Siemens BIB Rejection Uchicago update

April 8, 2026

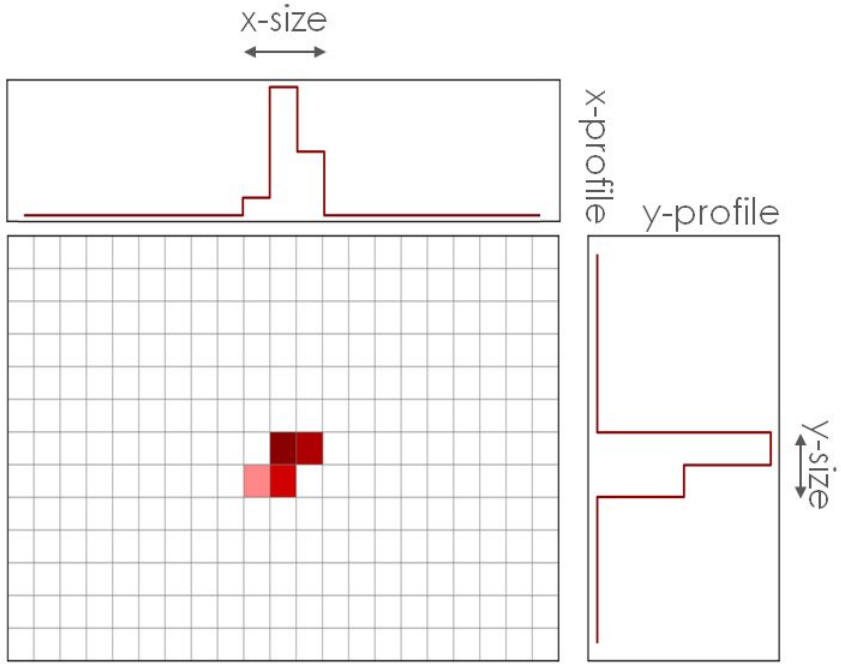
Daniel Abadjiev, Eliza Howard, Zoe Fulton, Ryan Michaud, Eric You

Current status

- We are considering 3 models which filter clusters of BIB from signal
- Eric and Ryan have preliminary hyperparameter scans of models 1 and 2 using Tensorflow+Qkeras
  - Realized we should redo them because:
- We newly added a layer to quantize the inputs
- We newly log-normalized dataset inputs for model 2 to keep all intermediary numbers in the network nodes small
- To get a hardware estimation, we synthesized our models for FPGA with hls4ml+Vitis, and preliminarily for ASIC with Catapult HLS
  - There seems to be a bug in how Catapult handles concatenation layers
  - Haven’t been able to synthesize model 3, perhaps it’s too big
- **I have been working on validating that the synthesized C++ code matches the performance of Qkeras**

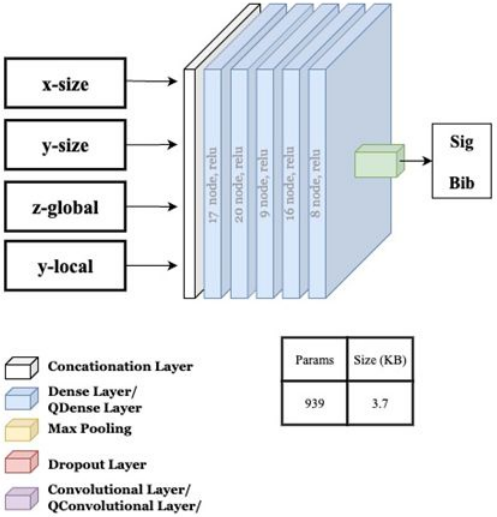
DATASET PARAMETERS

- PixelAV outputs a cluster shape in a region of interest
  - cluster "image" of deposited charge
  - x-profile, y-profile
  - x-size, y-size
- We shift the coordinates of the region of interest to center at the centroid of charge
  - y-local, z-global are charge centroid coordinates



