a wireless protocol classifier.

Feature based classification usually involves extraction of features from a signal (power, bandwidth, carrier frequency, high order statistics, etc.), and use those features as inputs to some kind of classification algorithm which could be machine learning or something else. This is known as expert feature based classification.

We and others speculate that it may be better for a machine learning model to formulate its own features to use in the classification process. There exist some works [3], [4], [5], [6] that explore this possibility, and they achieve competitive results with other traditional feature based approaches. However, we have not seen any prior work analyzing the potential or usage of machine learning in classification and identification of wireless protocols being present or not. In section V, we present a technique for applying NNs in real-time systems to identify when wireless protocols are present in the spectrum.

When multiple protocols are utilized in the same portion of spectrum being able to identify who else is out there can be crucial to friendly co-existence . There have been multiple works that describe the challenges in coexistence of multiple wireless technologies in the same bands [7] [8], and it is difficult for devices designed for one protocol to identify signals from another protocol.

Initially, in an effort to explore the robustness of NNs in this task we tested classification accuracy when passing randomly place windows of 1024 samples within frames or between frames to the NNs, as well as testing the performance when only looking at the beginning of packets. Furthermore, we passed the IQ data from MATLAB waveform generations through various channel models such as Rician, Rayleigh, and COST2100 [9]. NNs have been shown to not perform well under such multi-path channel models [4], and our simulations repeat that result. As a result, we only analyze additive white Gaussian noise (AWGN) channels in the rest of this paper.

This paper presents a threshold based method of identifying when wireless protocols are present in spectrum using deep learning, and benchmarks the method with LTE, Wi-Fi, and 5G protocols as example. Although some protocols have useful features such as beacons or private frequency bands that would make identification easy, we are demonstrating this algorithms ability in conditions when such features are not available as would be the case in adversarial situations.

The proposed method is as follows. A user will gather data and train neural networks on the beginning of frames of the different targeted wireless protocols offline, and the target outputs are one-hot vectors. During training artificial AWGN noise is applied to every training sample with SNR ranging from -40dB to +40dB. When deployed, the NN will observe a window of received baseband IQ samples and provide new outputs with every shift of the window. Based on these outputs the user will select a threshold close to, but less than 1 for each output neuron. When an output neuron exceeds the user selected threshold this is indication that the protocol of the corresponding output neuron is present in the spectrum.

The organization of this paper is as follows. In section II we discuss related signal classification algorithms and works that could benefit by employing the algorithm we present. Section III provides an overview of NN algorithms we analyze. The algorithm we present in this paper is described in V. Performance analysis of the algorithm and other discussion is found in section VI. Section VIII concludes with a brief recap of the algorithm, and other key takeaways from our results.

II. Related Work

There has been extensive literature in recent years regarding classification of modulation formats using machine learning. [4] analysed the performance of various NN architectures and their hyper-parameters in modulation classification with baseband IQ samples. The authors of [3] show that classification accuracy of modulations can be improved with a specialized

