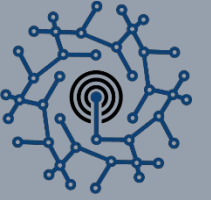




DEEPWAVE DIGITAL



making sense of signals

John D. Ferguson, Ph.D.

President / CEO

john@deepwavedigital.com

October 24, 2018



Deep Learning and Radio Frequency (RF) Systems

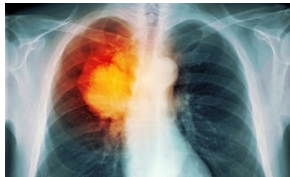
Deep Learning is Emerging

Cyber



- Intrusion Detection
- Threat classification
- Facial recognition
- Imagery analysis

Medicine



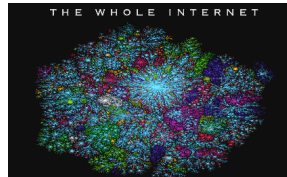
- Tumor Detection
- Medical data analysis
- Diagnosis
- Drug discovery

Autonomy



- Pedestrian / obstacle detection
- Navigation
- Street sign reading
- Speech recognition

Internet

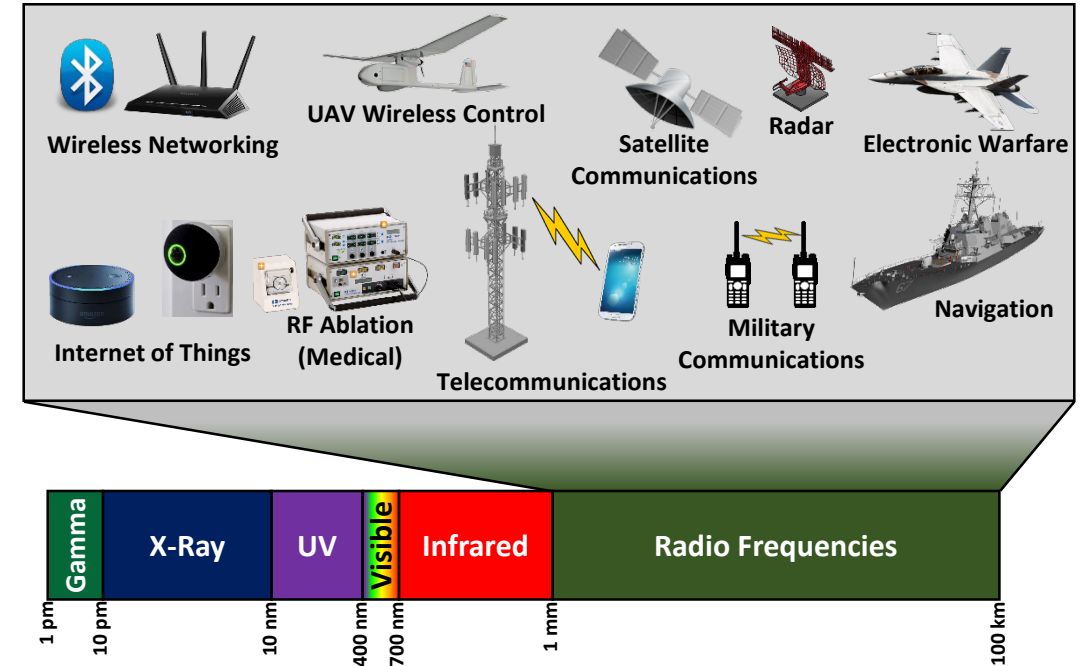


- Image classification
- Speech recognition
- Language translation
- Document / database searching



Enabled by low-cost, highly capable general purpose graphics processing units (GPUs)

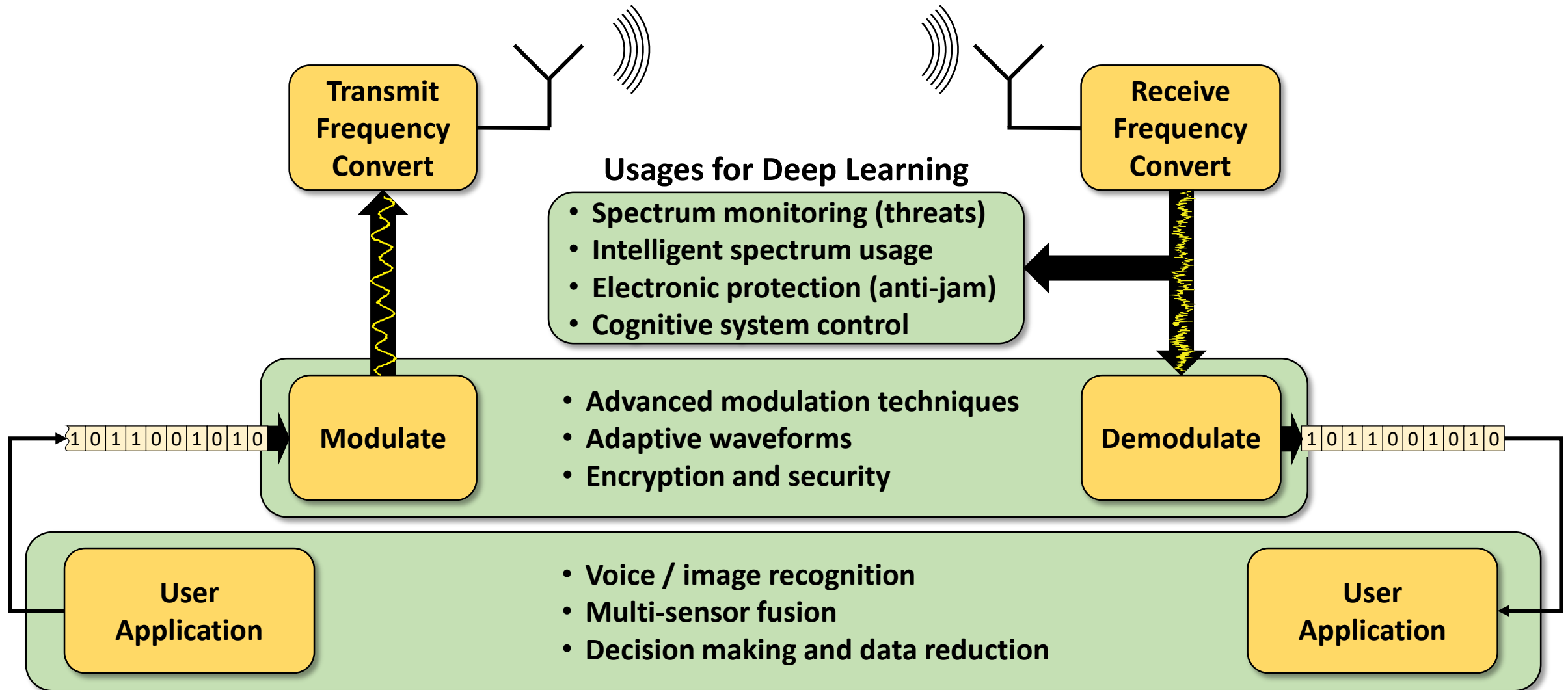
Radio Frequency Technology is Pervasive



Deep learning technology enabled and accelerated by GPU processors

- Has yet to impact design and applications in wireless and radio frequency systems

Intelligent Radio Frequency (IRF) Systems



Why Has It Not Been Addressed

Bandwidth Limitations

remote processing not possible

- AI requires large data sets
- Insufficient bandwidth to send to remote data center

Limited Compute Resources

at field site

- No RF systems exist with integrated AI computational processors

Complicated Software

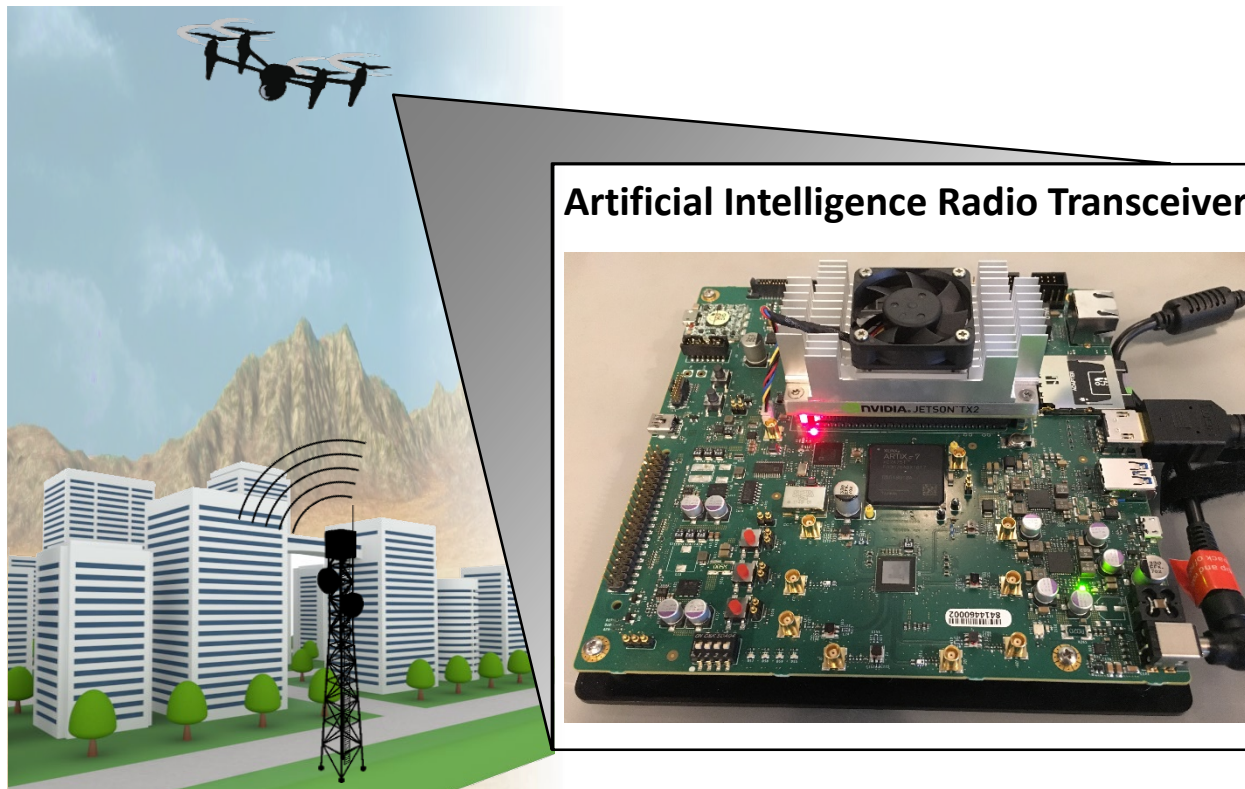
for RF and AI independently

- Disjointed software
- Difficult to program and understand

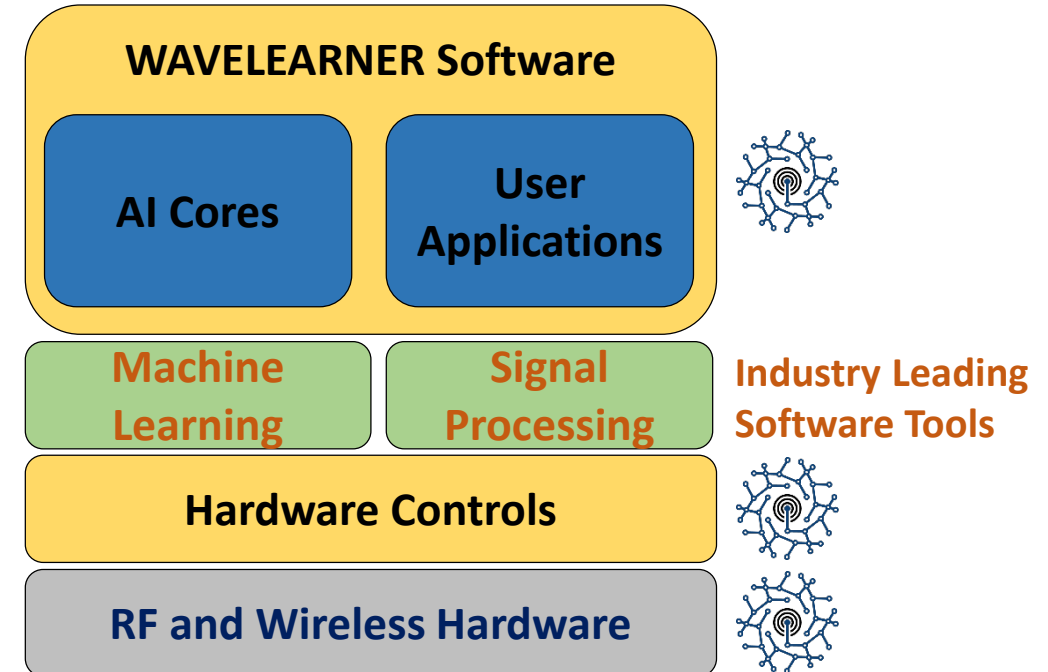
Our Solution and Platform

Approach - Enable the wide adoption of AI within wireless technology with our integrated hardware and software platform


Hardware for Real-world Applications



Easy to Program Software

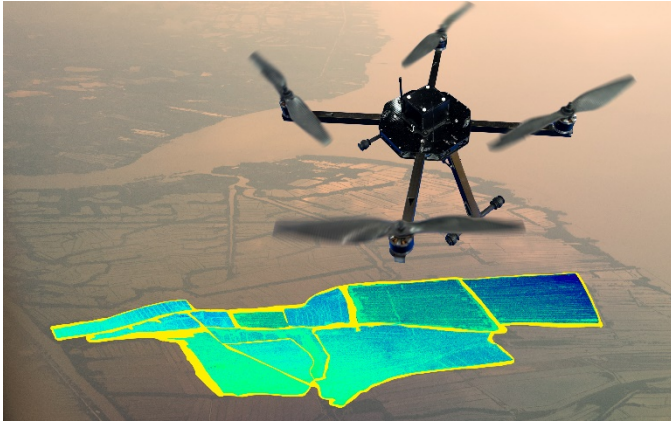


Outline

- Introduction
-  • Deep Learning in RF Systems
- Deepwave Digital Technology
- Example Signal Detection and Classification
- Real-time Signal Processing Benchmarks for GPUs
- Summary

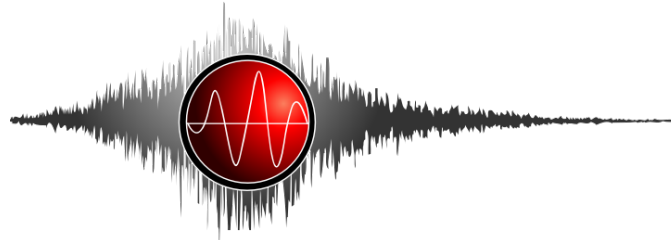
Deep Learning Comparison

Image and Video



- Multiple channels (RGB)
- x, y spatial dependence
- Temporal dependence (video)

Audio and Language



- Single channel
- Frequency, phase, amplitude
- Temporal dependence

Systems and Signals



- Multiple channels
- Frequency, phase, amplitude
- Temporal dependence
- Complex data (I/Q)
- Large Bandwidths
- Human engineered

Existing deep learning potentially adaptable to systems and signals

- Must contend with wideband signals and complex data types

Hardware for Deep Learning in RF Systems

	Training		Inference	
	Pros	Cons	Pros	Cons
CPU	<ul style="list-style-type: none"> • Supported by ML Frameworks • Lower power consumption 	<ul style="list-style-type: none"> • Slower than GPU • Fewer software architectures 	<ul style="list-style-type: none"> • Adaptable architecture • Software programmable • Medium latency 	<ul style="list-style-type: none"> • Low parallelism • Limited real-time bandwidth • Medium power requirements
GPU	<ul style="list-style-type: none"> • Supported by ML Frameworks • Widely utilized • Highly parallel / adaptable • Good throughput vs power 	<ul style="list-style-type: none"> • Overall power consumption • Requires highly parallel algorithms 	<ul style="list-style-type: none"> • Adaptable architecture • High real-time bandwidth • Software programmable 	<ul style="list-style-type: none"> • Medium power requirements • Not well integrated into RF • Higher latency
FPGA	Not widely utilized, not well suited (yet)		<ul style="list-style-type: none"> • High power efficiency • High real-time bandwidth • Low latency 	<ul style="list-style-type: none"> • Long development / upgrades • Limited reprogrammability • Requires special expertise
ASIC	Not widely utilized, not well suited		<ul style="list-style-type: none"> • Extremely power efficient • High real-time bandwidth • Highly reliable • Low latency 	<ul style="list-style-type: none"> • Extremely expensive • Long development time • No reprogrammability • Requires special expertise


Critical Performance Parameters for Deep Learning in RF Systems

	Adaptability / Upgradability	Deployment Time	Lifecycle Cost	Real Time Bandwidth	Compute / Watt	Latency
CPU	Green	Green	Green	Red	Yellow (diagonal split)	Yellow
GPU	Green	Green	Green	Green	Yellow	Yellow (diagonal split)
FPGA	Yellow	Yellow (diagonal split)	Red	Green	Green	Green
ASIC	Red	Red	Green	Green	Green	Green

GPU signal processing can provide wideband capability and software upgradability at lower cost and development time