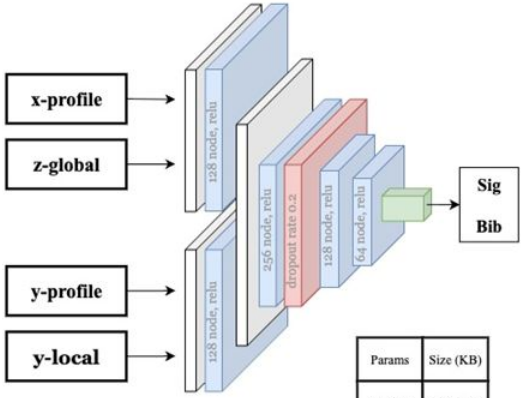
NEURAL NETWORK ARCHITECTURES

Model 1



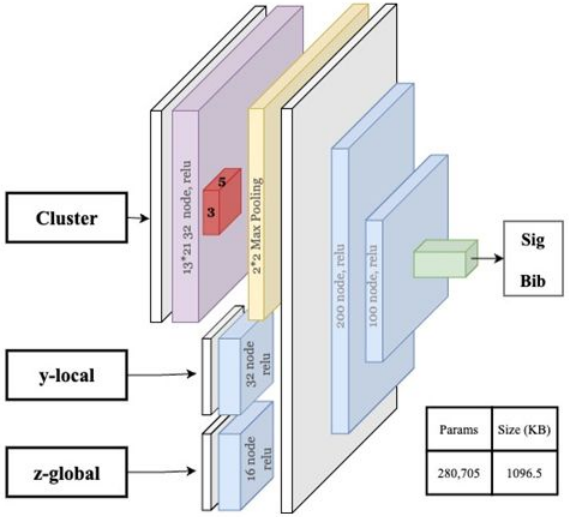
Params	Size (KB)
939	3.7

Model 2



Params	Size (KB)
111,873	437.0KB

Model 3

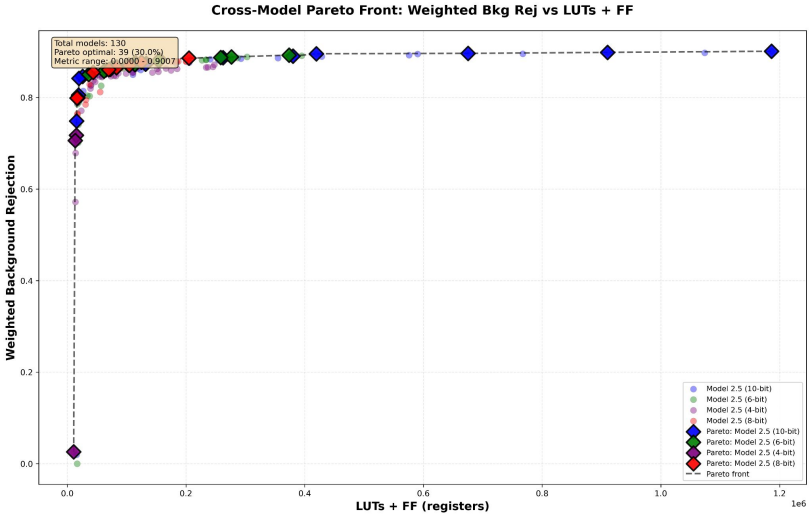


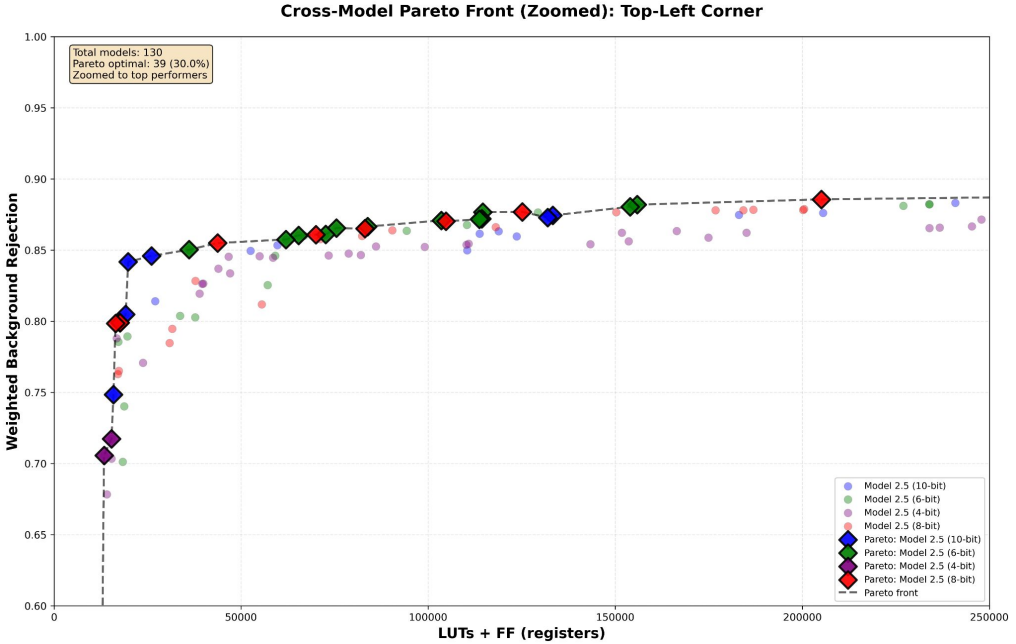
Params	Size (KB)
280,705	1096.5

- Note: Exact architecture (number of nodes) will change as we experiment with model size based on synthesis with hls4ml

