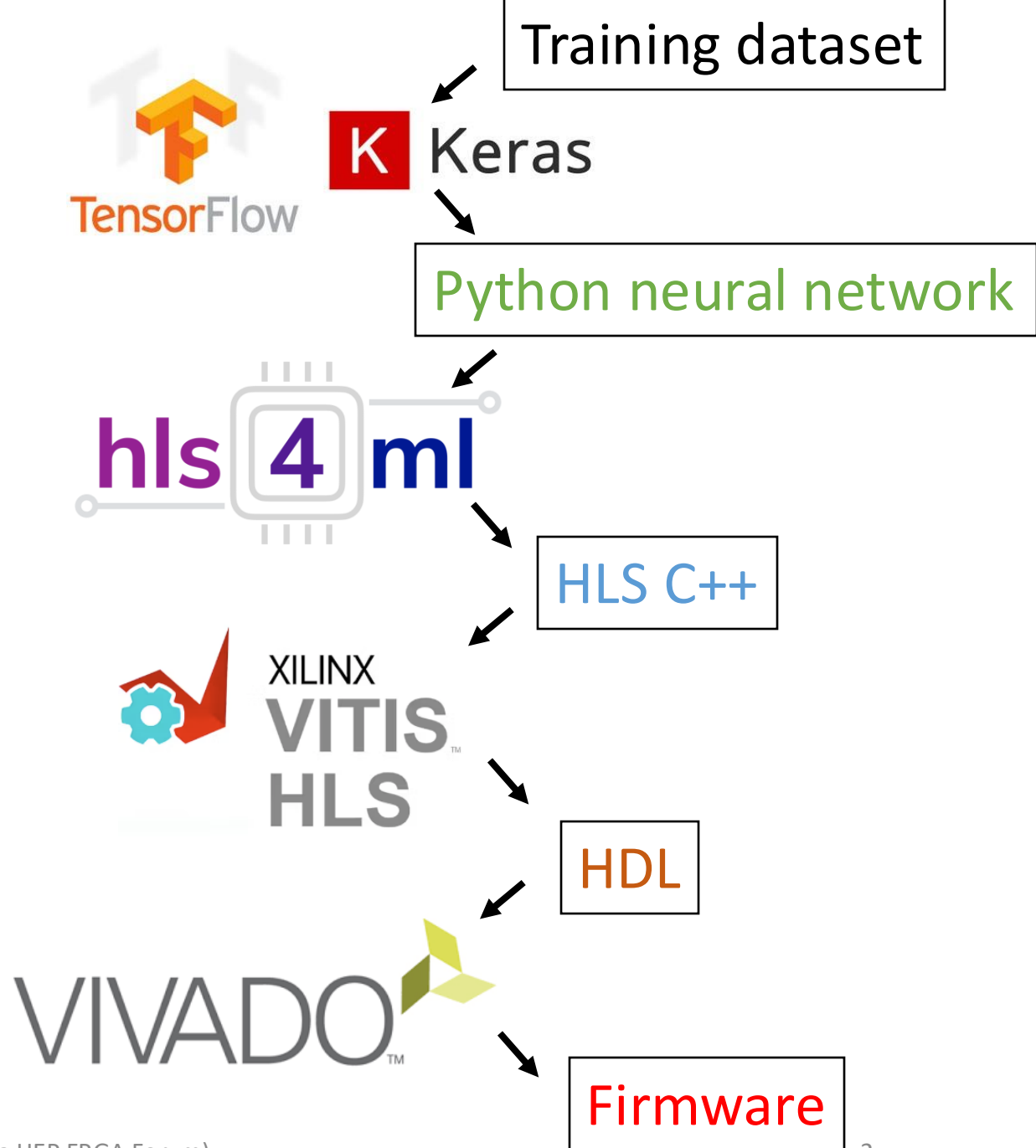



# Introduction to HLS4ML

# HLS4ML

- HLS4ML changes **python** neural networks to **HLS C++**.
- HLS changes **C++** to **HDL**.
- Vivado changes **HDL** to **firmware**.
- Will go through this pipeline.



