

Darknet windows 10 64bit asennus ja käyttöönotto

Darknetin avulla voi tehdä kuvan tunnistusta mutta sen avulla voidaan myös tehdä "weights" tiedostoja joita käytetään kuvan tunnistuksessa mallina, eli voit opettaa mitä tunnistetaan kuvista ja videolta weights tiedoston avulla.

Linkki projektiin: <https://github.com/AlexeyAB/darknet>

Github sivulla on kerrottu vaadittavat ohjelmat jotka tulevat olla asennettuna jotta se toimii. Mutta Tässä on dokumentissa on tiivistettynä kaikki miten asentaa kaikki tarvittava.

Darknetin ja ohjelmien asennus

(ohjelmien nimen vierässä olevissa suluissa on versiot jotka olen itse asentanut)

Suositeltavaa että ohjelmat asennetaan siinä järjestyksessä missä ne ovat tässä listassa(esim. cuda vaatii visual studion).

Python3.7

Visual Studio(2019 community)

Johon on asennettu "desktop development with c++". Ja jos koneellasi on jo visual studio niin muokkaa asennusta jos "desktop development with c++" ei ole asennettuna.

<https://visualstudio.microsoft.com/downloads/>

Nvidia Cuda(10.1 update2) ja Cudnn(v7.6.4 for CUDA 10.1)

Cuda ja cudnn ovat vahvasti suositeltu , sillä ne nopeuttavat kuvan tunnistusta ja weights tiedoston opettamista huomattavasti, jos tietokoneessa on vaatimukset täyttävä nvidia näytönohjain.

Lista yhteensopivista näytönohjaimista: <https://developer.nvidia.com/cuda-gpus>
(ohjaimet joissa on compute capability 3.0 tai suurempi)

Cuda lataus linkki: <https://developer.nvidia.com/cuda-downloads>

Valitse Käyttöjärjestelmän kohdalla windows, sen jälkeen valitse mikä windows versio koneellasi on, seuraavaksi valitse "installer type" kohdalla "exe(local)" ja lopuksi valitse download. Aja exe tiedosto kun se on ladannut ja asenna cuda.

Cudnn lataus linkki: <https://developer.nvidia.com/cudnn>

Jotta voit ladata cudnn tiedoston tarvitset nvidia tilin jos sinulla on se jo valmiiksi tehtynä kirjaudu sisään ja lataa cudnn.

Ohjeet cudnn asennukseen: <https://docs.nvidia.com/deeplearning/sdk/cudnn-install/index.html#install-windows>

Cmake(3.17.2)

Opencv ja darknet buildaamista varten.

<https://cmake.org/download/>

Opencv(4.3.0)

Opencv asennus ohjeet videolla:

Tämä video ennen Cmake generate vaihetta!!

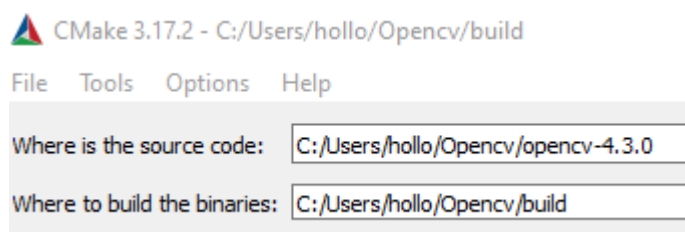
<https://youtu.be/saDipJR14Lc?t=333> linkissä näytetään mistä kohtaa ruksittaa WITH_CUDA ja WITH_CUDNN ja mistä kohtaa voi muuttaa mitkä versiot asennetaan.

Opencv asennus video: <https://www.youtube.com/watch?v=fqpYLM6SCw>

Opencv source tiedoston lataus linkki: <https://opencv.org/releases/>

Valitse "sources" uusimmasta versiosta ja lataa se. Kun opencv sources tiedosto on ladannut pura se valitsemaasi kansioon ja käynnistä cmake.

Cmakessa "where is the source code:" kohtaan etsi tiedosto johon purit opencv sources tiedoston ja "where to build the binaries:" johon opencv asennetaan.



