Tutorial on:

Practical Use of SDR for Machine Learning in RF Environments

Contributors:

Neel Pandeya
National Instruments

Tathagata Mukherjee
Assistant Professor
University of Alabama Huntsville
https://www.cs.uah.edu/~tm0130/

Debashri Roy
Associate Research Scientist
Northeastern University
http://www1.coe.neu.edu/~droy/

Outline

Part1: Neel (90 mins)

- Introduction to signal processing concepts for SDR, USRP radio hardware architecture, UHD device driver and UHD API, and GNU Radio
- Configuration of the USRP Radio
- Demos and examples of various SDR systems

Part2: Tathagata (30 mins)

- Introduction to ML Concepts
- Discussion of FM radio-based positioning using SDR
- Direction Finding using SDR
- Introduction to Adversarial Learning

Part3: Debashri (45 mins)

- RF ML Problems and Challenges
- Introduction to the Transmitter Identification problem
- Example of Transmitter Identification using Python Jupyter notebook

Q&A: 15 mins

Outline

Part1: Neel (90 mins)

- Introduction to signal processing concepts for SDR, USRP radio hardware architecture, UHD device driver and UHD API, and GNU Radio
- Configuration of the USRP Radio
- Demos and examples of various SDR systems

Part2: Tathagata (30 mins)

- Introduction to ML Concepts
- Discussion of FM radio-based positioning using SDR
- Direction Finding using SDR
- Introduction to Adversarial Learning

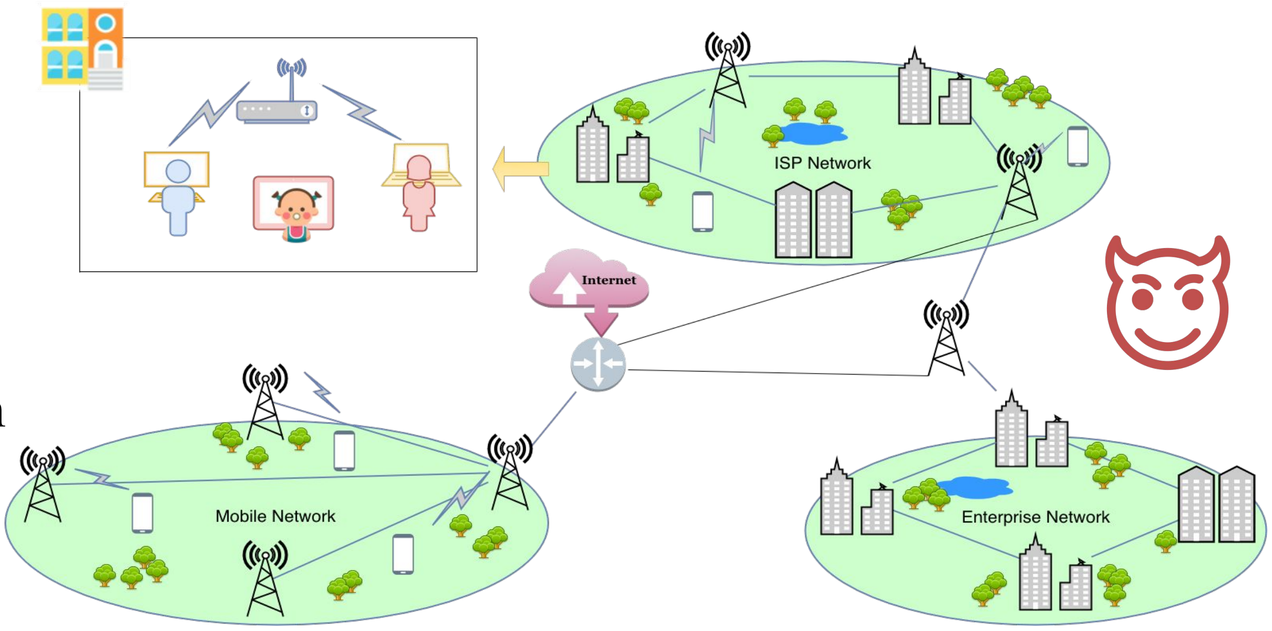
Part3: Debashri (45 mins)

- RF ML Problems and Challenges
- Introduction to the Transmitter Identification problem
- Example of Transmitter Identification using Python Jupyter notebook

Q&A: 15 mins

Wireless Networks: Current Scenario

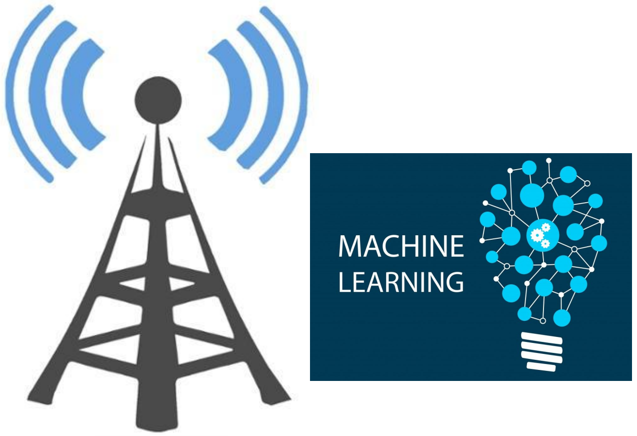
- Omnipresent
- **Backbone** of modern world wireless communication
- Still **evolving**
- Secure communication



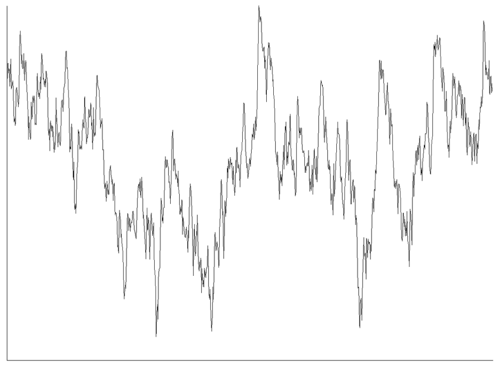
- Radio channels are fraught with **uncertainties**
  - Signal **fading** due to multi-path propagation
  - **Shadowing** due to manmade and natural objects
  - **Interference**
- **Reliability** and **Quality**:
  - Ensuring both is **challenging**
- **Security**:
  - **Real-time** communication
  - **Cryptography** techniques could be **overhead**.
  - **Way out???**

Learn about the Environment and Automate Security □ Machine Learning

Use of AI in RF Domain



Ever-changing RF Channel

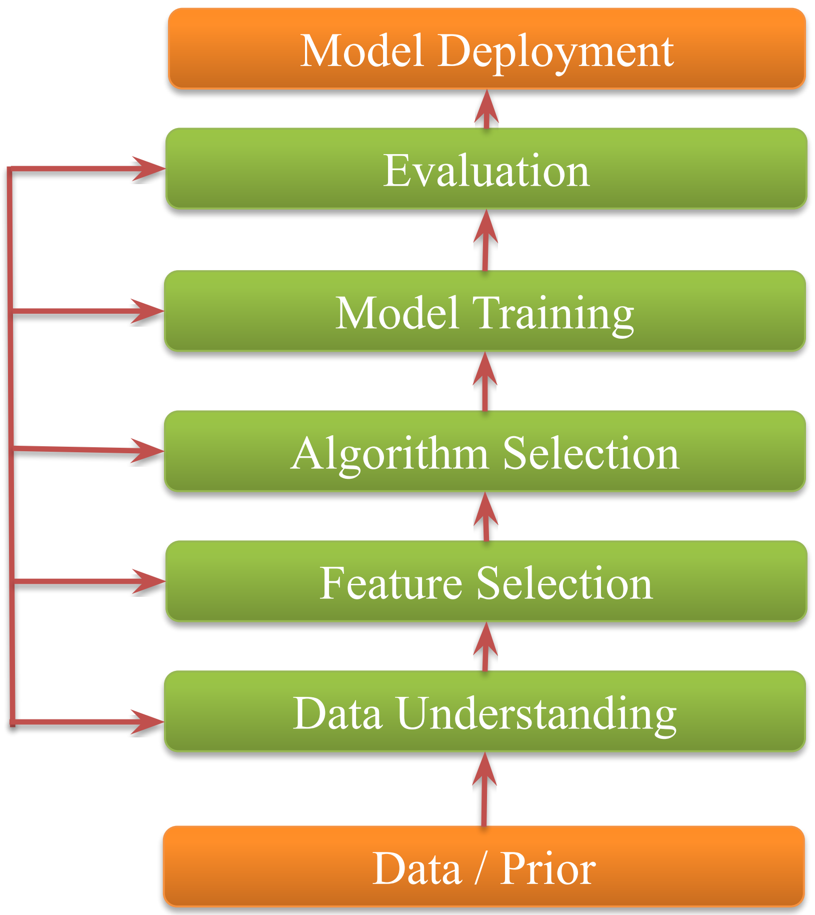


Traditional ML

Radio Frequency Machine Learning

[1] Images from Google

Machine Learning Life Cycle



Do not code the pattern, let the machine learn through the data…

I/Q Representation of RF Signal Data

- $V(t) = A * \sin(2 * \pi * f * t + \Phi)$

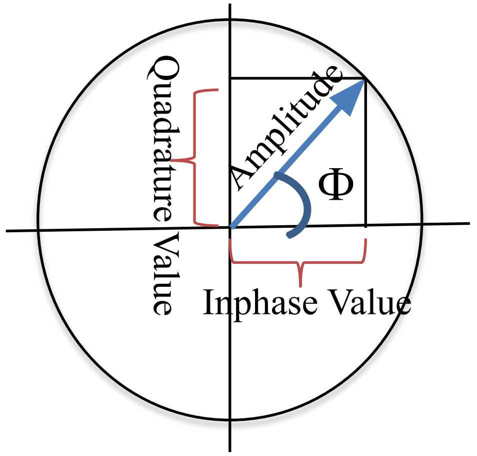
  where:

  A: peak amplitude

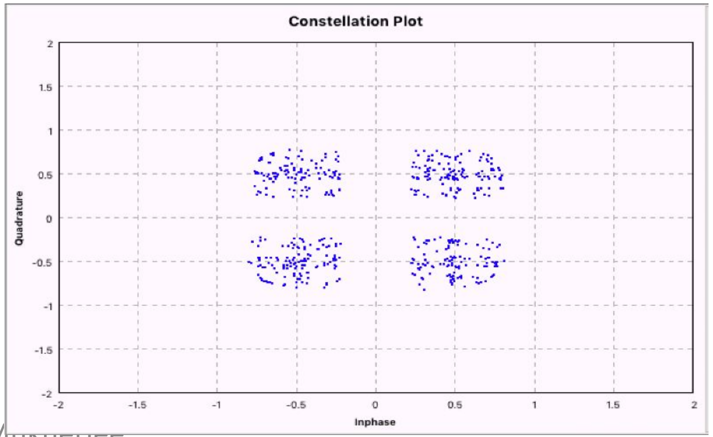
  f: frequency

  t: time

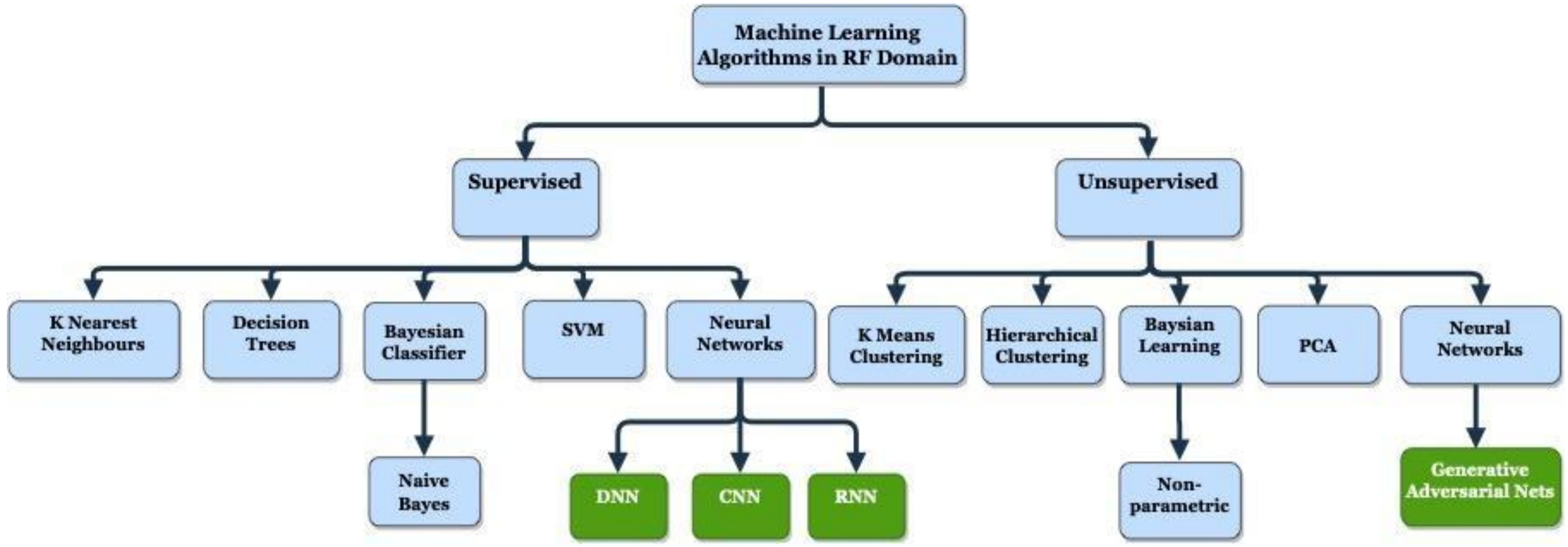
  Φ: phase shift



- These amplitude and phase changes is to **encode** information upon a sine wave **Modulation**
- Modulated **Carrier RF** $= I.\cos 2\pi f t + Q.\sin 2\pi f t$



