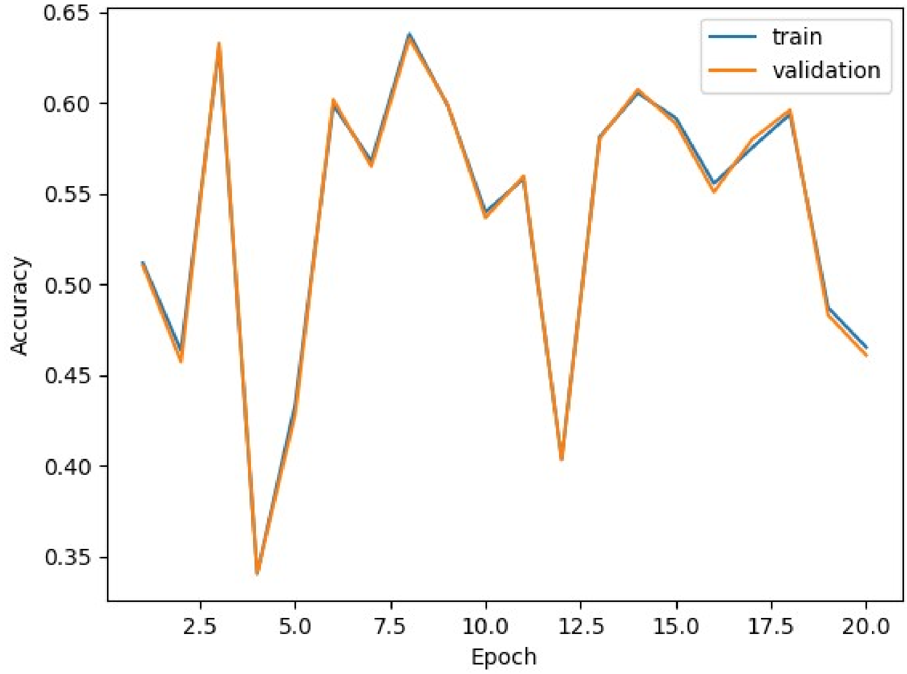
Homework 1

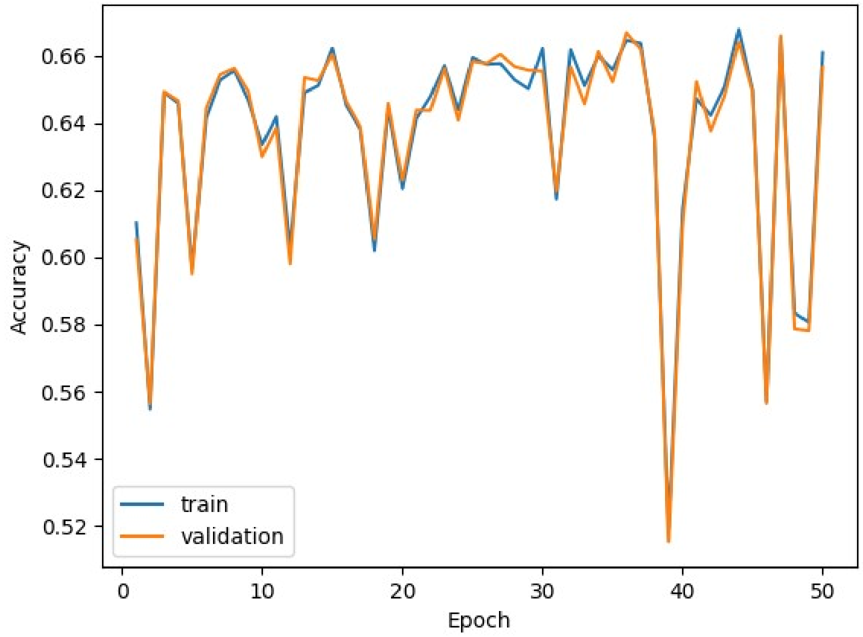
- João Câmara 95598
- João Silva 79730

Question 1

1. Linear Classifier
   a) Perceptron



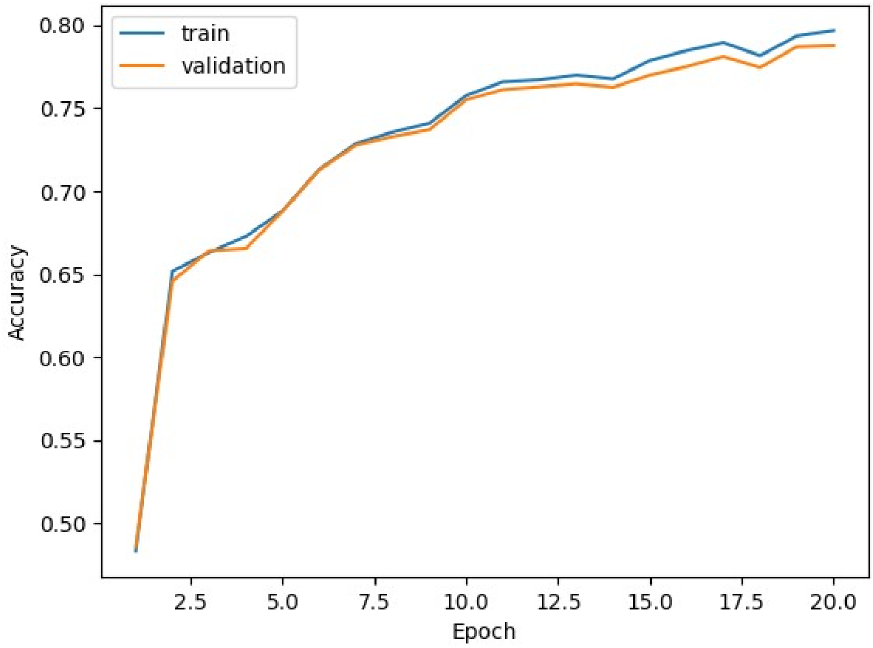
   b) Logistic Regression

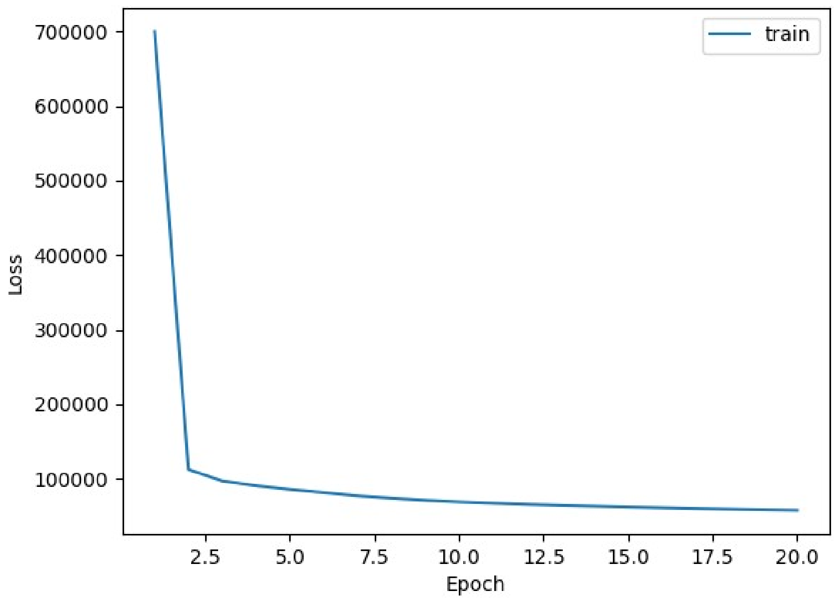


2. Multi-layer Perceptron

“A logistic regression model using pixel values as features is not as expressive as a multi-layer perceptron using relu activations. However, training a logistic regression model is easier because it is a convex optimization problem”
This statement is true. Logistic regression is a simple linear model that is limited in its expressiveness compared to a multi-layer perceptron with non-linear activations like ReLU. Nevertheless, training a logistic regression model is easier because it is a convex optimization problem, which means that it has a single global minimum and gradient descent is guaranteed to find this minimum. In contrast, training a multi-layer perceptron is more complex due to non-convex optimization, which can have multiple local minima, making it harder to train.

b)

