Ahmed Fahmy

900160127

Assignment#2 PMDL

**Overview:**

* Source Code is attached, and comments explain main features. Random seed is inside the code.
* Initially, images were scaled and normalized using cv2 to 40x40, yet that was changed to 100x100 to increase accuracy
* Initially, normal SGD was implemented, as discussed below, but eventually Nesterov momentum was implemented and used
* It is possible to change the number of layers, the neurons in the hidden layers, as well as the activation function, as the sigmoid, tanh, relu and l\_relu are implemented.
* The alpha from the nesterov momentum and the learning rate are tunable hyperparameters
* The numerical differentiation using computational graphs is implemented
* Below will be a detailed explanation of the most significant runs made, and decisions why the subsequent iterations were modified.
* Class by class accuracies is included for the final model
* Final activation used was tanh. RELU and LRELU implemented but produced worse results. They were noticeably faster.
* The final model had (100x100x3) input neurons, 3 layers of 700 neurons, and 5 output neurons one for each class.
* The ACCR for each iteration us included. The best ACCR was **52.6%**

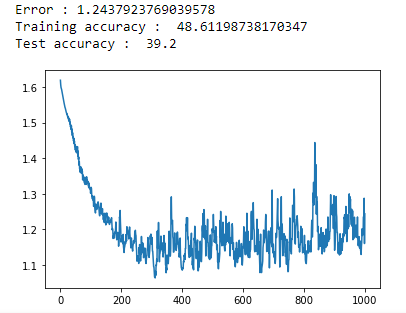
**Iteration By iteration progress:**

Graphs are Error vs Epochs

Using Sigmoid, 1 hidden layer of 400 neurons:

learning rate = 0.1, epochs = 1000:

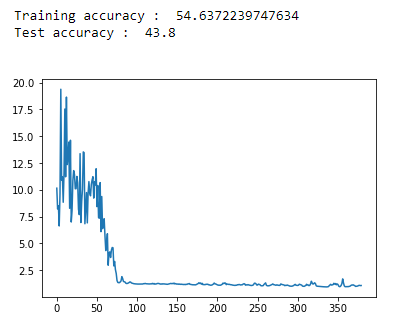
Significant evidence of overfitting as training > testing accuracy



USING TANH + Nesterov, 1 hidden layer of 400 neurons:

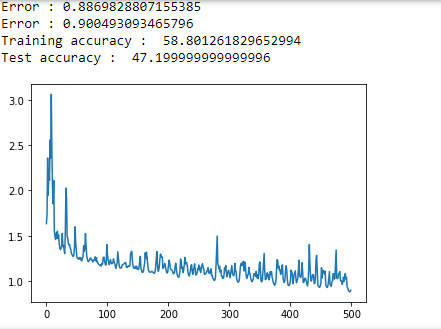
learning rate = 0.15, epochs = 400, alpha = 0.7:

Seemed like learning rate was too high as it reached low value quickly then stabilized



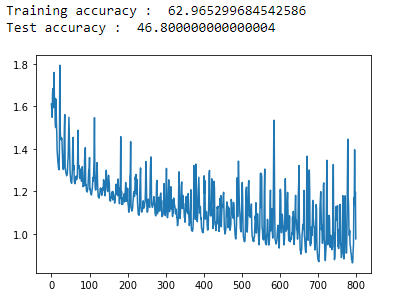
USING TANH + Nesterov, 1 hidden layer of 400 neurons:

learning rate = 0.02, epochs = 500, alpha = 0.6:

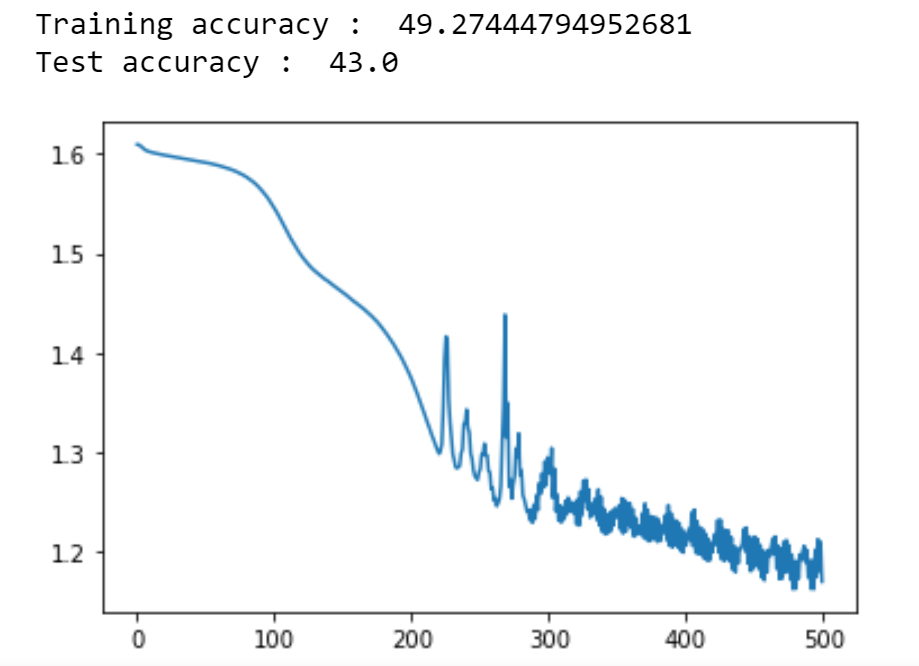


In all previous runs, images were scaled down to 40\*40 pixels, so 40\*40\*3 input nodes. Now, I attempted to change that to 100\*100\*3 input nodes (100\*100 pixels).

learning rate = 0.02, epochs = 800, alpha = 0.6:

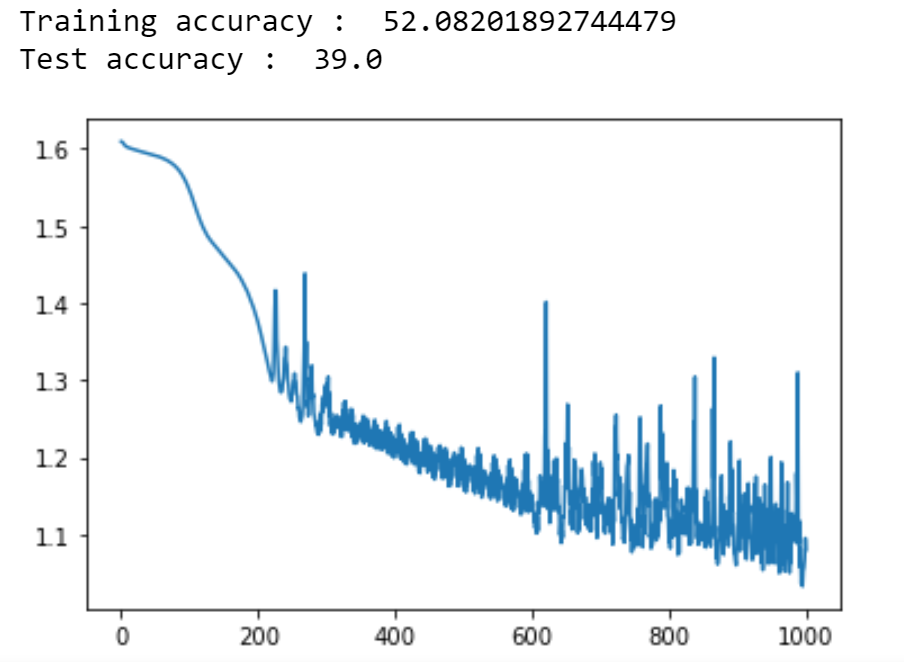


2 hidden layers, 150 neurons each. Learning rate = 0.02, epochs=500, alpha=0.6

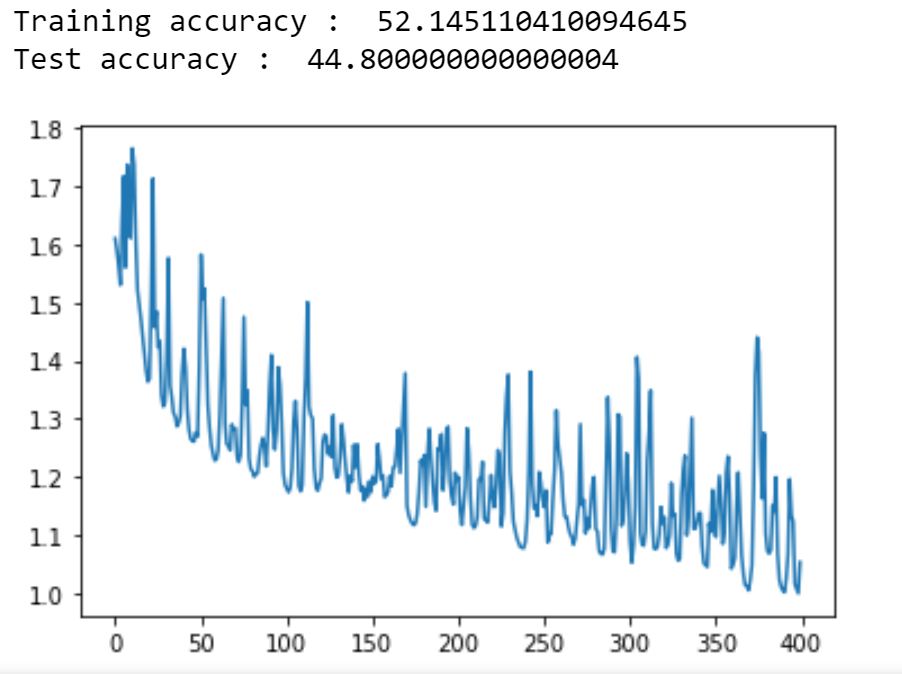


2 hidden layers, 150 neurons each. Learning rate = 0.02, epochs=1000, alpha=0.6

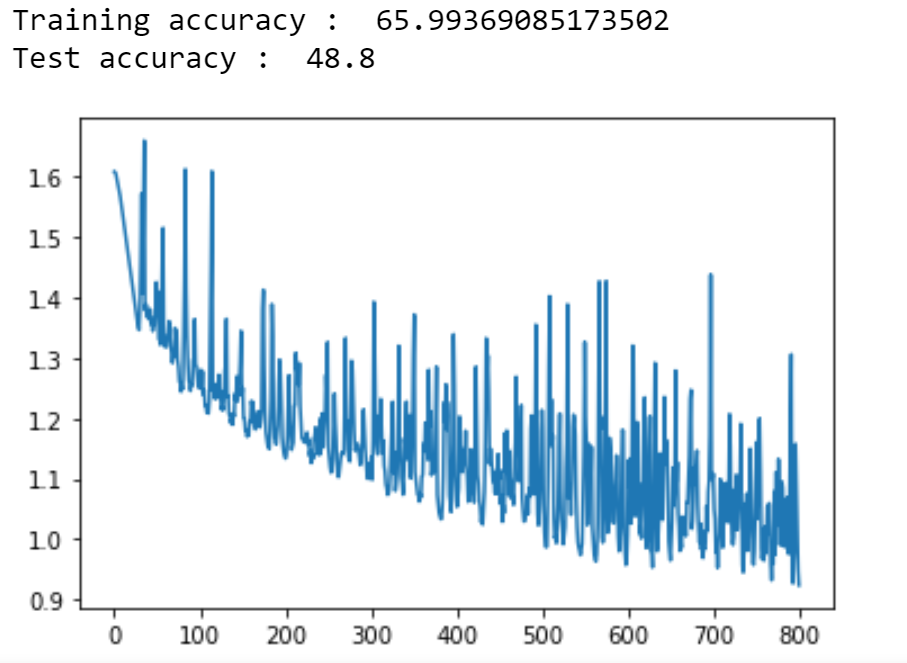
Evidence of overfitting.



2 hidden layers, 1000 neurons each. Learning rate = 0.02, epochs=400, alpha=0.6

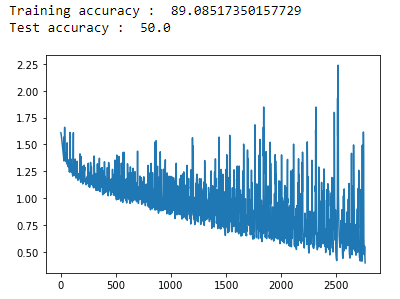


3 hidden layers, 700 neurons each. Learning rate = 0.02, epochs=400, alpha=0.6



3 hidden layers, 700 neurons each. Learning rate = 0.02, alpha=0.6

Epochs = while loop until error is <0.4

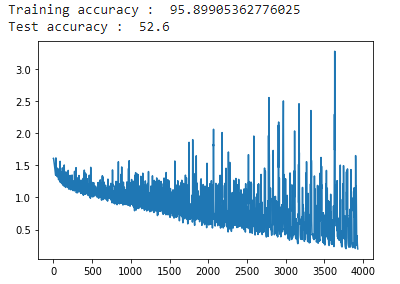


**Best run**

3 hidden layers, 700 neurons each. Learning rate = 0.02, alpha=0.6

Epochs = while loop until error is <0.2

Evidence of overfitting motivated a trial run below with less complexity.



Accuracy per class:

Daisy: 42%

Rose:44%

Dandelion: 71%

Sunflower:49%

Tulip: 59%

3 hidden layers of 100 neurons each. LR=0.07, Alpha=0.6

model.pkl downloaded to keep the Network object for prediction.

This is included in the folder, however it is not the best run in terms of test accuracy.

