TinyML-Empowered On-Device Spectrum Sensing

Filip Angheluta
Supervisor: Prof. Qing Wang

EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 24, 2024

Name of the student: Filip Angheluta
Final project course: CSE3000 Research Project
Thesis committee:

Abstract

This paper, in answering the question "Can efficient on-device spectrum sensing be achieved on microcontrollers?", presents a simple yet comprehensive approach to signal classification using Convolutional Neural Networks (CNNs) optimized for deployment on resource-constrained devices. Using data generated via MATLAB's Wireless Toolbox, as well real world data obtained from testbeds, we created a robust dataset of 9000 samples for training our model. The steps we took while developing a CNN model that performs efficiently on microcontrollers include: data augmentation (preprocessing), model compression and quantization. The model significantly outperformed baseline accuracy metrics and maintained competitive inference times, despite the hardware limitations of microcontrollers. This reinforces the idea that Deep Learning has great potential in signal classification. Our research has the potential of being applied to smart homes, IoT networks, industrial automation, and public safety, where our optimized model facilitates efficient spectrum utilization and minimizes interference.

1 Introduction

Wireless communication technologies have played an important role in the development of our society as we know it today. The absence of coordination of these coexisting technologies, such as WiFi, Bluetooth and Zigbee, results in interference problems, which can lead to unreliable or slow communication. Spectrum sensing and signal classification are important for ensuring good quality communication despite this spectrum congestion, particularly in embedded systems, where there are strict limits on computing power. While Deep Learning (DL) has shown promise in accurate signal classification on various frequency bands, implementing DL-based methods on resource constrained devices is still an unsolved problem. Existing approaches, such as the usage of In-phase and Quadrature (I/Q) samples or Short-Time Fourier Transform (STFT), struggle to strike a balance between high accuracy and low latency on-device signal classification.

Exploring related research, we take note of several efforts in signal classification using both I/Q samples and STFT. One study focused on classifying Bluetooth, WiFi, and Zigbee signals in the 2.4 GHz band, achieving high accuracies using a dual-channel Convolutional Neural Networks (CNN) and a novel technique called Spectrum Painting [2]. Similarly, STFT-based approaches have shown promise, offering improved accuracy compared to I/Q-based methods, at the expense of increased computational demands and time. Attempts to address the latency of STFT-based classification through techniques like cropping have been suggested, aiming to find a balance between accuracy and computational complexity [4].

In addition, other studies have highlighted the effectiveness of CNNs in identifying and classifying various wireless signals. For example, Schmidt et al. demonstrated the potential of CNNs in wireless interference identification [5], while Zhang et al. focused on signal detection and classification in shared spectrum environments using deep learning [7]. These studies highlight the versatility and robustness of deep learning techniques when used for complex signal analysis tasks.

Our paper proposes several ways to improve the inference time and model size of TinyML-based architectures. Empowered by previous research on the matter, we focus on model compression and data augmentation. Furthermore, we highlight potential practical applications of spectrum sensing in real-world scenarios.

2 Background and Motivation

The wireless technologies discussed in this paper, WiFi, Bluetooth, and Zigbee, have all become incorporated into our lives by facilitating seamless connectivity in homes, workplaces, and various industrial environments. These technologies often operate in the same frequency bands, particularly the 2.4 GHz band, leading to potential interference and communication issues. Effective spectrum sensing and signal classification are a useful tool for managing this spectrum congestion, ensuring reliable communication.

Spectrum sensing involves detecting and identifying different signals within a given frequency band, enabling dynamic spectrum access and efficient spectrum utilisation. Traditional methods of spectrum sensing, such as energy detection and matched filtering, often struggle with distinguishing between different types of signals and handling varying noise levels. As a result, there has been a growing interest in applying deep learning techniques, particularly Convolutional Neural Networks (CNNs), to enhance the accuracy of spectrum sensing [6].

