QONNX

EDGE ML SCHOOL
23.–27. SEPT. 2024
MARIUS KÖPPEL

ETH zürich

Why ONNX (Open Neural Network Exchange)?

Caffe2

Why ONNX (Open Neural Network Exchange)?

Caffe2

PyTorch

Why ONNX (Open Neural Network Exchange)?

Caffe2

PyTorch

TensorFlow

Why ONNX (Open Neural Network Exchange)?

Caffe2

PyTorch

TensorFlow

Keras

Why ONNX (Open Neural Network Exchange)?

Caffe2

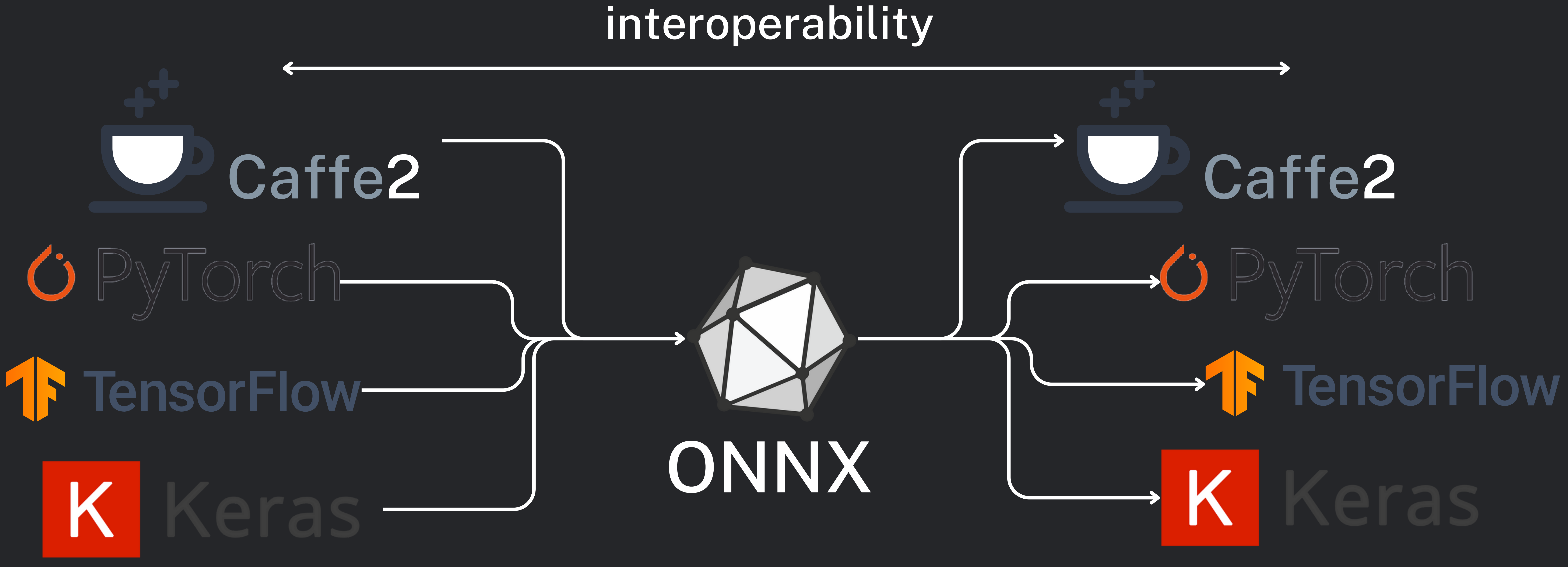
PyTorch

TensorFlow

Keras

And many many more

Why ONNX (Open Neural Network Exchange)?



export to onnx

load from onnx

Basic ONNX Graph

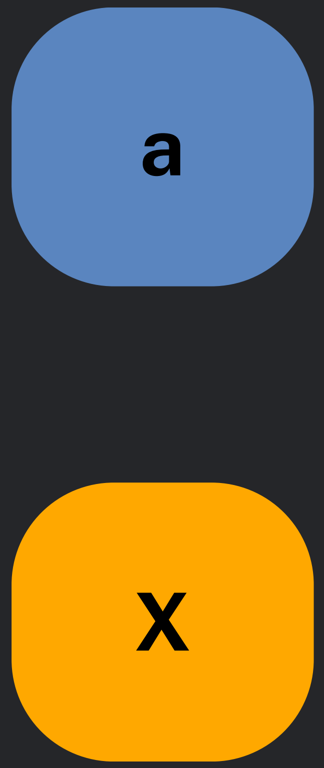
y = x @ a + c

```
def onnx_linear_regressor(x):
    y = onnx.Add(onnx.MatMul(x, a), c)
    return y
```

Basic ONNX Graph

y = x @ a + c

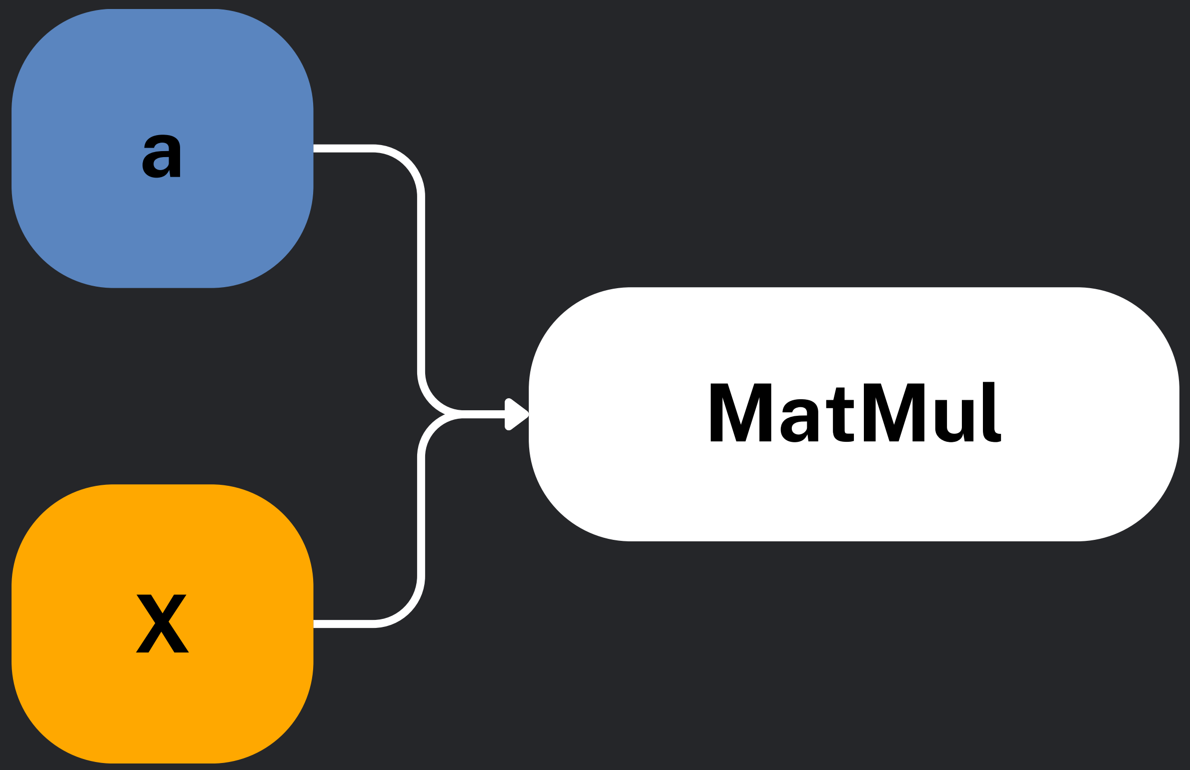
```
def onnx_linear_regressor(x):
    y = onnx.Add(onnx.MatMul(x, a), c)
    return y
```



