ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

DEEPWAVE DIGITAL

Deep Learning for Radio Frequency Systems

JOHN FERGUSON
CEO, Deepwave Digital

3/21/2019

Outline

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

Deep Learning and Radio Frequency (RF) Systems

Deep Learning is Emerging

Cyber

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

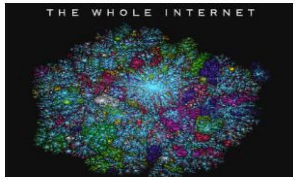
Medicine

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

Autonomy

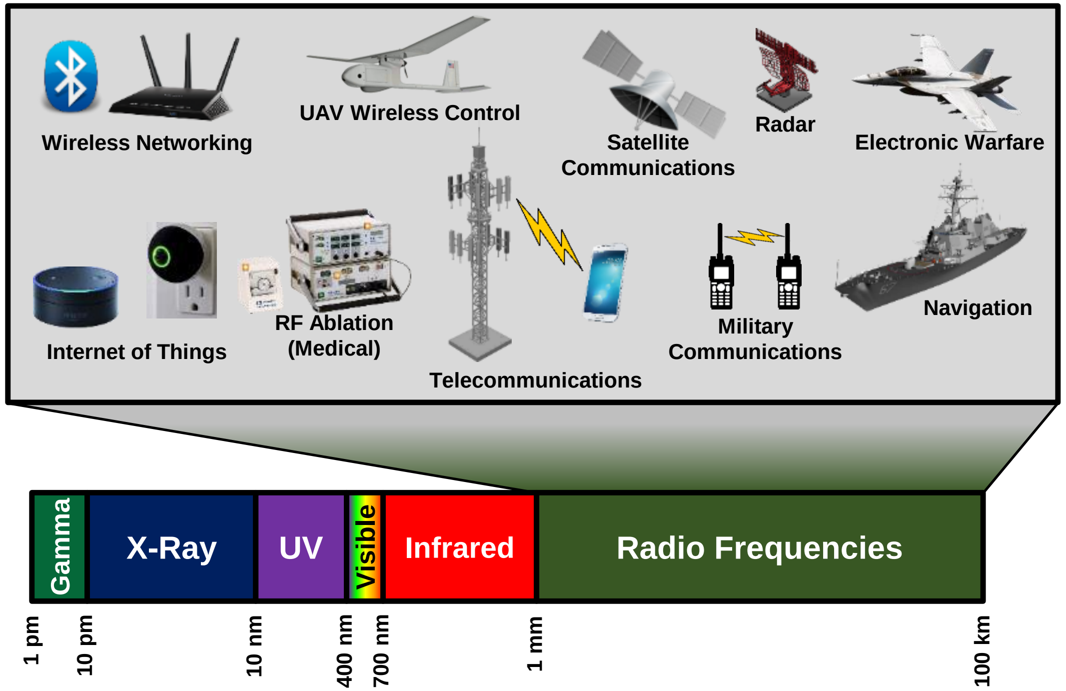
- Pedestrian / obstacle detection
- Navigation
- Street sign reading
- Speech recognition

Internet



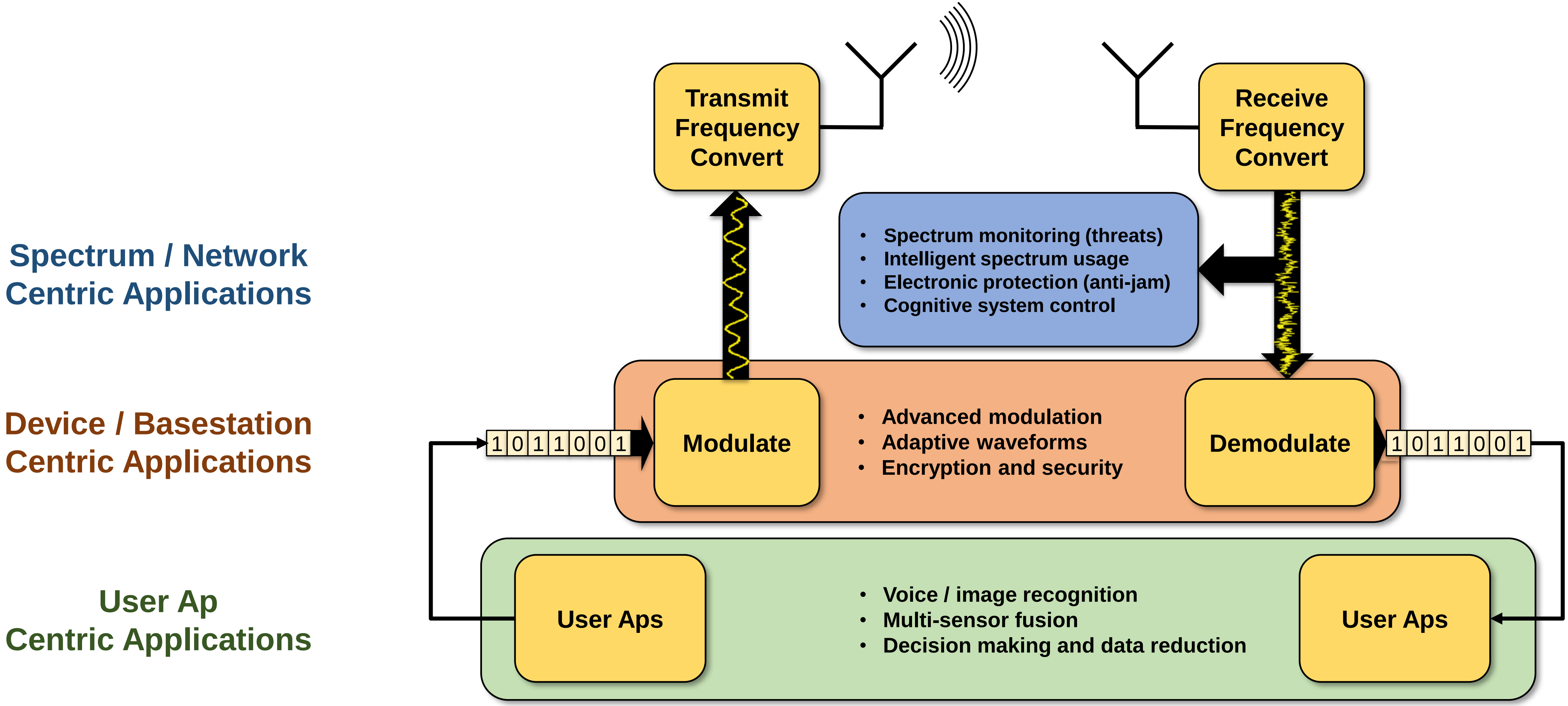
- Image classification
- Speech recognition
- Language translation
- Document / database searching

RF Technology is Pervasive



Deep learning technology has yet to make significant impact into RF systems

Where to Use Deep Learning in RF Systems



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

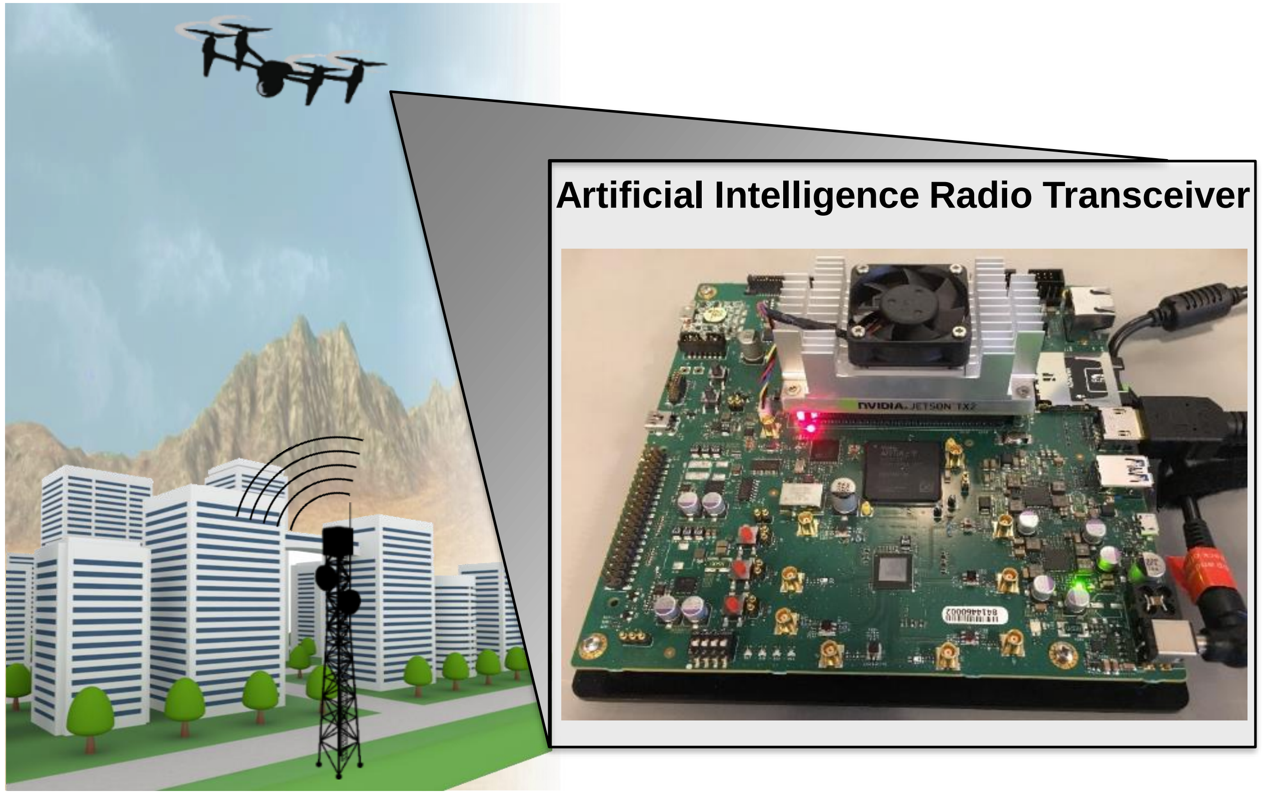
- No RF systems exist with integrated AI computational processors