Machine Learning in RF Domain



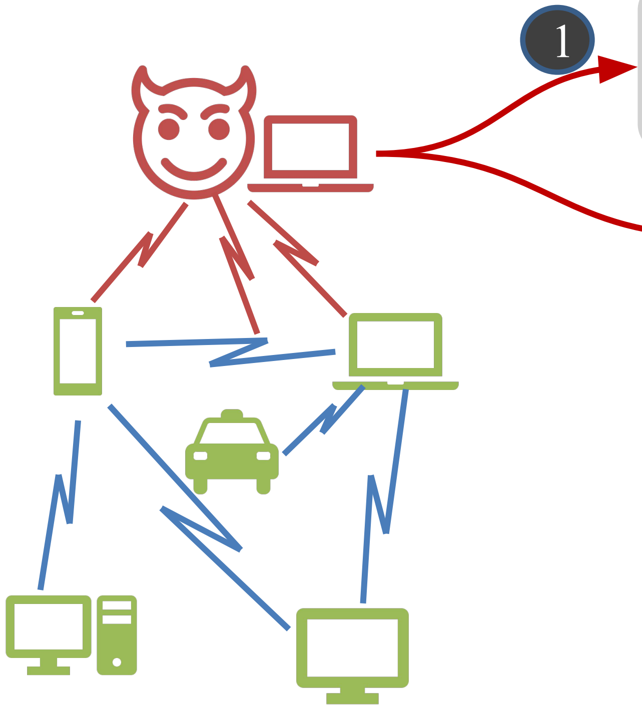
Applicability of different ML Algorithms greatly depends on **Data**.

Existing Datasets: **Synthetic and Real**

Collected Data: **Multi** dimensional and **large**

*radioML, "RFML 2016," https://github.com/radioML/dataset, 2018.

Radio Frequency Adversarial Learning (RFAL) Framework



Problems

Interfere or hack into the existing transmissions

Pose as an authentic transmitter □ Fake transmitter

Solutions

Design a ML based framework to detect such adversaries

Design a robust transmitter fingerprinting approach

Papers

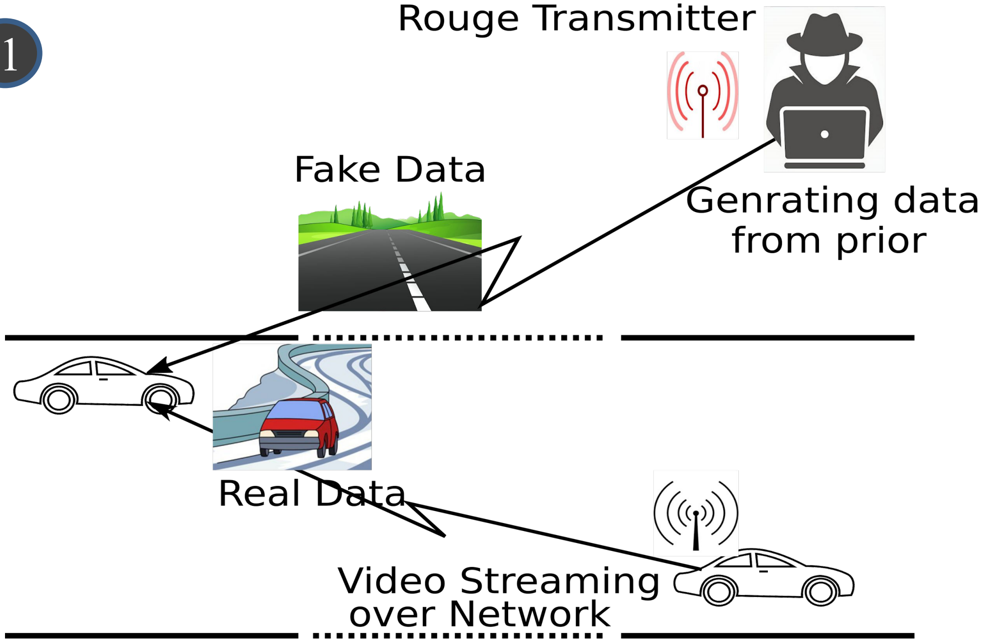
[1] Debashri Roy, Tathagata Mukherjee, Mainak Chatterjee, Eduardo Pasiliao, *Detection of Rogue RF Transmitters using Generative Adversarial Nets* , IEEE Wireless Communications and Networking Conference (WCNC), 2019.

[2] Debashri Roy, Tathagata Mukherjee, Mainak Chatterjee, Erik Blasch, Eduardo Pasiliao, *RFAL: Adversarial Learning for RF Transmitter Identification and Classification* , In IEEE Transactions on Cognitive Communications and Networking, 2020.

[3] Debashri Roy, Tathagata Mukherjee, Mainak Chatterjee, Eduardo Pasiliao, *RF Transmitter Fingerprinting Exploiting Spatio-temporal Properties in Raw Signal Data.* , IEEE International Conference on Machine Learning and Applications (ICMLA), 2019.

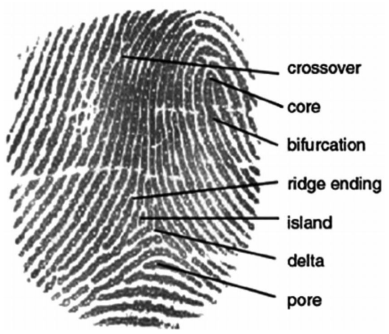
Detection of Adversaries and Transmitter Fingerprinting

1



Learn, **characterize**, and **determine** such **rouge** transmitters by proposing and implementing Generative Adversarial Networks (**GAN**) based learning techniques for **RFML** systems.

 Radio Frequency Adversarial Learning (**RFAL**) Framework



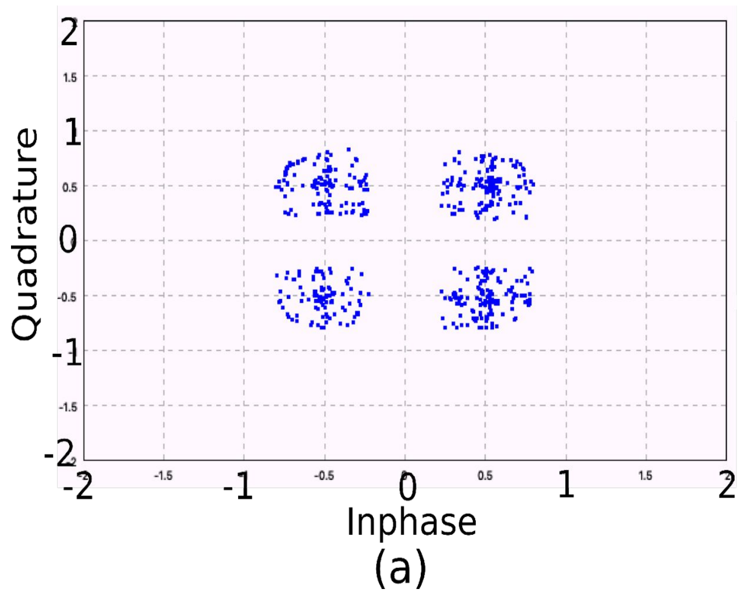
Transmitter Fingerprinting:

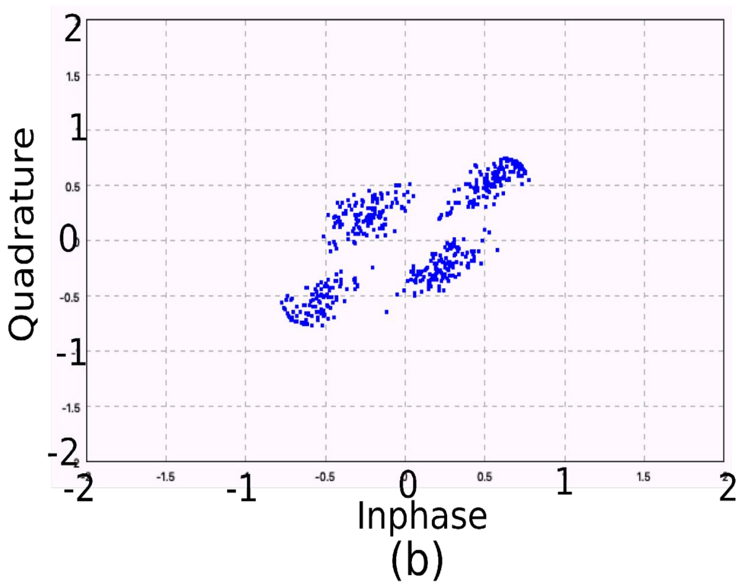
- Exploit **intrinsic** properties in RF data
- Exploit **spatial** or **temporal correlations** in the transmitted RF data.

Feature Selection: Inherent Noise

- Inherent noise imposed by radio hardware[4]:
  - Noisy mixers
  - Noisy oscillators
  - Imbalanced low pass filters
- Unique to each hardware

Learn and **characterize** that **Noise** !!!





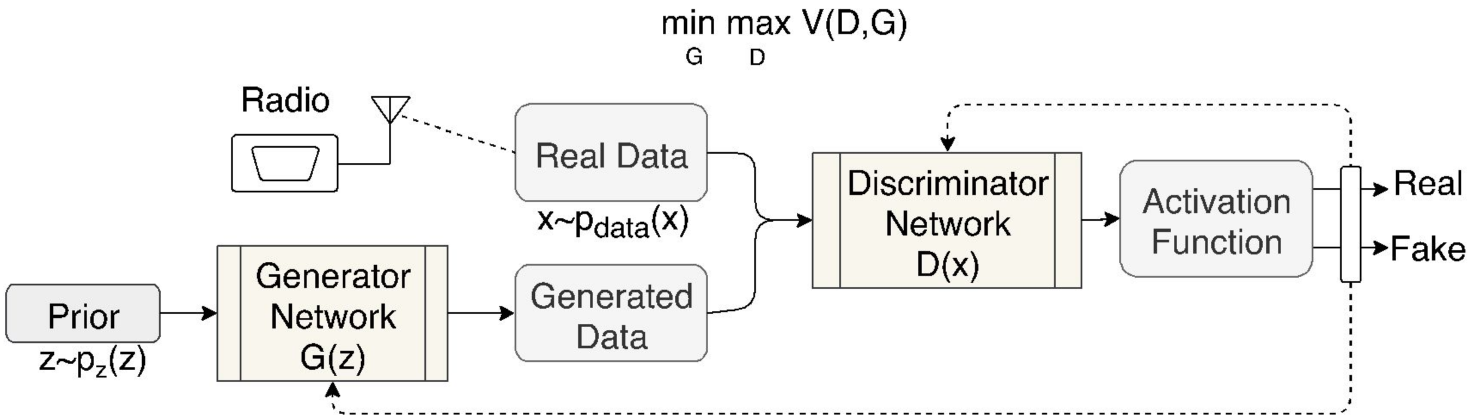
IQ Imbalance for QPSK: (a) Before (b) After 45° Phase Noise

[4]M. D. L. Angrisani *et al.,* "Clustering-based method for detecting and evaluating I/Q impairments in radio-frequency digital transmitters," IEEE Transactions on Instrumentation and Measurement, vol. 56, no. 6, pp. 2139–2146, 2007.

Generative Adversarial Nets

- **Generator (*G*):** generates the "**fake**" data and learns about **real data distribution** over **time**.