Basic ONNX Graph

$y = x \mathbin{@} a + c$

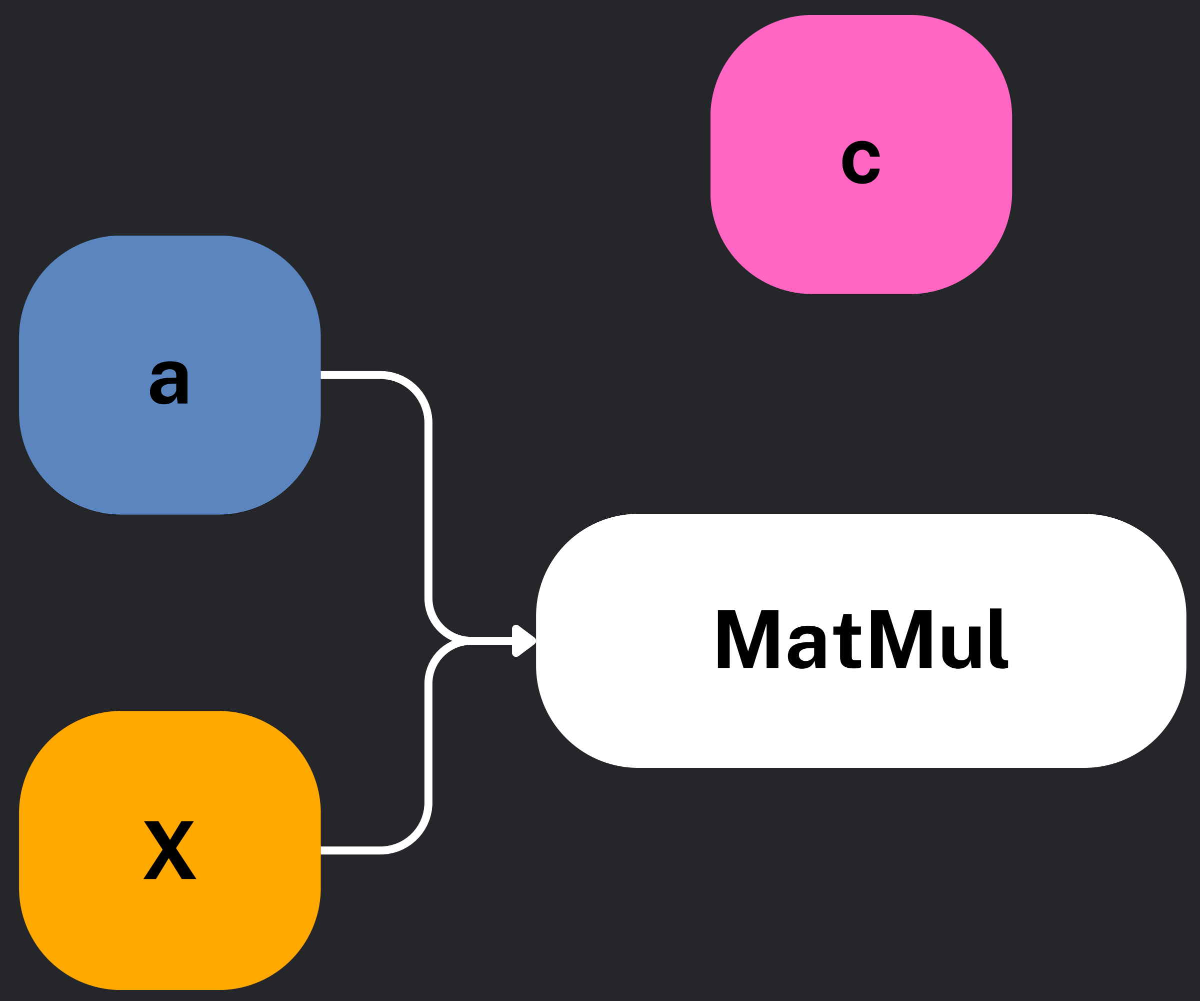
```
def onnx_linear_regressor(x):
    y = onnx.Add(onnx.MatMul(x, a), c)
    return y
```



Basic ONNX Graph

$y = x @ a + c$

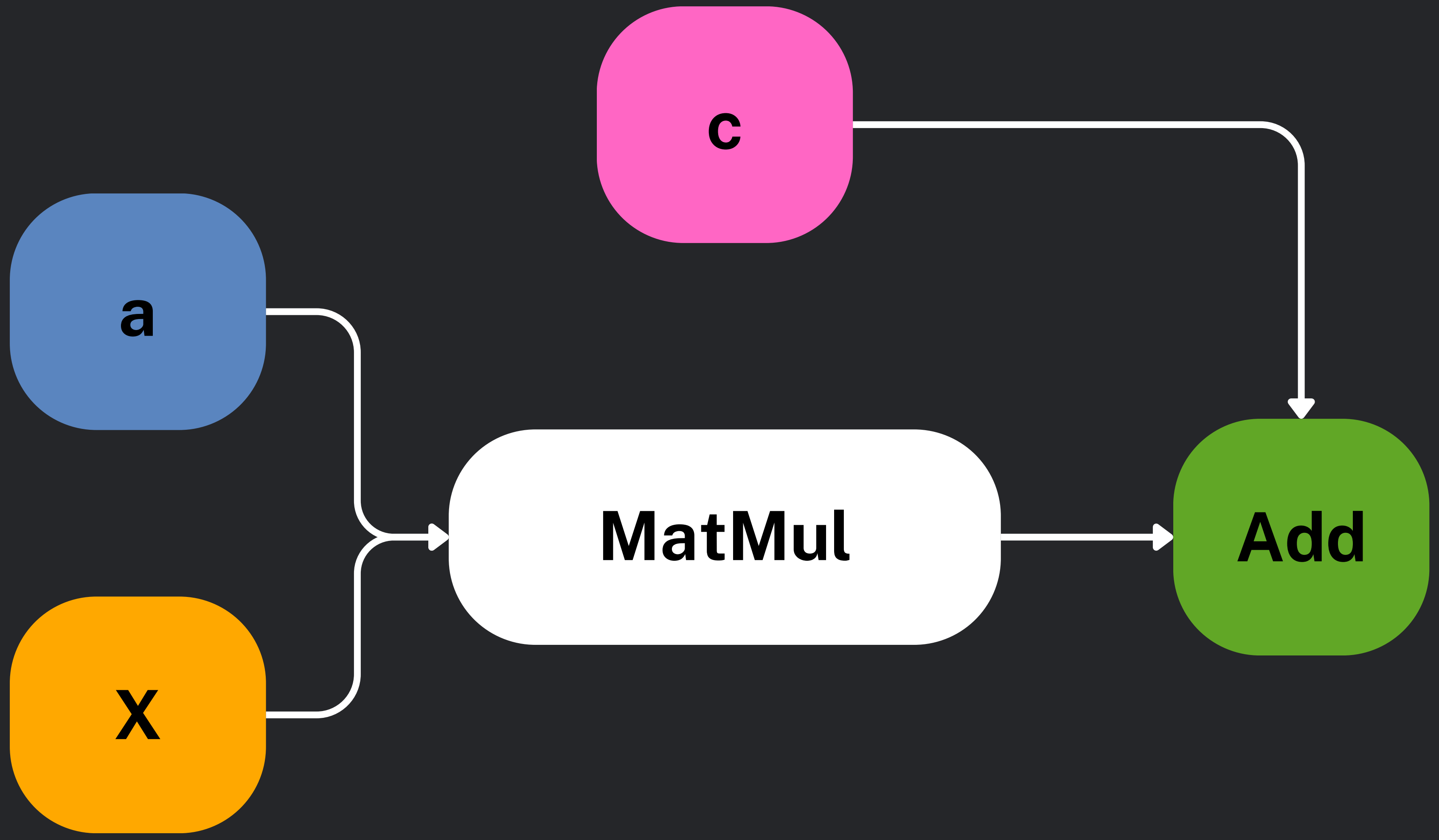
```
def onnx_linear_regressor(x):
    y = onnx.Add(onnx.MatMul(x, a), c)
    return y
```



Basic ONNX Graph

$y = x @ a + c$

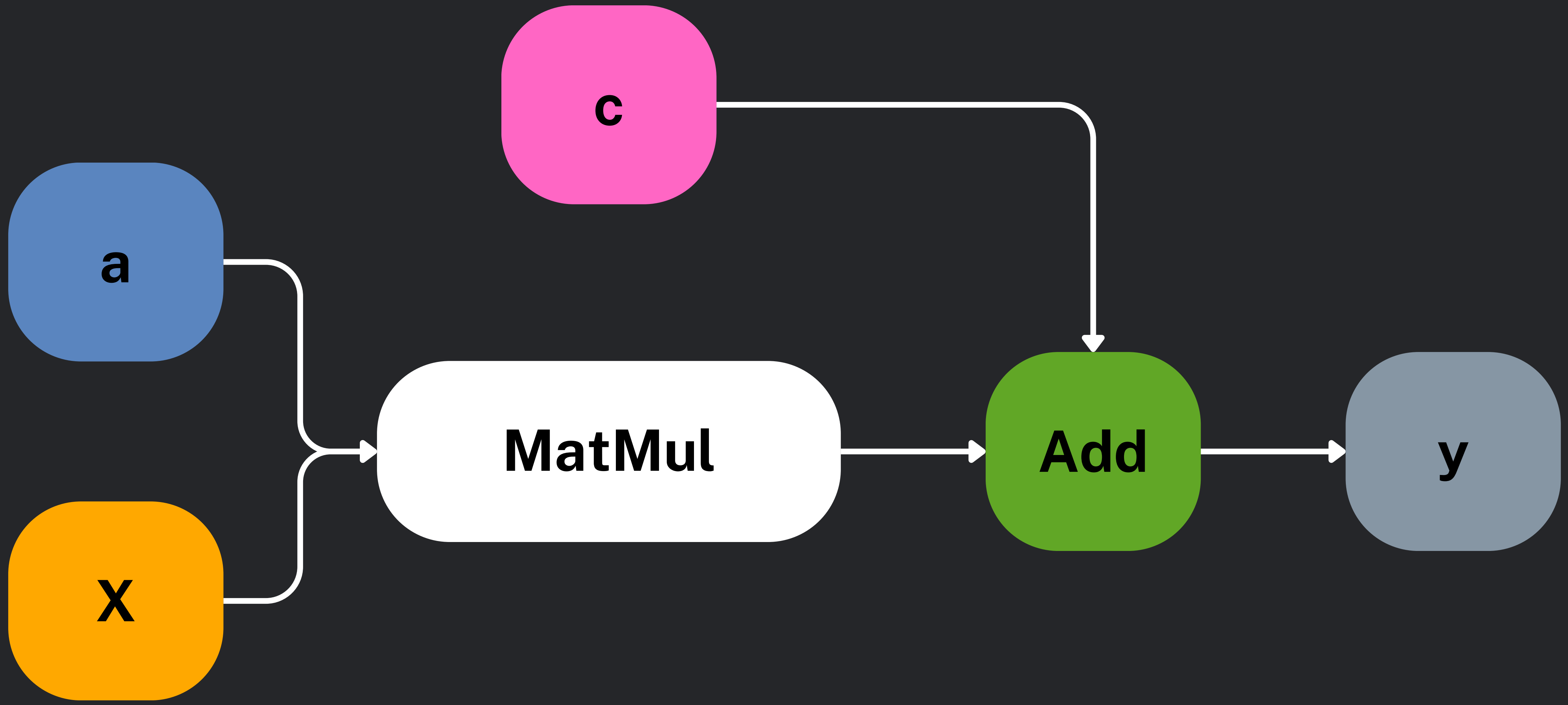
```
def onnx_linear_regressor(x):
    y = onnx.Add(onnx.MatMul(x, a), c)
    return y
```



