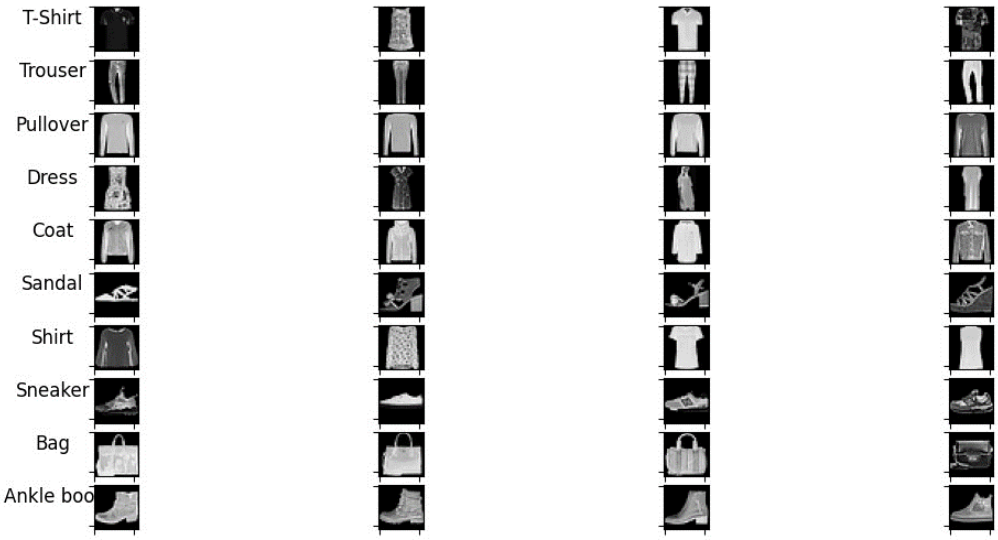
Deep Learning Ex2

Yonatan Golan : 208980888
Noam Diamant: 208520262

*There aren't special instructions for running the code

Part 1 - Visualize the Data



Part 2 - Logistic Regression Classifier

We ran through several different options for values of the hyperparameters (batch size, learning rate and regularization coefficient), we calculated loss and accuracy, and we took the combination of hyperparameters which gave us the max accuracy on the validation set. Then we took the params of the model – w which fit those hyperparameters and we tested our model and save the results to "lr_pred.csv" as required.
Results:

Hyperparameters			performance			
Batch size	Learning rate	Regularization coefficient	Train accuracy	Train loss	**Validation accuracy**	Validation loss:
128	0.001	1e-07	0.87237	0.37139	0.85402	0.41807
128	0.001	0.001	0.87042	0.59641	0.85321	0.63828
128	0.01	1e-07	0.87915	0.34382	0.84446	0.45907
128	0.01	0.001	0.87011	0.74957	0.84875	0.81113
128	0.05	1e-07	0.84741	0.72077	0.80598	0.93494
128	0.05	0.001	0.82194	1.71856	0.80321	1.74450
256	0.001	1e-07	0.87219	0.37132	0.85411	0.41811
256	0.001	0.001	0.87138	0.61879	0.85313	0.66301
256	0.01	1e-07	0.87415	0.36036	0.84268	0.46771
256	0.01	0.001	0.86663	1.00627	0.84821	1.06137
256	0.05	1e-07	0.83944	1.13314	0.77366	1.89113
256	0.05	0.001	0.81346	3.57250	0.76304	4.05942
512	0.001	1e-07	0.87178	0.37177	0.85393	0.41828
512	0.001	0.001	0.87174	0.63231	0.85330	0.67760
512	0.01	1e-07	0.84883	0.57930	0.84643	0.52787
512	0.01	0.001	0.84320	1.78736	0.84482	1.68659
512	0.05	1e-07	0.83565	1.60015	0.83473	1.38329
512	0.05	0.001	0.80071	9.81590	0.84580	9.17333

Our main insights on the effect of the Hyperparameters are:

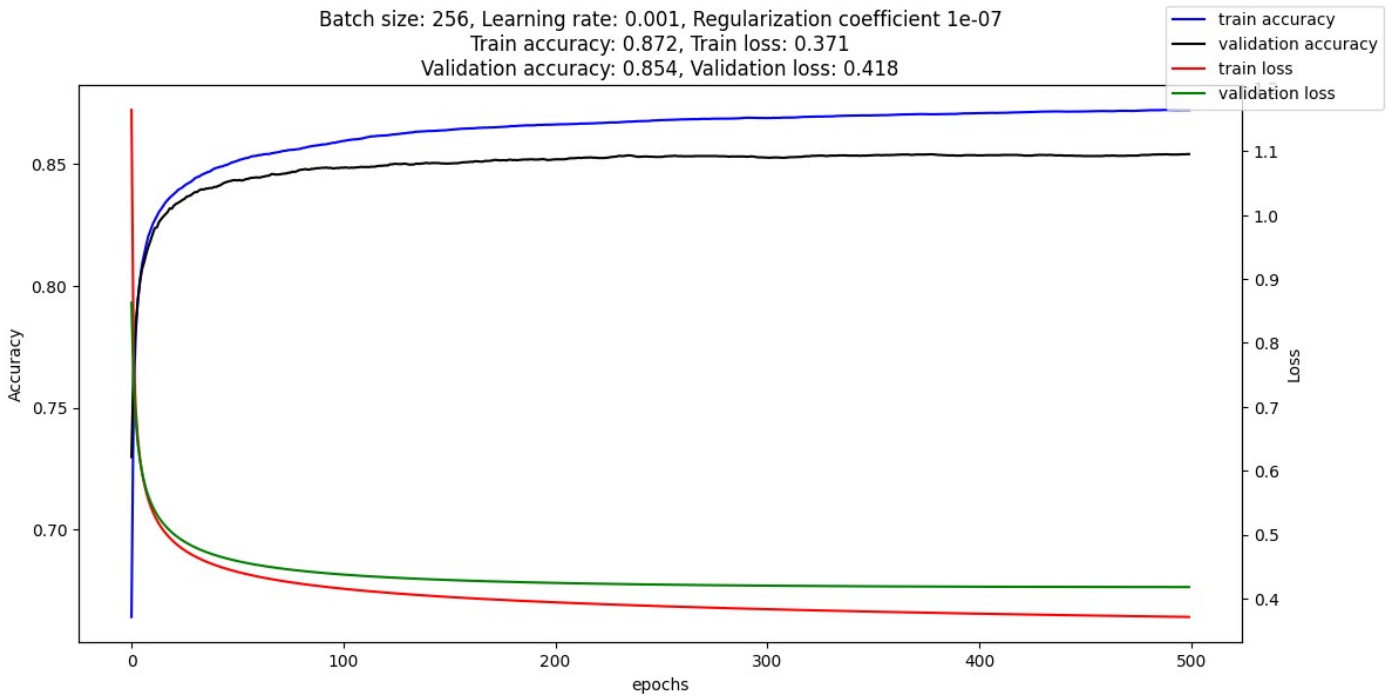
1.**Batch Size**: Increasing batch size tends to provide more stable updates to the model parameters, but larger batch sizes might lead to slower convergence. In our results, batch sizes of 128 ,256 and 512 were tried, and generally, smaller batch sizes seem to perform slightly better (especially in a look on the loss and in the cases where the learning rate and the Regularization has bigger values).

2.**Learning Rate**: The learning rate controls the step size during optimization. Too high a learning rate might cause the model to overshoot optimal parameters, while too low a learning rate might result in slow convergence. From our results, a learning rate of 0.001 appears to be more stable across different regularization coefficients (mainly in compared to 0.05).

3.**Regularization Coefficient**: Regularization helps prevent overfitting by penalizing large parameter values. A smaller regularization coefficient (1e-07) generally leads to better validation performance in our results.

Overall, the logistic regression model demonstrates reasonable performance, but there is room for optimization and fine-tuning of hyperparameters to further enhance its effectiveness.