- Disjointed software
- Difficult to program and understand

Deepwave's 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

WAVELEARNER Software

- AI Cores
- User Applications

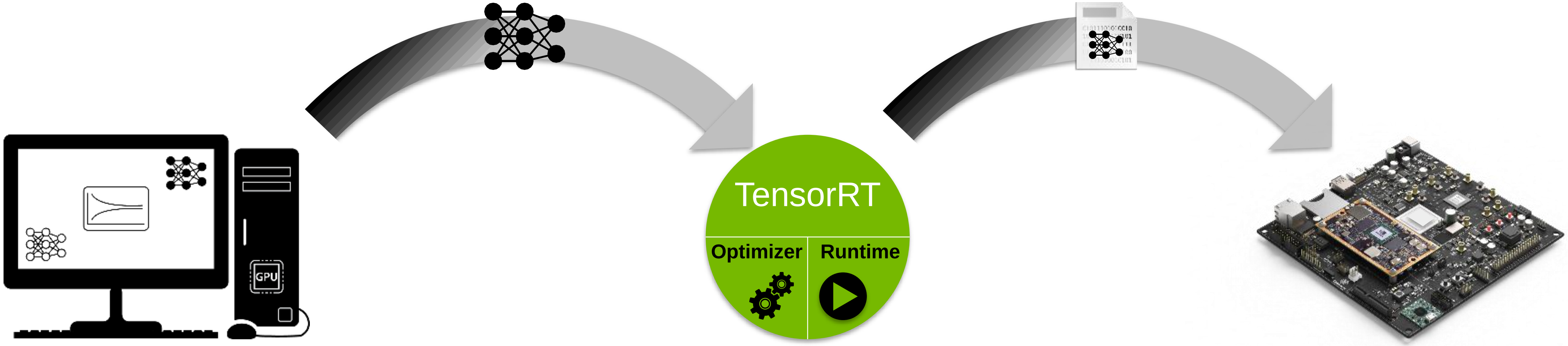
Machine Learning | Signal Processing

Industry Leading Software Tools

Hardware Controls

RF and Wireless Hardware

Inference at the Edge with GR-Wavelearner



Train Neural Network

TensorFlow
PyTorch
mxnet
Cognitive Toolkit
Caffe2
PaddlePaddle
Chainer

Optimize Neural Network

TensorRT
NVIDIA

Deploy Application

Deepwave Digital
GNURadio

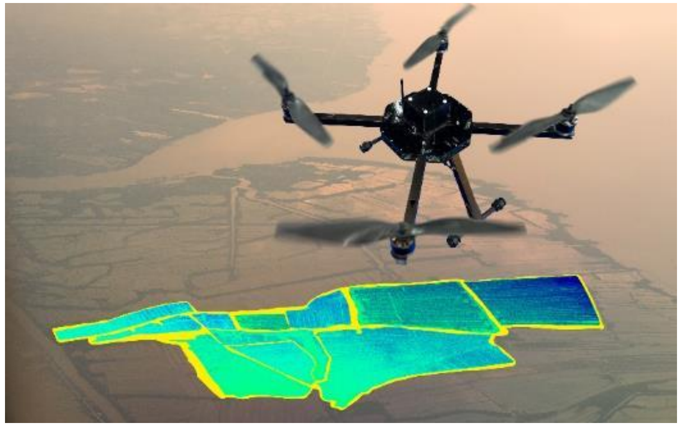
ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

Deep Learning in RF Systems

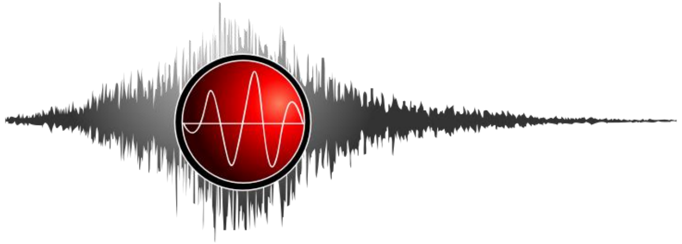
Deep Learning Comparison

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
- 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**

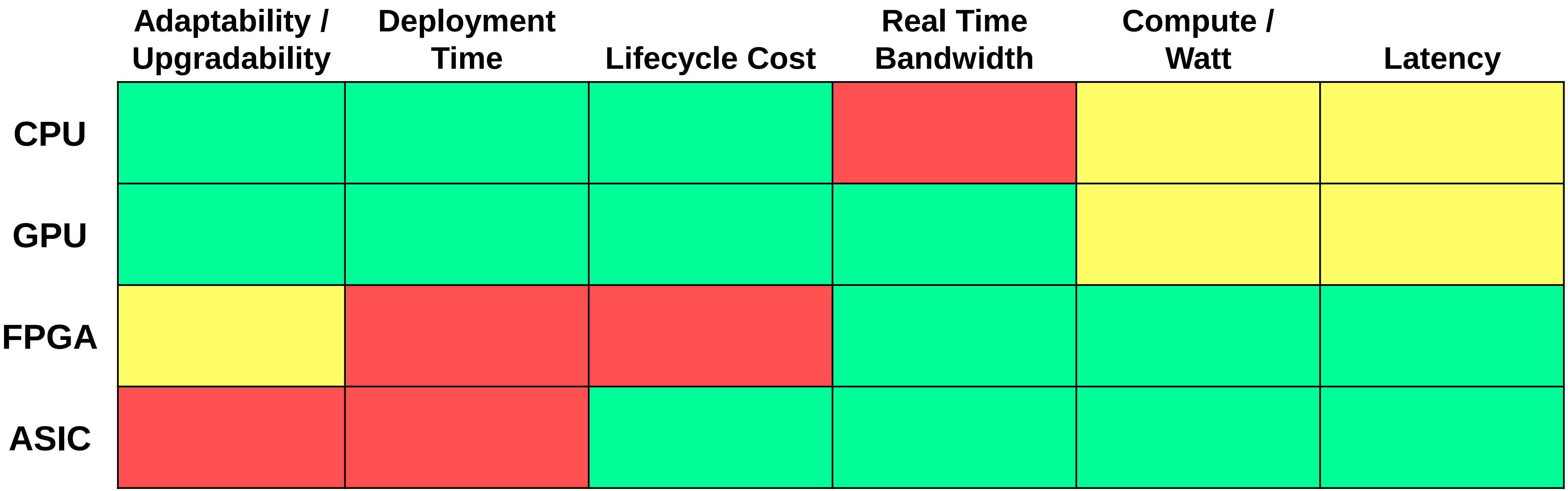
Hardware for Deep Learning in RF Systems

	Training	
	Pros	**Cons**
CPU	• Supported by ML Frameworks • Lower power consumption	• Slower than GPU • Fewer software architectures
GPU	• Supported by ML Frameworks • Widely utilized • Highly parallel / adaptable • Good throughput vs power	• Overall power consumption • Requires highly parallel algorithms
FPGA	Not widely utilized, not well suited (yet)	
ASIC	Not widely utilized, not well suited	

Hardware for Deep Learning in RF Systems

	Training		Inference	
	Pros	**Cons**	**Pros**	**Cons**
CPU	• Supported by ML Frameworks • Lower power consumption	• Slower than GPU • Fewer software architectures	• Adaptable architecture • Software programmable • Medium latency	• Low parallelism • Limited real-time bandwidth • Medium power requirements
GPU	• Supported by ML Frameworks • Widely utilized • Highly parallel / adaptable • Good throughput vs power	• Overall power consumption • Requires highly parallel algorithms	• Adaptable architecture • High real-time bandwidth • Software programmable	• Medium power requirements • Not well integrated into RF • Higher latency
FPGA	Not widely utilized, not well suited (yet)		• High power efficiency • High real-time bandwidth • Low latency	• Long development / upgrades • Limited reprogrammability • Requires special expertise
ASIC	Not widely utilized, not well suited		• Extremely power efficient • High real-time bandwidth • Highly reliable • Low latency	• Extremely expensive • Long development time • No reprogrammability • Requires special expertise

Hardware for Deep Learning in RF Systems



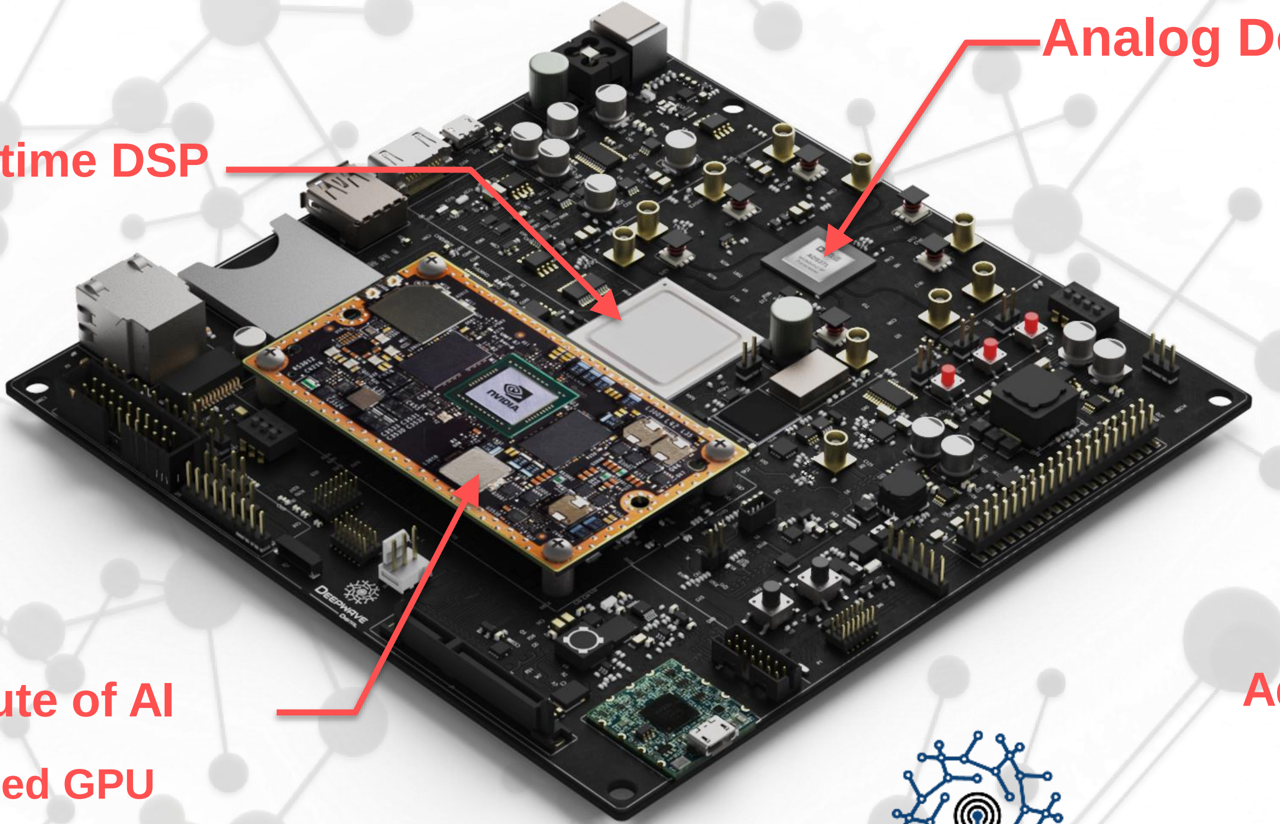
GPU signal processing can provide wideband capability and software upgradability at lower cost and development time

- Must contend with increased latency (~2 microsecond)

AHEAD OF WHAT'S POSSIBLE™

Deepwave Digital Technology

Artificial Intelligence Radio Transceiver (AIR-T)



Analog Devices wireless transceiver

- AD9371
- 2x2 MIMO
- 0.3 – 6 GHz
- Built-in calibration (QEC, LO)

FPGA for real time DSP

GPU for Edge compute of AI

- NVIDIA embedded GPU
- 256 GPU cores
- 4 CPU cores (ARM)
- 8 Gbytes memory

Additional Connectivity

- 1 PPS / 10 MHz for GPS sync
- External LO input
- HDMI, USB 2.0/3.0, SATA, Ethernet, SD Card, GPIO

