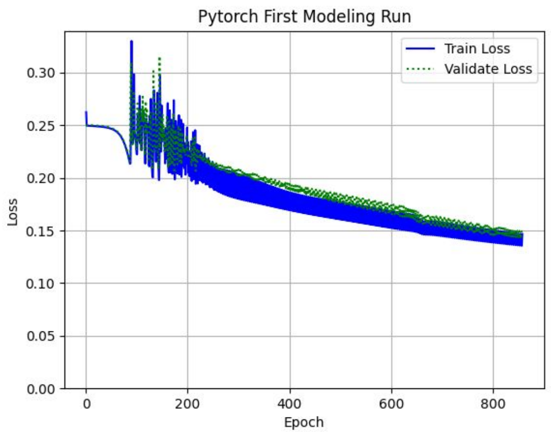
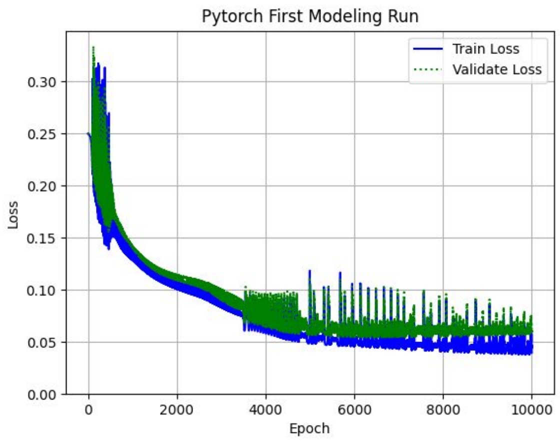
Two charts and less than two hundred words describing your comparison. Did the models converge similarly? Did they generalize similarly? What differences remain between the implementations? How much does randomization affect the PyTorch run (vs your implementation where the framework set a seed)?





I got an accuracy simple of 91.91% (89.37% - 94.44%) which is similar to what I got for my own hand-created two-layer fully connected model.

In terms of generalization, it seems the pytorch model is far harder to train, with loss declining far slower (although the code runs a lot faster per epoch) and in a farm more jittery manner. This is probably the effect of randomized initiation that is not the case for my own model with a constant seed. As you can see in the first graph, in some cases, the initial values are so bad, it never really converges that well. Another architectural change is that in my model, my training values are between 0-1 whereas it is normalized in pytorch by data transformers. Further in my model, each sample updates the weights whereas here it may be aggregated by pytorch SGD optimizer (not sure)? Perhaps batching would be required and a learning schedule to compensate for the jitter and initialization. Because we never saw validation cross training (it would be nice to know how to do early stopping with pytorch also).