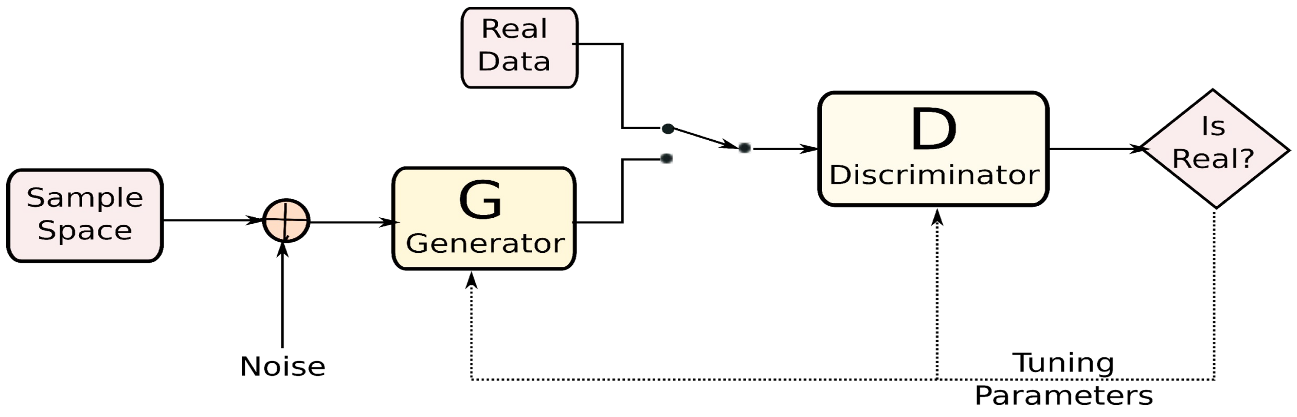
- **Discriminator (*D*):** tries to distinguish "**fake**" data from "**real**" data by estimating the probability that the sample came from real data rather than ***G***.



$$\min_G \max_D V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(x)}\left[\log\left(1 - D(G(z))\right)\right]$$

Proposed GAN Model

- Identifying fake transmitters from trusted ones.
- Some RF properties to consider:
  - Signal Phase
  - Signal Amplitude
  - Modulation Schemes
  - **Sample Space I/Q Data**

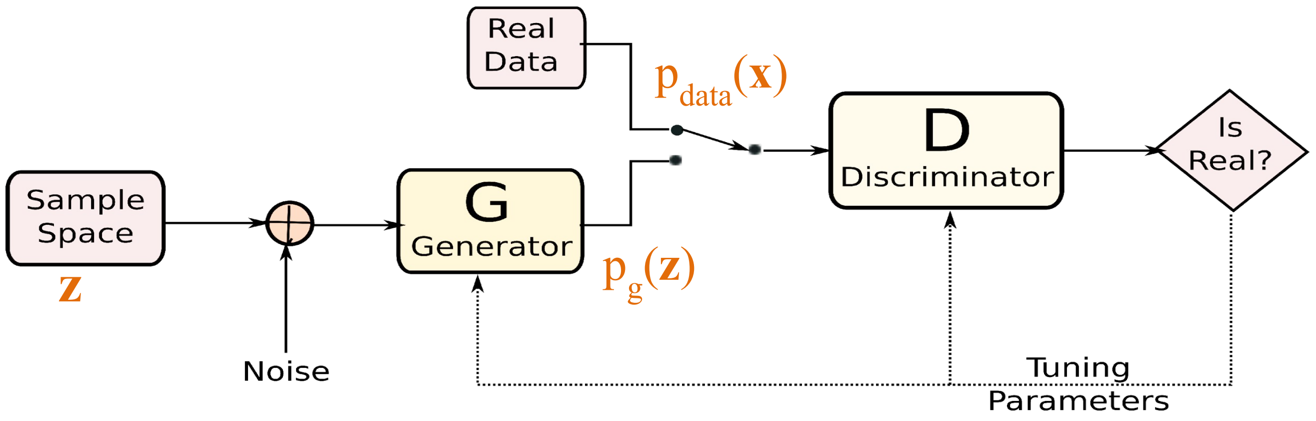


Proposed GAN Architecture

Proposed Generative Model

- Goal: to generate **fake I/Q** data by learning sample space of real data.
- Priors: a **sample** space of I/Q data

Sample space: **x**



- $p_g(\mathbf{z})$ is the generator's distribution over **z**.
- $p_{data}(\mathbf{x})$ is the data distribution over **x**.

Objective: To learn the probability distribution $p_g(\mathbf{z})$ over sample space $(\mathbf{z})$.

Proposed Discriminative Model

- Goal: Maximize the cost function:

$$C(D,G) = \mathbb{E}_{y\sim p_{data}(\boldsymbol{x})}[\log D(x)] + \mathbb{E}_{x\sim p_g(\boldsymbol{z})}[\log(1 - D(G(x)))]$$

- D(x) is the probability that x came from $p_{data}(\mathbf{x})$ than $p_g(\mathbf{z})$.

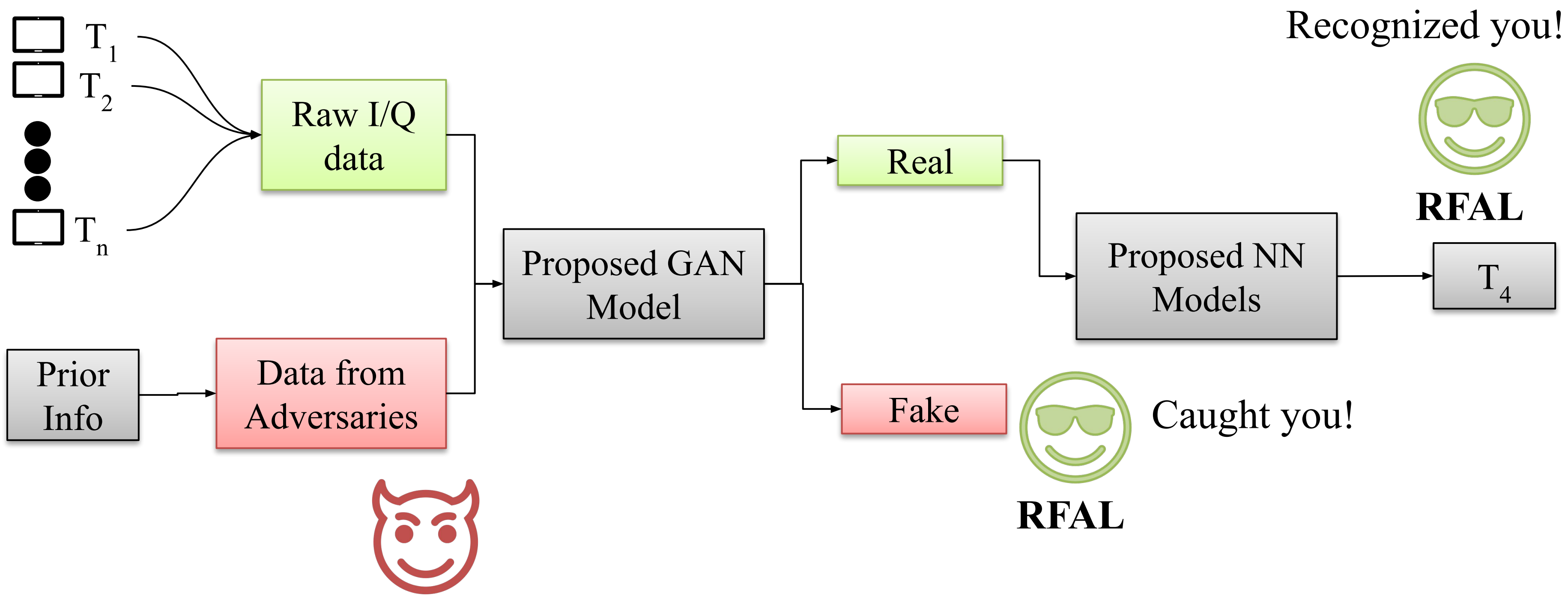
- The GAN training: $min_G max_D C(D,G)$

- One unique optimal discriminator per GAN framework:

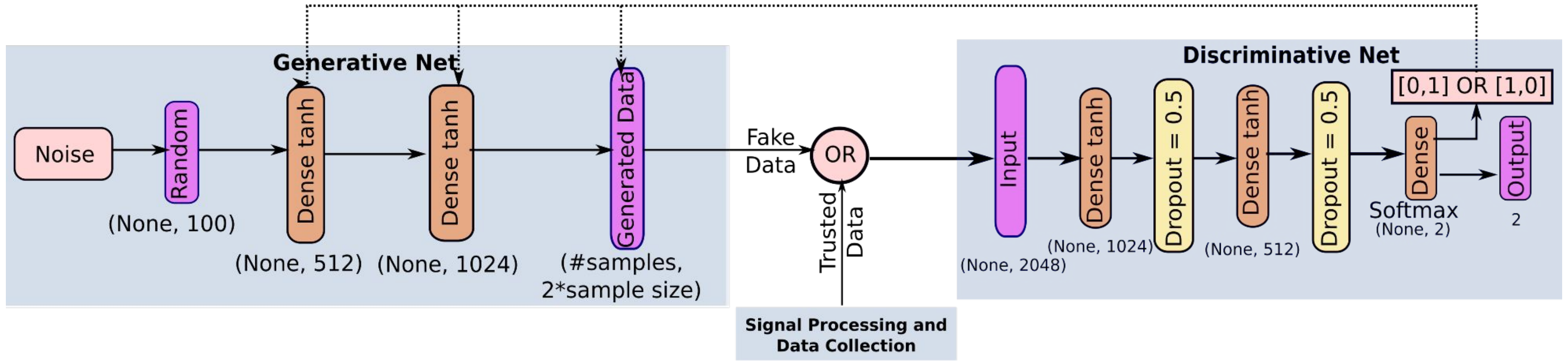
$$D^*(x) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{z})}$$

- Optimal generator when $p_g(\mathbf{z}) = p_{data}(\mathbf{x})$.

RFAL Architecture



Proposed GAN Model



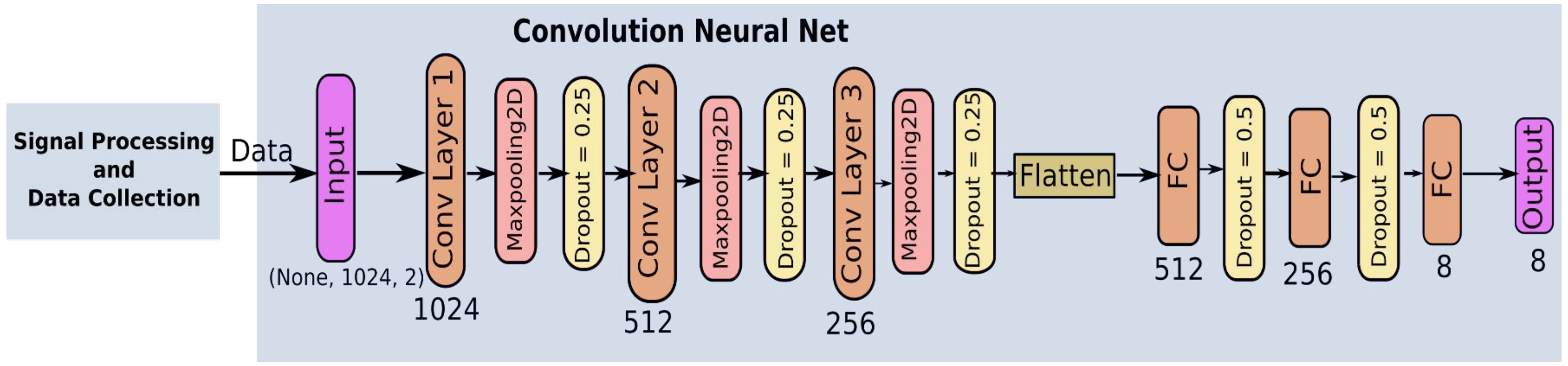
Generator
Number of layers: 3
Random number range: [-1, 1]
Activation Functions: tanh
Optimization: Adam [5]
Learning rate: 10^{-4}

Discriminator
Number of layers: 4
Activation Functions: tanh
Optimization: Adam [5]
Learning rate: 10^{-3}

GAN Model
Number of epochs: 200
Training: Categorical cross entropy

[5]D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” CoRR, vol. abs/1412.6980, 2014.

Proposed NN Models

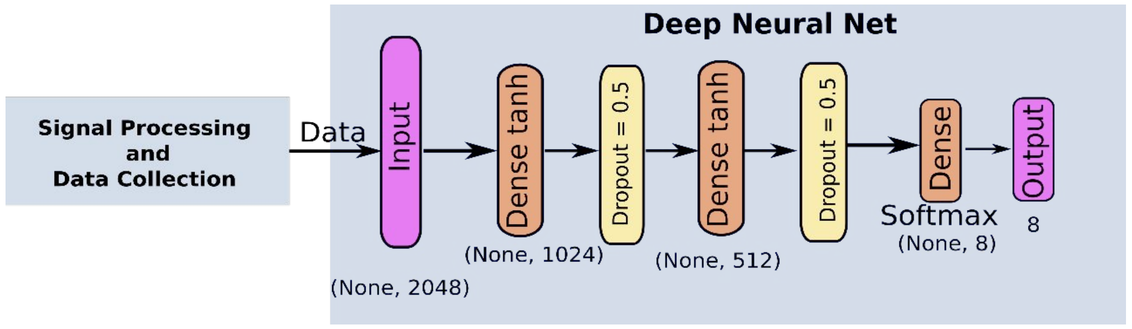


Convolutional Neural Network

Number of layers: 6
Activation Functions: ReLU
Kernel Size: (2, 3)
Conv stride: (2, 2)
Optimization: Adam [5]
Learning rate: 10^{-3}
Pool size: (2, 2)
Pool stride: (2, 2)
Training: Categorical cross entropy

[5]D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” CoRR, vol. abs/1412.6980, 2014.

