Lazare Plisson-Arcos

CE6146 Introduction to Deep Learning

Assignment 1

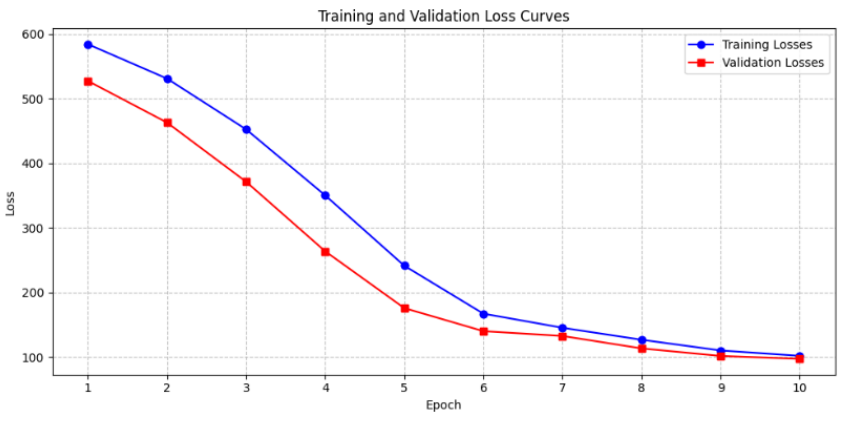
I will be using google Colab for this assignment.

Task 1: Implement a Feedforward Neural Network for Regression

Explanation of the chosen architecture, hyperparameters, and observations from the learning curves:

- At the start my architecture was 3 fully connected layers (64, 32, 1 Units). The more layers and units there is, the more complex pattern you can capture but it has drawback. It also increases the model's capacity and may require more data. Our architecture can capture complex pattern in the data but it's not so much, so it avoids overfitting. The output layer must be 1 unit because we need one neuron because regression. Indeed, we want to predict a single continuous value.
- Then I reduced to 2 fully connected layers because I noticed that the plotting of training and validation losses were smoother this way.
- I'm using ReLU because it is a simple and easy to use activation function, it's very efficient.
- A test size of 20% is a common practice and is good for the size of our data.
- I defined the random state to an arbitrary number (11) so I will get the same split of data every time I run the code.
- I used a batch size of 64 because it's standard starting point.
- The input size is set to match the number of features in the Boston Housing dataset (13). This is a common practice as it ensures that the model's architecture is compatible with the data.
- I started at a learning rate 0.001 because it is a reasonable starting point before tuning. After tuning it was at 0,01.
- I used Adam because it's very popular, converges faster and has great results.
- I used Mean Square Error because it's efficient for regression problem.
- I started with 50 epochs because it is a good number for our dataset but then I saw by plotting the graph that 10 is enough.

When using a learning rate of 0.01, the decrease of the rate of loss is quicker. I reduced the number of fully connected layer from 3 to 2 to make the curves smoother. We succeeded in having both training and validation losses decrease smoothly and converge to a low value. This indicates that the model is learning effectively and generalizing well. It's neither overfitting nor underfitting.



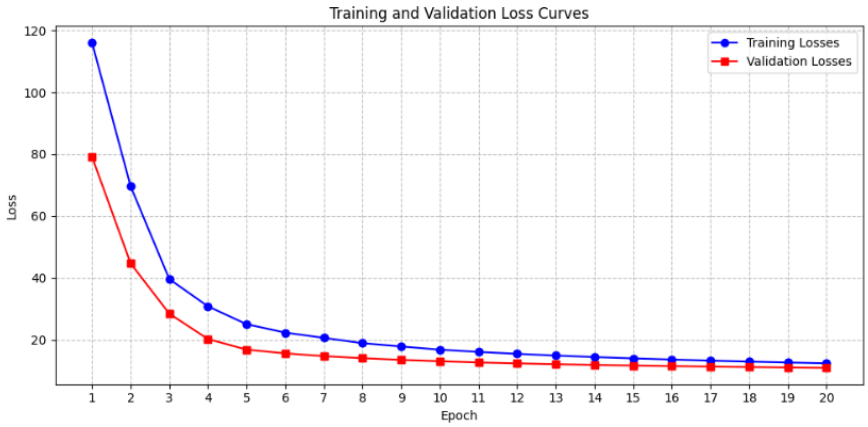
Task 2: Breast Cancer Dataset

Explanation of the chosen architecture, hyperparameters, and observations from the learning curves:

Neural Network Architecture:

- I'm using two linear fully connected layers and one activation function ReLU. I have chosen to use only 2, to avoid overfitting and because the model was already adapting well. The input layer has 30 features (corresponding to the 30 attributes in the dataset). I have chosen an output layer of 1 neuron with a sigmoid activation, because it works well for binary classification. It outputs values between [0, 1] that represent class probabilities.
- We used a binary cross-entropy loss function because it's a binary classification.
- I started by standardizing the features to have a mean of 0 and a standard deviation of 1. This helps the neural network converge more effectively.
- We continue with the optimizer Adam for the same reason as before (because it's very popular, converges faster and has great results).
- I started at a learning rate 0.001 because it is a reasonable starting point before tuning. After tuning I kept the learning rate at 0,001 because the model was already learning fast.