# Example case used in HLS4ML tutorial [[link](#)]

- Train a network to find what generated a AK8 jet
  - AK8 jet could be from  $W^+W^-$ ,  $ZZ$ ,  $t\bar{t}$ ,  $q\bar{q}$ ,  $gg$ , ...
- Dataset: DOI: 10.5281/zenodo.3602254
  - Generated with Madgraph5, Pythia8
  - 16 features of AK8 jet 
  - 5 labels ( $W^+W^-$ ,  $ZZ$ ,  $t\bar{t}$ ,  $q\bar{q}$ ,  $gg$ ) for AK8 jet

Observables
$m_{\text{mMDT}}$
$N_2^{\beta=1,2}$
$M_2^{\beta=1,2}$
$C_1^{\beta=0,1,2}$
$C_2^{\beta=1,2}$
$D_2^{\beta=1,2}$
$D_2^{(\alpha,\beta)=(1,1),(1,2)}$
$\sum z \log z$
Multiplicity

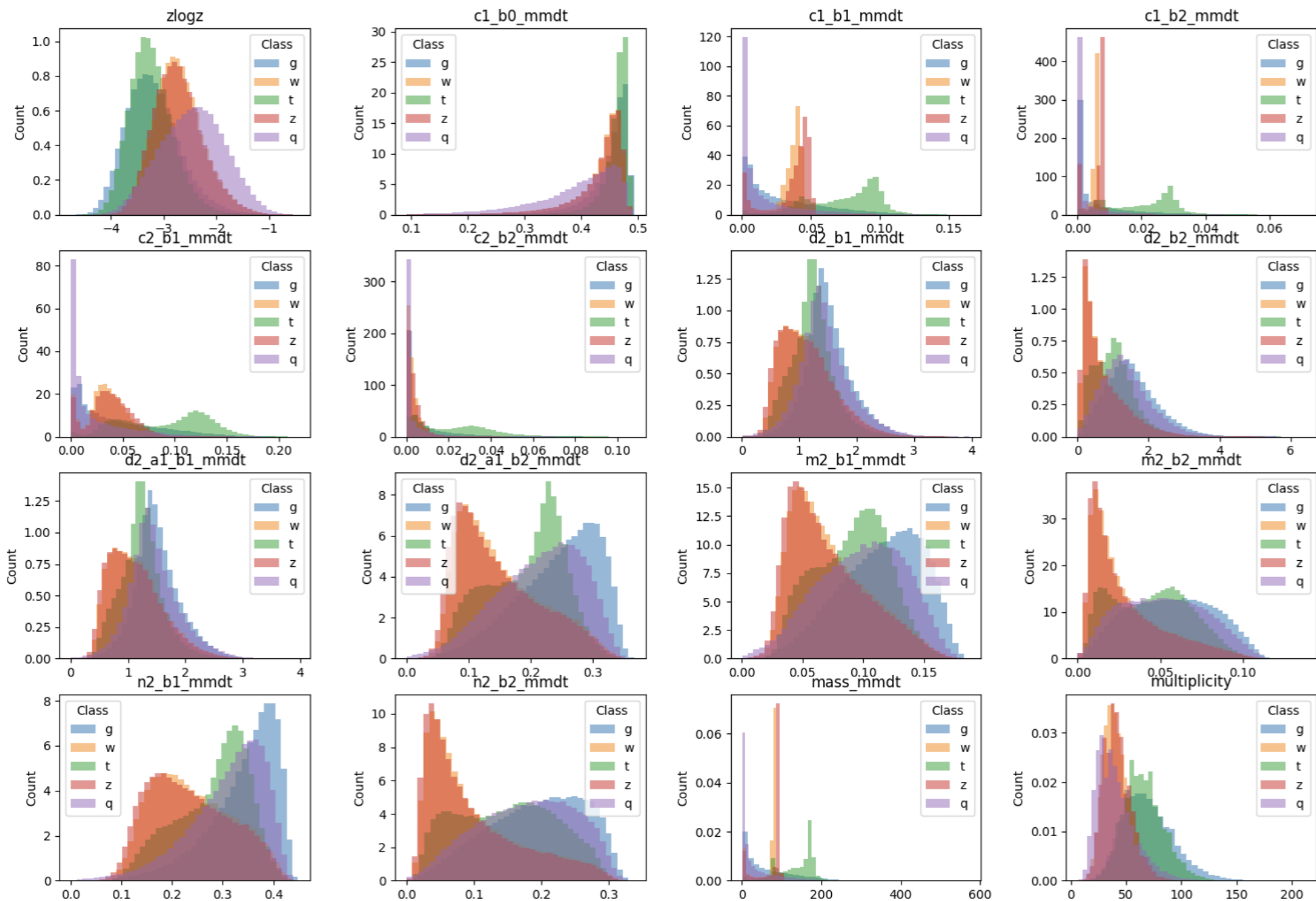
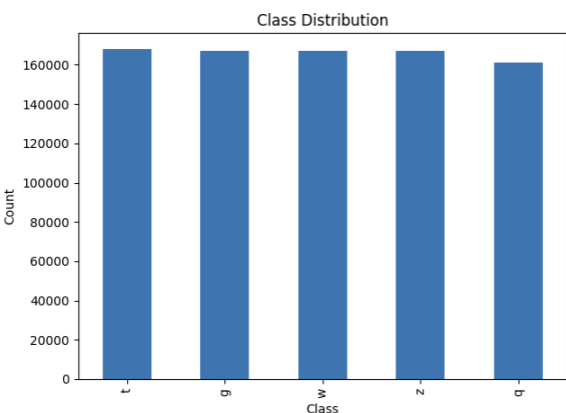
# Dataset