Proposed NN Models

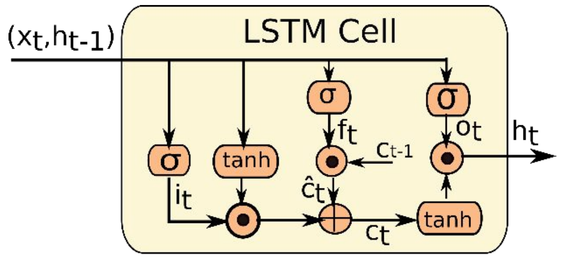


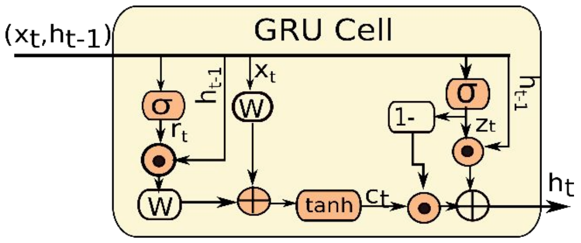
Deep Neural Network

Number of layers: 4
Activation Functions: tanh
Optimization: Adam [5]
Learning rate: 10^{-3}
Training: Categorical cross entropy

[5] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," CoRR, vol. abs/1412.6980, 2014.

Proposed NN Models: RNN





tanh	tanh activation
σ	sigmoid activation
⊕	sum over all elements
⊙	Hadamard product

tanh	tanh activation
σ	sigmoid activation
⊕	sum over all elements
⊙	Hadamard product
W	weight multiplication

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f)$$
$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i)$$
$$\widetilde{c_t} = \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_{c_{t-1}})$$
$$c_t = f_t \,.\, c_{t-1} + i_t \,.\, \widetilde{c_t}$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$$
$$h_t = o_t \,.\, \tanh(c_t)$$

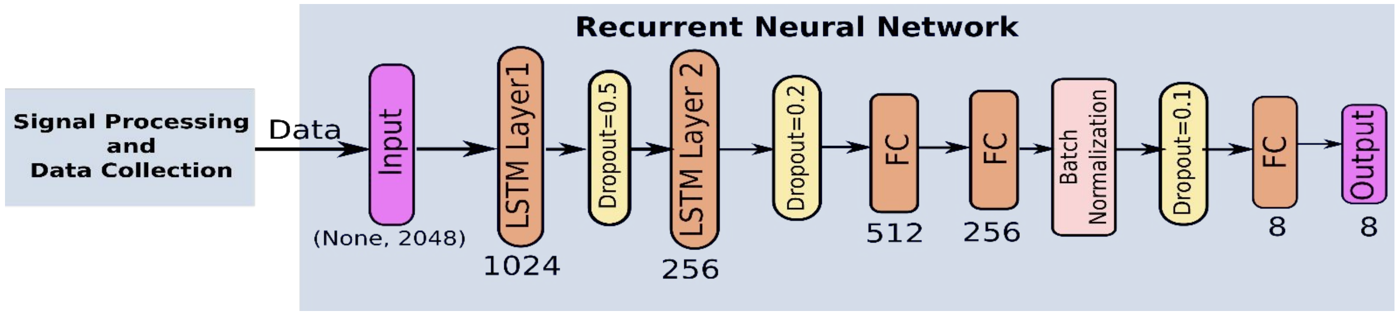
$$z_t = \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z)$$
$$r_t = \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r)$$
$$c_t = \tanh(W_{xc}x_t + W_{hc}(r_t \,.\, h_{t-1}))$$
$$h_t = (1 - z_t).c_t + z_t \,.\, h_{t-1}$$

f_t: Forget gate; i_t: Input gate; o_t: Output gate

z_t: Reset gate; r_t: Update gate

c_t: Cell state; W: Weights; b: Biases

Proposed NN Models: RNN



Recurrent Neural Network with LSTM Cells

Activation Functions:
- ReLU for LSTM layers
- tanh for dense (Fully Connected) layers

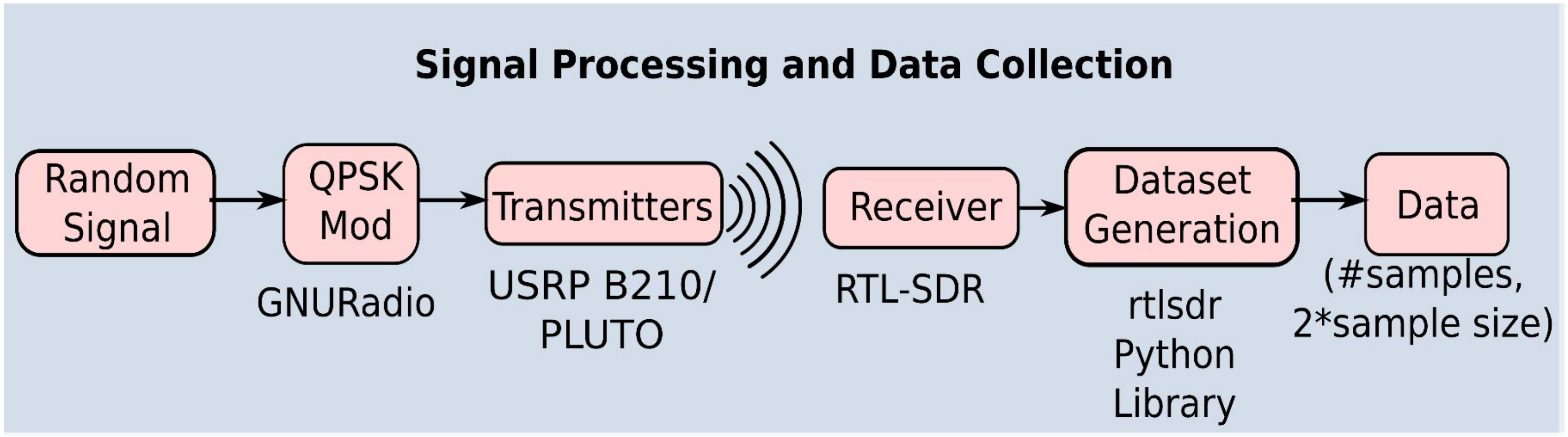
Optimization: Stochastic gradient descent

Learning rate: 10^{-3}

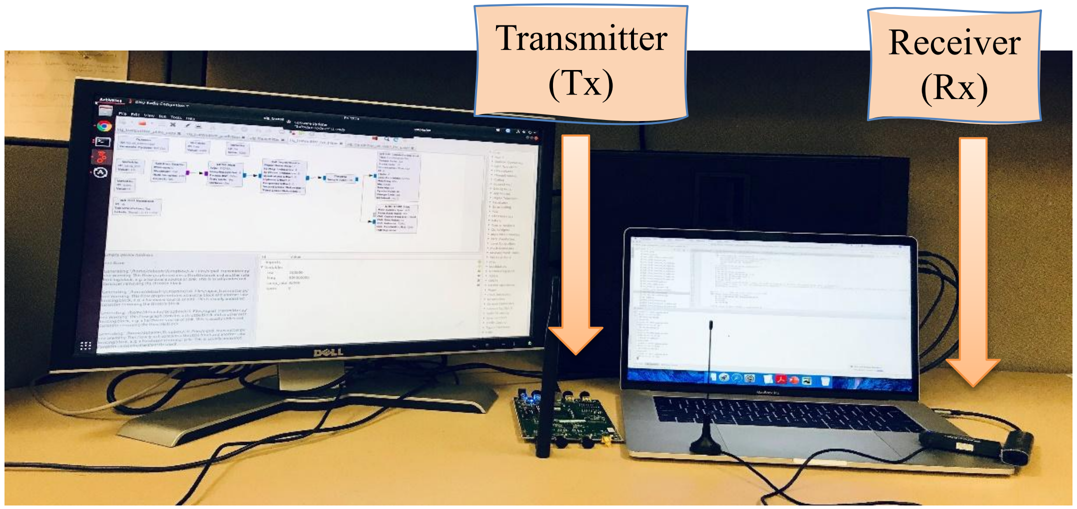
Training: Categorical cross entropy

We design another RNN model with GRU Layers with same configuration

Signal Generation



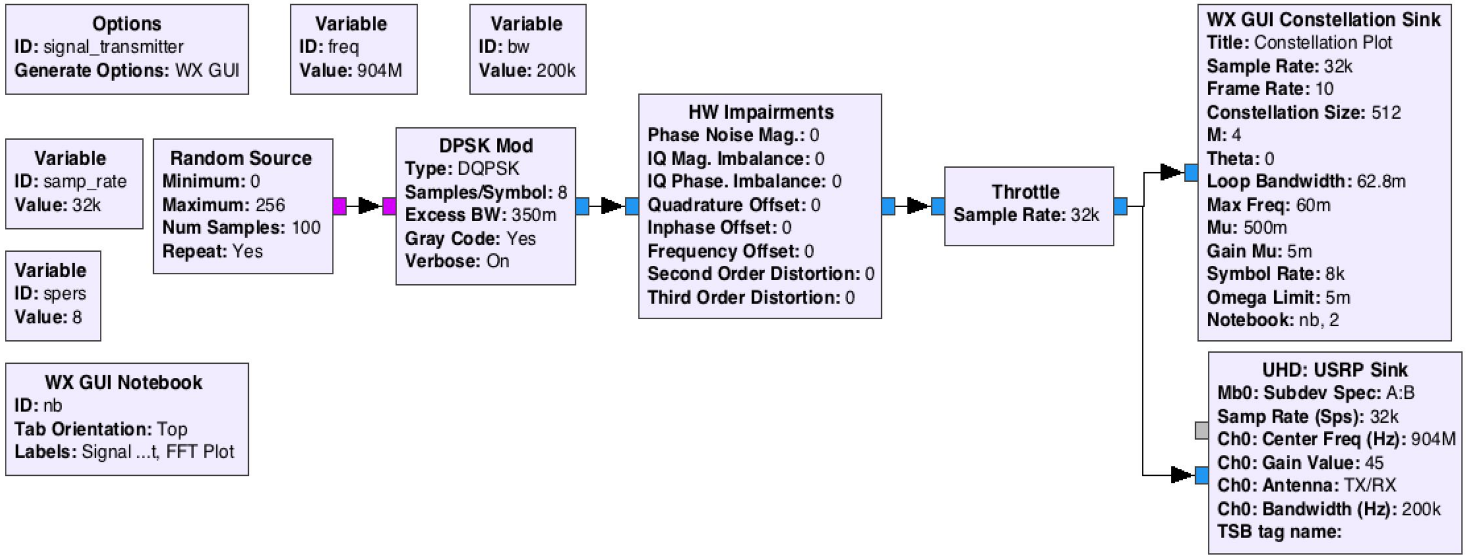
Signal Generation and Data Collection Technique



Data Collection

Transmitter (USRP B210):

- Frequency Range: 70 MHz - 6 GHz
- Used Frequency: 904 MHz (ISM)
- Gain: 45 dB
- Total transmitters: 4 * 2



Options
ID: signal_transmitter
Generate Options: WX GUI

Variable
ID: freq
Value: 904M

Variable
ID: bw
Value: 200k

Variable
ID: samp_rate
Value: 32k

Variable
ID: spers
Value: 8

WX GUI Notebook
ID: nb
Tab Orientation: Top
Labels: Signal ...t, FFT Plot

Random Source
Minimum: 0
Maximum: 256
Num Samples: 100
Repeat: Yes

DPSK Mod
Type: DQPSK
Samples/Symbol: 8
Excess BW: 350m
Gray Code: Yes
Verbose: On

HW Impairments
Phase Noise Mag.: 0
IQ Mag. Imbalance: 0
IQ Phase. Imbalance: 0
Quadrature Offset: 0
Inphase Offset: 0
Frequency Offset: 0
Second Order Distortion: 0
Third Order Distortion: 0

Throttle
Sample Rate: 32k

WX GUI Constellation Sink
Title: Constellation Plot
Sample Rate: 32k
Frame Rate: 10
Constellation Size: 512
M: 4
Theta: 0
Loop Bandwidth: 62.8m
Max Freq: 60m
Mu: 500m
Gain Mu: 5m
Symbol Rate: 8k
Omega Limit: 5m
Notebook: nb, 2