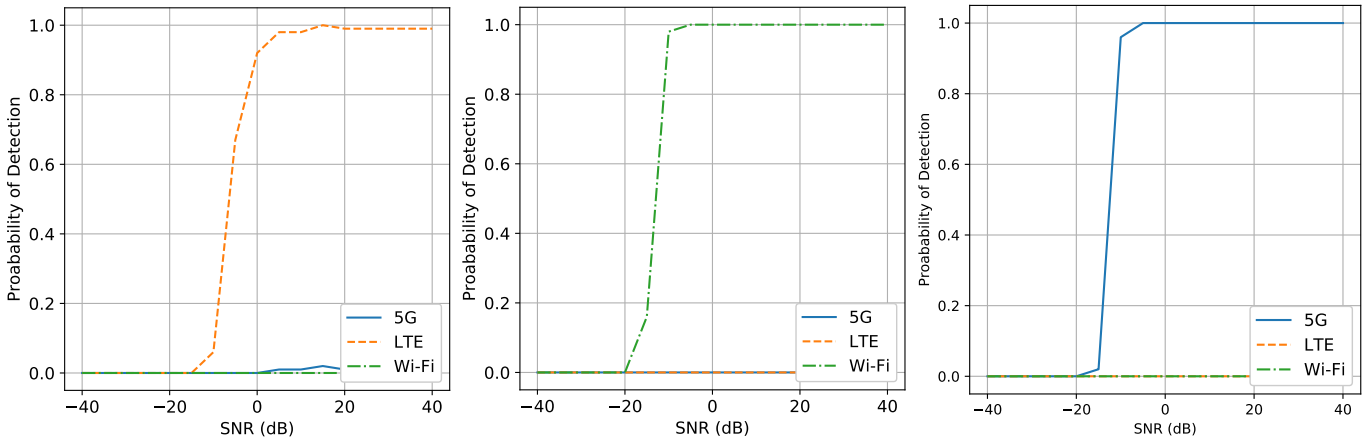

Fig. 3. Probability of detecting protocols when only one protocol is present in the spectrum at varying SNR. From left to right the only protocol present is LTE, Wi-Fi, and 5G

training strategy, where they train a single NN in stages to differentiate modulation formats in a hierarchical manner. [6] also explored a hierarchical manner of modulation classification, but these authors used multiple NNs each specializing in narrowing down the modulation class which ultimately performed better than [3].

Few existing works have explored protocol classifications as well. [10] explores classifying between slotted ALOHA and Time Division Multiple Access (TDMA) by using power mean and power variance over time as features to be classified with a Support Vector Machine (SVM). Some works such as [5] have explored using IQ data with deep learning algorithms. to classify different applications being used under the same communication protocol.

III. Neural Network Architectures

In this section, we will briefly describe the various NN architectures explored in this work. This section only gives a very brief discussion of what NN techniques are explored in this paper, and we encourage the reader to look in reference [11] if they desire more explanation of these techniques.

A NN is nothing more than a large deterministic functions with a set of changeable parameters denoted as θ, and training a NN is the process of changing θ to increase classification accuracy. NNs act upon an input x to produce a prediction $h(\theta, x) = \hat{y}$ where h is the structure of the NN. A loss function $\ell(\hat{y}, y)$ is a measure of how different the prediction $\hat{y}$ is from the correct output y, and NNs seek to minimize this loss during training.

Most NNs are trained using the back-propagation algorithm. Back-propagation is an efficient algorithms to calculate the gradient of multiple variables in a large complex equation. For NNs, it is used to calculate the gradient of θ with respect to the loss $\ell(\hat{y}, y)$. A common optimization algorithm for training neural networks in stochastic gradient decent which modifies theta in steps as

$$\theta_n = \theta_{n-1} - \eta \nabla_\theta \ell(\hat{y}, y) \tag{1}$$

where η is known as the learning rate, θ_n is the new set of changeable parameters in the network, and θ_{n-1} are the old ones.

Depending on the problem, a user must decide what form the output of their NN model will take on. For a problem where a user wants to classify between two classes, it is common to use a single neuron as the output, with a sigmoid activation function. The sigmoid neuron outputs $\hat{y} \in [0, 1]$, and the user can select a threshold for class selection between the two classes. With sigmoid output neurons, binary cross entropy is typically used as the loss.

$$\ell(y, \hat{y}) = -log_e(1 - y - \hat{y}(-1)^{1+y}) \tag{2}$$

In the case of more than two classes in a classification problem, the softmax layer is often used. With softmax layers, each neuron is assigned a corresponding class, each neuron outputs a value between 0 and 1, their sum is 1, and each output can be interpreted as the probability the input belongs to the corresponding class. Categorical cross entropy is the most common loss with softmax output layers.

For our problem, we want to identify when particular wireless protocols are present or not. This calls for a bank of sigmoids for the output layer. Each protocol gets a sigmoid output neuron, and if any neuron exceeds a threshold the protocol will be deemed detected. The loss typically used with sigmoid bank output layers is binary cross entropy.