- 16 features

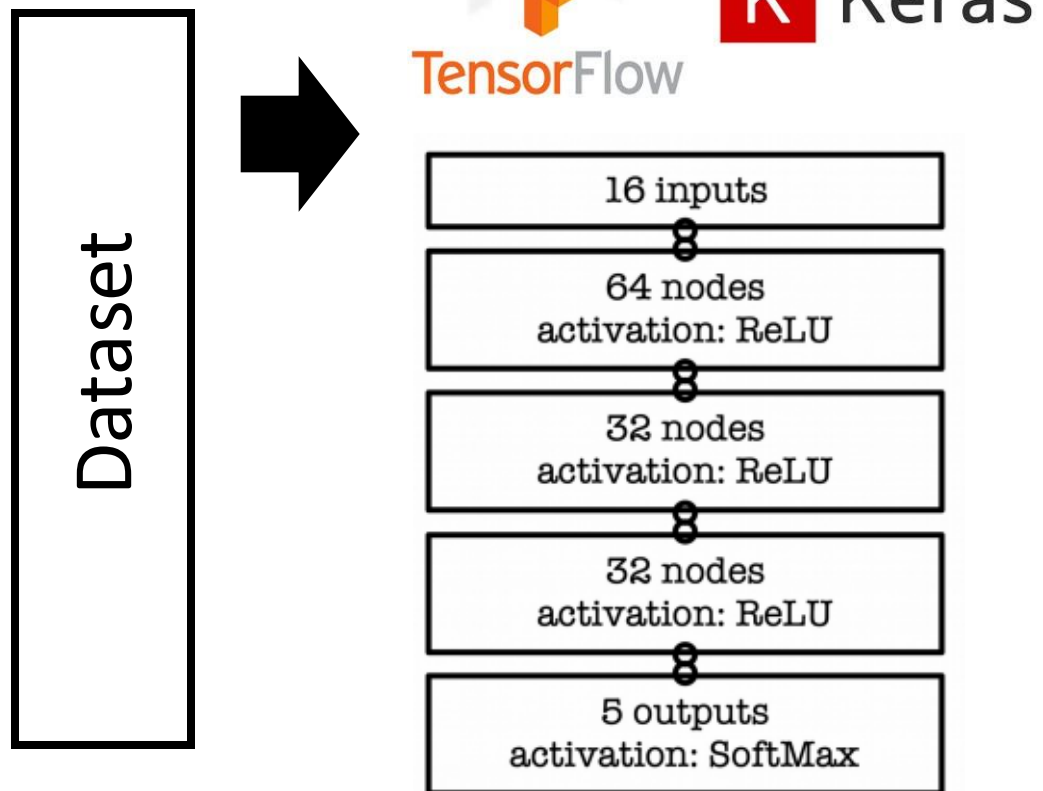
of AK8 jet

- Each class

160k events

