Deepwave Digital

making sense of signals

John D. Ferguson, Ph.D.

President / CEO

info@deepwavedigital.com

March 28, 2018

Deep Learning and Radio Frequency (RF) Systems

Deep Learning is Emerging

Cyber

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

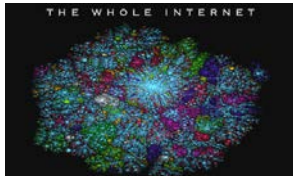
Medicine

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

Autonomy

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

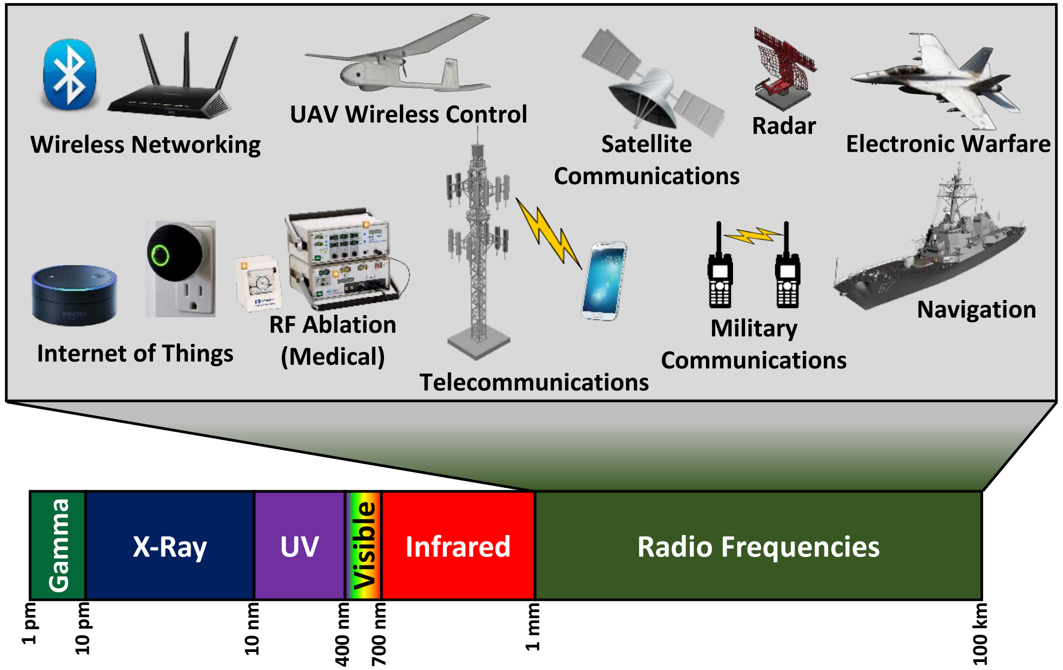
Internet



- 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



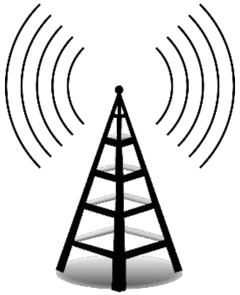
Deep learning technology enabled and accelerated by GPU processors

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

Why Use AI in Wireless Technology

Reduce Human Capital

with AI analytics



Increase Platform Reliability

with reduced down-time

Increase Cyber Security

with wireless network monitoring

AI will increase wireless system reliability and security while simultaneously reducing human capital and cost

Why Has It Not Been Addressed

Bandwidth Limitations

remote processing not possible

Limited Compute Resources

at field site

Complicated Software

for RF and AI independently

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

- No RF systems exist with integrated AI computational processors

- Disjointed software
- Difficult to program and understand

Datalink Bandwidth Limitations

- Raw RF signal must be digitized to apply deep learning
  - Nyquist says two samples per Hertz required to reconstruct the signal
  - Each sample is 16 bits (unpacked)

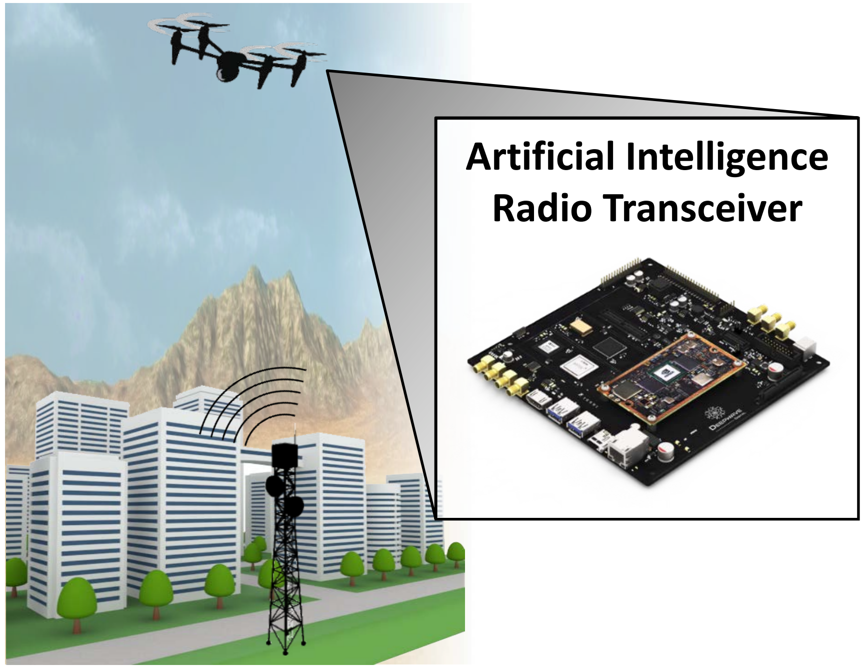
Frequency Band	Bandwidth (MHz)	Data Rate (Mbps)
FM Radio (1 channel)	0.2	6
FM Radio (all channels)	20	640
ISM (915)	26	832
ISM (2.4)	100	3,200
Automotive Radar (64)	500	16,000

- Sending raw digital signal to remote data center is unfeasible
  - Datalinks and fiber optic connections already primary resources for signal's data content
  - Remote locations have limited connectivity
- Reasonable solution is to move the computational engine to the edge

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

Simplification Layer

Deepwave IP Cores

User Developed Applications

Machine Learning

Signal Processing

Industry Leading Software Tools

Hardware Controls

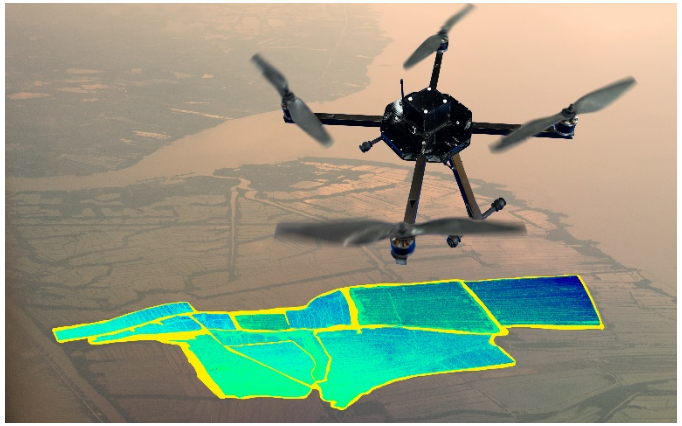
RF and Wireless Hardware

Outline

- Introduction
- Deep Learning in RF Systems
- Deepwave Digital Technology
- Example Signal Detection and Classification
- Training and Inference of Classifier
- Deploying a Deep Learning RF Systems

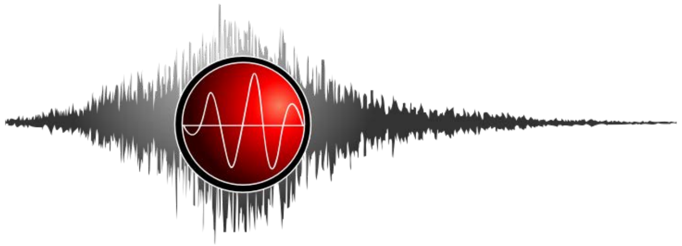
Deep Learning Comparison

Image and Video



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

Audio and Language



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

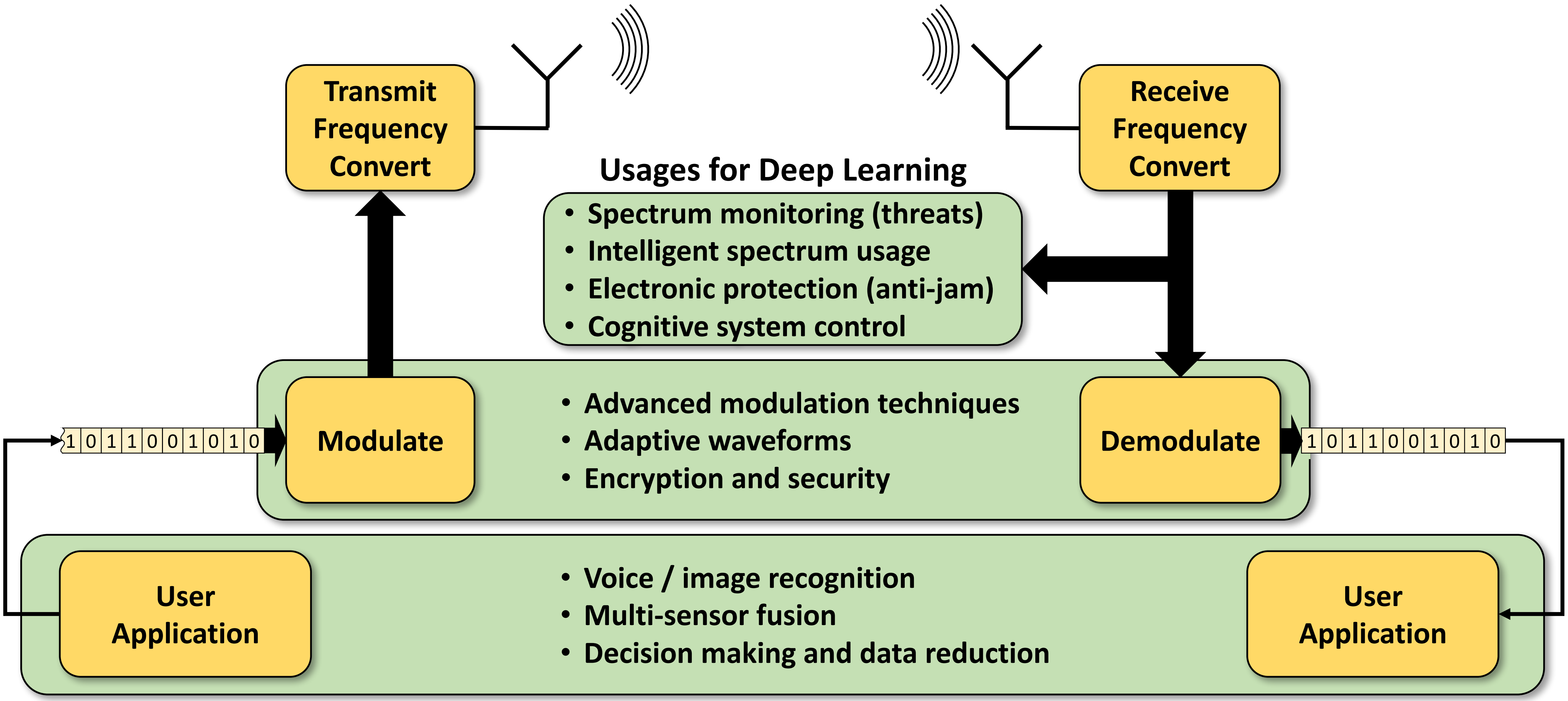
Systems and Signals

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

Existing deep learning potentially adaptable to systems and signals

- **Must contend with complex data types**

Intelligent Radio Frequency (IRF) Systems



Deep Learning Processors for RF Systems: Training

	Pros	Cons
CPU	• Supported by frameworks	• Slower than GPU • Fewer software architectures
GPU	• Most utilized • Highly parallel and adaptable • Good throughput vs. power req.	• Overall power consumption
FPGA	Not widely utilized, not well suited (yet)	
ASIC	Not widely utilized or well suited	

Deep Learning Processors for RF Systems: Inference

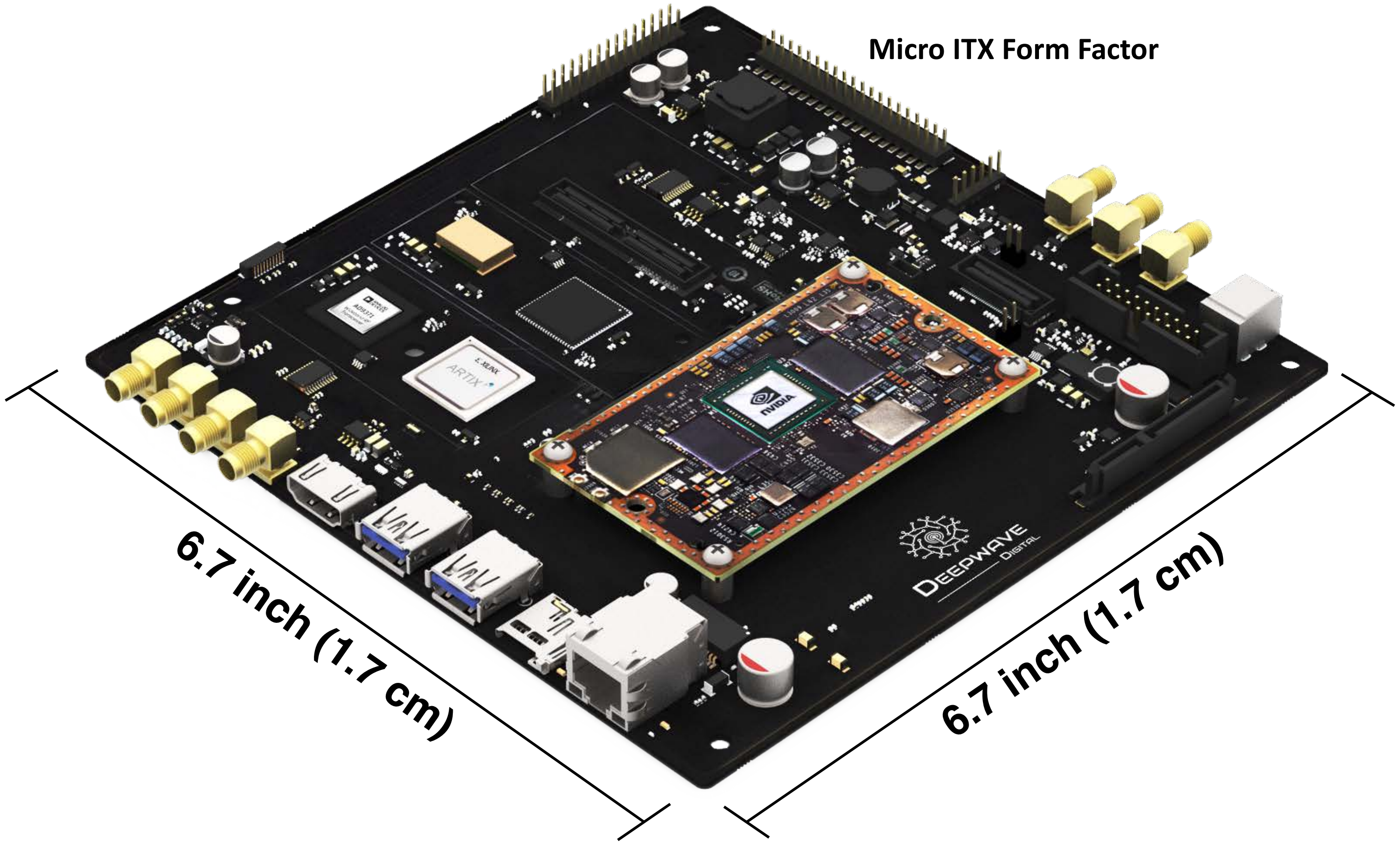
	Pros	Cons
CPU	• Adaptable architecture • Software programmable • Widely utilized in software defined radios	• Low parallelism • Medium power requirements
GPU	• Adaptable architecture • High throughput • Software programmable	• Medium to high power requirements • Not well integrated into RF systems
FPGA	• Power efficient • Somewhat reprogrammable	• Long development time • Specialty expertise required
ASIC	• Extremely power efficient • Highly reliable	• Not adaptable • Long development time • Specialty expertise required