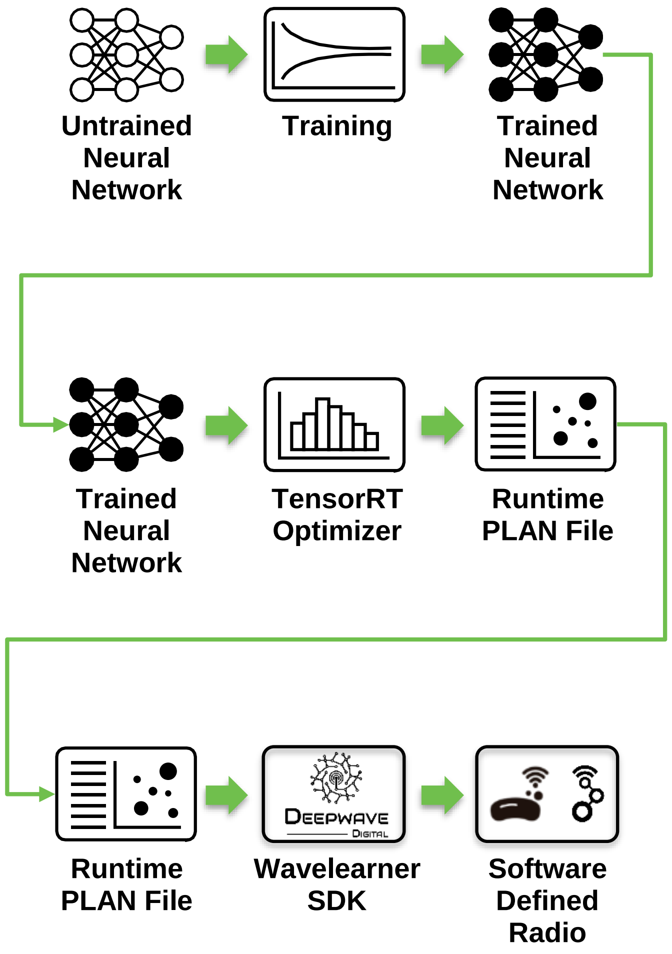
NVIDIA

DEEPWAVE DIGITAL

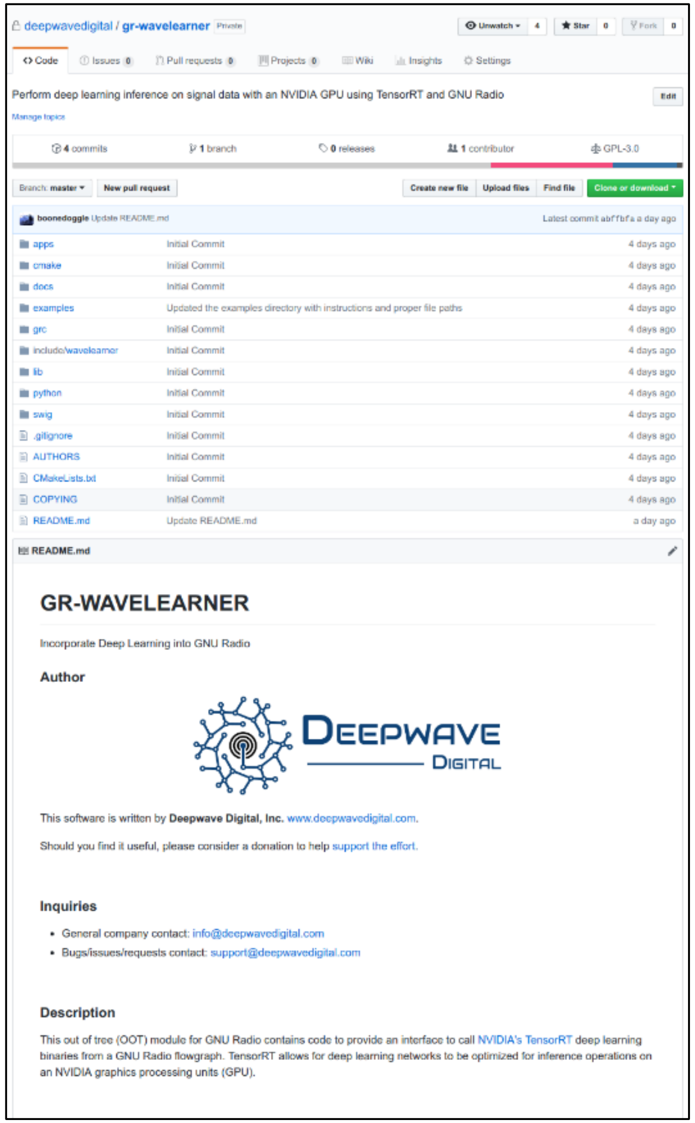
ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

Neural Network Deployment in Three Steps

- **Step 1 – Train**
  - Using TensorFlow, MATLAB, Keras, PyTorch, etc.
- **Step 2 – Optimize**
  - Using NVIDIA's TensorRT
- **Step 3 – Deploy**
  - Using Deepwave's GR-Wavelearner and GNU Radio



Open Source GR-Wavelearner Software

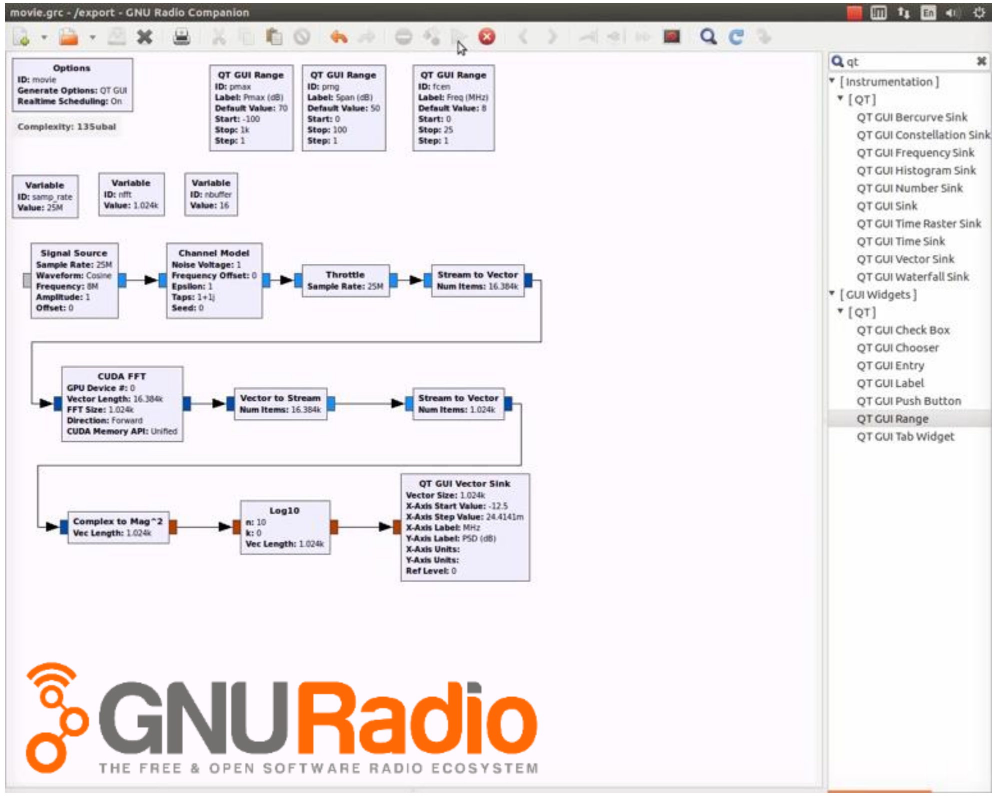


- **Goal is to help the open source community easily deploy deep learning within signal processing applications**
- **Well documented README with dependency installation instructions to get started quickly**
  - Ubuntu 16.04 recommended, Windows 10 supported
  - NVIDA Docker Container 18.08*
- **Signal classifier example provided:**
  - GNU Radio Flowgraph
  - Python source code
  - PLAN files that are executable on the AIR-T and Maxwell
  - Signal data file example for testing
- **Support for TensorRT 5.0**
- **Available at: deepwavedigital.com/wavelearner**

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

GNU Radio – Software Defined Radio (SDR) Framework

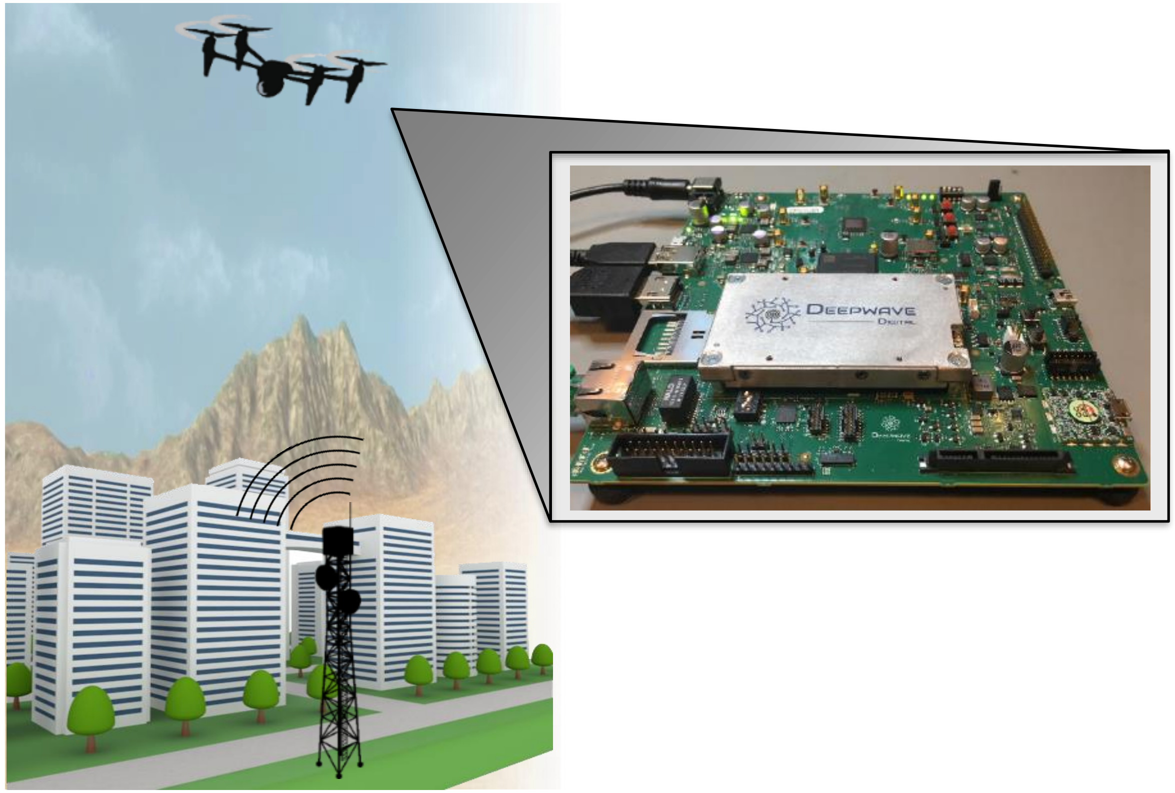
- **Popular open source software defined radio (SDR) toolkit:**
  - RF Hardware optional
  - Can run full software simulations
- **Python API**
  - C++ under the hood
- **Easily create DSP algorithms**
  - Custom user blocks
- **Primarily uses CPU**
  - Advanced parallel instructions
  - Recent development: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