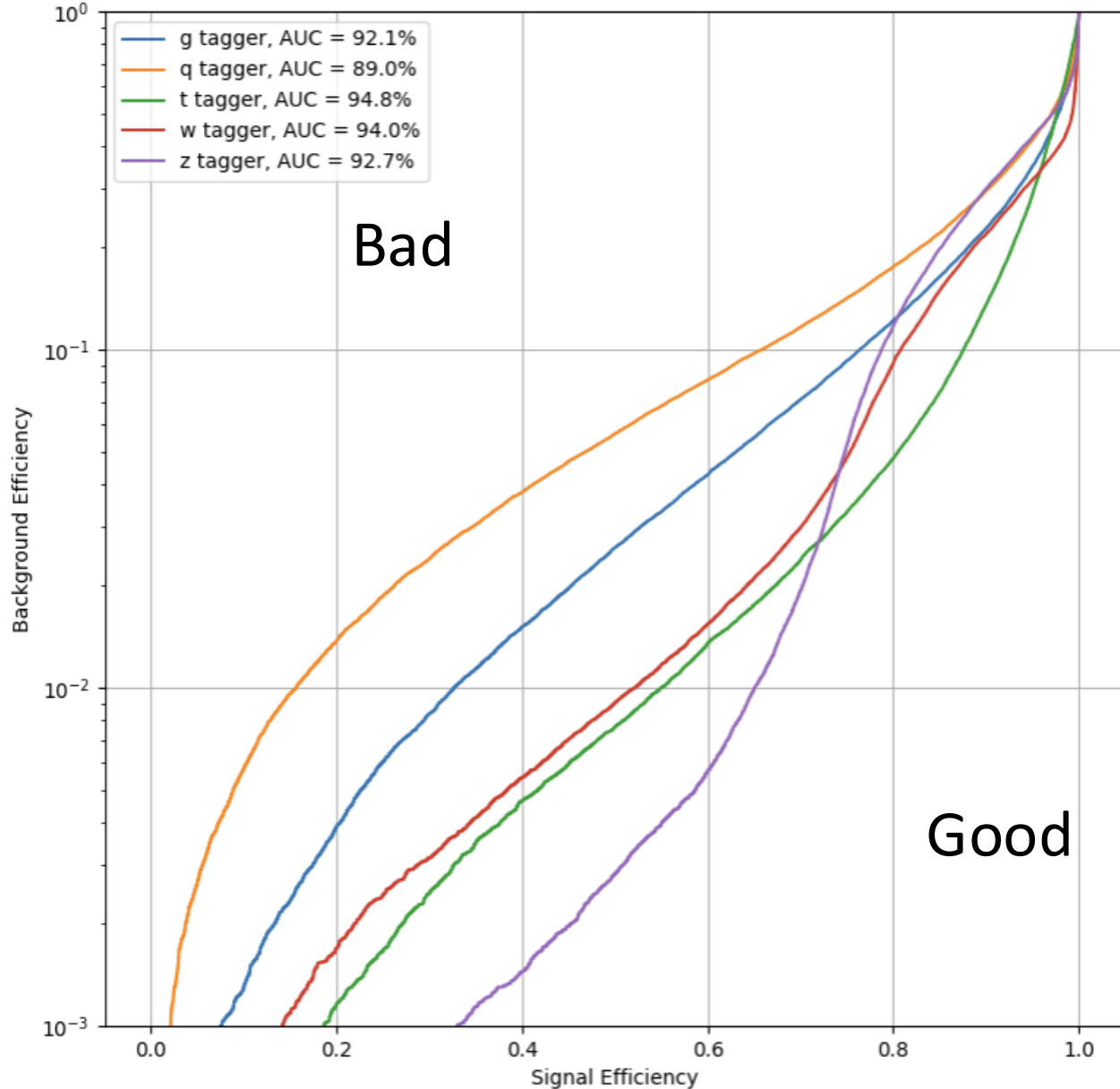


# Making model



- Can train with Keras

- Model saved into HDF5 format.