Attaching here the results of best combination (argmax of validation accuracy):



Part 3b - Neural Network with One Hidden Layer

Similar to the previous part – we experiment several different combinations with different hidden layer sizes, activation functions, batch size, learning rate, regularization coefficient, and dropout of the hidden layer. Results:

Hyperparameters						performance			
Batch size	Learning rate	Regularization coefficient	Activation Function	Hidden Size	Dropout Prob(keep)	Train accuracy	Train loss	**Validation accuracy**	Validation loss:
128	0.5	1e-08	relu	256	1	0.86103	0.40402	0.85384	0.42135
128	0.5	1e-08	relu	256	0.9	0.85772	0.41287	0.85268	0.42995
128	0.5	1e-08	relu	256	0.8	0.85779	0.41265	0.85304	0.42859
128	0.5	1e-08	relu	256	0.5	0.85413	0.42524	0.84732	0.44013
128	0.5	1e-08	relu	128	1	0.85815	0.41192	0.85259	0.42883
128	0.5	1e-08	relu	128	0.9	0.85792	0.41408	0.85259	0.42987
128	0.5	1e-08	relu	128	0.8	0.85605	0.41910	0.85080	0.43463
128	0.5	1e-08	relu	128	0.5	0.85250	0.42833	0.84688	0.44357
128	0.5	1e-08	relu	10	1	0.84795	0.44379	0.83866	0.46302
128	0.5	1e-08	relu	10	0.9	0.84978	0.43755	0.84045	0.45557
128	0.5	1e-08	relu	10	0.8	0.84701	0.44869	0.83929	0.46348
128	0.5	1e-08	relu	10	0.5	0.84194	0.45724	0.83705	0.47095
128	0.5	1e-08	sigmoid	256	1	0.80190	0.55493	0.80027	0.56334
128	0.5	1e-08	sigmoid	256	0.9	0.79739	0.56347	0.79652	0.57184
128	0.5	1e-08	sigmoid	256	0.8	0.79315	0.57267	0.79312	0.58109
128	0.5	1e-08	sigmoid	256	0.5	0.78368	0.59774	0.78187	0.60583
128	0.5	1e-08	sigmoid	128	1	0.79487	0.56911	0.79402	0.57723
128	0.5	1e-08	sigmoid	128	0.9	0.79328	0.57527	0.79179	0.58345
128	0.5	1e-08	sigmoid	128	0.8	0.78652	0.58878	0.78589	0.59711
128	0.5	1e-08	sigmoid	128	0.5	0.77455	0.61953	0.77464	0.62770
128	0.5	1e-08	sigmoid	10	1	0.70308	0.83127	0.69482	0.83867
128	0.5	1e-08	sigmoid	10	0.9	0.70426	0.85287	0.69830	0.85990
128	0.5	1e-08	sigmoid	10	0.8	0.69987	0.83325	0.69500	0.84087
128	0.5	1e-08	sigmoid	10	0.5	0.69281	0.91991	0.69054	0.92519
128	0.5	1e-08	tanh	256	1	0.85846	0.40198	0.85250	0.41649
128	0.5	1e-08	tanh	256	0.9	0.85717	0.40540	0.85223	0.41914
128	0.5	1e-08	tanh	256	0.8	0.85632	0.40916	0.85080	0.42281
128	0.5	1e-08	tanh	256	0.5	0.85232	0.41967	0.84830	0.43253
128	0.5	1e-08	tanh	128	1	0.85817	0.39976	0.85402	0.41377
128	0.5	1e-08	tanh	128	0.9	0.85737	0.40344	0.85205	0.41651
128	0.5	1e-08	tanh	128	0.8	0.85636	0.40671	0.85170	0.41995
128	0.5	1e-08	tanh	128	0.5	0.85266	0.41827	0.84768	0.43086
128	0.5	1e-08	tanh	10	1	0.83868	0.46797	0.83500	0.48262
128	0.5	1e-08	tanh	10	0.9	0.84835	0.45231	0.84143	0.47218
128	0.5	1e-08	tanh	10	0.8	0.84455	0.45994	0.84098	0.47516
128	0.5	1e-08	tanh	10	0.5	0.83511	0.49408	0.82937	0.50950

Our main insights are:

1. **Impact of Hidden Layer Size**: Varying the size of the hidden layer affects model performance. Larger hidden layer sizes (e.g., 256) tend to yield better accuracy and lower loss compared to smaller hidden layer sizes (e.g., 10). This suggests that a more complex model with a larger number of hidden units can capture more intricate patterns in the data, resulting in improved performance. It seems that the smaller layer size gave fair results in terms of saving effort and its simplicity in compared to the other layers.

2. **Activation Function Comparison**: Comparing the performance across different activation functions (relu, sigmoid, tanh), it appears that the relu and the tanh activation functions generally performs better in terms of accuracy and loss.