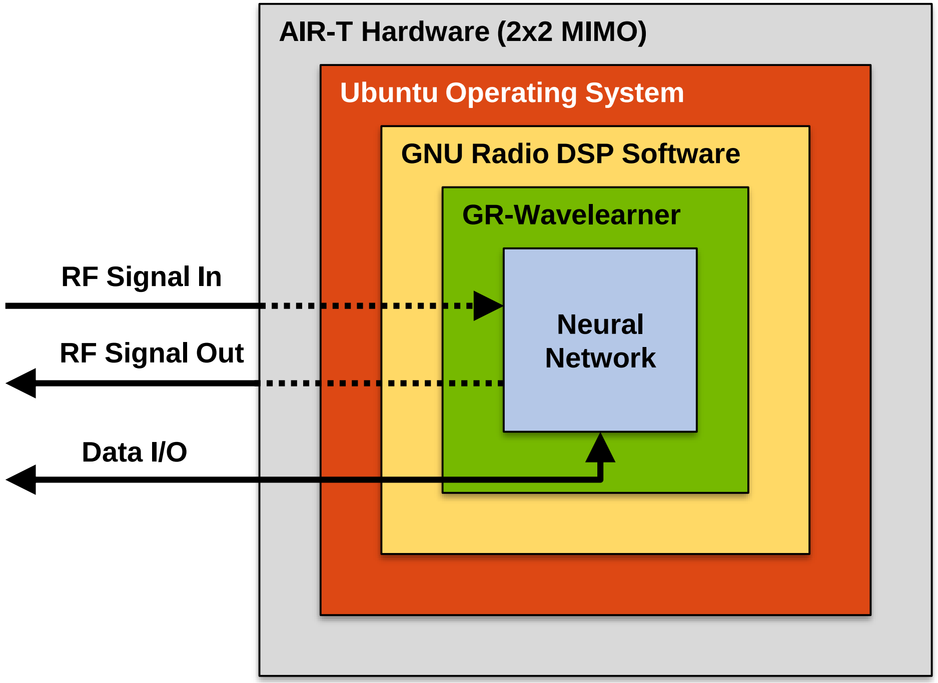
Deepwave's Solution and Platform

Seamless Deployment of Deep Learning in RF Applications

Artificial Intelligence Radio Transceiver (AIR-T)



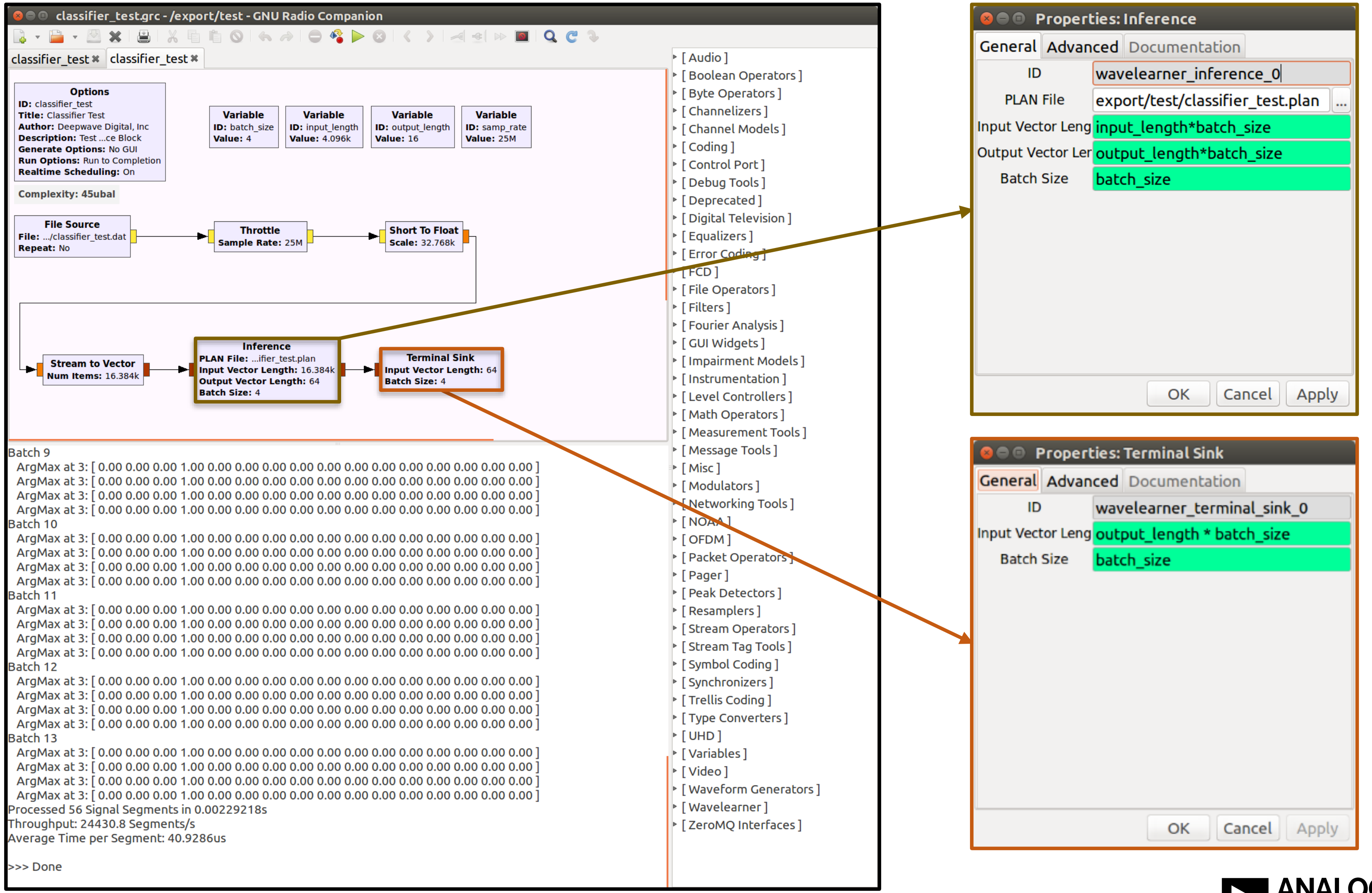
GR-Wavelearner Software



The AIR-T pre-installed with all software tools necessary for deployment

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:**
  - **<u>Inference</u> – TensorRT wrapper for GNU Radio**
  - **<u>Terminal Sink</u> – Python module for displaying classifier output**



Training to Deployment Workflow

Train → Optimize → Deploy

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

Native support for TensorRT

TensorFlow
MathWorks®
K

Support Via ONNX**

Caffe2
Cognitive Toolkit
PaddlePaddle
PyTorch
Chainer
mxnet

* Deep Neural Network

**Open Neural Network Exchange

Deepwave's Training to Deployment Workflow

Train → **Optimize** → Deploy

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

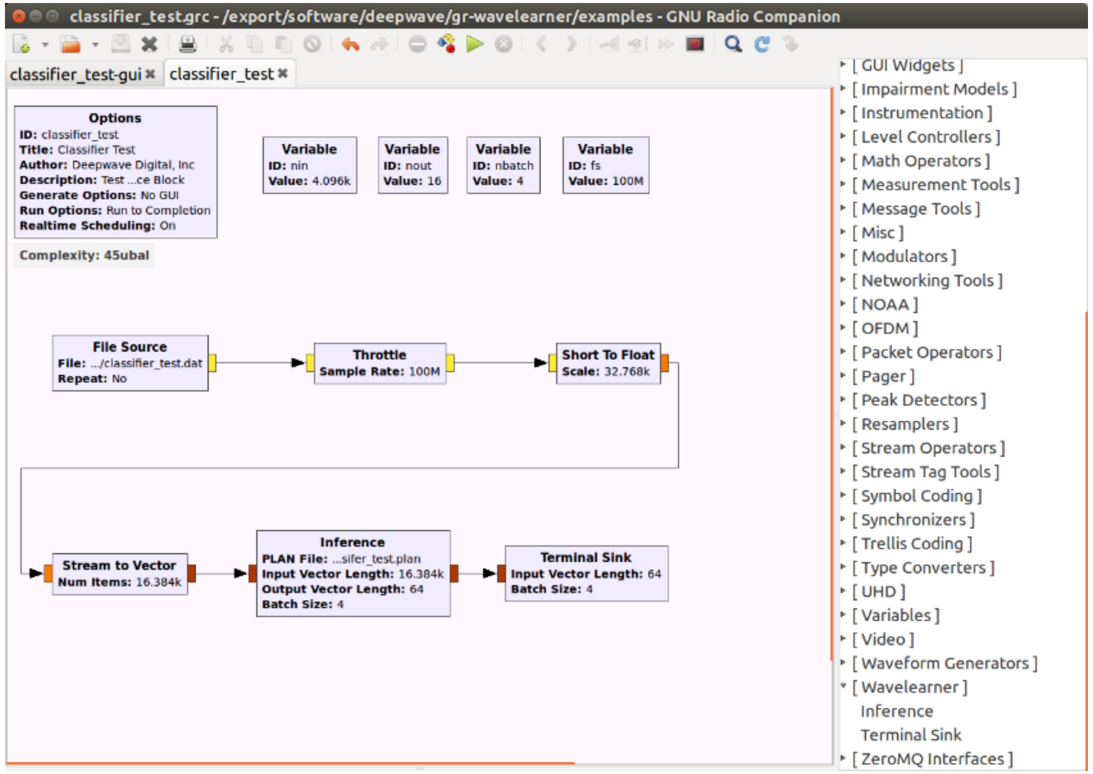
- Not all layers are supported, but most common ones are
- PLAN file must be created on deployment architecture
  - Python conversion not available on ARM (Jetson)
  - Limited transferability of PLAN files

* Deep Neural Network

Deepwave's Training to Deployment Workflow

Train → Optimize → **Deploy**

GNU Radio Companion GUI



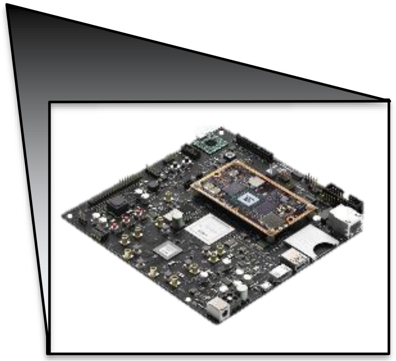
Python API
Real time DNN* RF system in 35 lines of code!

```python
#!/usr/bin/env python2
from gnuradio import blocks
from gnuradio import gr
import wavelearner

class ClassifierTest(gr.top_block):
    def __init__(self):
        gr.top_block.__init__(self)

        # Define signal processing functions
        self.source       = blocks.file_source(gr.sizeof_short, inputdata, False)
        self.throttle     = blocks.throttle(gr.sizeof_short, fs, True)
        self.type_convert = blocks.short_to_float(1, norm)
        self.buffer       = blocks.stream_to_vector(gr.sizeof_float, nbatch*nin)
        self.inference    = wavelearner.inference(planfile, nin*nbatch, nout*nbatch, nbatch)
        self.terminal_out = wavelearner.terminal_sink(nout * nbatch, nbatch)

        # Connect functions in flowgraph
        self.connect((self.source, 0), (self.throttle, 0))
        self.connect((self.throttle, 0), (self.type_convert, 0))
        self.connect((self.type_convert, 0), (self.buffer, 0))
        self.connect((self.buffer, 0), (self.inference, 0))
        self.connect((self.inference, 0), (self.terminal_out, 0))

# Define Settings
inputdata = '/path/to/classifier_test.dat'
planfile  = '/path/to/classifier_test.plan'
fs        = 100e6
nin       = 4096
nout      = 16
nbatch    = 4
norm      = 32768
tb        = ClassifierTest()
tb.start()
tb.wait()
```