A deep FNN consists of many dense layers. A dense layer is multiplying the input to the layer by a matrix of weights, then adding that result to a vector of biases, then passing the results into an element-wise nonlinear activation function (such as ReLU, sigmoid, tanh, etc.). That result is then passed into other layers depending on the network architecture.

A CNN consists of convolutional layers. A convolutional layer is a simple convolution operation where we convolve the input to the layer with one or more filters of trainable weights, and the weights are modified according to their respective gradients with respect to the loss during training.

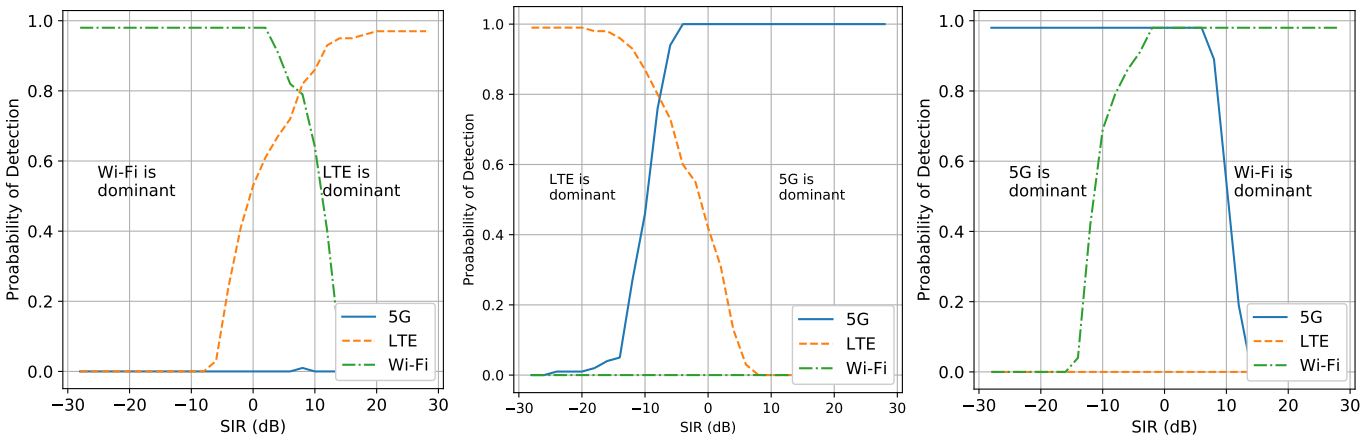

Fig. 4. Probability of detection/misdetection when two protocols are interfering at various SIR.

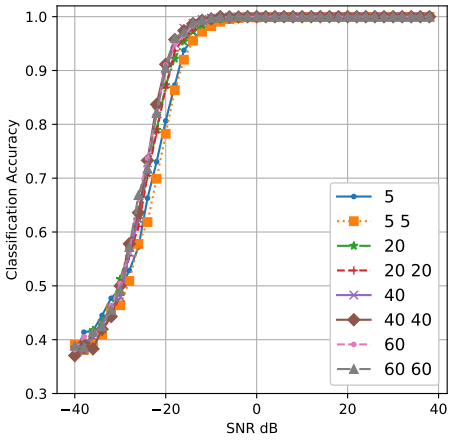

Fig. 5. Variable sizes of FNN for 1024 IQ samples under AWGN

IV. Data Collection

In this paper, to demonstrate the ability of NNs to identify wireless protocols we identify LTE, WiFi, and 5G frames. We collect baseband IQ samples of the waveforms for these frames using MATLAB toolboxes. MATLAB has toolboxes that contain waveform generation for all these protocols. Within these toolboxes, we randomize various parameters for the signals. The idea behind doing this is that if we get as much variation within the different classes as possible, we can prove just how effective NNs are in finding differences despite the variability. Additionally, in practice a user will not have control over signals that belong to different protocols, so being able to identify a large variety within those protocols is important. Table I displays all of the parameters that were changed and/or randomized within the protocols. All of the protocols were constrained to work for single-input-single-output (SISO) only. In this paper, all training and test data are created according to Table I, and they are split by 80%/20% for training/testing data respectively.

