Spectrum Sensing with Tiny Machine Learning

How tiny can we go?

Author: Seth Schröder (s.d.schroeder@student.tudelft.nl) **Supervisor**: Prof. Qing Wang

TUDelft

Introduction

The wireless spectrum is super crowded causing collisions and making it harder to communicate efficiently. Devices will need to adapt by dynamically switching between different protocols and frequency bands. To do this you need to listen to the radio waves to know which bands are free to switch to – this is called **spectrum sensing**.

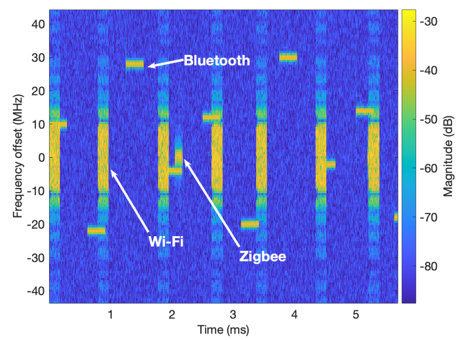


Figure 1: Spectrogram

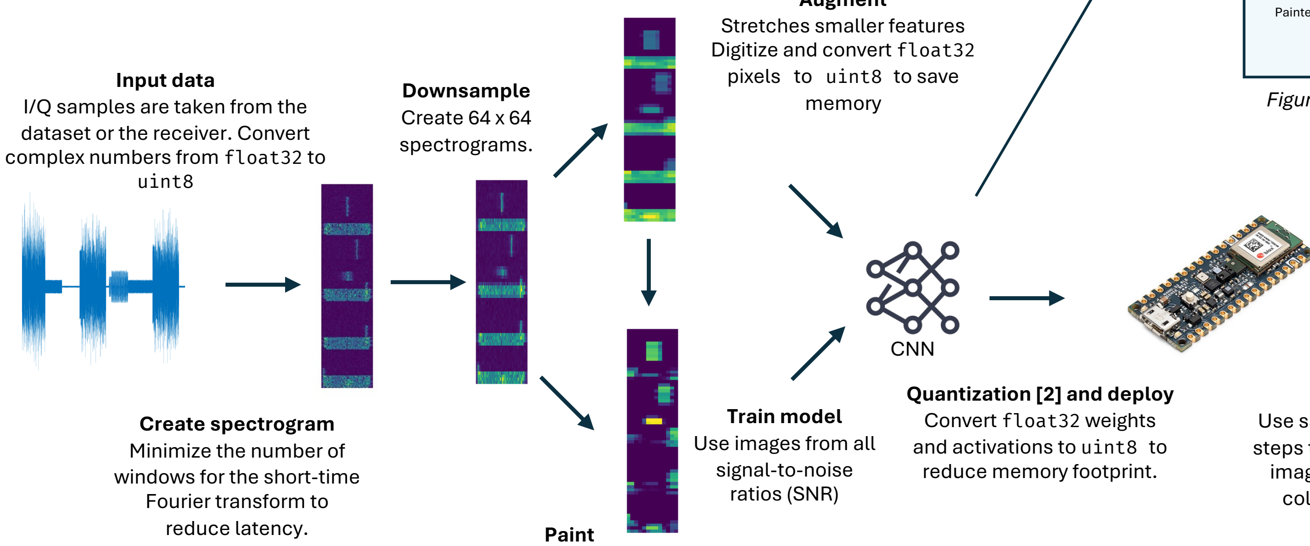
Motivation

Deep learning methods of detecting signal patterns are suitable for dynamic and chaotic radio environments. But few papers have tested these on resource-constrained devices. **Spectrum Painting** is a promising method that runs with low latency (2 ms) and high accuracy (~90%) on a Raspberry Pi 4B.

Challenge: Will it work on microcontrollers?

Methodology

The goal is to reduce the memory footprint and latency of the pre-processing and inference steps.



Input data
I/Q samples are taken from the dataset or the receiver. Convert complex numbers from `float32` to `uint8`

Create spectrogram
Minimize the number of windows for the short-time Fourier transform to reduce latency.

Downsample
Create 64 x 64 spectrograms.

Augment
Stretches smaller features
Digitize and convert `float32` pixels to `uint8` to save memory

Paint
Removes larger features.
Digitize and convert `float32` pixels to `uint8` to save memory

Train model
Use images from all signal-to-noise ratios (SNR)

Quantization [2] and deploy
Convert `float32` weights and activations to `uint8` to reduce memory footprint.

Inference
Use same pre-processing steps to create the training images on I/Q samples collected from radio receiver.