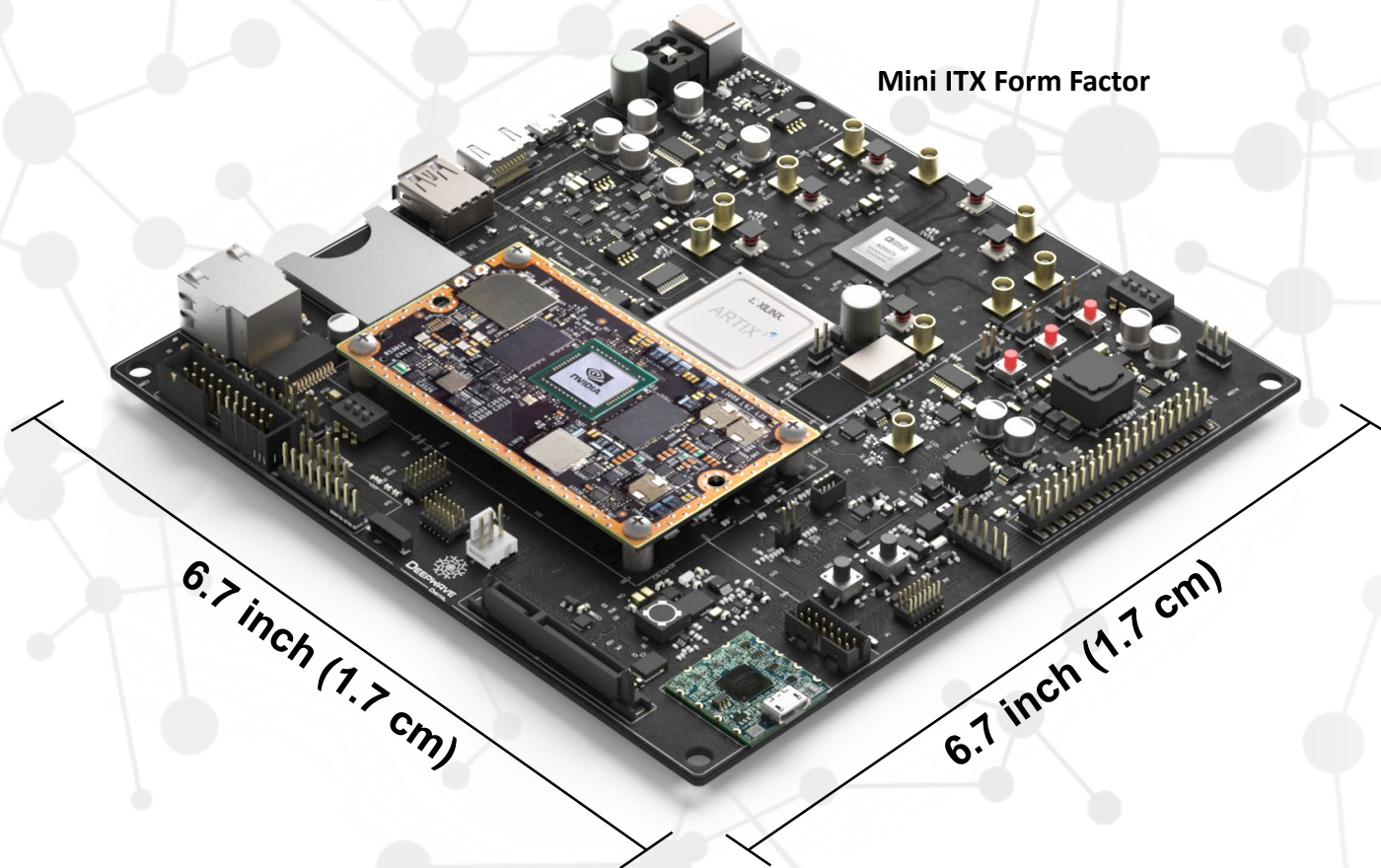
- Must contend with increased latency (~2 microsecond)

Outline

- Introduction
- Deep Learning in RF Systems
-  • Deepwave Digital Technology
- Example Signal Detection and Classification
- Real-time Signal Processing Benchmarks for GPUs
- Summary

Artificial Intelligence Radio Transceiver (AIR-T)

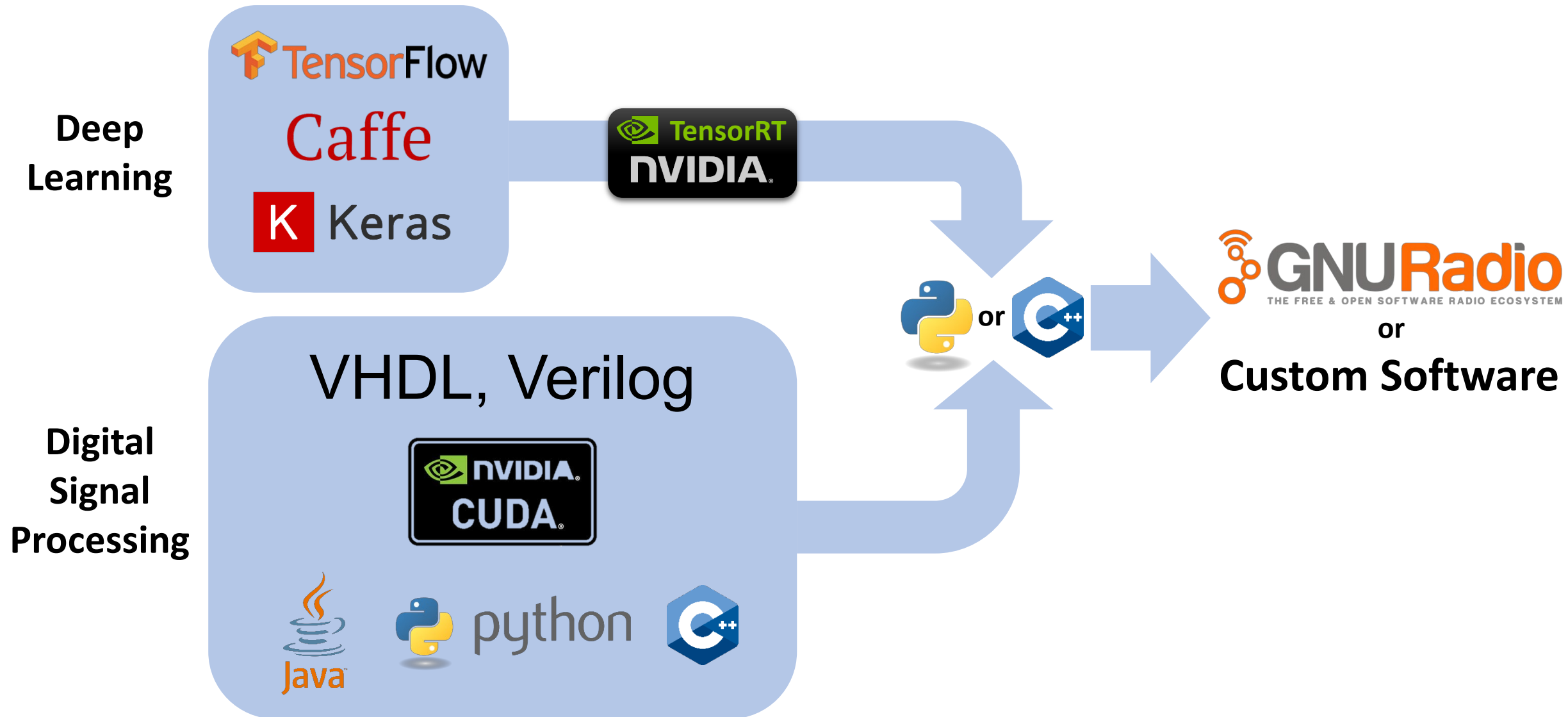
AIR-T



Hardware Specifications

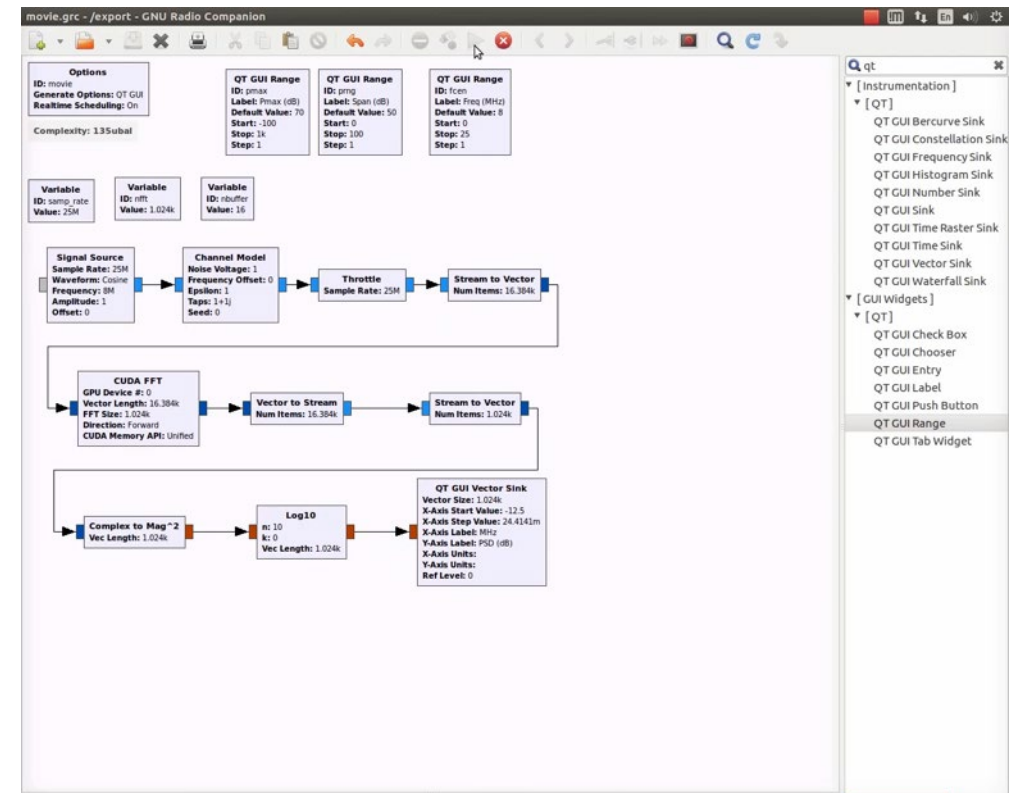
- **2x2 MIMO Transceiver**
 - Analog Devices 9371 chip
 - Tunable from 300 MHz to 6 GHz
 - 100 MHz bandwidth per channel
- **Digital Signal / Deep Learning Processors**
 - Xilinx Artix 7 FPGA
 - NVIDIA Jetson TX2
 - ARM Cortex-A57 (quad-core)
 - Denver2 (dual core)
 - Nvidia Pascal 256 Core GPU
 - Shared GPU/CPU memory
- **Connectivity**
 - 1 PPS / 10 MHz for GPS Synchronization
 - External LO input
 - HDMI, USB 2.0/3.0, SATA, Ethernet, SD Card, GPIO

Simplified Programming



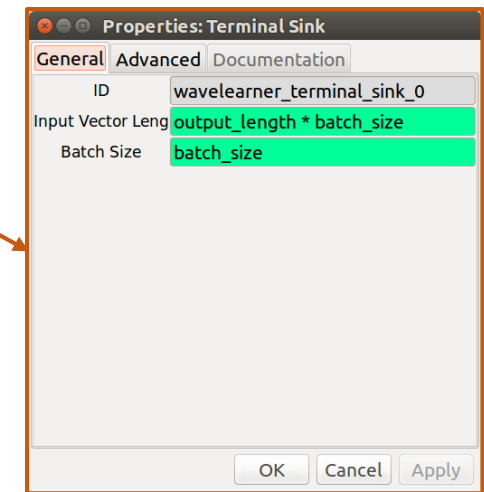
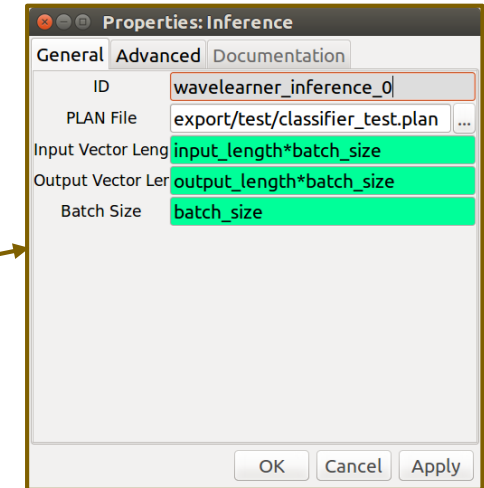
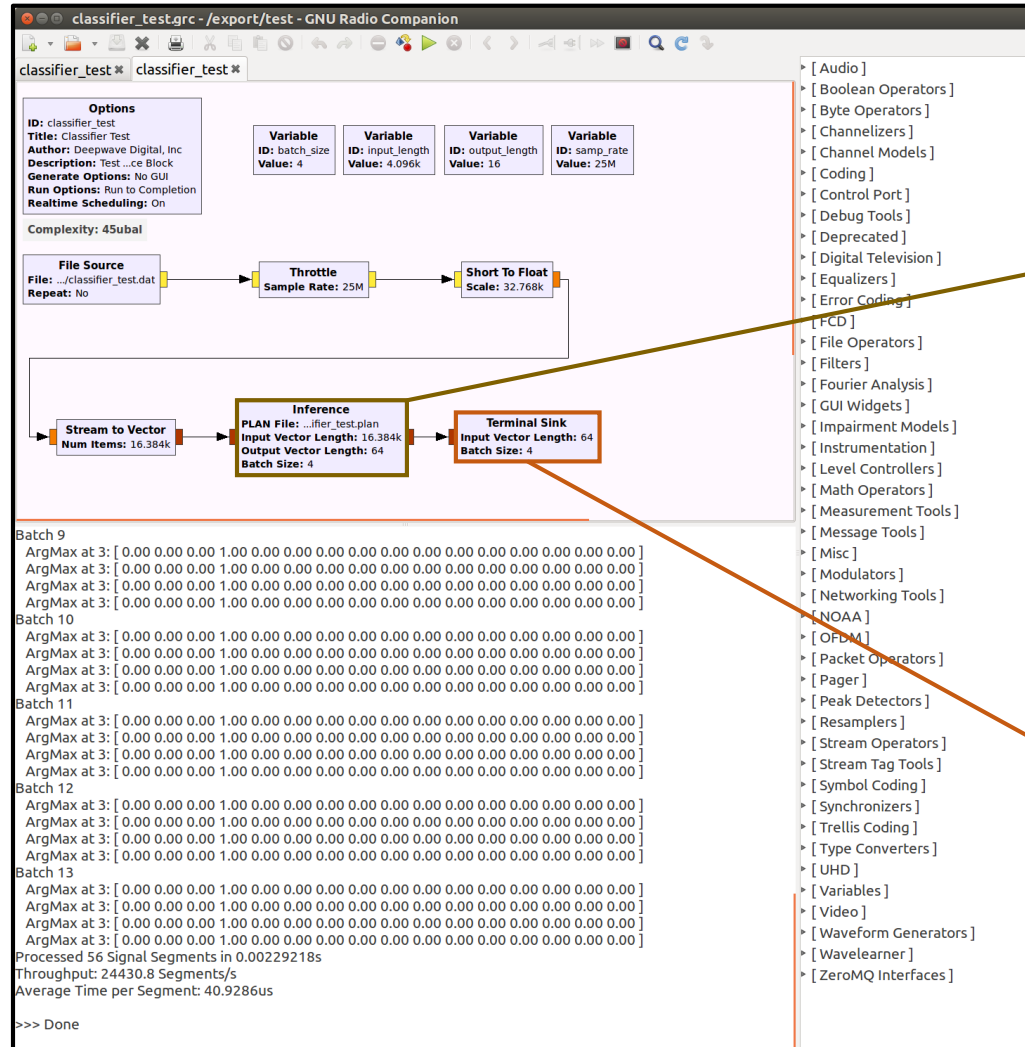
GNU Radio – Software Defined Radio (SDR) Framework

- **Popular open source software defined radio (SDR) toolkit:**
 - RF Hardware optional
 - Can run full software simulations
- **Python API**
 - C++ under the hood
- **Easily create DSP algorithms**
 - Custom user blocks
- **Primarily uses CPU**
 - Advanced parallel instructions
 - Recent development: RFNoC for FPGA processing
- **Deepwave is integrating GPU support for both DSP and ML**

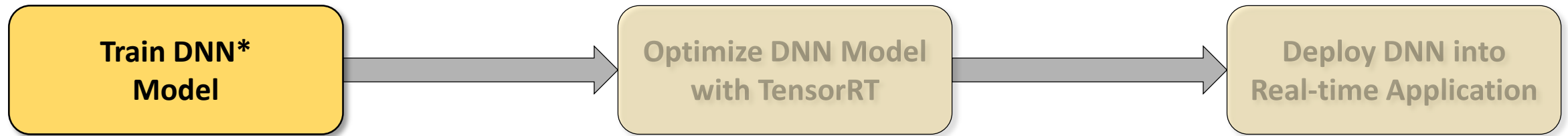


GR-Wavelearner

- Out of tree (OOT) module for GNU Radio
- Allows users to easily incorporate deep learning into signal processing
- C++ and Python API
- Open source GPLv3 license
- Two blocks currently:
 - Inference – TensorRT wrapper for GNU Radio
 - Terminal Sink – Python module for displaying classifier output



Deepwave's Training to Deployment Workflow



- Workflow utilizes TensorRT for deployment
- Allows for training on wide array of frameworks

Native support for TensorRT



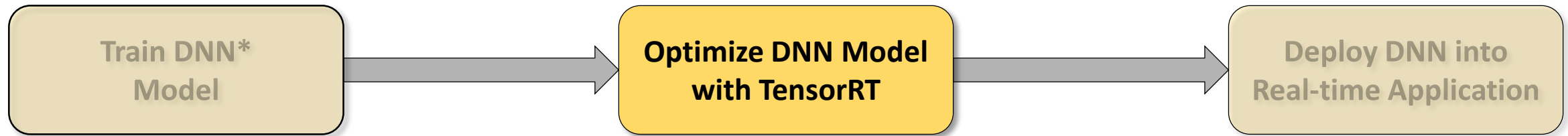
Support Via ONNX**



* Deep Neural Network

**Open Neural Network Exchange

Deepwave's Training to Deployment Workflow



TensorFlow Example

Step 1: Freeze graph (make variables constants)

