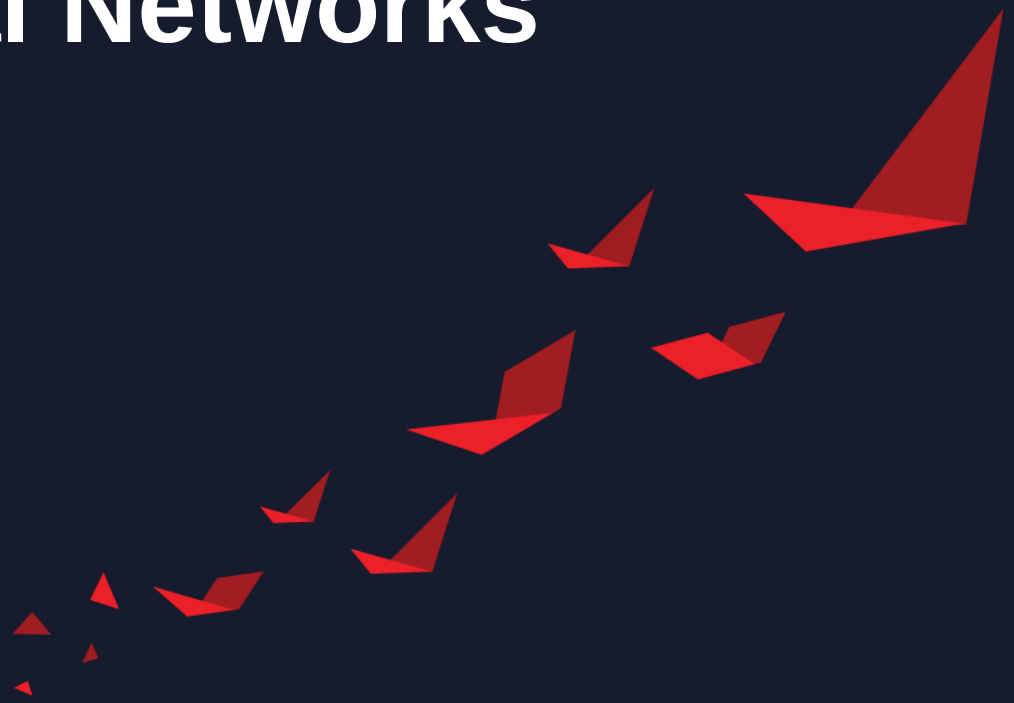
XILINX

2021 Challenge ITU AI/ML for 5G Lightning-Fast Modulation Classification with Hardware-Efficient Neural Networks

Michaela Blott
Alessandro Pappalardo
Yaman Umuroglu
Felix Jentzsch



Background

- **Xilinx**
  - Fabless semiconductor company, founded in Silicon Valley in 1984
  - Today: ~4000 employees, $3B revenue
  - Invented the FPGA
- **Xilinx Research - Dublin**
  - Established almost 15 years ago
  - ~10 researchers plus university program
  - Highly active internship program
- **Focus: FPGAs in Machine Learning**

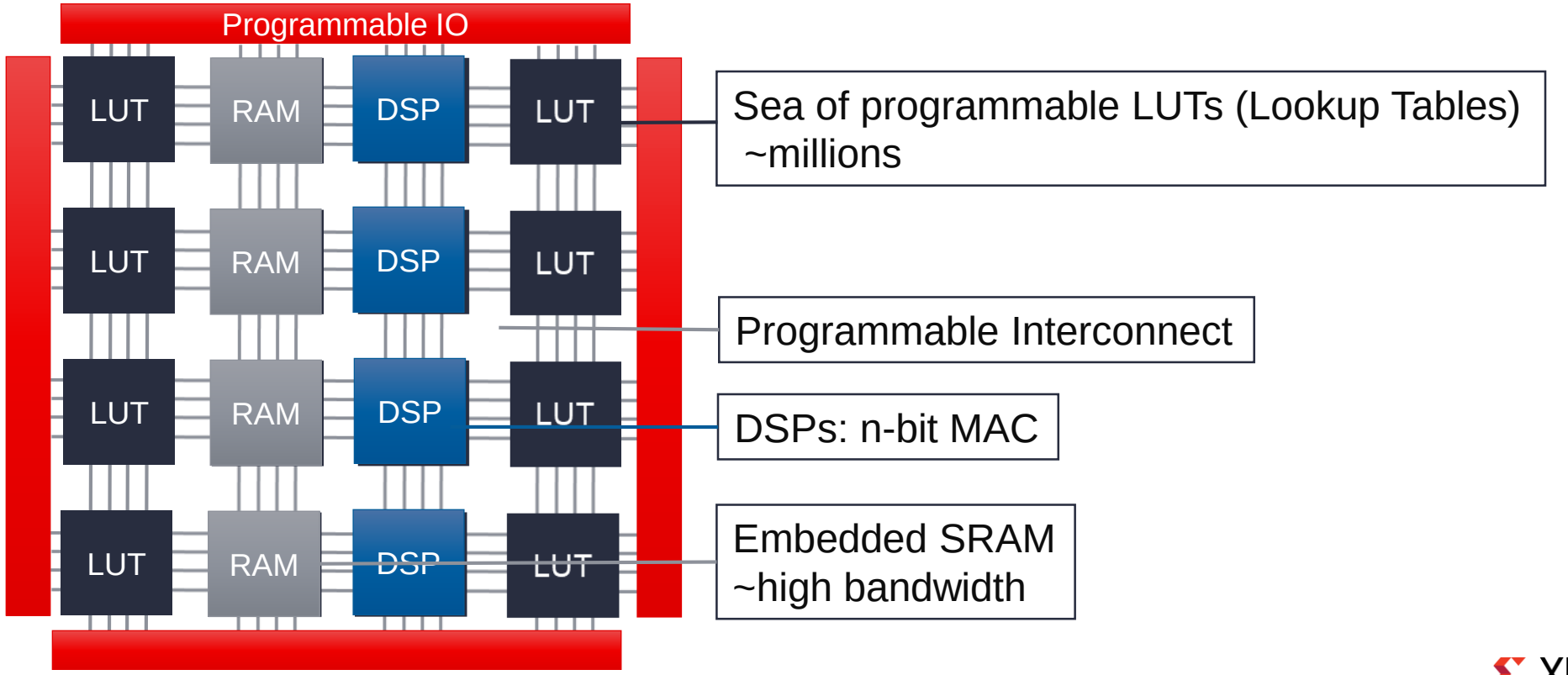
What are FPGAs?
Customizable, Programmable Hardware Architectures

The **chameleon** amongst the semiconductors…

Customizes IO interfaces, compute architectures, memory subsystems

to meet the specific application requirements

FPGA:

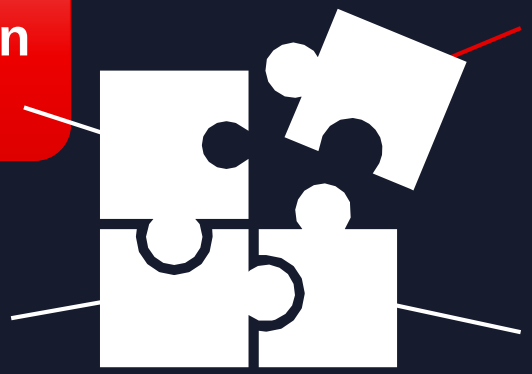


XILINX.

Goal: Enabling many future DNN-based RF applications with extreme throughput and ultra-low latency

Enabling Lightning-Fast ML+Radio

RadioML Modulation Classification
Great showcase of what's possible



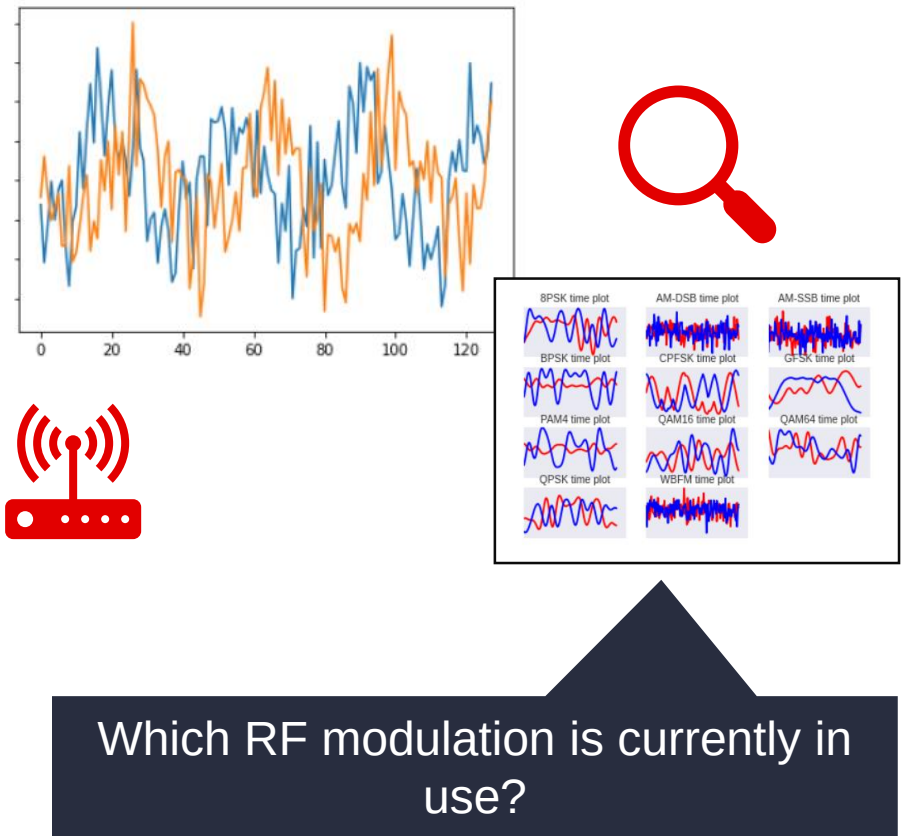
Challenge:
Your optimized DNNs

RFSoC
Adaptable radio platform with 6GHz DAC & ADCs and FPGA

FINN
DNN-to-FPGA compiler

Yaman Umuroglu

Modulation Classification: what's in my RF spectrum?



