Report Luca Sichi

Bag of words classifier:

grid_points():

The implementation is pretty straightforward

descriptors_hog():

Here I implemented 3 helper functions that assist in calculating the hog descriptors. Here I also had to change the datatype of the np arrays from 16 bit to 32 bit.

[10,32.5,55,77.5,100,122.5,145,167.5] these are my bins (in degree)

create_codebook():

here my biggest issue was dealing with nan, else it is straightforward.

bow_histogram():

straightforward.

create_bow_histograms():

straightforward.

bow_recognition_nearest():

here we decide based on the norm what label we give the image.

Hyperparameters:

Finally I had to choose some k and numiter. I ran multiple tests and got the following results:

k=60

iterations=100

test pos sample accuracy: 0.7346938775510204

test neg sample accuracy: 0.78

==========

k=50

iterations=100

test pos sample accuracy: 0.7551020408163265

test neg sample accuracy: 0.8200000000000001

==========

k=40

iterations=100

test pos sample accuracy: 0.8367346938775511

test neg sample accuracy: 0.76

==========

k=30

iterations=100

test pos sample accuracy: 0.8775510204081632

test neg sample accuracy: 0.6

==========

k=20

iterations=100

test pos sample accuracy: 0.8163265306122449

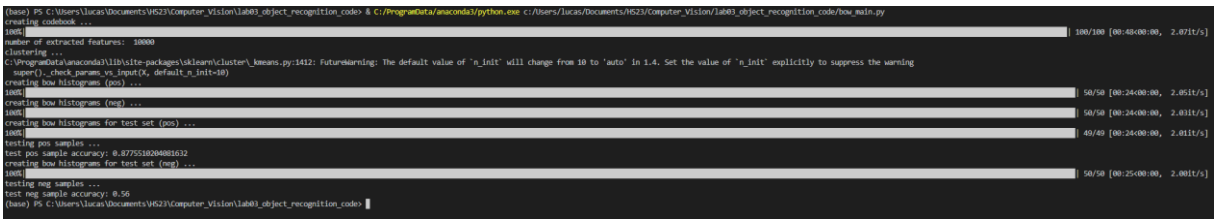
test neg sample accuracy: 0.64

==========

As we can see there is a tradeoff between the positive accuracy and the negative accuracy.

k=40 and numiter=100 seems to give decent results.

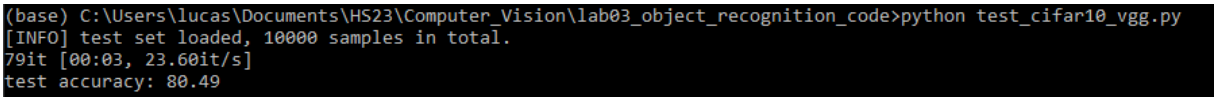
Log with k = 30 and numiter = 100

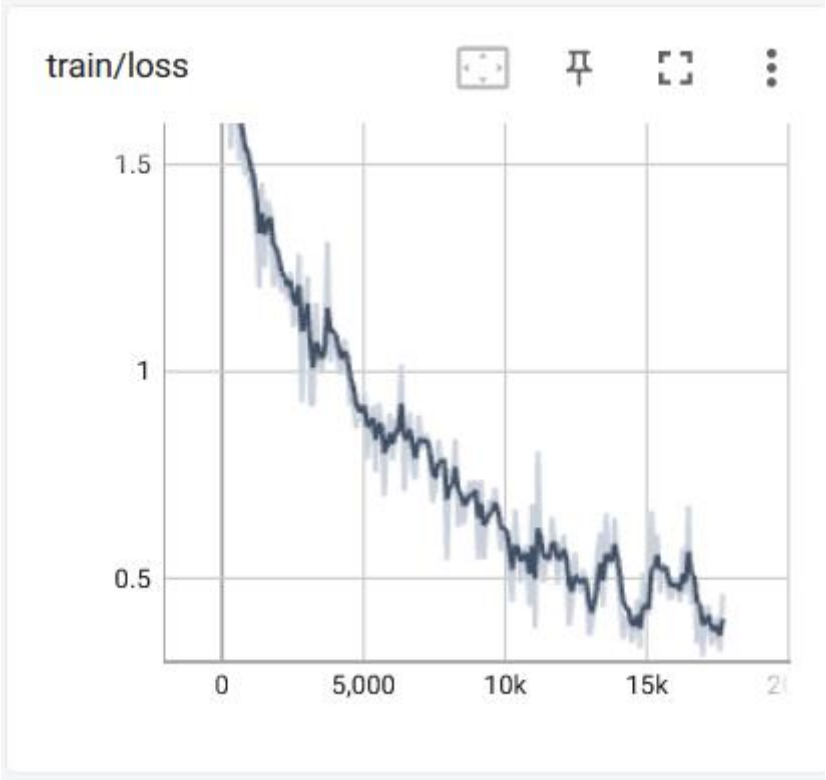


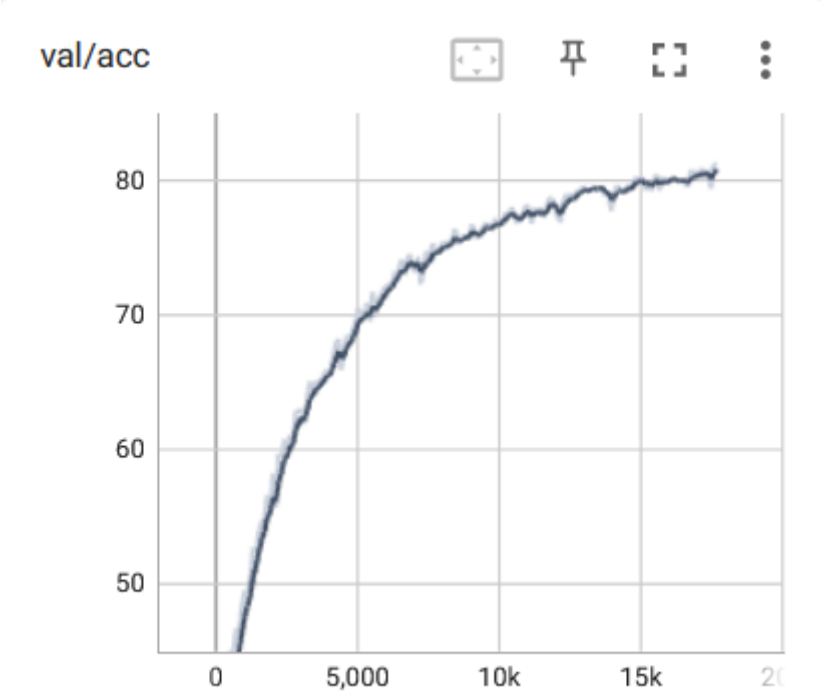
CNN Classifier

Here the main task was finding out the stride and padding. I used the formula on the documentation to calculate the corresponding values and ended up with padding=1 and stride=2 for a given kerne size of 3. Then I did the same for the max pool filters.

Test_cifar:




train/loss
1.5
1
0.5
0
5,000
10k
15k


val/acc
80
70
60
50
0
5,000
10k
15k