Step 2: Convert DNN model to UFF File

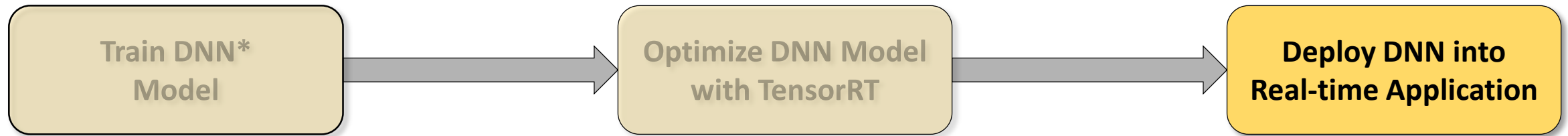
Step 3: Convert UFF File to PLAN File

Note: This step must be completed on deployment architecture

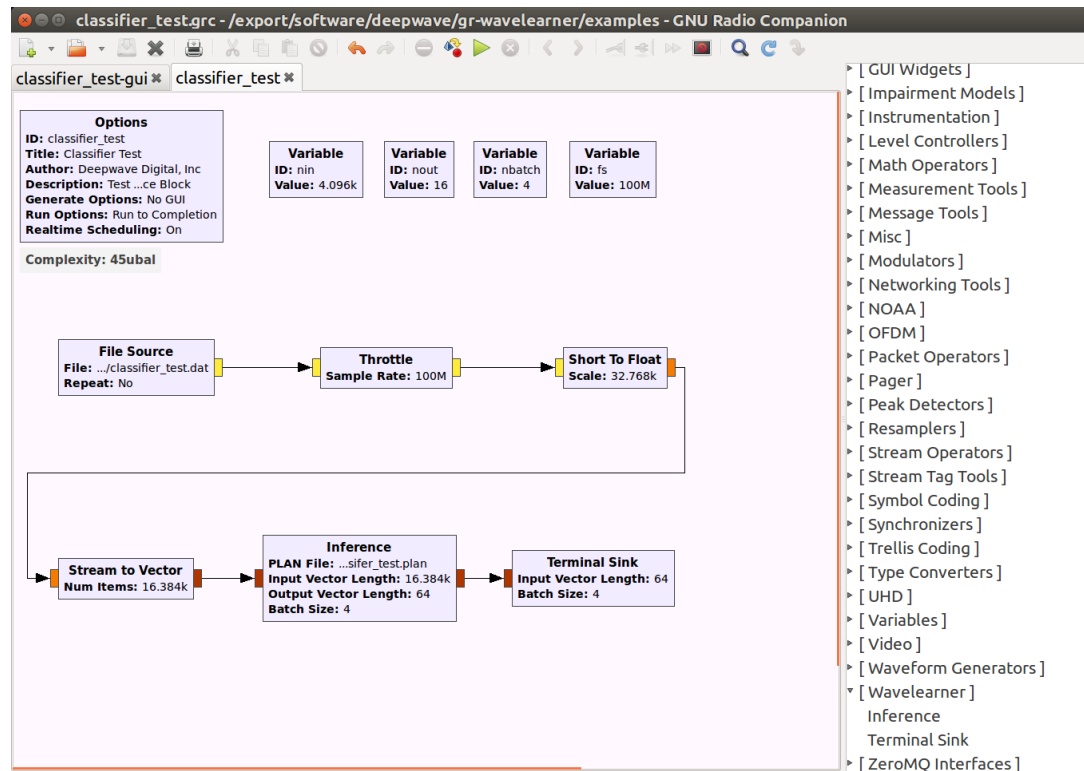
Caveats

- Not all layers are supported, but most common ones are
- PLAN file must be created on deployment architecture
 - Python conversion not available on ARM (Jetson)
 - Limited transferability of PLAN files

Deepwave's Training to Deployment Workflow



GNU Radio Companion GUI



Python API

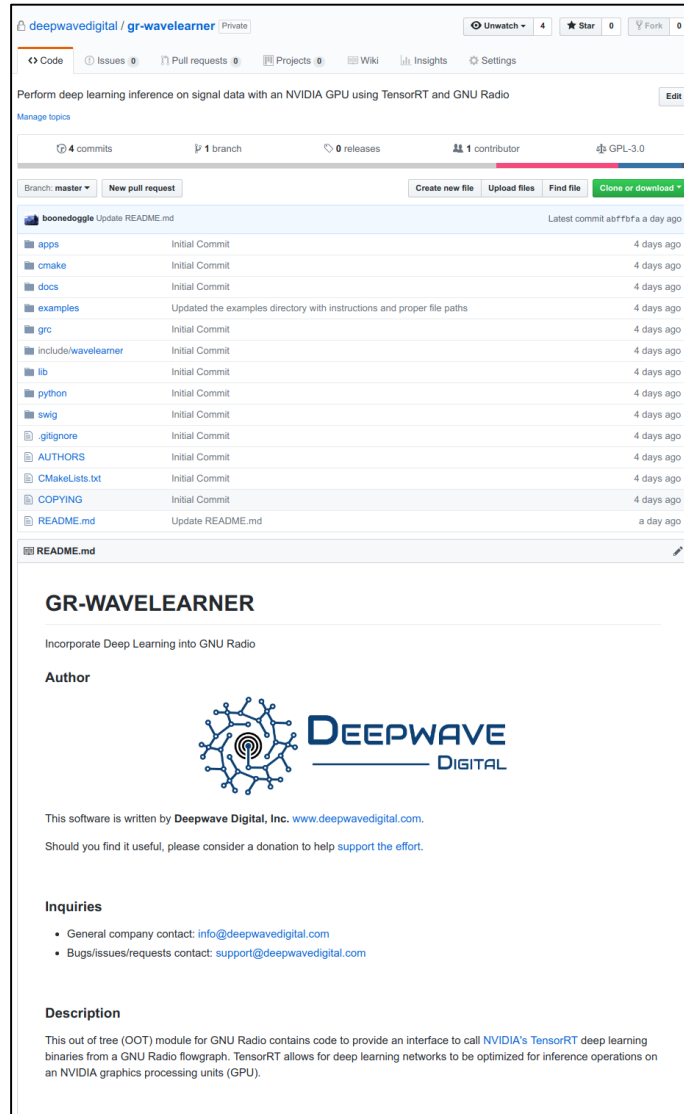
(Real time DNN* RF system in 35 lines of code!)

```

1 #!/usr/bin/env python2
2 from gnuradio import blocks
3 from gnuradio import gr
4 import wavelearner
5
6 class ClassifierTest(gr.top_block):
7     def __init__(self):
8         gr.top_block.__init__(self)
9
10        # Define signal processing functions
11        self.source = blocks.file_source(gr.sizeof_short, inputdata, False)
12        self.throttle = blocks.throttle(gr.sizeof_short, fs, True)
13        self.type_convert = blocks.short_to_float(1, norm)
14        self.buffer = blocks.stream_to_vector(gr.sizeof_float, nbatch*nin)
15        self.inference = wavelearner.inference(planfile, nin*nbatch, nout*nbatch, nbatch)
16        self.terminal_out = wavelearner.terminal_sink(nout * nbatch, nbatch)
17
18        # Connect functions in flowgraph
19        self.connect((self.source, 0), (self.throttle, 0))
20        self.connect((self.throttle, 0), (self.type_convert, 0))
21        self.connect((self.type_convert, 0), (self.buffer, 0))
22        self.connect((self.buffer, 0), (self.inference, 0))
23        self.connect((self.inference, 0), (self.terminal_out, 0))
24
25        # Define Settings
26        inputdata = '/path/to/classifier_test.dat'
27        planfile = '/path/to/classifier_test.plan'
28        fs = 100e6
29        nin = 4096
30        nout = 16
31        nbatch = 4
32        norm = 32768
33        tb = ClassifierTest()
34        tb.start()
35        tb.wait()
36
  
```

* Deep Neural Network

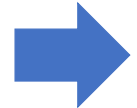
GR-Wavelearner Available Now on GitHub



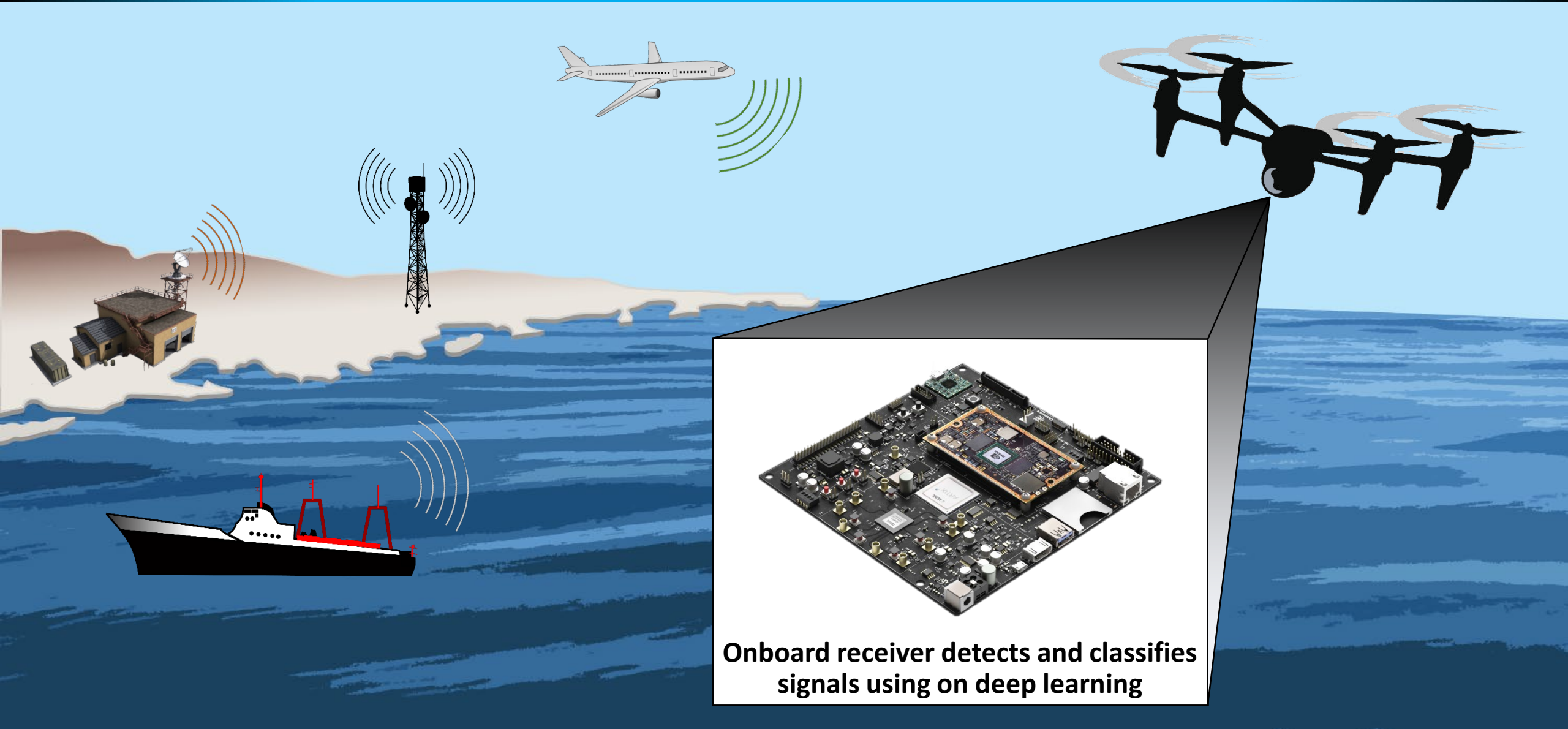
- Goal is to help the open source community easily deploy deep learning within signal processing applications
- Looking for the community to use module and provide feedback
- Well documented README with dependency installation instructions to get started quickly
 - Ubuntu 16.04 recommended, Windows 10 supported
 - Tested with NVIDIA Docker Container 18.08*
- Signal classifier example provided:
 - GNU Radio Flowgraph
 - Python source code
 - PLAN files that are executable on the AIR-T and GTX 950M
 - Signal data file example for testing
- Just added support for TensorRT 5.0

https://docs.nvidia.com/deeplearning/sdk/tensorrt-container-release-notes/rel_18.08.html

Outline

- Introduction
- Deep Learning in RF Systems
- Deepwave Digital Technology
-  • Example Signal Detection and Classification
- Real-time Signal Processing Benchmarks for GPUs
- Summary

Multi-transmitter Environmental Scenario



Radar Signal Detector Model: Transmitted Signals

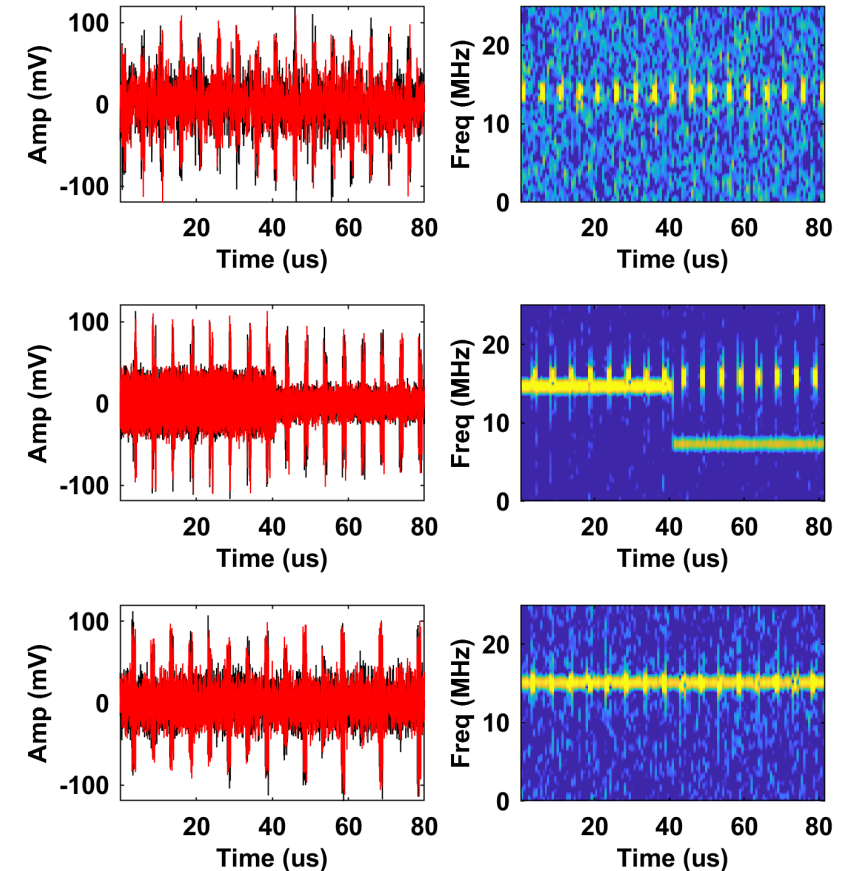
Radar Waveform	<i>Nothing</i>	<i>Interference</i>	<i>Surveillance</i>	<i>Ground (LFM1)</i>	<i>Ground (LFM2)</i>	<i>MTI</i>	<i>Airborne (Med PRF)</i>	<i>Airborne (High PRF)</i>	<i>Ground (Frank Code)</i>	<i>Nautical (Short Range)</i>	<i>Nautical (Long Range)</i>	<i>Nautical (Long Range)</i>	<i>Ground (NLFM1)</i>	<i>Ground (NLFM2)</i>	<i>Ground (NLFM3)</i>
Linear Pulse			X	X	X					X	X	X			
Non-Linear Pulse													X	X	X
Phase Coded Pulse									X						
Pulsed Doppler						X	X	X							

Technique demonstration shown with nominal radar signals

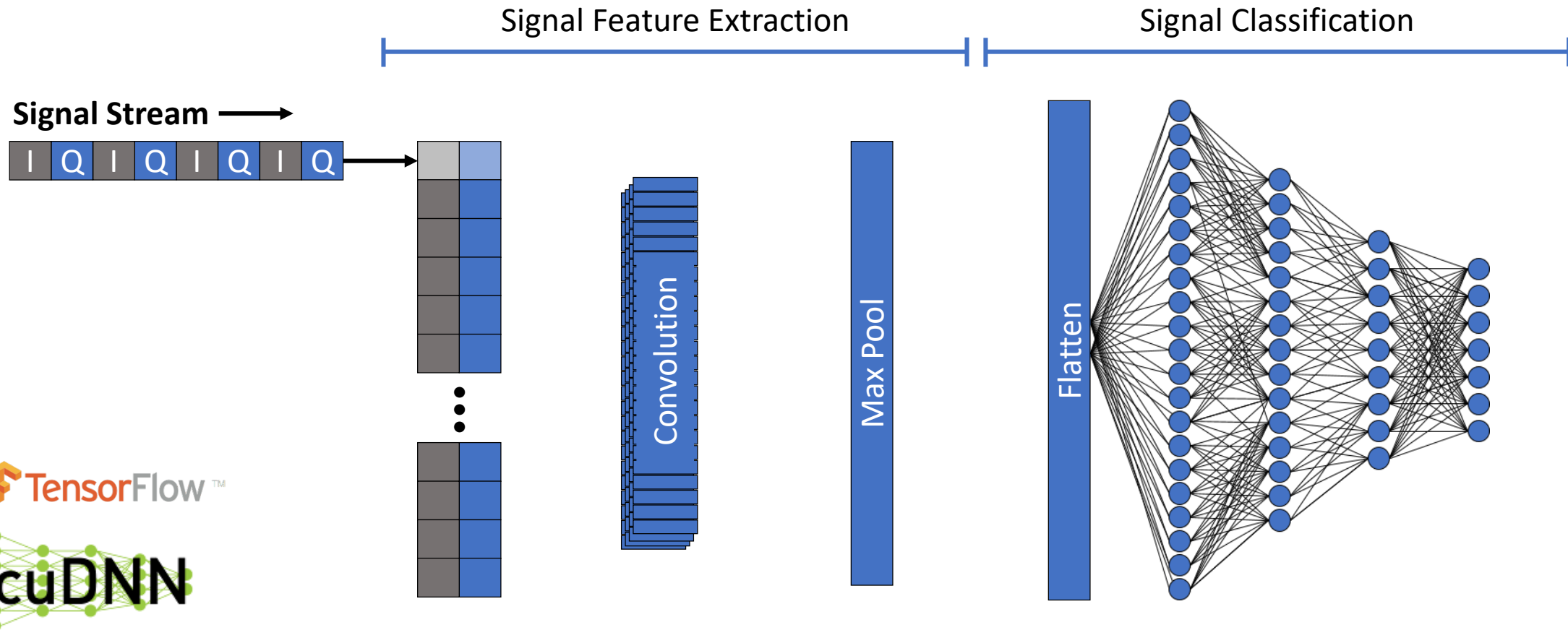
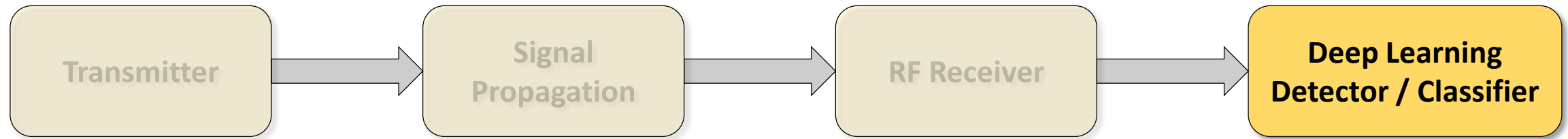
- Method applicable to communications, cellular, and other RF protocols

Dataset Overview

- Goal: Develop a deep learning classifier that detects signals below noise floor
 - Requires training on noisy data with and without interference
- Swept SNIR from -35 dB to 20 dB in 1 dB increments
 - 1000 training segments per SNIR
 - 500 inference segments per SNIR
 - Up to 3 interferers in each segment