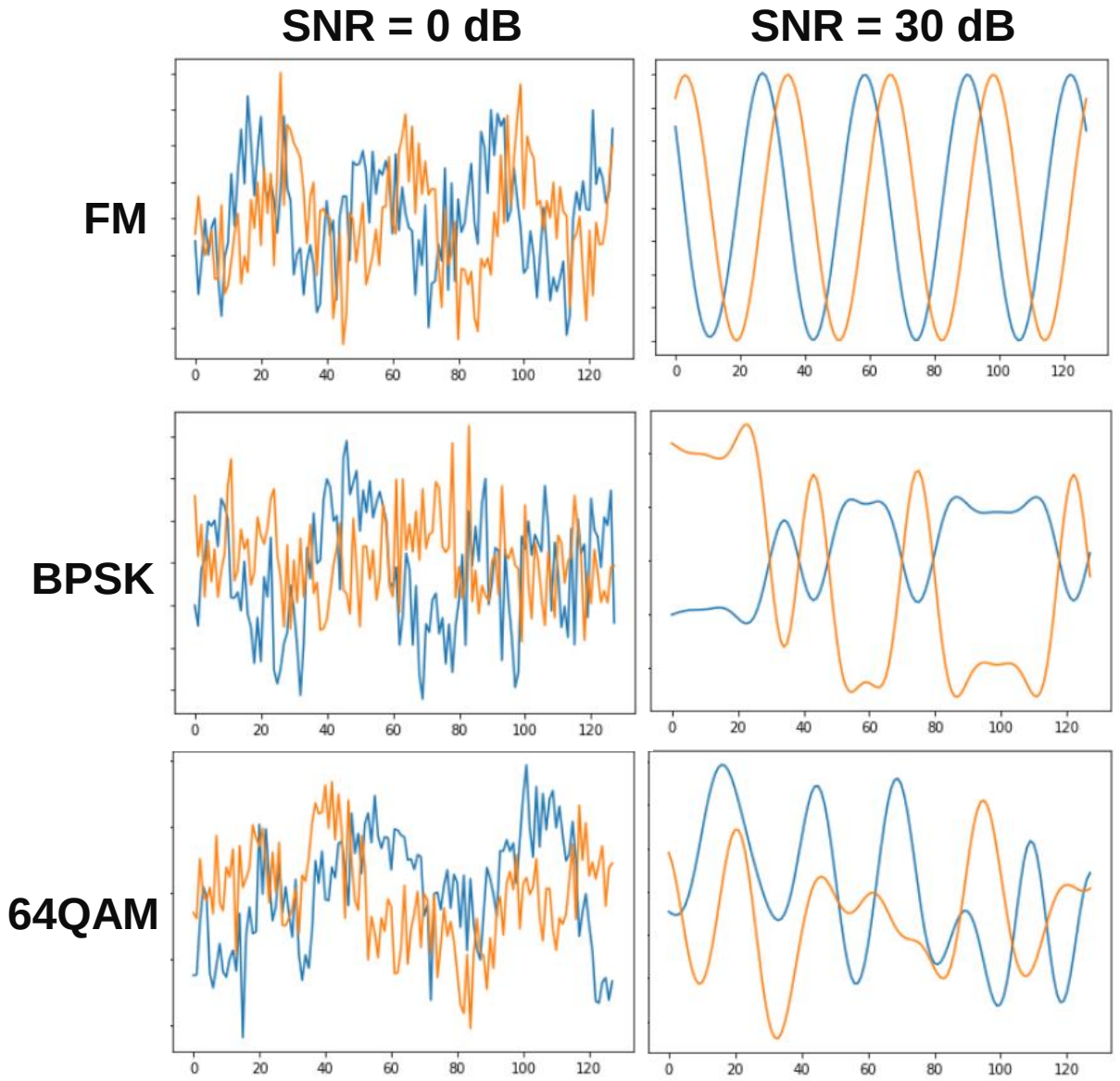
- Rapidly label + understand RF spectrum
- Key enabler for...
  - spectrum interference monitoring
  - radio fault detection
  - dynamic spectrum access
  - numerous regulatory and defense applications
- DNNs promising for modulation classification
  - Especially for short-time observations

[O'Shea et al., Over the Air Deep Learning Based Radio Signal Classification, IEEE JSTSP'17]

XILINX.

Dataset: RadioML 2018.01A

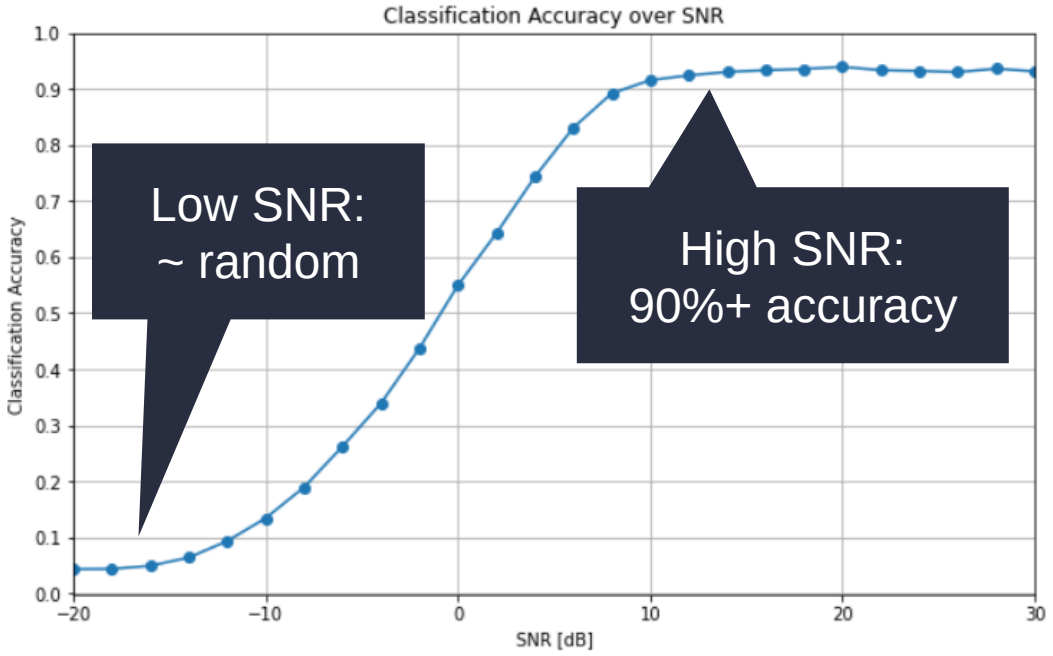
- Provided by **DeepSig**
  - Licensed under CC BY-NC-SA 4.0
- Rich data for modulation classification
  - 24 modulation types
  - @ 26 discrete SNRs (-20 dB to +30 dB)
  - Total of 2.5M examples
- 1024 complex samples per frame
  - As in-phase/quadrature (I/Q) components
  - Shape: 1024x2



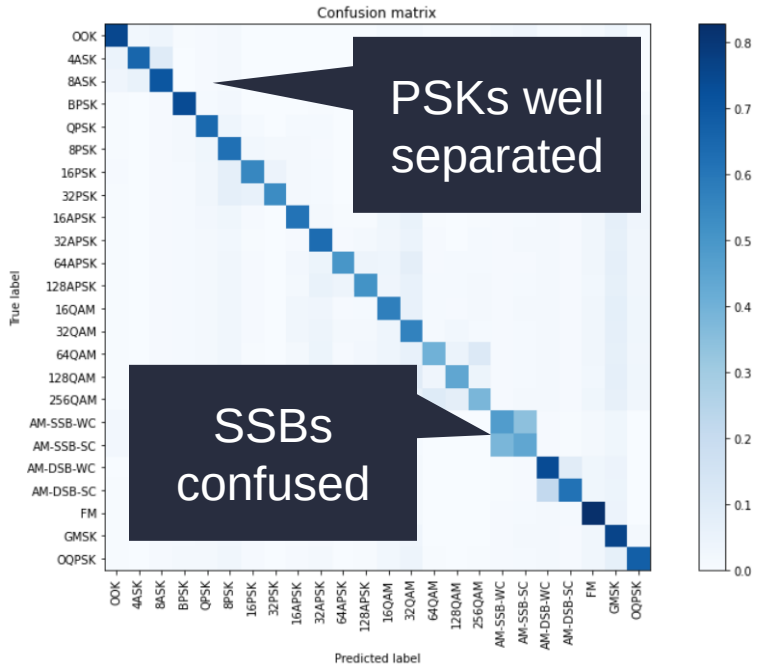
Examples of frame segments

XILINX.