Outline

- Introduction
- Deep Learning in RF Systems
- Deepwave Digital Technology
- Example Signal Detection and Classification
- Training and Inference of Classifier
- Deploying a Deep Learning RF Systems

Artificial Intelligence Radio Transceiver (AIR-T)

AIR-T

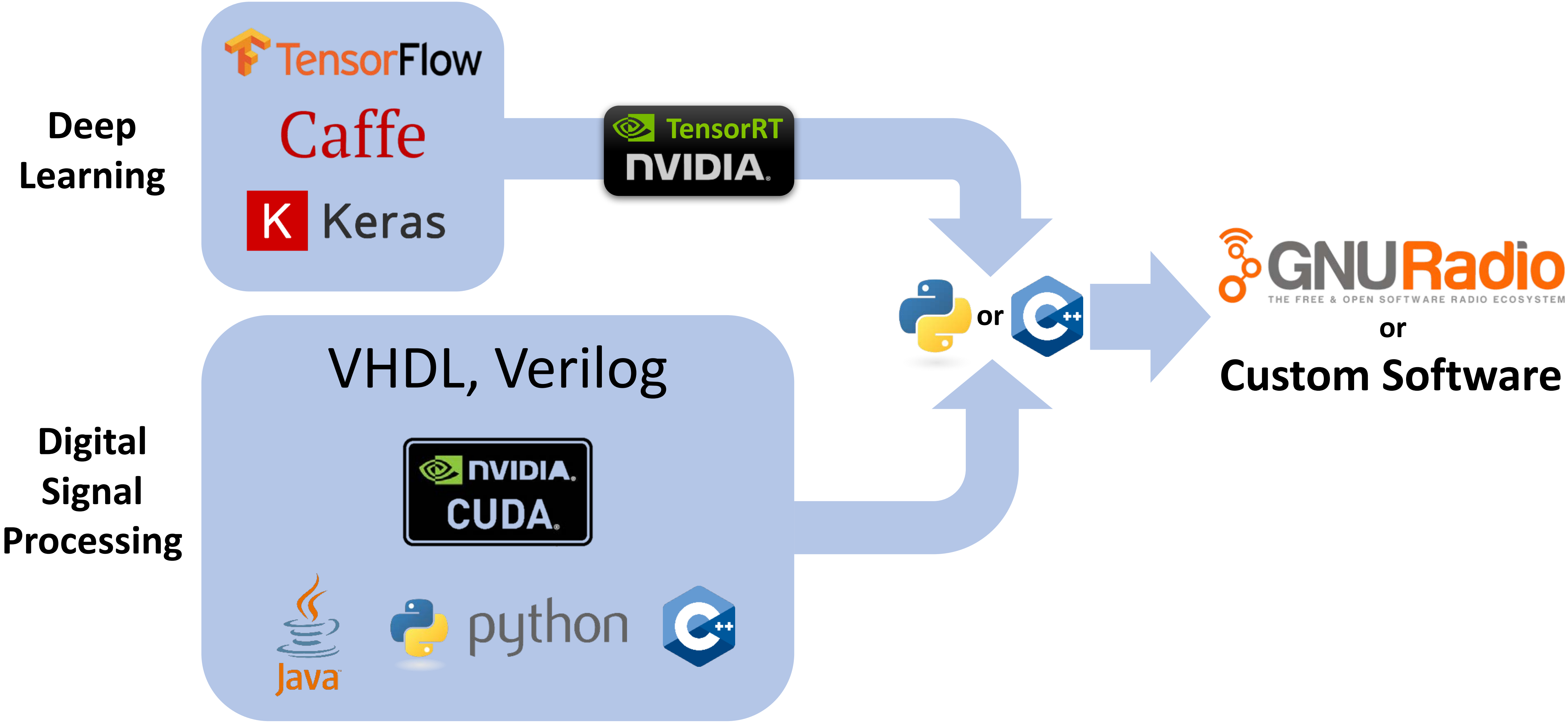


Artistic rendering. Exact design may slightly differ

Specifications

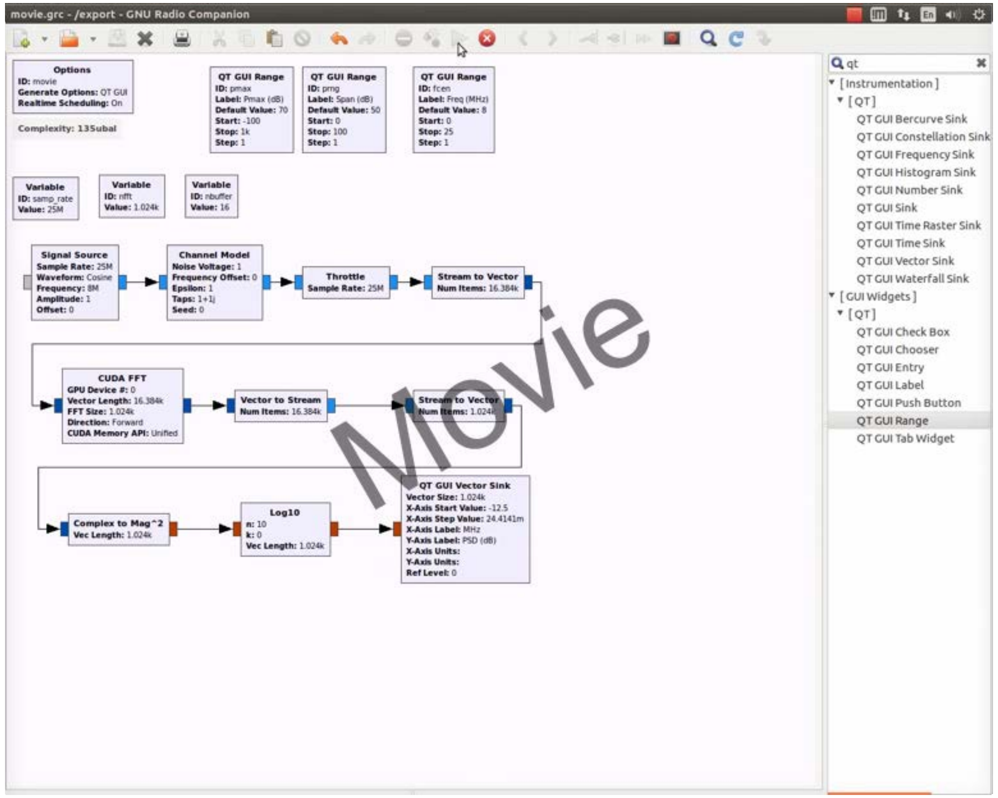
- **Dual Channel Transceiver**
  - 300 MHz to 6 GHz
  - 100 MHz bandwidth per Rx channel
  - 250 MHz bandwidth per Tx channel
- **Digital Signal / Deep Learning Processors**
  - Xilinx Artix 7 FPGA
  - ARM Cortex-A57 (quad-core)
  - Denver2 (dual core)
  - Nvidia Pascal 256 Core GPU
  - Shared CPU/CPU memory
- **Connectivity**
  - 1 PPS / 10 MHz for GPS Synchronization
  - HDMI, USB 2.0/3.0, SATA
  - Ethernet, WiFi, Bluetooth
- **Dual Power Mode (22 / 14W)**

Simplified Programming



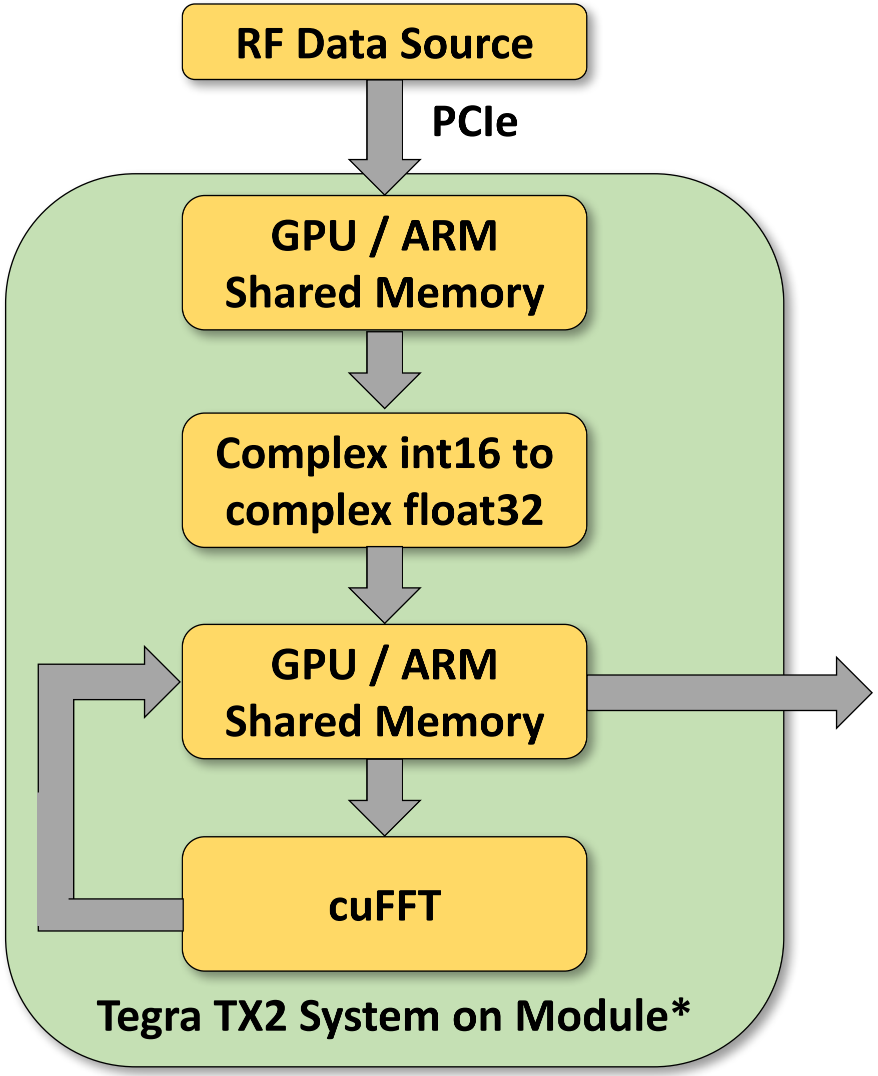
GNU Radio – Software Defined Radio (SDR) Framework

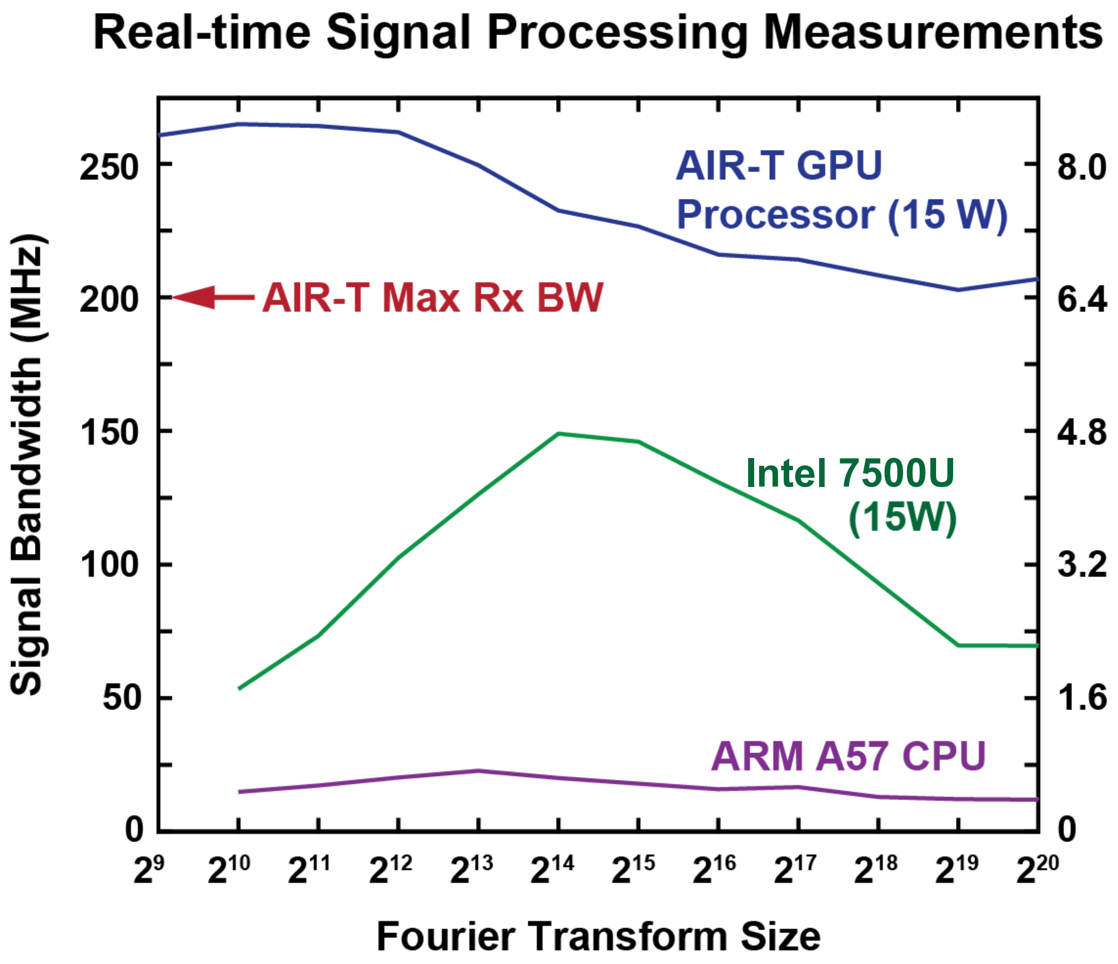
- **Popular open source SDR toolkit:**
  - RF Hardware optional
  - Can run full software simulations
- **Python API**
  - C++ under the hood
- **Easily create DSP algorithms**
  - Custom user blocks
- **Primarily uses CPU**
  - Advanced parallel instructions
  - Recent development: RFNoC for FPGA processing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



GNURadio
THE FREE & OPEN SOFTWARE RADIO ECOSYSTEM

AIR-T Initial Performance Testing





* GNU Radio excluded from cuFFT benchmark

Current GNU Radio Limitations for GPU Processing

- **GNU Radio handles memory management**
  - Cannot currently tell it about existing memory buffers
  - Must allocate special CUDA memory (even on Jetson TX2)
  - Requires memcpy() each time GNU Radio block operates
    - Extra copy each direction