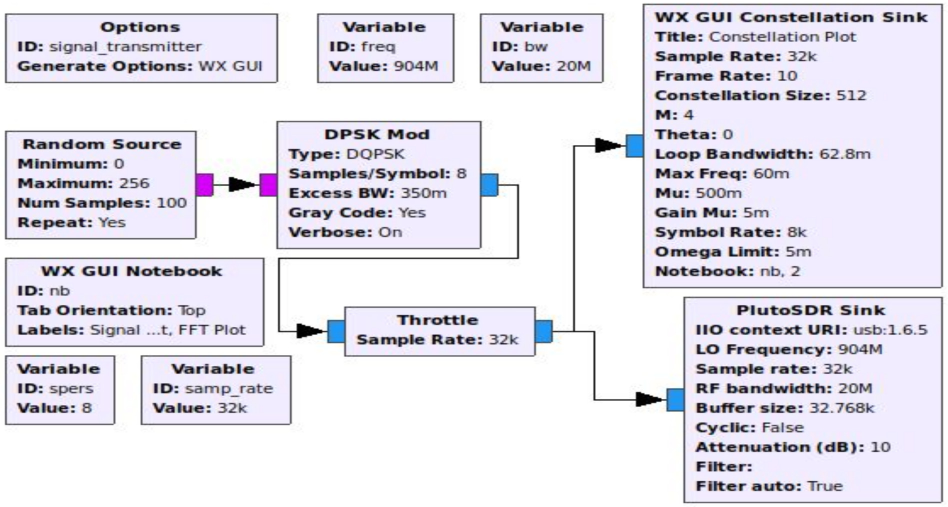
UHD: USRP Sink
Mb0: Subdev Spec: A:B
Samp Rate (Sps): 32k
Ch0: Center Freq (Hz): 904M
Ch0: Gain Value: 45
Ch0: Antenna: TX/RX
Ch0: Bandwidth (Hz): 200k
TSB tag name:

[6]http://www.ettus.com/all-products/UB210-KIT/

Data Collection

Transmitter (PADALM-PLUTO):

- Frequency Range: 325 MHz – 3.8 GHz
- Used Frequency: 904 MHz (ISM)
- Gain: 45 dB
- Total transmitters: 1



Options
ID: signal_transmitter
Generate Options: WX GUI

Variable
ID: freq
Value: 904M

Variable
ID: bw
Value: 20M

WX GUI Constellation Sink
Title: Constellation Plot
Sample Rate: 32k
Frame Rate: 10
Constellation Size: 512
M: 4
Theta: 0
Loop Bandwidth: 62.8m
Max Freq: 60m
Mu: 500m
Gain Mu: 5m
Symbol Rate: 8k
Omega Limit: 5m
Notebook: nb, 2

Random Source
Minimum: 0
Maximum: 256
Num Samples: 100
Repeat: Yes

DPSK Mod
Type: DQPSK
Samples/Symbol: 8
Excess BW: 350m
Gray Code: Yes
Verbose: On

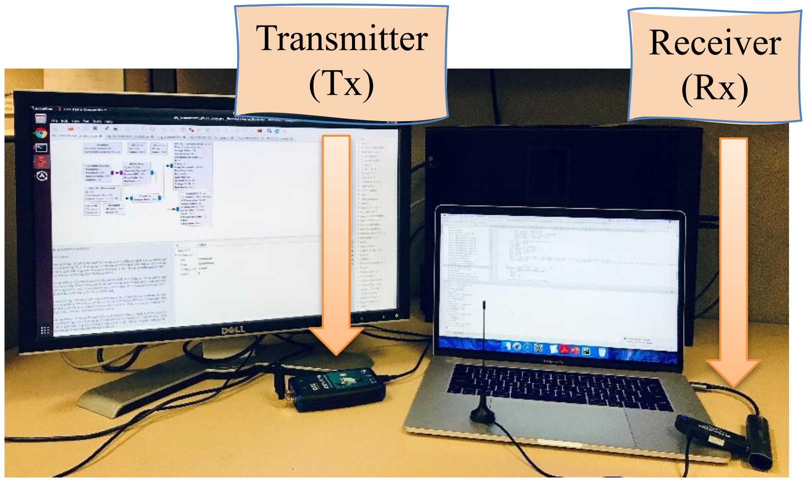
WX GUI Notebook
ID: nb
Tab Orientation: Top
Labels: Signal ...t, FFT Plot

Throttle
Sample Rate: 32k

PlutoSDR Sink
IIO context URI: usb:1.6.5
LO Frequency: 904M
Sample rate: 32k
RF bandwidth: 20M
Buffer size: 32.768k
Cyclic: False
Attenuation (dB): 10
Filter:
Filter auto: True

Variable
ID: spers
Value: 8

Variable
ID: samp_rate
Value: 32k

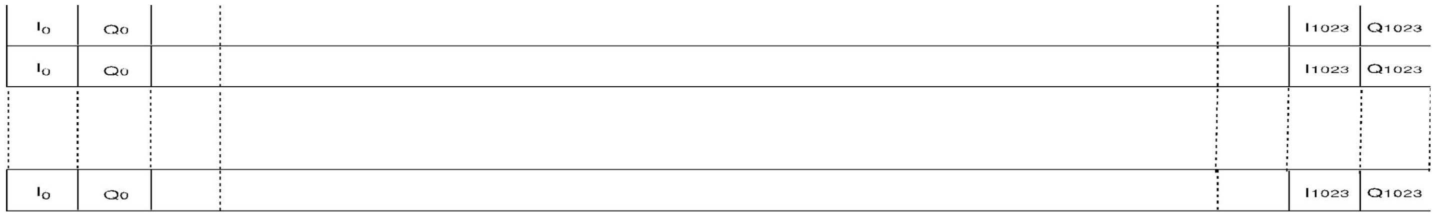


[7]https://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/adalm-pluto.html#eb-overview

Data Collection

Receiver

- Frequency Range: 500 kHz - 1766 MHz
- Used Frequency: 904 MHz
- Sample Rate: 1024



Collected Data

- Raw I/Q Signal Data: Sample size 1024
- **Homogeneous Dataset** (SNR 30dB):
  - 6.8 GB: Using 4 USRP SDRs, 160,000 rows and 2048 columns.
  - 13.45 GB: Using 8 USRP SDRs, 320,000 rows and 2048 columns.
- **Heterogeneous Dataset** (SNR 30dB):
  - 2.86 GB: Using 1 PLUTO-SDR and 1 USRP SDR, 80,000 rows and 2048 columns
- **Varying SNR:**
  - ~13 GB: 3 more datasets with 8 USRP SDRs for SNR 20dB, 10dB, and 0 dB, 320,000 rows and 2048 columns.

Experimental Setup

- Data collection on **16 GB** Intel machine.
- A Ryzen 8 Core system with **64 GB** RAM, a GTX **1080 Ti** GPU unit, and **11 GB** memory.
- Python Libraries: *tensorflow*, *keras*, *numpy*, *scipy*, and *matplotlib*

Correlation in Dataset

Representation of collected data:

$$x_t = \left[[(I,Q)_i]^t\right], \mathrm{t} = 1, 2, \dots, \mathrm{M};\ \mathrm{t} = 1, 2, \dots, \mathrm{T}] \in C^M$$

T: number of training samples; **M**: Sample size; **t**: timestamp; **(I, Q) ∈ *C*** is number in the complex plane

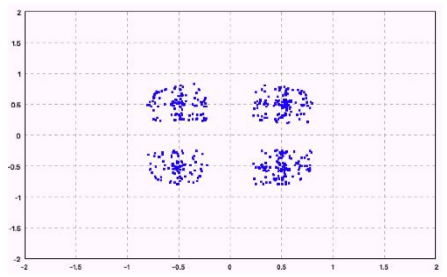
$$[I_0\, Q_0 I_1\, Q_1 I_2\, Q_2 I_3\, Q_3 \dots I_{1023}\, Q_{1023}]^t$$

Q: How to measure correlation in such dataset?

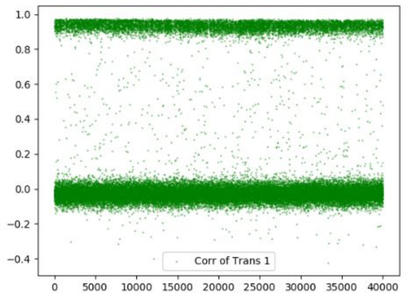
Ans: Remember, **QPSK** modulation, **($I_0I_1I_2I_3$, $I_4I_5I_6I_7$)**, and **($Q_0Q_1Q_2Q_3$, $Q_4Q_5Q_6Q_7$)**

Correlation using Pearson's method $(r) = \dfrac{\sum_{i=0}^{(M-1)}(I_i - \bar{I})(Q_i - \bar{Q})}{\sqrt{\sum_{i=1}^{(M-1)}(I_i - \bar{I})^2}\sqrt{\sum_{i=0}^{(M-1)}(Q_i - \bar{Q})^2}}$

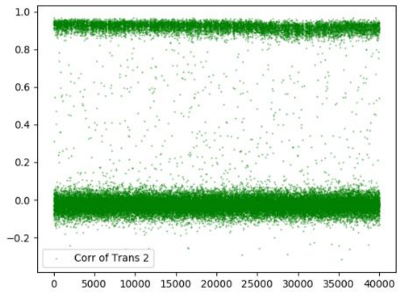
$$\bar{I} = \frac{1}{M}\sum_{i=0}^{(M-1)} I_i$$



