4a.

Autograd makes it easier to backpropagate through a network by storing gradient functions for each node in the network on a forward pass, leading to a backward computation graph which can be used for easy backpropagation. This is similar to what we did in our code (storing inputs to each layer to compute gradients later) but is faster since it doesn't actually compute any Jacobians, but rather just computes Jacobian vector products directly for each node. This can make model building easier since, for large networks, the Jacobians themselves get very complicated. The autograd system breaks each operation down into primitive operations with simple derivatives to make the entire process much faster computationally. Another useful feature of Pytorch is the ability to freeze parts of the network for fine-tuning, an ability that our current model does not have.

4b.

PyTorch uses tensors instead of numpy arrays because despite them being very similar, tensors are designed to run on GPUs. This makes them faster to work with than numpy arrays, which can be especially important when doing the many array operations involved in training a neural network.