# Convert python model with HLS4ML

- HLS4ML makes C++ code for the network.

➤ Integerizes the weights and inputs.

```
// hls-fpga-machine-learning insert layer-precision
typedef ap_fixed<16,6> input_t;
typedef ap_fixed<16,6> model_default_t;
typedef ap_fixed<16,6> layer2_t;
typedef ap_uint<1> layer2_index;
typedef ap_fixed<16,6> layer4_t;
typedef ap_fixed<18,8> relu1_table_t;
typedef ap_fixed<16,6> layer5_t;
typedef ap_uint<1> layer5_index;
typedef ap_fixed<16,6> layer7_t;
typedef ap_fixed<18,8> relu2_table_t;
```

Weights  
precision

```
layer2_t layer2_out[N_LAYER_2];
#pragma HLS ARRAY_PARTITION variable=layer2_out complete dim=0
nnet::dense<input_t, layer2_t, config2>(fc1_input, layer2_out, w2, b2); // fc1

layer4_t layer4_out[N_LAYER_2];
#pragma HLS ARRAY_PARTITION variable=layer4_out complete dim=0
nnet::relu<layer2_t, layer4_t, relu_config4>(layer2_out, layer4_out); // relu1

layer5_t layer5_out[N_LAYER_5];
#pragma HLS ARRAY_PARTITION variable=layer5_out complete dim=0
nnet::dense<layer4_t, layer5_t, config5>(layer4_out, layer5_out, w5, b5); // fc2

layer7_t layer7_out[N_LAYER_5];
#pragma HLS ARRAY_PARTITION variable=layer7_out complete dim=0
nnet::relu<layer5_t, layer7_t, relu_config7>(layer5_out, layer7_out); // relu2

layer8_t layer8_out[N_LAYER_8];
#pragma HLS ARRAY_PARTITION variable=layer8_out complete dim=0
nnet::dense<layer7_t, layer8_t, config8>(layer7_out, layer8_out, w8, b8); // fc3

layer10_t layer10_out[N_LAYER_8];
#pragma HLS ARRAY_PARTITION variable=layer10_out complete dim=0
nnet::relu<layer8_t, layer10_t, relu_config10>(layer8_out, layer10_out); // relu3

layer11_t layer11_out[N_LAYER_11];
#pragma HLS ARRAY_PARTITION variable=layer11_out complete dim=0
nnet::dense<layer10_t, layer11_t, config11>(layer10_out, layer11_out, w11, b11); // output

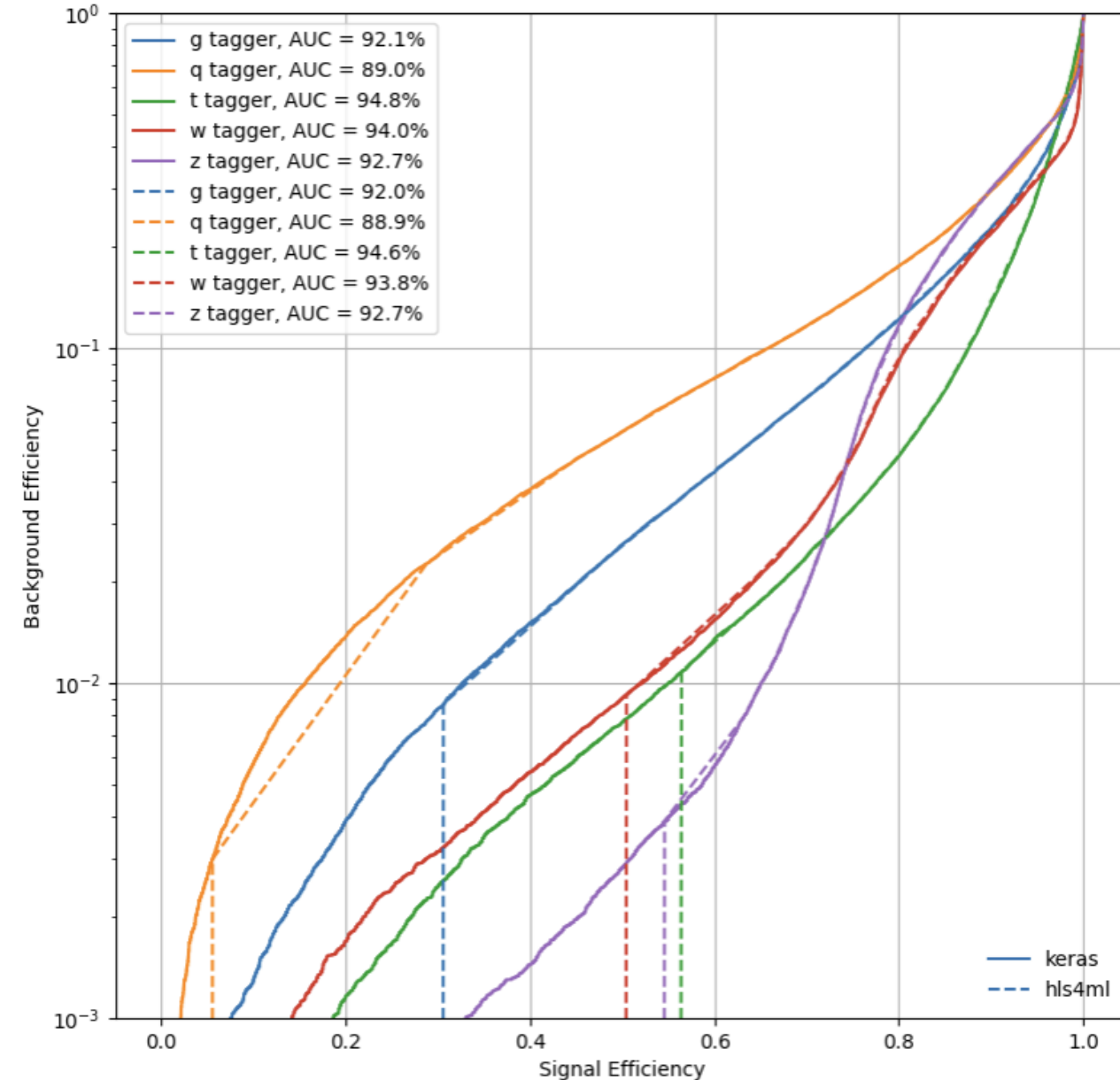
nnet::softmax<layer11_t, result_t, softmax_config13>(layer11_out, layer13_out); // softmax
```