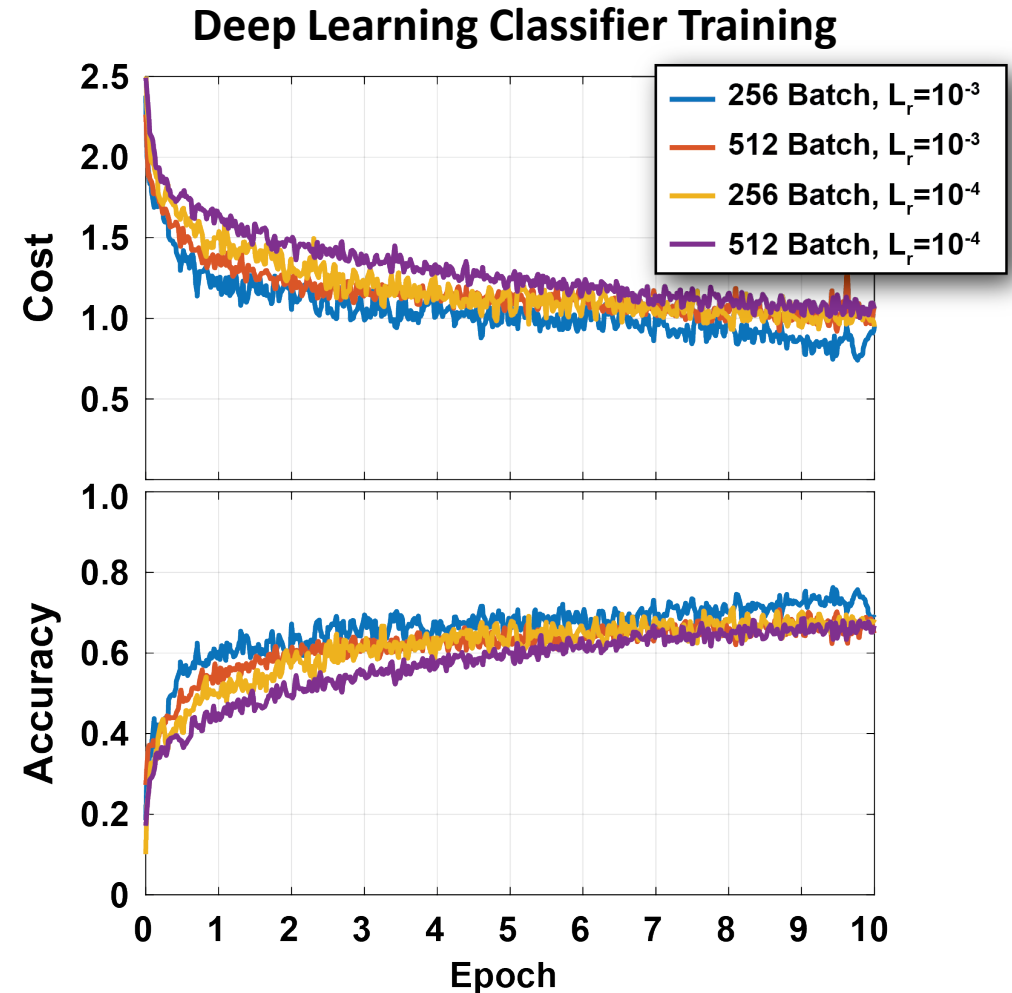


Radar Signal Detector Model: Example Classifier

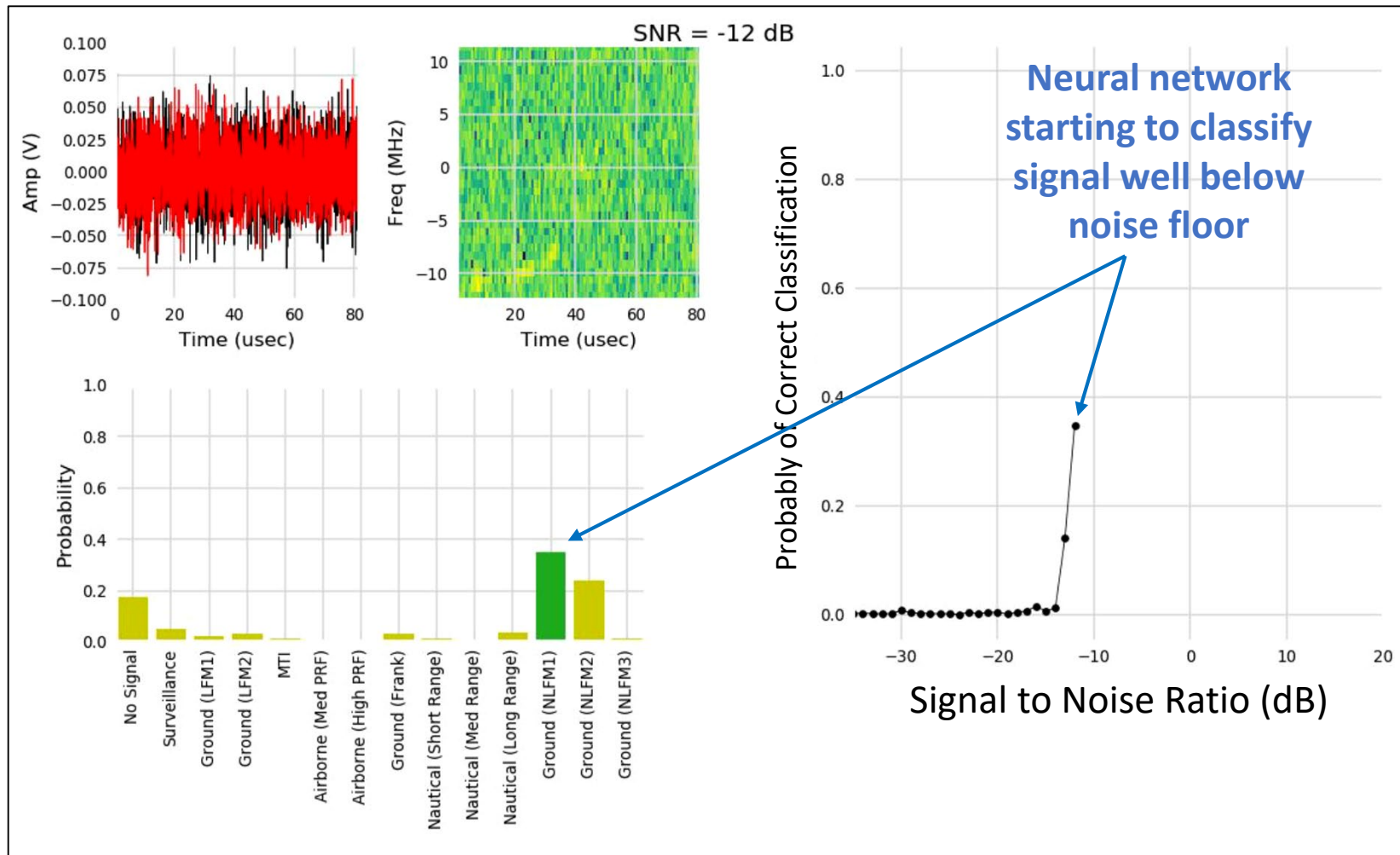


Training Process and Progress

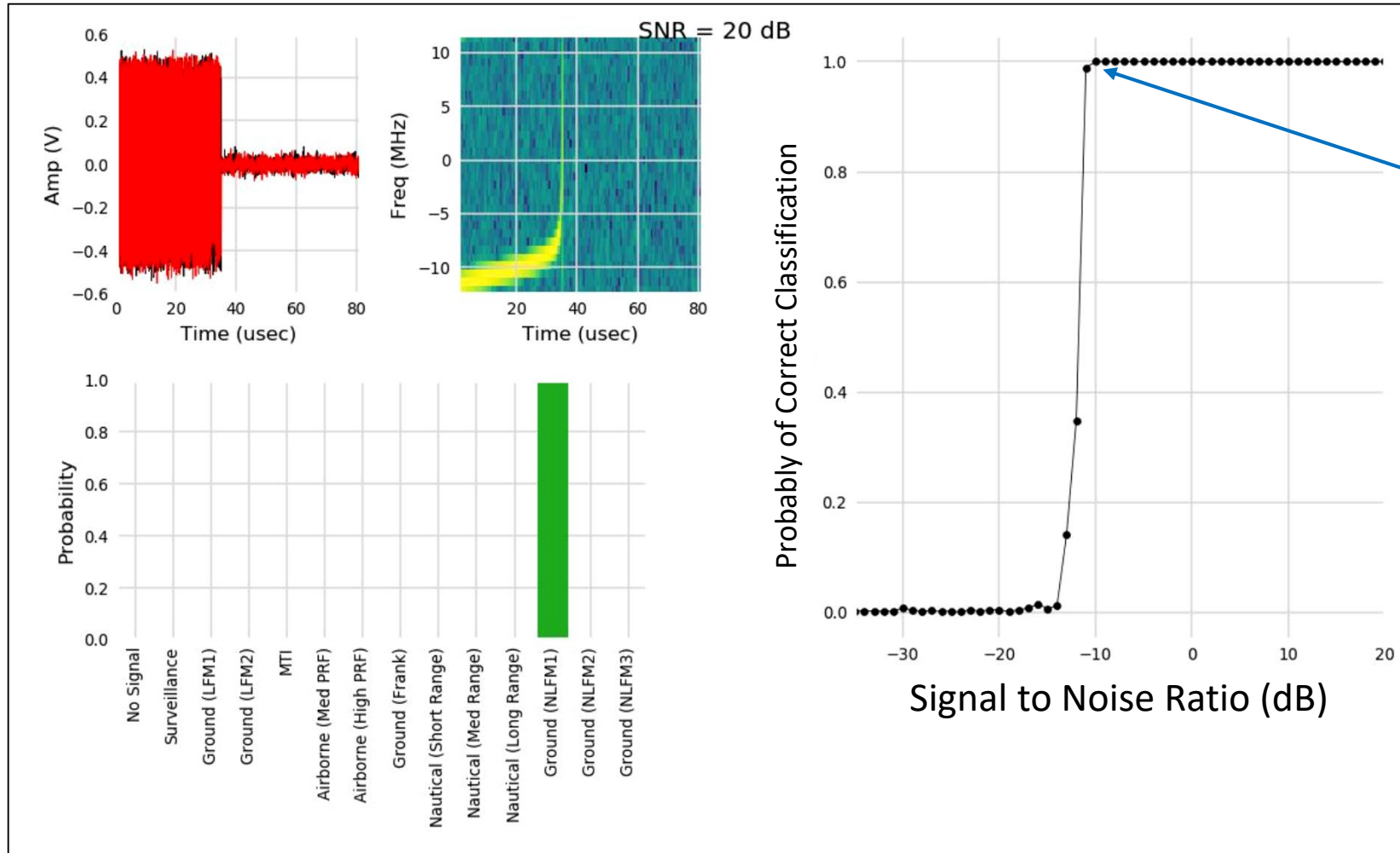
- 1000 training segments per SNR
 - 55 different SNR values
- Training on low SNR values increase detection sensitivity
- 100% accuracy not expected due to training at extremely low SNR values
- Softmax cross entropy
- Adam Optimizer



Detecting and Classifying Low Power Signals

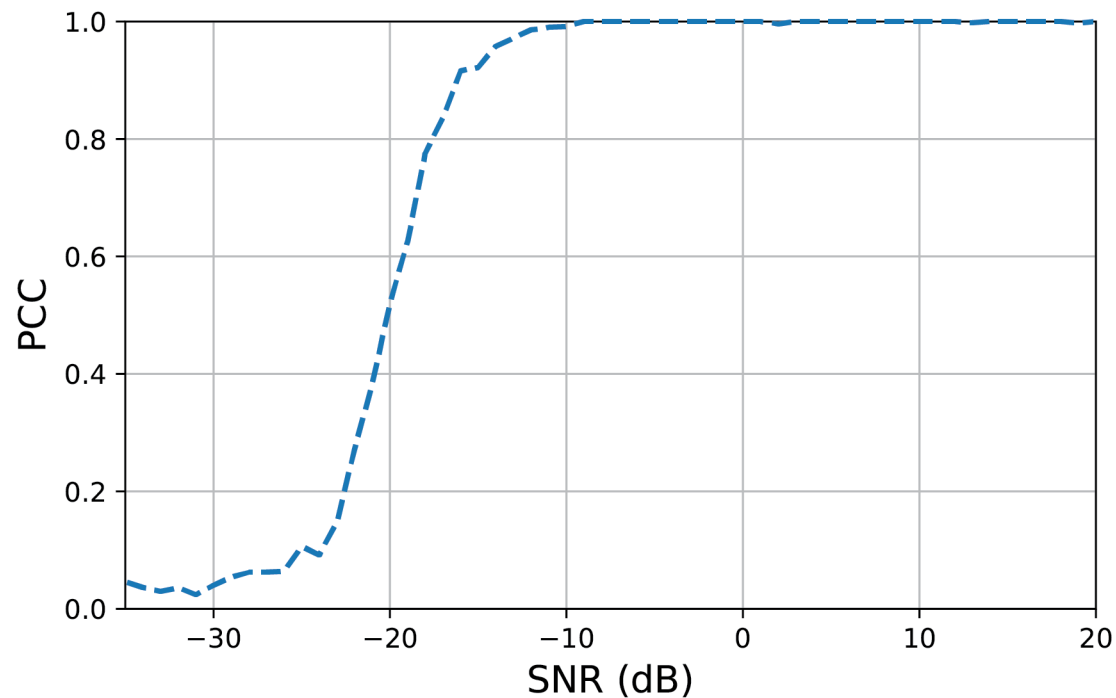


Detecting and Classifying Low Power Signals



Receiver Operating Characteristic (ROC) Curve

**Probability of Correct Classification for
Radar Signal on Previous Slide**

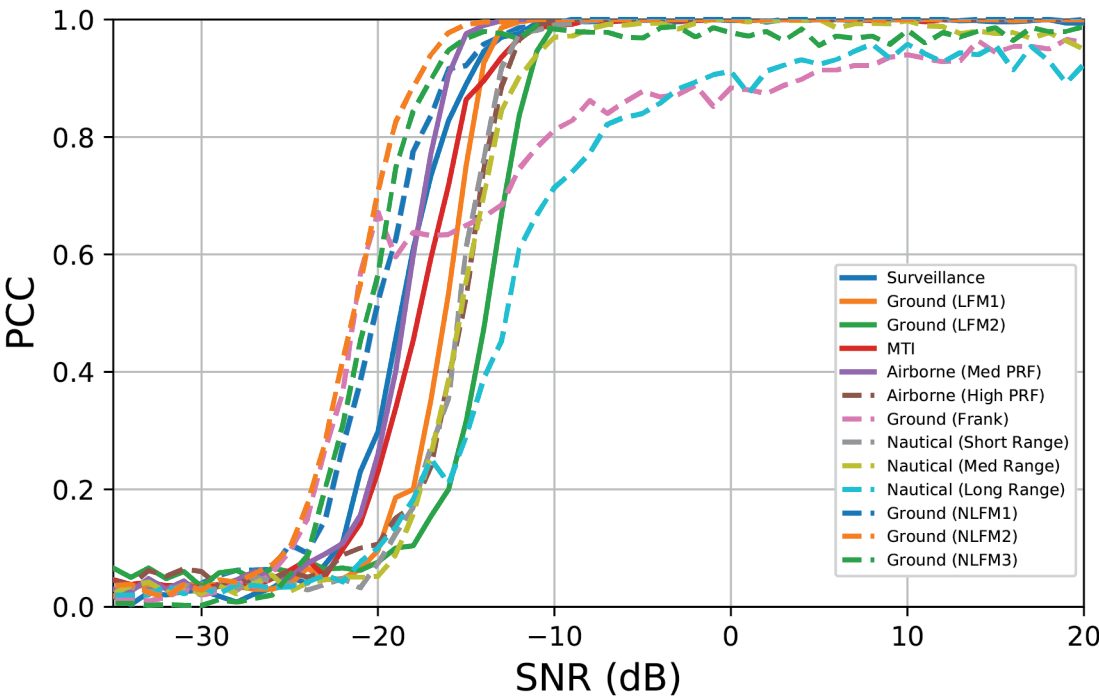


Decibel (dB) Refresher

Signal-to-Noise Ratio (dB)	Receiver Noise Power (milliwatts)	Received Signal Power (milliwatts)
20	1	100
10	1	10
0	1	1
-10	1	0.1
-20	1	0.01
-30	1	0.001

Receiver Operating Characteristic (ROC) Curve

Probability of Correct Classification
for All Radars

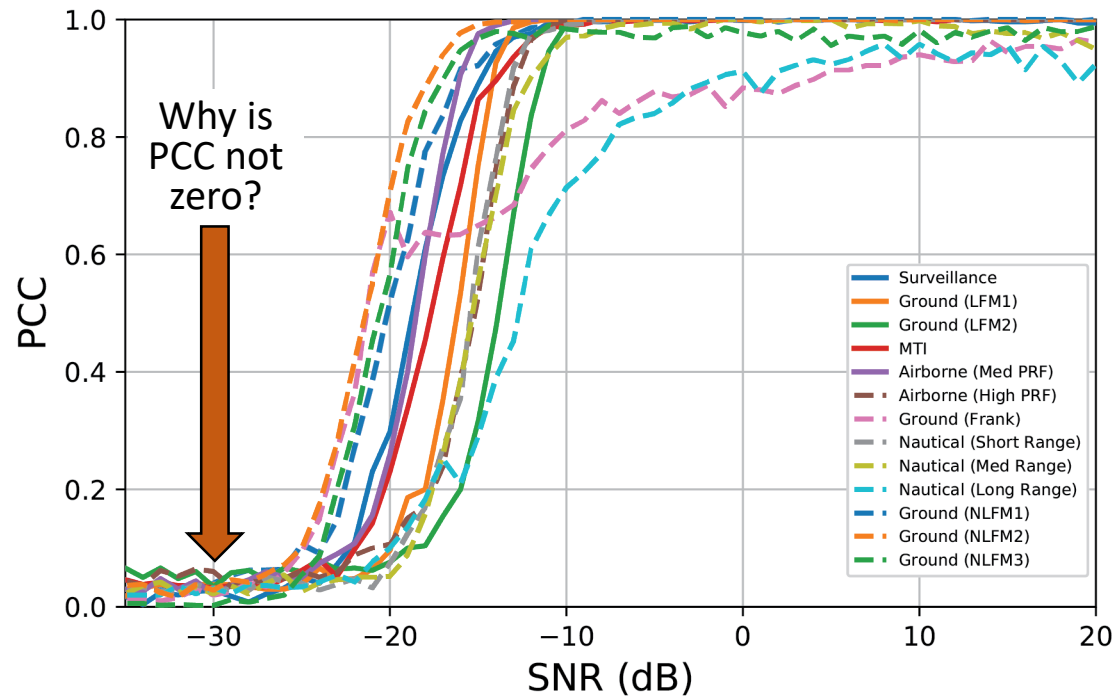


Decibel (dB) Refresher

Signal-to-Noise Ratio (dB)	Receiver Noise Power (milliwatts)	Received Signal Power (milliwatts)
20	1	100
10	1	10
0	1	1
-10	1	0.1
-20	1	0.01
-30	1	0.001

