An Optimized Faster Region-Based CNN for 1D Spectrum Sensing and Signal Identification in Cluttered RF Environments

Todd Morehouse, Charles Montes, and Ruolin Zhou
Department of Electrical and Computer Engineering
University of Massachusetts, Dartmouth, MA

2023 IEEE Future Networks World Forum (FNWF) | 979-8-3503-2458-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/FNWF58287.2023.10520407

***Abstract*—In this paper, a faster region-based convolutional neural network (FRCNN) has been optimized to allow processing 1D signals and improve the inference time of the architecture for spectrum sensing and signal identification in cluttered RF environments. The number of wireless devices in operation continues to grow, contributing to an ever increasing spectrum congestion. These devices may be competing for scarce resources, or may use new capabilities to interfere in spectrum bands they do not belong. Spectrum sensing is a quintessential ability for cognitive radios, allowing them to detect the presence of transmitters. This may be used to optimize spectrum usage, increasing throughput and performance, or for security purposes, such as monitoring for abnormal activity at an airport. In each of these scenarios, the ability to accurately and adeptly sense the spectrum is required. Object detection has been shown to be excellent at locating signals in a congested environment. Our research optimizes FRCNN object detection for processing 1D signals, dramatically reducing computational complexity. This transformed model is capable of handling 1D FFTs directly, instead of image inputs. We show that our approach achieves higher performance than other state of the art models, through both synthesized and over-the-air tests. Additionally, we show the ability of the detector to allow processing and identifying of multiple signals, by applying automatic modulation classification to each detected signal.**

I. Introduction

The number of wireless devices in operation continues to grow, contributing to ever increasing spectrum congestion. These devices may be competing for scarce spectrum resources, where optimal allocations are required to maintain communications. In other scenarios, government, military, and commercial sectors may share the same spectrum bands, and may wish to secure the spectrum. In each of these scenarios, the ability to accurately and adeptly sense the spectrum is required, where each signal present must be identified and localized. This task becomes increasingly difficult, due to the diverse nature of wireless channels and wide applications of software defined radio, as well as the dynamic nature of the environment. It is important for a spectrum sensing algorithm to be deployable in any scenario, and still accurately sense the spectrum and reliably identify signals.

In this paper, we optimize faster region based convolutional neural network (FRCNN) for spectrum sensing, where we identify and localize wireless signals in frequency domain. Additionally, we consider a dynamic environment, where no centralized channel allocation is present. This allows wireless devices to transmit at any frequency and bandwidth, but requires a more sophisticated detection method. An example of this is shown in Fig. 1, which was synthesized using the method outlined in Sec. II. Since signals may be present anywhere within this channel, and with any bandwidth, an object detection method is appealing here. However, the standard implementations for object detection are intended for processing images, and are not fit for processing received baseband samples directly. This is evident in current literature, which either saves spectral plots as images [1], [2], or creates spectrograms before performing detection [3]–[6]. Thus, in this paper we optimize FRCNN to achieve spectrum sensing from a 1D FFT input. We choose to use FRCNN, as the two-stage approach can achieve higher accuracy [1], especially at lower signal-to-noise ratio (SNR) [5]. We show how each detected signal can be separated and processed individually, by applying standard automatic modulation classification (AMC) after detection. This can be extended to further applications in classification or signal processing, such as performing security related check. To prove the generalization of our model, we test our system over-the-air (OTA), using two USRP N2901 software defined radios (SDRs). These devices allow controlling the physical layer of wireless communications using high level software, and provide an easy platform to integrate machine learning with communications.

