OpenCV Haar Cascade erstellen:

Systemanforderung:

* OpenCV muss vom Quellcode kompiliert werden (pip3 installopencv-python reicht nicht)
* Bisher hab ich nur auf Ubuntu getestet
* Desto besser der Prozessor ist, desto schneller geht das Training
* Min. 3-4GB Arbeitsspeicher

Notwendige Ordnerstruktur:

* Ordner „data“ zum Speichern des Cascade-Files
* Ordner „info“ zum Speichern der positiven Bilder
* Ordner „neg“ zum Speichern der negativen Bilder
* positive\_bild.jpg (Bild mit dem trainiert werden soll, auf max. 50x50px zurechtgeschnitten; idealerweise 50x50px falls quadratisch, andernfalls lange Seite max. 50px)

# CascadeClassifier erstellen

* 1. Bilderdatensatz auf image-net.org aussuchen
  2. Python-Script um Bilder herunterzuladen und im „neg“ Ordner zu speichern, auf 100x100px zu skalieren und in Graustufen umzuwandeln

import cv2

importnumpyas np

importurllib.request

importos

defstore\_raw\_images():

neg\_images\_link="http://www.imagenet.org/api/text/imagenet.synset.geturls?wnid=n00007846"

neg\_image\_urls=urllib.request.urlopen(neg\_images\_link).read().decode()

ifnotos.path.exists('neg'):

os.makedirs('neg')

pic\_num=980

foriinneg\_image\_urls.split('\n'):

try:

print(i)

urllib.request.urlretrieve(i,"neg/"+str(pic\_num)+".jpg")

img= cv2.imread("neg/"+str(pic\_num)+".jpg", cv2.IMREAD\_GRAYSCALE)

resized\_image= cv2.resize(img,(100,100))

cv2.imwrite("neg/"+str(pic\_num)+".jpg",resized\_image)

pic\_num+=1

exceptExceptionas e:

print(str(e))

* 1. Python-Script um bg.txt Datei zu erstellen, die für späteres Training notwendig ist

defcreate\_pos\_neg():

forfile\_typein['neg']:

forimginos.listdir(file\_type):

iffile\_type=='neg':

line =file\_type+"/"+img+"\n"

withopen("bg.text","a")as f:

f.write(line)

* 1. Danach muss der Cascade-Ordner im Terminal geöffnet werden

opencv\_createsamples-img palette\_front\_50\_47.jpg -bgbg.text-info info/info.lst-pngoutputinfo-maxxangle0.5-maxyangle0.5-maxzangle0.5-num1750

#-num muss kleiner als die Anzahl der negativen Bilder

* 1. Erstellung des Vektorfiles für die positiven Bilder

openv\_createsamples-info info/info.lst-num1750-w 48-h 48-vecpositives.vec

#-num muss gleich groß sein wie im vorherigen Schritt

#-w und -h gibt Größe des zu trainierenden Classifiers an, desto größer #desto länger dauert das Training und desto mehr Rechenleistung ist nötig

#48x48 bereits ziemlich rechenintensiv

* 1. Training

opencv\_traincascade-data data-vecpositives.vec-bgbg.text-numPos1600-numNeg800-numStages15-w 48-h 48

#-numPos darf nicht größer sein als in den vorherigen Schritten festgelegt

#Verhältnis von numPos zu numNeg sollte in der Regel 2:1 sein, deswegen -#numNeg 800

#-w und -h muss genauso groß sein wie im vorherigen Schritt festgelegt

#-numStages gibt die Anzahl der zu trainierenden Stages an, desto mehr #Stages, desto besser wird die Cascade (kann aber auch übertrainiert #werden). Jedoch dauert das Training auch um einiges länger

* 1. Fertige Cascade-Datei

Im Ordner data sind nun die einzelnen .xml-Dateien der Stages abgespeichert.

„cascade.xml“ ist der fertige Cascade-Classifier.

# Classifier verwenden

importnumpyasnp

import cv2

defnothing():

pass

cascade = cv2.CascadeClassifier('cascade.xml')

cap = cv2.VideoCapture(0)

#Trackbars fürdetectMultiScale parameter (scale\_factor und min\_neighbours)

cv2.namedWindow("Trackbars")

cv2.createTrackbar("scale","Trackbars",11,20, nothing)

cv2.createTrackbar("min","Trackbars",3,6, nothing)

cv2.createTrackbar("Size","Trackbars",100,200, nothing)

while1:

ret,img=cap.read()

resize\_val= cv2.getTrackbarPos("Size","Trackbars")

img= cv2.resize(img,(int(img.shape[1]\*resize\_val/100),int(img.shape[0]\*resize\_val/100)))

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

objects=cascade.detectMultiScale(gray,cv2.getTrackbarPos("scale","Trackbars")/10, cv2.getTrackbarPos("min","Trackbars"))

for(x, y, w, h)inobjects:

print("Object detected")

print("X: {:d}, Y: {:d}, W: {:d}, H: {:d}".format(x, y, w, h))

cv2.rectangle(img,(x, y),(x + w, y + h),(255,255,0),2)

break

cv2.imshow('img',img)

k = cv2.waitKey(30)&0xff

if k ==27:

break

cap.release()

cv2.destroyAllWindows()