The primary motivation for this research is to address the challenges posed by the coexistence of multiple wireless technologies in the same frequency band. Interference between WiFi, Bluetooth, and Zigbee signals can lead to unreliable communication, reduced data throughput, and degraded performance of wireless networks. Efficient spectrum sensing can help with some of these issues by enabling methods like dynamic spectrum management and interference avoidance [1].

Deep learning techniques, specifically CNNs, have proven themselves to be useful for various signal processing tasks due to their ability to automatically learn and extract features from complex data. However, deploying these models on resource-constrained devices, such as microcontrollers, can be challenging.

This research aims to use CNNs for multi-label classification of wireless signals, focusing on improving inference time and model size to make deployment on resource-constrained devices feasible. We aim to prove the feasibility of efficient spectrum sensing that can be practically deployed in real-world environments.

3 Problem description

In this section, we discuss the problem our research addresses, looking at the nature of the data, the method of classification, and the resource constraints that were considered.

A spectrogram is a visual representation of a signal, which shows the magnitudes of frequency components over time. In our case, we will obtain 64x64 size images, where rows represent segments of time, columns represent frequency components and the color intensity represents the magnitude of the frequency. We considered a time slice of roughly 11 milliseconds for each spectrogram, and the frequency band from 2.4GHz to 2.48Hz.

In answering our research question "Can efficient on-device spectrum sensing be achieved on microcontrollers?", we are looking to efficiently classify signals by creating a spectrogram out of a small slice of the signal and identifying the presence of the following labels: Bluetooth, WiFi and Zigbee. This kind of problem is known as multi-label classification, a variant of the classification problem where multiple non-exclusive labels may be assigned to each instance.

Fig. 1 and fig. 2 show sample spectrograms from our dataset, with the presence of Bluetooth (B), Wifi(W) and Zigbee(Z) marked using one letter for each label.

When attempting to solve this problem, one should keep in mind the resource constraints of microcontrollers, and aim to create a model that is as small as possible, such that it fits into memory and minimizes inference time.

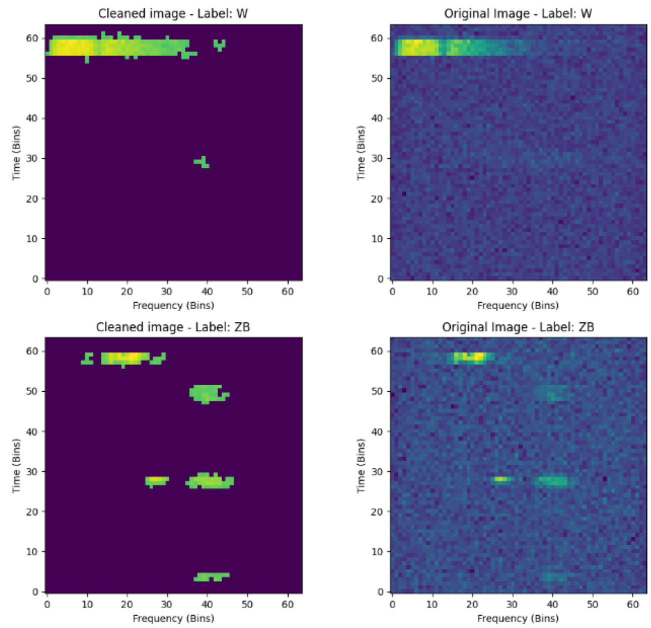

Figure 1: Wild environment samples from the testbed

4 Data generation and collection

For the purpose of evaluating our CNN model, we generated synthetic wireless signals using MATLAB's Wireless Toolbox. This toolbox allowed us to simulate overlapping Bluetooth, WiFi and Zigbee signals with various signal-to-noise ratios (SNR). This ensures that our dataset covers a wide range of real world scenarios.

Furthermore, we were granted access to the wild environment data collected from the testbed described in the Spectrum Painting paper [2], which we incorporated into our dataset.