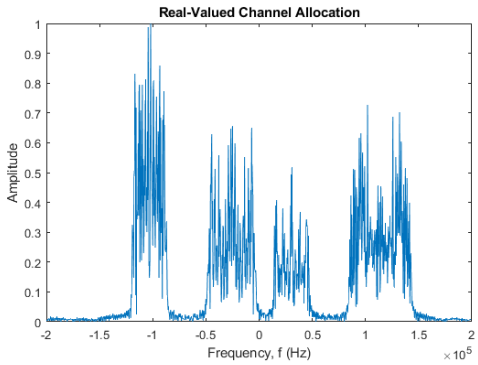

Fig. 1: Synthesized Bursty Transmission Signal

Deep learning has seen extensive application to spectrum

sensing, such as CNNs [7][8]. These methods use the feature extraction abilities of deep learning to identify if a primary user is active on a channel, and outperform traditional detectors. More recently, object detection and segmentation methods have been used to localize signals in multi-user environment, which adds the ability to detect and localize multiple transmissions in a band. In O'Shea et al [6], the authors showed that you-only-look-once (YOLO) can be used to find signals present in a spectrogram. In Prasad et al [3], [4] the authors used FRCNN to jointly detect signals within the spectrogram image, and classify detections as Bluetooth, WiFi, or microwave oven interference. They were able to achieve a mean average precision (mAP) of 0.713 under the SNR interval between 15dB and 50dB. They tested their system OTA, but were only able to achieve a mAP of 0.125. In Vagollari et al [5], the authors used YOLO to localize signals within a spectrogram, and classify them by their modulation type. They were able to achieve an mAP of 87% and a mean intersection over union (mIoU) of 90%. Other authors investigated object detection on frequency contents only. In Ghanney and Ajib [1], the authors locate radio frequency interference within a fast Fourier transform (FFT) generated plot. In order to process the plot with YOLO, they save images of the plots. Their results achieve an mAP of 0.81 at an IoU of 0.5. In Kayraklik et al [2], the authors generate power spectral density (PSD) plots from baseband signals collected OTA. They save their PSD plots as images so that they can be processed by You Only Learn One Representation (YOLOR) and Detectron2. These implementations are limited to images by their object detection baselines. **We posit that the state of the art usage of 2D images for signal processing is inefficient and less effective.** In signal processing, especially wireless signals, data is often single dimension. While a workaround enables using 2D object detection by converting 1D signals to images, it adds unnecessary steps and computational complexity. 2D Object detection requires searching a 2D space instead of a 1D space, resulting in exponentially higher computation. Our work optimizes FRCNN to process 1D signals directly, greatly reducing the inference time and improving the performance in spectrum sensing.

Our contributions are summarized as follows

- Optimizing FRCNN for 1D signal processing and spectrum sensing.
- Showing the efficacy of the 1D transformation through improved performance.
- Demonstrating real-world generalization using SDR.

Additionally, we test our system's ability to isolate signals for further processing using automatic modulation classification (AMC). We demonstrate that multiple signals can be classified using this approach.

Our system was able to achieve higher performance than object detectors in current literature. Vagollari et al [5] had the highest performance. **Our mAP was found to be 0.99 compared to the authors' 0.87.** Our mIoU was found to be comparable to theirs, with ours measuring 0.90 without AMC. Our work generalized well to real world environments, which we show with our OTA metrics. Other authors tested their model OTA but were not able to demonstrate that their model generalized well [4]. We also show a significant increase in inference and training speed, by approximately 5x. Additionally, we show the networks ability to perform further signal processing on individual signals by applying automatic modulation classification (AMC) on detected signals. The AMC design is from our previous research [9][10]. Since the AMC output relies on the results of FRCNN, it can reduce the mAP. In this case, the signals are still detected, only their modulation type is mislabelled.

This paper is organized into the following sections: First we discuss the RF environment in Sec. II, then the FRCNN optimizations in Sec. III, finally we show our test results, both synthesized and OTA, in Sec. V, then conclude in Sec. VI.

II. RF Environments

First, we define the wireless environment that we are operating in. We consider a spectrum of arbitrary size, where multiple transmitters are present. We observe the entire spectrum from a single receiver. The goal of the sensing algorithm is to be deployable in diverse scenarios, and still being capable of sensing all signals in a spectrum. To this end, the environment we considered had no discrete channel allocation, which would include pre-defined parts of the wireless band, with an associated primary user. We synthesized this environment using MATLAB's communications toolbox. In previous research, we found that this method generalized well in OTA tests [9]–[11].

The process for synthesizing samples could be seen in Fig. 2. MATLAB provides the necessary functions for generating data, modulating, filtering, and mixing signals. A random number of signals were generated independently, up to five signals, each with random gains to represent different power and distance transmitters. Then, each signal is summed in baseband. The combined signal's power was normalized to 1 by the equation shown in Eq. 1, where x is the signal sample vector and N is the number of samples. The normalized signal was passed through the channel effects shown in Fig. 3, including Rician fading, and additive white Gaussian noise (AWGN). Finally, the FFT of the signal was taken, along with the frequency location of each signal present, to be used for training FRCNN. An example of the resultant synthesized channel could be seen in Fig. 4.

$$y = \frac{x}{\sqrt{\frac{1}{N}\sum |x^2|}} \tag{1}$$

III. Optimizing FRCNN

Faster region-based CNN [12] is an augmentation to traditional CNNs, to allow region detection within an image. FRCNN consists of three main components: a convolutional feature extractor, region proposal network (RPN), and region of interest classifier shown in order in Fig 5.

