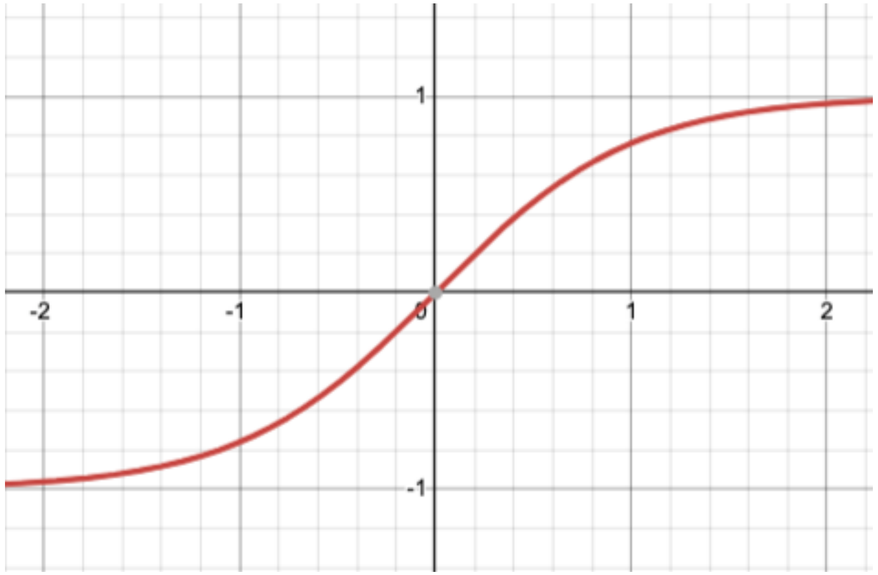
Quantized Network!