Receiver Operating Characteristic (ROC) Curve

Probability of Correct Classification for All Radars

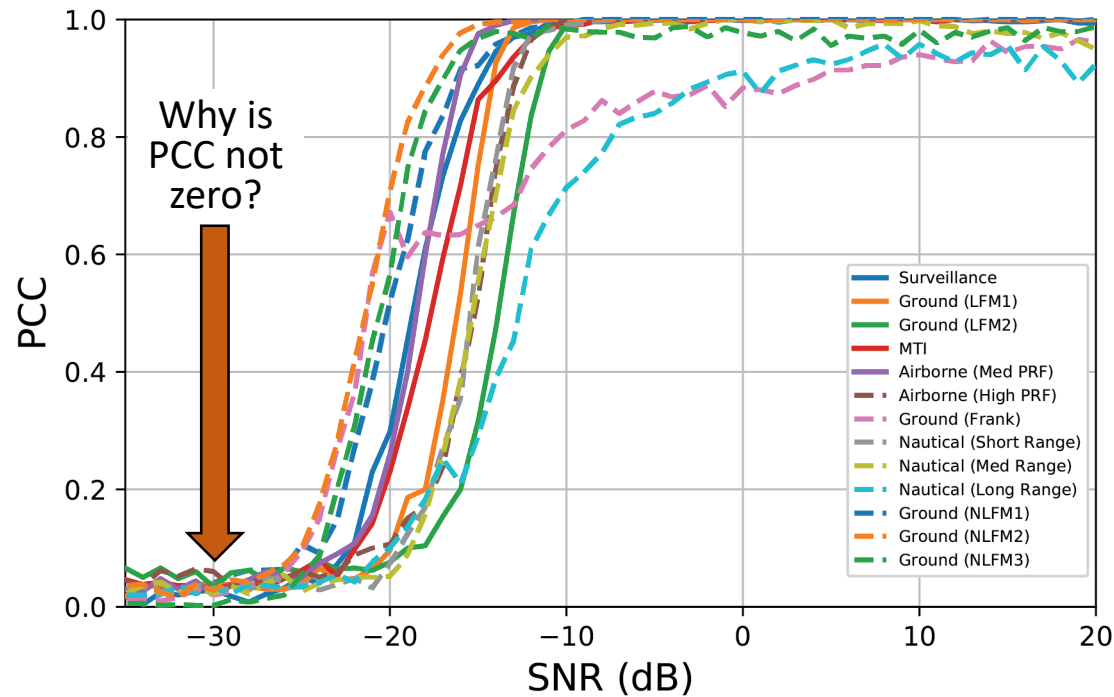


Decibel (dB) Refresher

Signal-to-Noise Ratio (dB)	Receiver Noise Power (milliwatts)	Received Signal Power (milliwatts)
20	1	100
10	1	10
0	1	1
-10	1	0.1
-20	1	0.01
-30	1	0.001

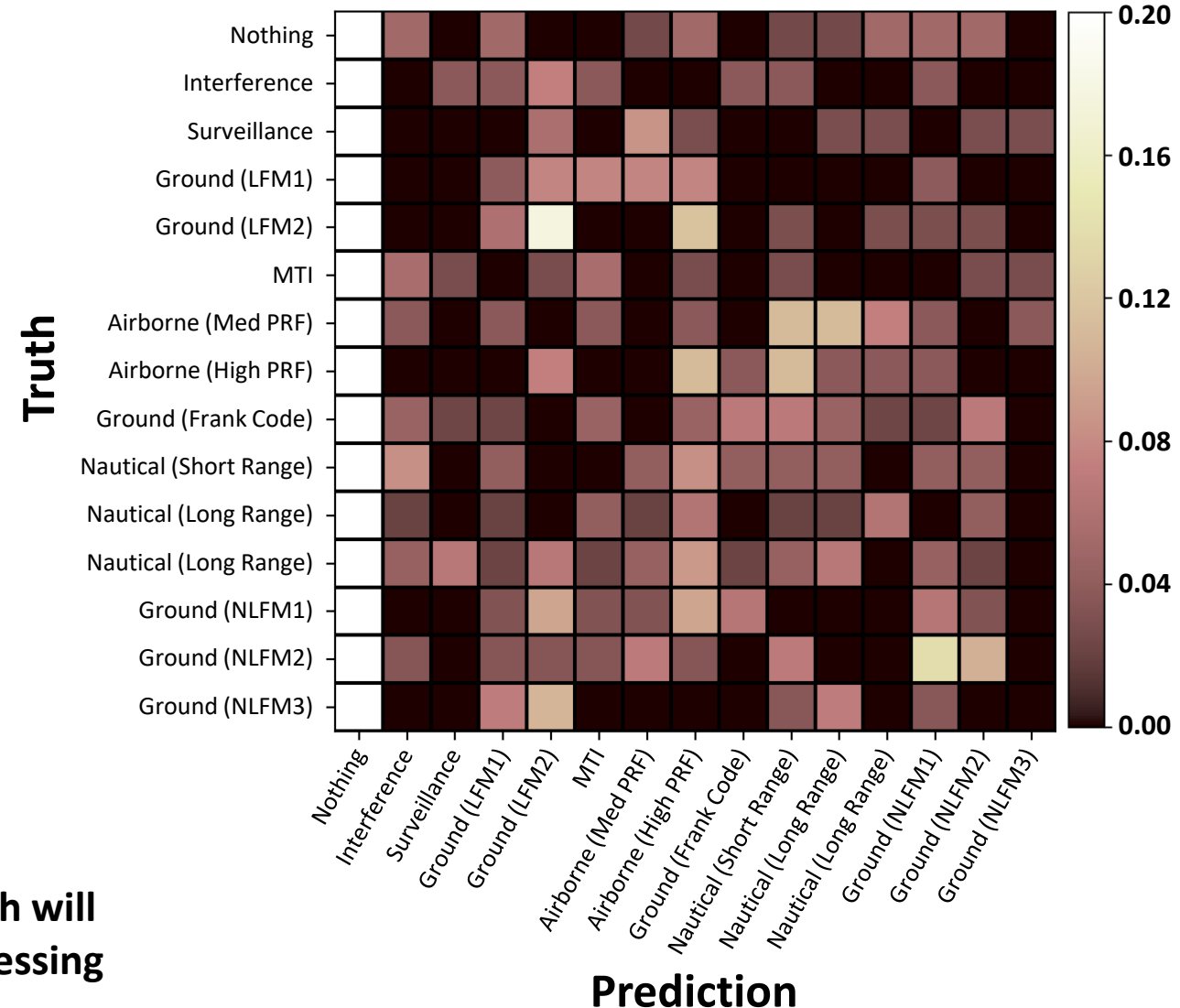
Receiver Operating Characteristic (ROC) Curve

Probability of Correct Classification for Various Radars

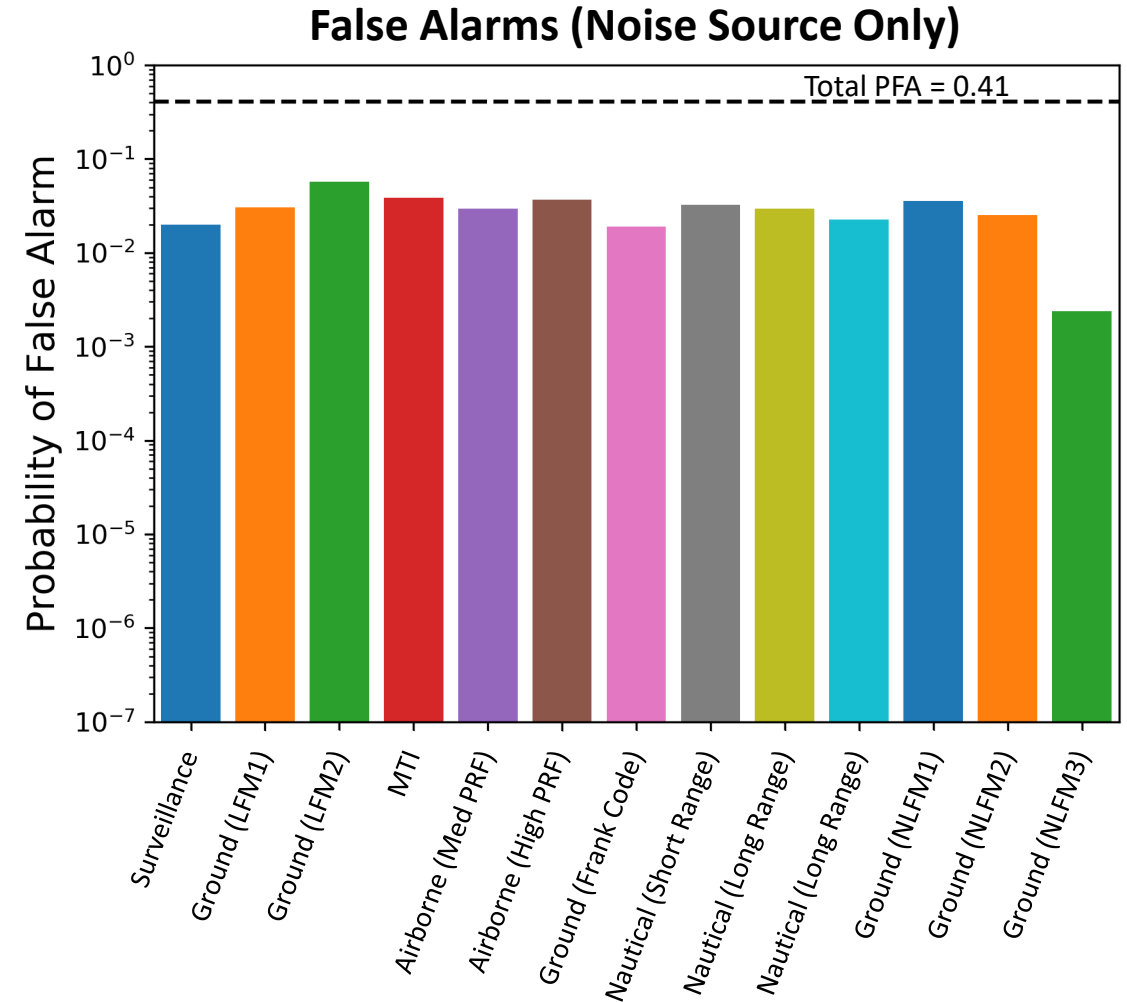
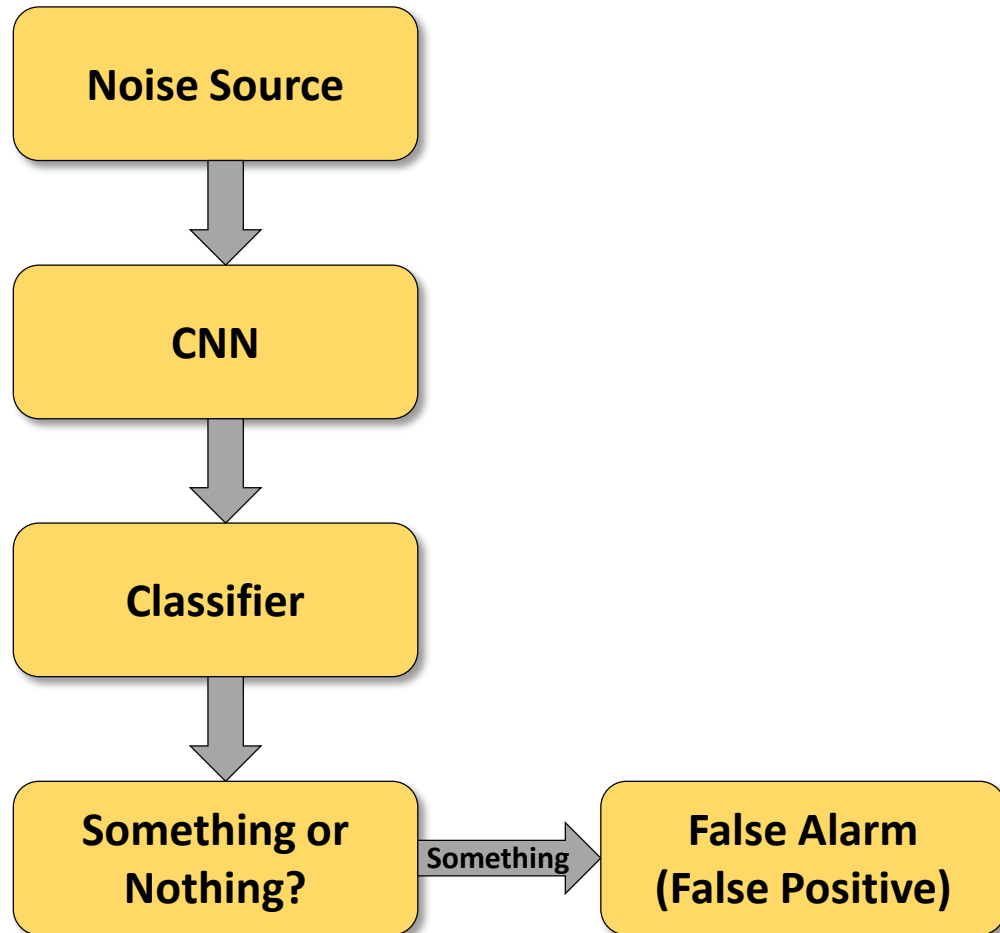


DNN appears to be randomly guessing at low SNR which will create unnecessary requirements on downstream processing

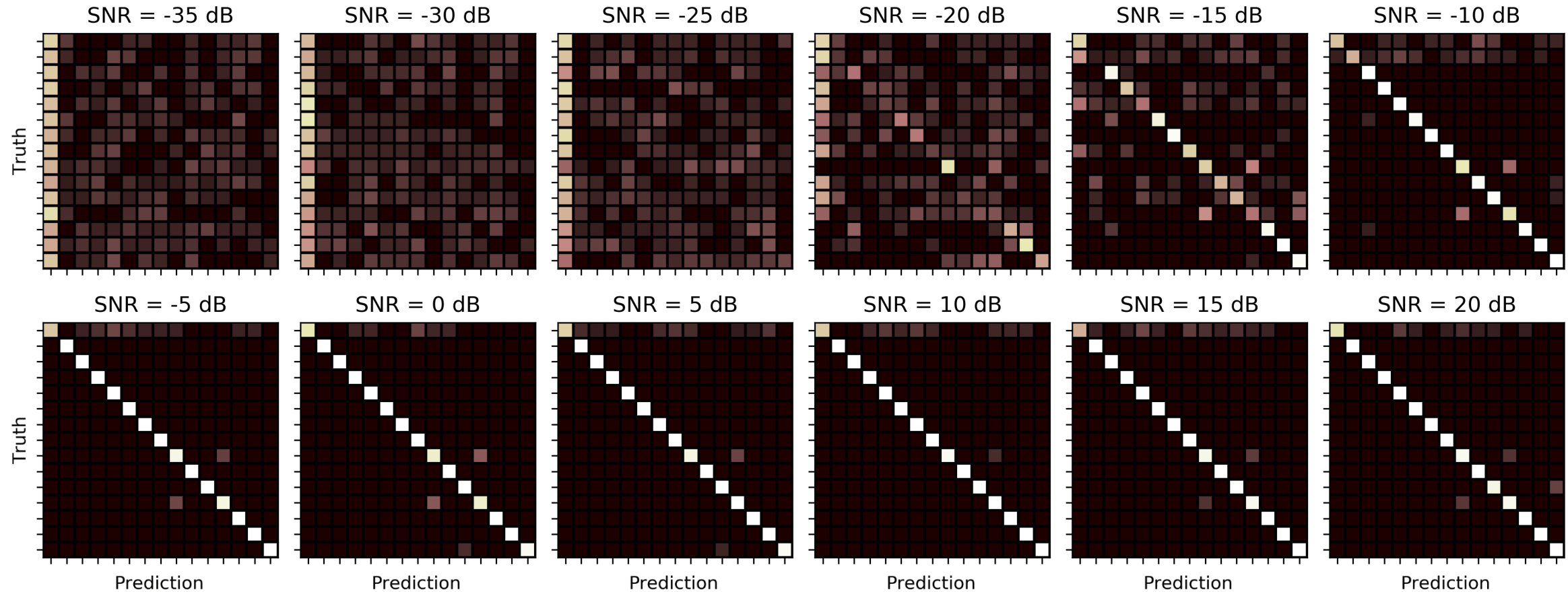
Confusion Matrix (-35 dB SNR)



Methodology for Testing False Positive Rate



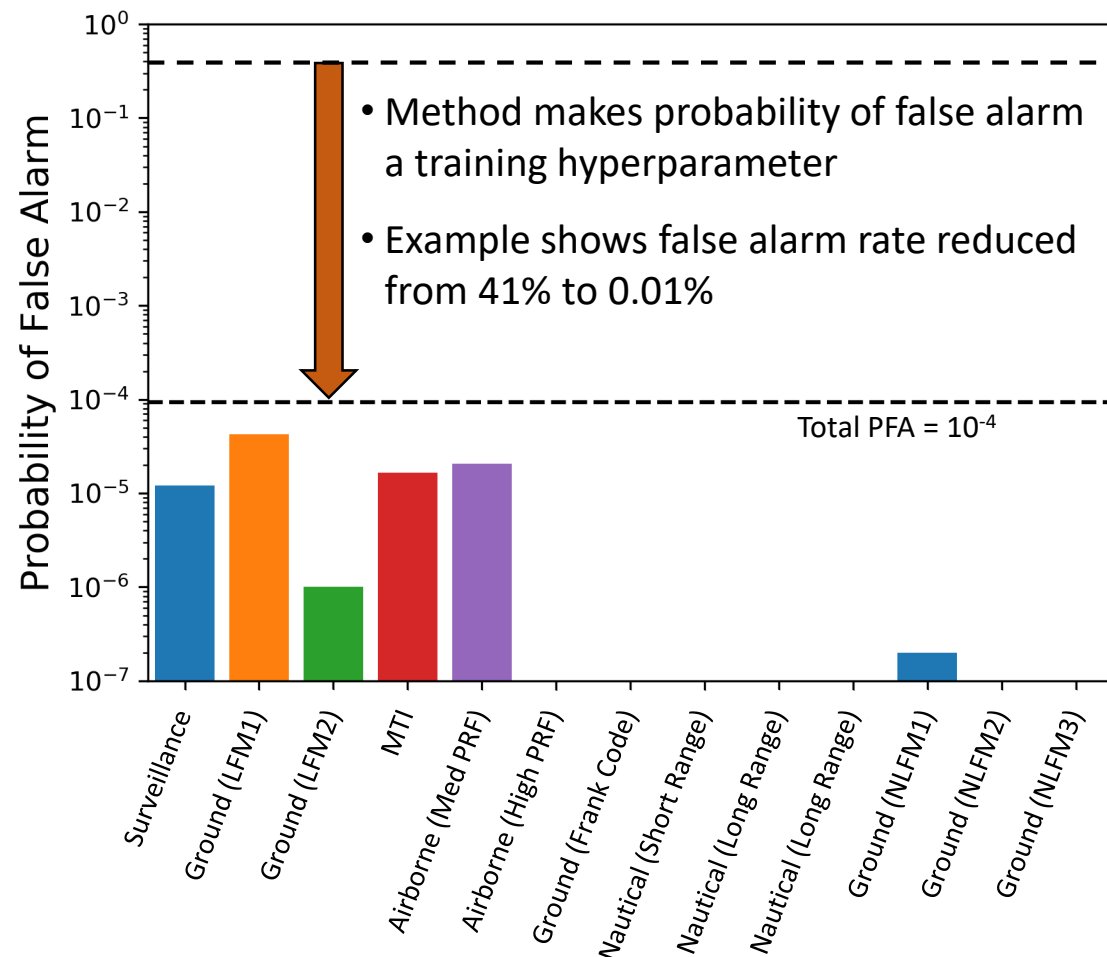
Confusion Matrix and Signal to Noise Ratio



