Simple quantization problem – 1

Problem description:

Please choose a ML model you like (complexity should be at least similar to ResNet-18 for MNIST/cifar10, with layers more than 10), the weights, input and output per layer are floating-point 32 format by default. **Without re-training**, quantize weights, input and output per layer to 8-bits, and minimize its impact on the final classification accuracy.

Note: please do the output quantization before the activation function. For simplicity, just consider ReLu or ReLu6.

Note: please choose a pretrained model with high quality (For example, 95% accuracy for MNIST is bad quality, >99% accuracy for MNIST is good quality. Good quality model means its performance should not be too far away with the SoTA in similar model size).

Note: Please DO NOT re-train model in anyways.

DO NOT use any existing quantization tools (which means you just quote the existing quantization functions), please prepare your answers to the following points:

1. Explain the dataflow of one complete convolution layer in your neural network. It should include convolution, activation, and de-quantization (if needed) steps. Explain the bit-accuracy at each step.
2. Show one example of how you quantize one layer of weights to 8-bit, explain the cost function you used. Are there better cost functions?
3. Show one example of how you quantize input/output of one convolution layer to 8-bit, explain how you do the quantization here and how do you minimize the error.

Optional:

4. Can you quantize the NN to 6-bit or even lower for weights, and input/output per layer? What would be the concern?
5. If we apply quantization before and after the activation (one 8-bit quantization at the output of convolution, another 8-bit quantization at the output of activation), what will be the impact on final accuracy?

You can use any programming languages (Python is preferred), but the post-training quantization code should be open-source and written by yourself.