Kun kansiot ovat valittu paina configure valitse valikosta finish. Jos configuroinnin jälkeen on vielä punaisia laatikoita configuroi uudestaan. Seuraavaksi ruksita "Advanced" laatikko haku palkin vierestä ja etsi "WITH_CUDA" ja "WITH_CUDNN" ja varmista että ne ovat ruksitettuna.



Lataa myös https://github.com/opencv/opencv_contrib/tree/4.3.0 tiedosto ja varmista että se vastaa opencv versiotasi. Kun opencv contrib tiedosto on ladattu pura se valitsemaasi kansioon ja CMake ohjelman kohdassa "OPENCV_EXTRA_MODULES_PATH" aseta sen kohdalle polku opencv_contrib-4.3.0/modules tiedostoon.



Ja seuraavaksi tarkista <https://developer.nvidia.com/cuda-gpus> linkistä näytönohjaimesi "compute capability" ja mene Cmake:lla kohtaan CUDA_ARCH_BIN ja poista muut versiot paitsi näytönohjaimesi "compute capability" versiot ,eli jos versiosi on 7.5 poista kaikki muut paitsi 7.0 ja 7.5

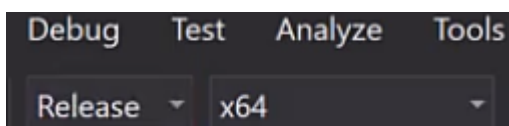


Kun nämä säädöt on tehty ja CMake:ssa ei enään näy punaista niin paina configure ja sen jälkeen generate.

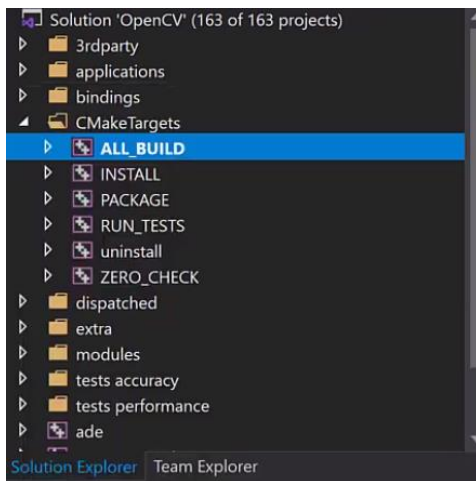
Tämä luo kansioon tarvittavat tiedostot jolla saat buildattua opencv:n.

Eli mene kansioon jonka asetit kohtaan "where to build the binaries:" etsi sieltä tiedosto OpenCV.sln tiedosto ja tuplaklikkaa sitä, tämä käynnistää visual studion.

Aluksi vaihda visual studion keskeltä ylhäältä debug pois ,aseta release tilalle ja varmista että toinen asetus on x64.



Tämän jälkeen avaa cmaketargets ja ALL_BUILD kohdassa paina hiiren oikeaa ja valitse build. Tässä saattaa kestää kauan.