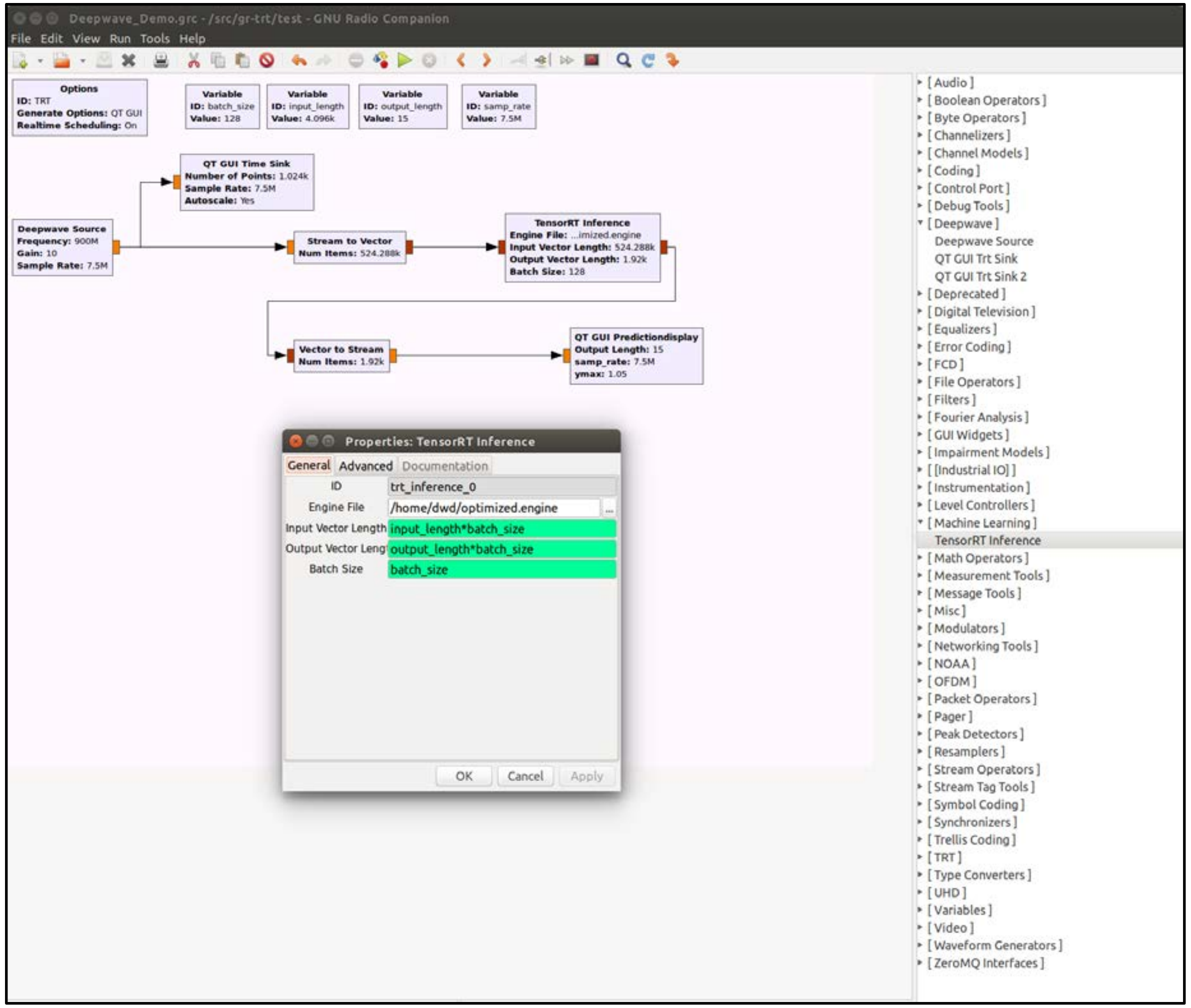
- **GNU Radio has open feature request to support custom buffer allocators***
  - Deepwave willing to collaborate with GNU Radio open source project to advance issue

* Issue #950 on Github

Inference using GNU Radio and TensorRT

- TensorRT chosen as initial inference library for AIR-T
  - Optimized inference for Nvidia based hardware
  - Significant speedups over TensorFlow on Jetson TX2 for image processing
  - Native support for:
    - TensorFlow, Caffe, other frameworks

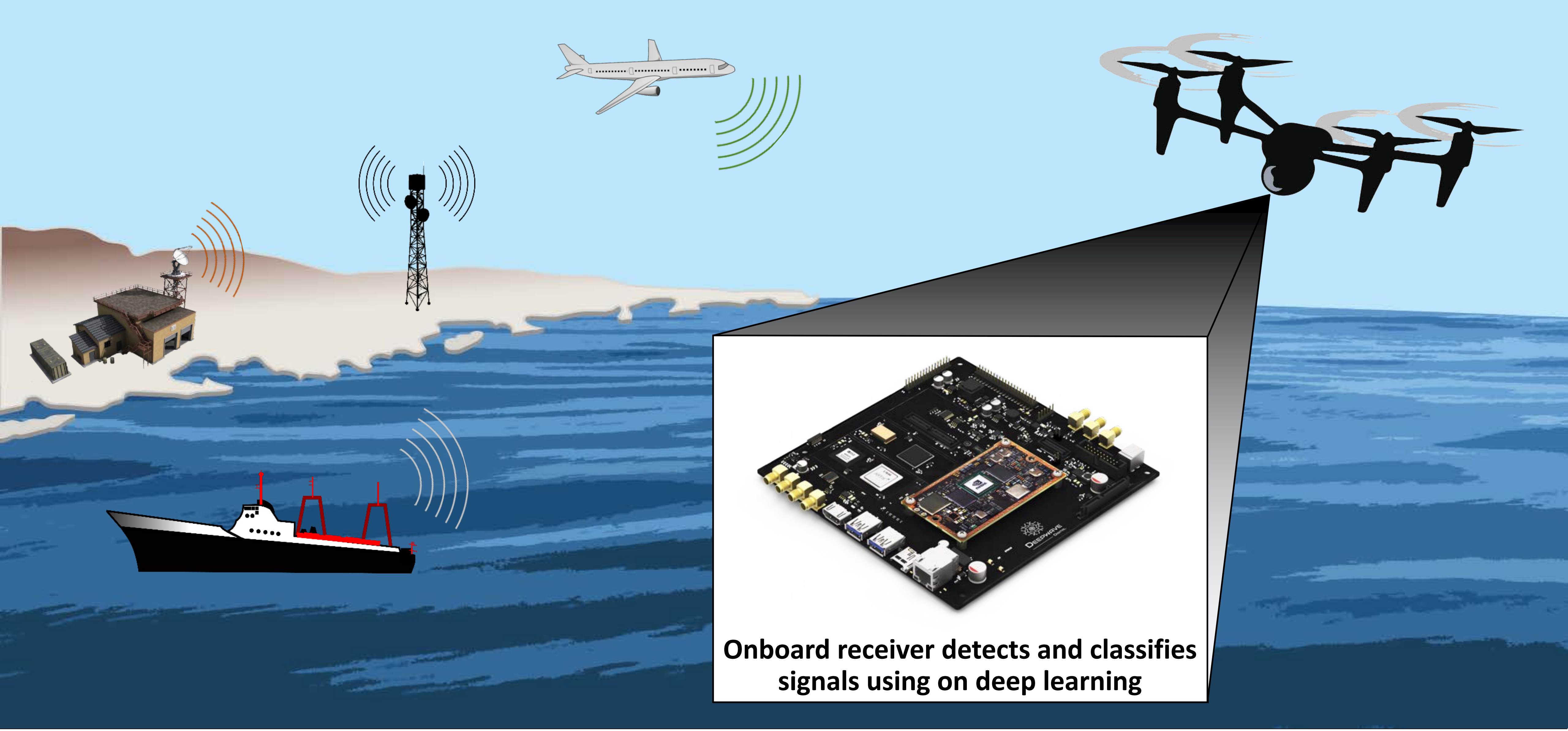
- Deepwave working on GNU Radio OOT Module (gr-trt) to execute TensorRT inference within GNU Radio



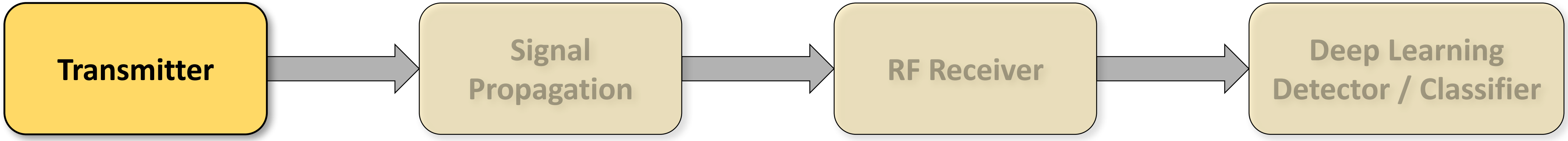
Outline

- Introduction
- Deep Learning in RF Systems
- Deepwave Digital Technology
- Example Signal Detection and Classification
- Training and Inference of Classifier
- Deploying a Deep Learning RF Systems

Multi-transmitter Environmental Scenario

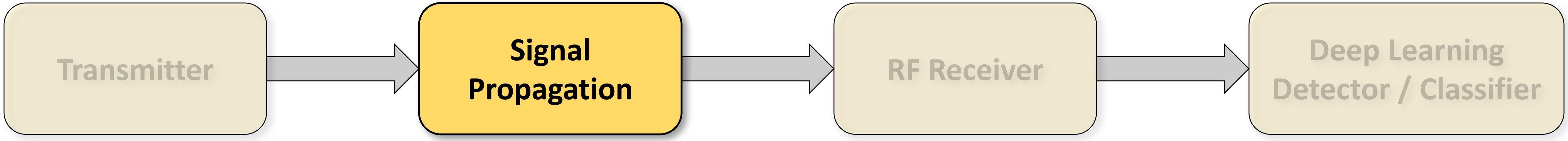


Radar Signal Detector Model: Transmitted Signals



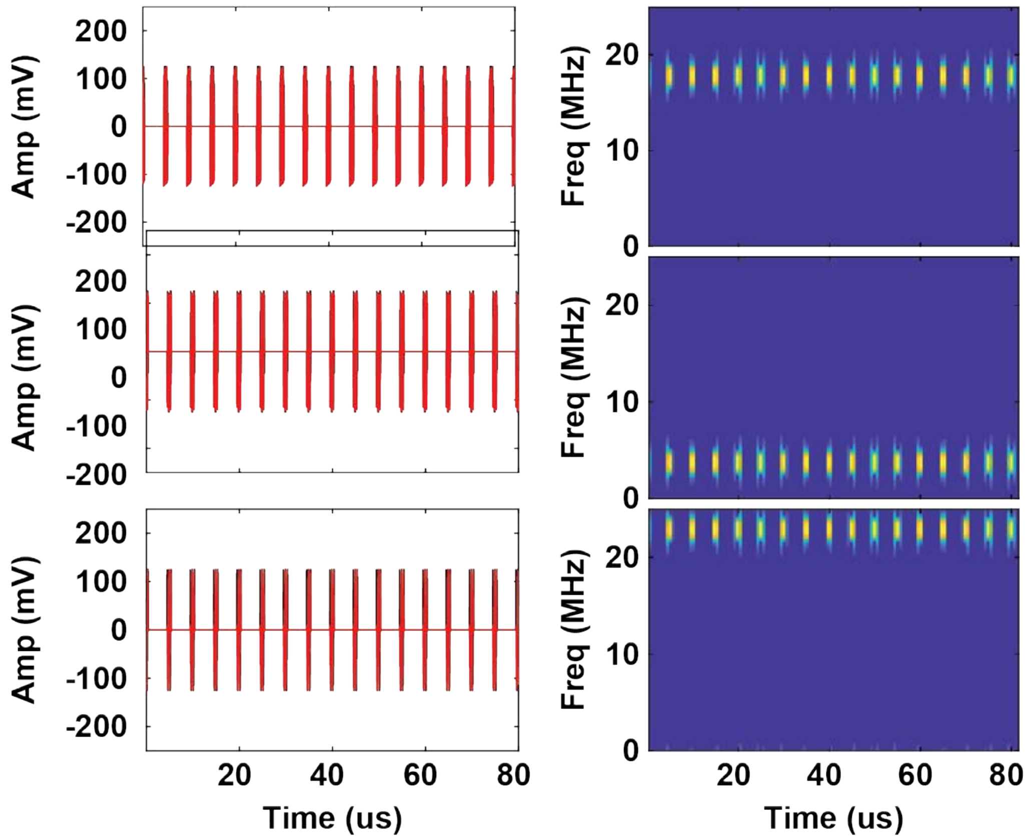
Radar Waveform	Nothing	Interference	Surveillance	Ground (LFM1)	Ground (LFM2)	MTI	Airborne (Med PRF)	Airborne (High PRF)	Ground (Frank Code)	Nautical (Short Range)	Nautical (Long Range)	Nautical (Long Range)	Ground (NLFM1)	Ground (NLFM2)	Ground (NLFM3)
Linear Pulse			X	X	X					X	X	X			
Non-Linear Pulse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

Radar Signal Detector Model: Propagation

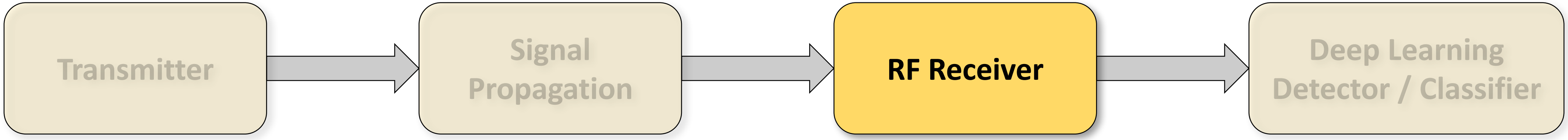


Data Set Generation:

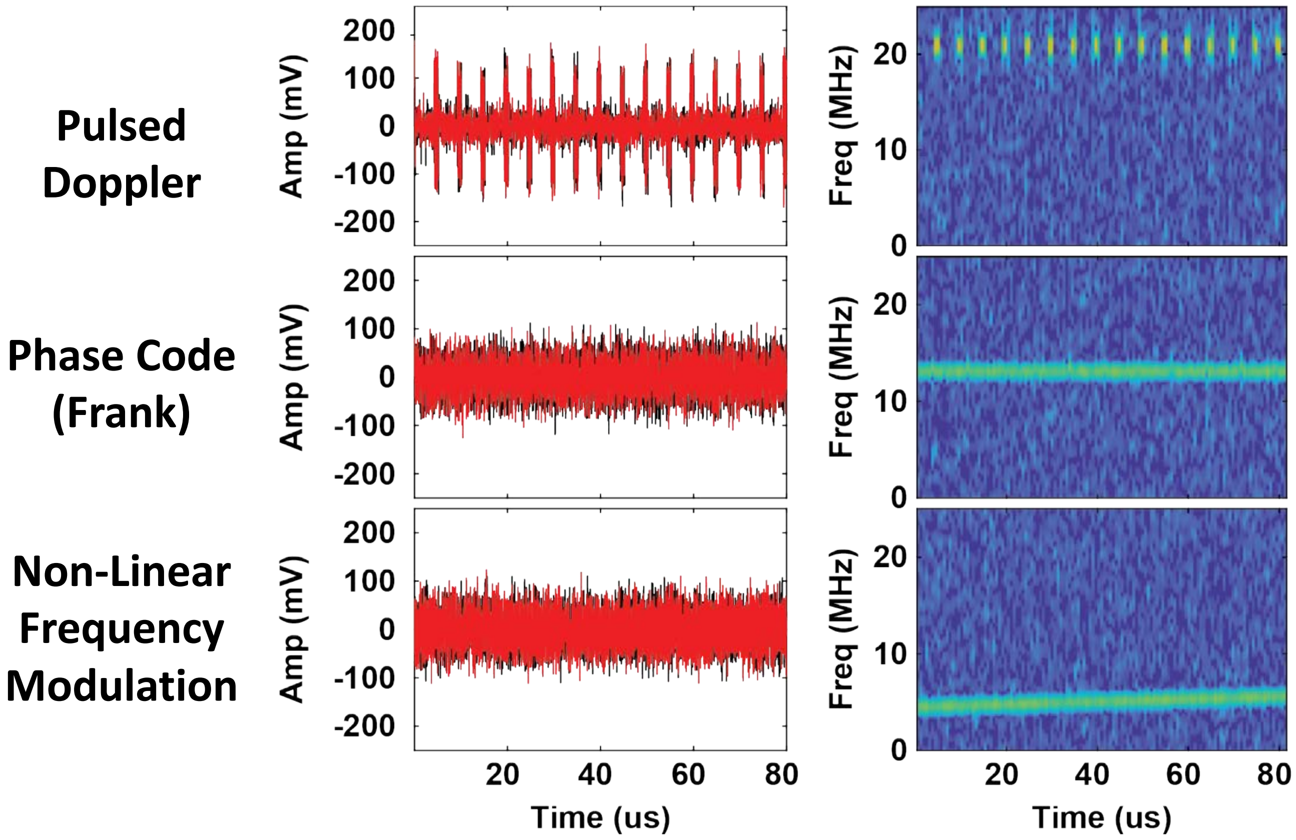
- Random phase shift applied to each signal segment
- Signal frequency changes between coherent processing intervals
- Transmitter range modeled as received signal to noise radios