Challenges: Accuracy is SNR- and Modulation-Dependent



Typical average accuracy over SNR

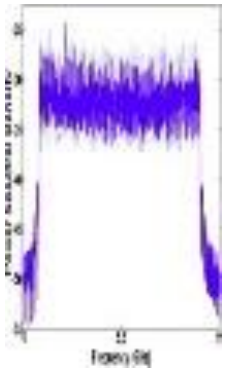


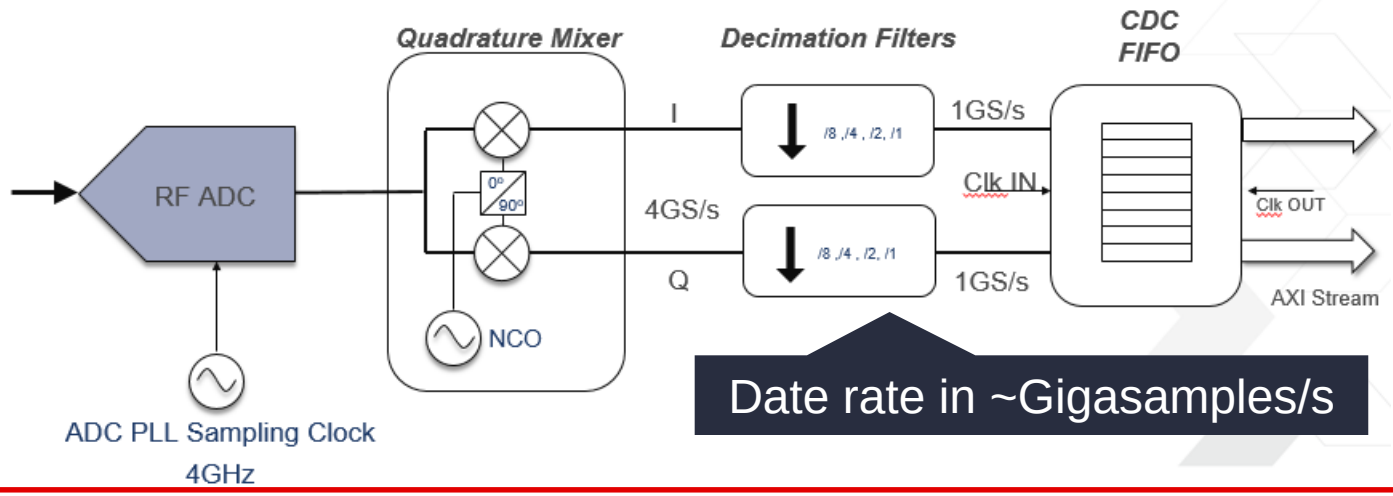
Typical confusion matrix

Need lots of data, careful training and testing

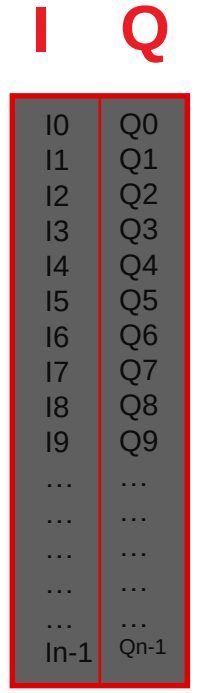
XILINX.

Challenges: Inference Throughput





Date rate in ~Gigasamples/s

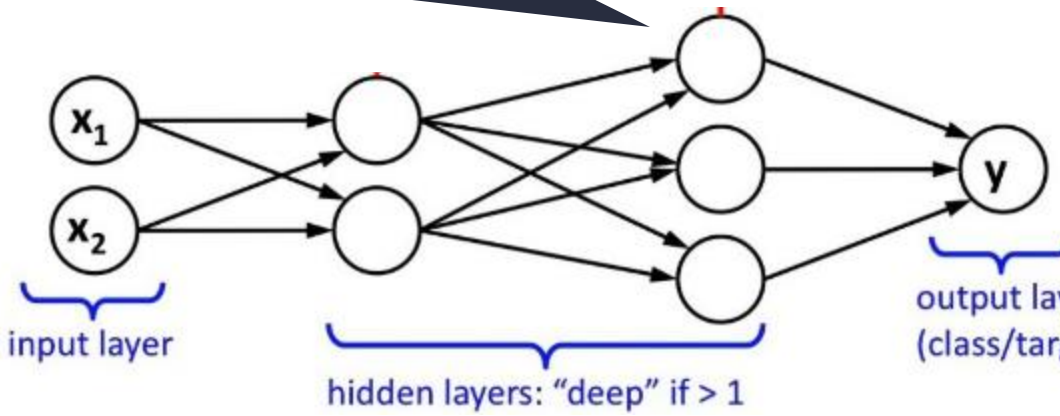


2x1024 samples

RF IN **Sample** **Mix and Down Convert** **Arrange Baseband Samples into a "Frame"**

Neural Network

DNN inference: large compute+memory footprint



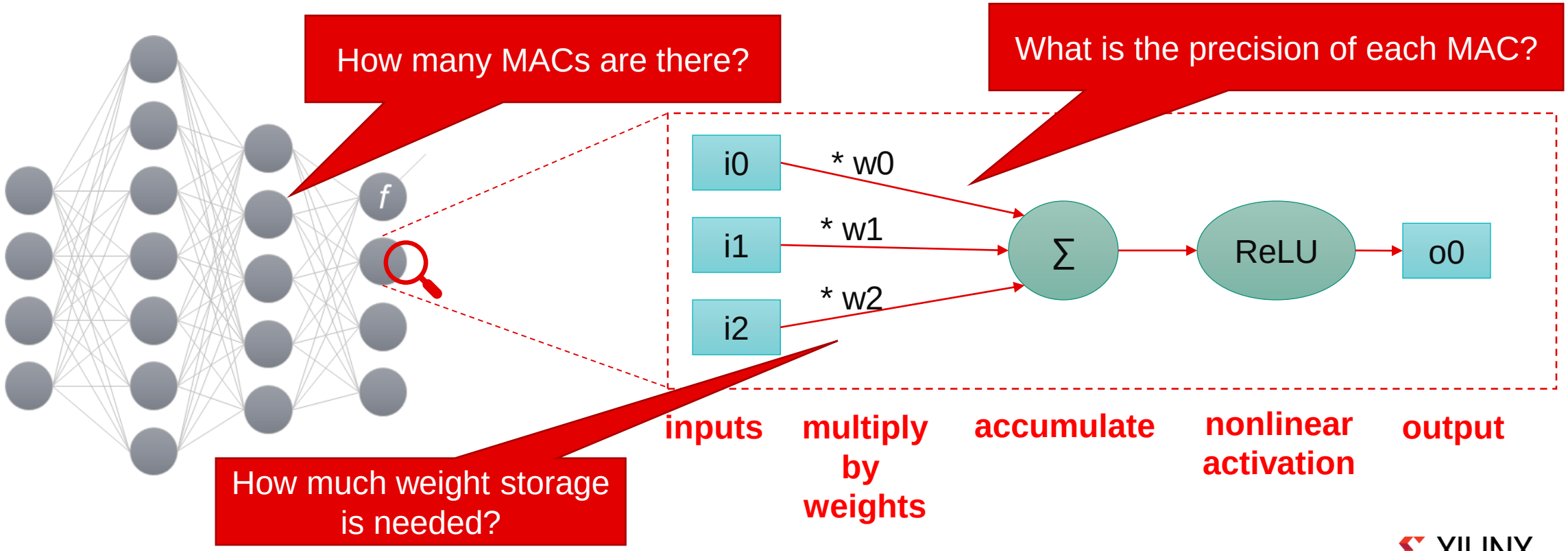
Output: Modulation Type

Sample-rate inference throughput is a challenge!

XILINX.

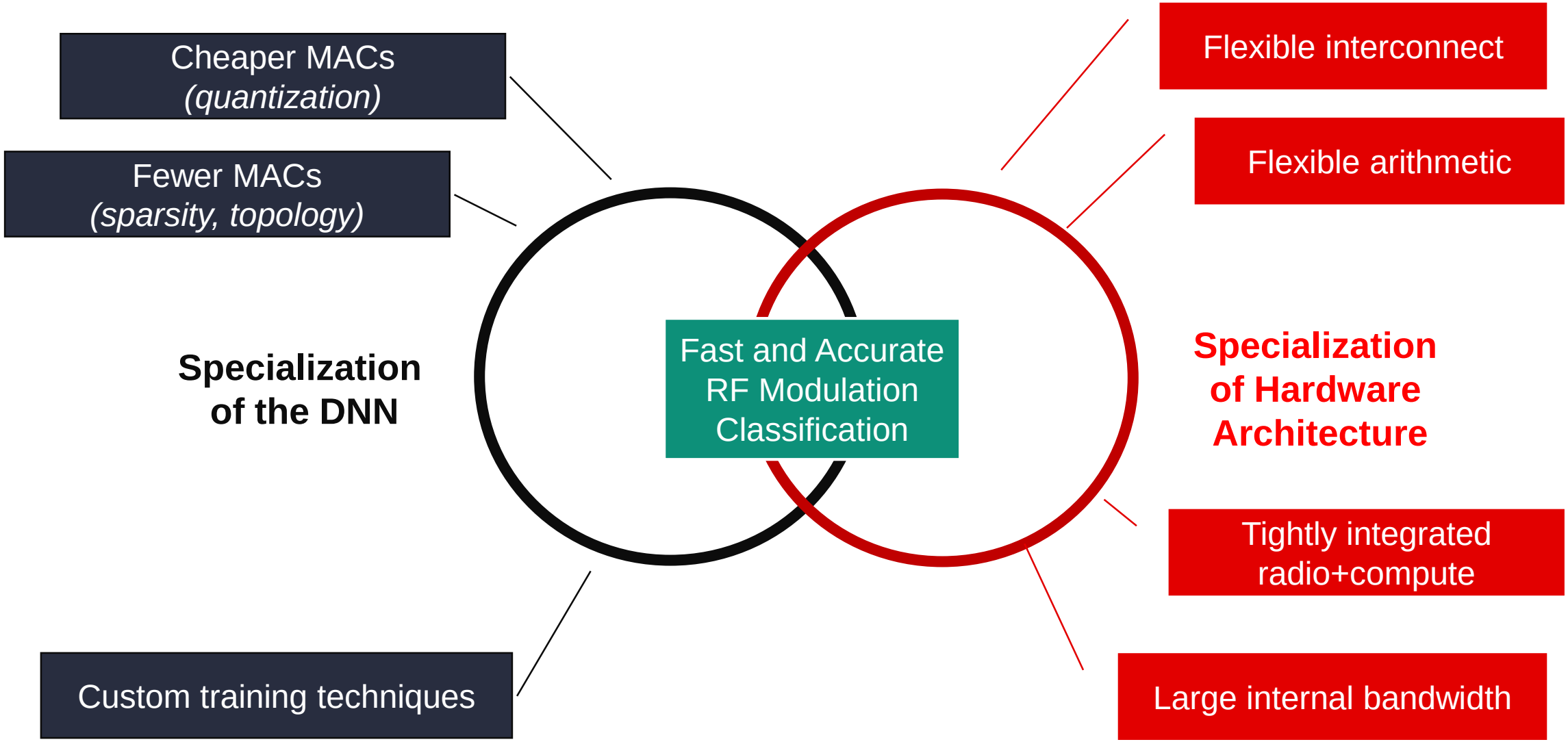
How fast can the DNN go?