In the 1D architecture, the samples of a signal are input into the network. These samples are arranged in order so that

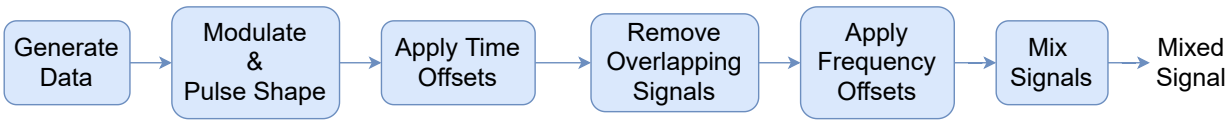

Fig. 2: Process for synthesizing signals in MATLAB

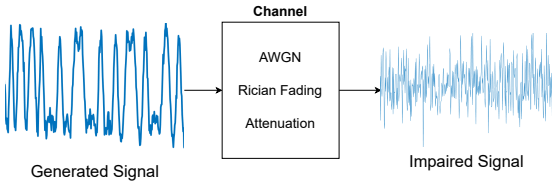

Fig. 3: Channel Impairment Process for Synthesized Signals

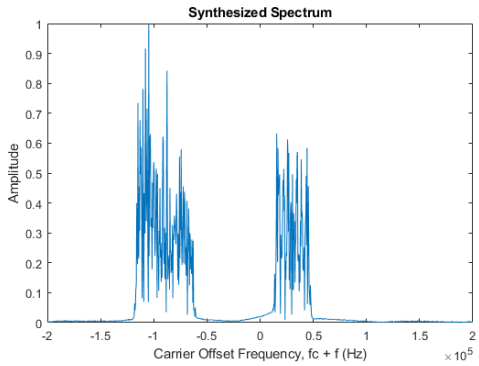

Fig. 4: Synthesized Channel

spatial information can be exploited, such as time or frequency features. The RPN outputs a fixed number of predictions, some are foreground (a signal) and others are background. The classifier refines these predictions to determine which predictions contain a signal and better isolate detected signals. This is done by assigning a probability to each prediction, and thresholding by probability. This can result in duplicated predictions over the same signal; non-max suppression (NMS) is applied to remove duplicated detections that may overlap with each other.

A. Anchor Boxes and Region Proposals

The first stage of FRCNN involves proposing anchor box regions. Anchor boxes are an efficient way to search for objects within a dataspace, by searching specific locations for objects, using predefined aspect ratios. The goal of the RPN is to identify which anchor boxes overlap a ground truth, given an input. To achieve this, a convolutional feature extractor is used, which downscales the input into a feature matrix, proportional to the input size. Each element of the feature matrix maps to an anchor box in the input data. The RPN then consists of a simple convolutional network, which predicts whether each feature contains an object. Additionally, regression is performed to find the anchor box from the feature matrix directly. However, the anchor boxes may not fit the object well, and may be background detections.

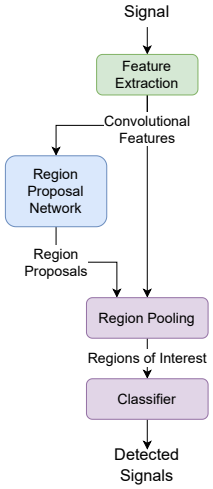

Fig. 5: Flowchart of 2D FRCNN

B. Region Pooling and Classification

The second stage of FRCNN filters and refines region proposals. The RPN will propose a fixed number of regions, regardless of the input. The second stage must determine which proposals are valid, and refine those proposals. First, region pooling will begin by windowing the feature matrix around the predicted anchor box, called the region of interest. The features are then classified as either “foreground” and “background”, or a user-defined structure. Simultaneously, regression is applied to the regions of interest, to fine-tune the predicted bounding box. The resultant outputs are the probability that a detection is foreground or background, and the predicted bounding box for the detection.