Our complete dataset consists of 9000 spectrograms out which 5000 are synthetic and 4000 from the wild testbed data. These can be grouped together by the label or SNR. Considering the 3 labels (B, W, Z), there are 8 possible combinations for the presence/absence of each label (N - no label, B, BW, BZ, BWZ, W, ...). These scenarios are spread evenly across the 5000 spectrograms. The 5 different considered SNR values are also spread evenly across the generated dataset, whereas the testbed data has the same SNR throughout. The SNR values are as follows, as shown in fig. 3: 5dB, 10dB, 15dB, 20dB, 25dB, 30dB.

Before entering the CNN, each signal sample must undergo the following pre-processing steps: Short-Time Fourier Transform (STFT), downsampling to 64x64 and data augmentation. STFT is the method used to generate our spectrogram from the wireless signals. As for the data augmentation, it involves removing values below a treshold of 10% of the max value present in the current spectrogram and removing all connected components from the image if they are below what could reasonable be a wireless communication signal feature (in this case, any connected component smaller than 4 pixels), as seen in fig. 1 and fig. 2.

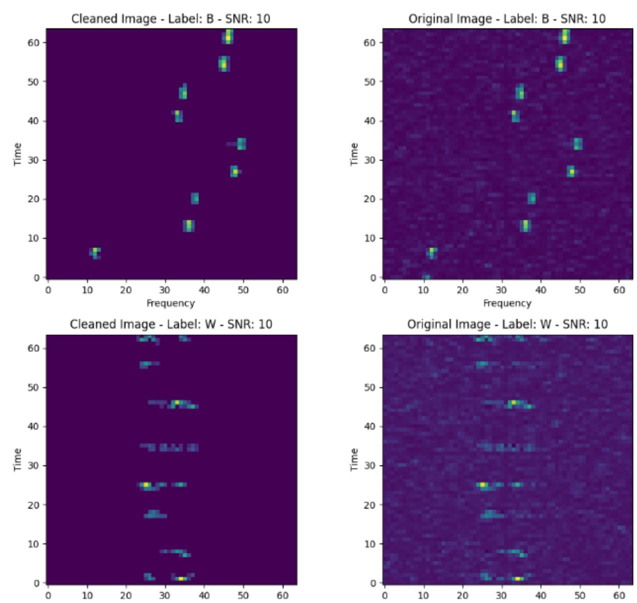

Figure 2: MATLAB Generated waveform samples

5 Model Design

Our Convolutional Neural Network (CNN) model is designed to perform multi-label classification of spectrograms, and it consists of three convolutional blocks, followed by a fully connected layer, and an output layer.

Our CNN achieves multi-label classification by having a sigmoid activation function in the output layer, which allows the model to output independent probabilities for each possible label: Bluetooth, WiFi and Zigbee. Using a carefully selected threshold, we classify each label as present if the probability reaches it.

The CNN model is a TensorFlow model, compiled using the Adam optimizer and a binary cross-entropy loss function, suitable for multi-label classification tasks. For evaluating the model's performance while training, we created a custom metric which uses the average absolute difference between predicted and actual binary labels. The model is trained for 150 epochs with early stopping criteria, which stops the training if our metric doesn't improve on the validation set for 20 epochs in a row, and always restores the weights of the CNN to the epoch that achieved the best metric on the validation set. We utilised a 70-15-15 split on the dataset for training, validation, and evaluation, respectively, to ensure accurate tuning while also testing the CNN's ability to generalize throughout the process of designing this network.

For optimization considerations, we applied post-training partial quantization to the model. This optimization technique reduces the model size and increases inference speed by converting the weights of the network when possible, from 32-bit floating point to 8-bit integers. The optimized model is then converted to TensorFlow Lite format and it's ready to be deployed on a microcontroller. This helps minimize inference time and model size, which we managed to compress down to roughly 243 trainable parameters. Similar optimization techniques have been successfully applied to other hardware platforms, such as NVIDIA's Jetson Nano, demonstrating the feasibility of deploying deep learning models on resource-constrained devices [3].