The result is pretty good because every time the training losses and validation losses are decreasing smoothly and converge to a low value. I reduced the epoch to 20 because it's already to a very low value at that time.



Task 3: Fashion-MNIST Dataset

Explanation of the chosen architecture, hyperparameters, and observations from the learning curves:

This time I will be using tensorFlow over PyTorch because it was way faster to compute in my case.

Neural Network Architecture:

Input Layer: The input layer flattens the 28x28 pixel images into an 1D array of 784 numbers. (28*28=784).

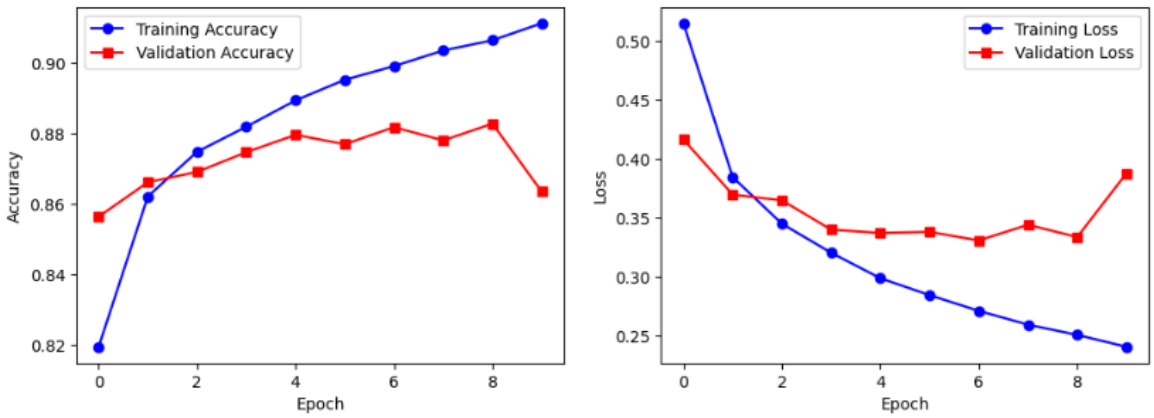
The first fully connected layers have 128 units and ReLU activation. This layer learns to capture patterns and features in the data. I'm using ReLU because it is a simple and easy to use activation function, it's very efficient.

The second fully connected layers have 10 units and sofmax activation. It's the output layers. I'm using 10 units because there are 10 clothing classes in Fashion-MNIST. I'm using softmax because it's good for image classification.

Preprocessing:
This time I did not do the standardization because it's image data. Instead, I did a normalization (feature scaling) by dividing by 255 so each pixel value is now between 0 and 1.

Hyperparameters:
I started with a low number of 10 epochs because it was taking a long time to compute.

The result was this:



I was not satisfied because even though the validation loss and validation accuracy are getting better at the start. It stops getting better or slow too much.

I implemented a learning rate scheduling. It changes the weights better as it converges.

I also used dropout regularization to avoid overfitting. I used common values for these changes.

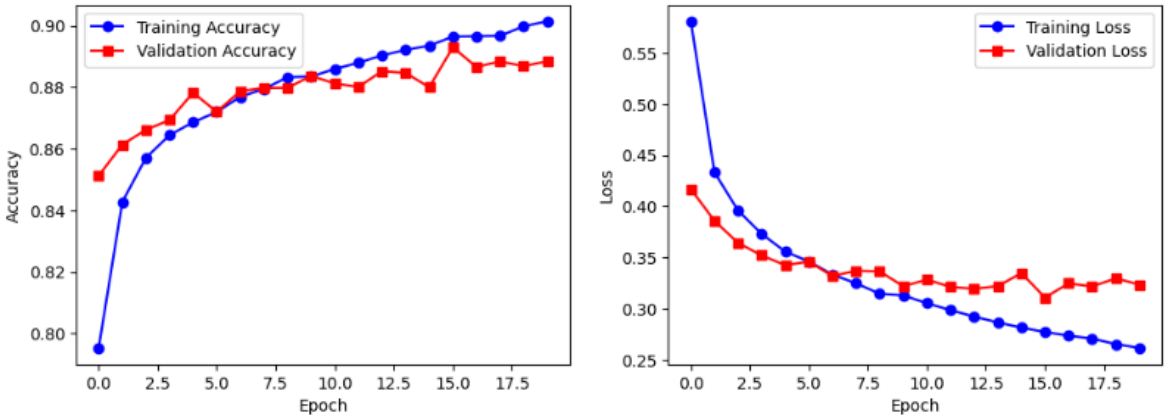
I added an early stopping, so my computer stops the program if the validation loss stopped improving. This way I'm not wasting time waiting for no improvement. I used a patience of 5 because otherwise it was stopping to early, when improvement was still possible.

At first, I was using a dropout of 0,5 but the result was better with 0,2.

I raised the maximum number of epochs to 20 because it was still improving after 10 epochs. I also raised the initial learning rate to 0,02.

I reduced the layers to 64 to avoid overfitting.

The final result is this:



We can see that the accuracy is higher, and the validation loss is a bit better.