V. System Design

FNNs and CNNs are constricted to a finite view of signals. As a result, it is best if they focus on the beginning of frames as they are the most unique and consistent part of the waveform for most protocols. Fig 2 shows where a user would implement the proposed algorithm. After a signal is sampled a window (we used rectangular) will shift with every received sample.

The user will design a NN that uses a sigmoid bank output layer with each neuron pertaining to a wireless protocol of interest for detection. Users should collect IQ data at their target sample rates offline, and window the beginning of frames. For training the NN, each window is treated as an input vector, and one-hot encoded labels are used as the desired output. During training we propose that the user should add artificial noise to each training sample with a random SNR of -40dB to +40dB at each training step. If the same training sample is used again, a new random SNR must be selected and newly generated accompanying noise. The input to the NN is always normalized to have a power of 0dB.

We propose that to indicate detection with a sigmoid bank NN for this application, the threshold for each of the output neuron should be close to 1. When an output neuron exceeds it's threshold, the corresponding protocol is deemed detected. As the threshold t_i for the i^{th} output neuron is increased, the probability of detecting the i^{th} class $P(i)$ approaches 0. In other words,

$$lim_{t_i \to 1}\Big(P(i)\Big) = 0 \tag{3}$$

and conversely,

$$lim_{t_i \to 0}\Big(P(i)\Big) = 1 \tag{4}$$

However, decreasing t_i also increases probability of false alarm, and increasing t_i increases the probability of misdetection. The user must tune t_i for each output neuron depending on the false alarm and misdetection probability constraints of their system.

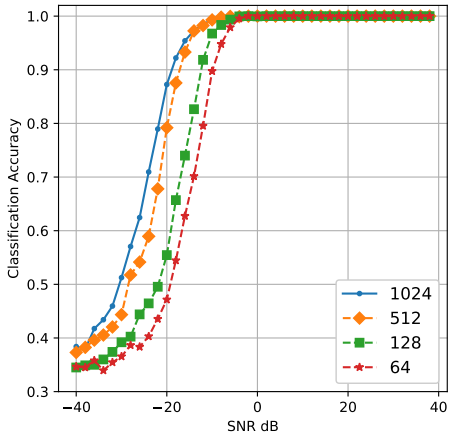

Fig. 6. Comparing classification accuracy under AWGN for different window sizes using a FNN of depth 1 and width 20. The label "1024" indicates that the input to the network is 1024 IQ samples concatenated and windowed.

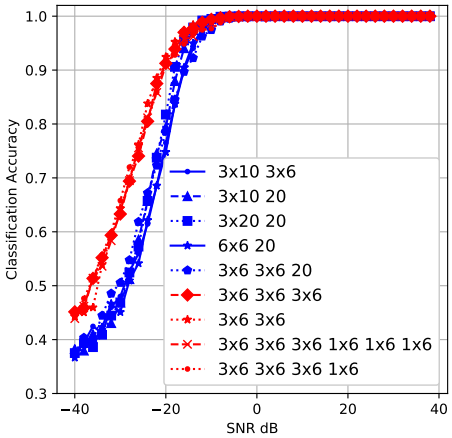

Fig. 7. Comparing varying CNN architectures with and without pooling on 1024 IQ samples under AWGN. Blue curves denote pooling is present in the NN. "3x10" denotes a convolutional layer with 3 filters of size 10, and "20" denotes a dense layer of 20 neurons. Multiple of such phrases on a label indicate that the layers are connected for that network.