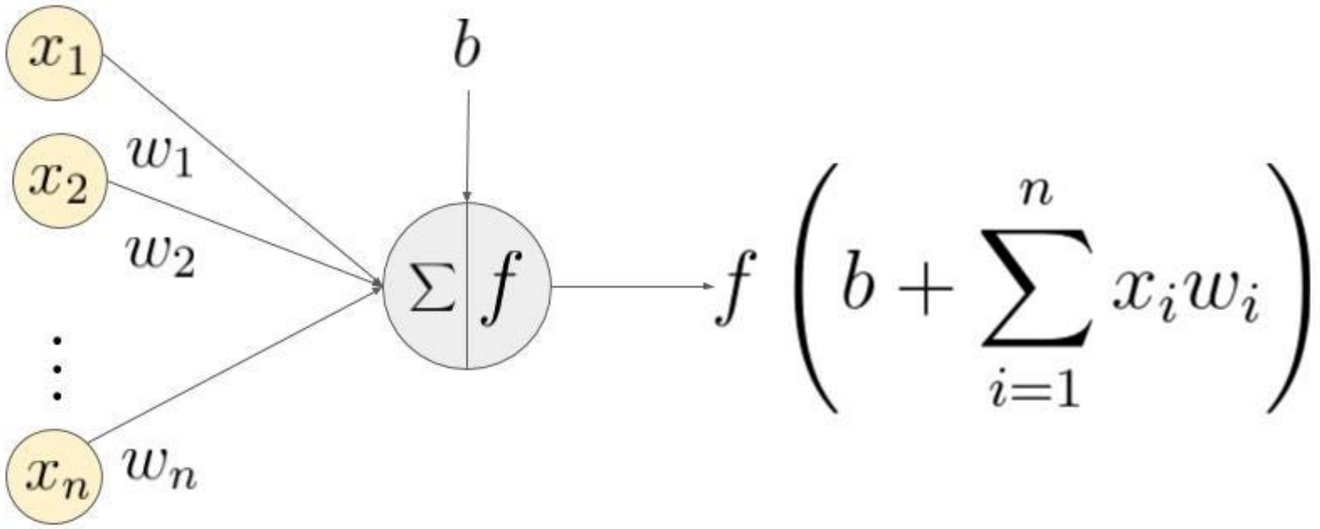
4/4/2025

Abraham Holtermann

Quantization Basics

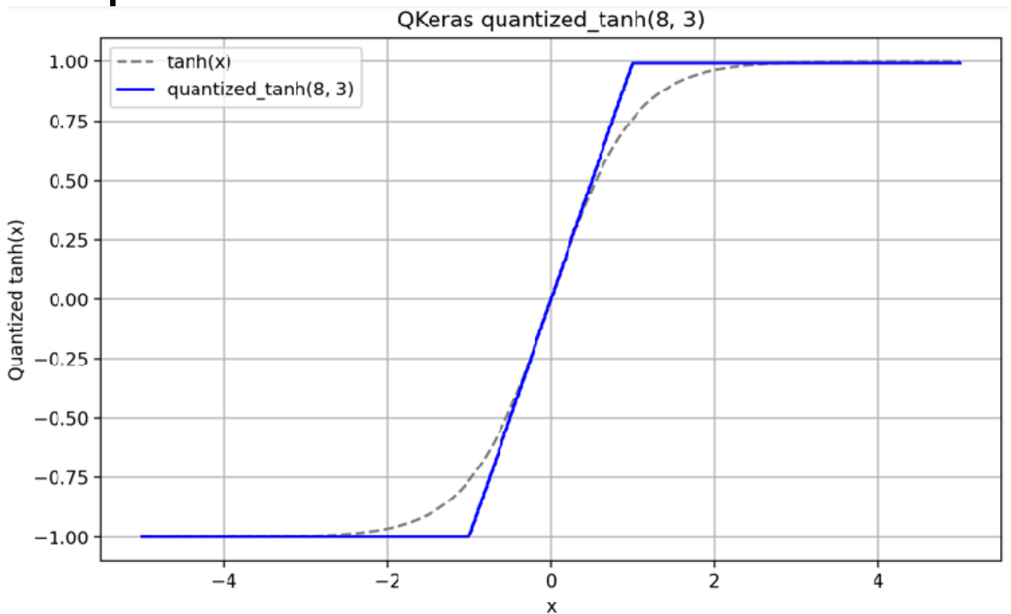
- Each node evaluates an activation function on a sum of weights (w) * inputs (x) + bias (b)
- Default: 32 bit floating point numbers
- **Quantized:** <X,Y> X bits above the decimal, Y bits below using clamp function
- Quantization applied to weights, bias, inputs, activation function