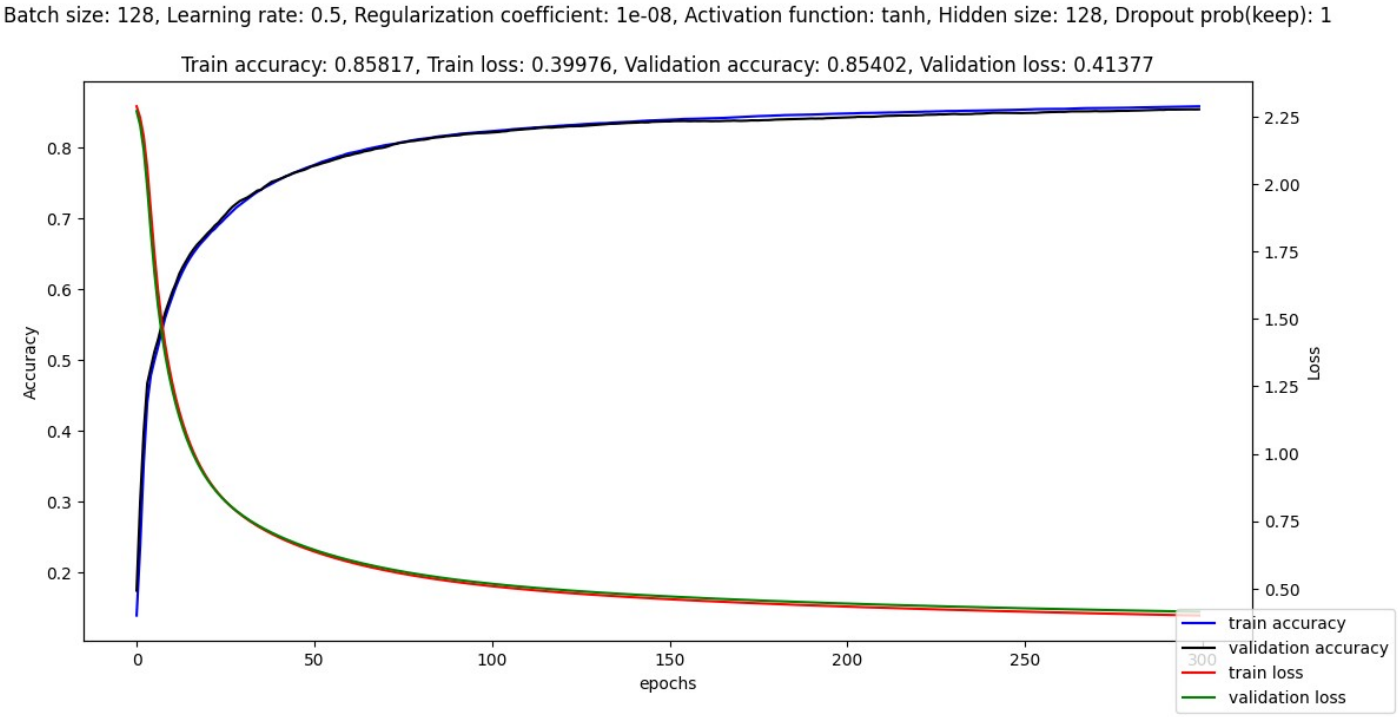
3. **Effect of Dropout Probability**: The dropout probability parameter had a minor impact on both train and validation performance metrics. Generally, higher dropout probabilities (e.g., 0.5) tend to lead to lower accuracy and higher loss values.

4. **Regularization Coefficient Impact**: The regularization coefficient is important for preventing overfitting, especially in scenarios with larger datasets or more complex models. In our case we used only one regularization coefficient value because of the big number of combinations we had and maybe trying another value could help us improve our results.

5. **Learning Rate and Batch Size**: Learning rate and batch size are also critical hyperparameters that can significantly influence training dynamics and model performance. It's essential to fine-tune these parameters along with others to achieve optimal results.

Overall, optimizing these hyperparameters based on the observed trends in performance metrics can lead to better model generalization and predictive accuracy.

Attaching here the results of best combination:



As the previous part we took the params of the model – w_1, w_2, b_1, b_2 which fits the hyperparameters which gave us the max accuracy on the validation set. We tested our model with those params and save the results to " NN pred.csv " as required.