- How much (time/energy/resources/...) is the **inference cost**?
  - Low cost = high throughput + low latency implementation becomes feasible
- Key determinants: cost of multiply-accumulate (MAC) operations and weights



XILINX.

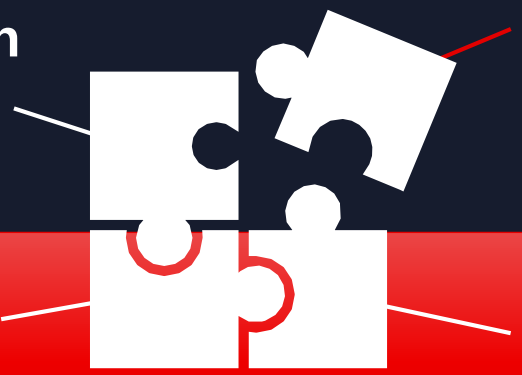
Specialization is essential



XILINX.

Enabling Lightning-Fast ML+Radio

RadioML Modulation Classification
Great showcase of what's possible



Challenge:
Your optimized DNNs

RFSoC
Adaptable radio platform with 6GHz DAC & ADCs and FPGA

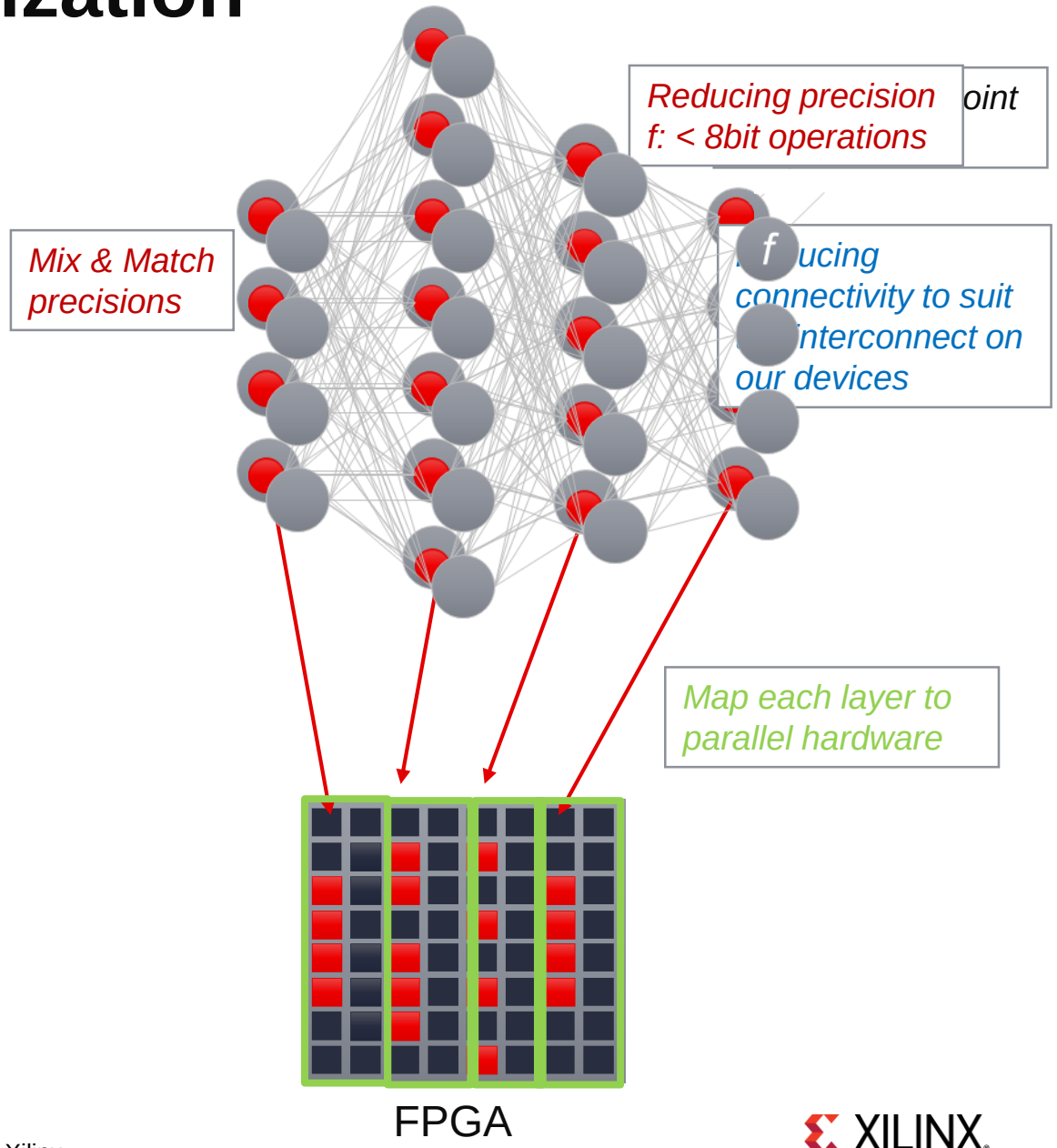
FINN
DNN-to-FPGA compiler

FPGAs are Masters of Specialization

- FPGAs can scale DNN performance through extreme specialization
- **Reduced precision quantized arithmetic**
  - **Arbitrary bitwidth**
  - **Mix & match bitwidths between layers**

Precision	MAC cost (LUTs)
1-bit	~1.1
2-bit	~4.4
4-bit	~17.6
8-bit	~70.4

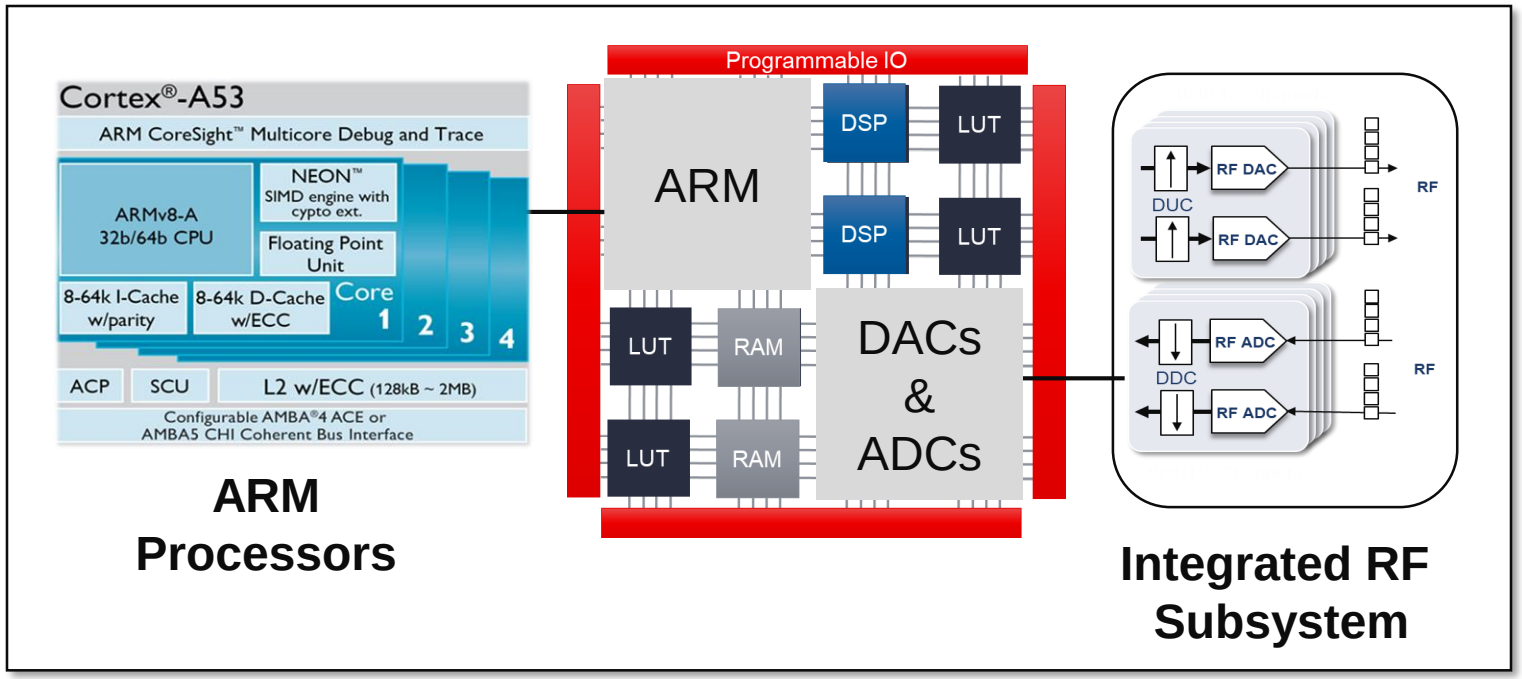
- **Fine-grained sparsity**
- **Layer-parallel dataflow implementation**