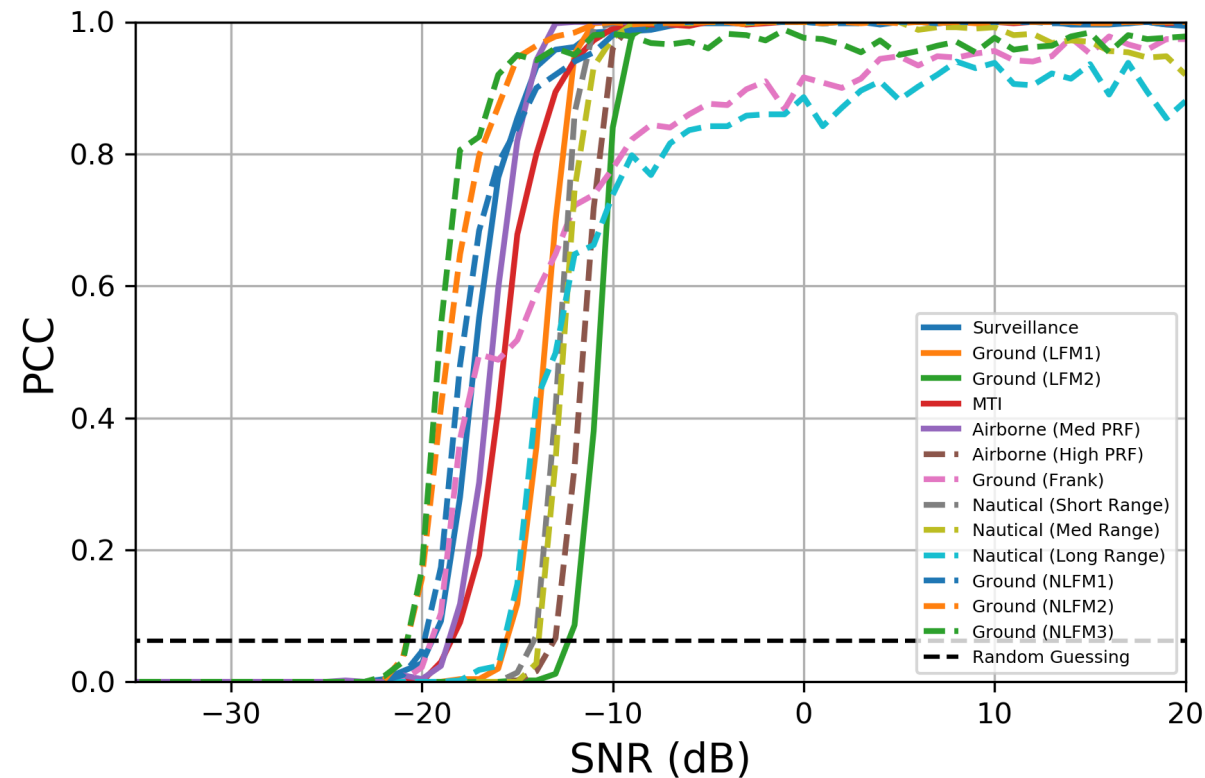
Significant false alarm rate limits algorithm's applicability and creates non-zero probability of correct classification (PCC) at low SNR values

Deepwave Training Method to Reduce False Alarms

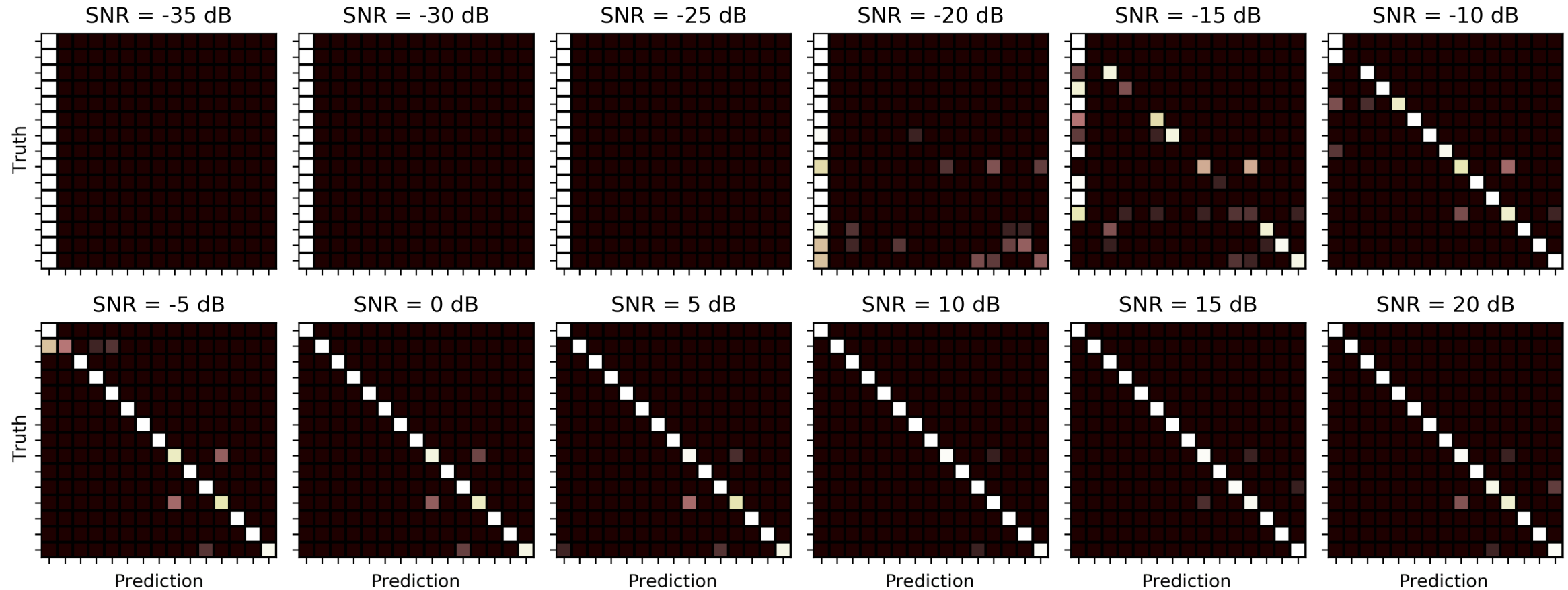
False Alarms (Noise Only)



Probability of Correct Classification for Various Radars



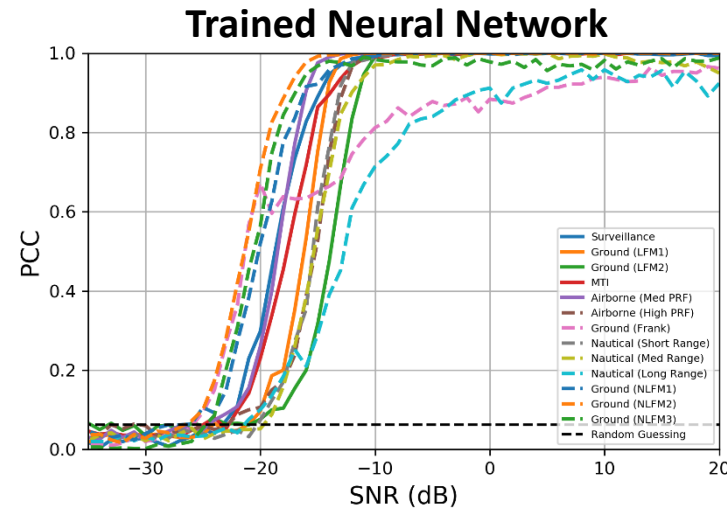
New Inference Confusion Matrix



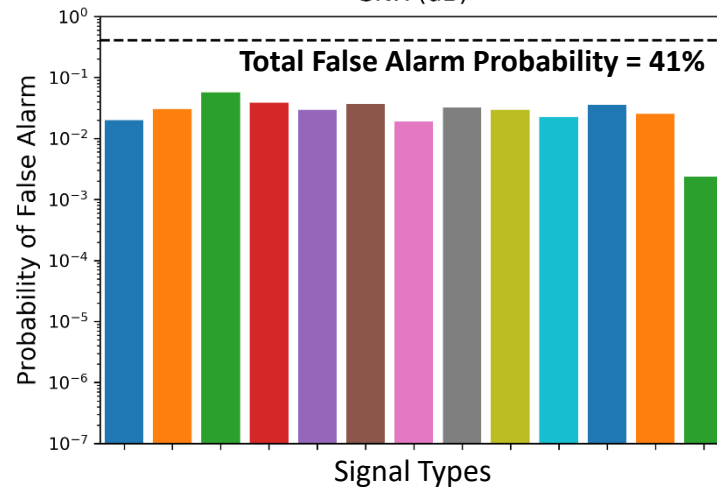
Deepwave training algorithms allows for tunability of false alarm rate

Deepwave Method to Reduce False Alarms

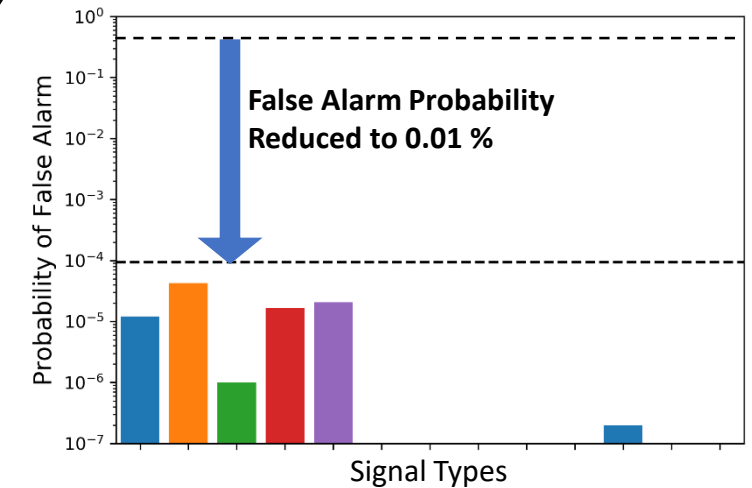
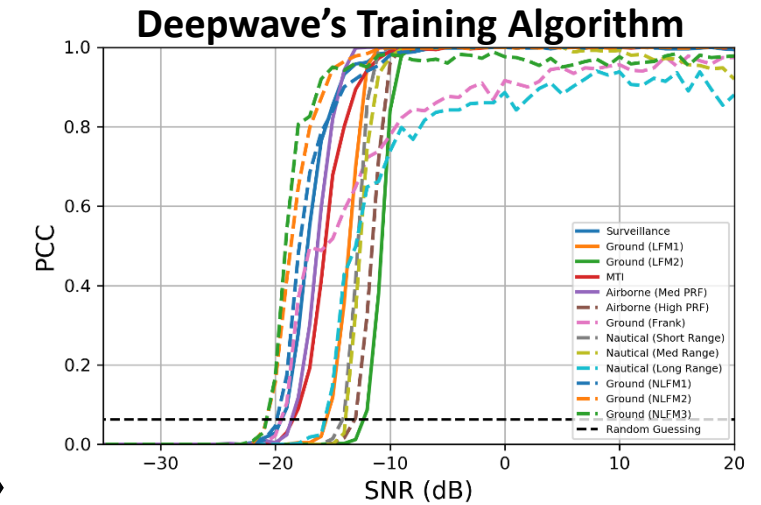
Probability
of Detecting and
Classifying Signal



Probability of
False Alarm
(False Positive)




Apply Deepwave's
False Alarm
Learning Algorithm

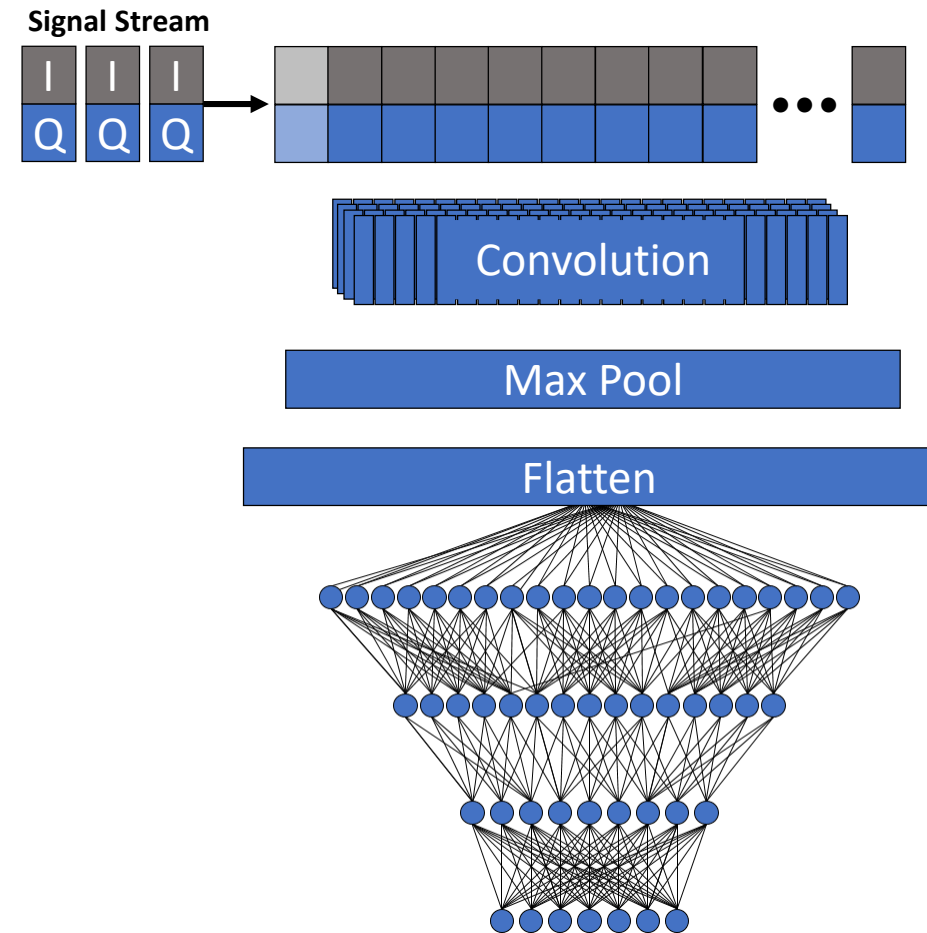


False alarm conscious training method has < 5dB impact on detector sensitivity

Outline

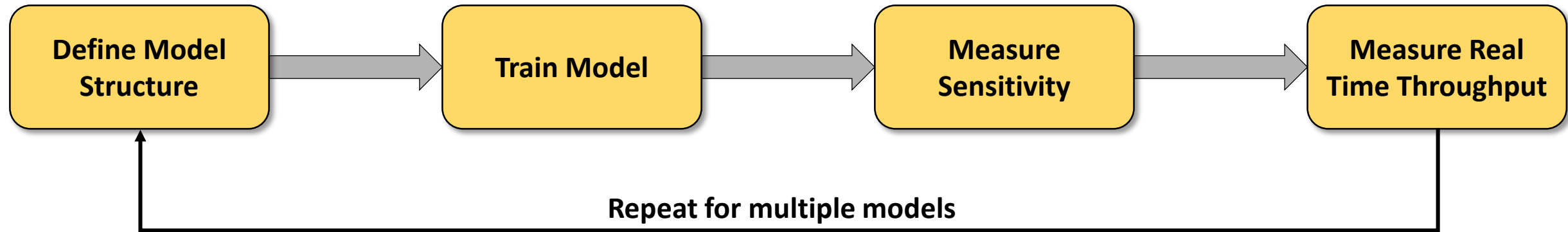
- Introduction
- Deep Learning in RF Systems
- Deepwave Digital Technology
- Example Signal Detection and Classification
-  • Real-time Signal Processing Benchmarks for GPUs
- Summary