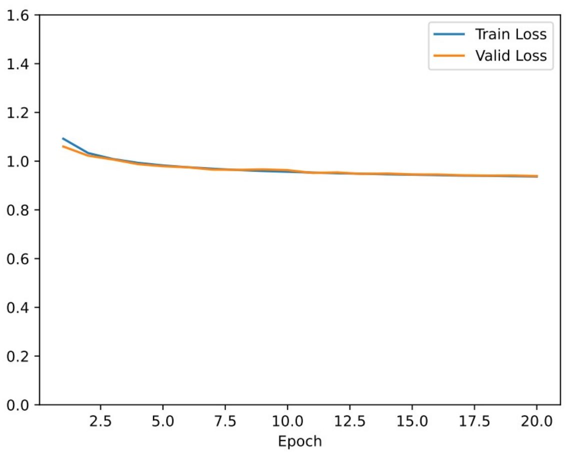


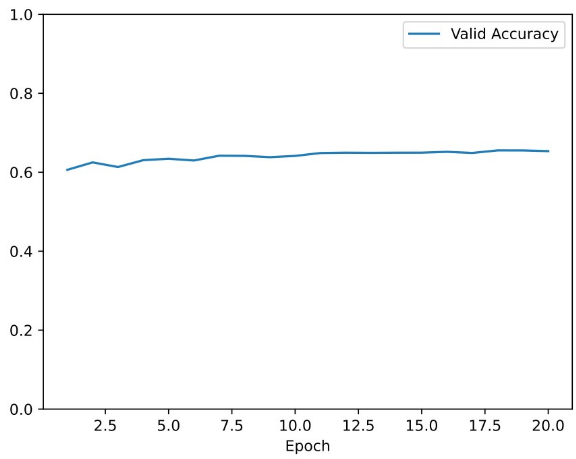


Question 2

Q2.1

A Learning Rate of 0.01 had the highest final validation accuracy, = 0.6535, with a final test accuracy of = 0.62. No overfitting.

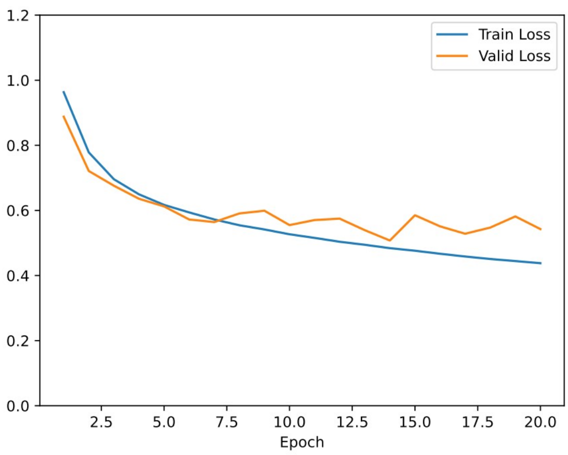




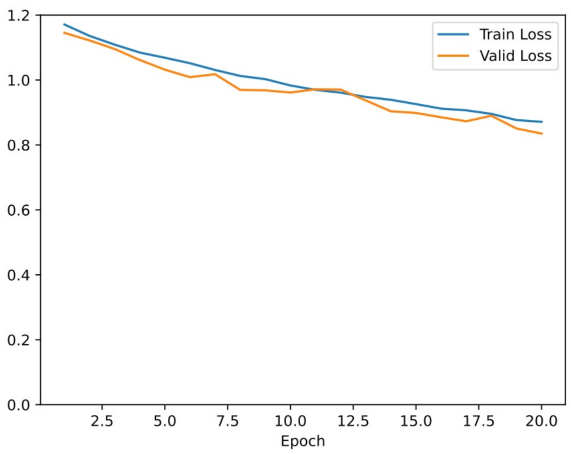
Q2.2

a) Batch Size of 16 had the best test accuracy, = 0.7675, of the two (, as well as final validation accuracy, = 0.8121), despite its Training and Validation Losses gap growing wider with Epochs. However, as a trade-off for its performance, its runtime was 2min 40s, whereas a Batch Size of 1024 ran for only 39s. As expected, since a smaller Batch Size translates to more samples per batch with DataLoader() (Batch Size 16 had 6093 samples, whereas Batch Size 1024 had 96 for the datasets used), and, therefore, more calculations/training iterations.

b)