Basic ONNX Graph

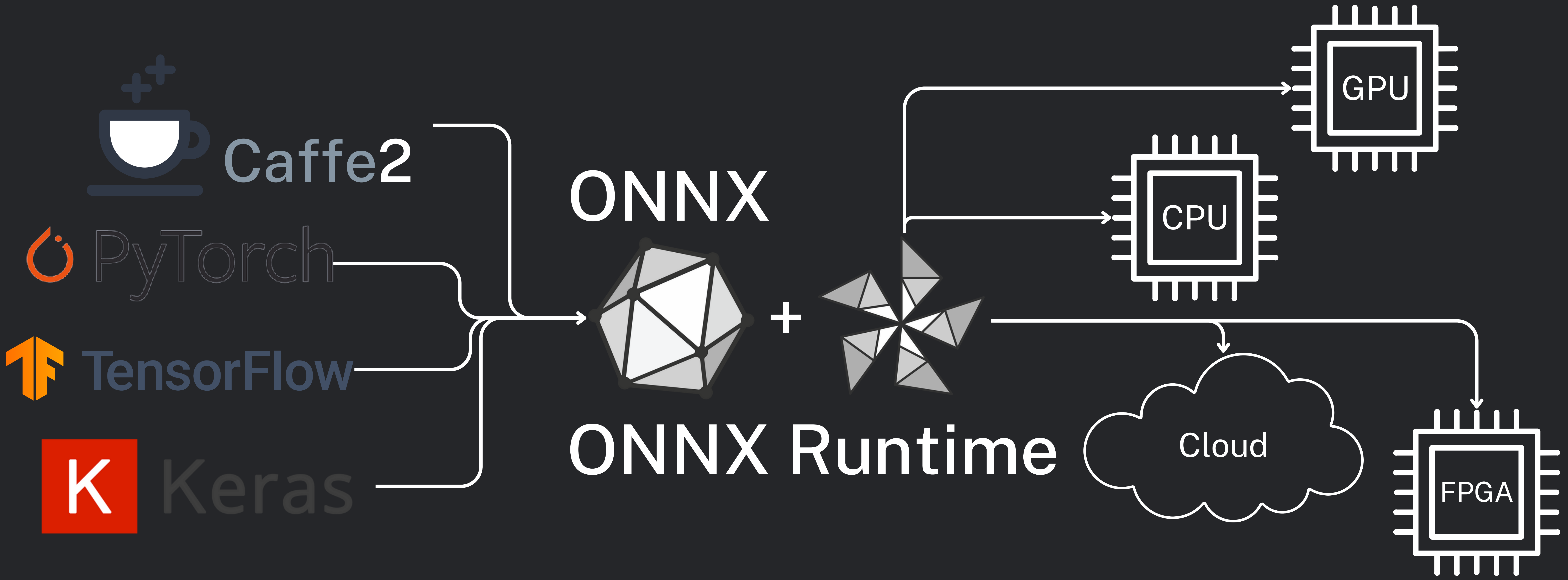
$y = x @ a + c$

```
def onnx_linear_regressor(x):
    y = onnx.Add(onnx.MatMul(x, a), c)
    return y
```



- Common language for any ML framework
- Operators (Add, Mul, Max, Min, etc.)
- ONNX data types (int8, float16, bool, etc.)