Classify
Identify which combinations of signals are present.

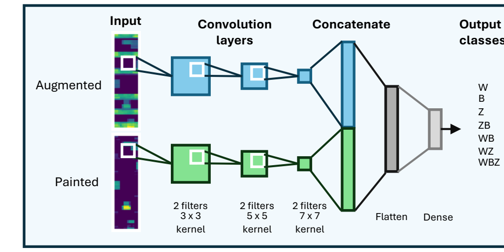


Figure 3: The CNN architecture proposed by the Spectrum Painting paper [1].

What is Spectrum Painting?

Solves the problem of detecting features in spectrograms with large features that may be occluding smaller ones (see Fig. 1). It stretches the smaller features and removes the large features (see Fig. 2). A Convolutional Neural Network (CNN) is trained on both of these images to identify which combinations of signals are present.

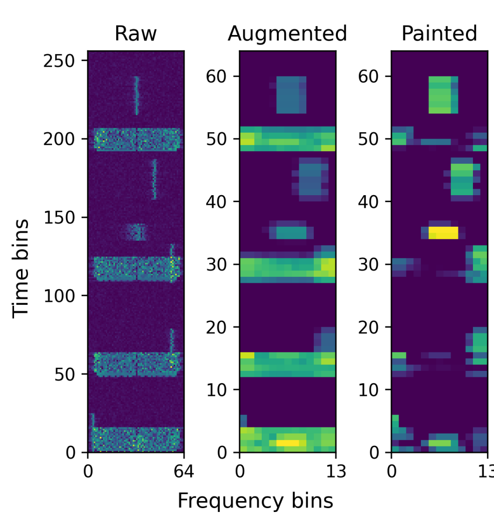


Figure 2: Spectrum Painting steps.

Results

Accuracy

8% accuracy improvement on average (Fig. 4).

- SNR 30 dB: **94.1%** accuracy vs 92.9%: ***+1.2%.***
- SNR 15 dB: **83.0%** accuracy vs 70.0%: ***+13.0%.***
- 2 filters has similar accuracy to 4 and 8 filters while having much lower latency.

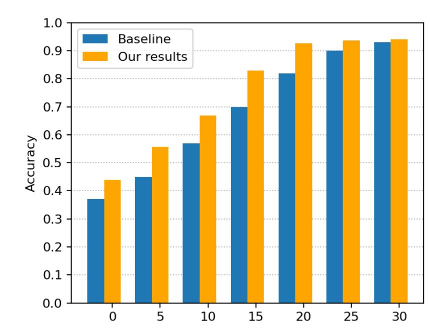


Figure 4: Our accuracy vs baseline.

Latency

- **159 ms** in total: **+31.7%.**
- 256 STFT windows gives a **68.6%** improvement over 1024 windows with similar accuracy (Fig 5). There is a loss in accuracy with fewer windows.
- Quantization reduces the latency and model size significantly for more complex models (Fig 6).

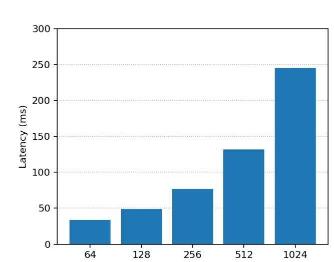


Figure 5: STFT windows vs latency.

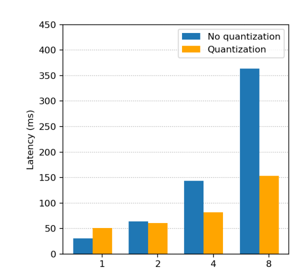


Figure 6: Quantization vs latency.

Limitations

- The dataset is generated and not tested in the wild.
- Dataset only contains Wi-Fi, Zigbee and Bluetooth signals in the 2.4 GHz band but in reality more signals, devices and sources of interference will be present.

Conclusion

Our method presents many promising optimizations for deep learning spectrum sensing techniques that use CNNs.
The next step is to optimize it further with custom hardware in a similar manner to the DeepSense paper [3]. This will decrease the latency substantially to a couple milliseconds, allowing it to be deployed on-device to decide whether to use Wi-Fi, Zigbee or Bluetooth.

References

[1] B. Li, W. Huang, W. Wang, and Q. Wang, *"Spectrum Painting for On-Device Signal Classification,"* 2024.
[2] R. Krishnamoorthi, "Quantizing deep convolutional networks for efficient inference: A whitepaper.", 2018.
[3] D. Uvaydov et al., DeepSense: Fast Wideband Spectrum Sensing Through Real-Time In-the-Loop Deep Learning, 2021