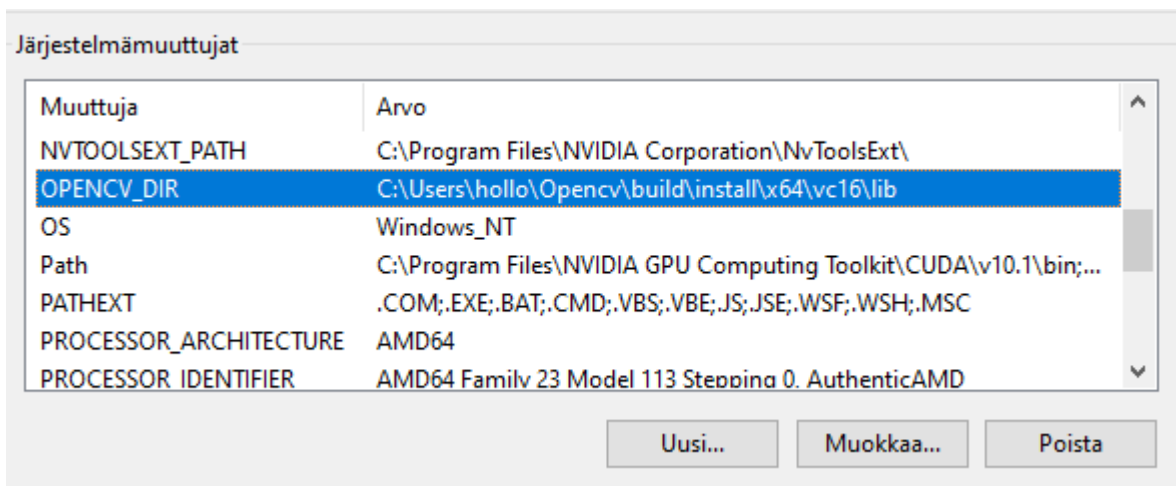


Kun ALL_BUILD on valmis niin tarkista että se on onnistunut, eli pitäisi olla 0 failed. Mutta jos jokin epäonnistui niin kokeile vielä kerran uudestaan ALL_BUILD kohdalla hiiren oikeaa ja build.

```
===== Build: 155 succeeded, 0 failed, 0 up-to-date, 0 skipped =====
```

Ja kun ALL_BUILD on onnistunut lopuksi hiiren oikealla paina INSTALL kohdalta ja valitse build. Tämän jälkeen opencv pitäisi olla asennettuna.

Lisää järjestelmämuuttuja opencv:lle jotta ohjelmat löytävät sen.



Darknet

Darknet github: <https://github.com/AlexeyAB/darknet>

Asennus video: <https://www.youtube.com/watch?v=saDipJR14Lc>

Lataa darknet projekti githubista pura se valitsemaasi kansioon.

Avaa cmake ja aseta polut kuten alla olevassa kuvassa.

(darknetin ei tarvitse olla repos kansiossa, riittää kunhan "where is the source code:" osoittaa purettuun darknet tiedostoon ja "where to build the binaries:" osoittaa puretun darknet tiedoston sisällä olevaan build tiedostoon.)

Where is the source code:	C:/Repos/darknet
Where to build the binaries:	C:/Repos/darknet/build

Kun polut on asetettu paina configure ja valitse finish.

Kun CMake on configuroinut tarkista että CMAKE_CUDA_COMPILER kohdassa

On polku nvcc.exe tiedostoon jos ei ole niin aseta se manuaalisesti, jos cuda ja cudnn ovat asennettuna.

CMAKE_CUDA_COMPILER	C:/Program Files/NVIDIA GPU Computing Toolkit/CUDA/v10.1/bin/nvcc.exe
---------------------	---

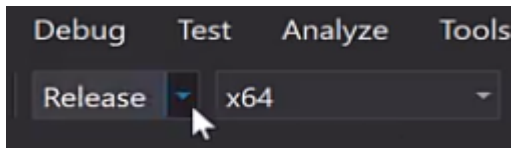
Ja katso myös että ENABLE_CUDNN ja ENABLE_CUDA on ruksitettuna.

CUDNN_LIBRARY	C:/P
CUDNN_LIBRARY_DLL	C:/P
ENABLE_CUDA	<input checked="" type="checkbox"/>
ENABLE_CUDNN	<input checked="" type="checkbox"/>
ENABLE_CUDNN_HALF	<input type="checkbox"/>
ENABLE_OPENCV	<input checked="" type="checkbox"/>
ENABLE_VCPKG_INTEGRATION	<input checked="" type="checkbox"/>
ENABLE_ZED_CAMERA	<input type="checkbox"/>