* Deep Neural Network

Example Signal Detection and Classification

Multi-transmitter Environmental Scenario



Receiver monitors congested spectrum using deep learning

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							

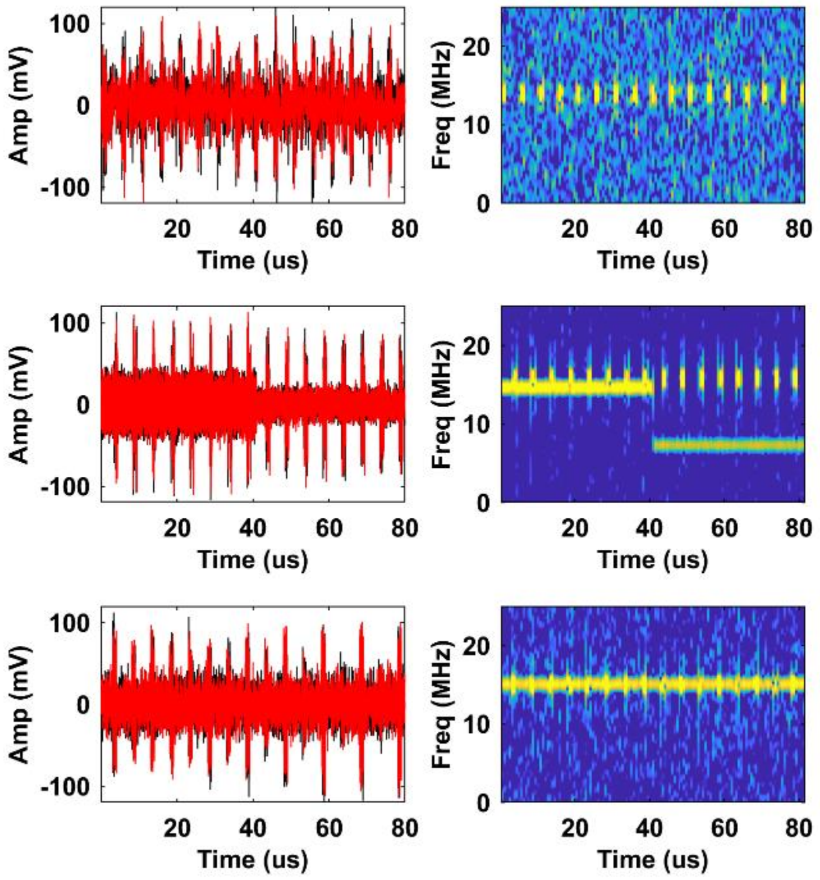
Technique demonstration shown with nominal radar signals

