a. I can think of several reasons why using autograd can be beneficial. The most significant reason should be because of the GPU-accelerated computations. When gradients are autocomputed it could easily be converted into computations that can be executed by the GPU. Several other reasons that I can think of are related to the possibility of some functions having highly complicated derivatives. The ones that we coded were trivial derivatives that could be coded as constant run-time operations. However as functions get more complex more computational power it may need, and more prone to error it can be.

b. There are several reasons that make pytorch tensors preferred over numpy arrays for common deep learning operations. One main reason is that pytorch tensors are designed to be streamlined for GPU base operations whereas numpy arrays are designed for CPU based operations.GPU's can more efficiently execute matrix and vector operations such as matrix multiplication and dot product. Another main reason is that pytorch tensors keep fields for differentiations i.e. gradients, which is a very common operation in backpropagation. Hence further streamlines neural network operations.