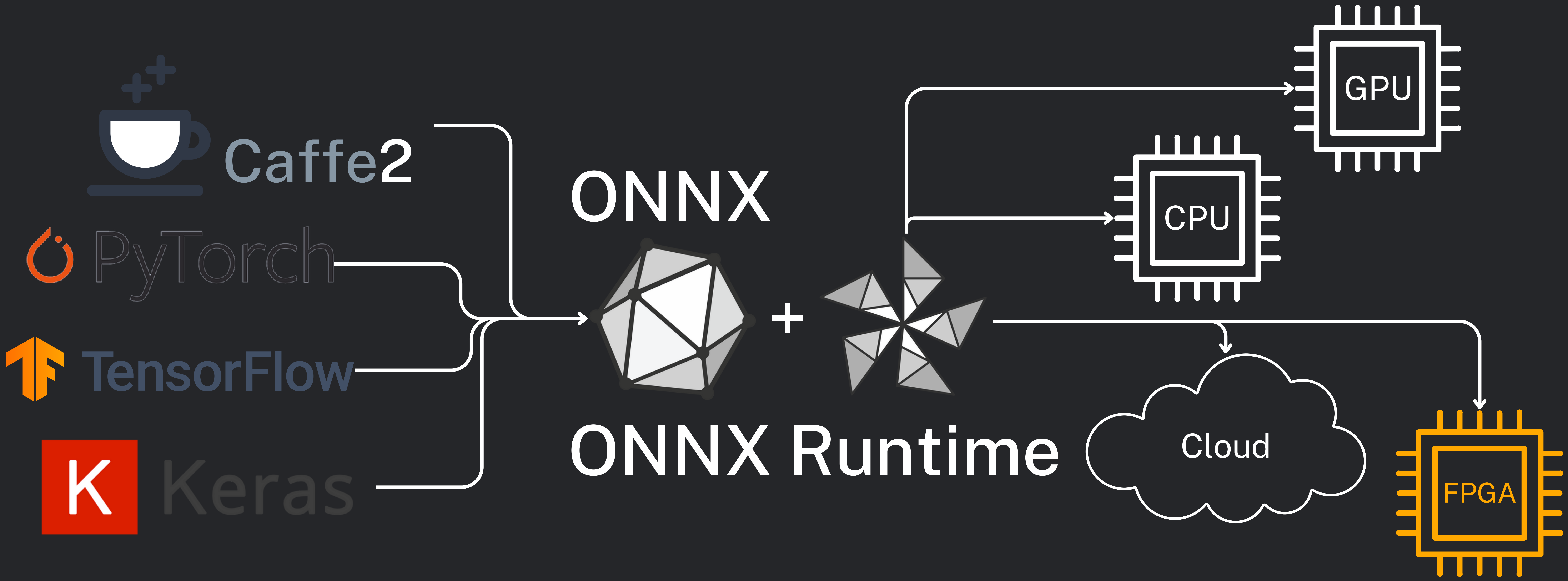
Deployment with ONNX



training

deployment

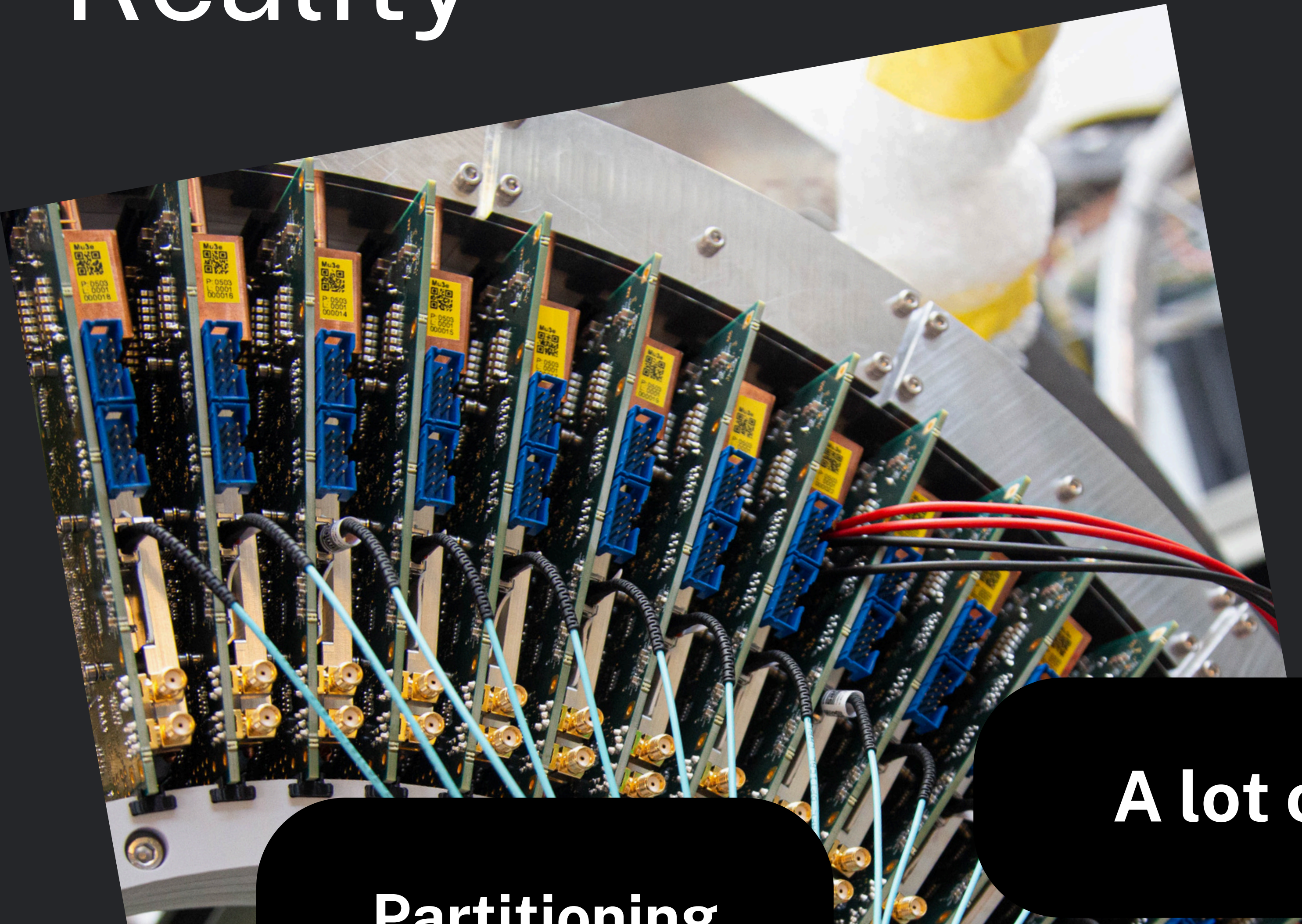
Deployment with ONNX



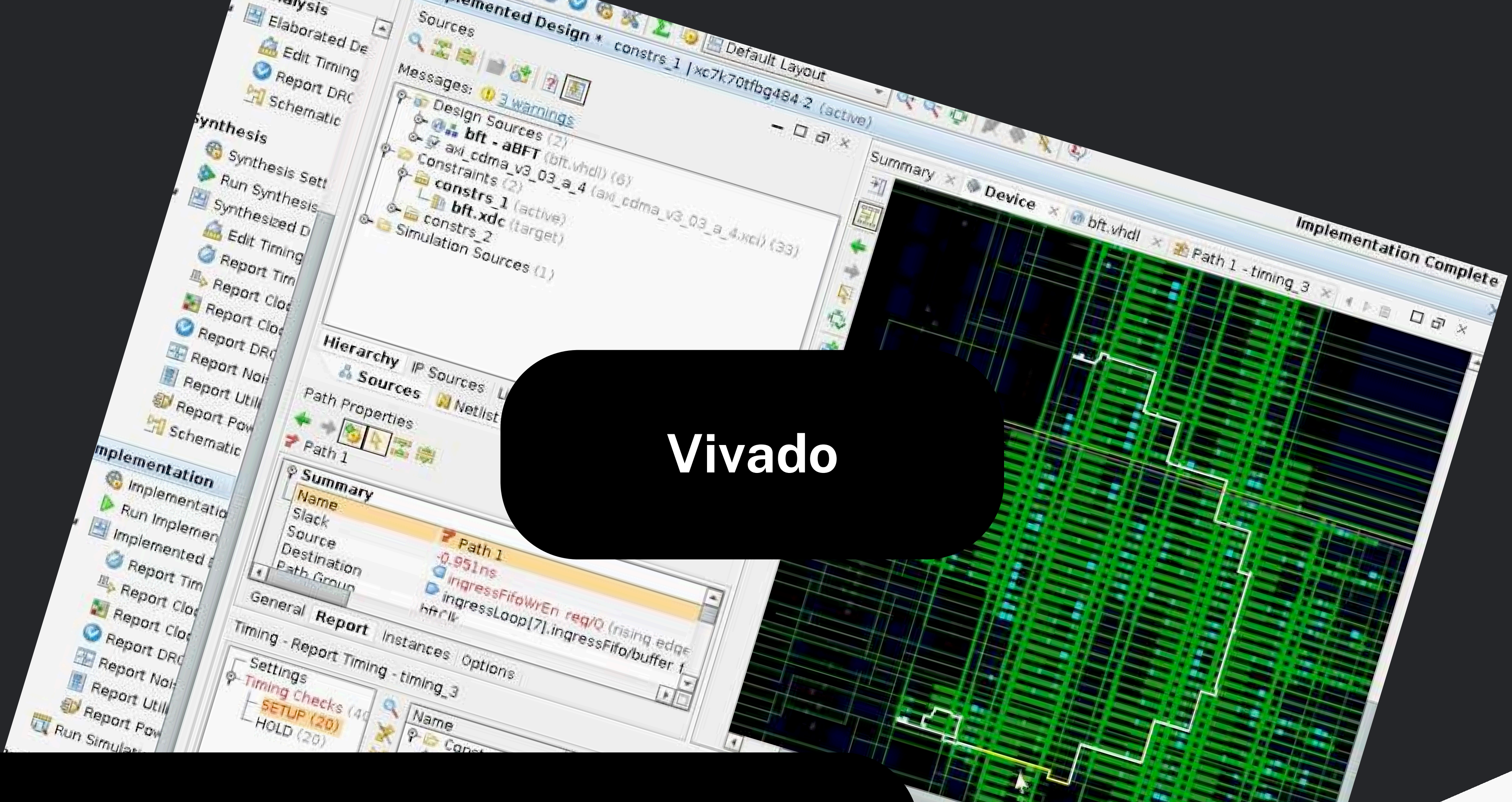
training

deployment

Reality

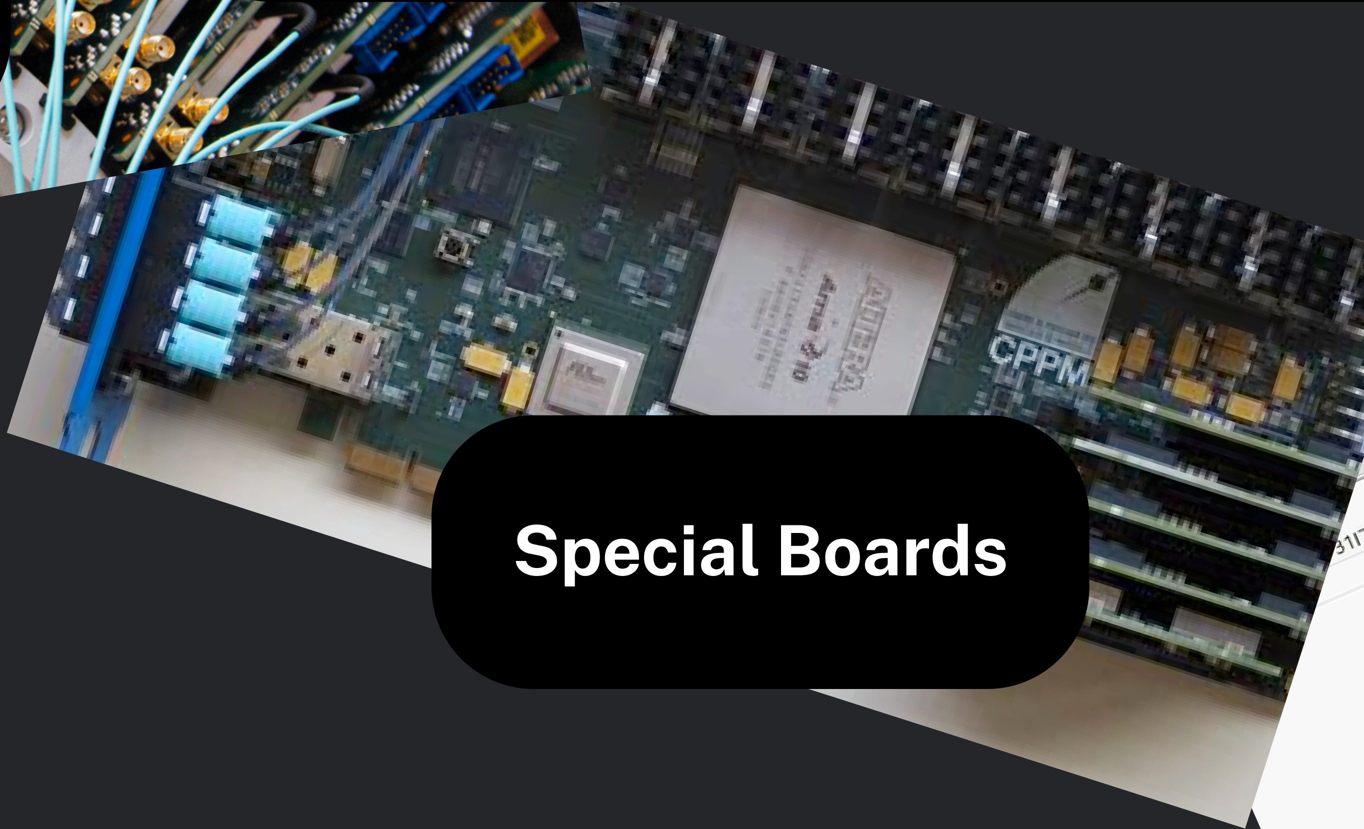


Partitioning

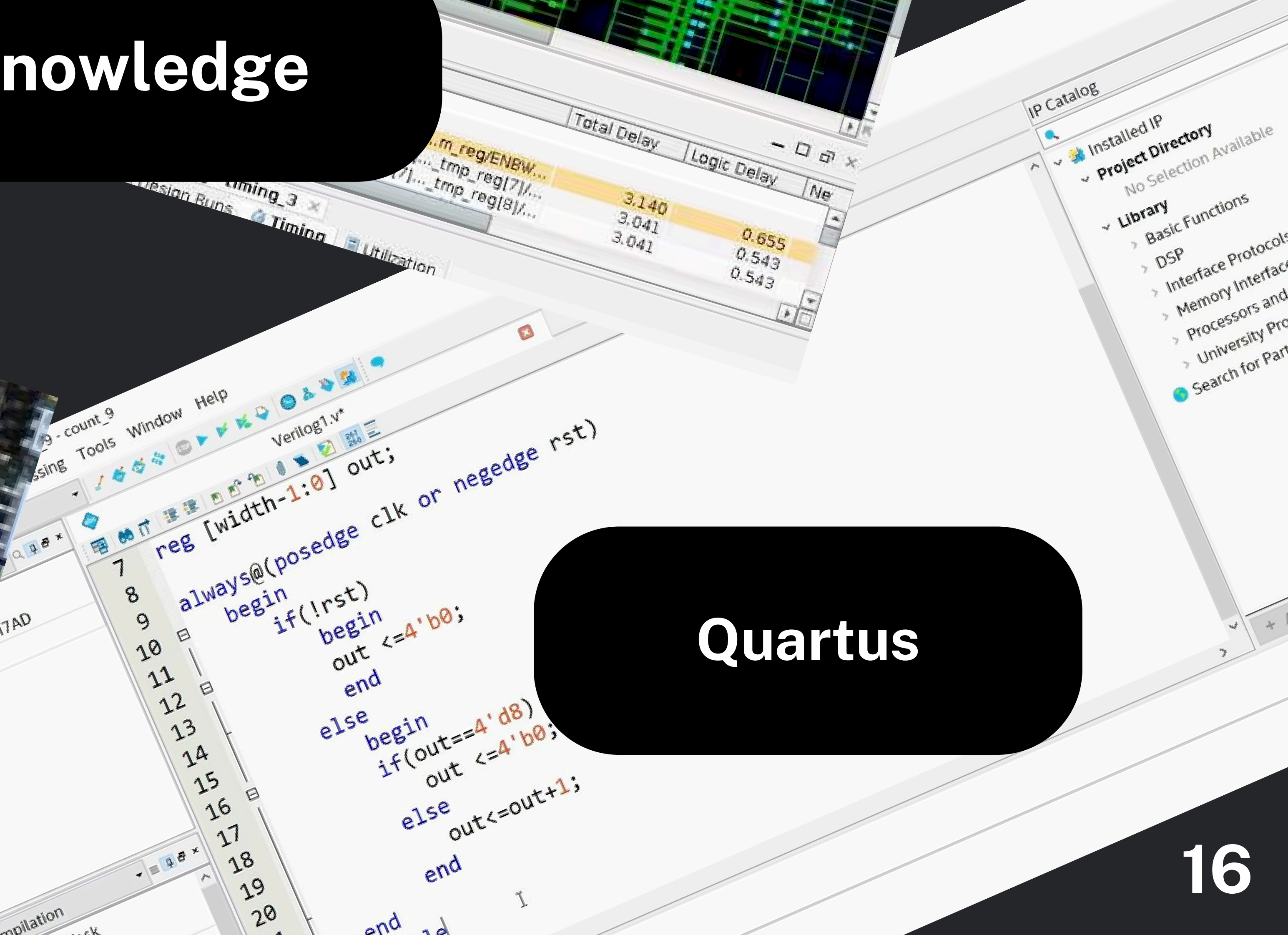


Vivado