6 Evaluation

The metrics used to evaluate our model were accuracy and inference time. The inference time was measured with our model running on an Arduino BLE 33 Sense microcontroller.

The baseline evaluation metrics we chose are a part of the results obtained in the Spectrum Painting paper [2], which evaluated a few different implementations of spectrum sensing, including something very similar to our own model, a CNN that takes a 64x64 spectrogram as input, referred to in the paper as 64-STFT. We use the accuracy numbers obtained there as baseline. As for the inference time, their 64-STFT model achieved 2.7 milliseconds on a Raspberry Pi 4B. To account for the great difference in clock speed and other hardware specifications, we will expect an inference time of 100 milliseconds as a baseline.

The accuracy of our CNN is influenced by the SNR of the input signals, because high levels of noise make it difficult to identify relevant features in our spectrograms. As shown in fig. 3, our results suggest that our model and data augmentation allow for reliable classification on SNRs of above 10, with the accuracy sharply falling for values lower than that.

Furthermore when testing our model using just the testbed data, we obtain an accuracy of 90%, which is a significant improvement over the 80% baseline (fig. 4). Skipping the data augmentation step drops our accuracy right around the baseline, which suggests our method was effective in highlighting signal features and reducing noise. A similar effect can be observed in the evaluation using generated data, where skipping the augmentations seems to significantly drop the accuracy on low SNRs, which further reinforces the effectiveness of our augmentation.

We plotted a confusion matrix to show which labels are most likely to be misclassified by our model. This shows us each of the 3 labels' probability of misclassification. Given fig. 5 we can conclude that WiFi can be identified nearly perfectly, whereas classifying the signals that occupy fewer frequency bins, namely Bluetooth and Zigbee, is roughly 90% accurate.

When run on the Arduino BLE 33 Sense, our model achieves an average inference time of 180 milliseconds, which is just slightly worse than our baseline. This includes an average of 70 milliseconds for on-device pre-processing.

Overall, our model performed significantly better than our baseline in terms of accuracy, and performed slightly below the baseline in terms of inference time. This apparent underperformance in inference time is largely due to the arbitrary nature of the baseline, and the difficulty of accounting for significant hardware differences. Despite this difference, our model's performance remains within an acceptable range.

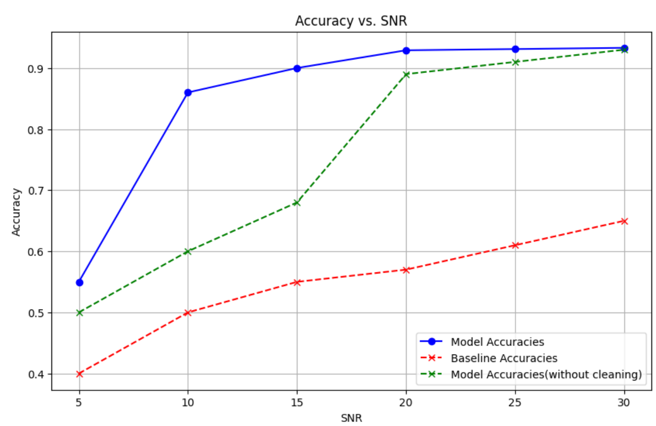

Figure 3: Accuracy evaluation using generated dataset, across different SNRs

7 Responsible research

One of the steps we took to ensure transparency and reproducibility was to make the code used for training the model, handling and processing the data, and generating the visual material present in this paper publicly available.

Furthermore, we made sure to be thorough when documenting the steps we took during the development of our