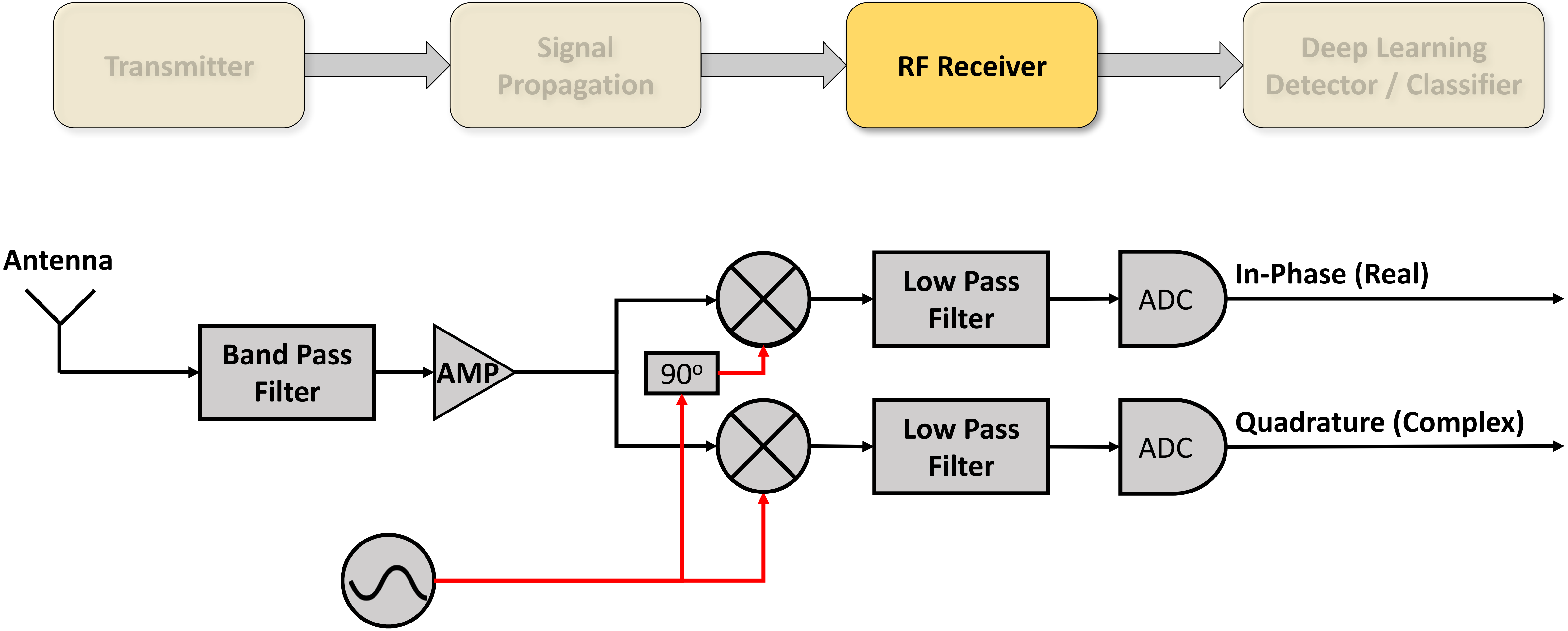
Radar Signal Detector Model: Received Signal



- RF Receiver system:
  - 65 dB gain
  - 5 dB noise figure
  - 14 ADC bits
  - 25 MHz bandwidth
- Hilbert Transformer receiver
  - Complex sample data

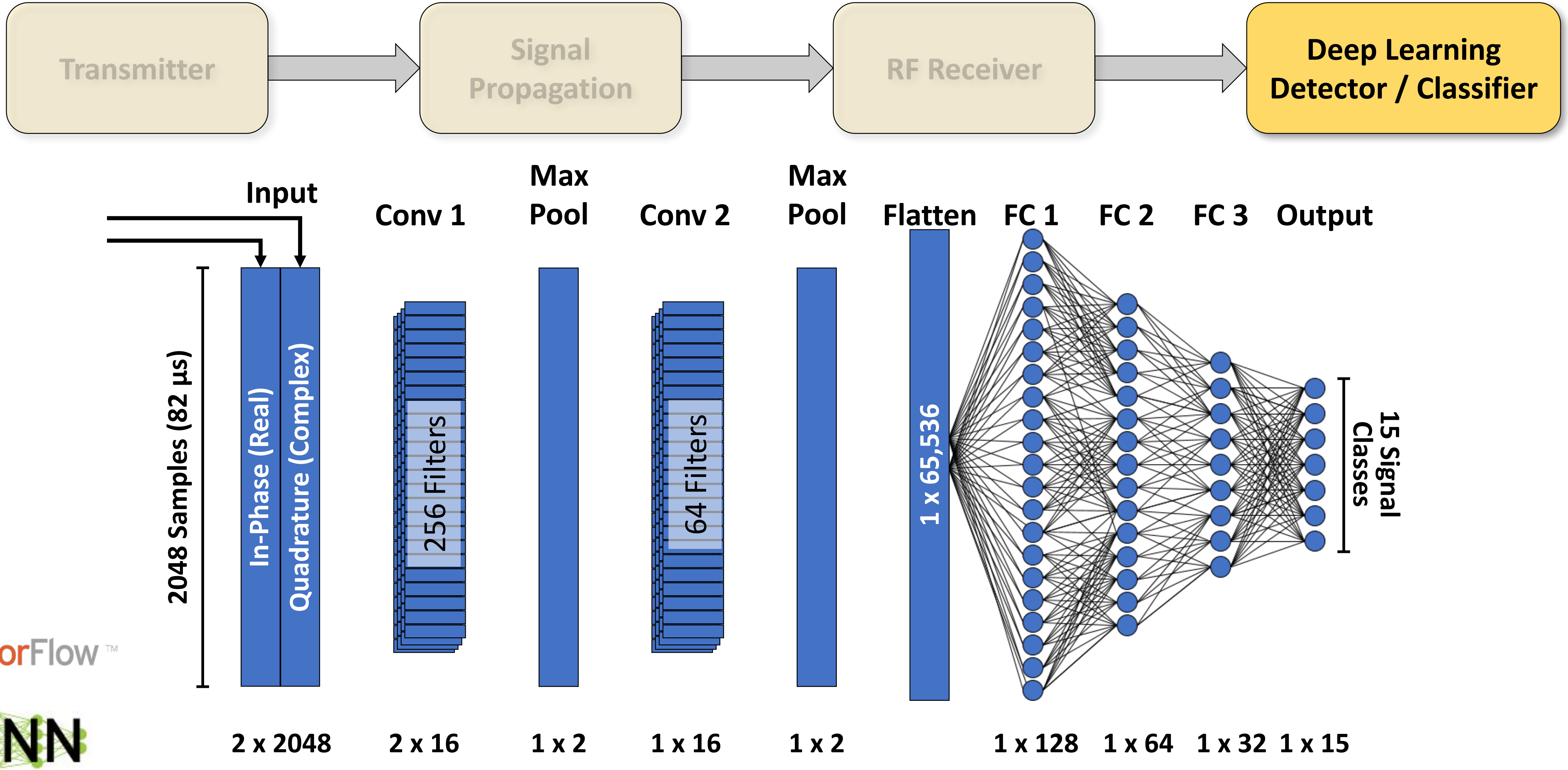


Radar Signal Detector Model: Receiver



- 5 dB Noise Figure
- 65 dB System Gain
- 14 Bits ADC

Radar Signal Detector Model: Example Classifier

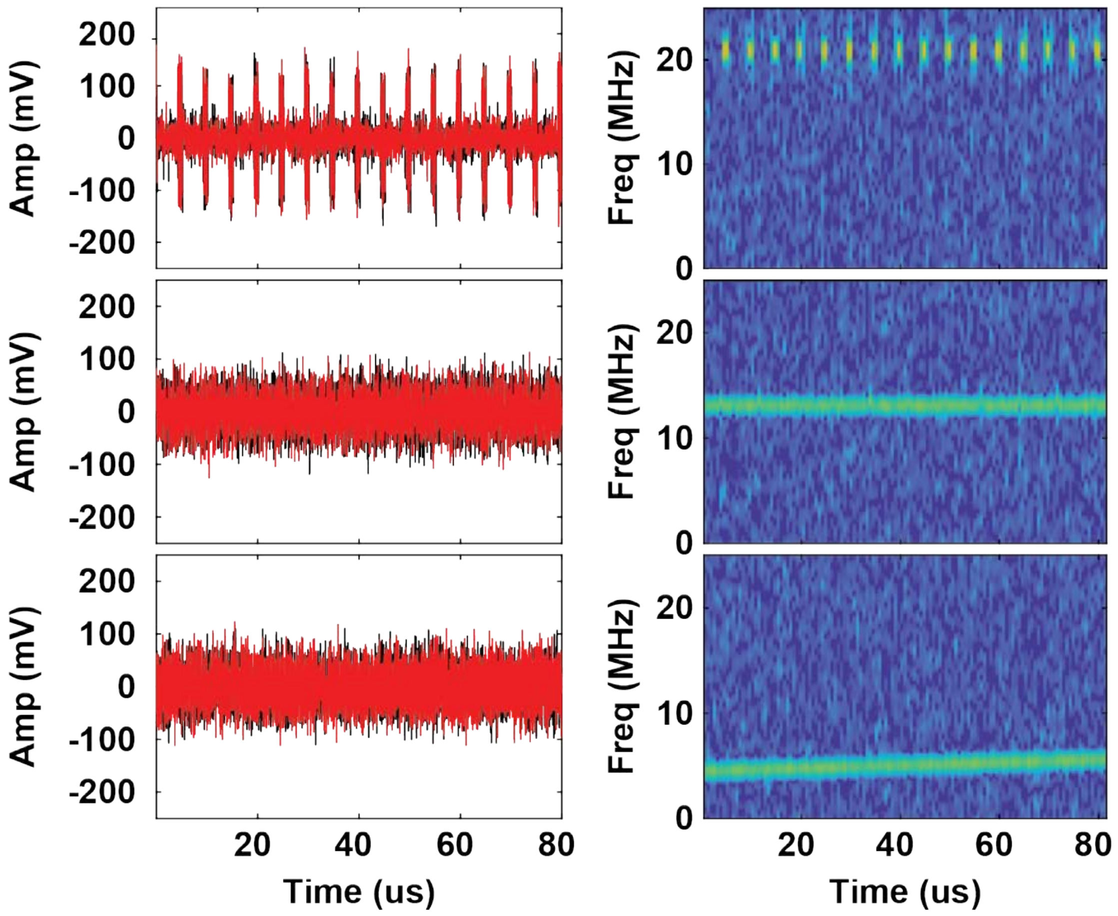


Outline

- Introduction
- Deep Learning in RF Systems
- Deepwave Digital Technology
- Example Signal Detection and Classification
- Training and Inference of Classifier
- Deploying a Deep Learning RF Systems

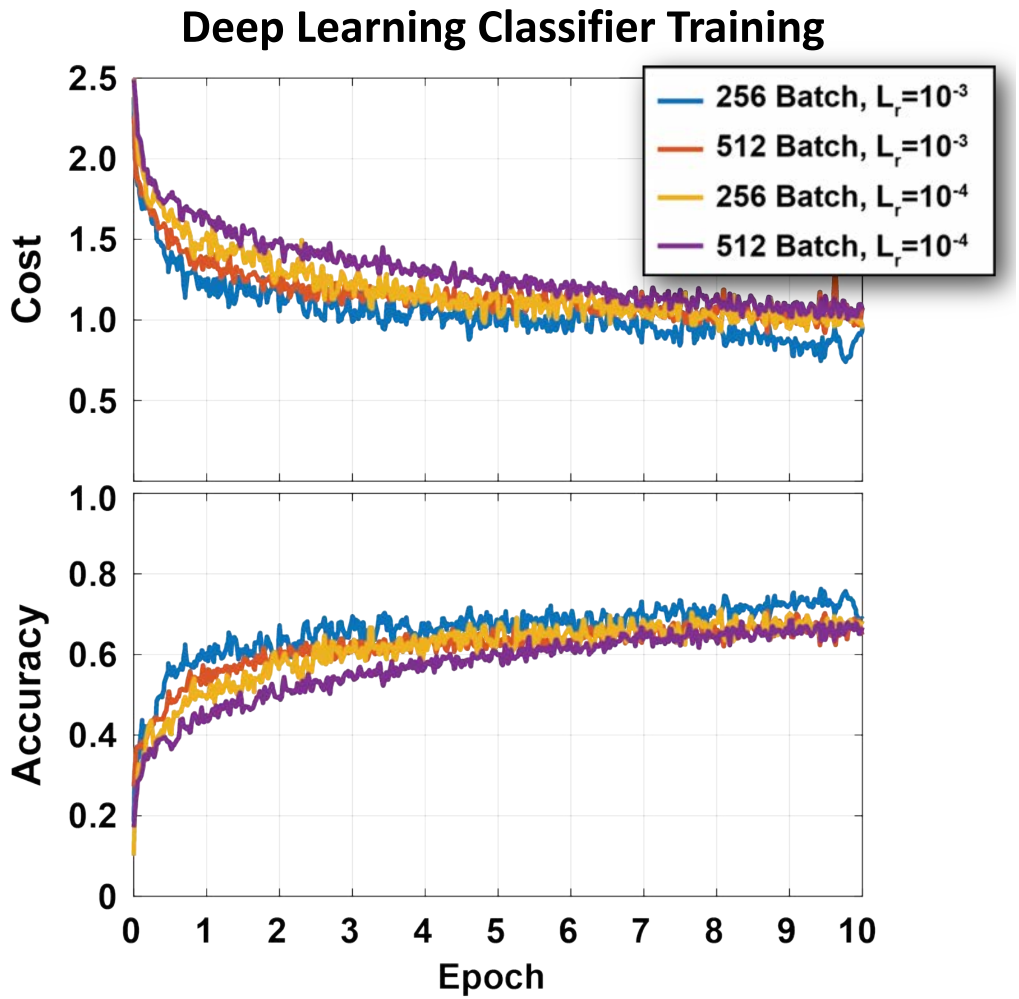
Dataset Overview

- Goal: Develop a deep learning classifier that detects signals below noise floor
  - Requires training on noisy data
- Swept SNR from -35 dB to 20 dB in 1 dB increments
  - 1000 training segments per SNR
  - 500 inference segments per SNR
- Used MATLAB to create data