A lot of tooling, software, knowledge



Special Boards

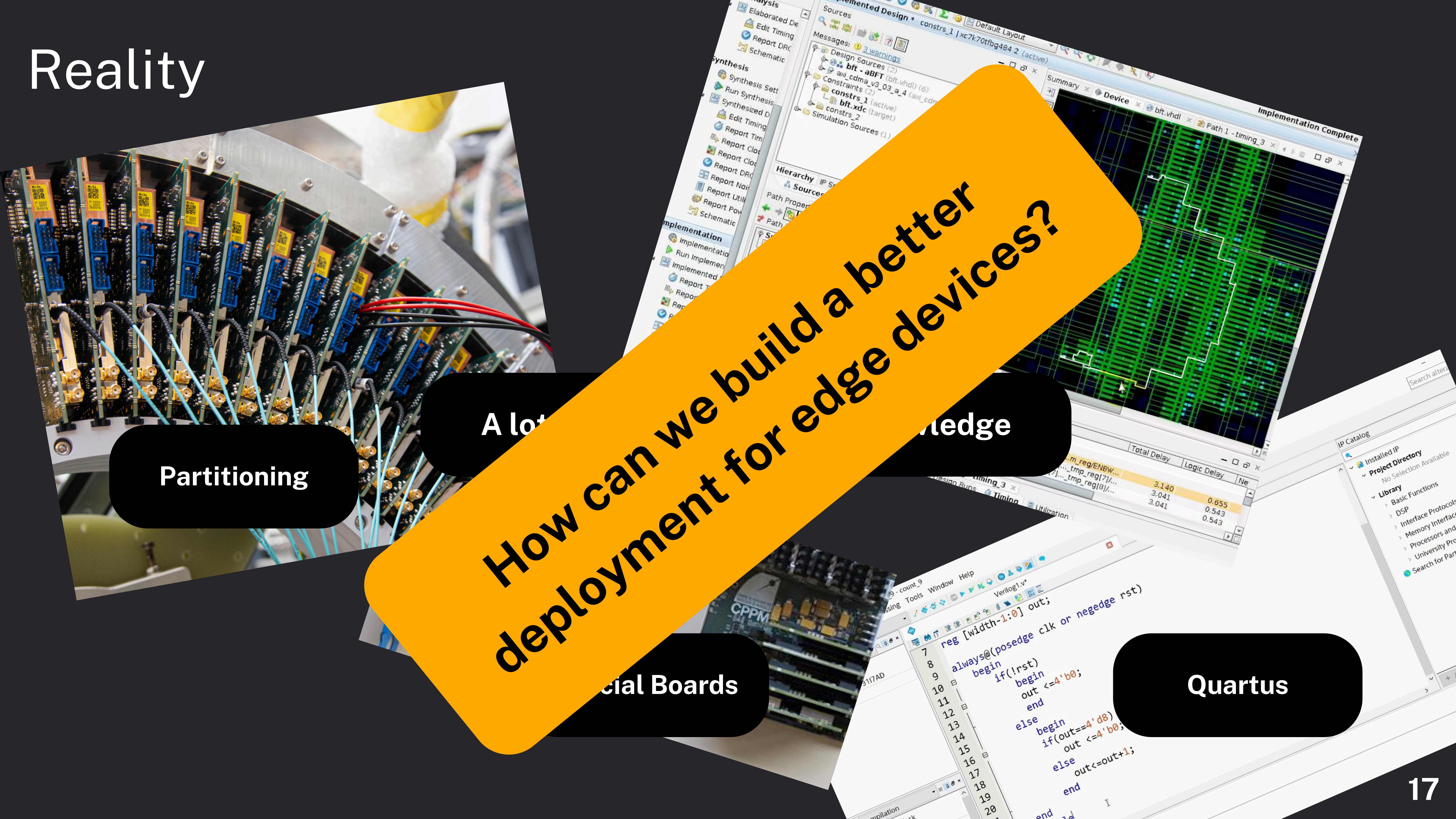


```verilog
reg [width-1:0] out;

always@(posedge clk or negedge rst)
  begin
    if(!rst)
      begin
        out <=4'b0;
      end
    else
      begin
        if(out==4'd8)
          out <=4'b0;
        else
          out<=out+1;
      end
end
```