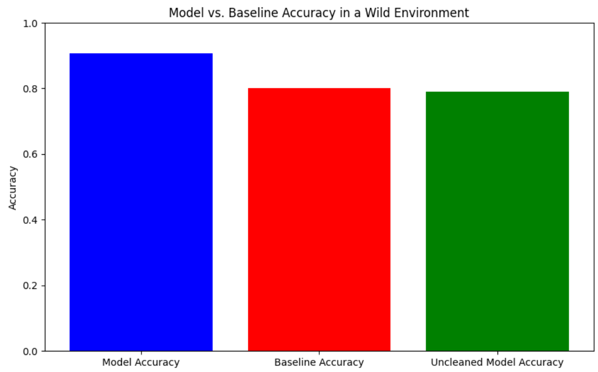

Figure 4: Accuracy evaluation using the wild environment dataset

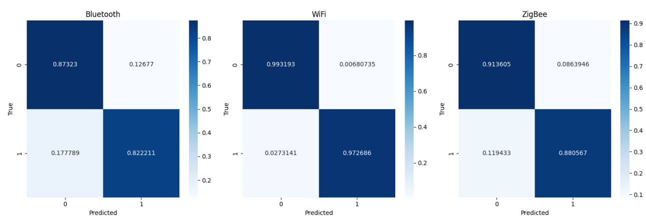

Figure 5: Confusion matrix of wild environment evaluation

model. Providing a comprehensive description of our data augmentation method, the structure and size of our model, and the integration of the testbed data makes our methodology clear and replicable. This could help future researchers understand and continue our work.

Finally, we made an effort to acknowledge the limitations of our evaluation in terms of inference time. We explained that the underperformance in inference time compared to our baseline was most likely caused by the significant hardware differences involved.

To conclude, our transparent methodology, thorough documentation and acknowledgement of limitations shows that commitment to responsible research was a fundamental aspect of our work.

8 Conclusions and Future Work

Our research addressed the challenge of designing a model that balances high accuracy and low latency such that inference, as well as all pre-processing steps, can be ran on-device, in real time, on heavily resource-constrained devices, specifically microcontrollers.

Leveraging data augmentation, model compression and quantization, we developed a CNN model that can perform well on microcontrollers, demonstrating potential practical applicability. The model outperformed baseline accuracy and showed promising inference time.

Compressing the model more is a reliable way of further decreasing the inference time, with techniques such as model pruning and quantization-aware training. These options should definitely be considered for any continuation of this research.

The practical applications of this research include a variety of crucial components of modern life, such as industrial automation, public safety, smart homes and IoT networks.

References

[1] Farrukh Aziz Bhatti, Muhammad Jaleed Khan, Ahmed Selim, and Francisco Paisana. Shared spectrum monitoring using deep learning. *IEEE Transactions on Cognitive Communications and Networking*, 7(4):1171–1185, 2021.

[2] Li Bingyang, Huang Weiqing, Wang Wen, and Wang Qing. Spectrum painting for on-device signal classification. In *IEEE WoWMoM*, pages 1–10. IEEE, 2024.

[3] Jaron Fontaine, Adnan Shahid, Robbe Elsas, Amina Seferagic, Ingrid Moerman, and Eli De Poorter. Multi-band sub-ghz technology recognition on nvidia's jetson nano. In *2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall)*, pages 1–7. IEEE, 2020.

[4] W Max Lees, Adam Wunderlich, Peter J Jeavons, Paul D Hale, and Michael R Souryal. Deep learning classification of 3.5-ghz band spectrograms with applications to spectrum sensing. *IEEE transactions on cognitive communications and networking*, 5(2):224–236, 2019.

[5] Malte Schmidt, Dimitri Block, and Uwe Meier. Wireless interference identification with convolutional neural networks. In *2017 IEEE 15th international conference on industrial informatics (INDIN)*, pages 180–185. IEEE, 2017.

[6] Adnan Shahid, Jaron Fontaine, Miguel Camelo, Jetmir Haxhibeqiri, Martijn Saelens, Zaheer Khan, Ingrid Moerman, and Eli De Poorter. A convolutional neural network approach for classification of lpwan technologies: Sigfox, lora and ieee 802.15. 4g. In *2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 1–8. IEEE, 2019.

[7] Wenhan Zhang, Mingjie Feng, Marwan Krunz, and Amir Hossein Yazdani Abyaneh. Signal detection and classification in shared spectrum: A deep learning approach. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*, pages 1–10. IEEE, 2021.