Model 2 (Eric You); inputs unquantized, unnormalized

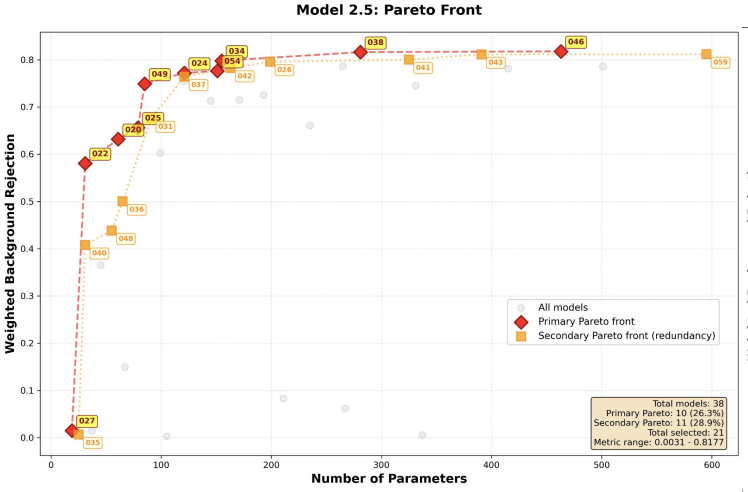
- Eric investigating model 2, which has x/y_profile, y_local, z_global as inputs
- At each quantization level, synthesized models on the pareto front for FPGA and use number of registers to estimate hardware usage
- Evaluated BIB background rejection at 99%, 98%, 95% signal efficiency
- Next will redo this with input quantization, normalized dataset