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

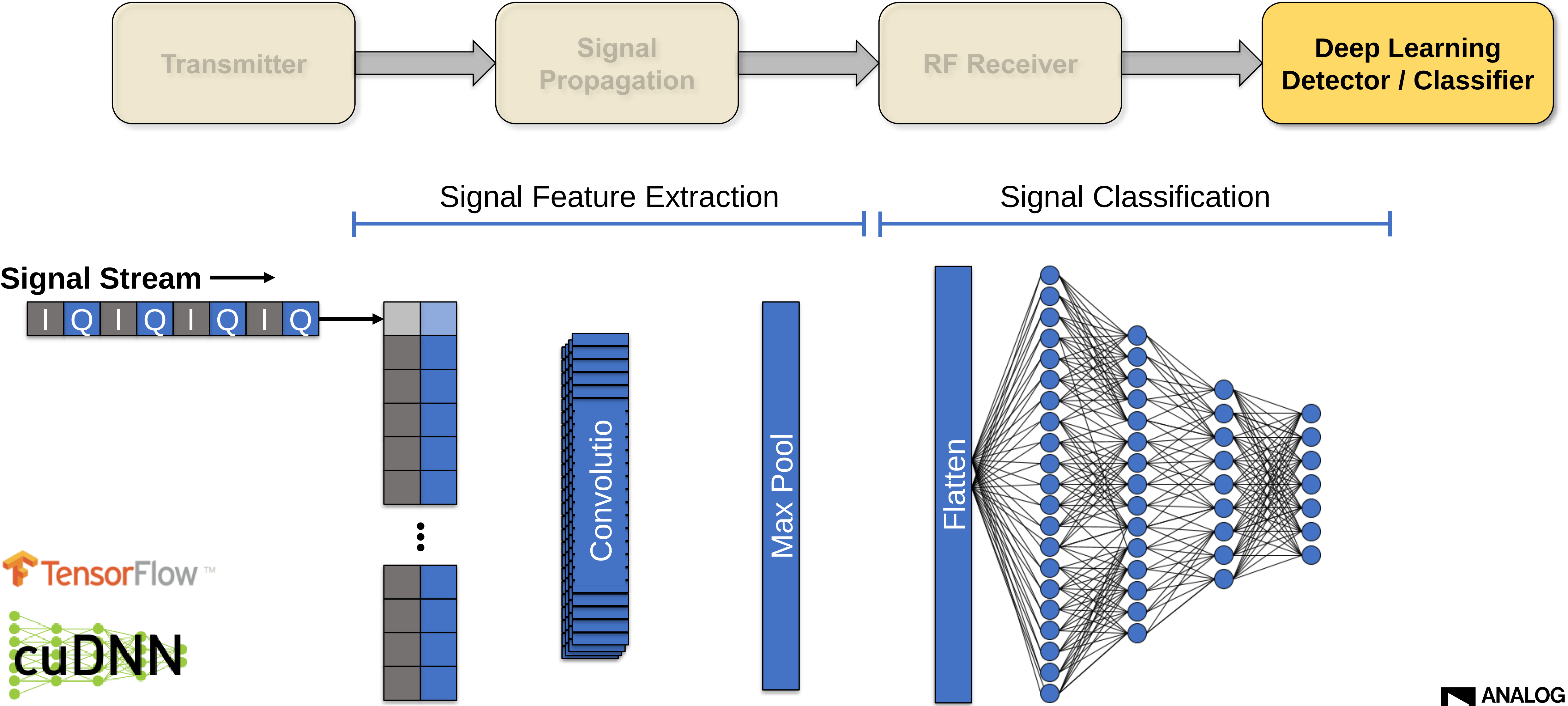
Dataset Overview

- ► 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

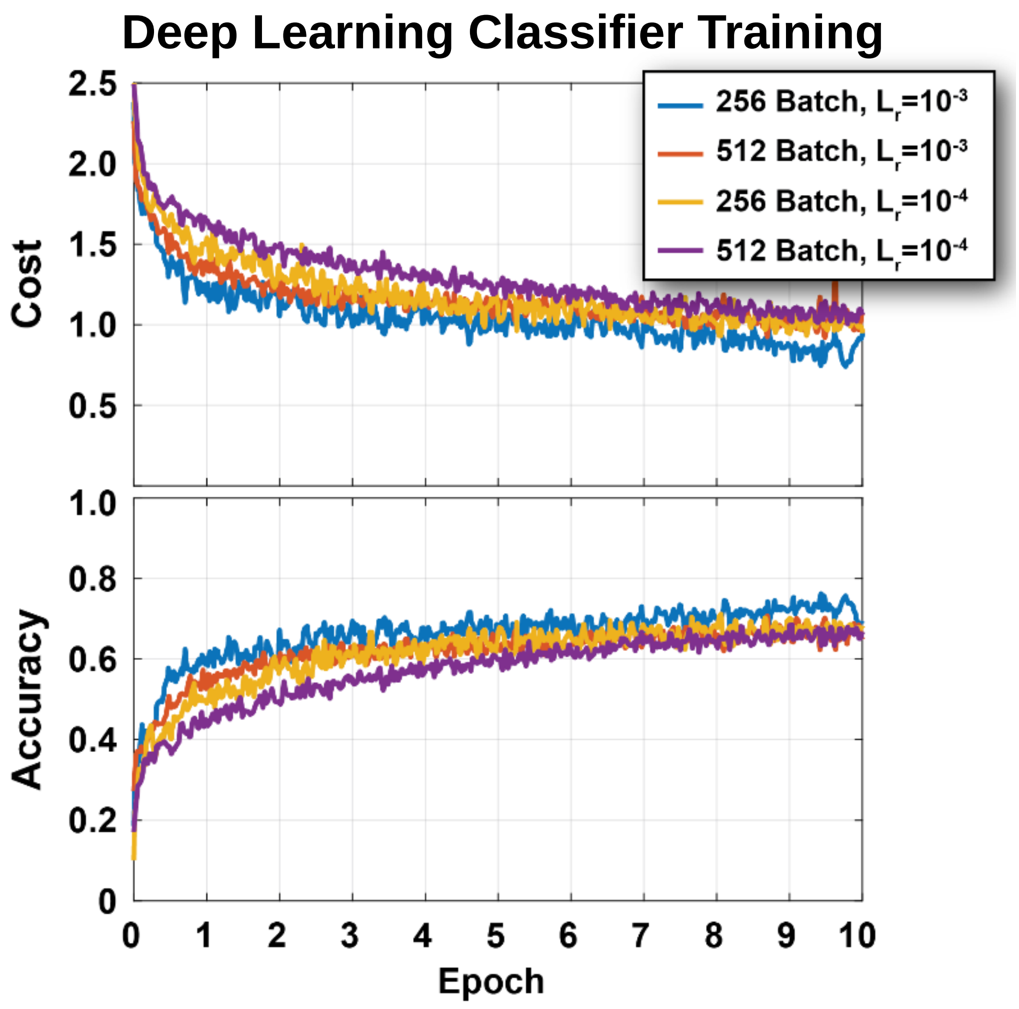


Radar Signal Detector Model: Example Classifier



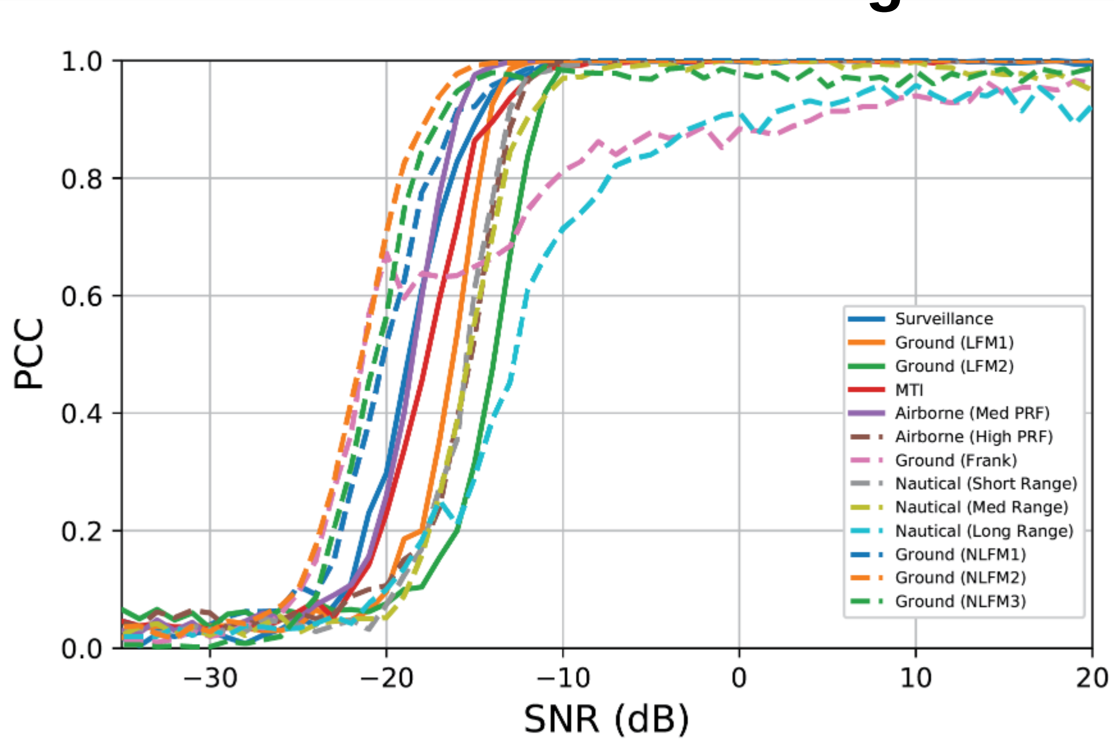
Training Process and Progress

- 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



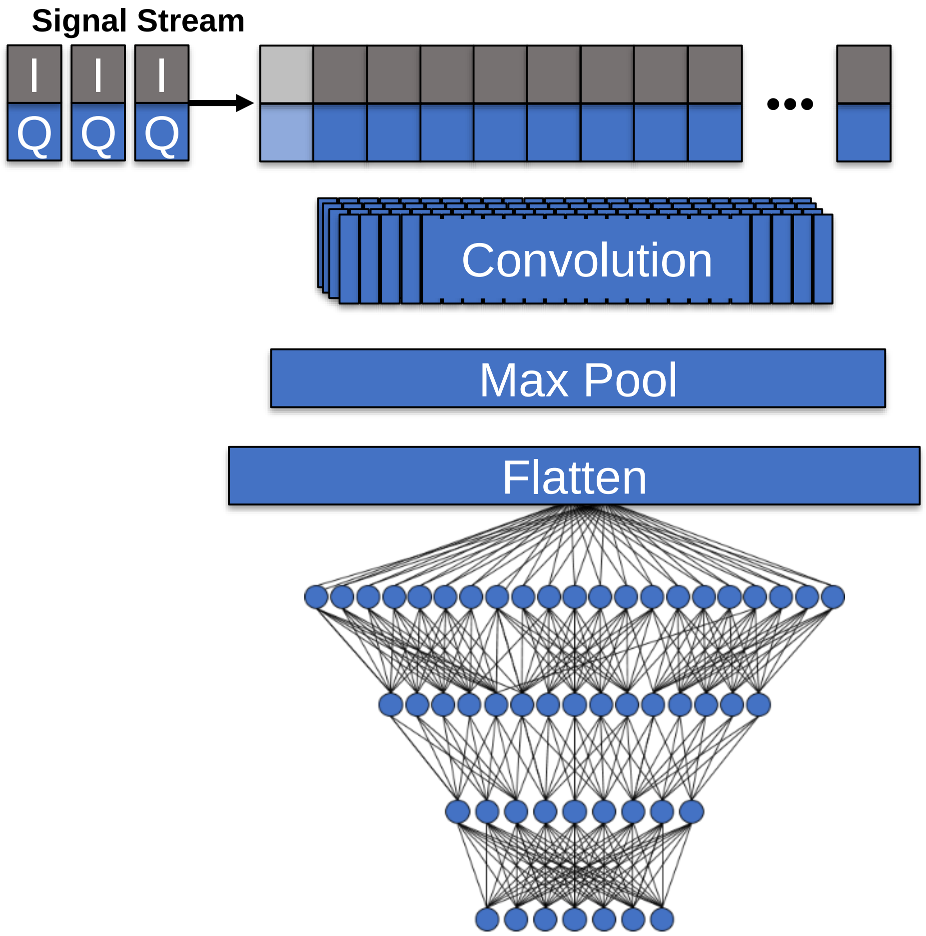
Receiver Operating Characteristic (ROC) Curve

Probability of Correct Classification for All Signals



Benchmarking Deep Learning Inference on Embedded GPUs

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

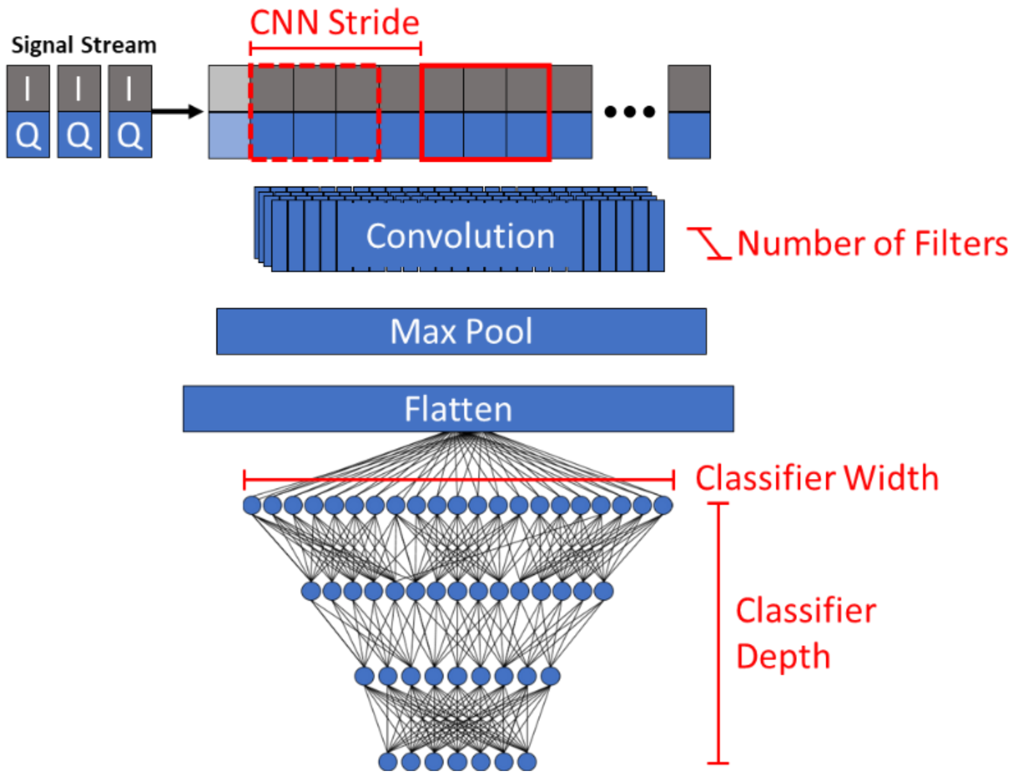
Performance Benchmarking Test Setup

Define Model Structure → **Train Model** → **Measure Sensitivity** → **Measure Real Time Throughput**

Performance Benchmarking Test Setup

Define Model Structure → Train Model → Measure Sensitivity → Measure Real Time Throughput

Repeat for multiple models



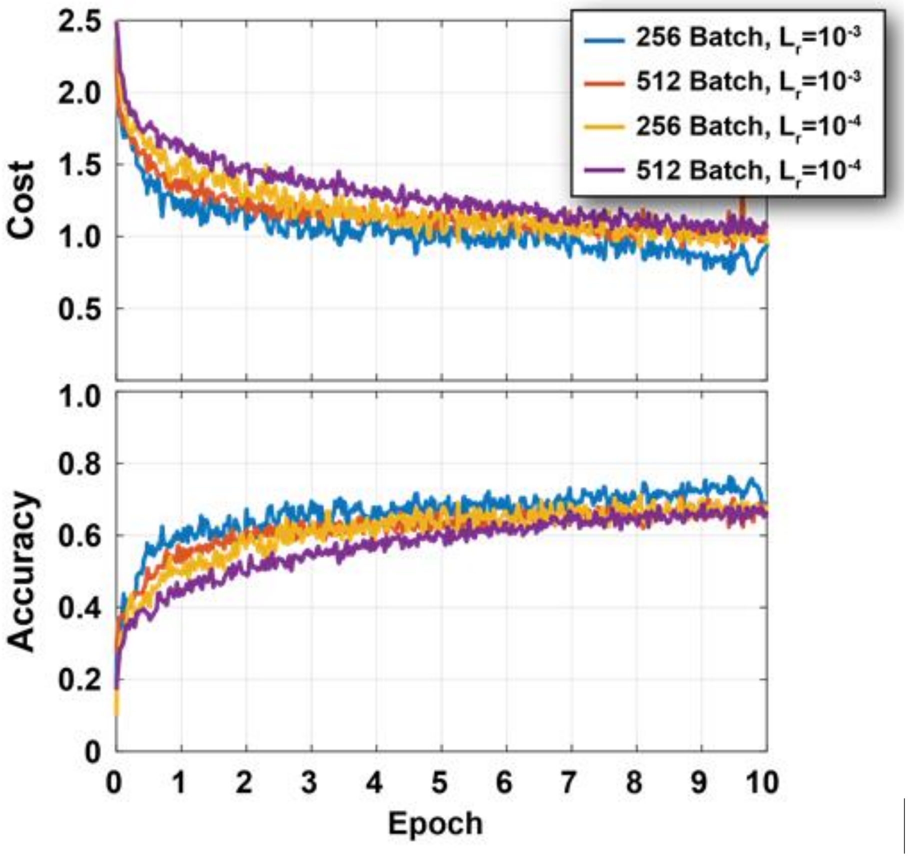
Model Tuning Variables

	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 Models Tested	728

Performance Benchmarking Test Setup

Define Model Structure → **Train Model** → Measure Sensitivity → Measure Real Time Throughput

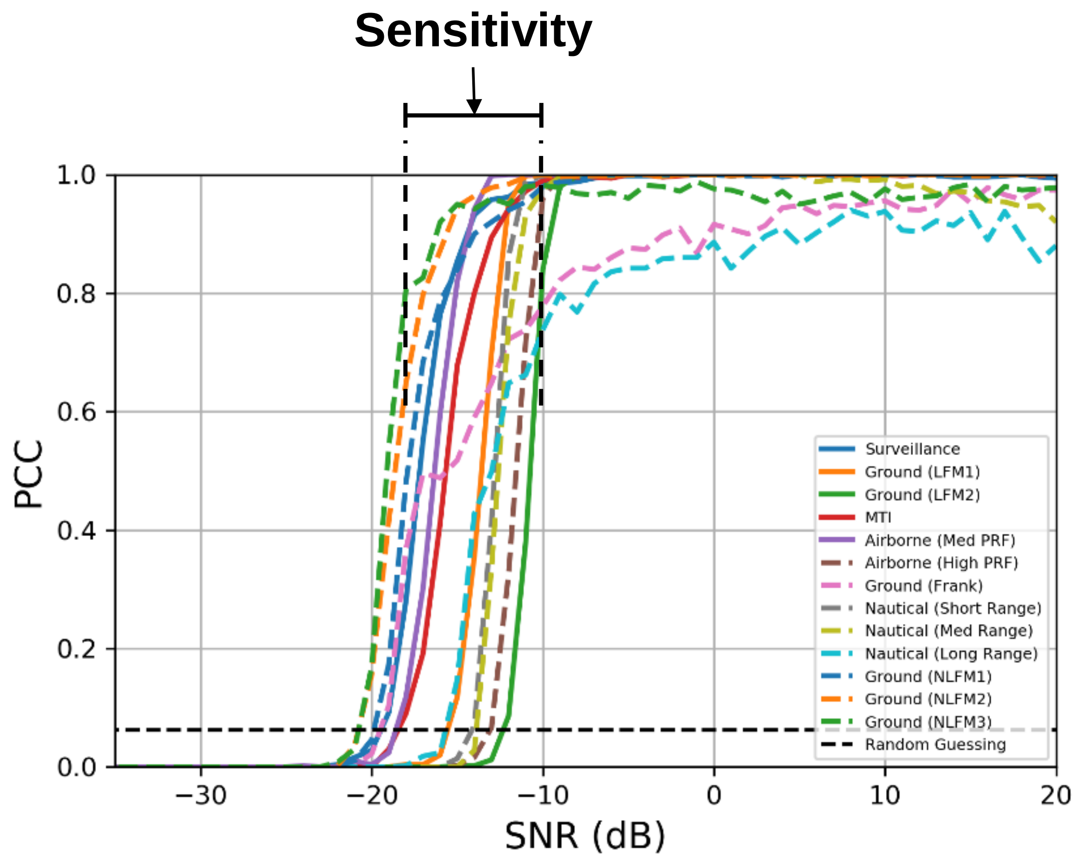
- 1000 training segments per SNR
  - 55 different SNR values
- Softmax cross entropy
- Adam Optimizer
- Quadro GP100 GPU
- Create UFF File for each model



