2/29/2024

Image Classifier Project

In order to create my feature vector I used a pre-trained neural network embedding. I augmented my data by horizontally flipping a fifth of the images, then I ran the augmented data as well as the original data through a pre-trained version of ResNet50 to extract feature vectors. I think I put all augmented data through ResNet, while putting only part of the original data though. This was mainly an accident, but it worked so I didn't touch it. The pre-processing hyperparameters were as required by pre-trained ResNet50:

Resize(256),

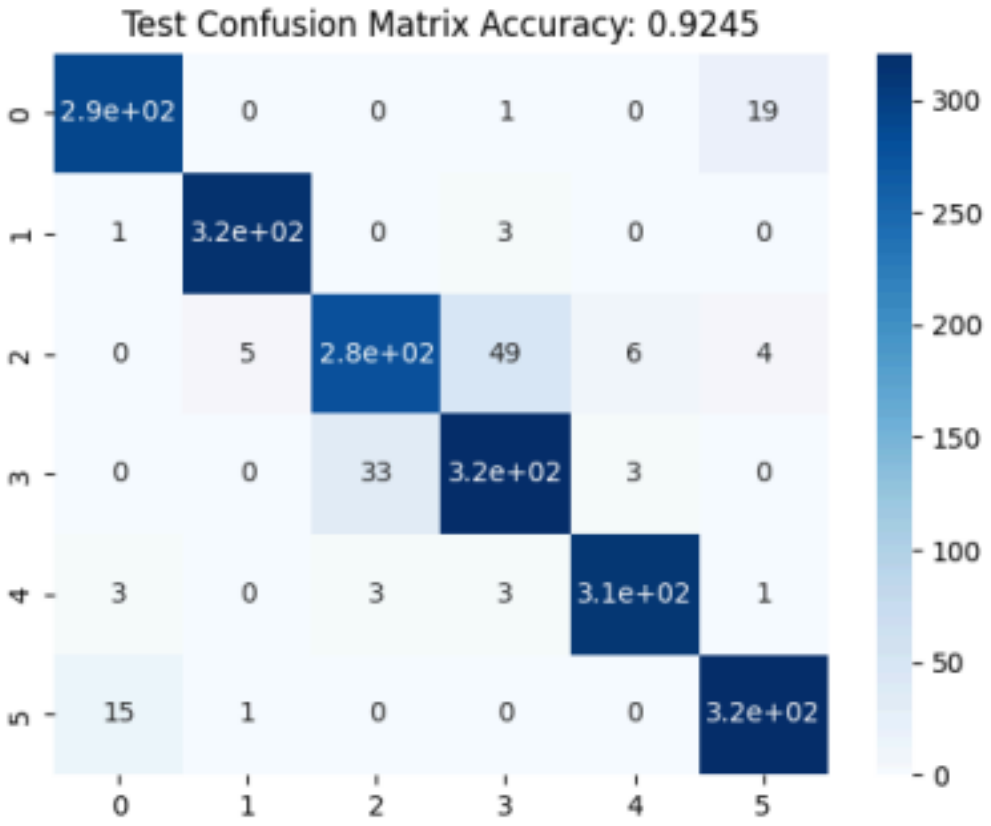
CenterCrop(224),

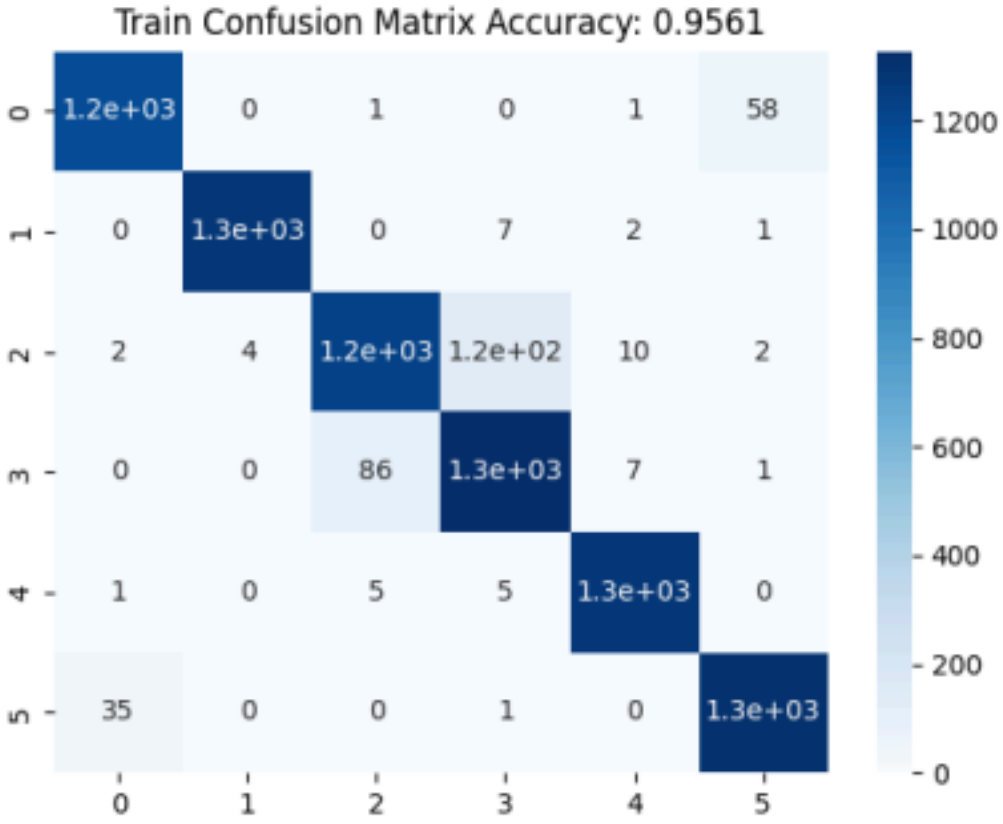
Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]).

Before the neural network, I was trying very hard to make HOG feature vectors work, but couldn't find a combination of classifier and HOG parameters with high accuracy. The data augmentation significantly increased the computational cost of obtaining feature vectors. The combined time my laptop took to run the augmentation and ResNet50 was 14.4 minutes. I considered the extra time worth it because it raised my accuracy significantly, while the many other things I tried did not.

For my classifier I used an SVC with a RBF kernel and a random state of 42. The time taken for fitting, training, and testing was around 1.25 minutes, with .195 minutes to fit and train. I needed a more economic classifier to make up for the computational cost of the augmentation. The program is around 15 minutes from beginning to end, although I think my laptop isn't efficient. The train accuracy is 95% and the test accuracy is 92.5%.

I originally was working with a Random Forest Classifier, and I used hyperparameter tuning while trying to make it viable. None of the results I got from hyperparameter tuning raised the test scores enough to warrant use.


Test Confusion Matrix Accuracy: 0.9245
0
1
2
3
4
5
2.9e+02
0
0
1
0
19
1
3.2e+02
0
3
0
0
0
5
2.8e+02
49
6
4
0
0
33
3.2e+02
3
0
3
0
3
3
3.1e+02
1
15
1
0
0
0
3.2e+02
0
1
2
3
4
5
300
250
200
150
100
50
0


Train Confusion Matrix Accuracy: 0.9561
0
1
2
3
4
5
1.2e+03
0
1
0
1
58
0
1.3e+03
0
7
2
1
2
4
1.2e+03
1.2e+02
10
2
0
0
86
1.3e+03
7
1
1
0
5
5
1.3e+03
0
35
0
0
1
0
1.3e+03
1200
1000
800
600
400
200
0