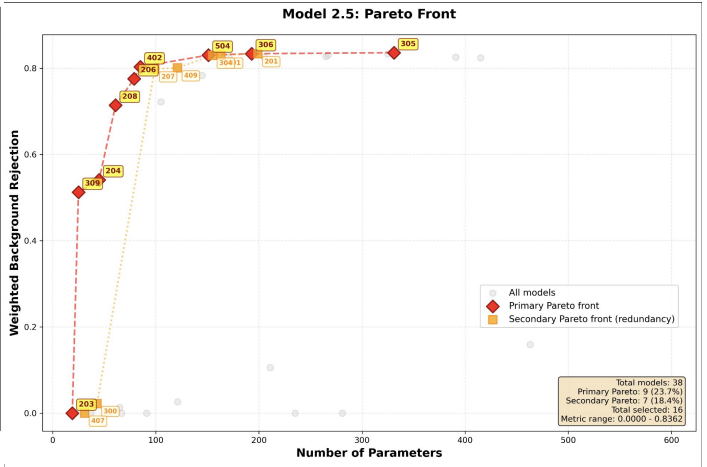


Model 1 (Ryan Michaud); inputs unquantized

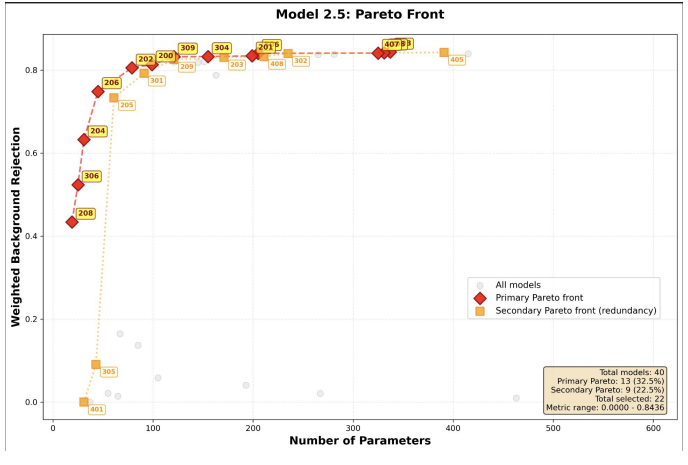
4w0i



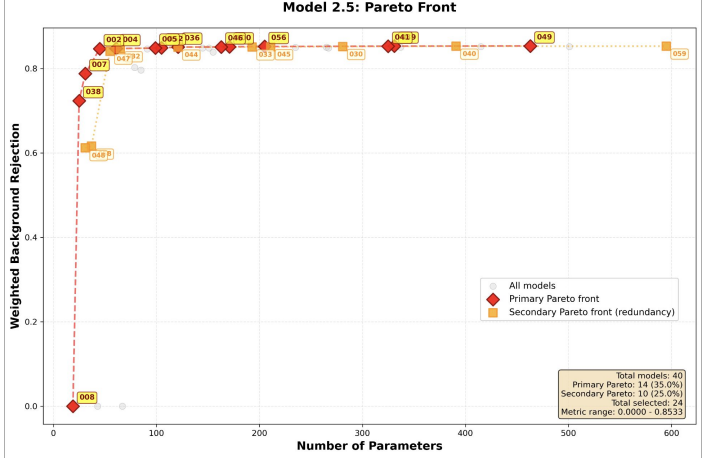
6w0i



8w0i



Unq.