There is significant drop in performance if the windows the NN is trained on are not at the beginning of frames. When training and testing CNNs and FNNs on random locations of frame, we only saw a 40% classification accuracy under the best conditions. With 3 classes, 33% accuracy is as good as random guessing. In Fig 1, we see the outputs of a FNN when a window slides across a couple of frames. Incorrect output neurons tend to get very close to 1, however they rarely surpass the maximum output we observe for the correct class.

VI. Performance Analysis

In this section we analyze the testing accuracy of the NNs, and probability of detection with the proposed approach. Training time for even our most complex NNs never exceeded 1 hour on a computer without a GPU, and a dual core Intel i5 processor. All NN implementation, training, and testing was programmed in Python using the Keras library within Tensorflow.

In Fig 5 we simulate probability of correct classification on the beginning of frames as SNR increases in an AWGN channel model. We see that there isn't very much gain in performance when increasing the depth or the width of the FNN. It is worth noting that classification accuracy with a FNN with one hidden layer and a width of 5 neurons performed slightly worse than a width of 20 neurons. The accuracy between width 40 and 60 are the same. We also see that increasing the depth of the network did not have any effects on performance improvement.

NNs can be extremely computationally complex, and reducing the size of a NN reduces the computational complexity. In Fig 6 we show probability of correct classification vs SNR on the beginning of frames, as the input window size decreases. A different FNN of depth 1 and width 20 was trained and tested on their respective window sizes to generate the results in Fig 6. As one would expect there is a decrease in performance as the window size decreases, because there is less information for the NN to understand. The convergence to 100% classification accuracy with respect to SNR only changes by a few dB.

It is common in CNNs to use max pooling. A max pooling layer takes a sliding window across it's input, and throws away all inputs from each window that are not the maximum in their respective windows. For the case of IQ samples, we choose the max based on magnitudes of the samples. In Fig 7 we show the effects of using max pooling windows of size 2 and 3 after convolutional layers when AWGN is the only channel impairment. As an example to explain the notation in Fig 7, *3x10 conv* means that 3 parallel convolutional filters of 10 taps are used in that layer. Max pooling clearly does not perform as well at low SNR, but CNNs with max pooling do converge to 100% classification at around the same SNR regions as CNNs without max pooling. Though performance is not quite as good at low SNR with pooling, it is important to note that max pooling significantly reduces computational complexity in a NN. We recommend users employ max pooling, because they still converge to 100% classification accuracy at the same SNR as those without, and they significantly reduce computational complexity.

ResNets were first presented in [2] for image classification, and were shown to be quite effective for modulation classification in [4]. They have been one of the most popular NN architectures, because of their great performance across many tasks. The intuition behind why ResNets are so effective is because they keep residual information of early layers all the way throughout a deep NN through addition. This allows the back-propagation algorithm to easily calculate the gradients for early layers, while still getting the benefits of having a deep NN. We show the performance of a few ResNet architecture in Fig 8. Since there is virtually no gain in increasing the depth of the network, we do not recommend doing so.

Fig 3 shows the probability of a CR employing the presented algorithm detecting various protocols, when only one protocol is present. Each data point is generated by having a width 20,

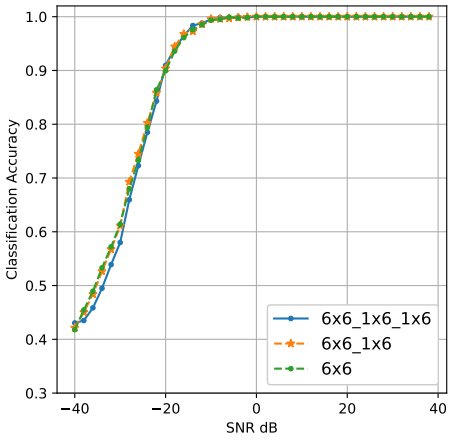

Fig. 8. Comparing classification accuracy under AWGN for different ResNet architectures.