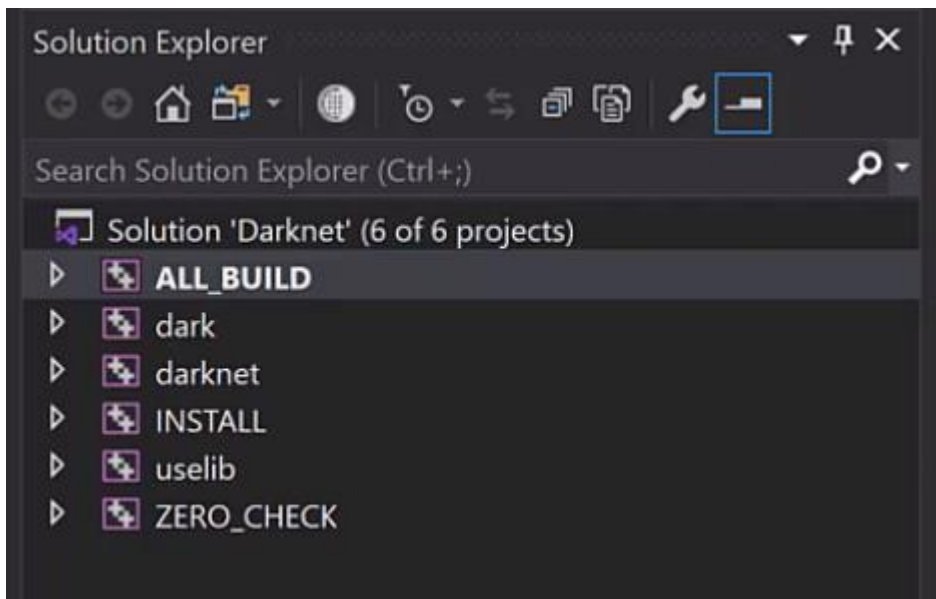
Tämän jälkeen paina configure nappia uudestaan ja jos punaista jää vielä niin configuroi uudestaan. Ja kun punaista ei enään ole paina generate.

Mene darknet/build kansioon ja etsi sieltä Darknet.sln tiedosto ja tuplaklikkaa sitä, tämä avaa visual studion.

Visual studiossa debug pois ,aseta release tilalle ja katso että viereinen asetus on x64.



Sen jälkeen valitse ALL_BUILD hiiren oikealla ja valitse build.



Kun build on onnistunut valitse INSTALL hiiren oikealla ja valitse build.

Testaa että asennus onnistui lataamalla <https://pjreddie.com/darknet/yolo/> sivulta weights tiedosto.

Easy!

You already have the config file for YOLO in the `cfg/` subdirectory. You will have to download the pre-trained weight file [here \(237 MB\)](#). Or just run this:

Ja kun olet ladannut tiedoston laita se darknet kansioon.

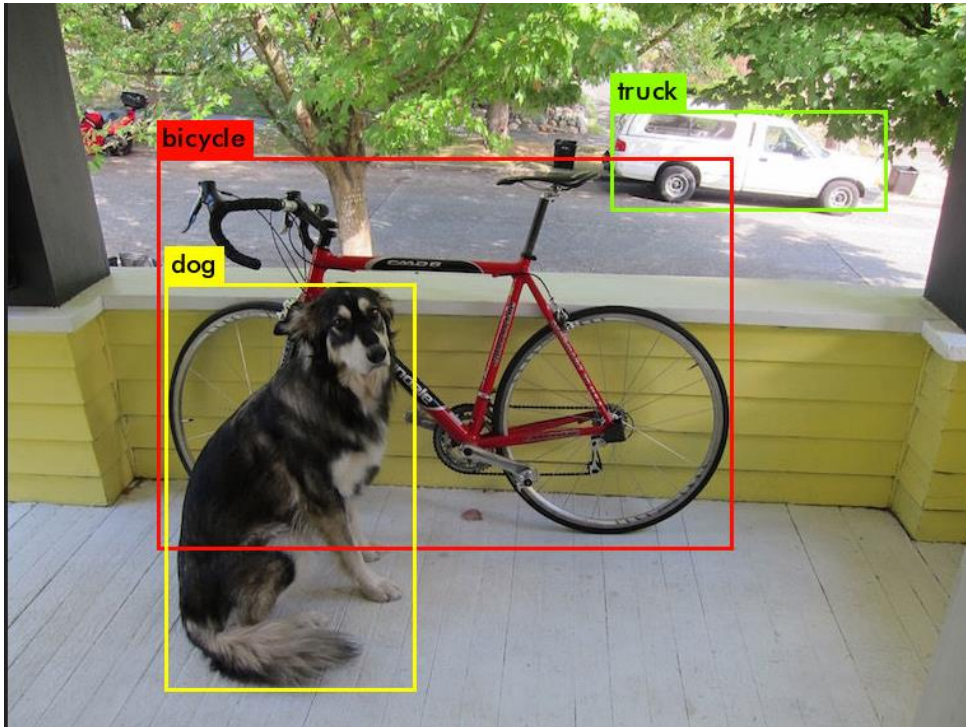
Tämän jälkeen voit testata että asennus on onnistunut avaamalla darknet kansion windows komentokehotteessa.