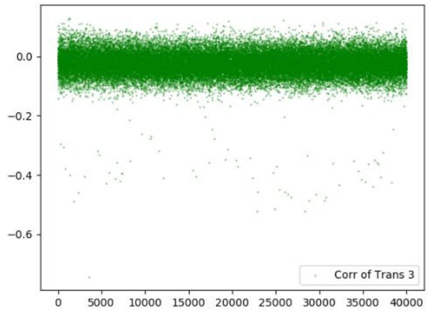
Correlation in Dataset



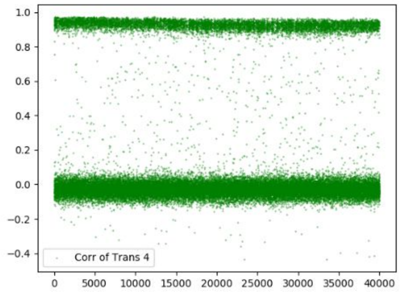
Trans ID 1



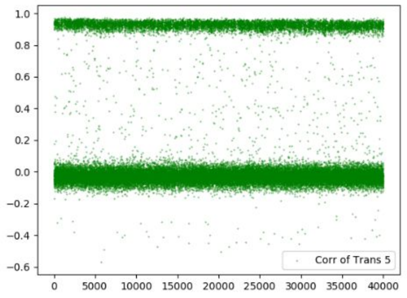
Trans ID 2



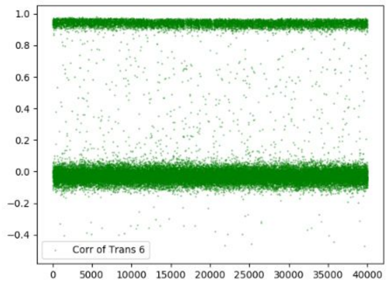
Trans ID 3



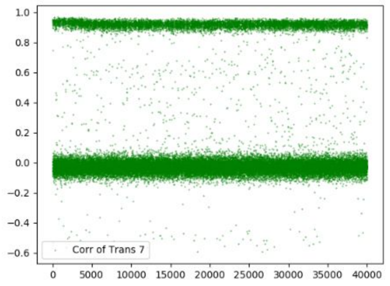
Trans ID 4



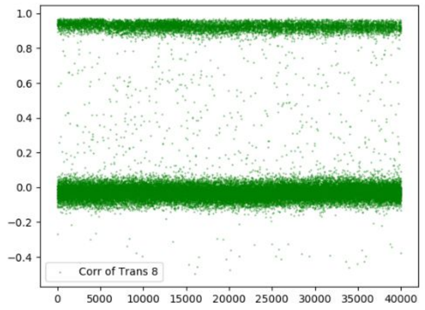
Trans ID 5



Trans ID 6



Trans ID 7



Trans ID 8

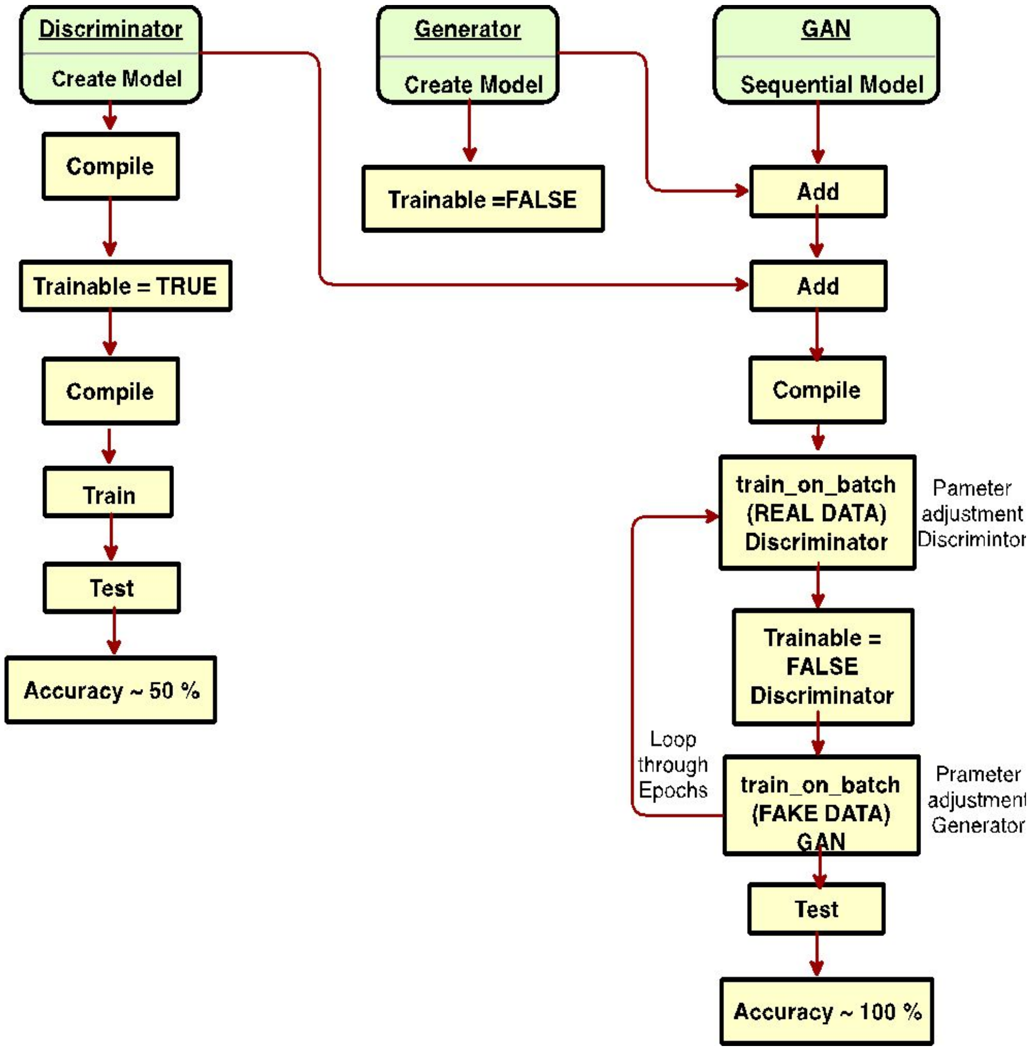
- **75%** of samples' correlation coefficients fall between **-0.1** and **0.1**, and **25%** close to **0.9**.
- For transmitter **ID 3**, all samples' correlation coefficients fall between **-0.1** and **0.1**.

 Poor Spatial Correlation

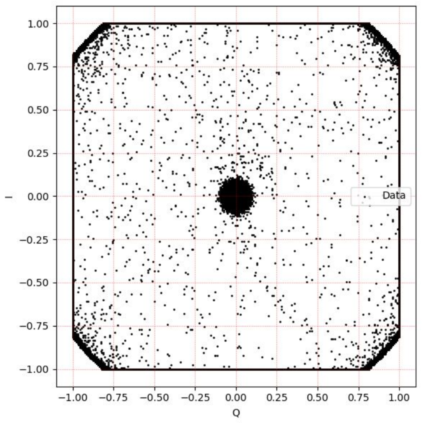
Implementation

- Our objective was to design:
  - a generative adversarial net **(GAN)** to **distinguish** rogue transmitters from trusted ones.
  - a convolutional neural network **(CNN)** to exploit the **correlation** in collected signal data of the trusted transmitters.
  - a deep neural network **(DNN)** to **classify** the trusted transmitters for fingerprinting.
  - a recurrent neural networks **(RNNs)** to improve classification accuracy exploiting the property of **time-series** data.

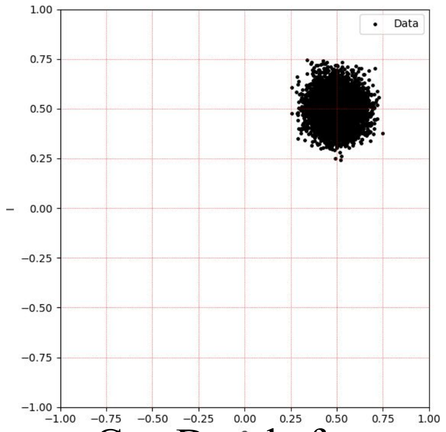
Experimental Analysis: GAN



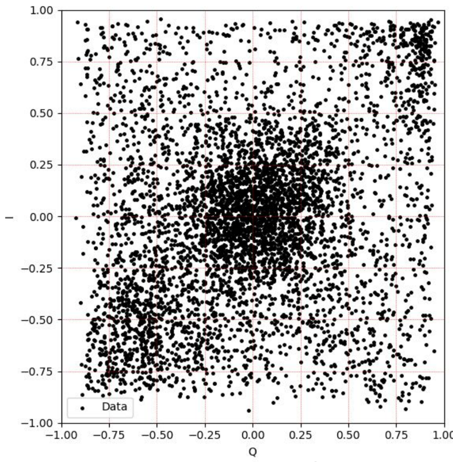
Experimental Results: Generator



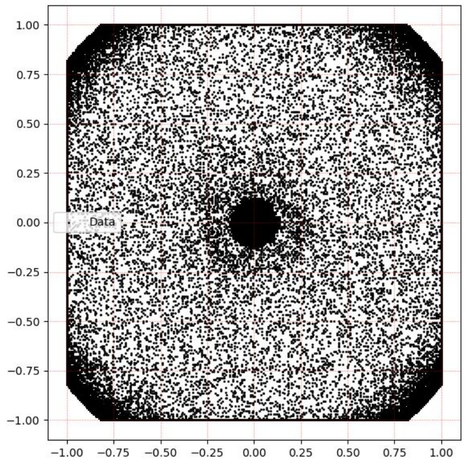
Real Data (128 samples)



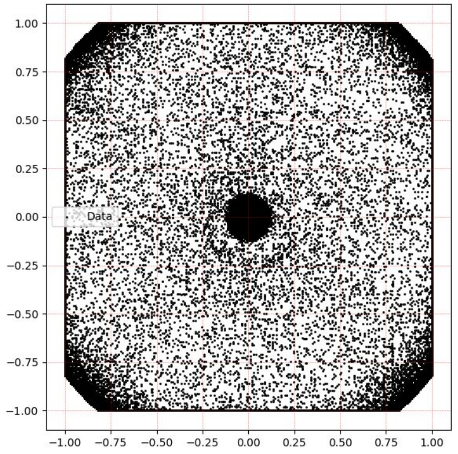
Gen Data before
GAN Training (128 samples)



Gen Data after
GAN Training (128 samples)



Real Data (2000 samples)

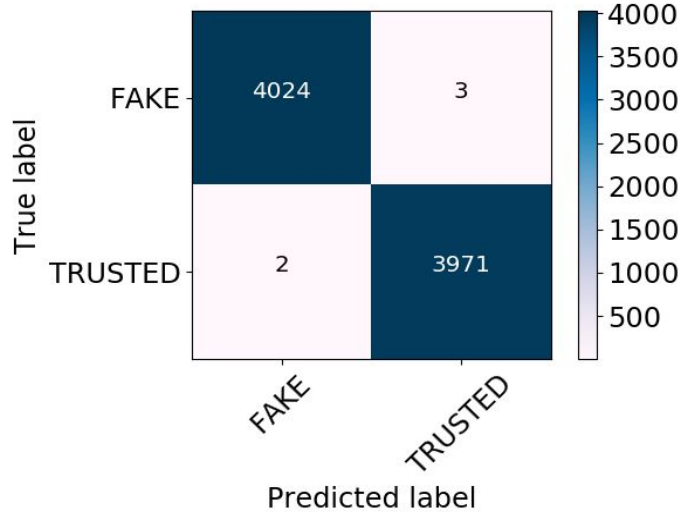


Gen Data after
GAN Training (2000 samples)

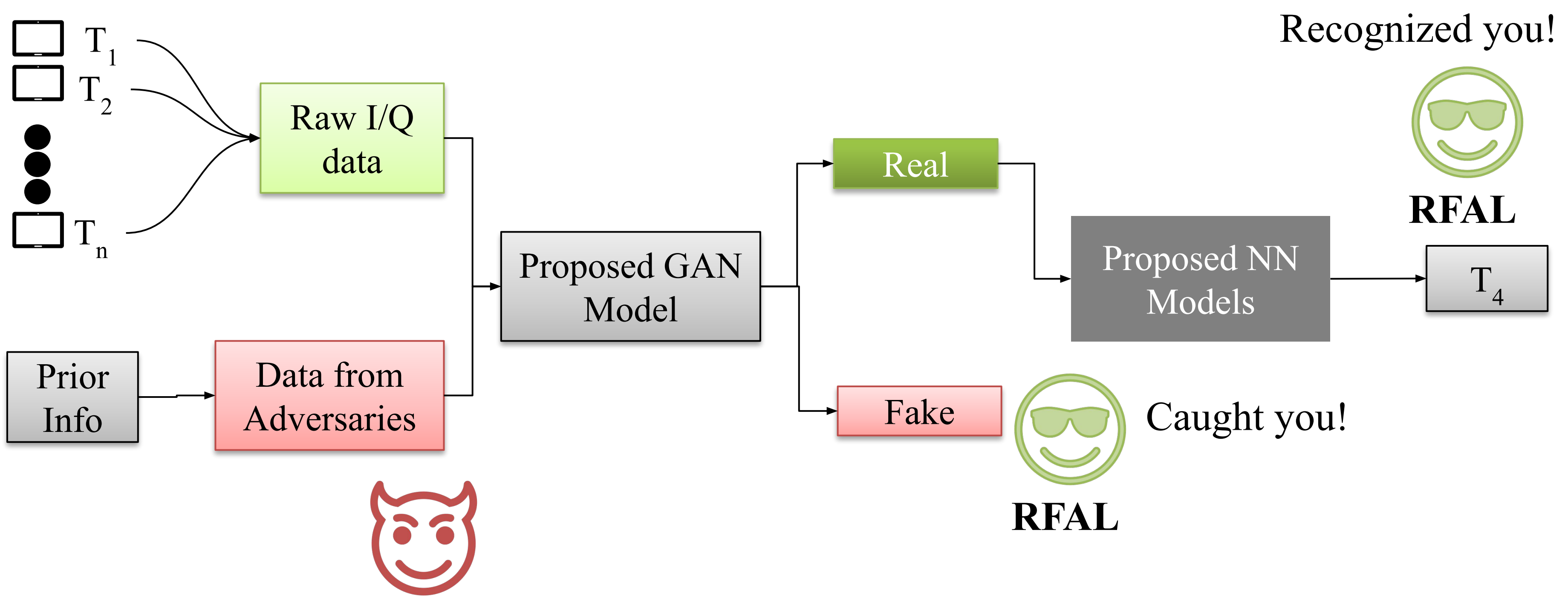
Experimental Results: Identification

Dataset (GB)	#Trans	Method	#Parameters	Accuracy (%)
			3.6 M (G)	
6.8	4	GAN (DNN)	6.8 M (D)	99.9
			10.4 M (GAN)	
			3.6 M (G)	
13.45	8	GAN (DNN)	6.8 M (D)	99.9
			10.4 M (GAN)	

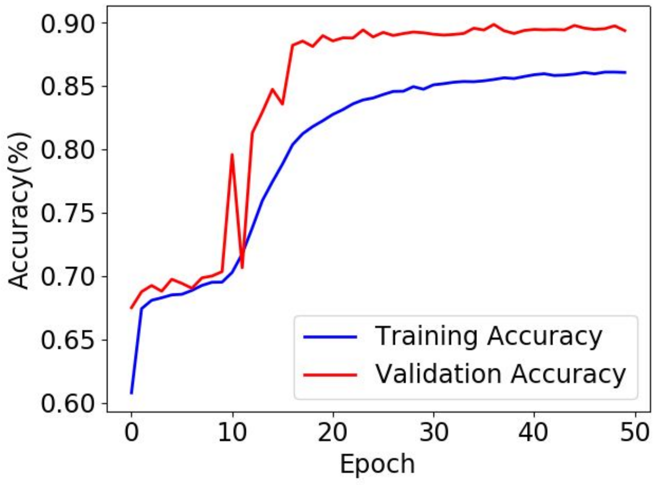
Confusion Matrix for Identification:

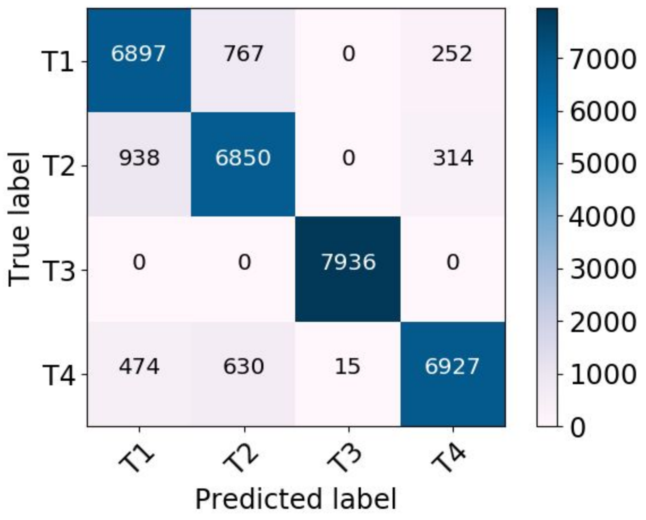


RFAL Architecture



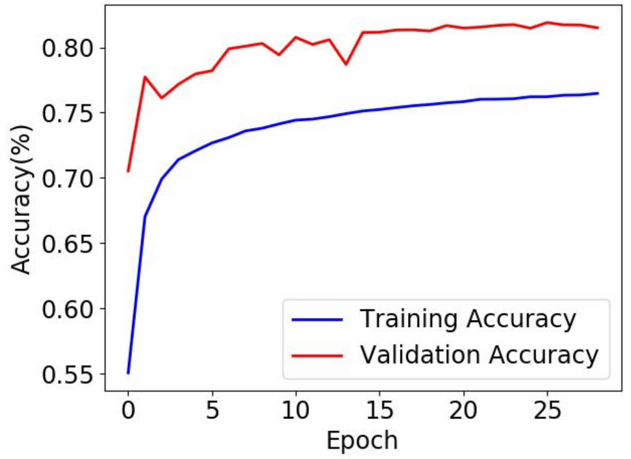
Experimental Results: CNN

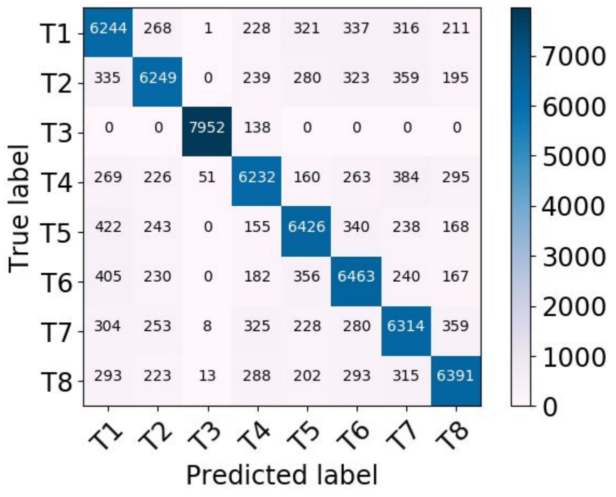




	T1	T2	T3	T4
T1	6897	767	0	252
T2	938	6850	0	314
T3	0	0	7936	0
T4	474	630	15	6927

4 Transmitters

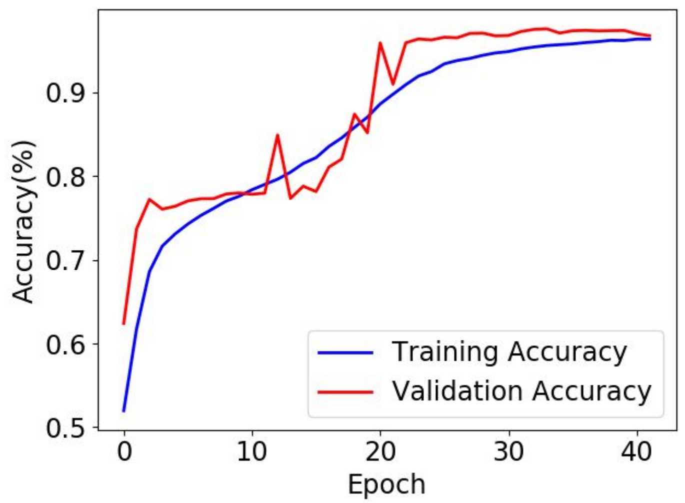


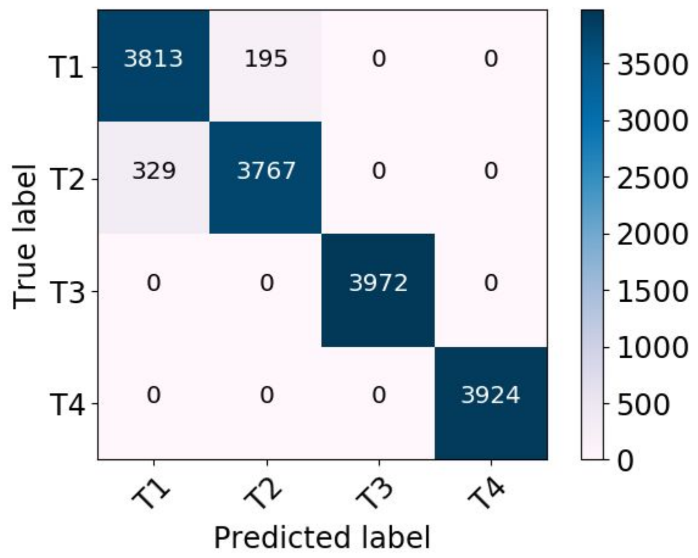


	T1	T2	T3	T4	T5	T6	T7	T8
T1	6244	268	1	228	321	337	316	211
T2	335	6249	0	239	280	323	359	195
T3	0	0	7952	138	0	0	0	0
T4	269	226	51	6232	160	263	384	295
T5	422	243	0	155	6426	340	238	168
T6	405	230	0	182	356	6463	240	167
T7	304	253	8	325	228	280	6314	359
T8	293	223	13	288	202	293	315	6391

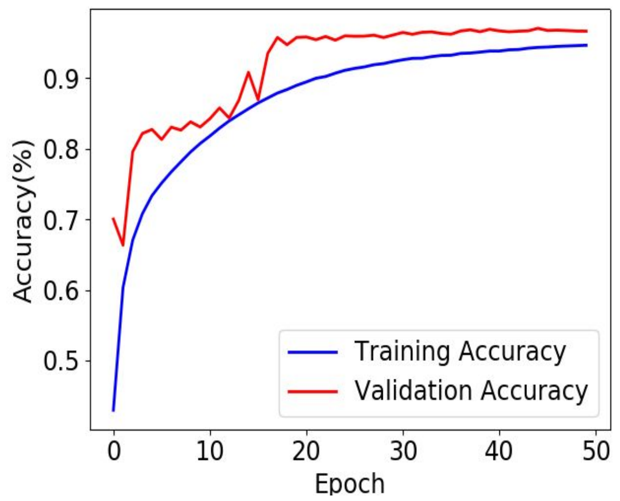
8 Transmitters

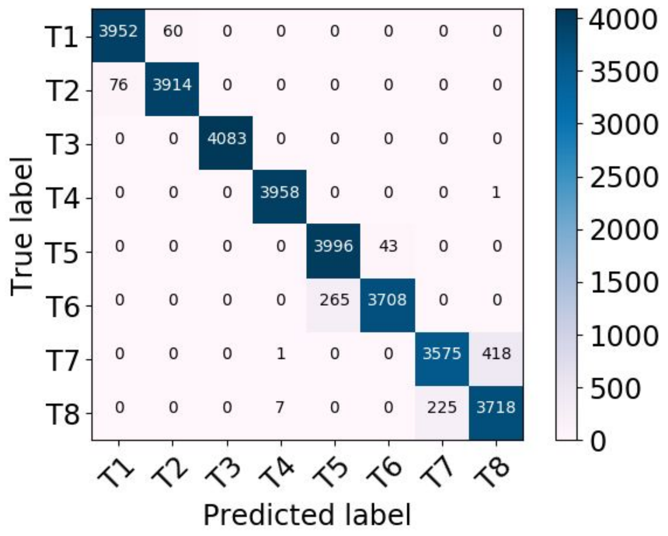
Experimental Results: DNN





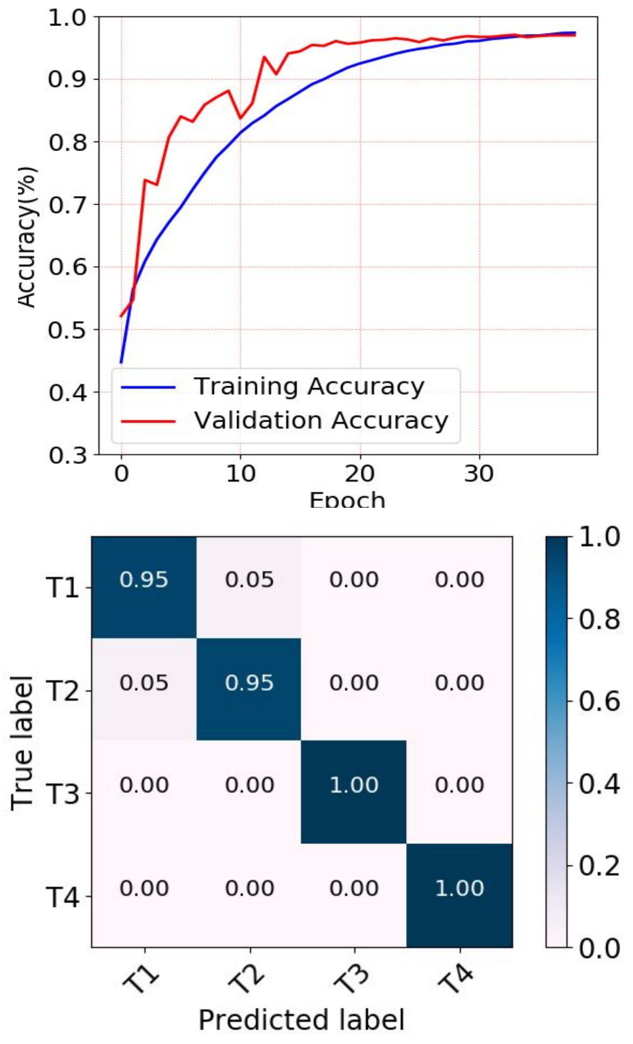
4 Transmitters



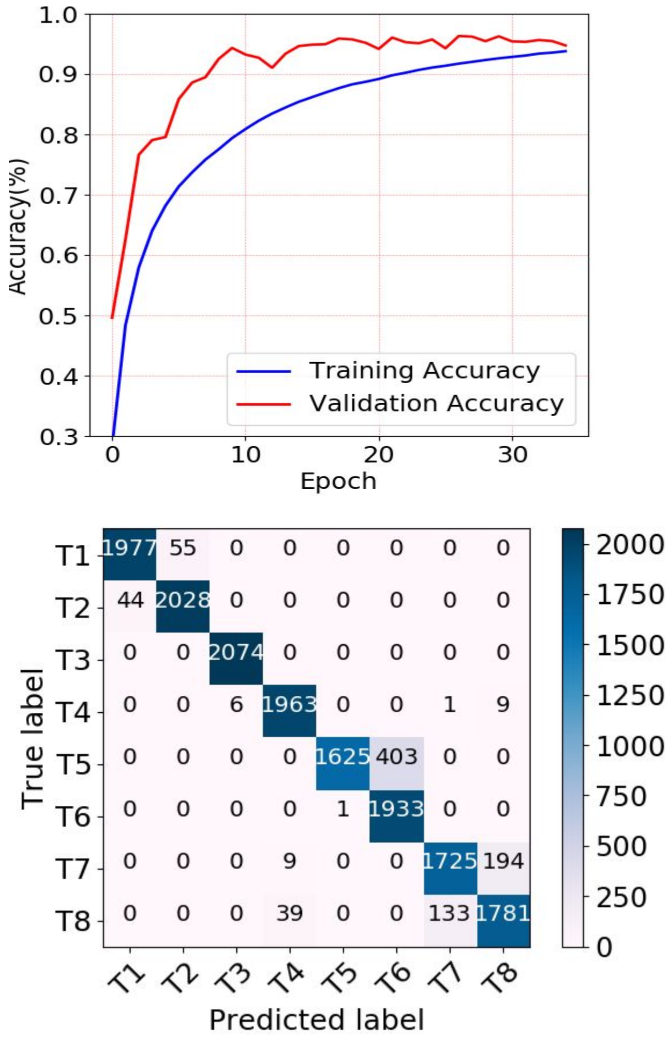


8 Transmitters

Experimental Results: RNN (LSTM)