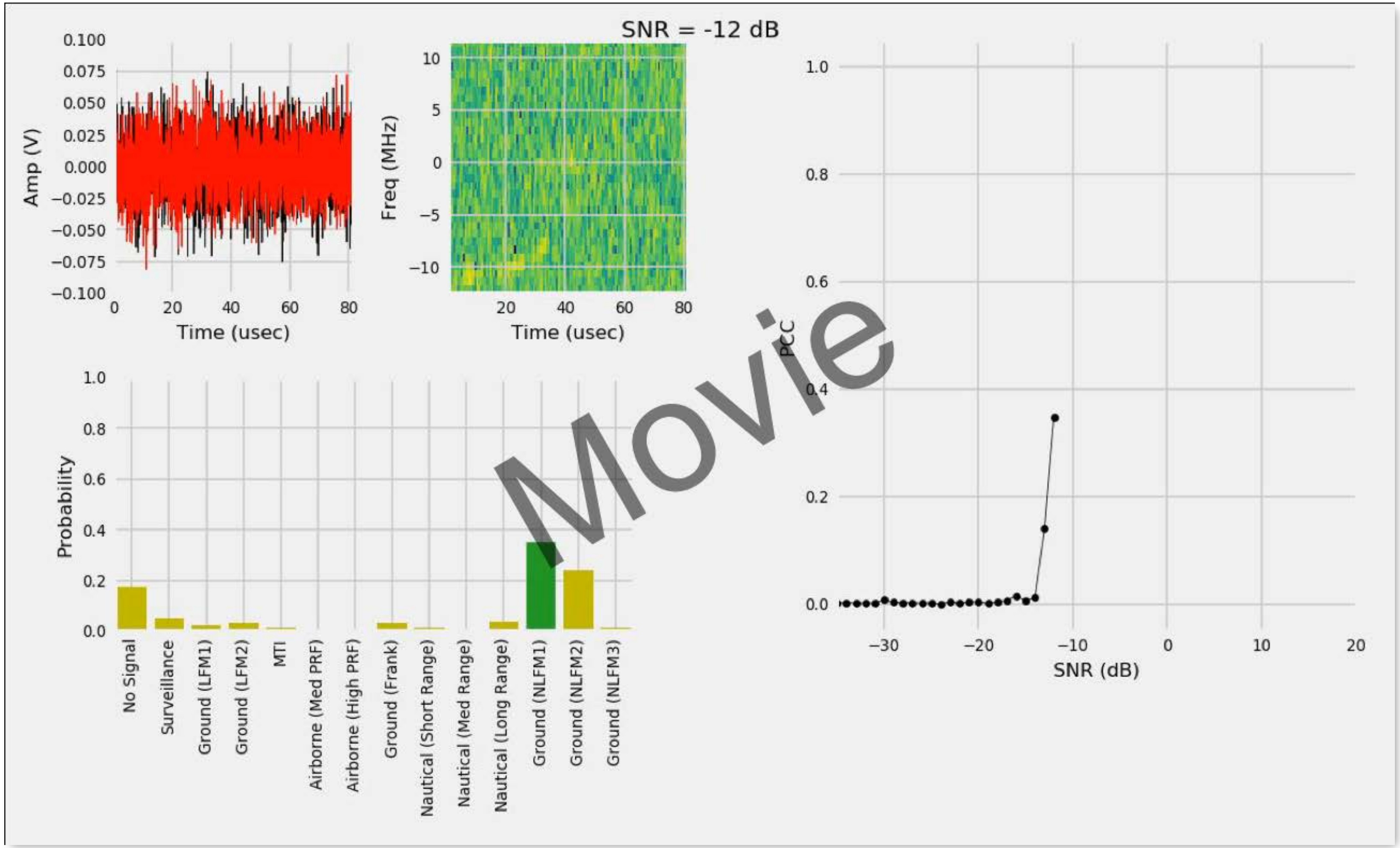


Training Process and Progress

- 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

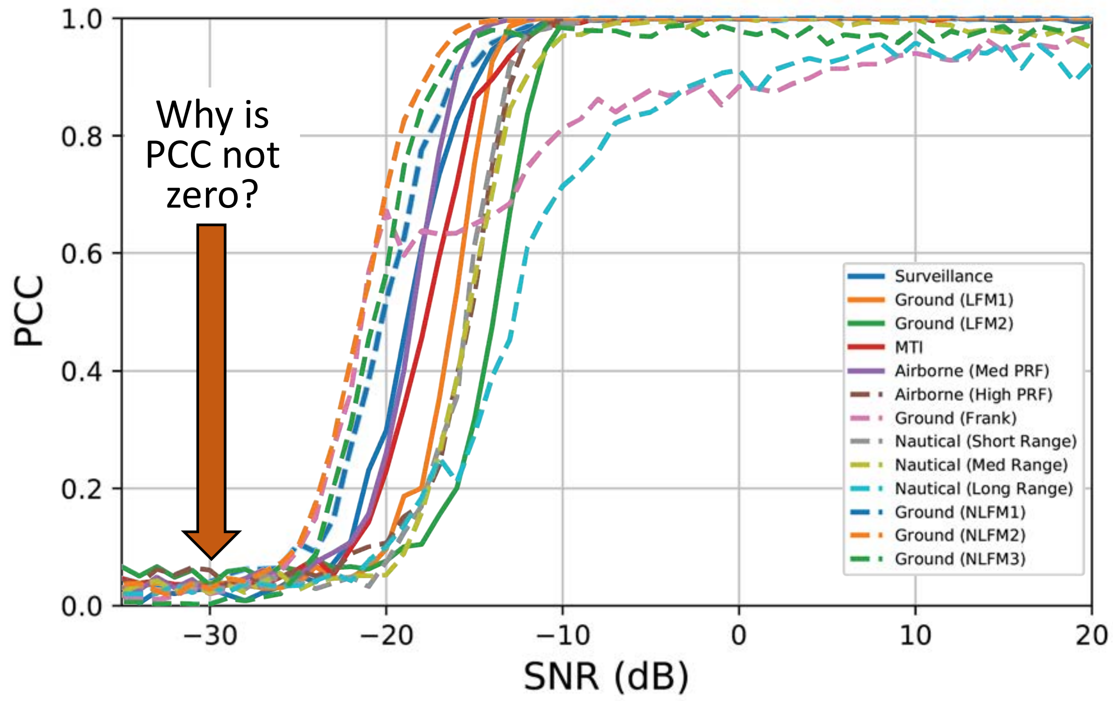


Detecting and Classifying Low Power Signals



Receiver Operating Characteristic (ROC) Curve

Probability of Correct Classification for Various Radars

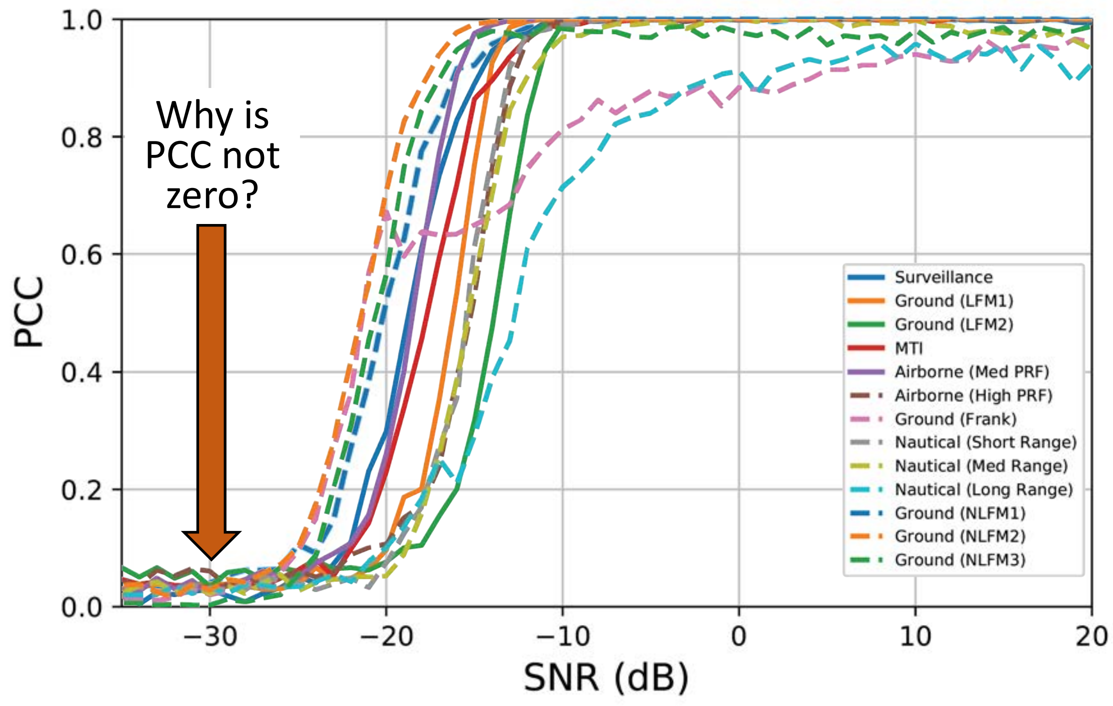


Decibel (dB) Refresher

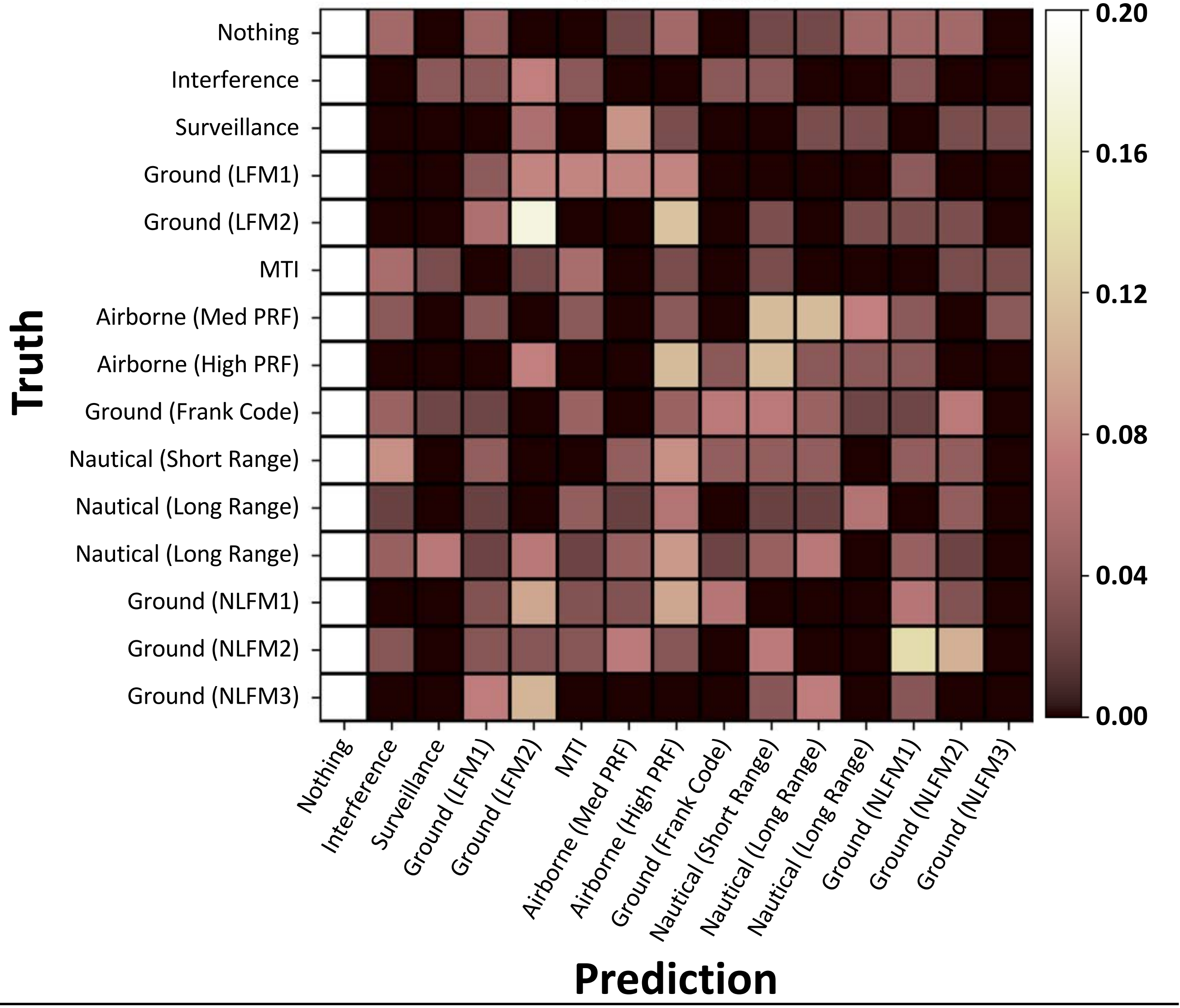
Signal-to-Noise Ratio (dB)	Receiver Noise Power (milliwatts)	Received Signal Power (milliwatts)
20	1	100
10	1	10
0	1	1
-10	1	0.1
-20	1	0.01
-30	1	0.001

Receiver Operating Characteristic (ROC) Curve

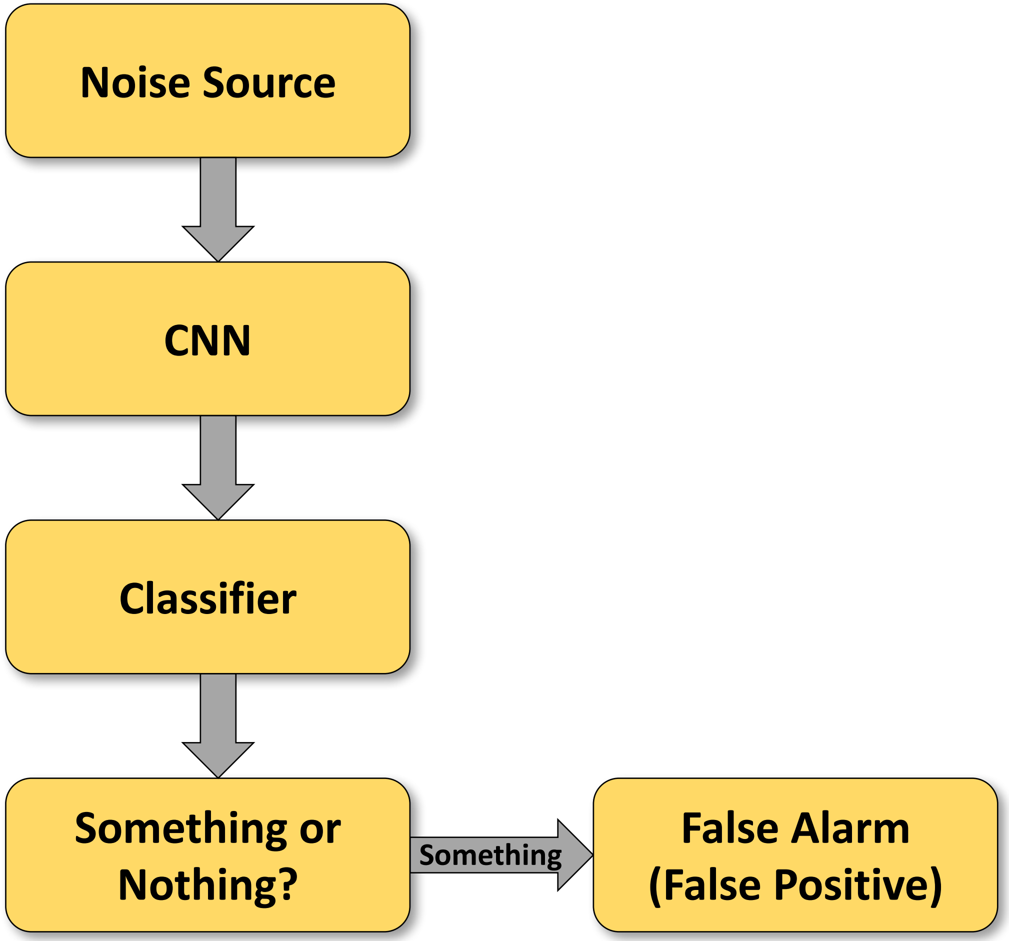
Probability of Correct Classification for Various Radars