C. Optimizing FRCNN for 1D

The theory and construction of FRCNN is fundamentally 2D, where anchor boxes, region proposals, and region of interest (ROI) pooling are 2D operations. Optimization for a 1D process requires redesigning these components, to accept a 1D input. First, bounding boxes, anchor boxes, and detections are redefined as intervals. For spectrum sensing, the interval represents the frequency bounds of a signal. The width of the interval represents the signal’s bandwidth. The RPN must be transformed to accept 1D signals, and propose intervals. Feature extraction consisted of blocks that downscale the input by a factor of 2, each block with three convolutional layers, followed by a max pooling layer. The convolutional layers use a filter size of [3,1], performing feature extraction along the signal axis. The max pooling layer uses a stride of [2,1] to downscale the input by a factor of 2. The convolutional layers

pad the input to ensure that truncation does not occur from the pooling layer stride. These blocks are stacked together to achieve a downscaling factor of 16. The RPN consists of convolution layers starting with [3,1] filter size then split between two [1,1] convolution filter sizes. Region pooling must be transformed to process 1D feature matrices of varying sizes. The output of the RPN is used to window the feature matrix around the proposed region. These features are then resized using linear interpolation. Finally, these ROIs can be used to produce the detections using classification and regression. The ROIs are flattened, then preprocessed using two fully connected layers (FCLs). An additional FCL is used to classify the ROIs as either "foreground" or "background". Finally, a fourth FCL is used to perform regression on the ROIs, to find their bounding intervals.

D. Two-Step Training FRCNN

We train FRCNN using a two-step process. Training is alternated between the RPN and the classifier. The output of the RPN stage after training is used as the training input for the classifier stage. Training is done using a single input signal at a time. Each input may have multiple signals present, each as a training sample. Additionally, "negative" samples are generated to represent predictions over background. The RPN requires separate training data to complete the two-step process. For each feature in the feature matrix, the corresponding interval on the input is checked for an object. Predefined anchor sizes are used to represent different signal bandwidths. If a ground-truth object is found to have an IoU greater than 0.7 with any anchor, it is considered a positive overlap. If the IoU is less than 0.3, then it is considered a negative overlap, and treated as background. Any IoU in between is considered ambiguous, and not used for training. The equation for IoU is shown in Eq. 2. The results are summarized in a binary indicator matrix, used to train the RPN classifier. The anchor boxes are summarized as RPN regression targets. The second-stage classifier is trained using the results from the RPN. Since the RPN may propose a large amount of regions, a subset of positive and negative samples is chosen randomly. This also ensures that these samples are balanced. For each selected sample, the regression targets are the ground truth bounding boxes, and the classification is "foreground" or "background".

$$\text{IOU(A,B)} = \frac{|A \cap B|}{|A \cup B|} \tag{2}$$

IV. Use Case using AMC

Spectrum sensing is often a first step process, prior to further signal processing or classification. We demonstrate a use case of the optimized FRCNN architecture to classify signals using AMC. In previous research, we implemented AMC using CNNs [9], [10]. We found that a simple linear network performs well, achieving accuracies of approximately 95%. The network consists of four feature extraction blocks. Each block is formed by a convolutional layer with a [1,8] filter, ReLU activation, batch normalization, and max pooling which divides the input dimension by a factor of 2. The features are flattened, and logits are computed using an FCL. Finally, a softmax layer finds class probabilities from the logits. In our original work, we classified 11 different modulation types. To simplify this use case, we classify only three, BPSK, 16QAM, and PAM4. We introduce an additional class for "No Signal" to represent false positive predictions from FRCNN. This can allow better filtering when false positives are high, typically at low SNRs. Each AMC sample was generated by separating baseband signals in time domain, using the detection results from FRCNN. First, for the signal of interest, the signal was shifted by the detected center frequency. Next, a low pass filter was applied, with the bandwidth tuned to the detected bandwidth. The output of the filter was then suited to be input into the AMC classifier. This process was repeated for every detection.

To train the AMC network, synthesized signals were used. The FRCNN spectrum sensing algorithm will not perfectly detect the signal in frequency domain, resulting in varying bandwidths and center frequency offsets. The filter will impart effects, such as phase shift. Therefor, AMC training samples were given random center frequency offsets, bandwidths, and phase shifts.

V. Training Results and Performance

The RF data was synthesized using the process described in Sec. II, and the parameters shown in Table I. We trained our networks using the synthesized data. The training parameters could be seen in Table II.