depth 1 FNN with 512 IQ inputs look at a sliding window across a couple of frames, and if the output from the FNN at any window exceeds a threshold of 0.9999 it is counted as detected. We generate each data point by doing this 900 times, each time with different signals. The middle plot from Fig 3 shows probability of detection when Wi-Fi is the only protocol present in the spectrum. This plot indicates that there is very low probability past 0dB SNR that LTE or 5G are incorrectly detected, and Wi-Fi is always properly detected. Unfortunately the left plot in Fig 3 indicates that 5G is rarely, but sometimes detected when LTE is the only signal present. This is likely because Wi-Fi has a very unique and consistent preamble, where as LTE and 5G frame beginnings are similar under certain protocol parameters.

Assuming one protocol will be the only thing present in the spectrum is impractical. LTE, Wi-Fi, and soon 5G will all be occupying the same spectrum in certain bands. To explore the effects of interference, we have superposed signals of two different protocols over each other at varying Signal to Interference Ratio (SIR). In Fig 4 we present probability of detection when there is interference between two different protocols. To generate a data point in Fig 4, we look at 900 instances of two protocols interfering, and observe a couple of frames. Each simulated interference has the first frames from each class beginning at the same time which is rare in practice.

VII. Comparative Analysis

In this section we present simulation analysis of other approaches to detection of wireless protocols.

Another approach to signal detection with NNs would be to introduce an output neuron that indicates no protocol of interest is present in the spectrum: a noise output neuron. With this approach a protocol is deemed detected whenever its protocol output neuron surpasses the noise output neuron, and softmax is used as the output layer. It would not make sense to include an output neuron for noise under the proposed training regime, because we train with SNRs as low as -40dB. To analyze performance of this approach, we alter the lower bound on AWGN to 0dB, as that is well past the point of 100% classification as shown in section VI. This approach falls short when it comes to false alarm. When a NN with the noise output neuron sees a couple of frames from any class at any SNR in our simulations, every class is deemed detected. This is a false alarm rate of 100%.

In purely classification problems it would make sense to use the softmax output layer, so another approach to protocol identification is to just use a softmax output. There are instances when no protocols are present, so thresholding is still needed with a softmax output layer as opposed to a sigmoid bank. Fig 9 shows the misdetection and false alarm probabilities when a softmax output layer is emplyed with our proposed approach as opposed to a sigmoid bank. The threshold that performed best for a softmax output in our dataset was 0.99. Softmax appears to struggle with false alarms between 5G and LTE much more than a sigmoid bank shown in figure 3.

VIII. Conclusion

This paper proposes and analyzes a deep learning based algorithm for identifying if certain protocols are present in spectrum. Users train a softmax NN to classify the beginning of frames belonging to certain protocols offline. The beginning of frame windows are given artificial AWGN noise ranging from -40dB to +40dB SNR, and the input is always normalized to 0dB power. When applied in practice, a sliding window of baseband IQ samples from an ADC is passed into the NN. When an output neuron passes a user defined threshold, this is indication the the protocol associated with that sigmoid neuron is present in the spectrum. We showed that depth does not provide any performance improvement. Using max pooling in a CNN is a good idea, because it reduces computational complexity, and converges to 100% classification accuracy at around the same SNR as NNs that don't use max pooling. We showed that reducing window size does not strongly affect the SNR ranges of good classification accuracy. When multiple protocols are present in the spectrum the algorithms is still able to identify which protocols are present. The proposed threshold based approach performs better significantly better than to have a designated noise output neuron in a softmax output layer.

Acknowledgment

The authors of this paper would like to thank Broadband Wireless Access Center (BWAC) for funding this research. BWAC is an industry sponsored subset of the National Science Foundation (NSF).

References

[1] I. F. Akyildiz, W.-Y. Lee, M. C. Vuran, and S. Mohanty, "Next generation/dynamic spectrum access/cognitive radio wireless networks: A survey," *Computer Networks*, vol. 50, no. 13, pp. 2127 – 2159, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128606001009

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 770–778.