XILINX

What is RFSoC?
RFSoC= FPGA + ARM + DACs & ADCs

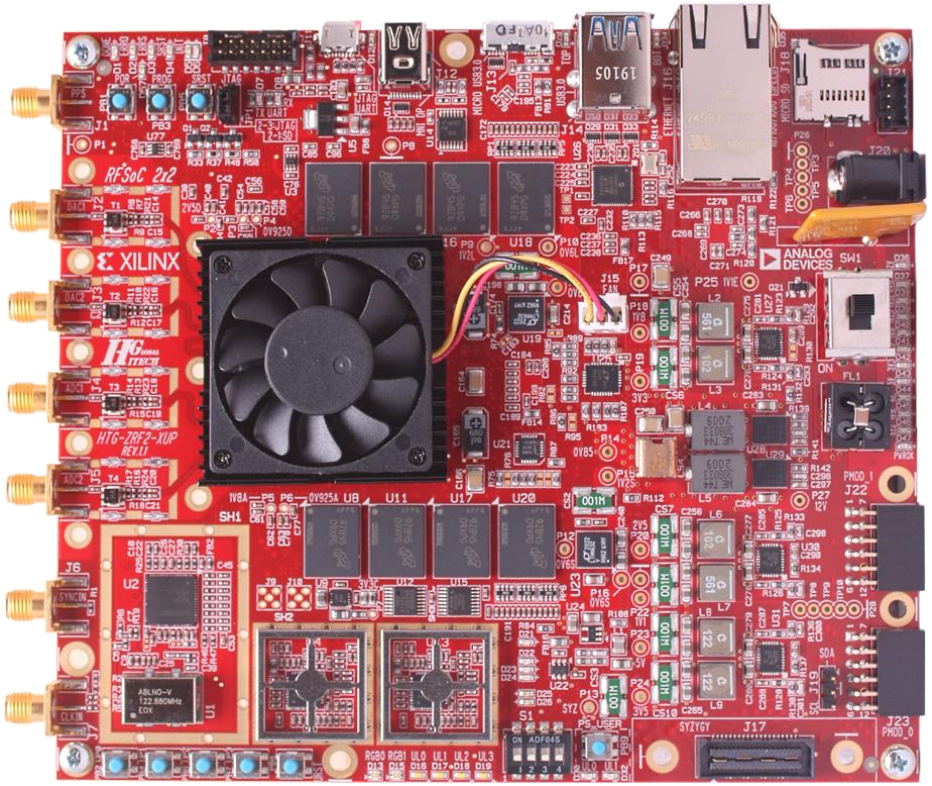
- Single-Chip Adaptable Radio Platform with Full Sub-6GHz Support
  - Elimination of analog RF signal processing and associated impairments
  - Increased flexibility to support wider bandwidths and multiple operating RF bands
  - Enablement of a software defined radio front end



XILINX.

Xilinx University Program: RFSoC 2x2 Project

- Affordable RFSoC 2x2 kit at $1,899 for academics
  - 2 RF DAC and 2 RF ADC channels
- PYNQ framework with JupyterLab IDE for exceptional ease-of-use
- Open-source resources including
  - Tutorial materials
  - Executable notebooks
  - Design examples

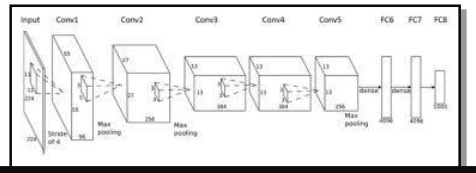


http://www.rfsoc-pynq.io/

DNN to FPGA Solution Stack

FINN

GitHub https://xilinx.github.io/finn/



Brevitas
Training in PyTorch
algorithmic optimizations

- Train quantized DNNs
- Library of standard layers
- Pretrained examples

ONNX Intermediate Representation

Focus of this challenge

FINN compiler
Specializations of
hardware architecture

- Hardware optimizations
- Generates dedicated FPGA accelerator

Deployment

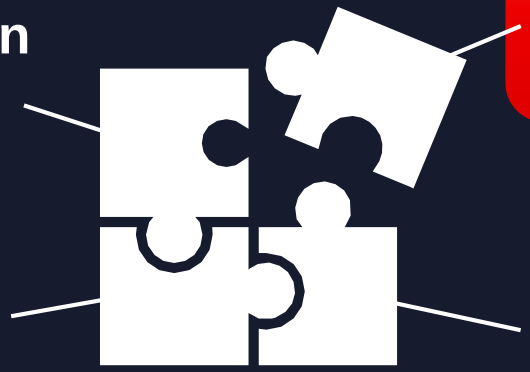
- Integrate accelerator into system and deploy

XILINX

Enabling Lightning-Fast ML+Radio

RadioML Modulation Classification
Great showcase of what's possible

RFSOC
Adaptable radio platform with 6GHz DAC & ADCs and FPGA

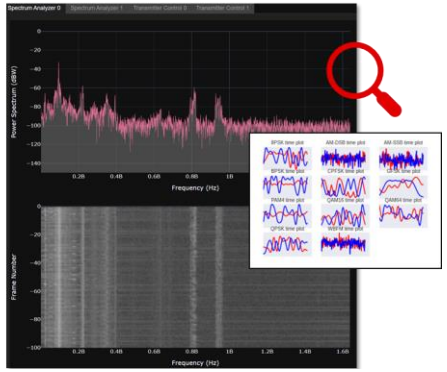


Challenge:
Your optimized DNNs

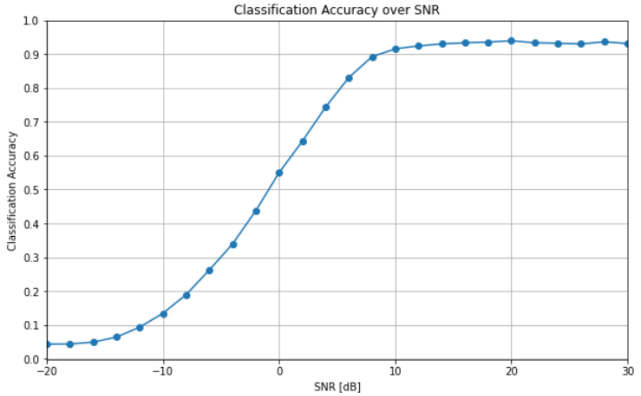
FINN
DNN-to-FPGA compiler

The Challenge

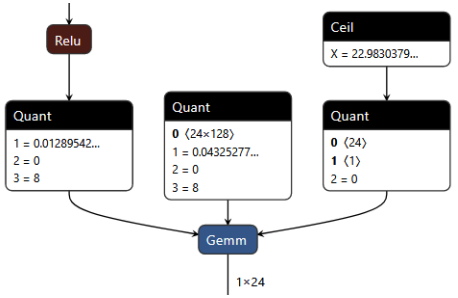
Your team



Train a DNN on
RadioML 2018.01A



Achieve at least **56.000%** average accuracy over full SNR range



Minimize **inference cost**

XILINX

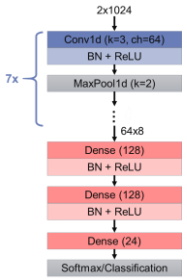
Our team

Brevitas

PyTorch

docker jupyter

Sandbox environment for
quantization-aware training



Provide training script for
baseline model

Support, evaluate and rank the **submissions**

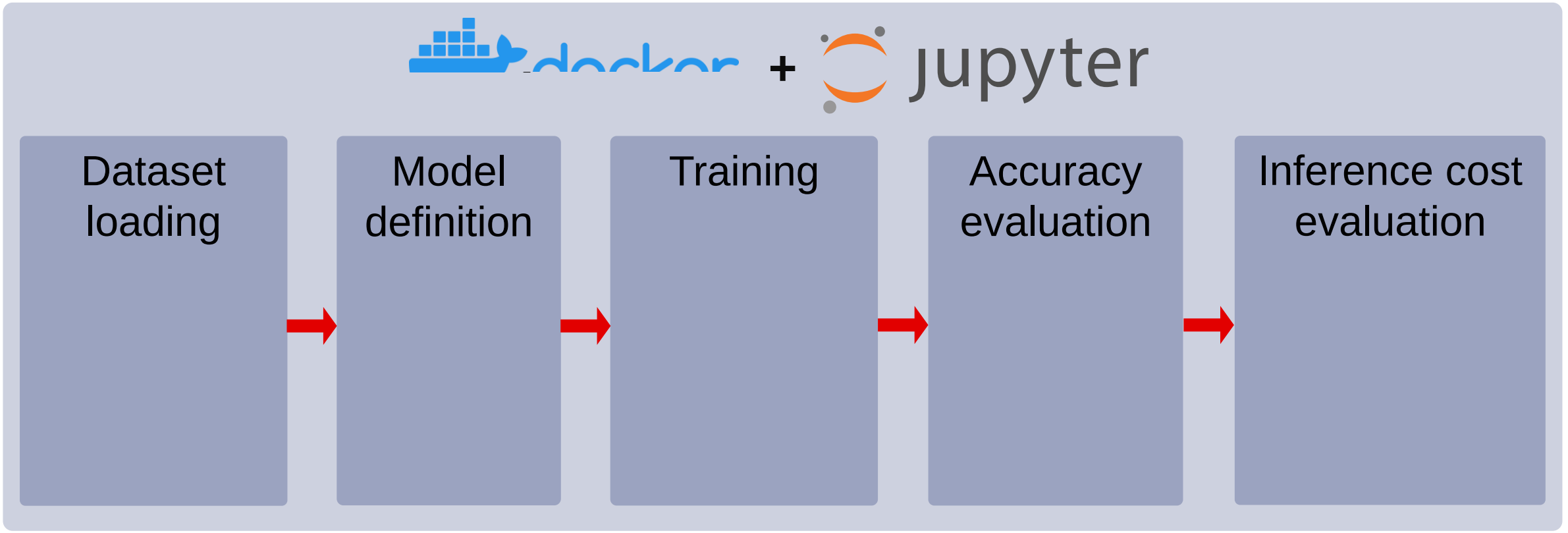
XILINX

What we provide

Sandbox

https://github.com/Xilinx/brevitas-radioml-challenge-21

- Reproducible environment including PyTorch and Brevitas
- Starting point: end-to-end guide for a baseline CNN