Quartus

Reality



Partitioning

A lot … ledge

…cial Boards

Quartus

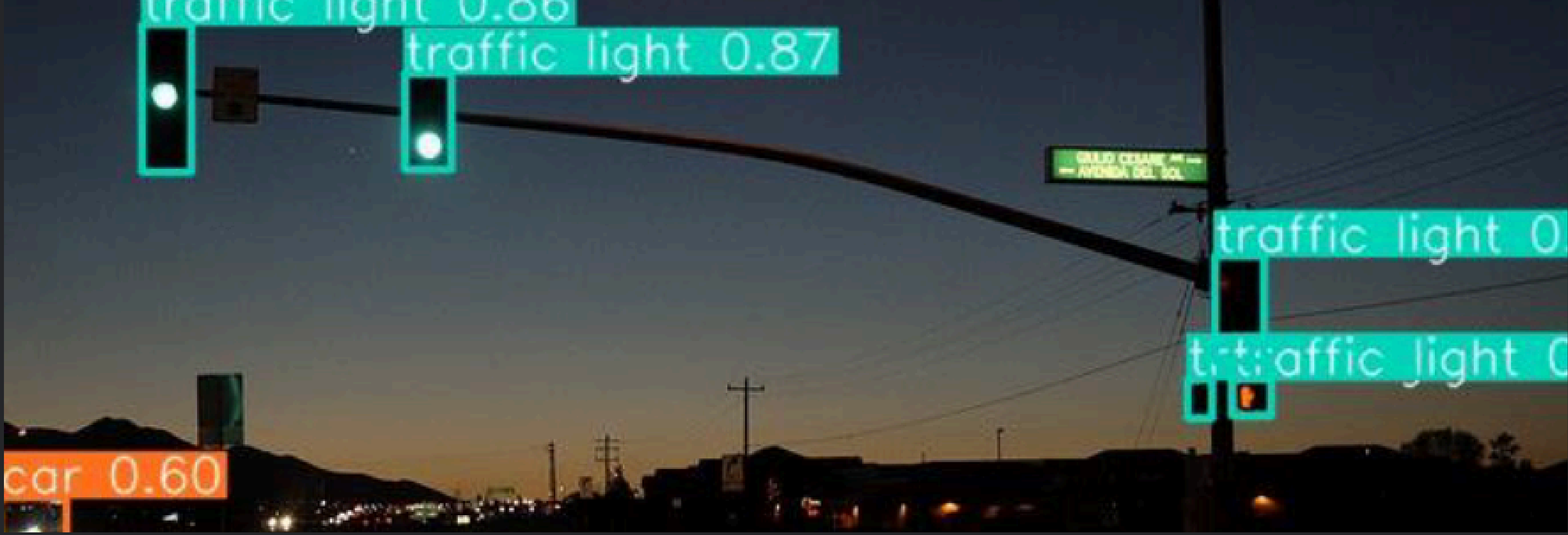
How can we build a better deployment for edge devices?

```
reg [width-1:0] out;

always@(posedge clk or negedge rst)
    begin
        if(!rst)
            begin
                out <=4'b0;
            end
        else
            begin
                if(out==4'd8)
                    out <=4'b0;
                else
                    out<=out+1;
            end
    end
```

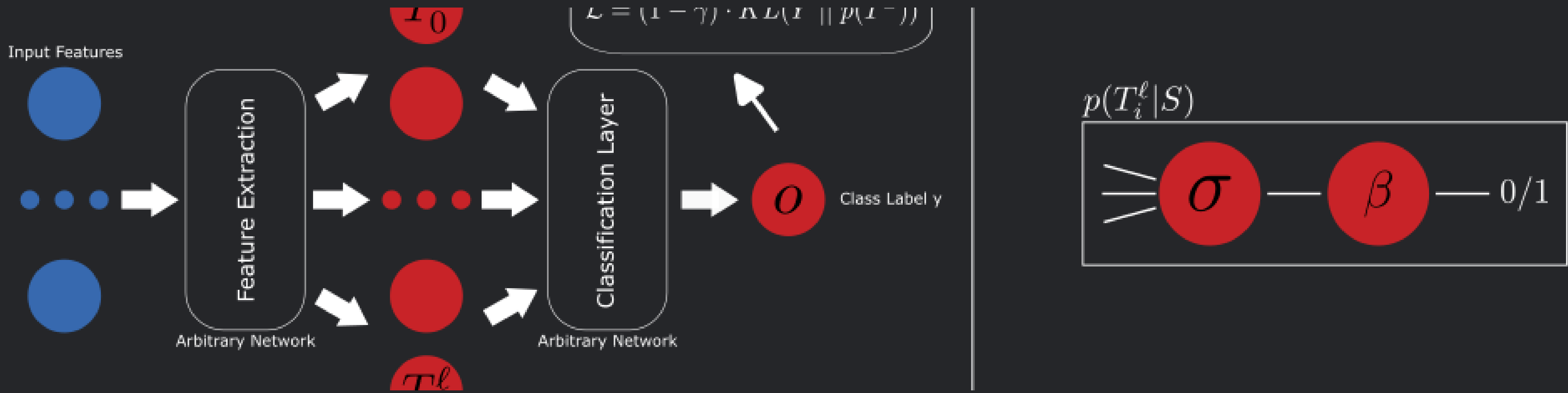
QNNs

- Need highly optimized deployment
- High throughput
- Low latency
- Low power constraints
- Nice mathematical properties
- Interpretability
- On edge devices
- …



