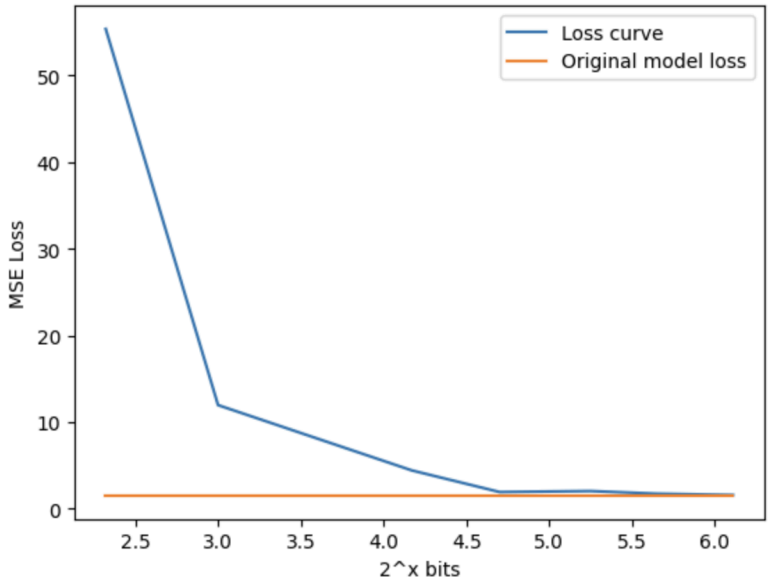
Task: Writing some code implementing a toy version of a neural net and quantizing it using rounding function

I found it interesting how increasing the granularity of the rounding function brought the loss of the quantized neural net down. Intuitively, this makes sense, but plotting the loss and observing the elbow in the plot was enlightening.



One thing I struggled with was figuring out how to access the weights in a torch model. I have worked with PyTorch before, but it has been a while since I last worked with the package. There was a bit of a learning curve, but I was able to find some helpful resources online.

In my investigation, I made deep copies of the weight matrices for the neural net because I wanted to test our different granularities for the quantization function by looping over parameters. Creating deep copies made this task easy because the original matrices would remain intact, and I would be able to have access to all the original weights and quantized weights in memory. I wanted to make sure I could store them all in memory, so I could collect data about loss and plot them. The alternative to deep copying the weight matrix would be to retrain the model every time I tested a new parameter for the quantization function. This would have been an inefficient method to carry out the task.

GitHub: https://github.com/saathvikpd/DSC180AB_Capstone