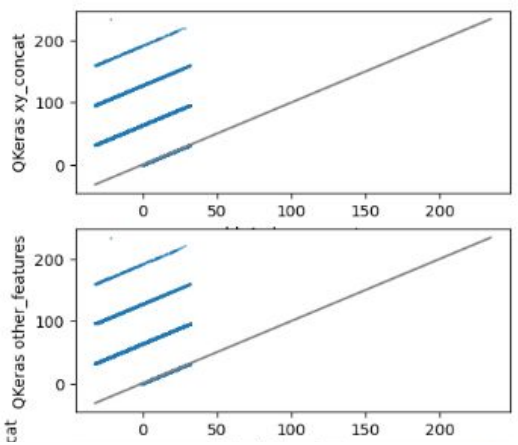


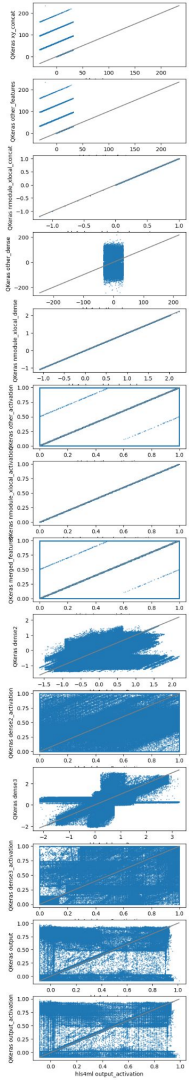
HLS verification

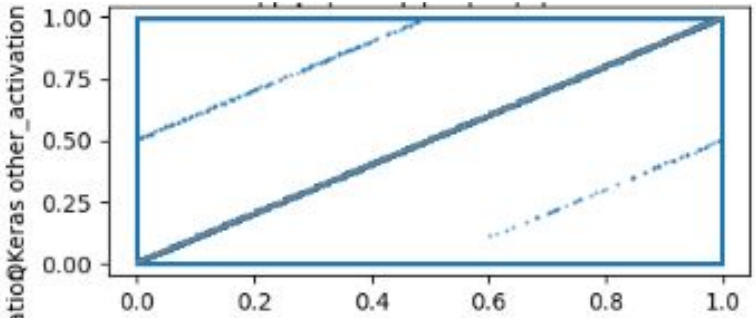
- HLS has a bit precision for every number. Qkeras only specifies coefficients
  - Weights, biases, and **results** of matrix multiplication at each layer
  - **Shift, slope**, and result of activation function
  - For the **intermediary numbers** Qkeras keeps as floats, hls uses default precision
- By default, HLS precision is <16,6> (16 total bits, 6**-1** integer bits, 1 sign bit)
  - Bits that overflow the allocation rollover, so 37 = 100101 -> -11010 = -26
  - Allegedly there should be some way so arithmetic saturates, not clear to me how to do this
- In dense layers if you add several numbers, you can get a result that is >1 and has integer bits, so you need integer bits for the results at each node
- Inputs are normalized to [-1,1] when maximum is reasonable, or log “normalized” to small numbers in case of clusters
  - So the numbers inside the network should stay close to 0, shouldn’t need many integer bits
- If hypothetically, one forgot to normalize the xprofiles and yprofiles, then inputs like 16.818283 could lead to numbers like 37.36195 appearing in the network and overflow to a totally different number, messing up the result

HLS Verification Model 2: original

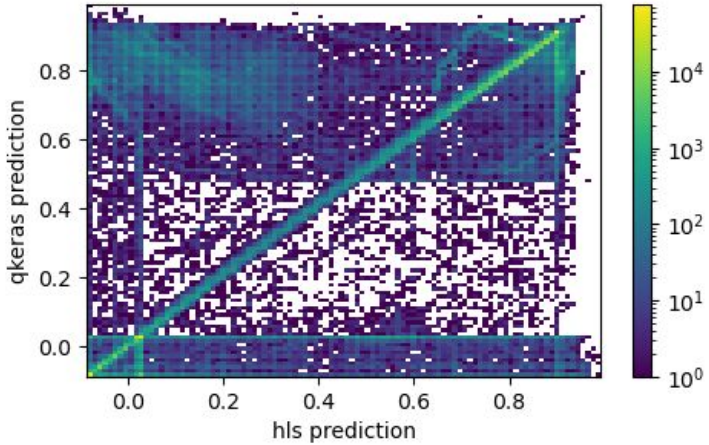
- Note: I actually realized the problem from looking at the printouts of the network trace, but the plots show it much nicer
- After tracing nodes and seeing the overflow, discovered that x/y profiles were unnormalized