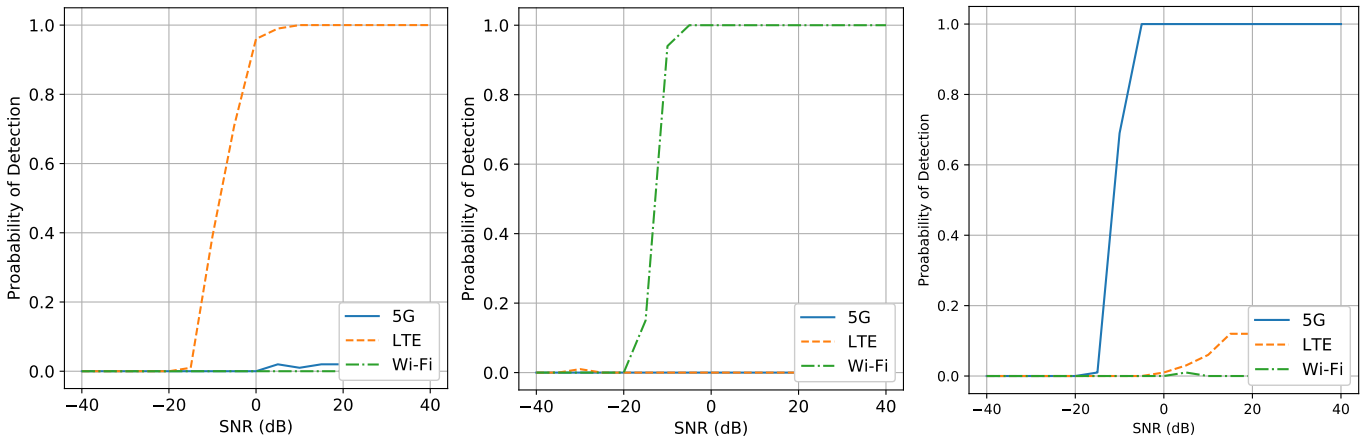

Fig. 9. Probability of detection/misdetection when only one signal is present, and a FNN uses a softmax output layer. From left to right the only protocol present is LTE, Wi-Fi, and 5G

[3] G. Vanhoy, N. Thurston, A. Burger, J. Breckenridge, and T. Bose, "Hierarchical modulation classification using deep learning," in *MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM)*, Oct 2018, pp. 20–25.

[4] N. E. West and T. O'Shea, "Deep architectures for modulation recognition," in *2017 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, March 2017, pp. 1–6.

[5] T. J. O'Shea, S. Hitefield, and J. Corgan, "End-to-end radio traffic sequence recognition with recurrent neural networks," in *2016 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, Dec 2016, pp. 277–281.

[6] K. Karra, S. Kuzdeba, and J. Petersen, "Modulation recognition using hierarchical deep neural networks," in *2017 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, March 2017, pp. 1–3.

[7] C. Cano, D. Lopez-Perez, H. Claussen, and D. J. Leith, "Using lte in unlicensed bands: Potential benefits and coexistence issues," *IEEE Communications Magazine*, vol. 54, no. 12, pp. 116–123, December 2016.

[8] M. Hirzallah, W. Afifi, and M. Krunz, "Full-duplex-based rate/mode adaptation strategies for wi-fi/lte-u coexistence: A pomdp approach," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 1, pp. 20–29, Jan 2017.

[9] L. Liu, C. Oestges, J. Poutanen, K. Haneda, P. Vainikainen, F. Quitin, F. Tufvesson, and P. D. Doncker, "The cost 2100 mimo channel model," *IEEE Wireless Communications*, vol. 19, no. 6, pp. 92–99, December 2012.

[10] Zhuo Yang, Yu-Dong Yao, Sheng Chen, Haibo He, and Di Zheng, "Mac protocol classification in a cognitive radio network," in *The 19th Annual Wireless and Optical Communications Conference (WOCC 2010)*, May 2010, pp. 1–5.

[11] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.