RF Fingerprinting Model

This project aims to develop a robust AI model for classifying 16 distinct Software Defined Radios (SDRs) based on their unique RF fingerprints. The core of the solution lies in leveraging a Convolutional Neural Network (CNN) to analyze subtle hardware imperfections present in the transmitted signals. These imperfections, often imperceptible to the human ear or standard signal analysis tools, act as unique signatures for each SDR. The CNN will be trained on a diverse dataset of signals from all 16 SDRs, learning to identify and differentiate the nuanced patterns introduced by variations in their physical components and manufacturing processes. The ultimate goal is to achieve highly accurate SDR classification, enabling applications such as device identification, anomaly detection, and enhanced security in wireless communication systems.

7/3/2025

Data preprocessed into a custom made dataset which is loaded into an iterable, a pytorch dataloader.

We have seen upwards of 80% accuracy on our 3 layer 1 dimensional CNN with signals emitted from 2ft and 8ft range. But less than 10% with distances ranging from 2ft to 62ft

To improve the accuracy on the entire dataset I have attempted the following:

- Batch normalization after each layer. Our matrices are only getting smaller as the convolutional operator is being applied to extract features.

- Power normalization, rather than just multiplying the input by 100 to increase the size of the values within each sample. I am using Pytorch's normalization function to keep all values near 1, especially since distance attenuation causes the signals to become incredibly small at the further ranges.

Change #1: Underfitting, training accuracy capped at 28% - Adam

- Commented out power normalization
- Increased output dimensions by double at every convolution to increase model complexity
- Increased the learning rate to 0.01 since I'm underfitting so much. Using a scheduler to decrease the learning rate every 5 epochs if the testing loss doesn't improve
- Investigating creating custom ReLU and Batch Norm functions that support complex data types

7/5/2025

- I have been reading the ORACLE research paper that this project is based on to learn more about this problem, what exactly these "hardware imperfections" are, and most importantly their CNN design. I've come across the following discoveries:

    1. Hardware imperfections are specifically referring to **IQ imbalance and DC offset**
        a. **IQ imbalance:** quadrature mixers are often impaired by gain and phase mismatches between parallel sections of the RF chain dealing with the I and Q signal paths
        b. **DC Offset:** quadrature mixers have a finite isolation between Local Oscillator and RF ports of a mixer, and a direct feedthrough from the Local Oscillator signal often gets coupled to the output.
    2. The CNN architecture
        a. IQ samples window size: 128 neurons
        b. 2 convolutional layers, 2 fully connected layers
        c. The first conv layer consists of 50 kernels, each of size 1x7 to generate 50 distinct feature maps
        d. The second conv layer consists of 50 kernels, each of size 2x7
        e. After each convolution layer there is a ReLU activation function.
        f. The output of the second conv layer is then provided as input to the first fully connected layer, which has 256 neurons.
        g. The second fully connected layer has 80 neurons to extract higher level non-linear combinations of the features from the previous layers.
        h. The classifier layer has a softmax classifier to output the probabilities of each sample being fed to the CNN.
        i. These hyperparameters were chosen carefully through cross validation.
        j. They used a dropout rate of 50% at the dense layers and a regularization parameter of 0.0001
        k. The network is trained using Adam optimizer with a learning rate of 0.0001
        l. The loss function was cross-entropy.
        m. The entire model was implemented in Keras/TensorFlow on a system with 8 NVIDIA Cuda enabled Tesla K80m GPUS. Total training time: ~30 mins
        n. They implemented early stopping with a patience of 10 epochs

Change #2:

- Adopting ORACLE's exact CNN architecture
- Sliding a window of size 128 with 50% overlap across the data
- No longer splitting our data into transmitter_distance, rather just transmitter_ID
- Splitting our data using stratified sampling to ensure an even distribution of samples per label throughout our training data, validation data, and testing data

7/10/2025
Change #3:
Adopting ORACLE's exact CNN architecture of 8 convolutional layers with alternating kernel sizes of 8 and 5, then 2 linear layers followed by a softmax classifier (implemented with 1 linear layer with 16 output channels) achieved 98%+ accuracy in the model.

Adam’s Notes

Notes on ORACLE’s dataset 2, demodulating signals and then adding impairments and associated impairment labels for classification (16 radios, 16 impairments per radio, 256 classes)

Fixed impairments are intentional, software-injected distortions introduced into a radio's transmitter chain. They are meant to exaggerate natural hardware imperfections to make radios more distinguishable but they’re not naturally occurring.

Common types of impairments:

Impairment	What It Does	Visual Effect on I/Q
IQ imbalance	Skews the amplitude and/or phase of the I and Q signals	Turns circular constellation into an ellipse
DC offset	Adds a constant voltage bias to the I or Q channel	Shifts constellation off-center
Phase noise	Adds random jitter to phase	Blurs symbols in constellation
Gain mismatch	Makes I and Q unequal in amplitude	Stretches one axis
Non-linearities	Introduces distortion when amplifying signal (clipping)	Distorts outer symbols

Notes:
Complex128 is a tuple of float64s
Complex128 = (float64, float64)

Baseband: the original, unmodulated signal
In-phase is the cosine-modulated baseband signal
Quadrature is the sine-modulated baseband signal
F_c is the carrier frequency

$z(t) = I(t) + jQ(t)$

I(t) = real part, the projection on cosine wave
Q(t) = imaginary part, the projection on the sine wave

These are nothing more than the **amplitudes** of 2, synchronized baseband channels that are split 90 degrees apart.

Impairments affect how perfect the signal is.

1. IQ Imbalance: occurs when the I and Q channels are imbalance which causes distortion in the constellation
    a. Gain mismatch: one channel is louder than the other
    b. Phase mismatch: they aren't perfectly 90 degrees apart
2. DC Offset: due to leakage from the local oscillator (LO), the signal may have a non-zero center which shifts the entire constellation off the origin