Kun komentokehote on auki ja olet darknet kansiossa aja komento

```
darknet.exe detect cfg/yolov3.cfg yolov3.weights data/dog.jpg
```

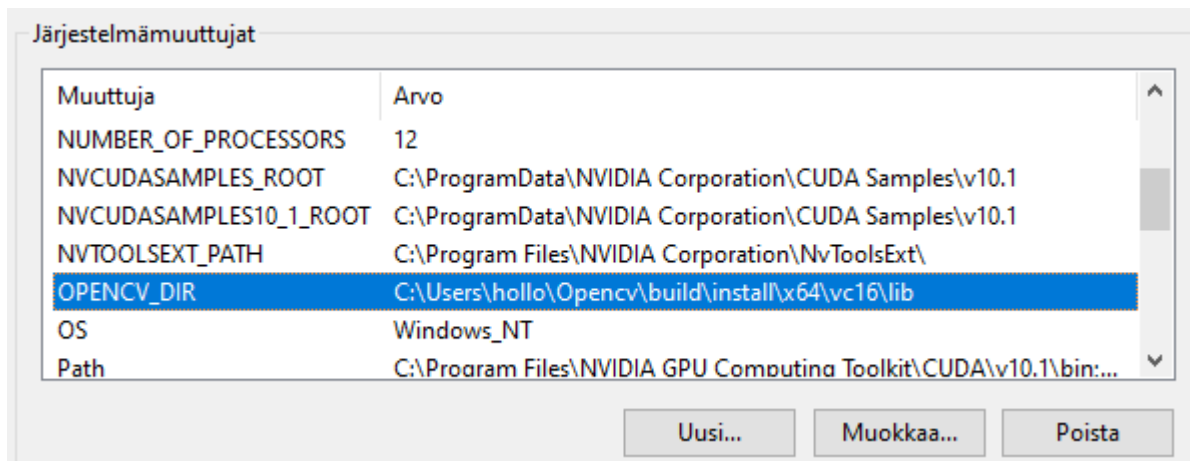
Ja jos kuva aukeaa ja siinä on tunnistettuna koira,auto ja pyörä niin asennus on onnistunut