Parameter	**Value**	**Unit**
Center Frequency	N/A	-
Baseband Sample Rate	400	kHz
Minimum Signal Bandwidth	20	kHz
Maximum Signal Bandwidth	60	kHz

TABLE I: RF Simulation Parameters

Parameter	Value
Epochs	40
Iterations	1000
Training Samples	1000
Optimizer	Adam
Learning Rate	10^{-5}
NMS Overlap	0.7
RPN Overlap Range	[0.3, 0.7]
Anchor Boxes	[20 kHz, 40 kHz, 60kHz, 80kHz, 100kHz]

TABLE II: Training Parameters

$$\text{Precision} = P[i] = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \tag{3}$$

$$\text{Recall} = R[i] = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \tag{4}$$

$$\text{mAP} = \frac{1}{N}\frac{1}{K}\sum_{i}^{N}\sum_{k}^{K} P_i[k] \cdot (R_i[k] - R_i[k-1]) \tag{5}$$

The network was evaluated using the mAP, shown in Eq. 5. Precision is the measure of a detectors accuracy, shown in Eq. 3, while the recall is the ability of a network to detect all objects present, shown in Eq. 4. The mAP finds the area under the precision-recall curve. Detections are filtered as positive or negative by their prediction probability. Detections above a probability threshold are considered positive. The mAP measures a detector's performance over different probability thresholds. We utilized the average precision function from the Scikit Metrics library in python. Additionally, we measure the mIoU for the detector, to demonstrate the localization accuracy.

We measured the mAP for the 1D FRCNN implementation, using the mixed SNR dataset, and found it to be 0.99, under the SNR interval between 10dB and 40dB. This performed significantly better than current literature, where the mAP in Prasad et al [4], where they achieved a mAP of 0.713 under the SNR interval between 15dB and 50dB, and a mAP of 0.63 under the interval of 5dB and 50dB. Additionally, we measured the mAP vs SNR, shown in Fig. 6. Our detector reliably produces a mAP above 0.9 at 10dB and higher, but quickly drops below this, until dropping to 0.82 at -5dB.

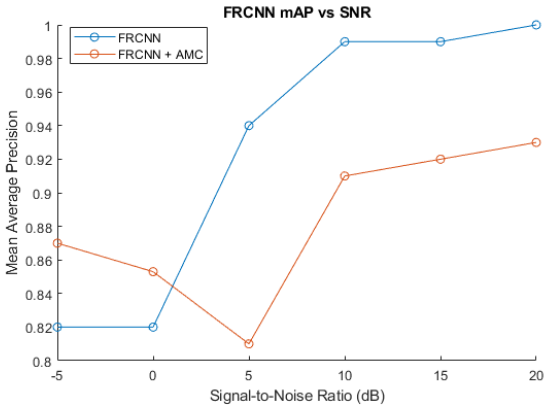

Fig. 6: mAP versus SNR for FRCNN.

Next, the AMC network was added to the architecture. Received FFTs were first passed through FRCNN to find the estimated signal locations. Each signal was isolated and filtered out, and then passed to the AMC architecture for classification. The test accuracy of this network was found to be 95%. The mAP versus SNR was measured, to evaluate the effectiveness of AMC with the architecture, which can also be seen in Fig. 6.

In addition to the mAP, we measured the mean-IoU for the FRCNN algorithm. While mAP is a popular metric in data science, it has some major shortcomings for this application. In spectrum sensing, accurate detection of the signal bounds is just as important as detection of the signal itself. In order to properly window the signal to analyze it further, its location in frequency domain must be accurately determined. This is especially true for the AMC case, where improper filtering can cause misprediction. We chose to measure the mean-IoU as the localization performance. These results could be seen in Fig. 7. It was found that the mean-IOU for FRCNN without AMC was 0.727, and with AMC was 0.482.

It can be observed that the mAP dropped considerably when AMC is introduced. However, for low SNR, the addition of AMC improves the mAP compared to without AMC. This is due to filtering out excessive false positives as "No Signal". In high SNR, when false positives are low, AMC can contribute error by misclassifying the modulation type. The detection is still found, but mislabelled.

