1. YOLOv3 mit vortrainiertem Netz ausführen

YOLOv3 kann entweder direkt über die Kommando Zeile ausgeführt werden (benötigt Installation von Darknet) oder in Python mithilfe des Moduls opencv-python.

Für beide Möglichkeiten werden folgende Input Files benötigt (Beispiel mit COCO Datenset):

Vortrainierte Gewichts Datei

<https://pjreddie.com/media/files/yolov3.weights>

Config File:

cfg\ yolov3.cfg oder <https://github.com/pjreddie/darknet/blob/master/cfg/yolov3.cfg>

(öffnen und Parameter batch und subdivision auf 1 setzten)

Data File:

\data\coco.data oder <https://github.com/pjreddie/darknet/blob/master/cfg/coco.data>

Names File (enthält Liste aller eintrainierten Klassen):

\data\coco.names oder <https://github.com/pjreddie/darknet/blob/master/data/coco.names>

Test Bilder/Videos, mit Objekten die in coco.names enthalten sind darauf

Bsp Bilder in data\... oder <https://github.com/pjreddie/darknet/tree/master/data> oder eigene

1. Ausführung über command line

Benötigt Darknet Installation, siehe install.docx

Aus dem Darknet Installationsverzeichnis heraus, mit pfadangaben der cfg, data, weights Dateien relativ zur Darknet.exe

1. Für Bilder:

darknet.exe detector test path/to/coco.data path/to/yolov3.cfg path/to/yolov3.weights path/to/test\_img.jpg

(in Linux mit ./Darknet anstatt darknet.exe)

1. Für Videos:

darknet.exe detector demo path/to/coco.data path/to/yolov3.cfg path/to/yolov3.weights path/to/test\_video.mp4

1. Für Webcam:

darknet.exe detector demo path/to/coco.data path/to/yolov3.cfg path/to/yolov3.weights -c <num>

(mit num für Webcam Index, falls mehrere angeschlossen, default 0)

1. Ausführung in Python

Mit Beispiel Script von <https://www.pyimagesearch.com/2018/11/12/yolo-object-detection-with-opencv/>

Für Bilder mit vorhandenem Beispiel Script yolo.py

1. Module cv2 installieren: <https://pypi.org/project/opencv-python/>
2. Ordner mit zu erkennenden Bildern und Ordner mit zu verwendenden names, cfg und weight Dateien erstellen.
3. Script ausführen: python yolo.py --image images/img\_name.jpg --yolo yolo-coco

Für Videos mit vorhandenem Beispiel Script yolo\_video.py

1. Zusätzlich <https://pypi.org/project/imutils/> installieren
2. Input/Output Ordner erstellen
3. Script ausführen über: python yolo\_video.py --input videos/airport.mp4 --output output/airport\_output.avi --yolo yolo-coco
4. Netzt für eigene Objekte trainieren

(Darknet Installation erforderlich, siehe install.docx)

1. Benötigte Input Files:

Zusätzlich zu Files aus I.:

Datenset, bestehend aus Bildern mit zu trainierenden Objekten und Label Text Files.

Text Files train.txt und test.txt, zur Aufteilung des Datenset in ein Trainings Set und ein Validierungs Set.

Und als Gewichtsdatei zum Starten: <https://pjreddie.com/media/files/darknet53.conv.74>

1. Vorbereitung
   1. Anpassen des Config File

Im Yolo.cfg File folgende Parameter anpassen

batch=64

gibt die Größe des Subset an, dass pro durchlauf verwendet wird, je größer, desto mehr Gpu-Speicher wird benötigt.

subdivision=8

Teilt den Batch nochmal auf in: Batch / Subdivision

wenn Gpu Speicher Fehler auftritt kann dies durch erhöhen des Subdivision Parameters verhindert werden, dabei immer verdoppeln (8, 16, 32, …)

max\_batches = classes \* 2000 (zB max\_batches =6000 für 3 klassen)

(entspricht der maximalen Anzahl an Durchläufen)

steps=80% von max\_batches, 90% von max\_batches (zB steps=4800,5400)

in allen [yolo] – Layern

classes = Anzahl der Klassen die trainiert werden sollen (zB classes=3)

* 1. Anpassen/Erstellen des Names File

Das Names File ist eine Textdatei die Namen der zu trainierenden Klassen enthält. Jeweils in einer eigenen Zeile.

Mit der Endung .names abspeichern.

* 1. Anpassung/Erstellung des Data File

Das data File enthält folgenden Inhalt:

classes=anzahl der klassen (selbe wie in cfg file)

train = path/to/train.txt

valid = path/to/test.txt

names = path/to/ .names

backup = backup/

und wird mit der Endung .data abgespeichert.

* 1. Erstellung des Datenset

Das Datenset besteht aus einem Ordner mit Bildern, auf denen Objekte der zu trainierenden Klassen dargestellt sind. Zu jeder Bild Datei wird ein gleichnamiges Text Datei benötigt, welche die Labels der Objekte auf dem Bild enthält. Z.B. für img1.jpg das zugehörige img1.txt File welches je zu labelndem Objekt auf dem Bild eine Zeile enthält, die etwa so aussieht:

0 0.687109 0.379167 0.255469 0.158333

Die erste Zahl gibt die Klasse an und entspricht dem Index der Klasse im Names File

Die vier folgenden Zahlen entsprechen <x\_center> <y\_center> <width> <height> der Bounding Box um das Objekt herum.

Zum (automatisierten) erstellen des Datensets siehe: **createDataset.docx**

* 1. train.txt und test.txt

Die Dateien train.txt und test.txt enthalten Pfade zu allen Bildern des Datensets.

Dabei je Bild eine Zeile mit: path/to/img/img1.jpg

Die gesamt anzahl der Bilder des Datensets wird dabei auf die beiden Dateien train und test so aufgeteilt, dass 90% der Bilder in train.txt angegeben werden und 10% der Bilder in test.txt.

Dafür kann das Python Script splitTrainAndTest.py verwendet werden. Dieses benötigt beim Aufruf den Pfad zu den Bildern als Argument.

python splitTrainAndTest.py img\_dir

1. Training

Um das Training zu starten folgenden Command verwenden:

darknet.exe detector train path/to/my.data path/to/my.cfg darknet53.conv.74

(in Linux ./Darknet statt Darknet.exe)

mit dem flag -dont\_show am Ende des Command kann das Loss Fenster deaktiviert werden.

Alle 100 Durchläufe wird im Backup Ordner die aktuelle Gewichts Datei gespeichert unter

backup/yolo-obj\_last.weights

Während des Trainings erscheinen Zeilen, die etwa so aussehen:

1: 11.367310, 11.367310 avg, 0.000100 rate, 20.268080 seconds, 1 images

Die erste Zahl ist die Durchlaufs Nummer

Die zweite zahl gibt den aktuellen Loss (Fehler) an und die dritte den durchschnitts Loss.

Je kleiner der Loss desto besser ist das Netz trainiert, wenn keine Verringerung mehr stattfindet, kann mit dem Training aufgehört werden. Üblicherweise werden pro Klasse 2000 Durchläufe benötigt, je nach Varianz des Datensets können aber auch schon früher brauchbare Ergebnisse entstehen.

Falls das Training unterbrochen wurde, kann mit den zuletzt in backup gespeicherten Gewichten weiter trainiert werden. Dafür wird der normale Command zum starten des Trainings verwendet nur anstelle der Start Gewichte darknet53.conv.74 den Pfad zur letzten yolo-obj\_last.weights angeben um mit diesen weiter zu trainieren.