Inference cost: theory

Only MAC-intensive layers!
(Conv, FC)

ß Ô ì Ø åÆæº ¼ æ

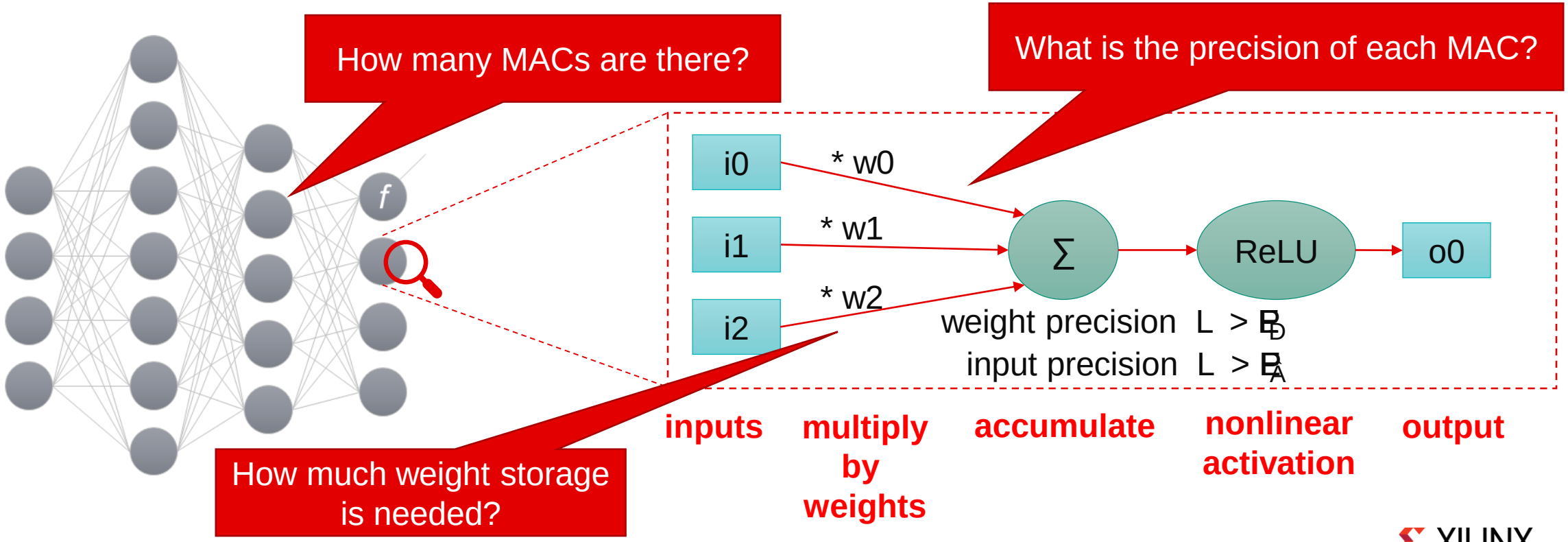
Computation cost >E4RK L O Í : Í > EB „ > EA;

ß Ô ì Ø åæø Ü Ú Û ç æ

Memory cost >E4PA I L Í : Í > EB ;

Normalized score (lower is better)

O ? K lN rÄäw„ $\frac{\text{> E4RP L O}}{\text{> = O A H4>EE4RPA L O}}$ E r äw„ $\frac{\text{> E4IPA I}}{\text{> = O A H4>EE4IPA I}}$



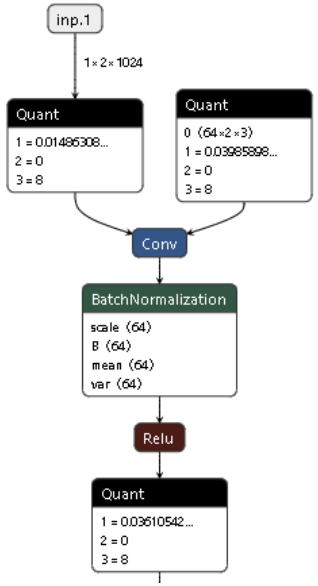
XILINX.

Inference cost: practice

- Measure inference cost using provided functionality in the sandbox

Brevitas + PyTorch

export()



inference_cost()

Shows how many MACs + weights at which precision

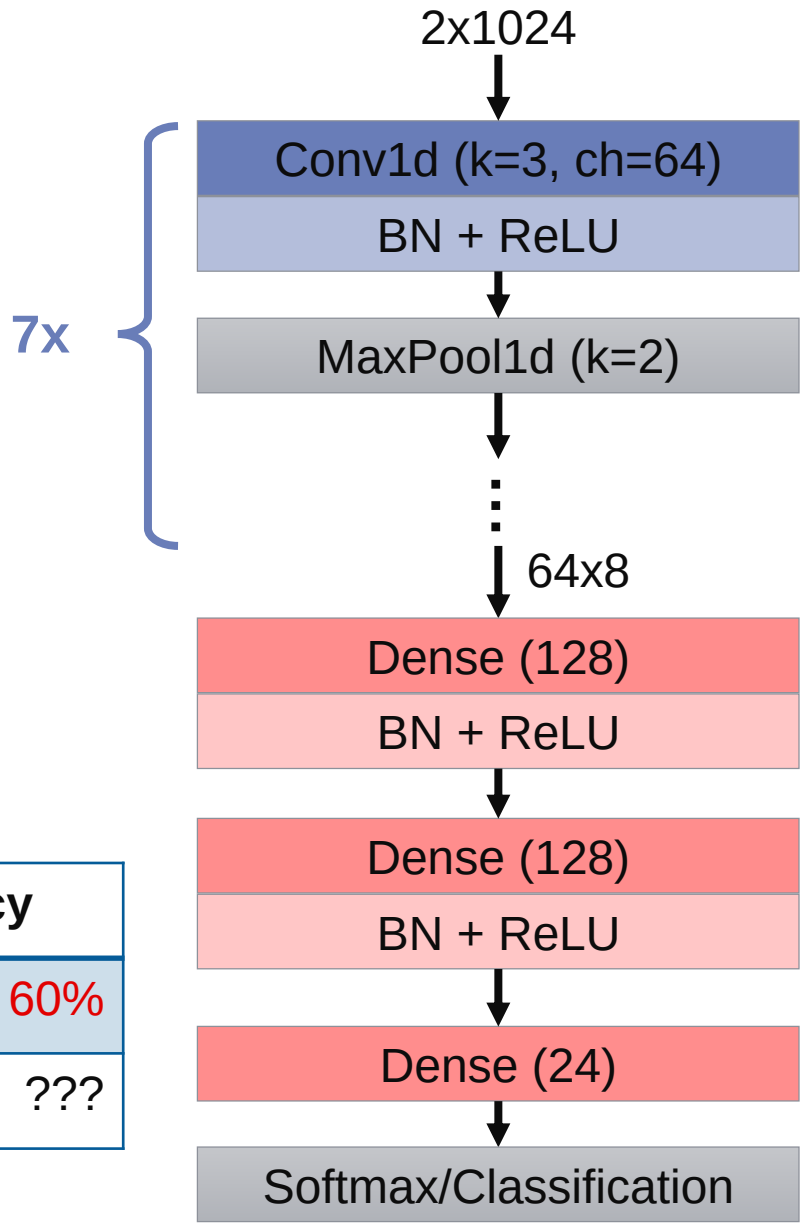
```
# detailed report
{
    " discount   _sparsity     " :   true   ,
    " mem_w_SCALEDINT8":   155617.0   ,
    " op_mac_SCALEDINT8_SCALEDINT8":   388096.0   ,
    " op_mac_SCALEDUINT8_SCALEDINT8":   12232215.0   ,
    " total_bops     " :   807699904.0   ,
    " total_mem_w_bits     " :   1244936.0   ,
    "unsupported"     :   "set()"
}

# final (baseline          - normalized) score
score = 1.0
```


XILINX.