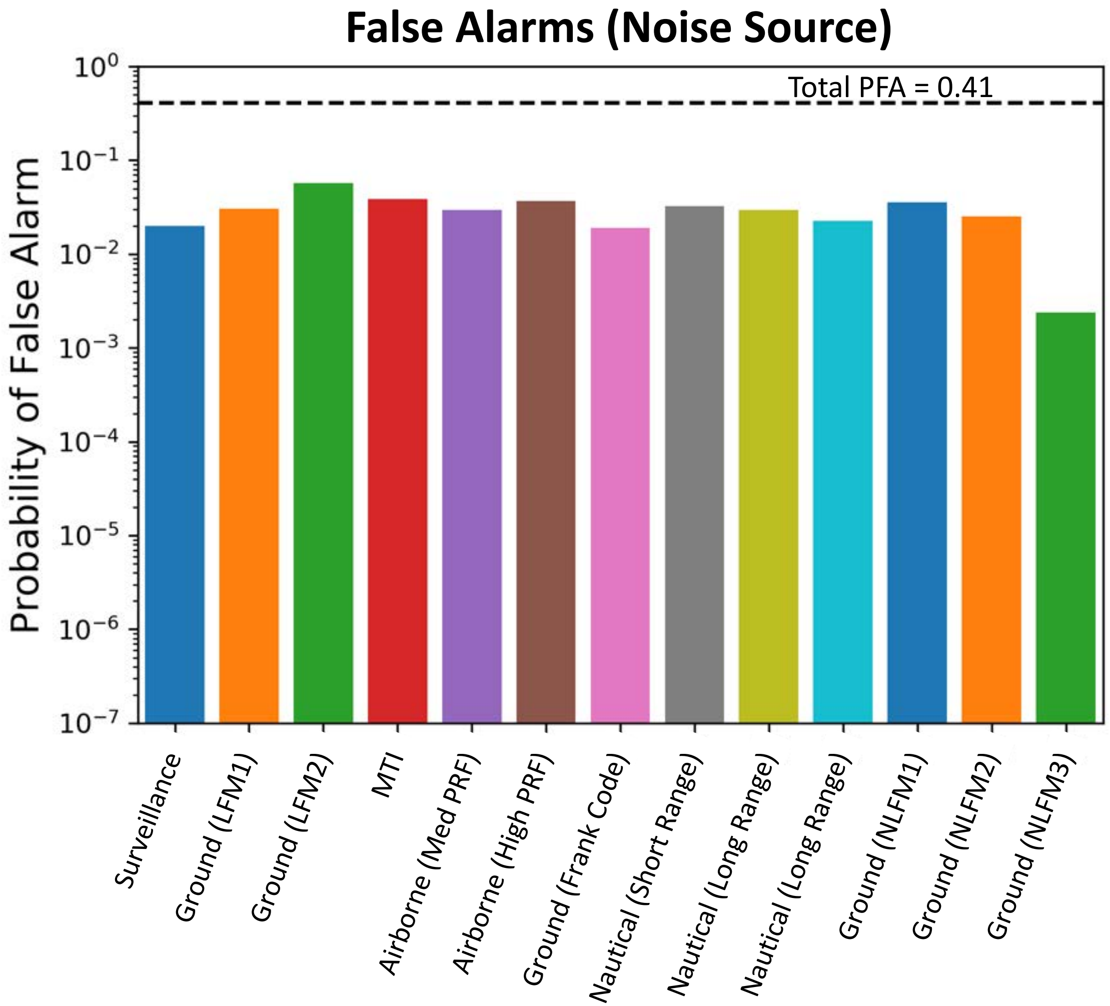


Confusion Matrix (-35 dB SNR)

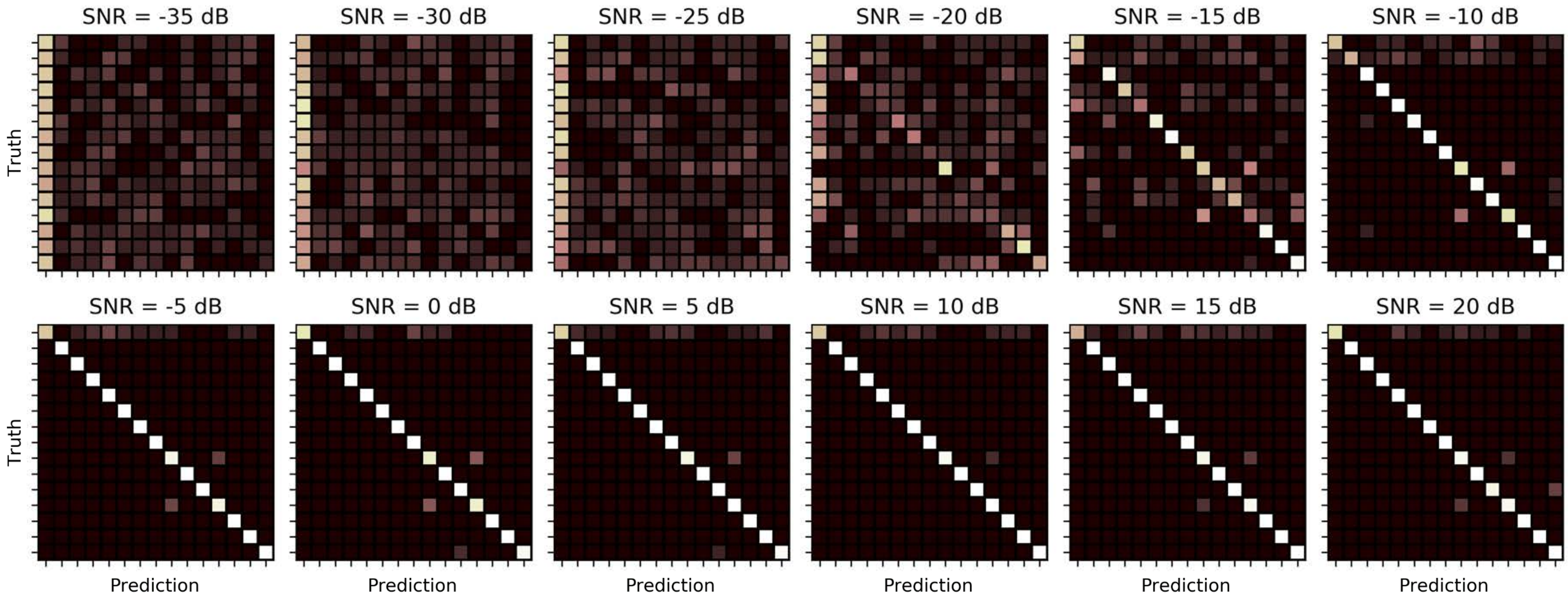


Measuring Probability of False Alarm





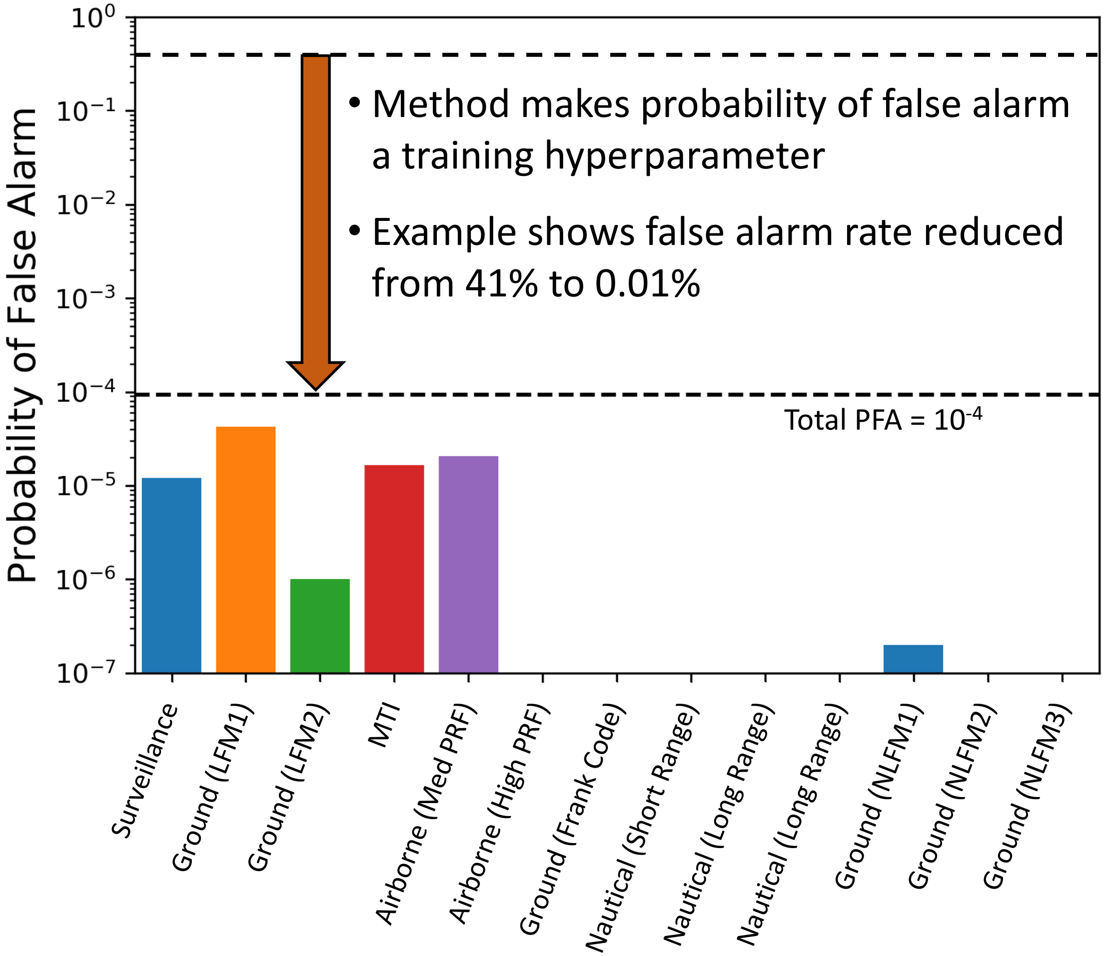
Confusion Matrix and Signal to Noise Ratio



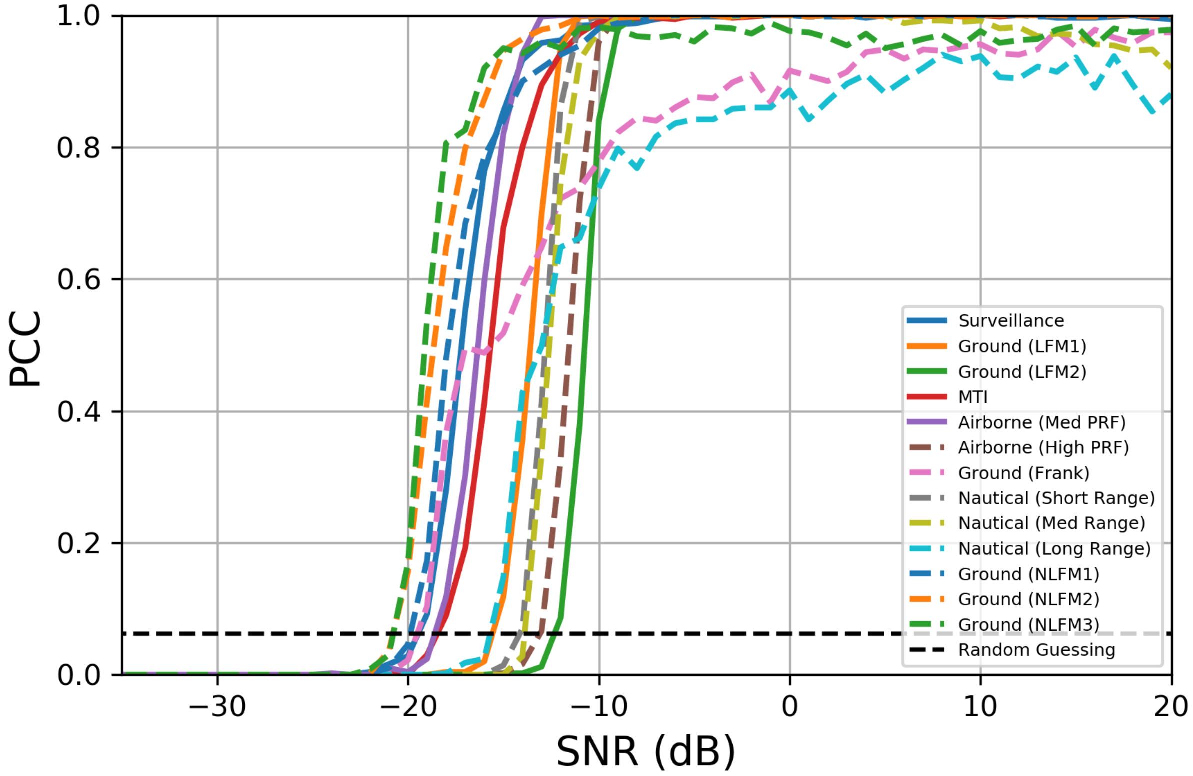
Significant false alarm rate limits algorithm's applicability and creates non-zero probability of correct classification (PCC) at low SNR values

Deepwave Training Method to Reduce False Alarms

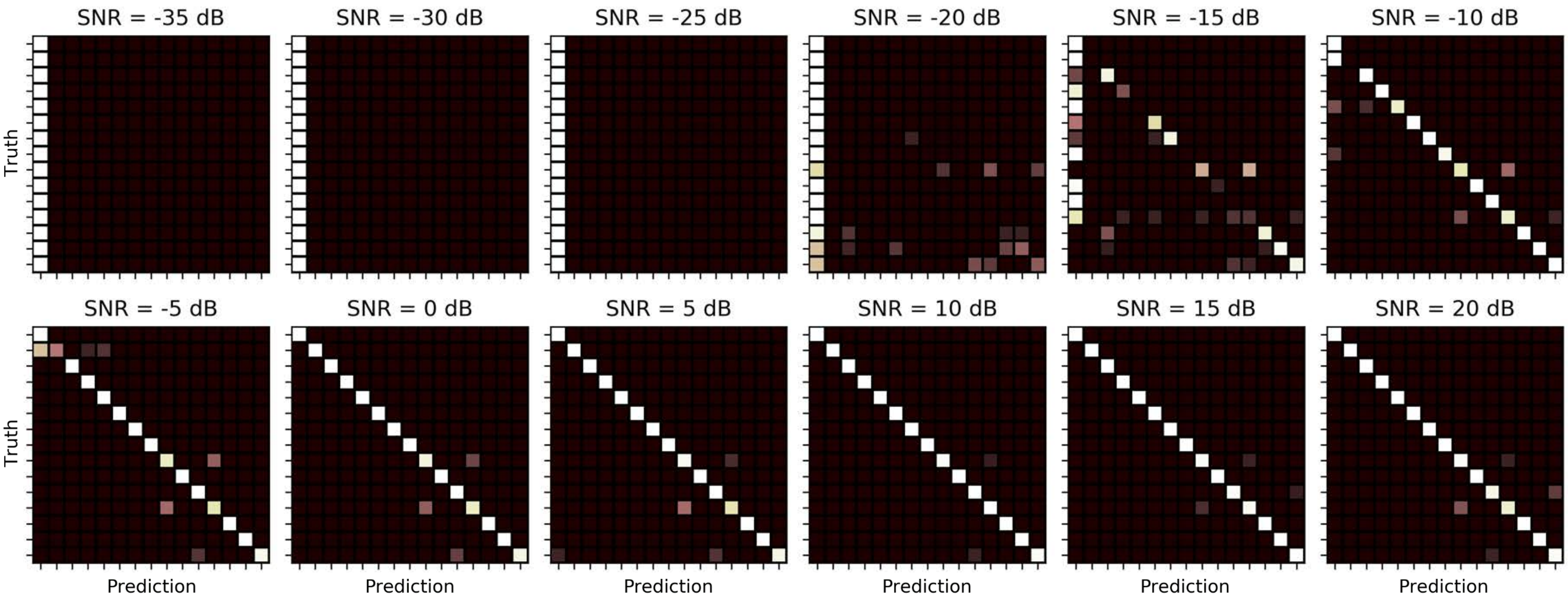
False Alarms (Noise Only)