# HLS4ML tools to checks network

- How good is the network?

- HLS4ML can check how good the integerization is (Dotted lines)

- Compare with float point calcuation.



# Convert HLS C++ to HDL

- Using Vitis HLS, C++ can be converted to HDL.
- Too many DSPs for small Pynq-Z2 FPGA...

```
entity myproject is
port (
    ap_clk : IN STD_LOGIC;
    ap_rst : IN STD_LOGIC;
    ap_start : IN STD_LOGIC;
    ap_done : OUT STD_LOGIC;
    ap_idle : OUT STD_LOGIC;
    ap_ready : OUT STD_LOGIC;
    fc1_input_ap_vld : IN STD_LOGIC;
    fc1_input : IN STD_LOGIC_VECTOR (255 downto 0);
    layer13_out_0 : OUT STD_LOGIC_VECTOR (15 downto 0);
    layer13_out_0_ap_vld : OUT STD_LOGIC;
    layer13_out_1 : OUT STD_LOGIC_VECTOR (15 downto 0);
    layer13_out_1_ap_vld : OUT STD_LOGIC;
    layer13_out_2 : OUT STD_LOGIC_VECTOR (15 downto 0);
    layer13_out_2_ap_vld : OUT STD_LOGIC;
    layer13_out_3 : OUT STD_LOGIC_VECTOR (15 downto 0);
    layer13_out_3_ap_vld : OUT STD_LOGIC;
    layer13_out_4 : OUT STD_LOGIC_VECTOR (15 downto 0);
    layer13_out_4_ap_vld : OUT STD_LOGIC );
end;
```