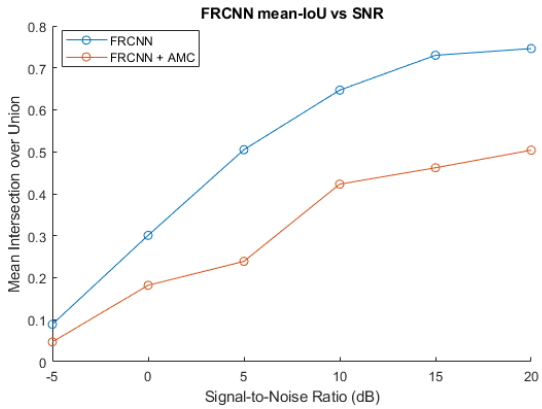

Fig. 7: Mean-IoU versus SNR for FRCNN.

We measure the reduction in computational complexity through the inference time. This is measured as the time between inputting a signal into FRCNN, and the time at which detections have been gathered. The times were measured using the same hardware, to reduce impact from different hardware. For FRCNN before our optimizations, the average inference time was measured as 3.18s, while after optimization it was measured as 0.68s, a speedup of approximately 5x.

A. *Over-the-Air Tests*

We evaluated our system using two USRP N2901 SDRs. The USRPs were configured with the RF parameters shown in Table III. The system was setup to continuously broadcast these samples, while simultaneously receiving and performing spectrum sensing through FRCNN. The received FFT was plotted, along with each signal's interval. The receiver output for a single frame could be seen in Fig. 8.

To create the hardware setup, GNURadio was used to control the SDRs, and Python was used to perform signal processing. We used ZMQ to transfer samples between a python script and GNURadio. The FFT is performed on baseband samples, and then normalized before being passed to the FRCNN architecture to detect each signal. A transmit process runs in parallel with the receive process, to produce test samples to detect. The signals are generated using the same process as the MATLAB synthesis, making use of the CommPy library [13]. We measured both the mAP and mIoU for this setup as 0.88 for both. The mAP remained high when tested OTA, suggesting that the model generalized to real-world scenarios well. Likewise, the mIoU decreased only

slightly, suggesting that the localization performance of the detector generalizes well.

Parameter	Value	Unit
Center Frequency	2.4	GHz
Baseband Sample Rate	200	kHz
Minimum Signal Bandwidth	10	kHz
Maximum Signal Bandwidth	50	kHz

TABLE III: RF Hardware Parameters

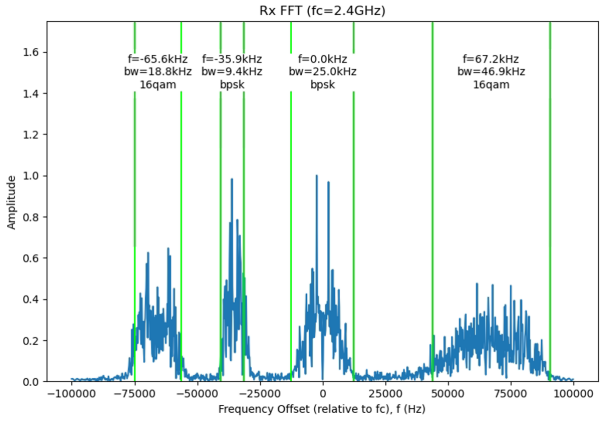

Fig. 8: OTA Test. Note that this is a representation of the spectrum, and not the input to the network.

VI. Conclusion

In this research, we optimized FRCNN for 1D spectrum sensing, allowing 1D signals to be processed. This improved inference time and reduced computational complexity. We measured a speed up of approximately 5x. The FFT was used to preprocess received signals, prior to spectrum sensing. The network was trained using synthesized RF signals, and evaluated with both synthesized and OTA data. We found the mAP to be 0.99 without AMC, and 0.72 with AMC, over the SNR interval 10dB to 40dB, achieving approximately 0.12 higher than current literature. Additionally, we measured the mIoU to be comparable to current literature, as 0.90 without AMC, and 0.48 with AMC. The decrease with AMC is due to misclassification. Finally, we utilized SDR to test our architecture OTA, to evaluate the feasibility in a real world environment. The mAP for this was measured to be 0.88, with AMC, which was significantly higher than other research that provided OTA metrics. Our tests suggest that optimized network generalizes well to real-world scenarios. In future work, the FRCNN algorithm should be improved at lower SNRs, increasing the ability to be applied in more diverse environments. Additionally, the introduction of a classifier network, such as AMC, decreased the performance of the network. More research should be done to evaluate how to improve this.