Probability of Correct Classification for Various Radars



New Inference Confusion Matrix



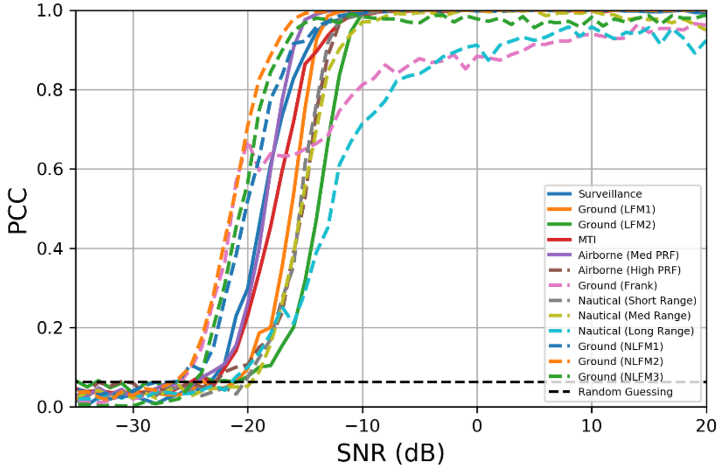
Deepwave training algorithms allows for tunability of false alarm rate

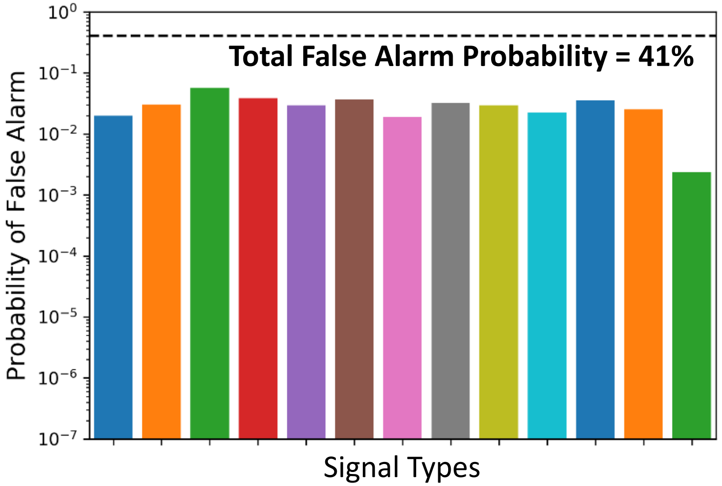
Deepwave Method to Reduce False Alarms

Probability of Detecting and Classifying Signal

Probability of False Alarm (False Positive)

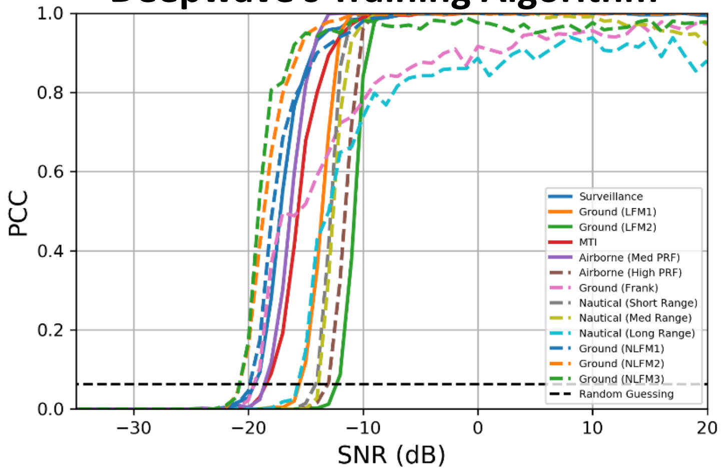
Trained Neural Network

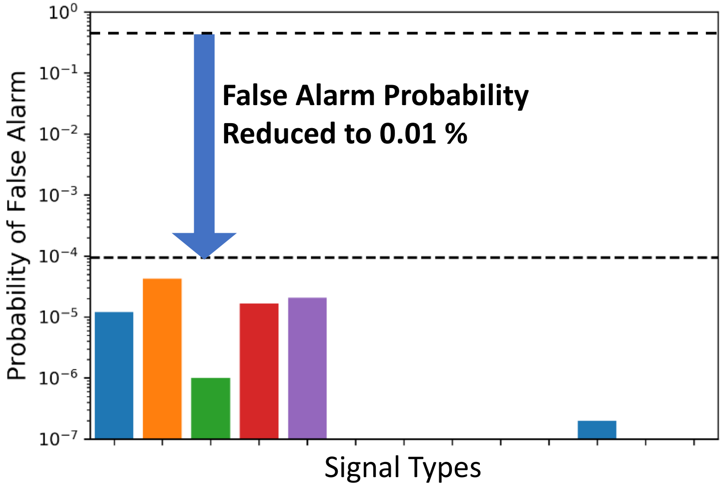




Apply Deepwave's False Alarm Learning Algorithm

Deepwave's Training Algorithm



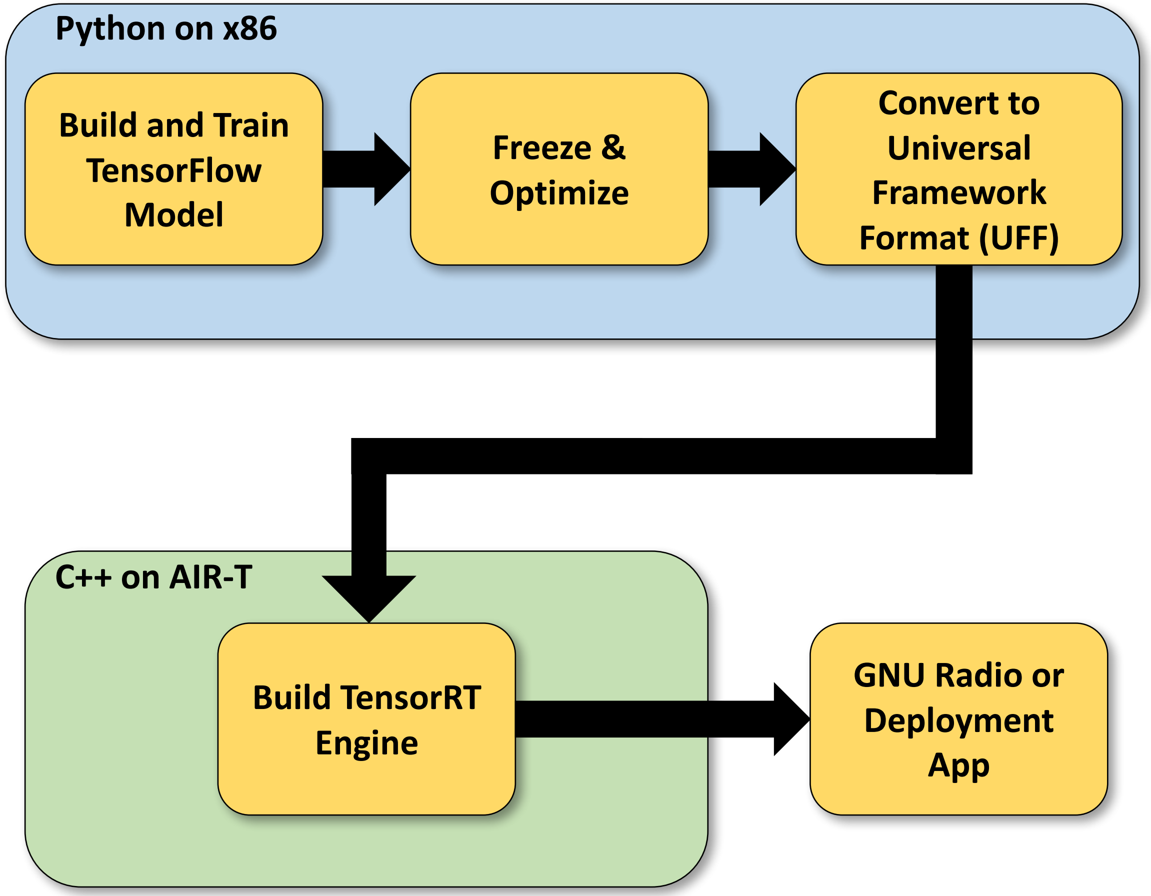


False alarm conscious training method has < 5dB impact on detector sensitivity

Outline

- Introduction
- Deep Learning in RF Systems
- Deepwave Digital Technology
- Example Signal Detection and Classification
- Training and Inference of Classifier
- Deploying a Deep Learning RF Systems

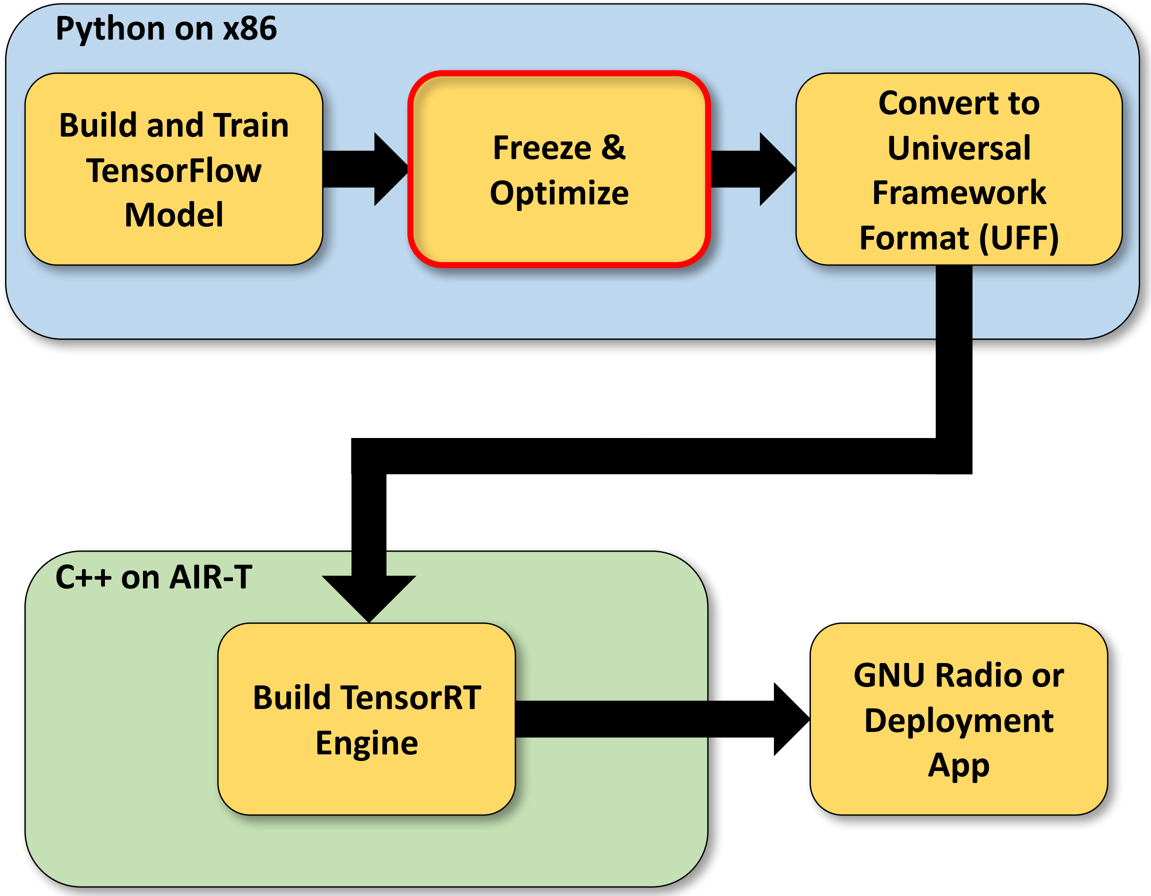
Creating Inference Engine for GNU Radio



Implementation Caveats

- Yesterday's announcement of TensorRT support in native TensorFlow is not included
- TensorRT Python API not supported on ARM
  - Some C++ required to build and run inference
  - Several steps can still be performed in Python, but on an x86 machine
- Inference performance tied to TensorRT kernel selection and optimization
  - RF case is somewhat unique (i.e., we're not processing images)
    - Unique network shape
    - Possible that optimizations have yet to be applied
- GNU Radio does not support float16

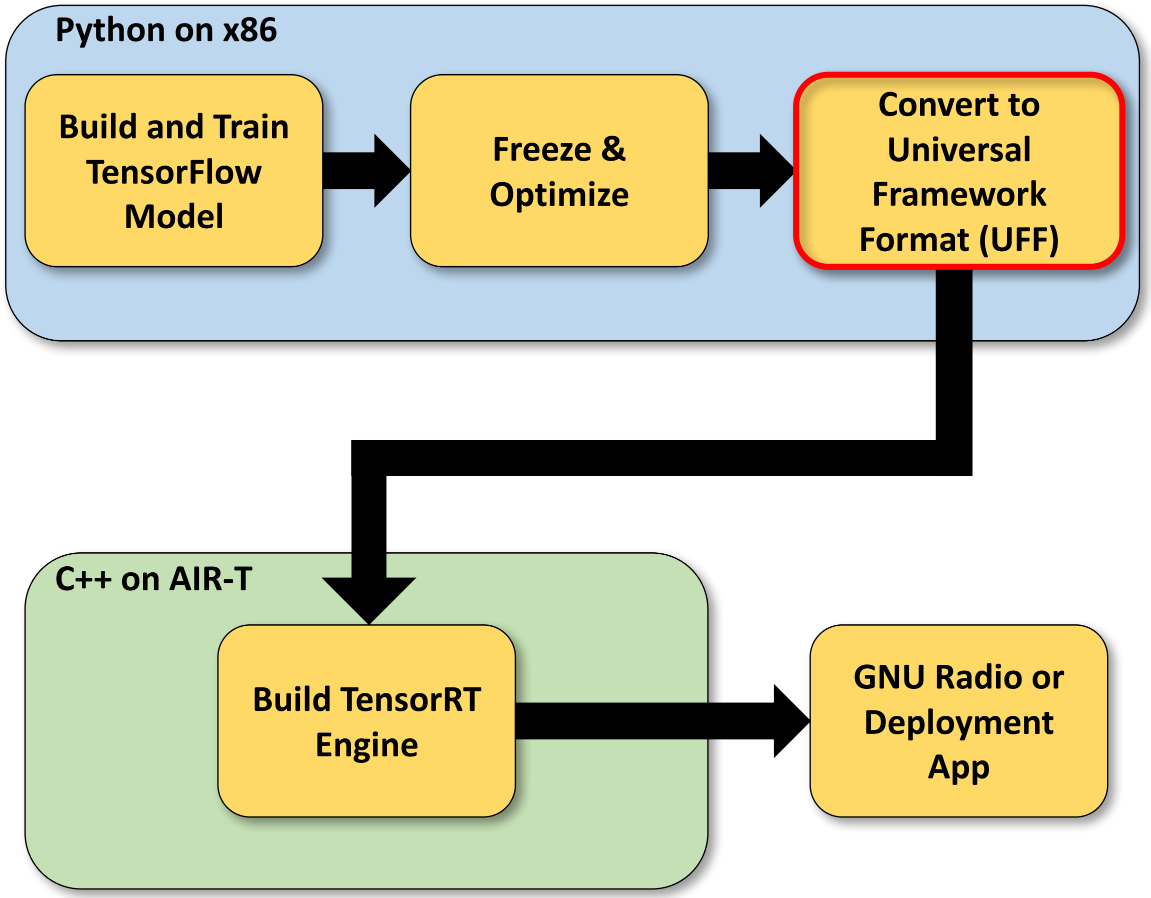
Creating Inference Engine for GNU Radio



Freeze and Optimize Trained Model

```python
import tensorflow as tf
from tensorflow.python.tools import optimize_for_inference_lib

input_model_name = 'somemodel/saved_model'
input_file = input_model_name + '.meta'
input_node_name = 'inputdata'
output_node_name = 'output/networkout'
output_file = 'somefile.pb'

saver = tf.train.import_meta_graph(input_file, clear_devices=True)
graph = tf.get_default_graph()
sess = tf.Session()
saver.restore(sess, input_model_name)

input_graph_def = graph.as_graph_def()

inference_graph_def = optimize_for_inference_lib.optimize_for_inference(
  input_graph_def,
  [input_node_name],
  [output_node_name],
  tf.float32.as_datatype_enum)

with tf.gfile.GFile(output_file, 'wb') as f:
    f.write(inference_graph_def.SerializeToString())
sess.close()
```