YOLOv10 arXiv:2405.14458

AXOL1TL arXiv:2312.10009

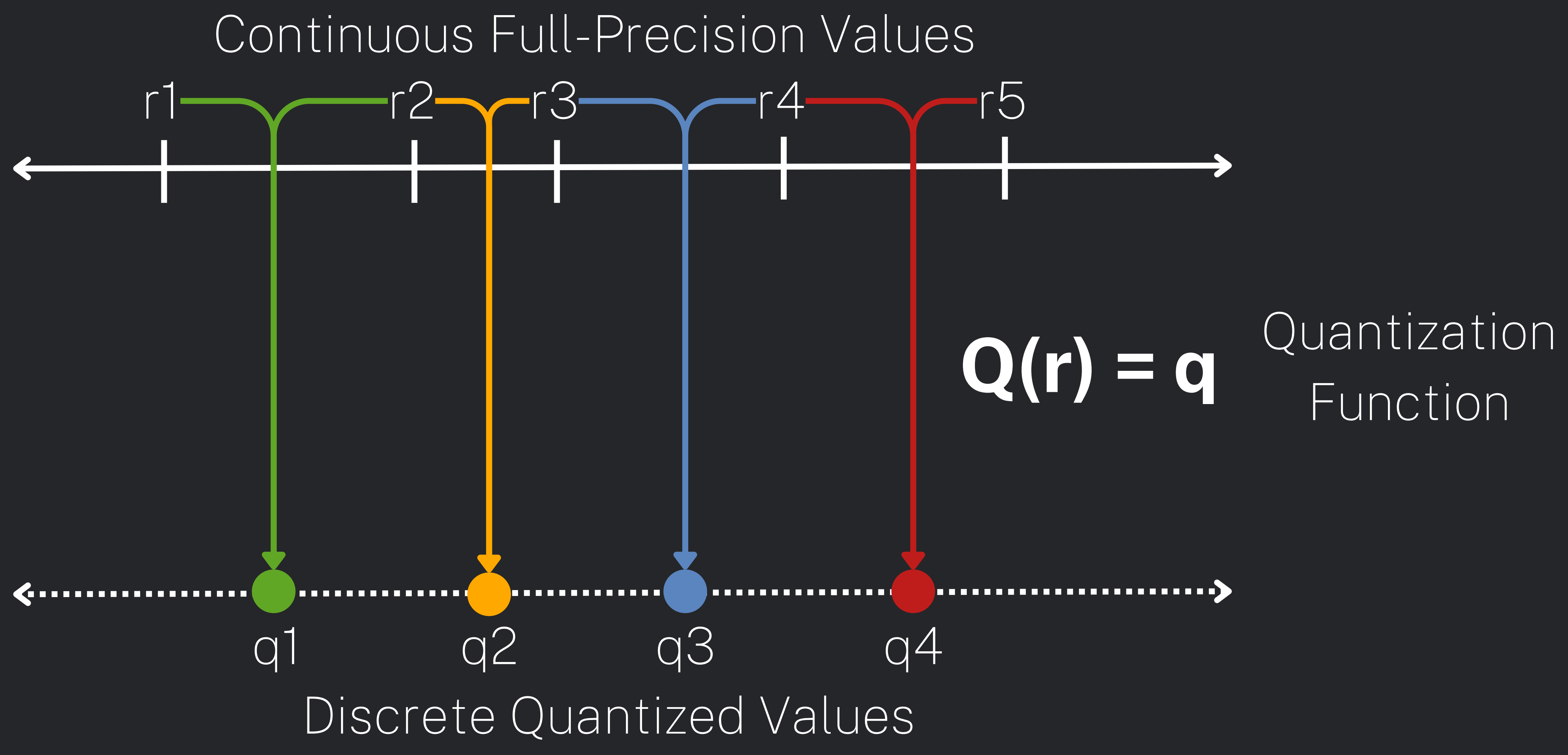


BinaryMI arXiv:2208.02656

Quantization in NNs

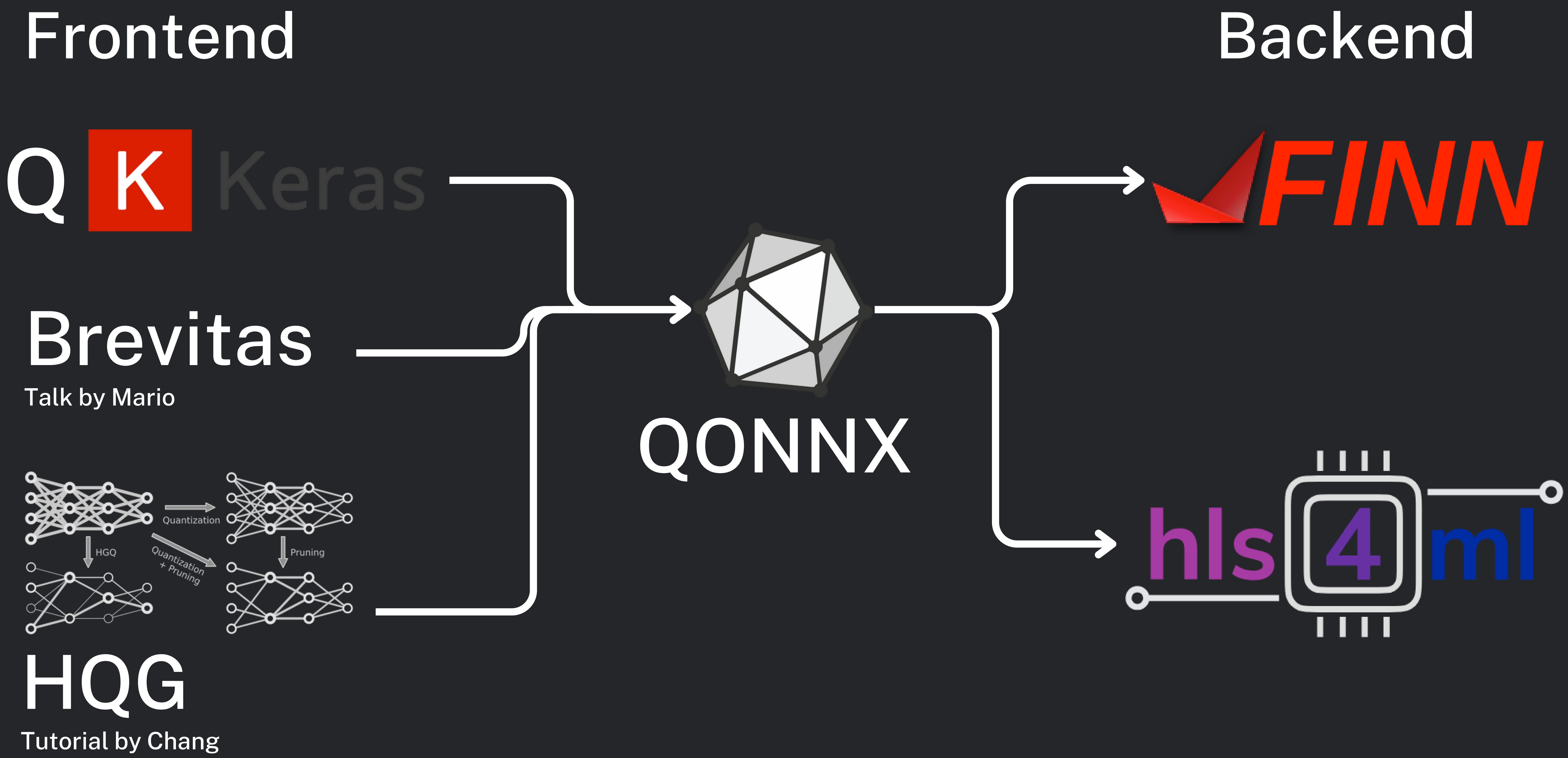
y = x @ a + c

- Each part of the NN can be quantized
- A lot of techniques / strategy out there
  - x-bit instead of floating point
  - Quantization-Aware Training (QAT)
  - Post-Training Quantization (PTQ)
  - Mixed-Precision Quantization
  - …
- Need for a common standard



Note: “everyone” is “ignoring” how to do “proper” backpropagation

Why QONNX?



Frontend

Backend

QKeras

Brevitas

Talk by Mario

HQG

Tutorial by Chang

QONNX

FINN

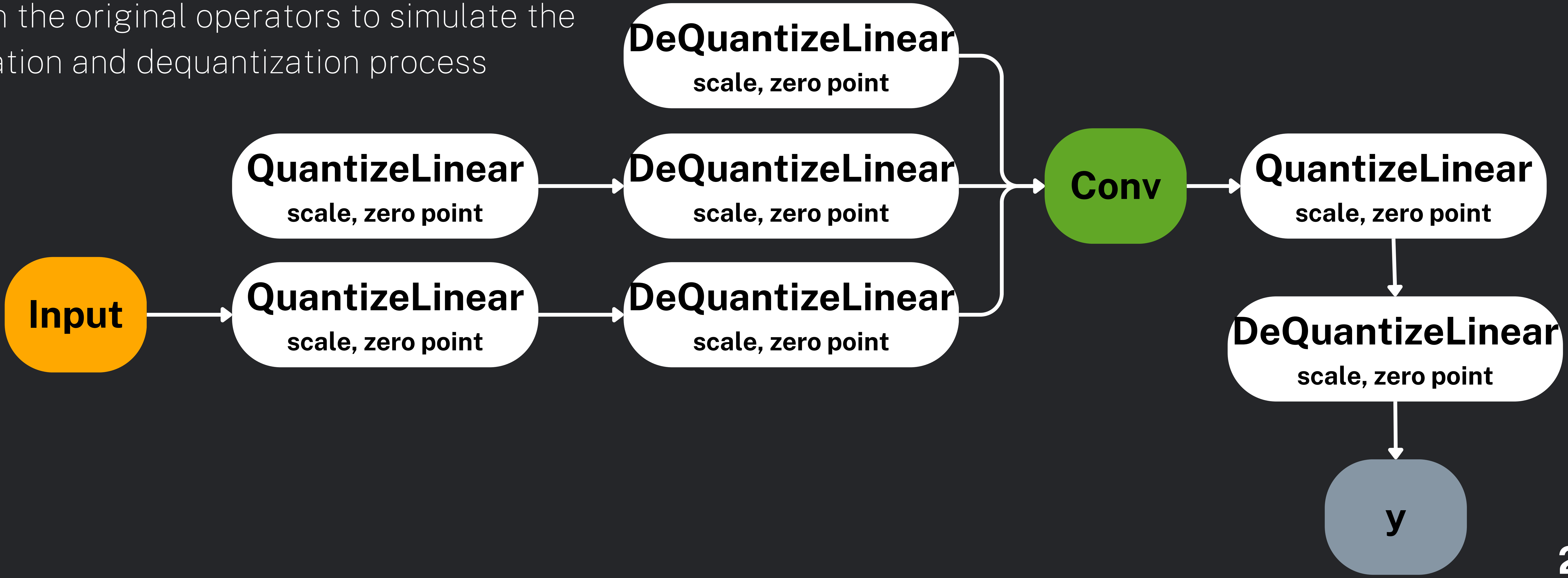
hls4ml

Current quantization formats in ONNX

Tensor-oriented

(QDQ - Quantize and DeQuantize)

- Inserts DeQuantizeLinear(QuantizeLinear(tensor)) between the original operators to simulate the quantization and dequantization process

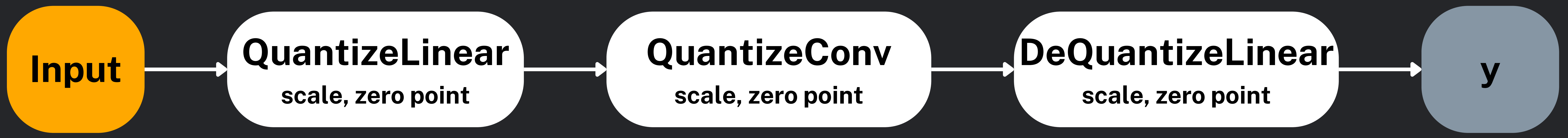


Current quantization formats in ONNX

Operator-oriented

(QOperator)

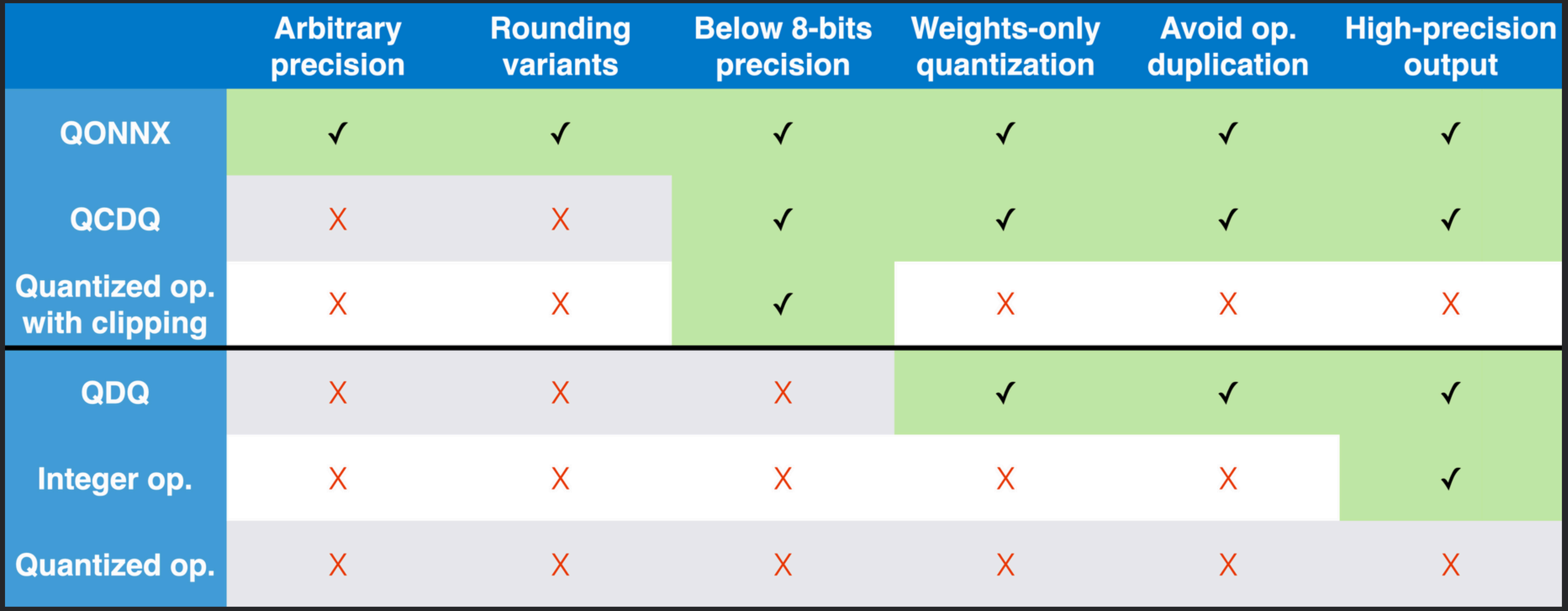
- All quantized operators own ONNX definitions
  - (like QLinearConv, MatMulInteger etc)