Baseline neural network

- “VGG10” 1D-CNN
- Topology proposed by dataset creators
- Proven performance against traditional classification techniques
- Quantized to 8-bit input/weights/activations
- **Score for ranking is normalized to this baseline**

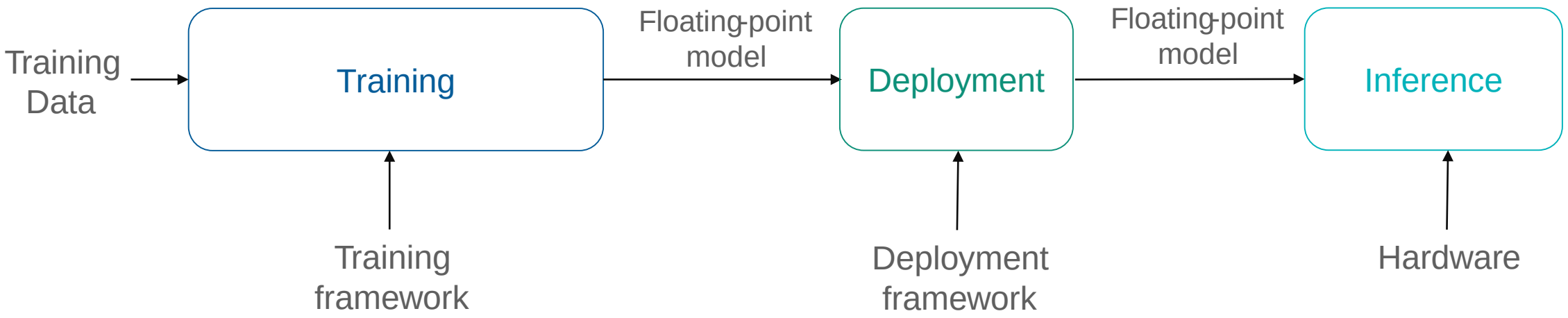
Quantization	>E4RKLO	>E4PAI	**Score**	**Accuracy**
8-bit	808M	1.2M	1.0	60%
4-bit	172M	485k	0.3	???



How to minimize inference cost?

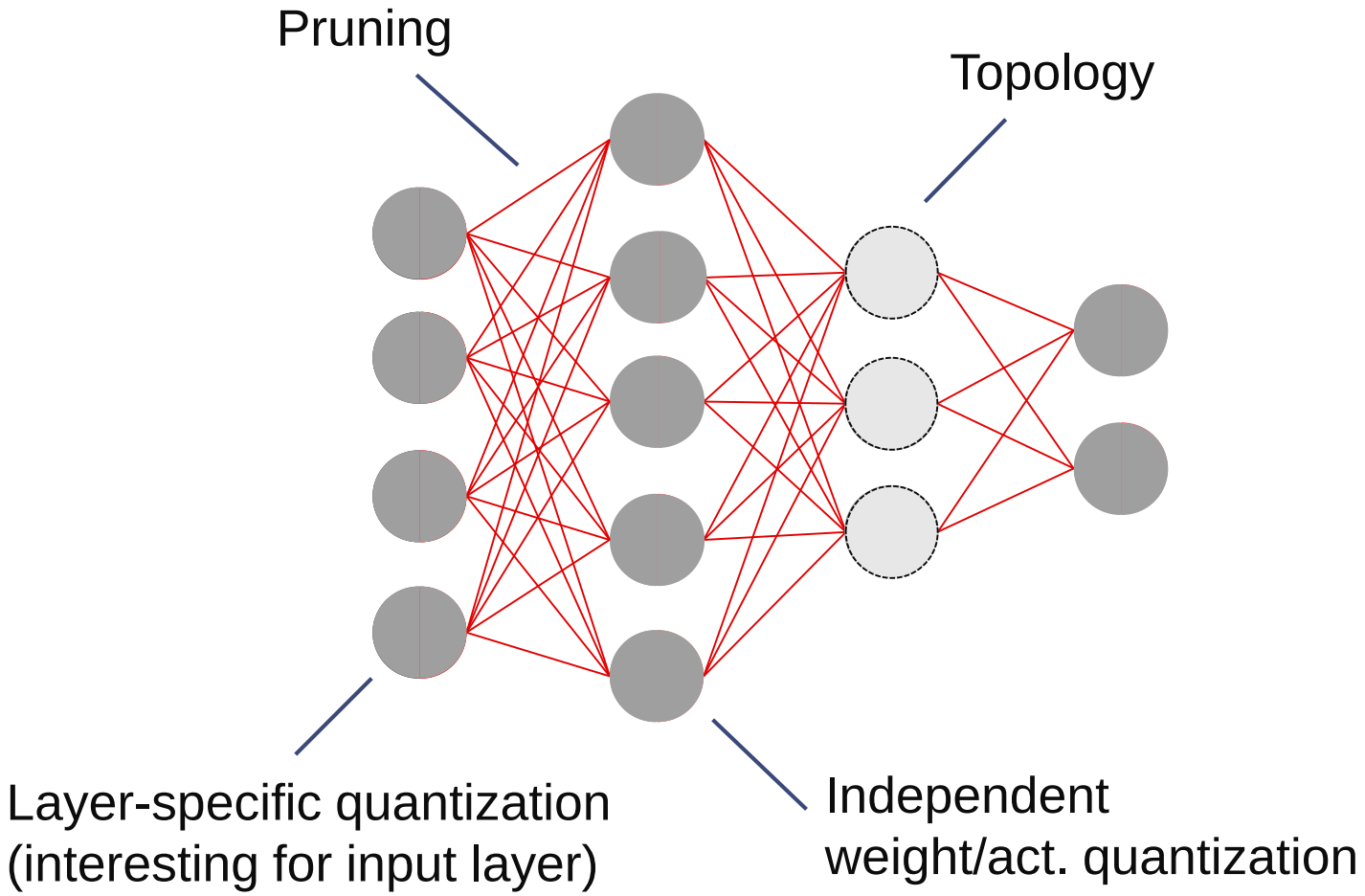
Alessandro Pappalardo

Standard training scenario

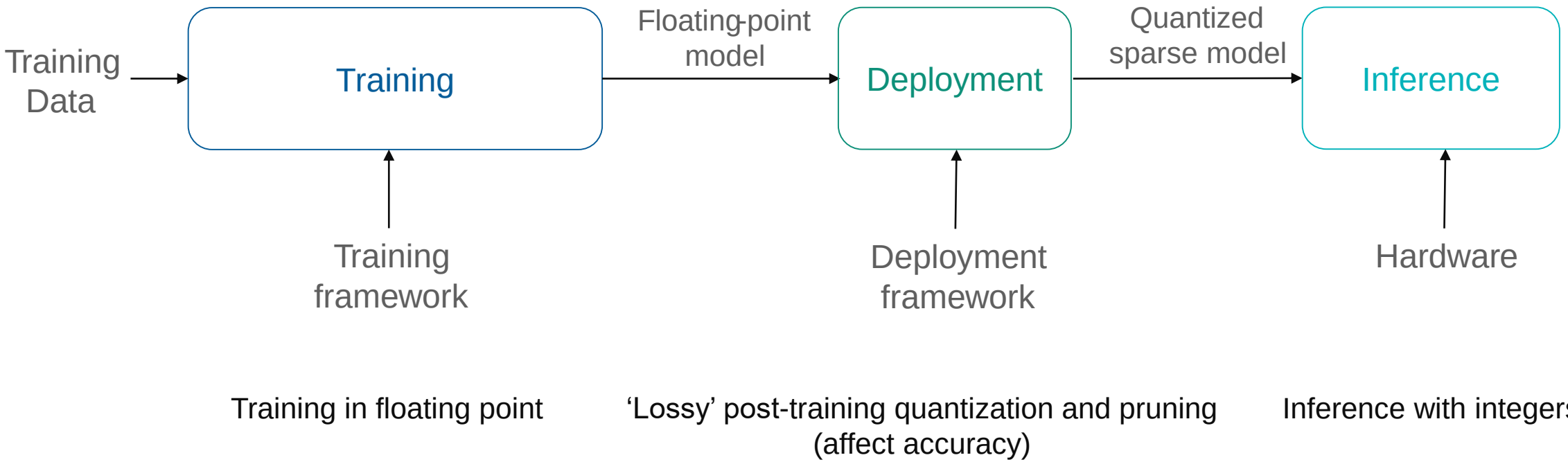


- High accuracy but not tuned for inference cost

Topology changes, quantization and pruning

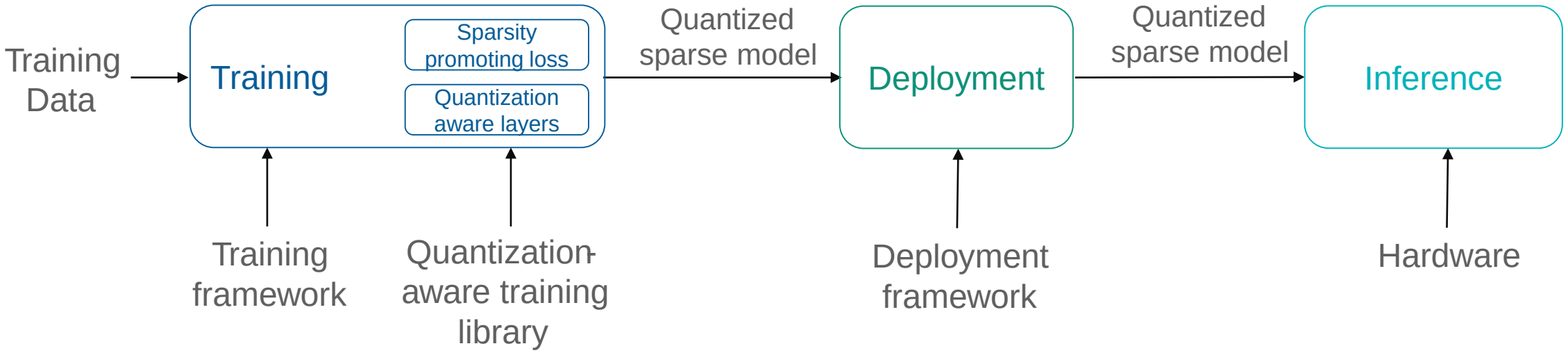


Standard training + post-training quantization/pruning



- Quantization and sparsity are introduced after training
- Limited opportunities for reducing inference cost without affecting accuracy