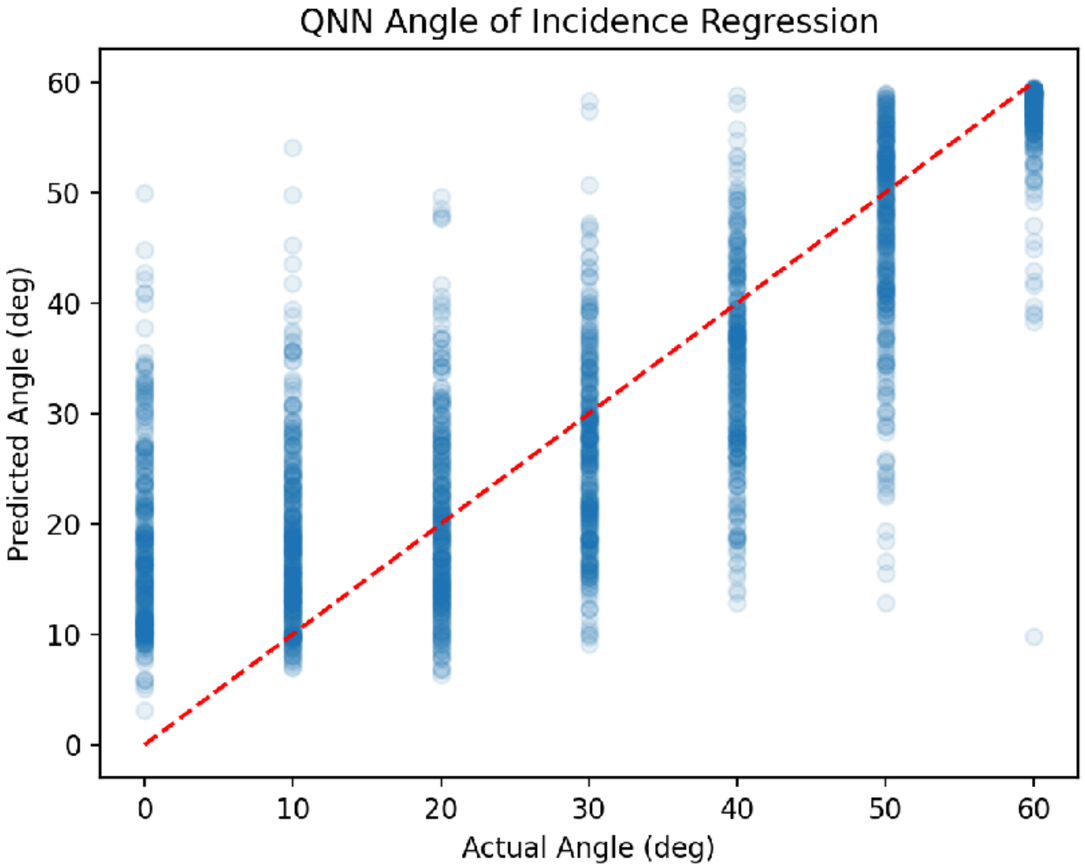
Our Quantized Network

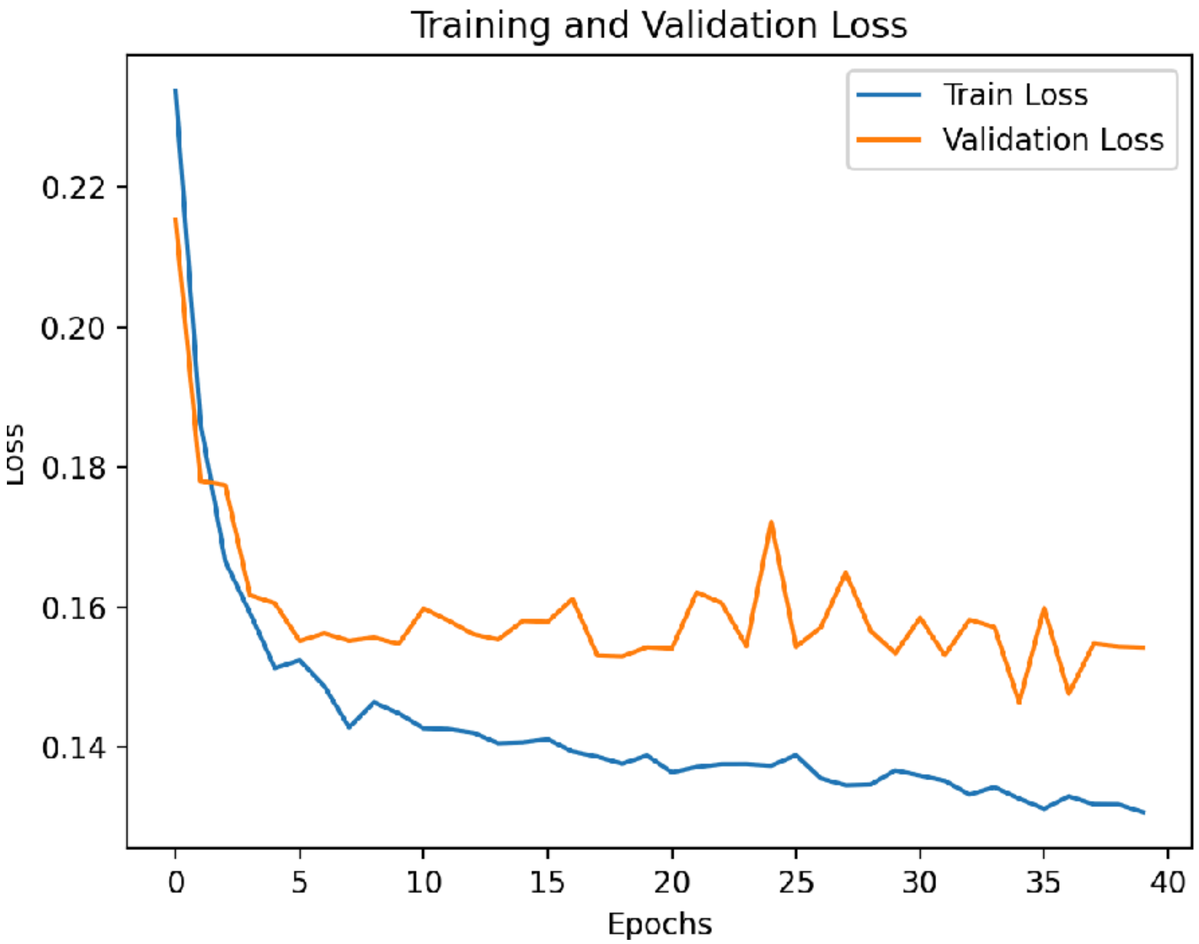
- effective_efield/Qregression.py
- QKeras + HLS4ML packages required
- <8,4> Precision instead of 32 bit floating point precision
- Produces HLS / ASICS compatible(?) output




```python
model = keras.Sequential([
    QDense(64,
           kernel_quantizer=quantized_bits(8, 4, alpha=1),
           bias_quantizer=quantized_bits(8, 4),
           input_shape=(X_train_scaled.shape[1],)),
    QActivation(quantized_tanh(8, 3)),

    QDense(32,
           kernel_quantizer=quantized_bits(8, 3),
           bias_quantizer=quantized_bits(8, 3)),
    QActivation(quantized_tanh(8, 3)),

    QDense(16,
           kernel_quantizer=quantized_bits(8, 3),
           bias_quantizer=quantized_bits(8, 3)),
    QActivation(quantized_tanh(8, 3)),

    QDense(1,
           kernel_quantizer=quantized_bits(8, 4),
           bias_quantizer=quantized_bits(8, 4)),
    QActivation(quantized_tanh(8, 3))
           #activation='tanh')

])
```