Critical Performance Parameters

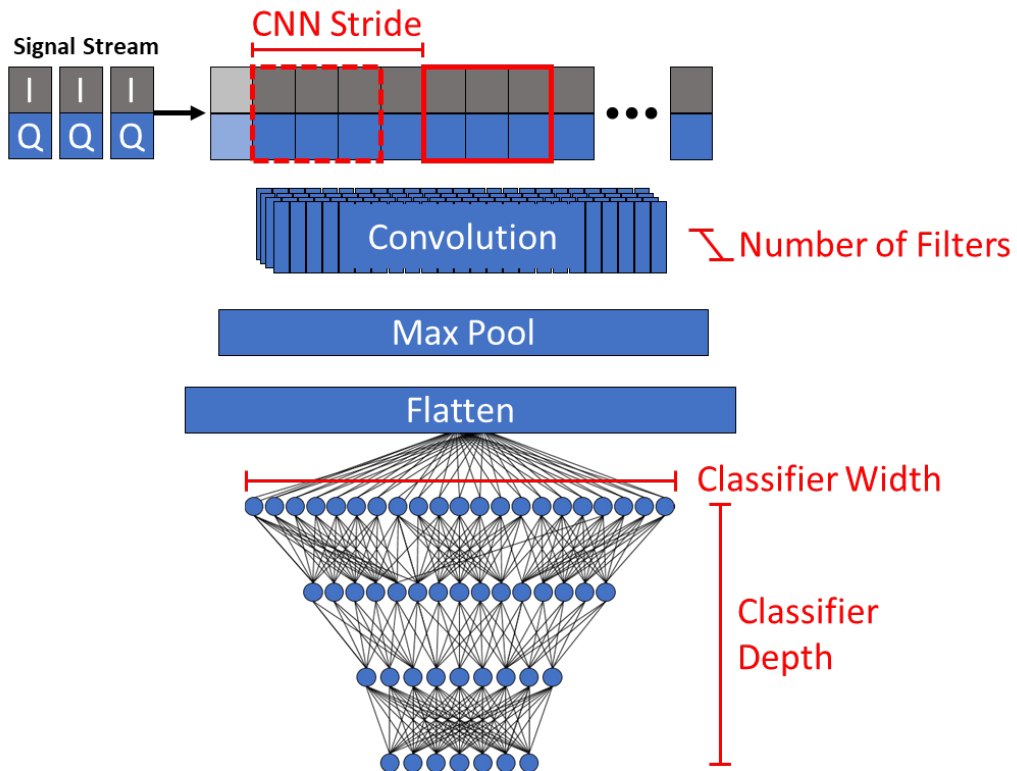


- What makes a DNN model “good?”
 - **High Sensitivity** – detects low powered signals
 - **Low false alarm rate** – minimize false positives
 - **High real time bandwidth**
 - **Low computational requirements**
 - **Low latency**
- Most of these critical performance parameters are adversarial

Performance Benchmarking Test Setup



Performance Benchmarking Test Setup



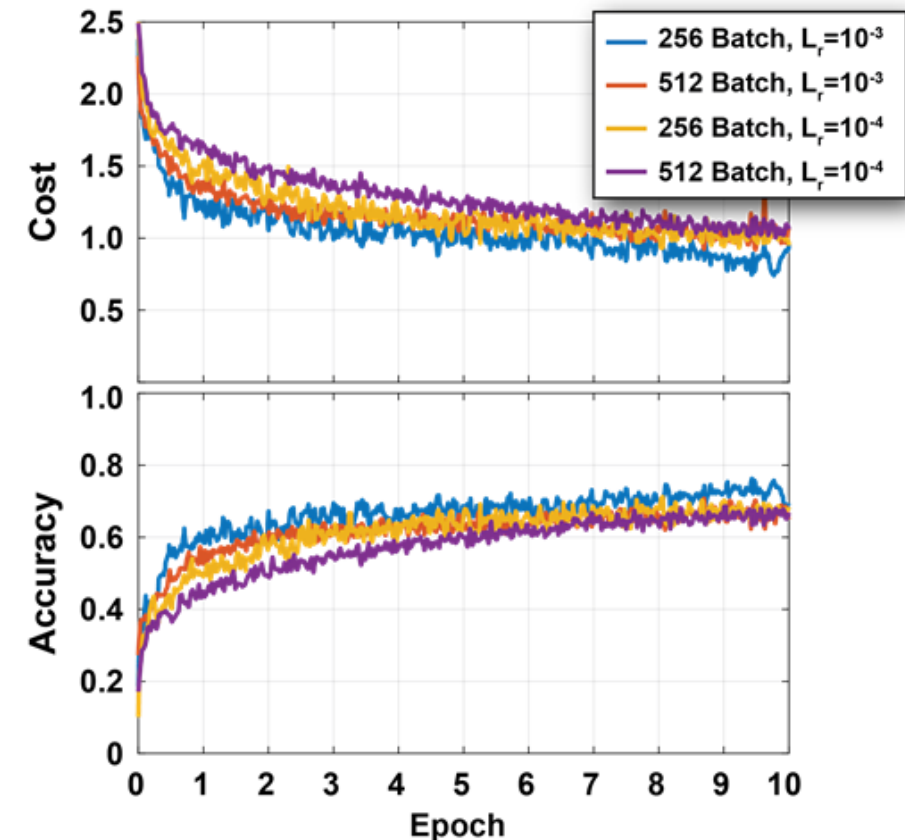
Model Tuning Variables

	Min Val	Max Val	Total
CNN Stride	1	16	9
Number of Filters	4	256	7
Classifier Layer 1 Width	64	128	3
Classifier Layer 2 Width	32	64	3
Classifier Layer 3 Width	0	64	2
Batch Size	1	256	8
Total Model Combinations Tested			728

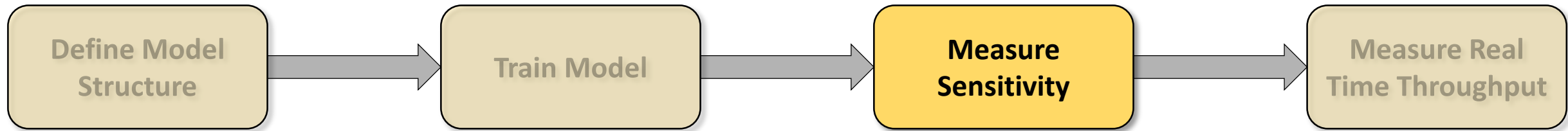
Performance Benchmarking Test Setup



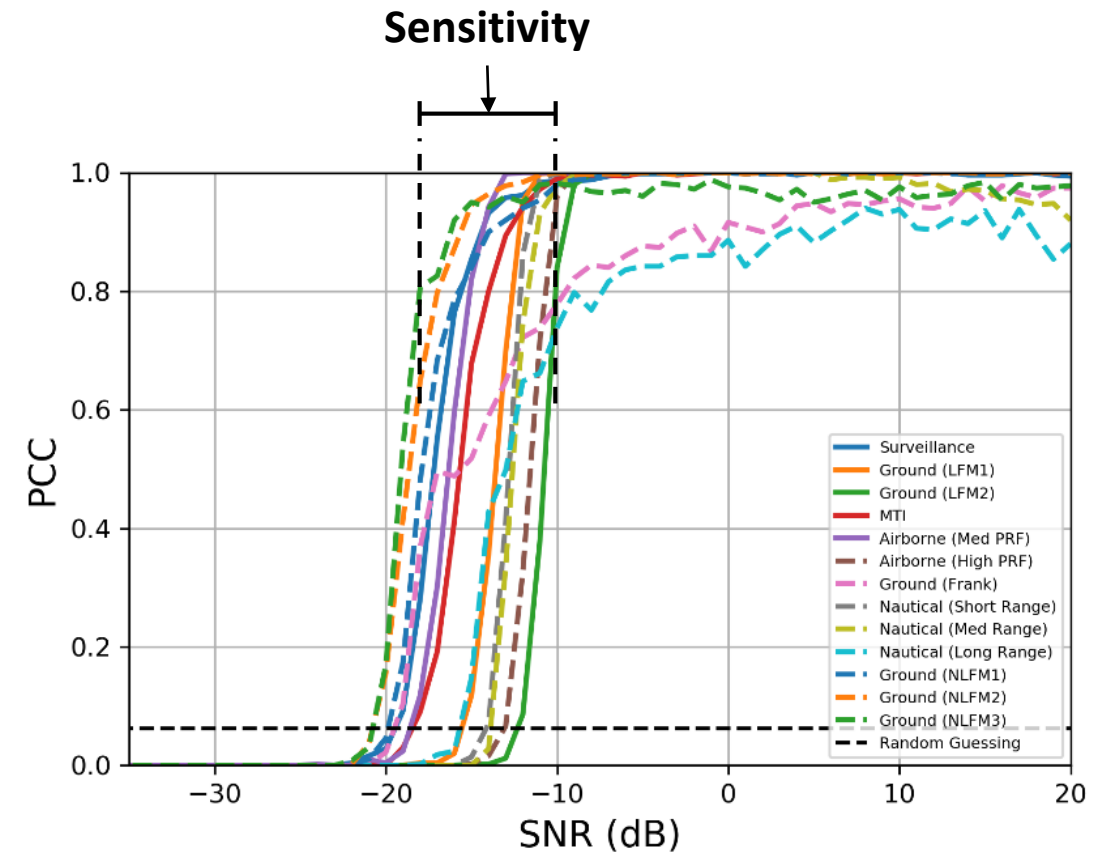
- 1000 training segments per SNR
 - 55 different SNR values
- Softmax cross entropy
- Adam Optimizer
- Quadro GP100 GPU
- Create UFF File for each model



Performance Benchmarking Test Setup



- Compute receiver operating characteristic (ROC) curve for each model
- Define sensitivity to be where median PCC = 50% for all signal types



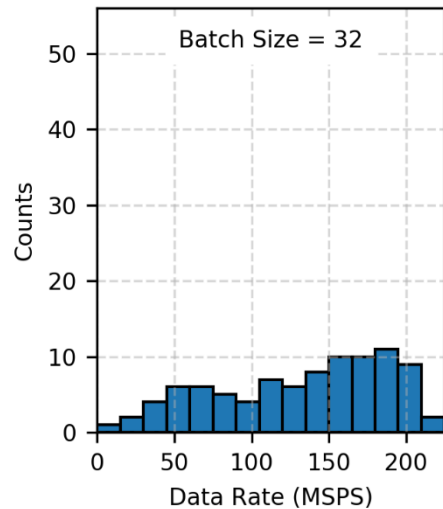
Performance Benchmarking Test Setup



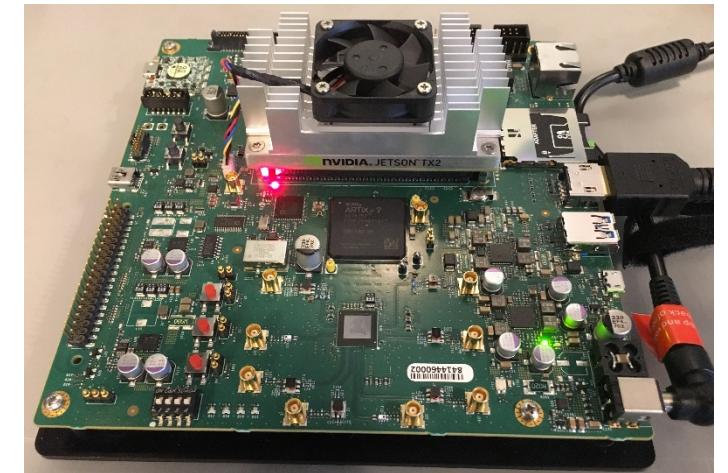
- **Create TensorRT PLAN file for each platform tested**
 - **Load signal data into RAM**
 - **Stream unthrottled data to gr-wavelearner**
- **Probe data rate at two locations:**
 - 1. Aggregate data rate for entire process**
 - Number of bytes processed / wall time
 - 2. Computation data rate in work() function**
 - Number of bytes process / computation time

Data Rate Benchmark for AIR-T (Tegra TX2)

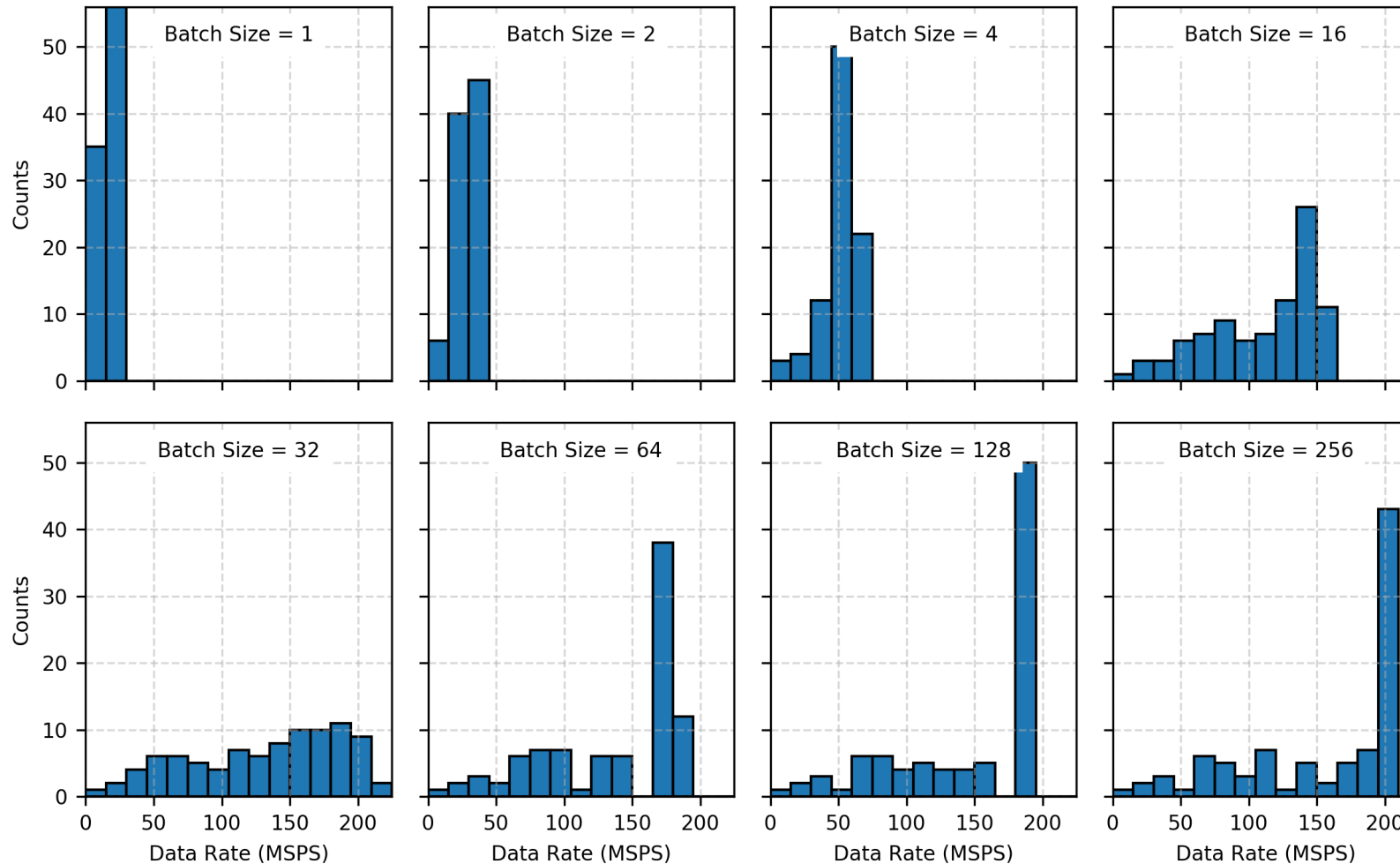
- Tested 91 different CNN classifier models
- Maximum real-time inference data rate for 8 different batch sizes
- Able to achieve 200 MSPS (real) with AIR-T



AIR-T

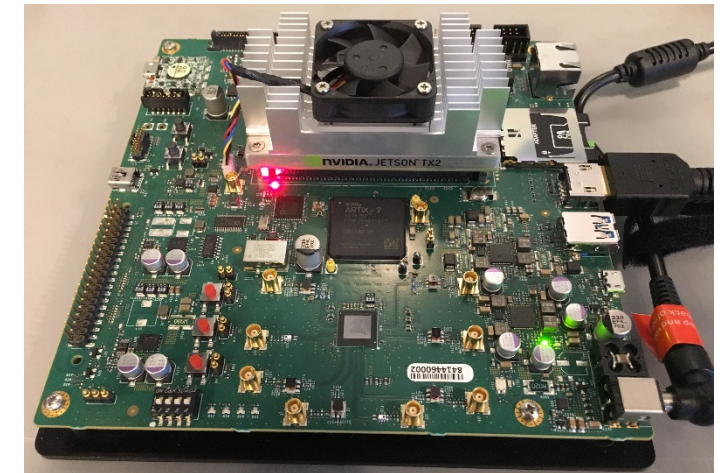


Data Rate Benchmark for AIR-T (Tegra TX2)

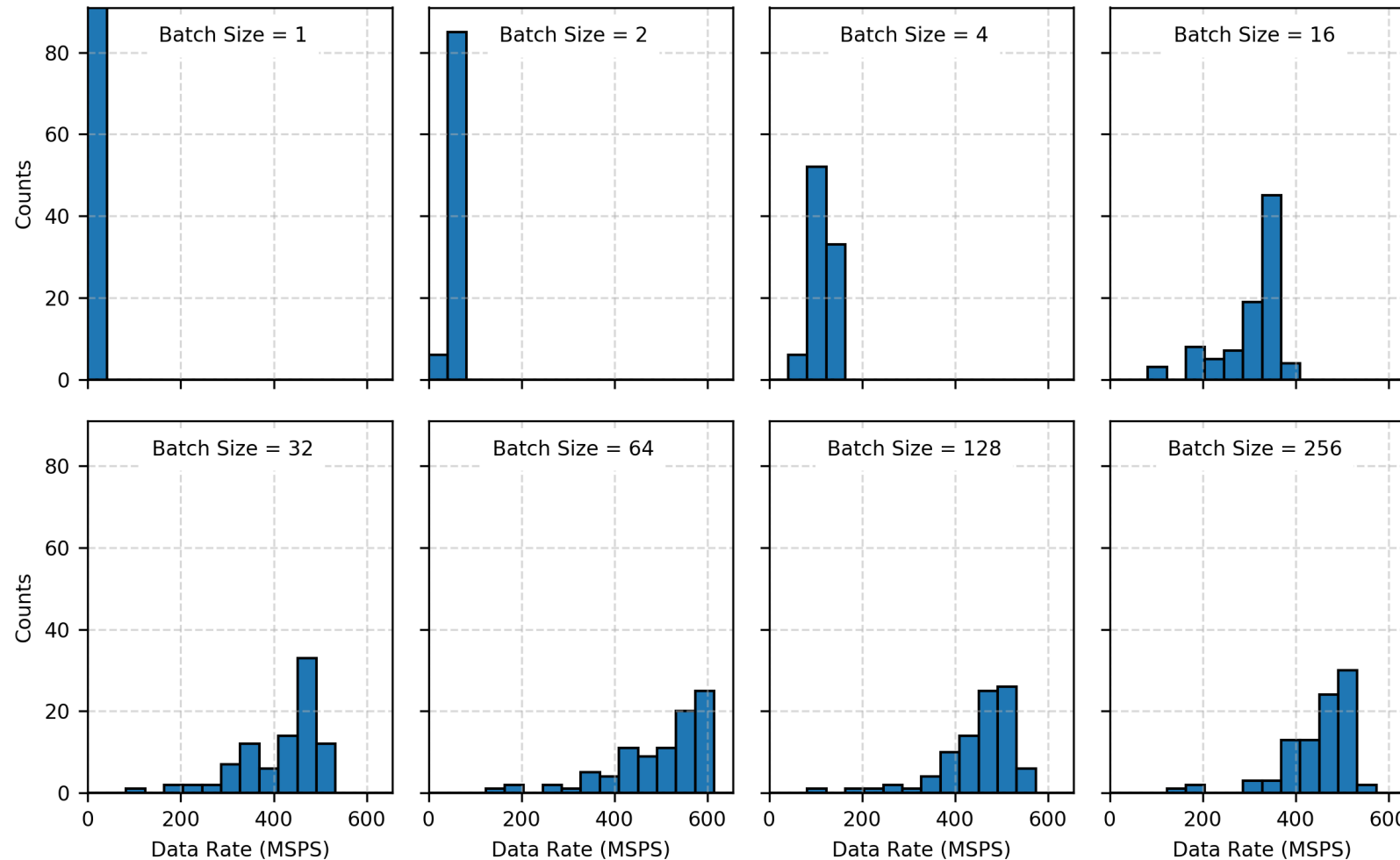


- Tested 91 different CNN classifier models
- Maximum real-time inference data rate for 8 different batch sizes
- Able to achieve 200 MSPS (real samples) with AIR-T