QONNX in a Nutshell

QONNX is a collection of specialized ONNX operators

- **Quant:** for 2-to-arbitrary-bit quantization, with scaling and zero-point
- **BipolarQuant:** for 1-bit (bipolar) quantization, with scaling and zero-point
- **Trunc:** for truncating to a specified number of bits, with scaling and zero-point
- **FixedPoint** (Coming soon).

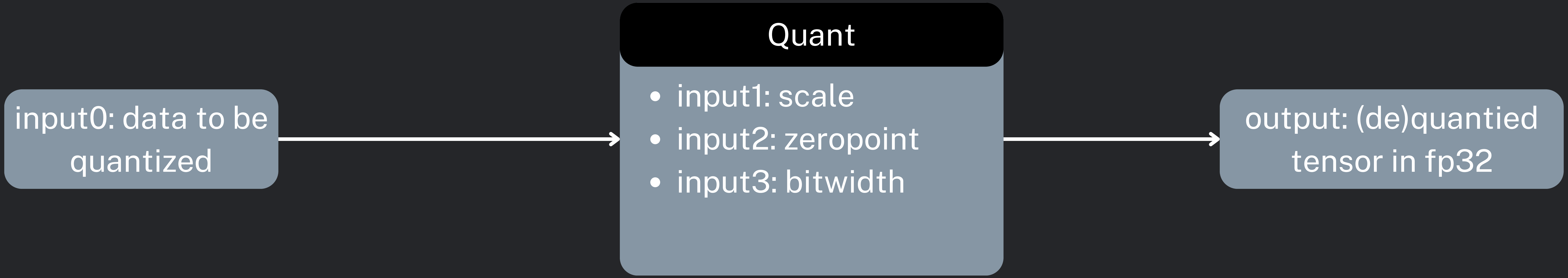


	Arbitrary precision	Rounding variants	Below 8-bits precision	Weights-only quantization	Avoid op. duplication	High-precision output
QONNX	✓	✓	✓	✓	✓	✓
QCDQ	X	X	✓	✓	✓	✓
Quantized op. with clipping	X	X	✓	X	X	X
QDQ	X	X	X	✓	✓	✓
Integer op.	X	X	X	X	X	✓
Quantized op.	X	X	X	X	X	X

Tools

- **Analysis Passes:** nodes statistics, inference cost **(Tutorial)**
- **Transformation Passes:** convert node types (float->int), pruning, etc.

How does the Quant op work?



input0: data to be quantized

Quant

- input1: scale
- input2: zeropoint
- input3: bitwidth

output: (de)quantied tensor in fp32

$$y = \mathrm{quantize}(x) = \mathrm{clamp}(\mathrm{round}(\frac{x}{s} + z), y_{min}, y_{max})$$

$$y_{max} = \begin{cases} 2^{n_b-1} - 1, & \text{if signed} \\ 2^{n_b} - 1 & \text{otherwise} \end{cases} \quad y_{min} = \begin{cases} -2^{n_b-1}, & \text{if signed} \\ 0 & \text{otherwise} \end{cases}$$

Inference cost with QONNX

Bit Operations (BOPs) in QONNX

The BOPs metric is used to assess the computational complexity and performance on NNs deployed to FPGAs and ASICs

$$\mathrm{BOPs} = mn[(1 - f_p)b_a b_w + b_a + b_w + \log_2(n)]$$

- **n/m:** # number of inputs/outputs
- **bw / ba:** bit width of the weights / activations
- **fp:** is the fraction of pruned layer weights
  - fp does account for pruned multiplication operations

more about BOPs arXiv:1804.10969 & arXiv: 2102.11289

```
> qonnx-inference-cost CNV_2W2A.onnx
> Inference cost for CNV_2W2A.onnx
{
# discount Multiply-Accumulate Operations (MAC) counts by layer sparsity (disregard zero-valued MACs and params)
 "discount_sparsity": true,
# mem_o_X: number of outputs with datatype X
 "mem_o_INT32": 142602.0,
# mem_o_X: number weights with datatype X
 "mem_w_INT2": 908033.0,
# op_mac_X_Y: # of MAC operations, datatype X by Y
# scaled integer datatypes have a tensor
# number of scaled int8 x int2 MACs
 "op_mac_SCALEDINT<8>_INT2": 1345500.0,
# number of int2 x int2 MACs
 "op_mac_INT2_INT2": 35615771.0,
# total number of MACs normalized to bit-ops (BOPS)
 "total_bops": 163991084.0,
}
```

Open-Source QONNX Repositories

Checked 23.09.2024

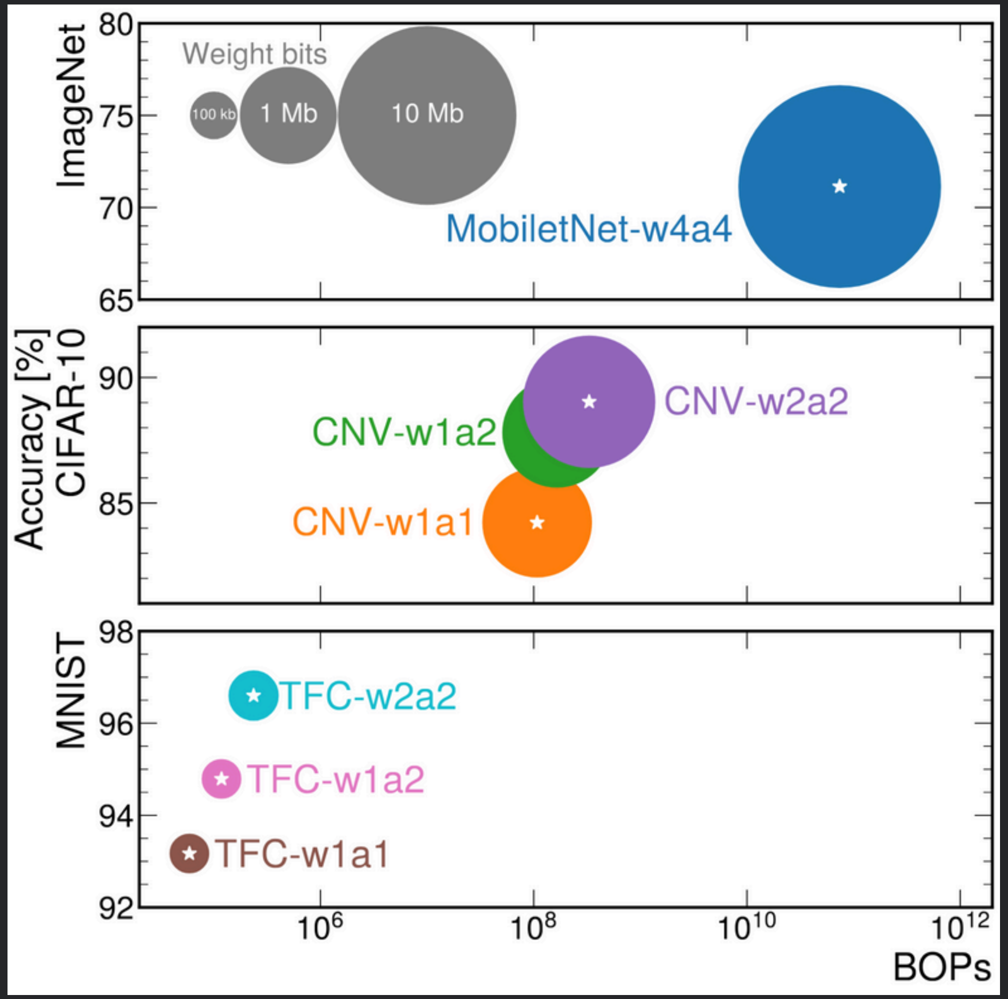
Python Toolkit

github.com/fastmachinelearning/qonnx

- QONNX is a set of specialized ONNX operators
- Execution of custom QONNX nodes
- Getting inference analysis
- Doing model transformation
- Multplie pretrained models available
- …

Model Zoo

github.com/fastmachinelearning/QONNX_model_zoo



Next model trained on physics data?

QONNX Tutorial

https://github.com/makoeppel/Edge-ML-School0924-QONNX