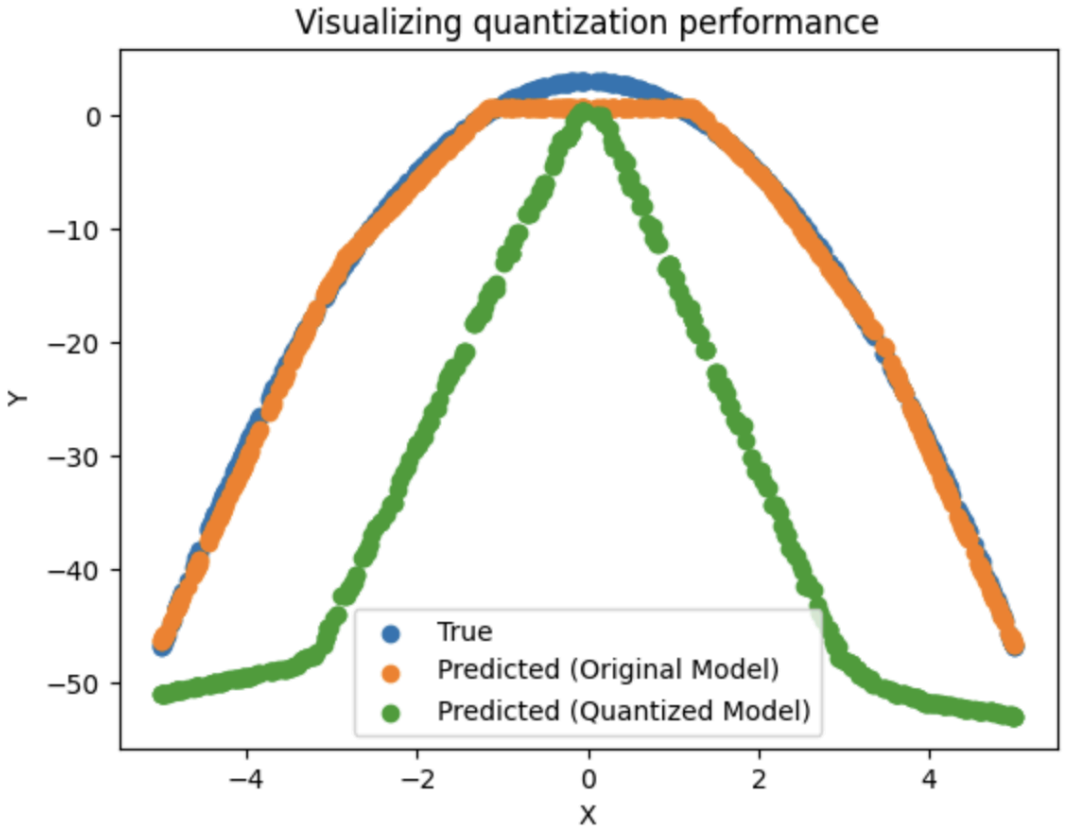
Task: Build a layer-wise quantizer from scratch



Github: https://github.com/saathvikpd/DSC180AB_Capstone/blob/main/Experimentation.ipynb

One result in my task that I found interesting was that the quantized model had more abrupt curves in the plot (visualized above). This was potentially because the neural net was already very simple (4 hidden layers, 5 neurons each), so compressing it to lower precision might have made the black-box function even simpler.

The main challenge here was learning the intricacies of building a quantizer. I had to do a lot of research on when to quantize and dequantize inputs and also structure the quantizer class in a clever way. These took the longest amount of time because I had to do some trial-and-error runs to get it right. However, now I have a nice template for layer-wise quantization that I can use for other algorithms as well.

One choice I made in this process was building a quantizer class to make the code look clean. Another reason for building the class was for efficiency. When an object is initialized, the class automatically computes the scaling factor and the quantized weights. Also, since PyTorch does not allow models to have integer weights, I chose to store the quantized weights as float values itself just for the sake of the experiment. Of course, for future use, I would have to find a

different way to convert weights to integer values and store them in the model given that memory and compute optimization is the main motivation behind quantization.