Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Computer Vision
and Geometry Lab

Computer Vision Lecture

Assignment 5 Report

Autumn Term 2021

Author:
Pascal Lieberherr

Contents

Chapter 1

Report

1.1 BoW Classifier: Explanation

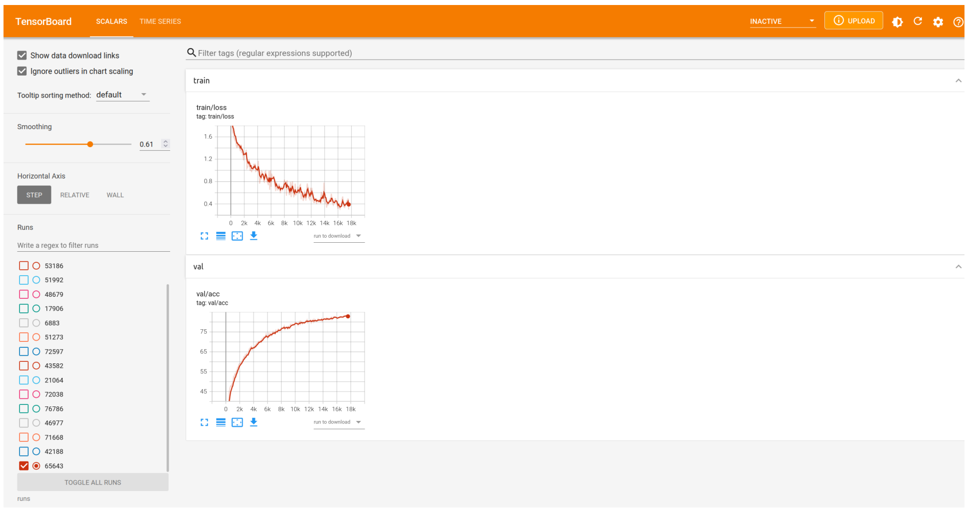
The first part of a BoW Classifier is to build a visual vocabulary for both labels i.e. images with cars and images without cars. Therefore, we have e.g. 100 train images for each class. For each image we want to extract 100 visual words. A visual word in our case is a 128 long vector. We simply take 100 points in the image which are equally spaced along x and y direction. Once we have these grid points, we compute a feature descriptor for each grid point. We use the Histogram of Oriented Gradients descriptor. Around each grid point a cell of 16×16 pixels is defined. This big cell is then divided into 4×4 pixel cells. For each 4×4 cell we compute the gradient direction and the gradient magnitude. In order to get the gradient direction we compute the x- and y-gradient and then make use of $numpy.arctan2(y_{gradient}, x_{gradient})$ which returns the direction. The magnitude is found through $sqrt(y_{gradient}^2 + x_{gradient}^2)$. Now, I compute an 8-bin histogram for each 4×4 cell for the oriented gradient weighted by the gradient magnitude. Once this is done for all 16 cells, I concatenate all 8-bin histograms to a 128 long vector. This procedure is then repeated for all grid points per image. In the end we have $nImages * nGridPoints$ of such descriptors i.e. 100*100. Next, we run k-means clustering in order to find k cluster centers out of the 10'000 visual words. Now, the k cluster centers form the visual vocabulary.

Once we built the visual vocabulary we are able to classify an image. We first represent the query image as a Bag of Words histogram whereas the different bins of the Bag of Words histogram are the k cluster centers. To do so each extracted HOG feature of the image is assigned to the closest cluster center of the visual vocabulary. In order to find the closest cluster center I make use of KD Tree from the library scipy to speed up finding the most similar vector.

Once we have the BoW histogram for the query image we find the closes BoW histogram of both the positive and negative visual vocabulary. We return the class for which the distance is smaller. Now, we can classify the image to either car or no car.

1.2 BoW Classifier: Results

With k=25 and numiter=550 I reach an accuracy for both negative and positive of approx. 0.95.



[✓] Show data download links
[✓] Ignore outliers in chart scaling
Tooltip sorting method: default

Smoothing 0.61

Horizontal Axis [✓ STEP] RELATIVE WALL

Runs
Write a regex to filter runs
53186
51992
48679
17906
6883
51273
72597
43582
21064
72038
76786
46977
71668
42188
[✓] 65643

Figure 1.1: The tensorboard for the VGG Lite training. The picture can also be found in the folder images.

1.3 Classfication with VGG: Explanation

This exercise was pretty straight forward. I implemented the architecture. Where I made use of the nn.Conv2d(), the nn.ReLU() and the nn.MaxPool2d() objects for the convolutional layers. And for the linear layer of nn.Linear() and nn.Dropout(). I applied dropout with p=0.2 and the first linear layer has 100 outputs. After I implemented the forward pass I could train the network.

1.4 Classfication with VGG: Results

I achieved an accuracy of 80.36 on the test set.