Acknowledgements

This work was supported by AFRL Beyond 5G SDR University Challenge program, the University of Massachusetts Dartmouth's Marine and Undersea Technology (MUST) Research Program funded by the Office of Naval Research (ONR) under Grant No. N00014-20-1-2170, and ONR Naval Engineering Education Consortium (NEEC) program under Grant No. N00174-22-1-0008. Thank you to Yinghan Xu for making his FRCNN code publicly available, which was used to understand and implement our transformed version [14].

References

[1] Y. Ghanney and W. Ajib, "Radio Frequency Interference Detection using Deep Learning," in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, May 2020, pp. 1–5, iSSN: 2577-2465.

[2] S. Kayraklik, Y. Alagöz, and A. F. Coşkun, "Application of Object Detection Approaches on the Wideband Sensing Problem," in *2022 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, Jun. 2022, pp. 341–346.

[3] K. N. R. S. V. Prasad, K. B. Dsouza, V. K. Bhargava, S. Mallick, and H. Boostanimehr, "A Deep Learning Framework for Blind Time-Frequency Localization in Wideband Systems," in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, May 2020, pp. 1–6, iSSN: 2577-2465.

[4] K. N. R. S. V. Prasad, K. B. D'souza, and V. K. Bhargava, "A Downscaled Faster-RCNN Framework for Signal Detection and Time-Frequency Localization in Wideband RF Systems," *IEEE Transactions on Wireless Communications*, vol. 19, no. 7, pp. 4847–4862, Jul. 2020, conference Name: IEEE Transactions on Wireless Communications.

[5] A. Vagollari, V. Schram, W. Wicke, M. Hirschbeck, and W. Gerstacker, "Joint Detection and Classification of RF Signals Using Deep Learning," in *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, Apr. 2021, pp. 1–7, iSSN: 2577-2465.

[6] T. O'Shea, T. Roy, and T. C. Clancy, "Learning robust general radio signal detection using computer vision methods," in *2017 51st Asilomar Conference on Signals, Systems, and Computers*, Oct. 2017, pp. 829–832, iSSN: 2576-2303.

[7] J. Xie, J. Fang, C. Liu, and X. Li, "Deep Learning-Based Spectrum Sensing in Cognitive Radio: A CNN-LSTM Approach," *IEEE Communications Letters*, vol. 24, no. 10, pp. 2196–2200, Oct. 2020.

[8] J. Xie, C. Liu, Y.-C. Liang, and J. Fang, "Activity Pattern Aware Spectrum Sensing: A CNN-Based Deep Learning Approach," *IEEE Communications Letters*, vol. 23, no. 6, pp. 1025–1028, Jun. 2019, conference Name: IEEE Communications Letters.

[9] C. Gravelle, T. Morehouse, and R. Zhou, "Deep Learning-Enabled Real-Time Recognition of Wireless Signals," in *2019 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, Nov. 2019, pp. 1–2, iSSN: 2334-3125.

[10] T. Morehouse, N. Rahimi, M. Shao, and R. Zhou, "Baseband Modulation Classification using Incremental Learning," in *2020 IEEE 63rd International Midwest Symposium on Circuits and Systems (MWSCAS)*, Aug. 2020, pp. 225–228, iSSN: 1558-3899.

[11] T. Morehouse, C. Montes, M. Bisbano, J. F. Lin, M. Shao, and R. Zhou, "Incremental learning-based jammer classification," in *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications III*, T. Pham, L. Solomon, and M. E. Hohil, Eds. Online Only, United States: SPIE, Apr. 2021, p. 78. [Online]. Available: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/11746/2588003/Incremental-learning-based-jammer-classification/10.1117/12.2588003.full

[12] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," Jun. 2015. [Online]. Available: https://arxiv.org/abs/1506.01497v3

[13] V. Taranalli, "CommPy," Mar. 2022, original-date: 2012-01-29T17:57:27Z. [Online]. Available: https://github.com/veeresht/CommPy

[14] Y. Xu, "Faster R-CNN (object detection) implemented by Keras for custom data from Google's Open Images…," Feb. 2019. [Online]. Available: https://towardsdatascience.com/faster-r-cnn-object-detection-implemented-by-keras-for-custom-data-from-googles-open-images-125f62b9141a