Performance & Resource Estimates

Modules

Loops

Hide empty columns

MODULES & LOOPS	LATENCY(CYCLES)	LATENCY(NS)	INTERVAL	PIPELINED	BRAM	DSP	FF	LUT	URAM
> <span>myproject</span> (8)	36	360.000	1	yes	4	3009	89040	128039	0

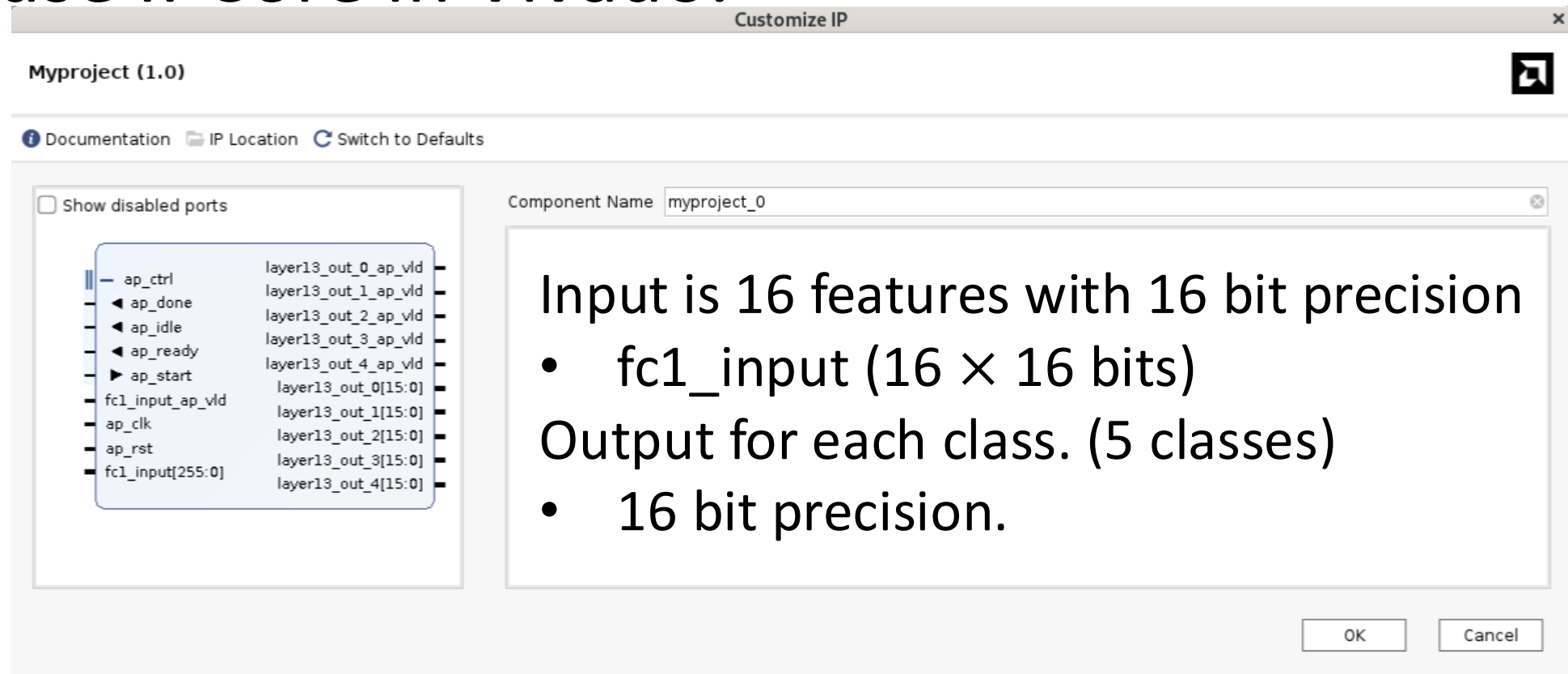


# Convert HLS C++ to HDL

- Can make it into a IPCore with Vitis HLS.

```
INFO: Created IP /home/hepdream/Work/FPGA_class/ML4HLS_helloworld/src/model_1/hls4ml_prj/hls/hls/hls/impl/ip/component.xml
INFO: Created IP archive /home/hepdream/Work/FPGA_class/ML4HLS_helloworld/src/model_1/hls4ml_prj/hls/hls/hls/impl/ip/xilinx_com_hls_myproject_1_0.zip
```

- Can use IPCore in Vivado.



# HLS4ML scripts

- HLS4ML has scripts that use Vitis HLS and Vivado.
- Firmware can be made only using the HLS4ML python script.
  - Sometimes it will be easier to do things with python.
  - Sometimes, it will be easier to do things with the Vitis HLS and Vivado GUI.

# Optimization

- The network could be optimized
- There are many types of optimization
  - Performance optimization (Ex: Better AUC)
  - Resource reduction optimization (Ex: Less DSPs)
  - Latency reduction optimization
  - Initiation Interval (II) reduction optimization
