Deep Learning Project 2

Fabian Kropfhamer

December 18, 2022

1 Introduction

The target of this project is to create a deep-learning framework. The challenge is hereby to create a framework only using basic tensor operations from pytorch. Especially without the autograd function from pytorch.

2 Architecture

The architecture for the framework is split up into activation functions, error functions, layers, and the sequential neural network itself. There is a layer function that serves as a base class. In base functionality, it keeps track of the last input.

The fully connected class inherits from the base layer class. It implements the forward and backward functions. It also holds weights and a bias.

The framework additionally includes three activation functions, sigmoid, tanh, and ReLU. Even though the backward function does not use the learning rate it still receives it but discards it to keep a common interface among the layers.

Lastly, the framework comes with one loss function and one optimization algorithm. The loss function implements the mean squared error function. It has a similar interface as the layers but does not inherit from the layer base class. The implemented optimization algorithm is stochastic gradient descent. Unlike pytorch, the optimization algorithm is not a separate class but is implemented through the backward functions of the layers as well as the loss function. The backward functions update the weights when present and return an error that can be passed to the previous layer.

For more convenient usage of the framework, it contains a utility class for creating a sequential neural network. This class receives a list of layers. Then the instance can be called and returns a prediction for the given input. It also has a backward method that passes the given error backward through all its layers.

3 Task

The task is to classify if a point is in the radius of $1/\sqrt{2\pi}$ the point (0.5, 0.5). This is a binary classification task. 1 corresponds to in the radius and 0 to out of the radius. The dataset is created by sampling random points between (0, 0) and (1,1). The train and test dataset consists of 1000 samples.

4 Results

With the proposed framework it is possible to create a neural network that can do binary classification. A network with 3 fully connected layers can achieve up to 90% accuracy on the given task. Furthermore, this network converges within 60 epochs when it is optimized with the stochastic gradient descent algorithm. Additionally, It achieves a loss lower than 0.1 on the task. It is also more powerful than a perceptron and can solve non-linear classification like the xor problem.

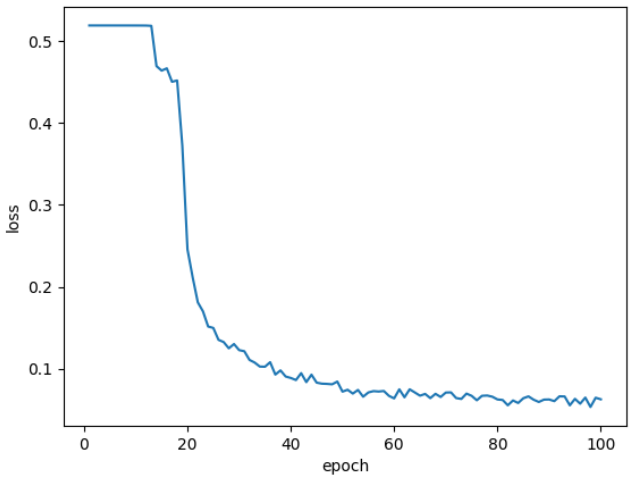

Figure 1: Plot of losses

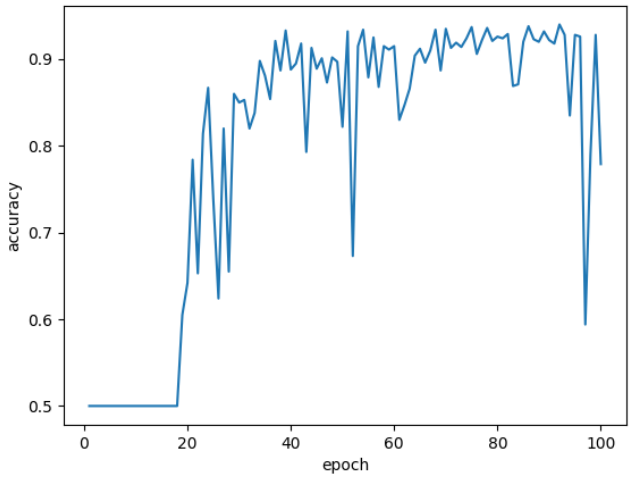

Figure 2: Plot of accuracies

5 Conclusion

In conclusion, it is possible to implement a framework for neural networks with only basic tensor operations. Also, networks built with this framework can handle and perform simple tasks quite well. Furthermore, this project shows what good job frameworks like pytorch and tensorflow do. These frameworks hide a lot of complexity which otherwise would be hard to handle. Additionally, it is difficult to keep the framework user-friendly while at the same time keeping performance in mind. Including different network architectures while reusing code is challenging.