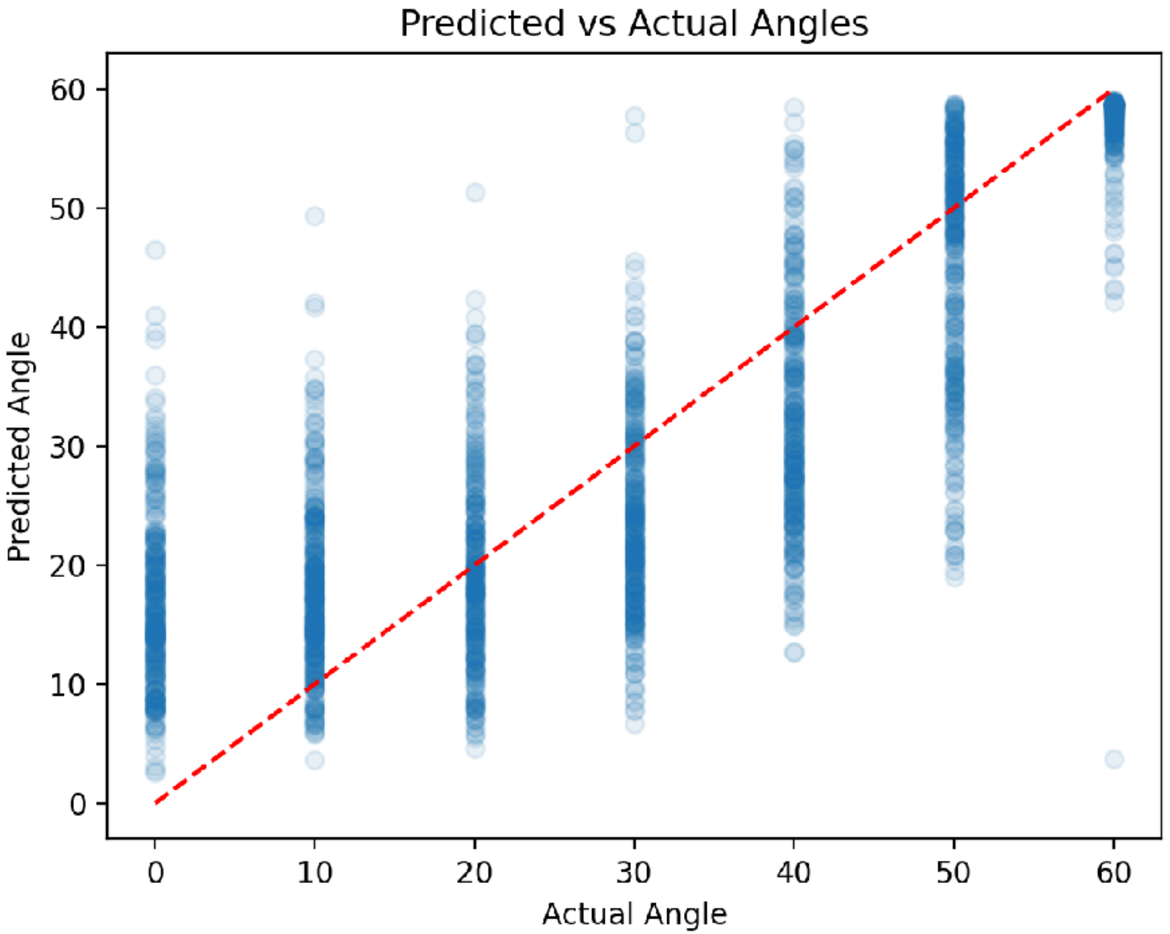
Results

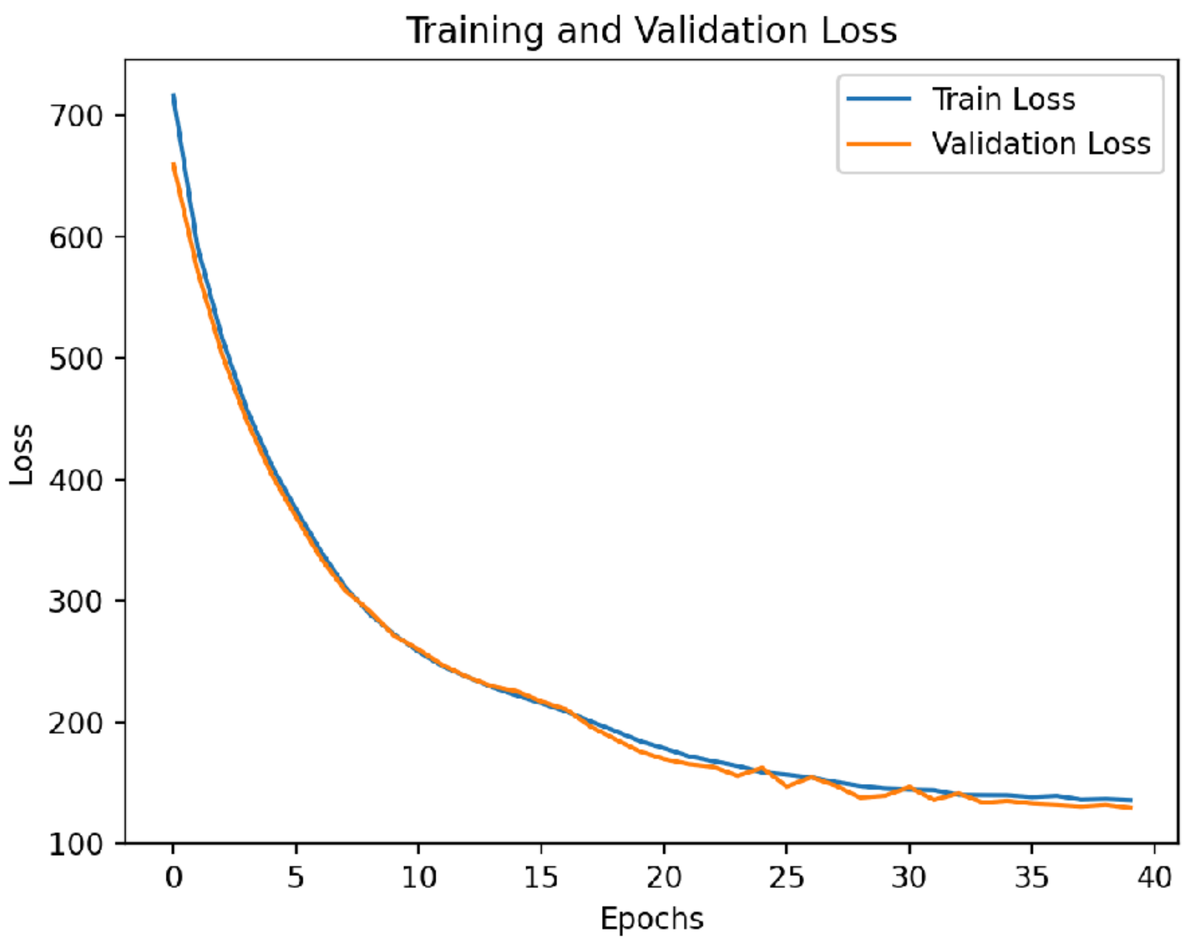




- MSE = 140 -> 11.8 Degrees off
- Substantial overfitting after 5 epochs

Comparison to old network





MSE = 146

Considerations moving forward

- **Optimizing this network**
  - Other forms of compression: (pruning etc.)
  - Latency / Initiation Interval
  - Reuse times
  - Power
  - Newer metrics (confusability matrix?)
- **Big Picture**
  - Other physics signals to find (beyond beam halo)
  - “Cylinder” and other more realistic simulations