Performance Benchmarking Test Setup

Define Model Structure → Train Model → **Measure Sensitivity** → Measure Real Time Throughput

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



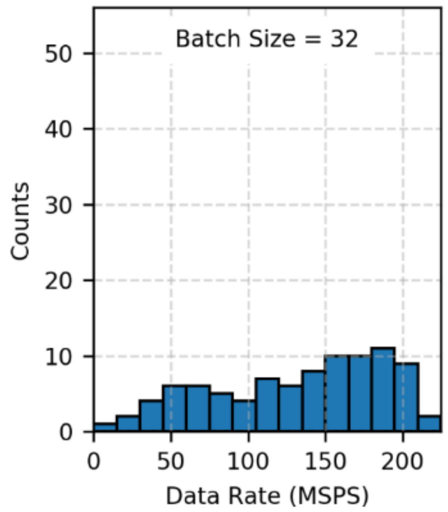
Performance Benchmarking Test Setup

Define Model Structure → Train Model → Measure Sensitivity → **Measure Real Time Throughput**

- 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

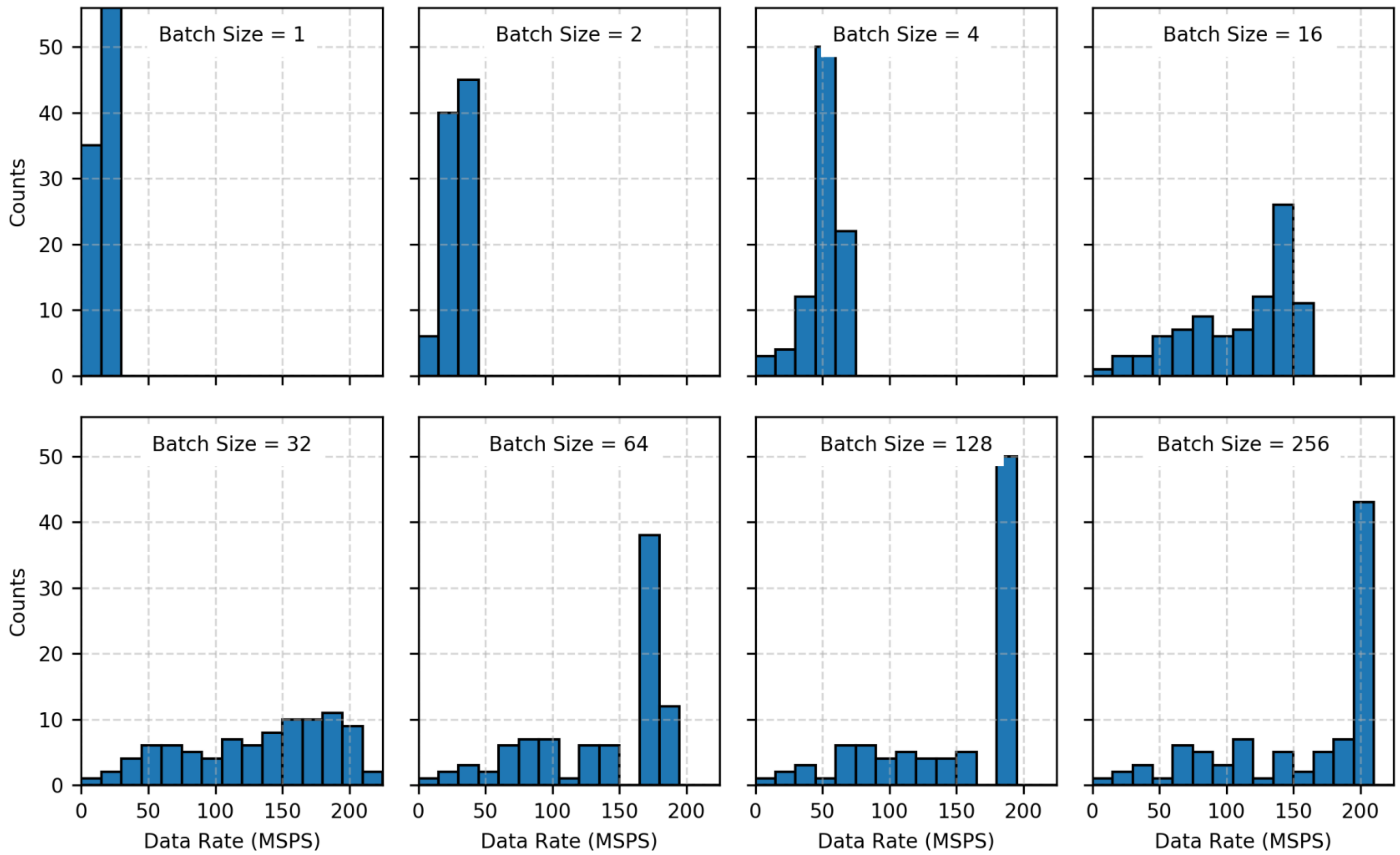
Data Rate Benchmark for AIR-T (Tegra TX2)



- 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

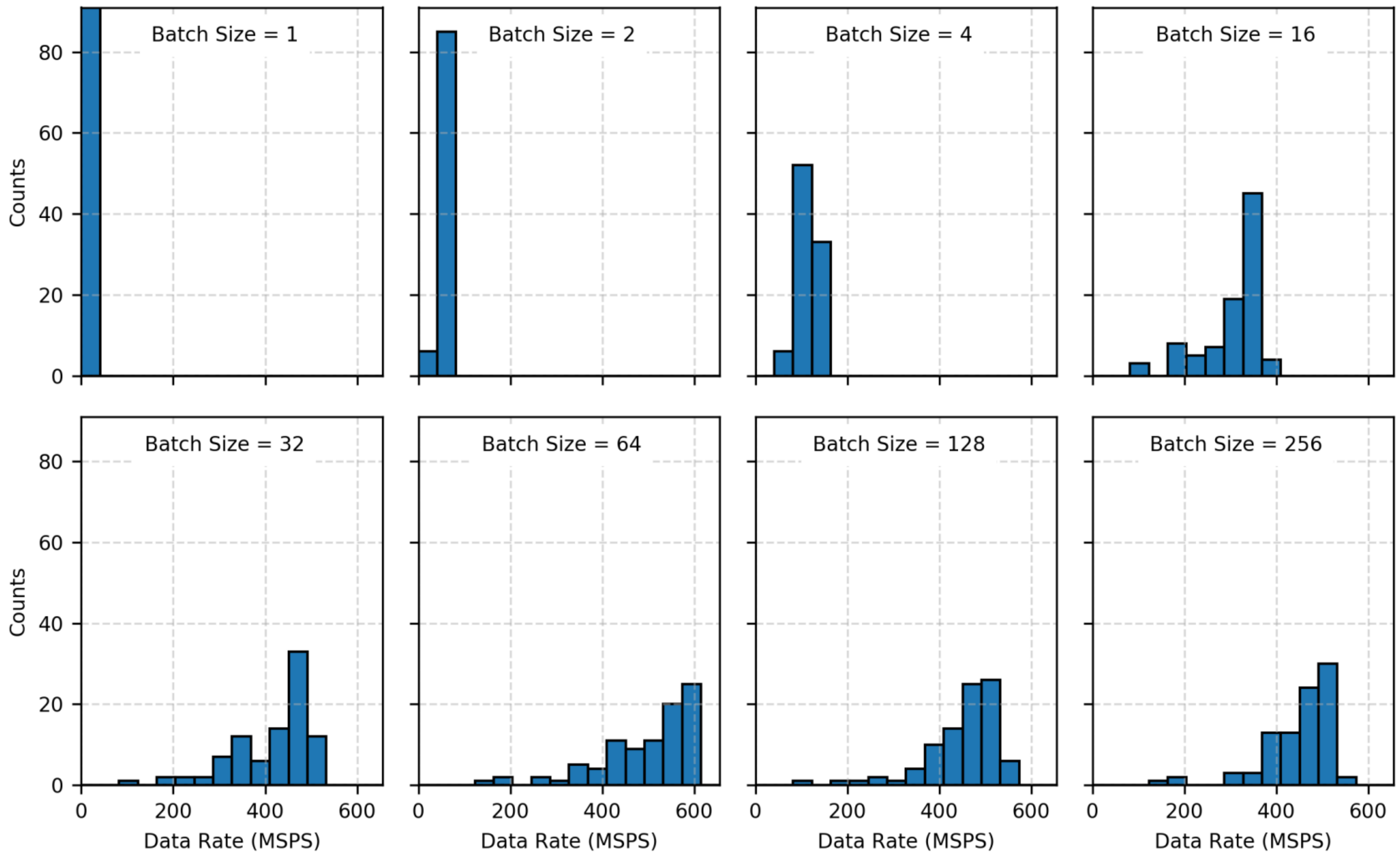
Data Rate Benchmark for AIR-T (Tegra TX2)



- 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)