AIR-T



Data Rate Benchmark for Desktop (Quadro P100)



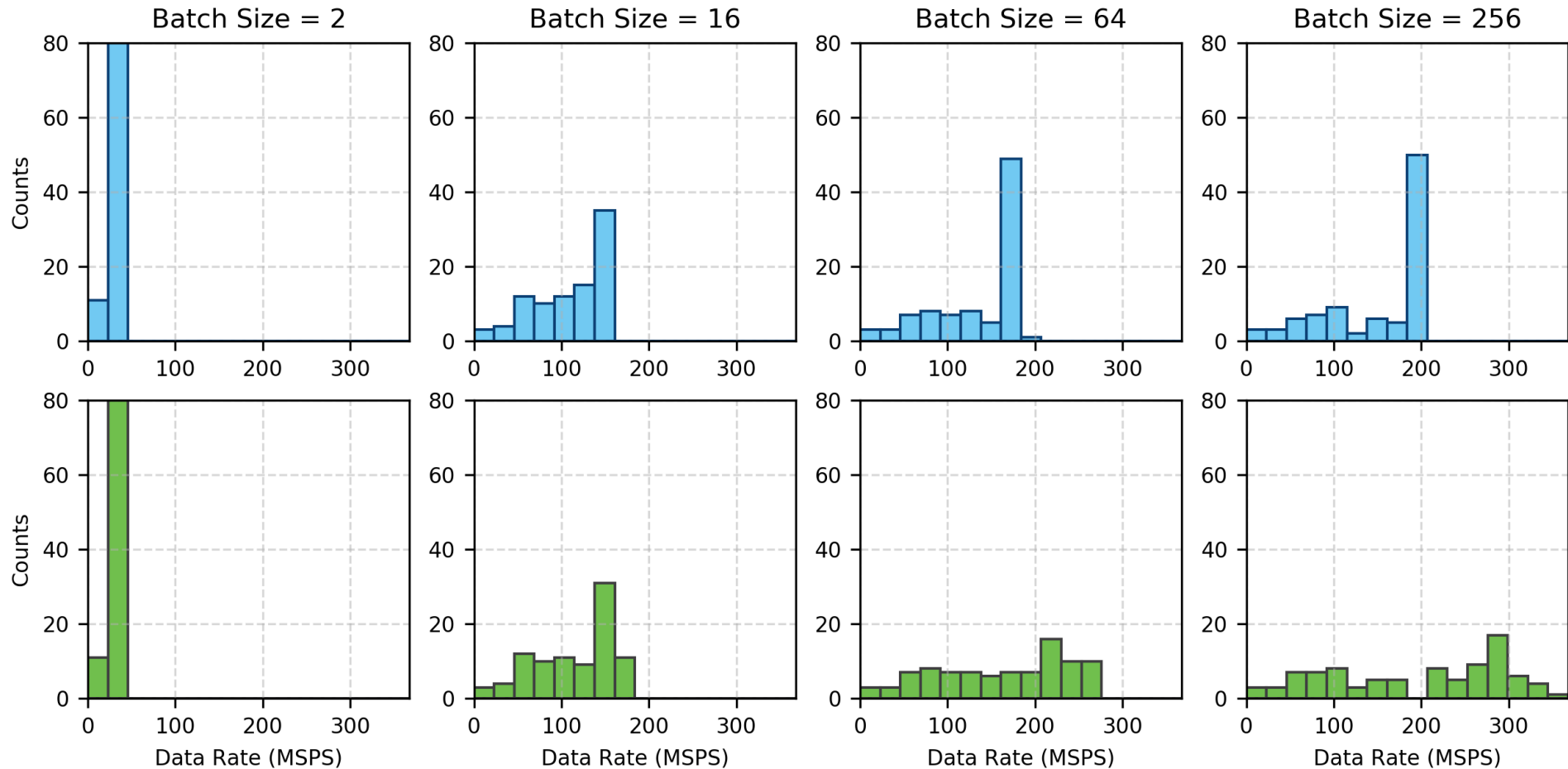
- Tested 91 different CNN classifier models
- Maximum real-time inference data rate for 8 different batch sizes
- Using unified memory will increase throughput

Desktop (GP100)



Wall Time vs. Compute Time for AIR-T

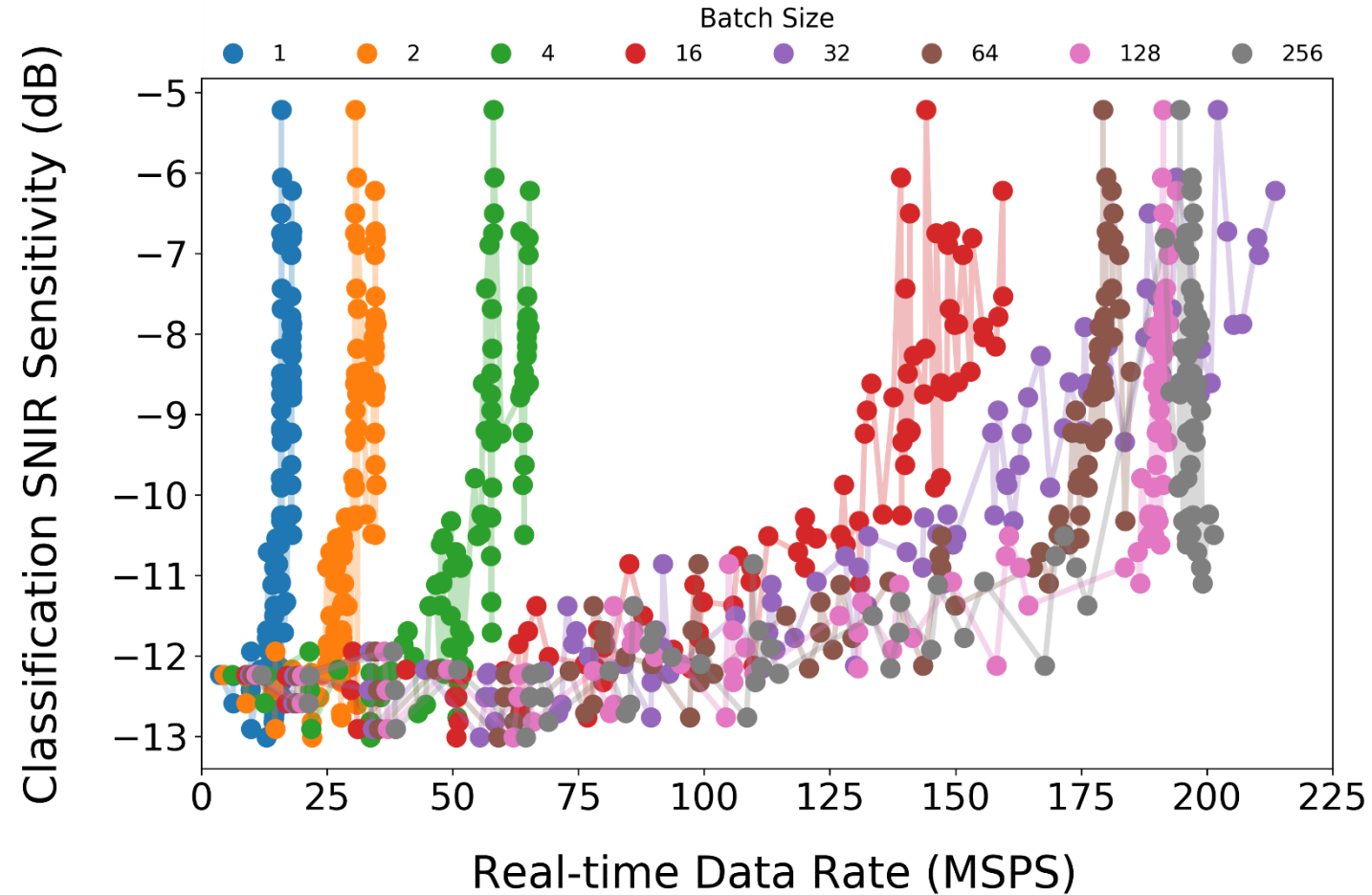
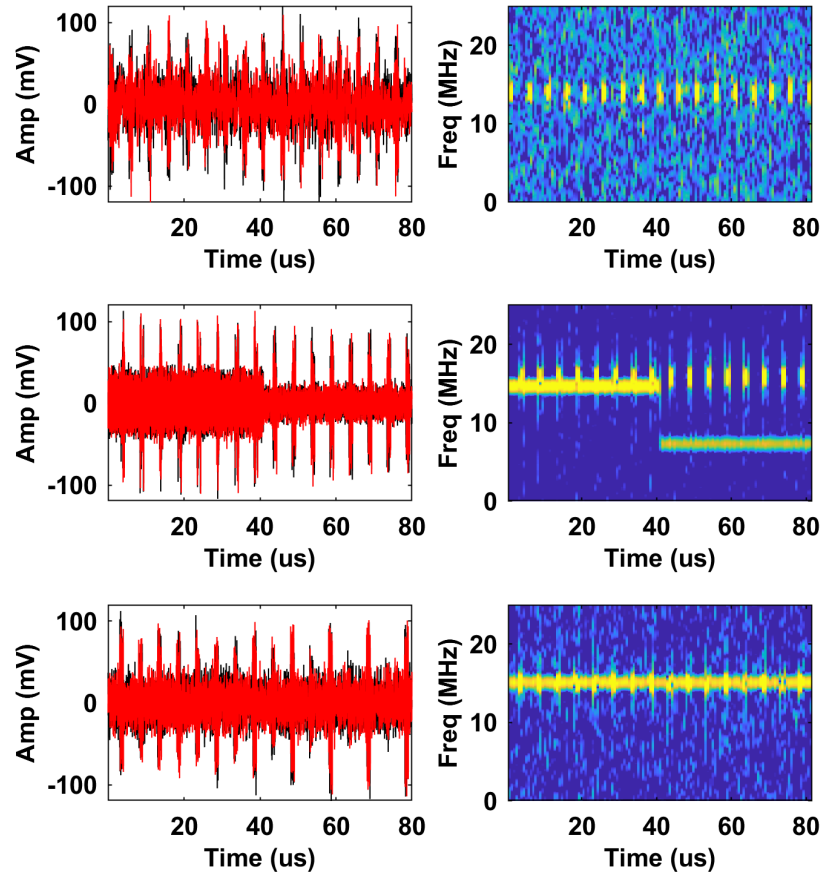
Wall
Time



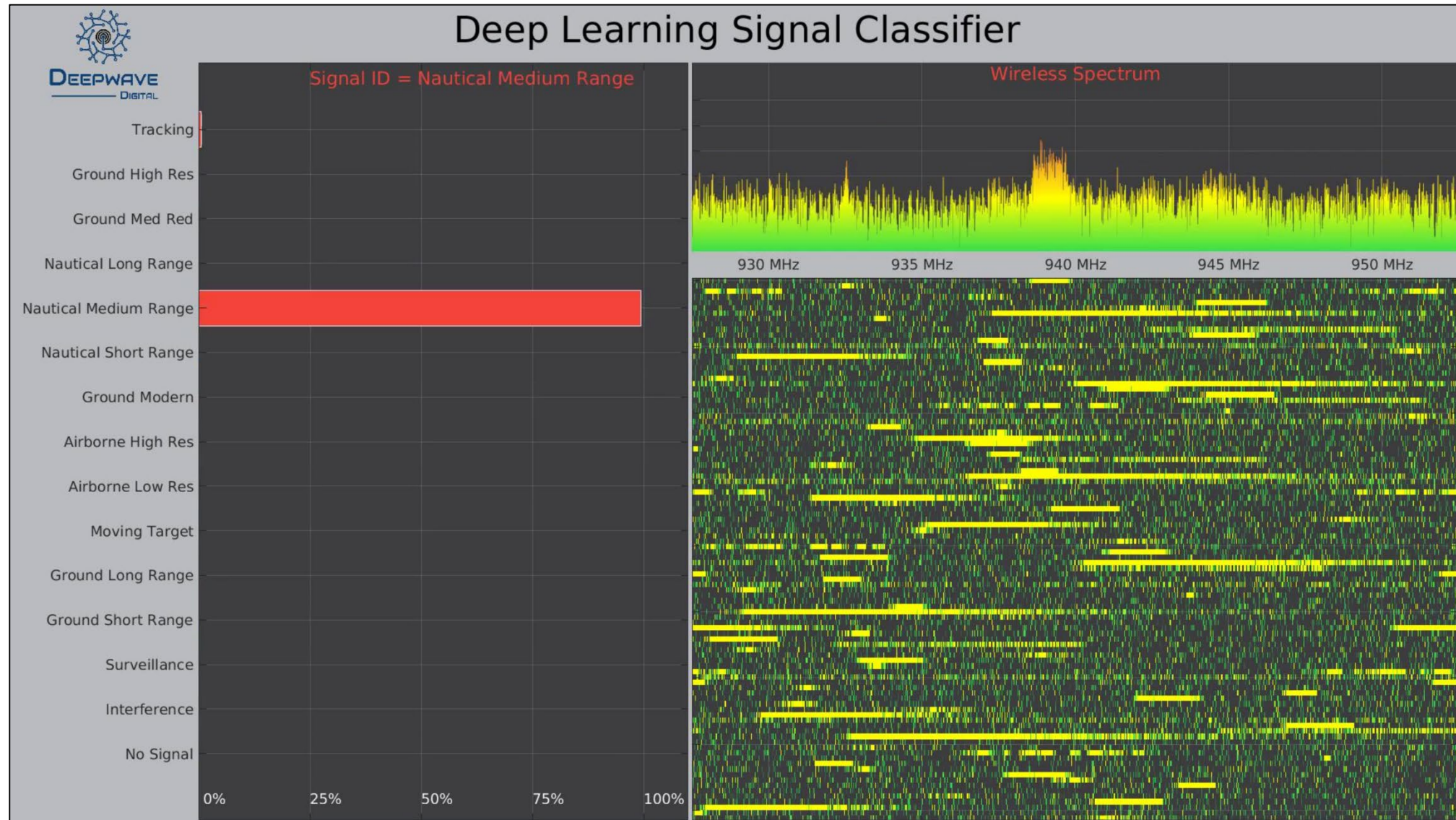
Compute
Time

Real time data rate limited by GNU Radio overhead

Model Accuracy Benchmarks



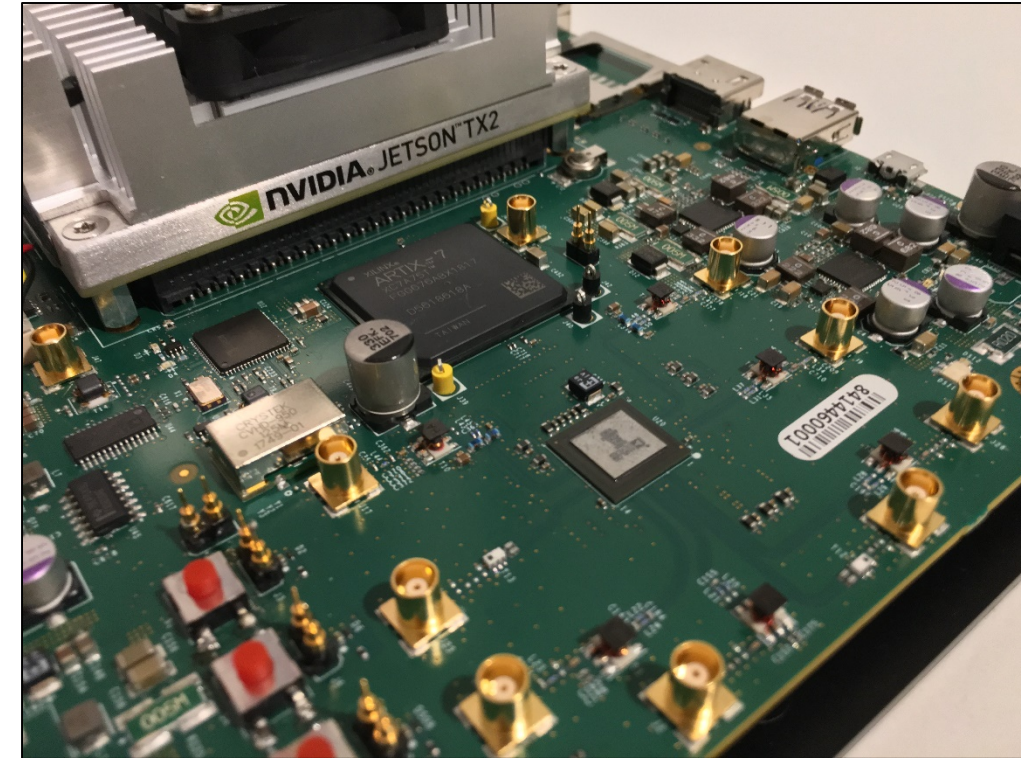
Deepwave Inference Display



Summary

- Deep learning within signal processing is emerging
 - Algorithms may be applied to signal's data content or signal itself
- High bandwidth requirements driving edge solutions
- Deepwave developed AIR-T
 - Edge-compute inference engine with MIMO transceiver
 - FPGA, CPU, GPU
- Announced released of GR-Wavelearner
 - Open source inference engine for signal processing
 - Available now on our GitHub page
- Benchmarking analysis demonstrates AIR-T with GR-Wavelearner capable of signal classification inference at 200 MSPS real-time data rates
 - Improvements likely in future release

More info at www.deepwavedigital.com/sdr





DEEPWAVE
— DIGITAL