- But they often counteract each other

# Many tricks in optimization

- The integerization precision could be changed.
- The weights can be set to integers when training.
  - No difference between python model and integerized C++ model.
- Small weights could be set to 0 to reduce resources.
- Resources could be reused using “rolling” loops.

# Example of optimization

- Previously we saw that too many resources were used.

➤ Pynq-z2 resources: BRAM(140), DSP(220), FF(106k), LUT(54k)

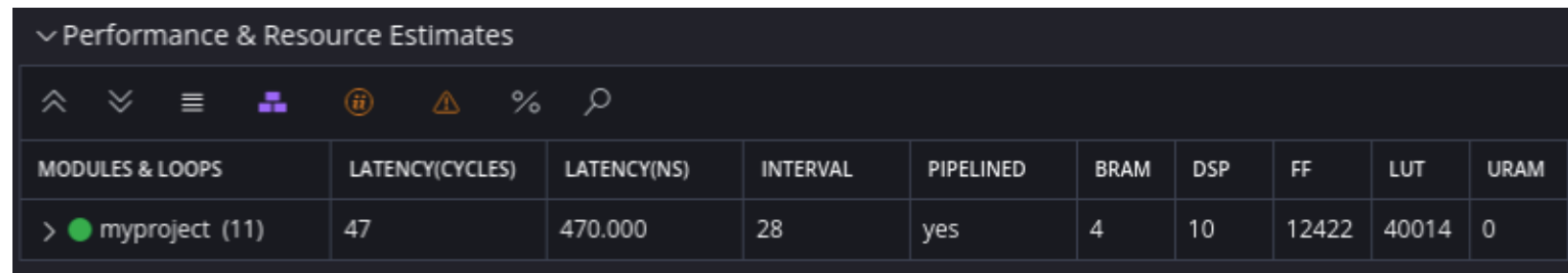


Performance & Resource Estimates

Modules: ☒ Modules    Loops: ☒ Loops    Hide empty columns: ☒

MODULES & LOOPS	LATENCY(CYCLES)	LATENCY(NS)	INTERVAL	PIPELINED	BRAM	DSP	FF	LUT	URAM
> ● myproject (8)	36	360.000	1	yes	4	3009	89040	128039	0

- We can try to “roll” loops to “reuse” resources and reduce bit size to fit it on the FPGA. (But performance is bad, and II is large.)



Performance & Resource Estimates

Modules: ☒ Modules    Loops: ☒ Loops    Hide empty columns: ☒

MODULES & LOOPS	LATENCY(CYCLES)	LATENCY(NS)	INTERVAL	PIPELINED	BRAM	DSP	FF	LUT	URAM
> ● myproject (11)	47	470.000	28	yes	4	10	12422	40014	0

- How much did you understand? [www.kahoot.it](http://www.kahoot.it)