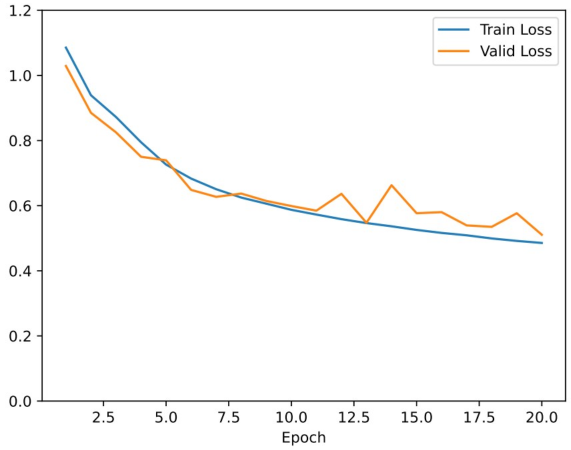


Batch Size = 16

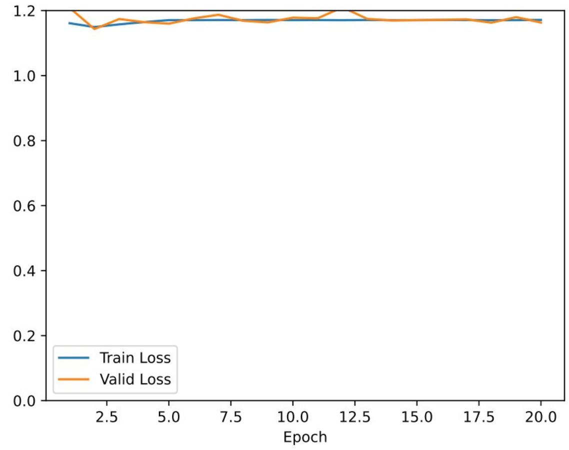


Batch Size = 1024

c) The best performing Learning Rate was 0.01 (with 0.1 as a close second) with a final validation accuracy = 0.8166, and the worst was Learning Rate of 1, with a significantly worse performance, = 0.4721, i.e., it tried to learn too fast. The Rate with the best final test accuracy, = 0.7675, was 0.1 (same configuration as the best one in 2.2.a)). Though not as poorly as 1, 0.001 also underperformed, i.e., it was too slow/would require more epochs, therefore, the optimal Learning Rate for this configuration ought to be in the interval]0.001,1[.



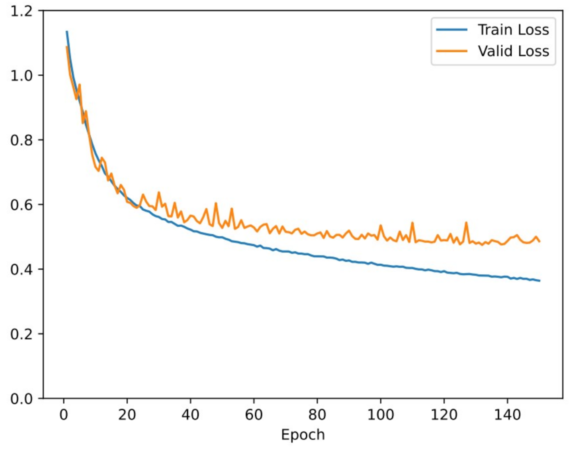
Learning Rate = 0.01



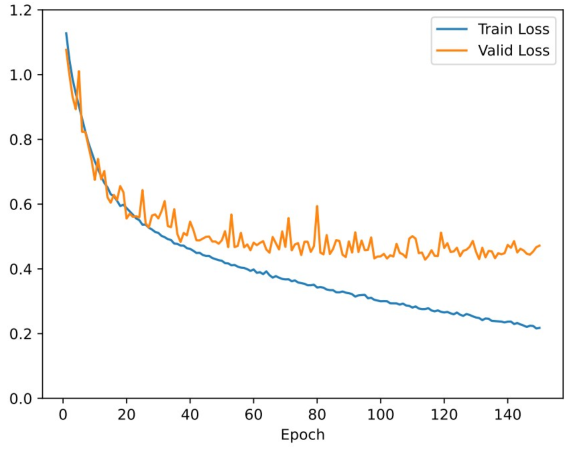
Learning Rate = 1

d) The gap between Training and Validation Losses increases with Epochs, particularly for the baseline configuration, therefore suggesting overfitting is occuring. This gap is diminished by the regularization techniques. The best configuration in terms of final validation accuracy is the baseline one, with an accuracy = 0.8572, followed by L2 regularization (= 0.8504), and, finally, the worst of the three, Dropout regularization (= 0.8311). However, in terms of final test accuracy, Dropout yields the best score = 0.7618. L2 regularization closes the Loss gap at the cost of lower final test and validation accuracies. Comparatively, Dropout regularization closes said gap even further, at a higher cost of final validation accuracy, however, improving final test accuracy, overall being the preferable option.

e)



Dropout



No regularization

Question 3

a) If we consider D = 3, A = -1 and B = 1, we get a non linearly separable problem, which the perceptron cannot handle.

b) Consider function f as the following pseudo code:
X = An array filled with 1's and -1's
IF SUM(X) ≥ A AND SUM(X) ≤ B THEN 1 ELSE 0

We can simulate a network architecture that implements this logic.
Being N the shape of the input, the first weight matrix is 2 by N, where the first line is filled with 1's and the second line is filled with -1's. The first bias is 2 by 1 with elements -A and B. The second weight matrix is one by two being two ones, and the second bias is -2.

João worked on questions 1 (code and report) and 3, and João Silva worked on question 2 (also code and report.)