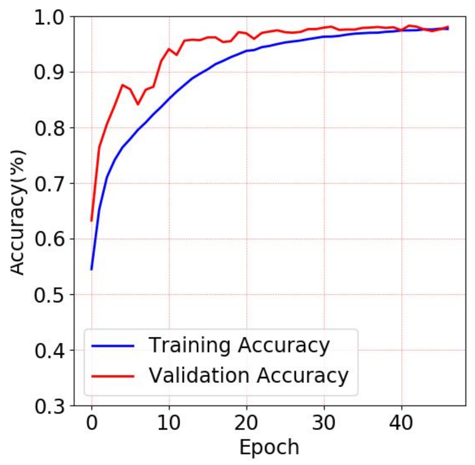


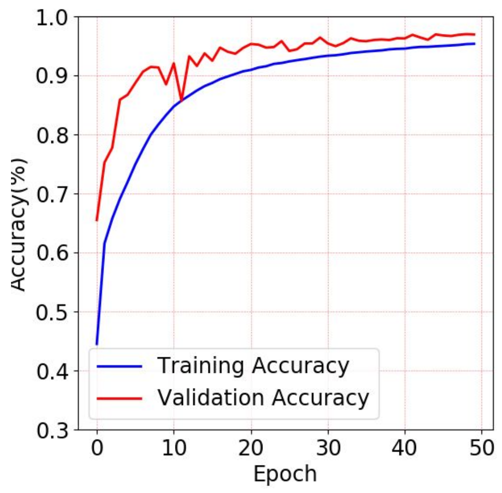
4 Transmitters

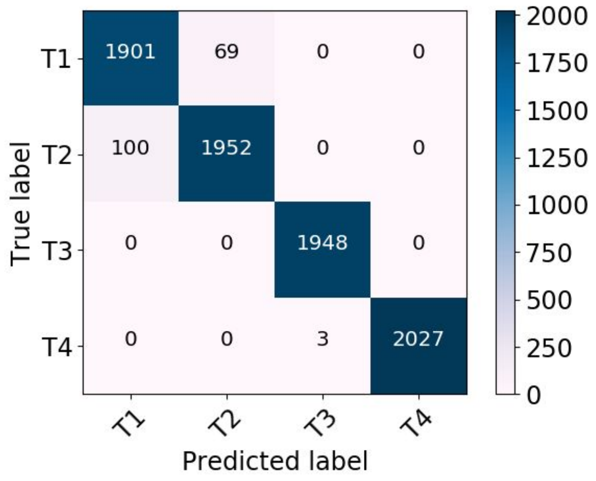


8 Transmitters

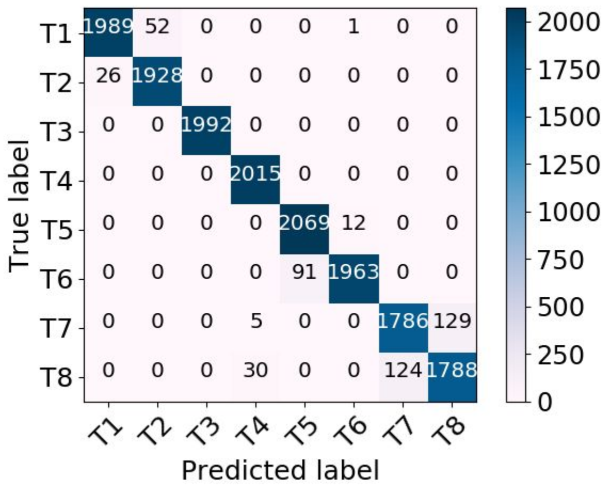
Experimental Results: RNN (GRU)







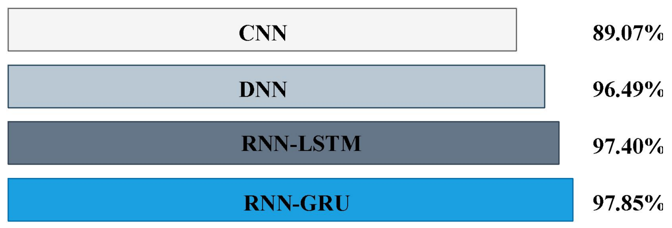
4 Transmitters



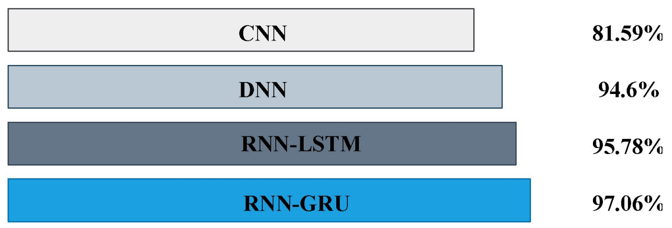
8 Transmitters

Experimental Results: Classification

Dataset (GB)	#Trans	Method	#Parameters	Accuracy (%)
6.8	4	DNN (4 layers)	6.8 M	97.21
13.45	8	DNN (4 layers)	6.8 M	96.60
6.8	4	CNN (6 layers)	38 M	89.07
13.45	8	CNN (6 layers)	38 M	81.59
6.8	4	RNN-LSTM (6 layers)	14.2 M	97.40
13.45	8	RNN-LSTM (6 layers)	14.2 M	95.78
6.8	4	RNN-GRU (6 layers)	10.7 M	97.85
13.45	8	RNN-GRU (6 layers)	10.7 M	97.06



4 Transmitters

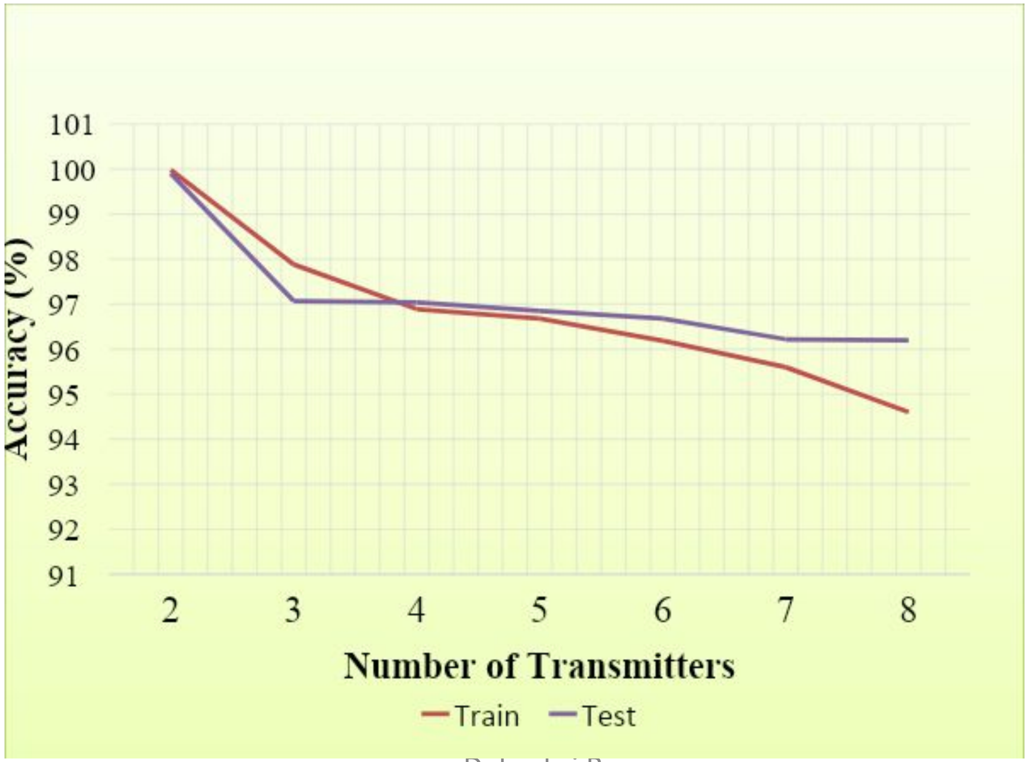


8 Transmitters

Experimental Results: Heterogeneous Data

Models	USRP-USRP	PLUTO-USRP
CNN	89.91%	99.91%
DNN	99.9%	100%
RNN	99.95%	100%

Training and Testing Accuracies with **increasing** number of **Transmitters**



Experimental Results: Varying SNR

SNR (dB)	Accuracy (%)		
	CNN	DNN	RNN (GRU)
0	51.53	85.12	92.3
10	78.64	92.24	95.64
20	81.3	94.60	97.02
30	81.59	94.60	97.06

Accuracies for Different Neural Network Models with **Varying SNR**

- **Better** accuracy for **all** models with **higher SNR**.
- RNN (**GRU**) gives **92%** accuracy at **0 dB** SNR too.

Performance Analysis

Approach	#Trans	SNR (dB)	Acc (%)	Inputs
Genetic Algorithm [1]	5	25	**85-98**	Transients
Multifractal Segmentation [2]	8	Not Mentioned	**92.5**	Transients
Orthogonal Component Reconstruction (OCR) [3]	3	20	**62-71**	Spurious Modulation
K-NN [4]	8	30	**97**	Transients
RNN [5]	-	20	**90**	Synthetic Dataset
RFAL (RNN)	8	30	**97.04**	Raw I/Q Data

Comparison of **RFAL** Implementation with the **Traditional** Ones

- **Existing** methods used some **extracted features** as **input**.
- **Some** works are tested on **synthetic datasets** only.

[1] J. Toonstra *et al.*, "A radio transmitter fingerprinting system ODO-1", CCECE, 1996.
[2] D. Shaw *et al.*, "Multifractal Modelling of Radio Transmitter Transients for Classification," IEEE WESCANEX, 1997.
[3] S. Xu *et al.*, "Individual Radio Transmitter Identification based on Spurious Modulation Characteristics of Signal Envelop," IEEE MILCOM, 2008.
[4] I. Kennedy *et al.* "Radio Transmitter Fingerprinting: A Steady State Frequency Domain Approach", IEEE VTC, 2008.
[5] S. Rajendran *et al.*, "Deep Learning Models for Wireless Signal Classification With Distributed Low-Cost Spectrum Sensors, IEEE TCCN, 2018.

Performance Analysis

Approach	#Trans	SNR (dB)	Acc (%)	Inputs
CNN [6]	7	30	**91.38**	Preprocessed data from MATLAB
CNN [7]	5	50	**98**	Preprocessed data from MATLAB
CNN [8]	-	-	**99.67**	ACARS data
DNN [9]	12	-	**84.4**	Raw Signal
Inception ResNet [10]	-	-	**98.1 & 96.3**	ACARS & ADS-B
CNN [11]	16	30	**99.5**	Demodulated symbols
CNN [12]	21	-	**99.99**	FIT/CorteXlab
RFAL (RNN)	8	30	**97.04**	Raw I/Q Data

Comparison of **RFAL** Implementation with the **State-of-the-art**

- **Existing** methods used some **processed data** as **input**.
- **Some** works are tested on **existing datasets** only.

[6] K. Merchant *et al.* "Deep learning for RF device fingerprinting in cognitive communication networks", IEEE JSTSP, 2018.
[7] S. Riyaz *et al.* "Deep Learning Convolutional Neural Networks for Radio Identification", IEEE Com. Mag., 2018.
[8] S. Zheng *et al.* "Big Data Processing Architecture for Radio Signals Empowered by Deep Learning: Concept, Experiment, Applications and Challenges", IEEE Access, 2018.
[9] K. Youssef *et al.* "Machine Learning Approach to RF Transmitter Identification", IEEE RFID, 2018.
[10] S. Chen *et al.* "Deep learning for large-scale real-world ACARS and ADS-B radio signal classification", CoRR, 2019.
[11] K. Sankhe *et al.* "ORACLE: Optimized Radio clAssification through Convolutional neuraL nEtworks", IEEE INFOCOM, 2019.
[12] C. Morin *et al.* "Transmitter Classification With Supervised Deep Learning", CoRR, 2019.

Experimental Setup: Configurations

Models	#Layers	Learning Rate	Batch Size	Epochs	Optimizers
CNN	7	10^{-4}	128	45-50	Adam
DNN	5	10^{-3}	128	35-40	Adam
RNN-LSTM	6	10^{-3}	128	30-35	SGD
RNN-GRU	6	10^{-3}	128	30-35	SGD

Comparison for **Configuration** Settings for Different **Models**

- Different **models** have different **hyper-parameter** values.
- Maximum **50** epochs with **early stopping** of patience **5**.

Next: A fingerprinting demo of the small dataset (2 radios) in jupyter notebook

Outline

Part1: Neel (90 mins)

- Introduction to signal processing concepts for SDR, USRP radio hardware architecture, UHD device driver and UHD API, and GNU Radio
- Configuration of the USRP Radio
- Demos and examples of various SDR systems

Part2: Tathagata (40 mins)

- Discussion of FM radio-based positioning using SDR
- Introduction to ML Concepts
- Introduction to Adversarial Learning

Part3: Debashri (35 mins)

- RF ML Problems and Challenges
- Introduction to the Transmitter Identification problem
- Example of Transmitter Identification using Python Jupyter notebook

Q&A: 15 mins