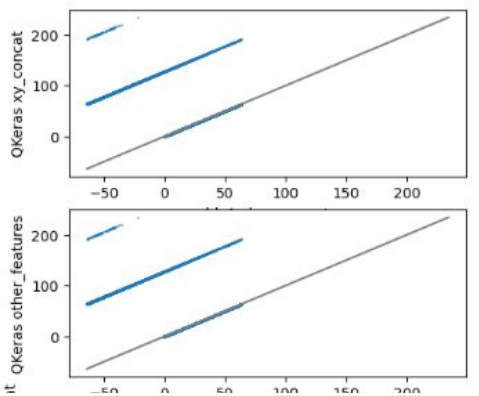


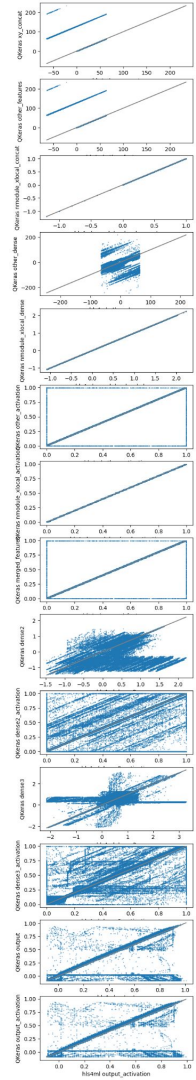
node value in qkeras vs. hls

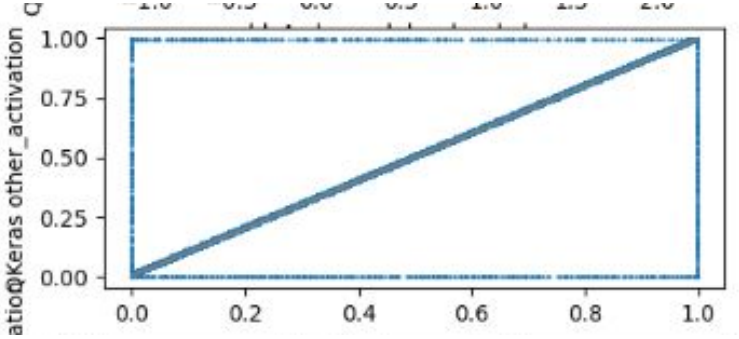


HLS Verification Model 2: Original inputs, more integer bits for node results

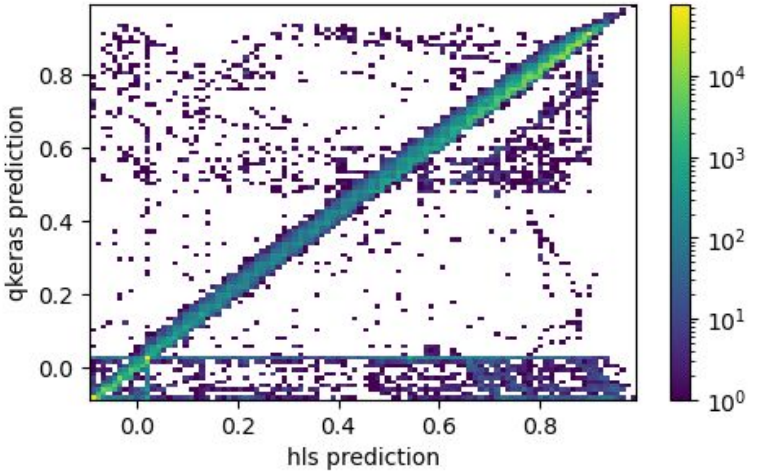
- From spot checking the trace of the network, found lots of examples of -37.etc, 57.etc so 5+1 integer bits is not enough, so tried 6+1 integer bits and improved validation