Quantization and sparsity aware training



Training in floating point with integer forward pass

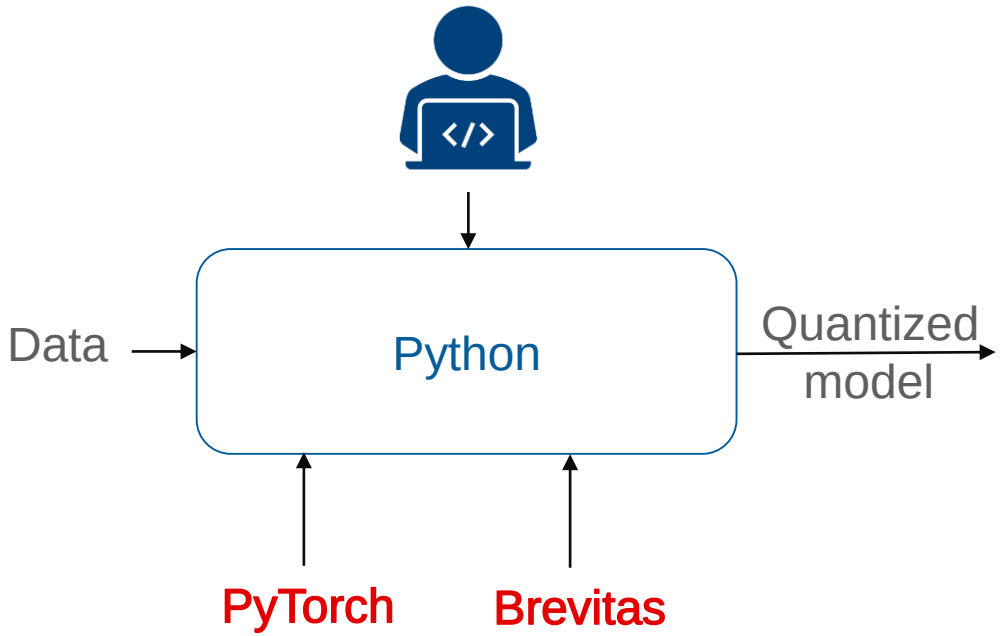
'Lossless' post-training optimizations (preserve accuracy)

Inference with integers

- Quantization and pruning are introduced during training
- More opportunities to balance accuracy with inference cost

Brevitas

- Brevitas is:
  - An open-source PyTorch library for quantization-aware training
  - Designed from the ground-up to be extended
  - Incremental complexity to trade off ease-of-use with finer control
  - Growing open source community with over 1500 downloads a month

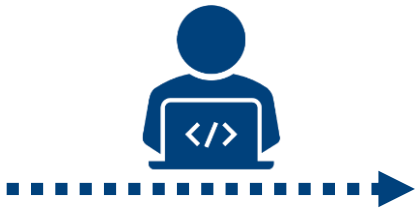


Core features

- Easy to apply and extend in Python
  - Large library of composable building blocks, easily extendible in Python
- Deep integration with the PyTorch ecosystem
  - Quantized tensors and layers can be freely mixed with floating-point tensor and layers
- Orthogonal to other tools and techniques for training efficient NNs
  - Sparsity promoting regularization
  - Custom efficient layers
- Efficient at training time on CPUs and GPUs
  - Extensive use of PyTorch own JIT compiler for a supported subset of Python (TorchScript)

Example: floating-point to 3b4b MLP in PyTorch

```
# Define float MLP
nn.Sequential (
    nn.Flatten (),
    nn.Linear (
        flat_input_size , hidden_size ,
        bias =True ),
    nn.ReLU(),
    nn.Dropout ( 0.1 ),
    nn.Linear (
        hidden_size , hidden_size ,
        bias =True ),
    nn.ReLU(),
    nn.Dropout ( 0.1 ),
    nn.Linear (
        hidden_size , num_classes ,
        bias =False ))
```



```
# Define quantized MLP
nn.Sequential (
    nn.Flatten (),
    qnn. QuantIdentity ( bit_width =4),
    qnn.QuantLinear (
        flat_input_size , hidden_size ,
        bias =True ,
        weight_bit_width =3),
    qnn.QuantReLU( bit_width =4),
    nn.Dropout ( 0.1 ),
    qnn.QuantLinear (
        hidden_size , hidden_size ,
        bias =True ,
        weight_bit_width =3),
     qnn.QuantReLU( bit_width =4),
    nn.Dropout ( 0.1 ),
    qnn.QuantLinear (
        hidden_size , num_classes ,
        bias =False ,
        weight_bit_width =3))
```

Logistics

Michaela Blott

Why submit to this challenge?

- Glory! We'll host a online workshop where top participants present
  - Award ceremony with Ivo Bolsens (Xilinx CTO)
  - Publish results
- Top-3 teams win local prizes

- Internship opportunities (subject to candidate suitability)
- Opportunity to jointly publish with workshop organizers
- Winners progress to global round of the ITU AI/ML in 5G competition
  - Top candidates evaluated by ITU judging committee
  - ITU awards and presentations at final conference in Dec. 2021
  - Invited journal article

Timeline

Registration

Evaluation

29.6.2021

31.8.2021

2.10.2021

27.10.2021

Dec.2021

Competition starts
Registration opens

Registration closes

Submission

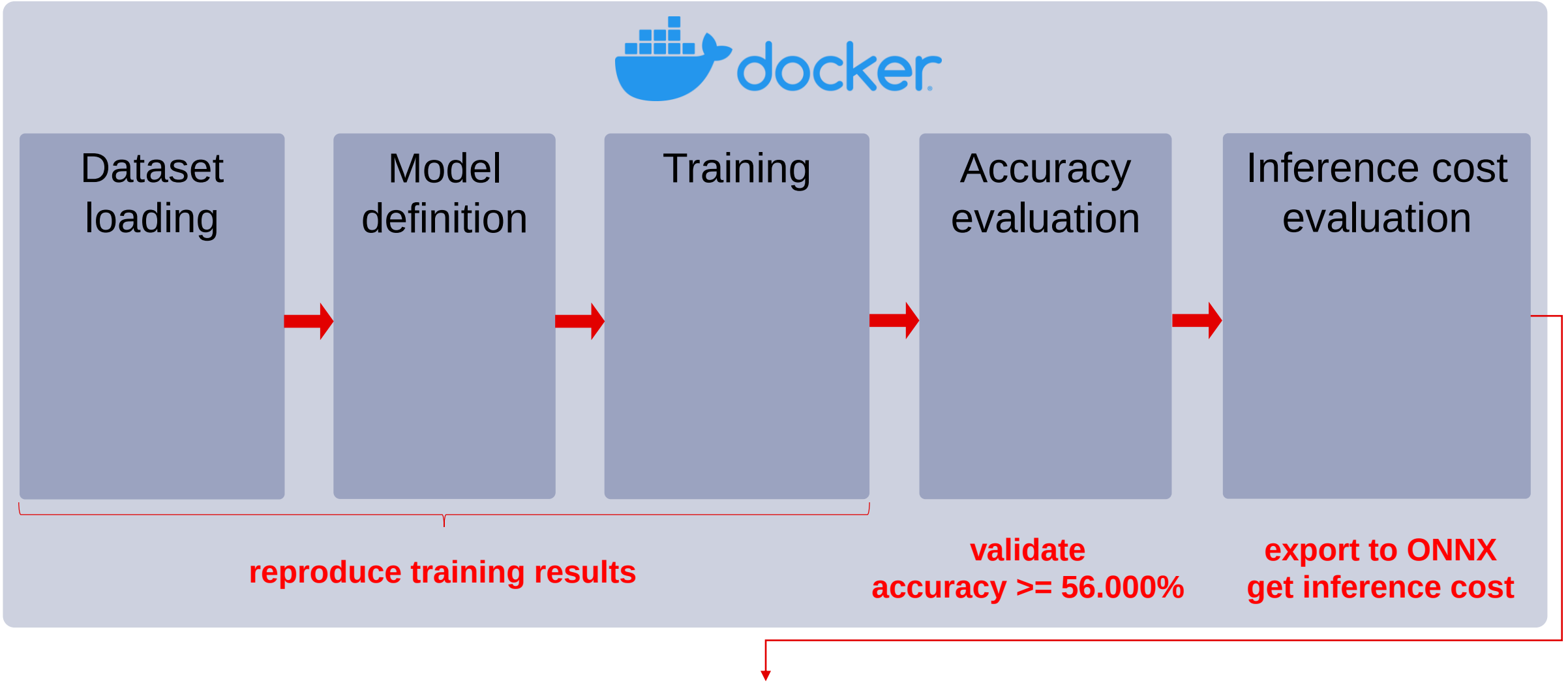
Winners progress

Workshop
Award Ceremony
Prizes

ITU Conference

XILINX.

Evaluation



General Rules & Submission Guidelines

- Register and submit on challenge portal (up to 4 people per team)
- Must be open source and end-to-end reproducible
- License restrictions
  - FINN & Brevitas open source
  - RadioML dataset (DeepSig Inc.) available under the Creative Commons Attribution - NonCommercial - ShareAlike 4.0 License (CC BY-NC-SA 4.0).
    - Read and adhere to the licensing terms
    - If an alternative license is needed, please contact directly info@deepsig.io.
  - Reference in all relevant academic papers when using these datasets.
- Read Full Evaluation criteria on *https://bit.ly/brevitas-radioml-challenge-21*

Resources

- Website, registration and submission: https://challenge.aiforgood.itu.int/matchmanager/#/match/matchdetail/34/matchinfo
- Repository (sandbox): https://github.com/Xilinx/brevitas-radioml-challenge-21
- FINN project page: https://xilinx.github.io/finn/
- Support through GitHub discussion page: https://github.com/Xilinx/brevitas-radioml-challenge-21/discussions
- Contact us directly: mblott@xilinx.com, yamanu@xilinx.com, alessand@xilinx.com

All of this can also be found on https://bit.ly/brevitas-radioml-challenge-21 under the Resources tab

We’re looking forward to your submissions!

Thank You