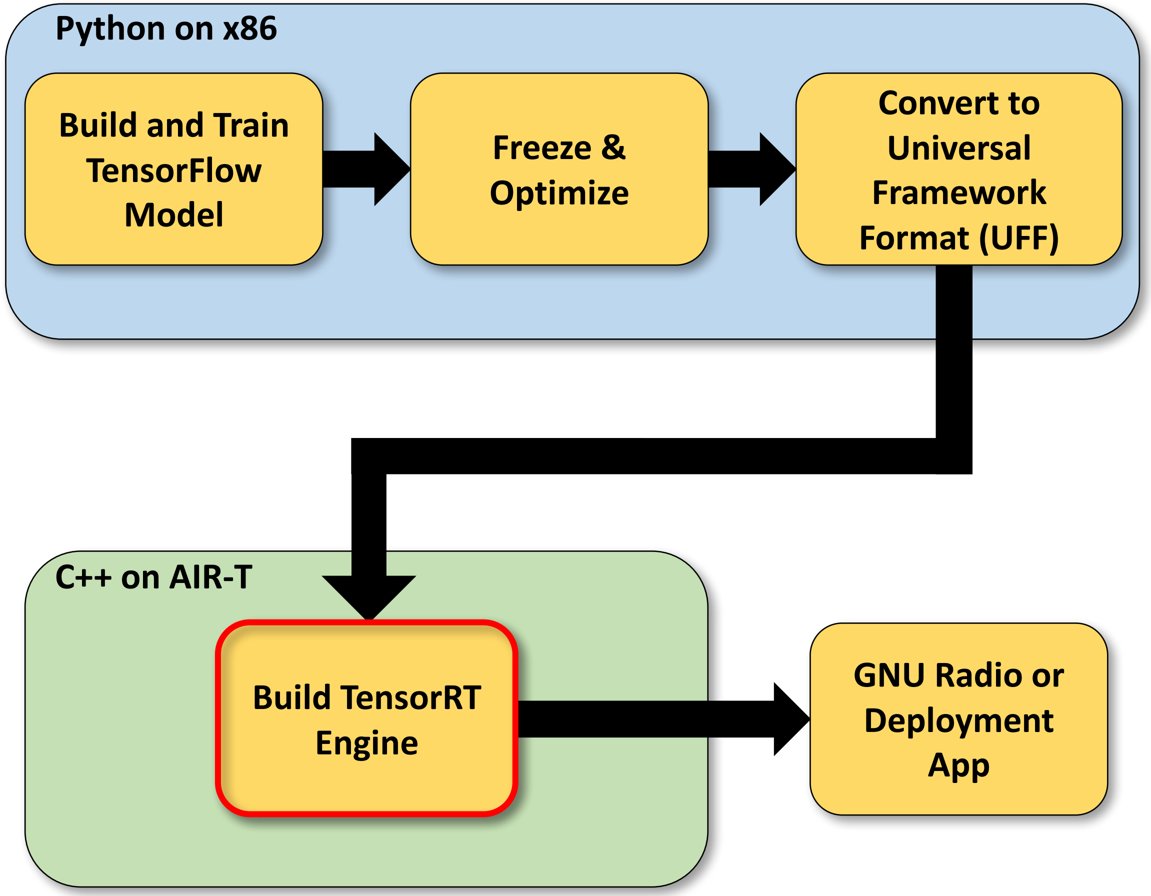
Creating Inference Engine for GNU Radio



Convert to UFF

```python
import uff

input_name = 'inputdata'
output_name = 'output/networkout'
input_filepath = 'somefile.pb'
output_filepath = 'somefile.uff'

uff_model = uff.from_tensorflow_frozen_model(
    frozen_file=input_filepath,
    input_nodes=[input_name],
    output_nodes=[output_name],
    output_filename=output_filepath,
    text=True,
    quiet=False
)
```

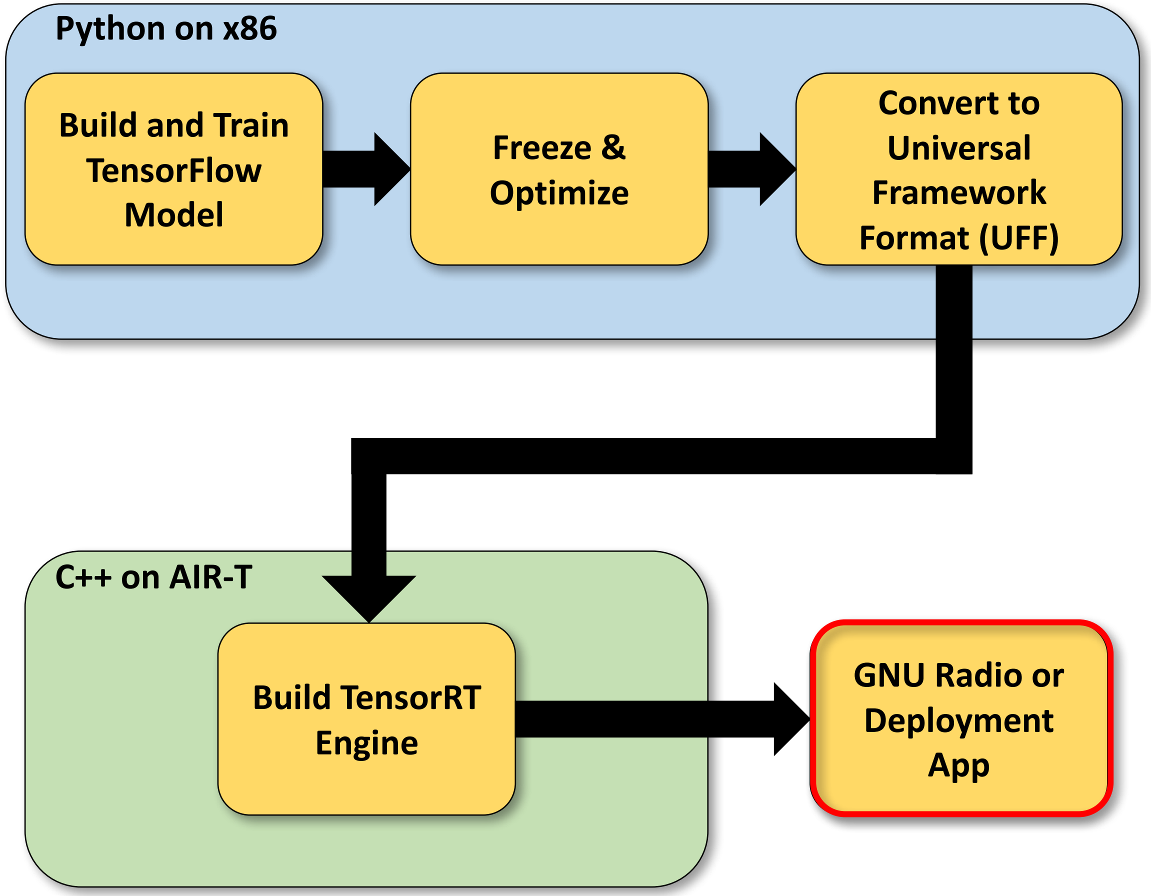
Creating Inference Engine for GNU Radio



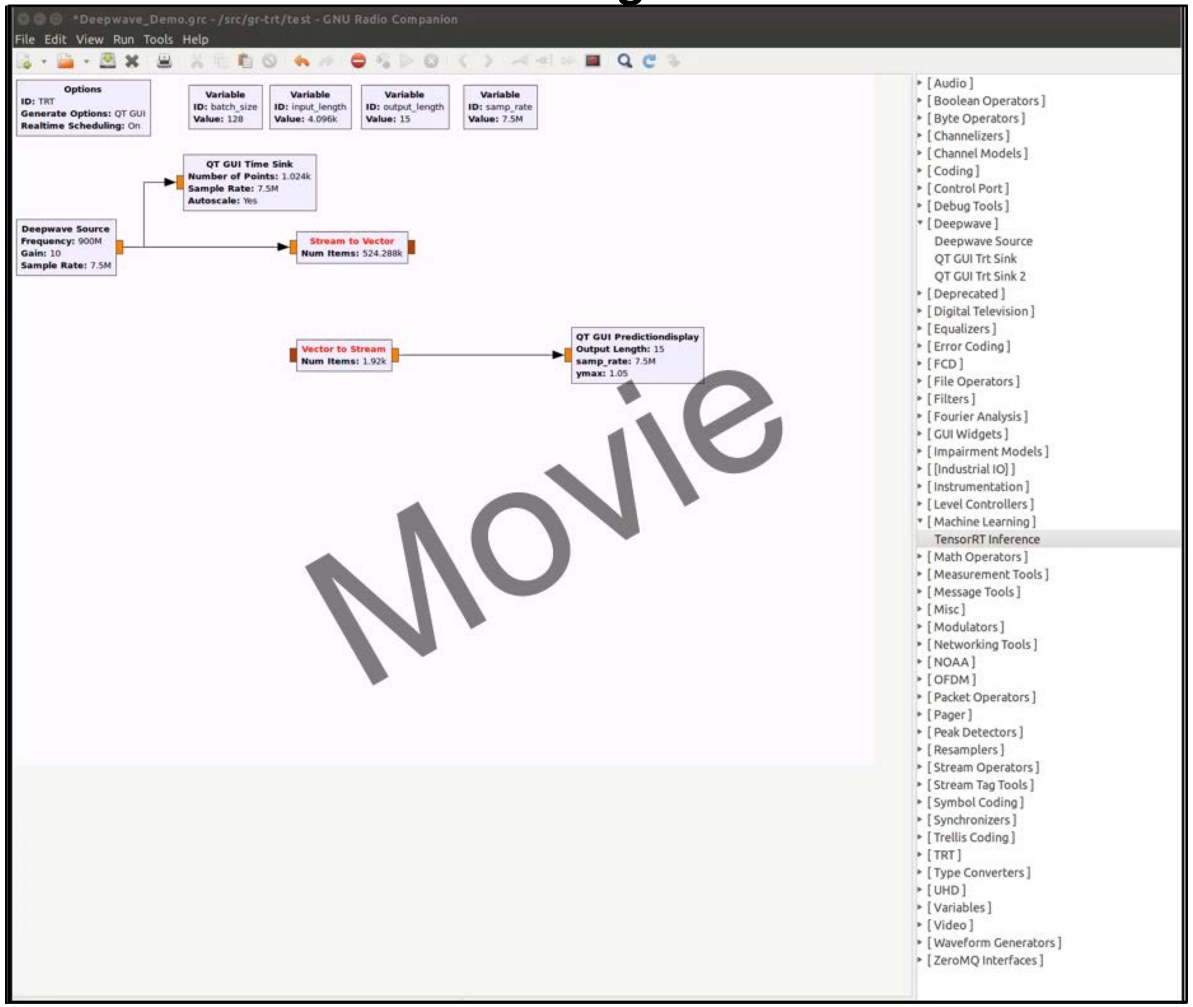
Build TensorRT Engine

```cpp
/* Copyright (c) 2018, NVIDIA CORPORATION. All rights reserved.
 * Full license terms provided in LICENSE.md file.
 */
/* snip - includes and namespace stuff here */
class Logger : public ILogger
{
  void log(Severity severity, const char * msg) override
  {
      cout << msg << endl;
  }
} gLogger;

/* snip - data conversion functions here */

int main(int argc, char *argv[])
{
  /* snip - parse command line arguments here */
  /* parse uff */
  IBuilder *builder = createInferBuilder(gLogger);
  INetworkDefinition *network = builder->createNetwork();
  IUffParser *parser = createUffParser();
  parser->registerInput(inputName.c_str(), DimsCHW(inputChannels, inputHeight, inputWidth));
  parser->registerOutput(outputName.c_str());
  if (!parser->parse(uffFilename.c_str(), *network, dataType))
  {
    /* snip - error handling goes here */
  }
  /* build engine */
  if (dataType == DataType::kHALF)
    builder->setHalf2Mode(true);
  builder->setMaxBatchSize(maxBatchSize);
  builder->setMaxWorkspaceSize(maxWorkspaceSize);
  ICudaEngine *engine = builder->buildCudaEngine(*network);
  /* serialize engine and write to file */
  ofstream planFile;
  planFile.open(planFilename);
  IHostMemory *serializedEngine = engine->serialize();
  planFile.write((char *)serializedEngine->data(), serializedEngine->size());
  planFile.close();
  /* snip - call destroy() on TensorRT objects here */
  return 0;
}
```

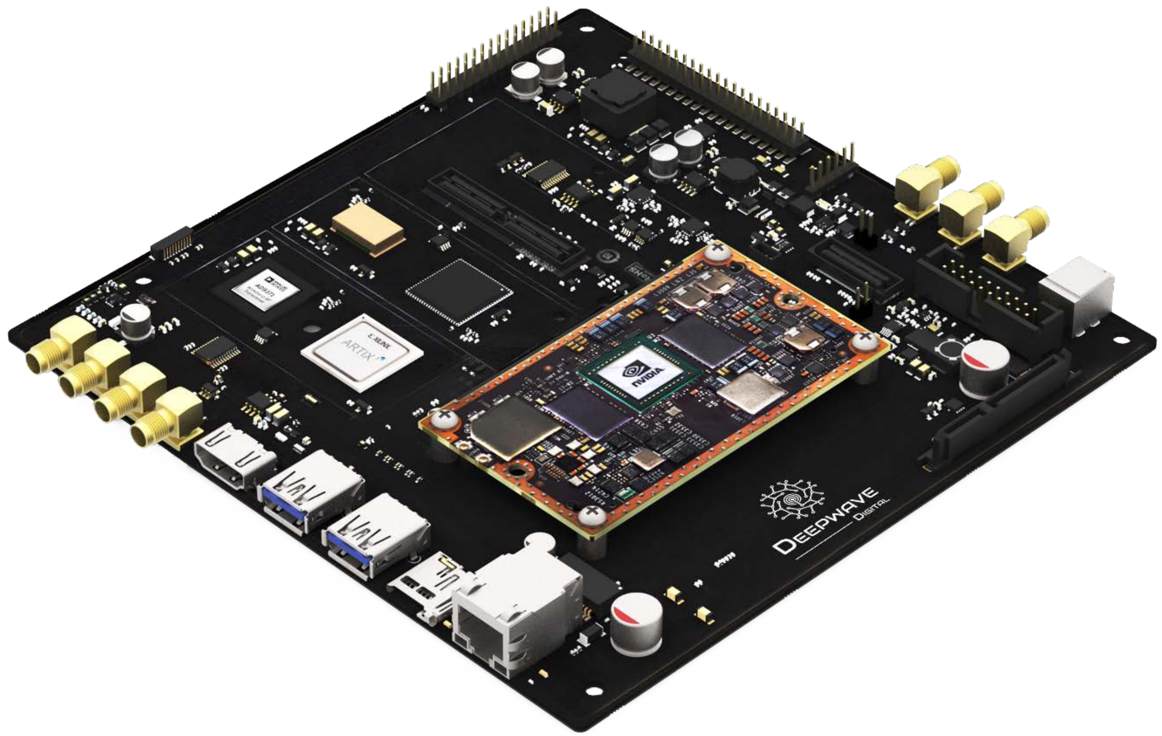
Creating Inference Engine for GNU Radio



Execute Inference Engine with GNU Radio



AIR-T Availability and Pricing



- Presales: Beginning April, 2018
  - 10% discount on all pre-orders
- Anticipated Ship Date: September 2018
- Early product testing available for select institutions:
  - Government and FFRDC labs
  - Currently looking for telecommunications partners

Contact us at sales@deepwavedigital.com

Exact specifications may slightly differ from drawing

DEEPWAVE DIGITAL