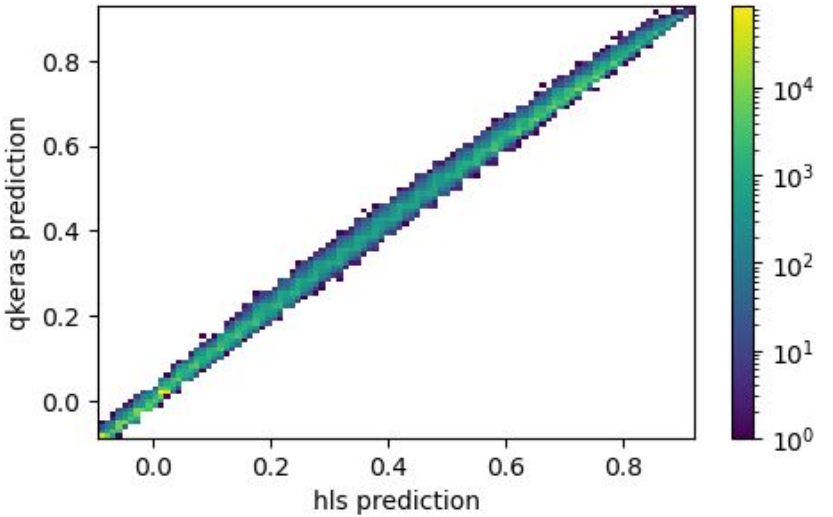


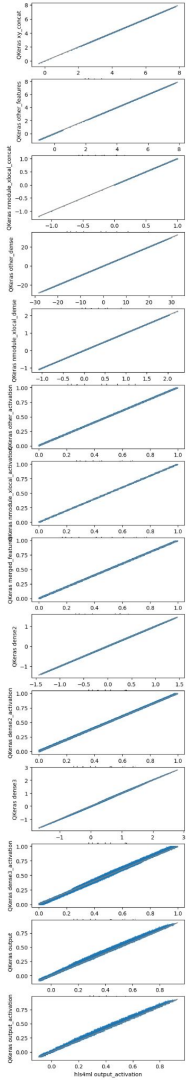
node value in qkeras vs. hls



HLS Verification Model 2: new “normalized” inputs

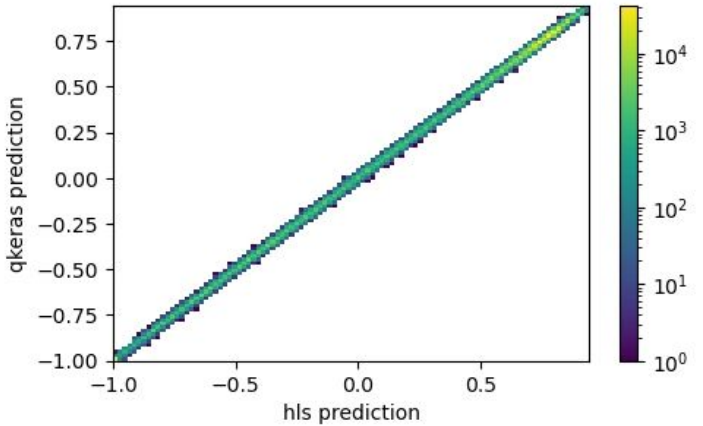
- I applied log “normalization” to the x/y profiles and regenerated the tf records
- Tried feeding these directly to the pretrained network
- Note: will need to retrain the network and redo this, but hls/qkeras now agree even with <16,6>
  default hls precision





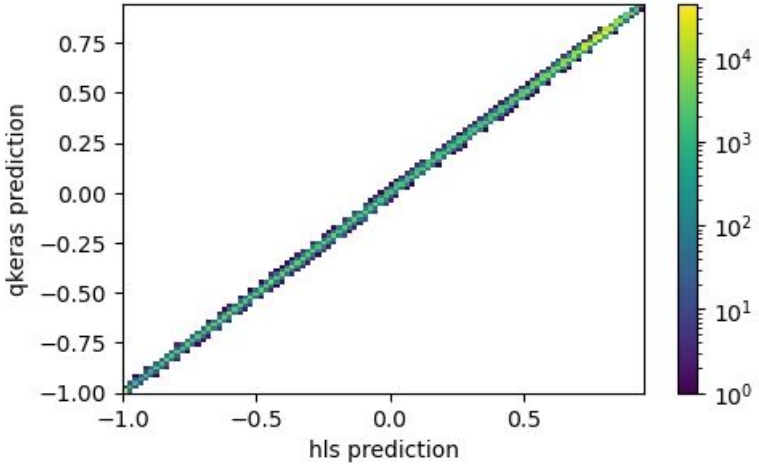
HLS verification Model 1

- 5+1 integer bits is plenty for model 1 since inputs are (log) normalized, so hls vitis and qkeras agree
- From trace printouts of this particular model 1, intermediary layers always have results in (-2,2), but the last dense layer sometimes has a result >2
  - So 2 integer bits is enough, but 1 is not enough for the last layer
- We should be able to reduce the number of bits used in result layers and hence further reduce size of synthesized model 1's

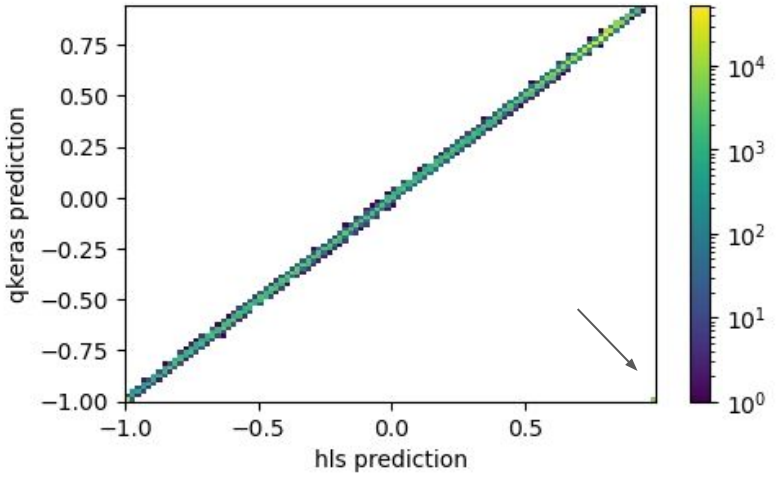


HLS Verification of Model 1 with not enough integer bits

Default precision: fixed<16,3>



2+1 integer bits is plenty!



precision fixed<16,2>:

1 bit is not enough