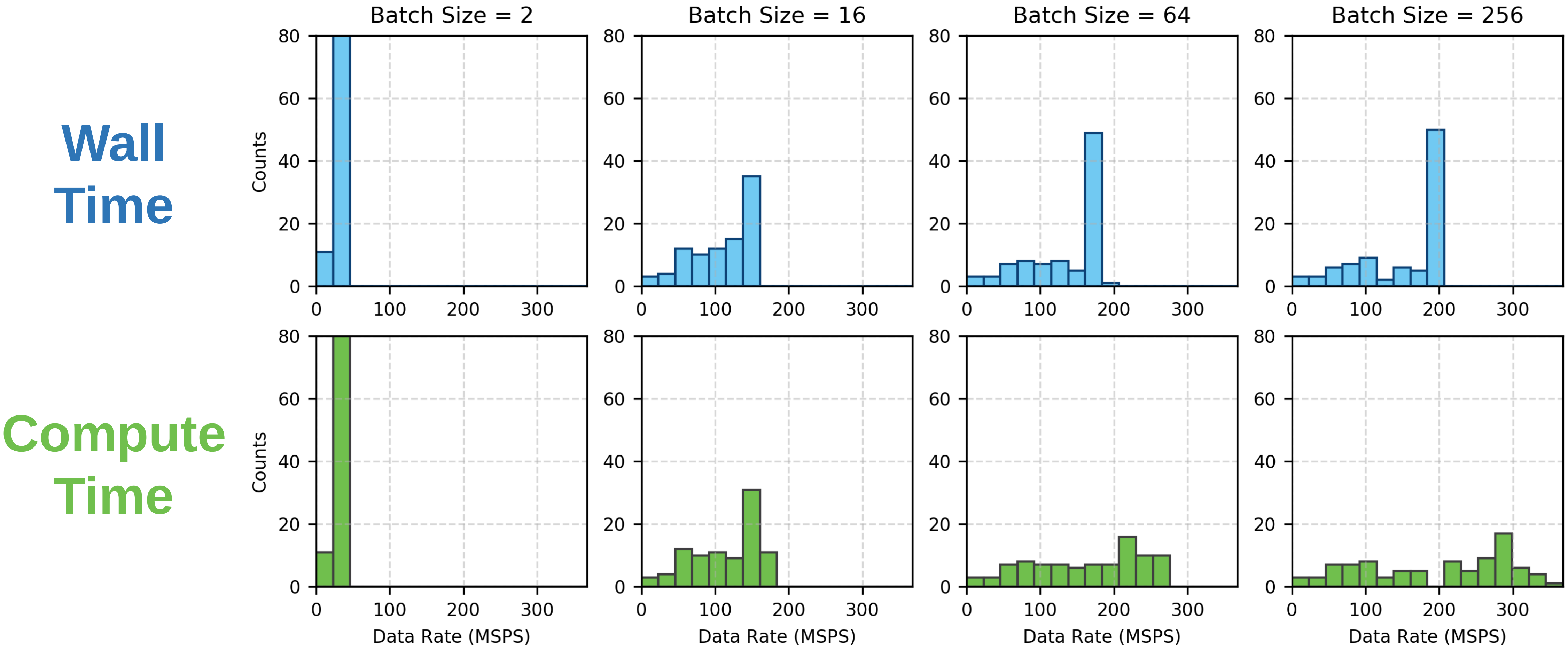
- Tested 91 different CNN classifier models
- Maximum real-time inference data rate for 8 different batch sizes
- Using unified memory will increase throughput

Desktop (GP100)

ANALOG DEVICES
AHEAD OF WHAT'S POSSIBLE™

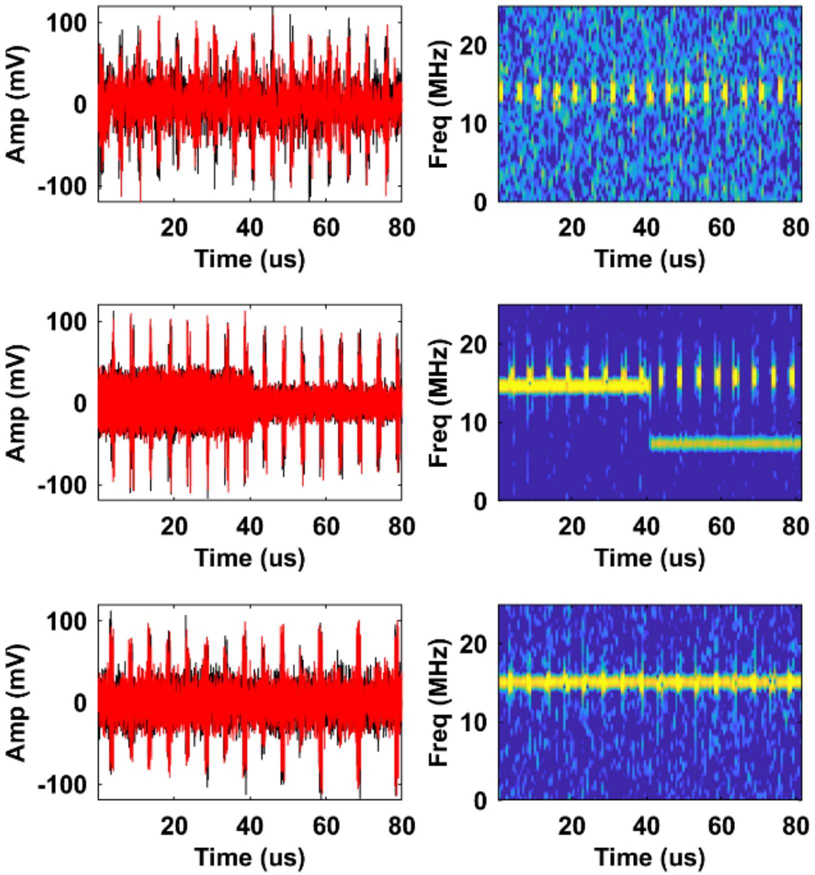
DEEPWAVE DIGITAL, INC

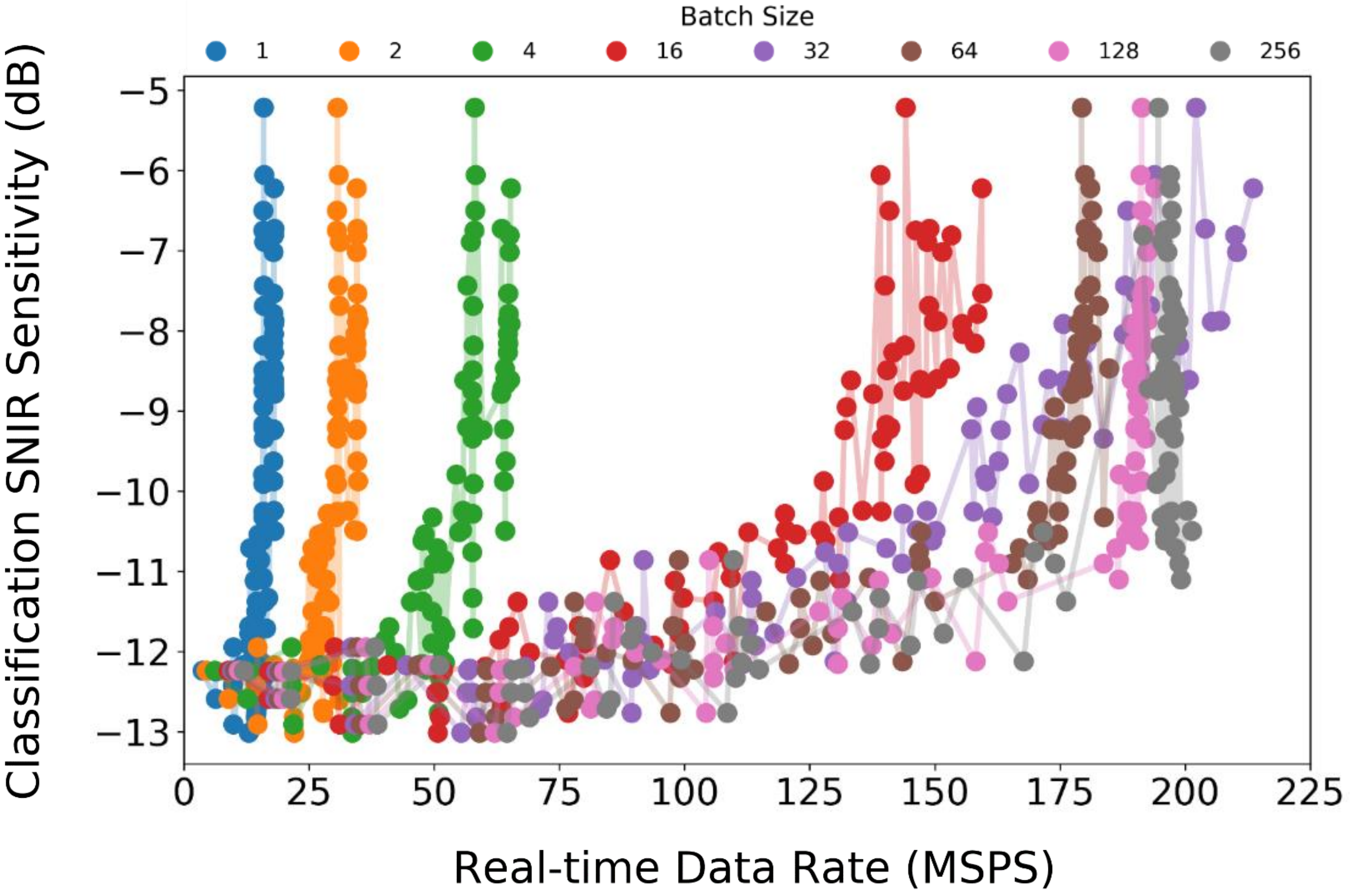
Wall Time vs. Compute Time for AIR-T



Real time data rate limited by GNU Radio overhead

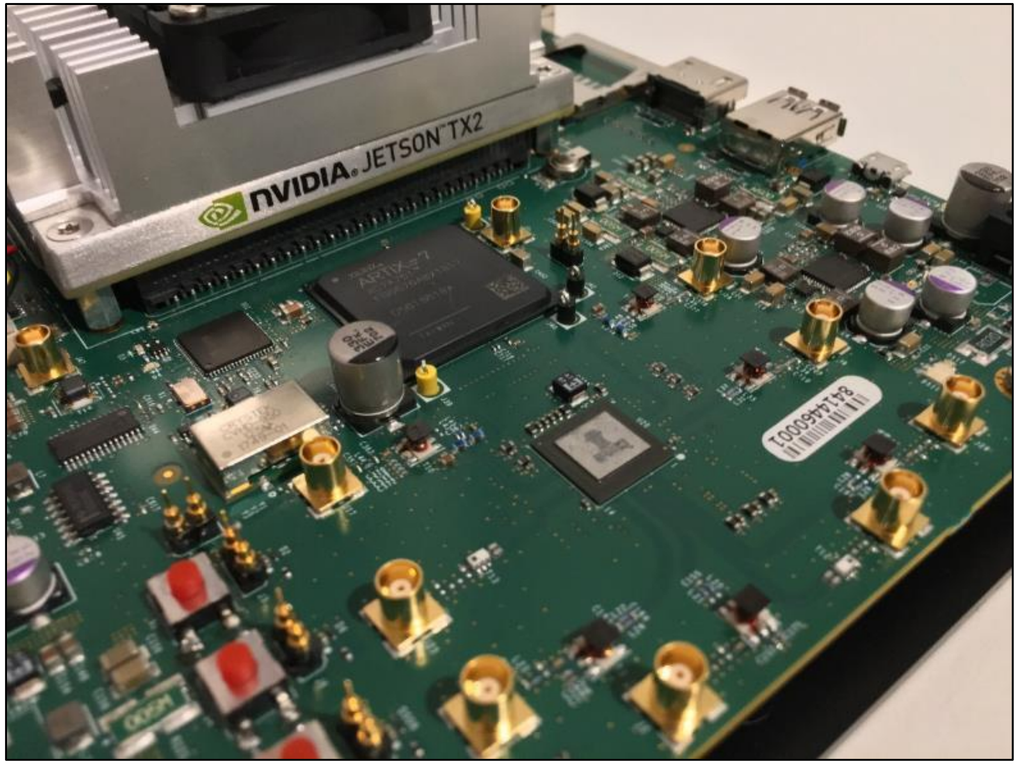
Model Accuracy Benchmarks





Summary

More info at www.deepwavedigital.com/sdr



- 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 for computation
- GR-Wavelearner software:
  - Open source inference engine for software-defined radios
  - Available now on Deepwave's 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

Thank You For Watching!

Additional Information on Deepwave Digital

www.deepwavedigital.com

View Additional Webcasts at
www.analog.com/Webcasts

Ask Questions on EngineerZone
ez.analog.com/Webcasts

Search for ADI Parts on Arrow
www.arrow.com