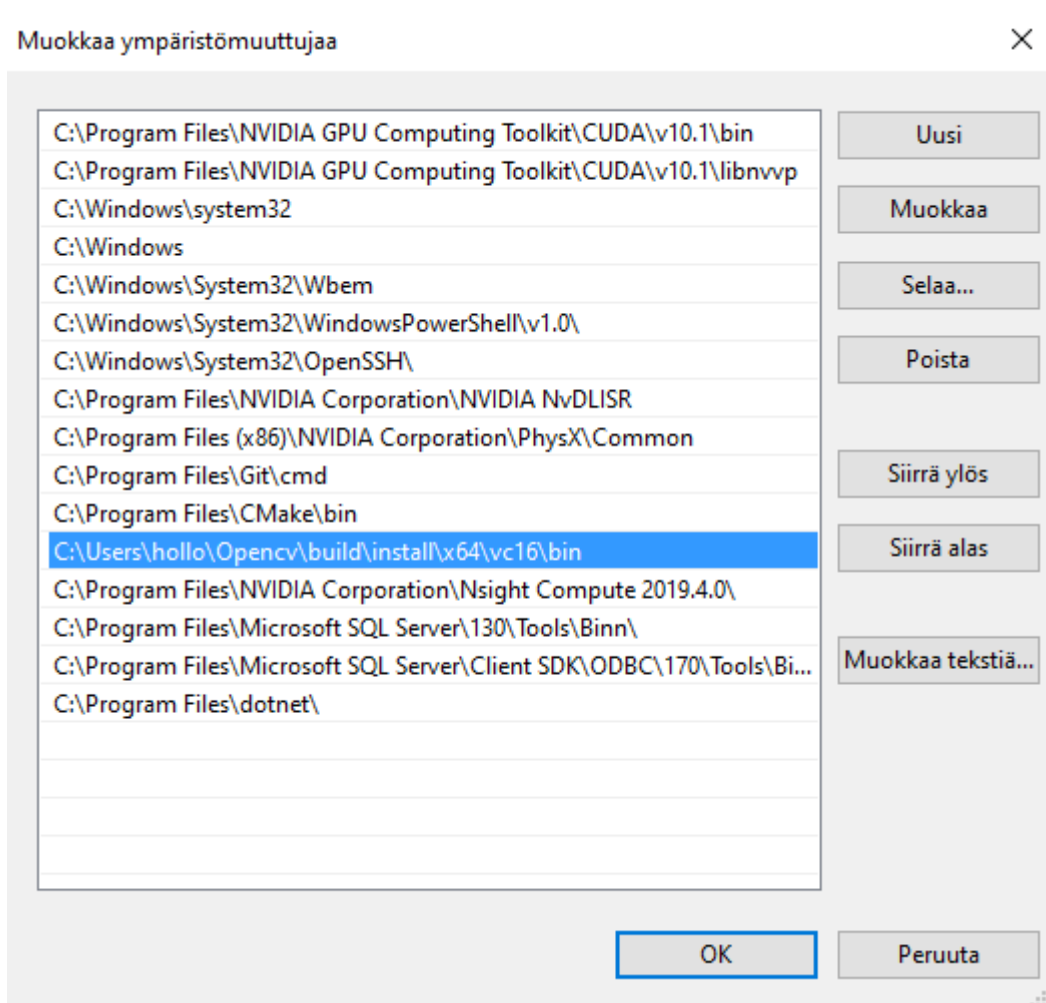
Mutta jos tulee varoituksia että puuttuu tiedostoja kuten pthreadVC2.dll alhaalla olevassa kuvassa 1 on ratkaisuja. Ja varoituksessa on esimerkiksi viesti "darknet missing opencv_highgui430.dll" niin muista lisätä ympäristönmuuttuja jossa on opencv kirjaston sijainnit (kuva 2) ja (kuva 3).

1. Copy `darklib.dll` and `uselib.exe` from `darknet/Release` folder to the `darknet/` root folder.
2. Also copy `pthreadGC2.dll` and `pthreadVC2.dll` from `darknet\3rdparty\pthreads\bin\` folder to the `darknet/` root folder.
3. Download <https://pjreddie.com/media/files/yolov3.weights> to the `darknet/` root folder.
4. Then run `uselib.exe data/coco.names cfg/yolov3.cfg yolov3.weights zed_camera` in the `darknet/` root folder.

Kuva 1:



Kuva 2: Lisää järjestelmä muuttuja OPENCV_DIR ja arvoksi lib tiedoston polku.



Kuva 3: Lisää Path järjestelmä muuttujaan tämä sijainti.

Oman weights tiedoston luominen

(eli oman kuvan tunnistuksen luonti)

Oman weights tiedoston luomiseen tarvitaan kuvia jotta weights tiedosto voidaan opettaa. Tähän on muutamia vaihtoehtoja kuten Google api (<https://storage.googleapis.com/openimages/web/index.html>) josta voi hakea kuvia joihin on valmiiksi merkattu esimerkiksi autoja ja ihmisiä.

Kuvien lataaminen

Linkki ohjelmaan jolla voi ladata google apista kuvat:

https://github.com/theAIGuysCode/OIDv4_ToolKit

Harjoitus kuvina voi myös käyttää omia kuvia mutta jos kuvissa ei ole merkattu esineitä ja asioita joita halutaan tunnistaa niin voi käyttää ohjelmaa

https://github.com/AlexeyAB/Yolo_mark jolla voi itse merkitä jokaiseen kuvaan esineet ja asiat.

Video ohjeistus kuinka ladata kuvat:

https://www.youtube.com/watch?v=_4A9inxGqRM

Lataa github linkistä projekti OIDv4 toolkit kun olet ladannut tiedoston pura se kansioon ja avaa kansio esimerkiksi PyCharm ohjelmalla tai windows komentokehotteella ja aja komento **"pip install -r requirements.txt"** .Tämä asentaa tarvittavat python kirjastot mitä projekti tarvii toimiakseen.

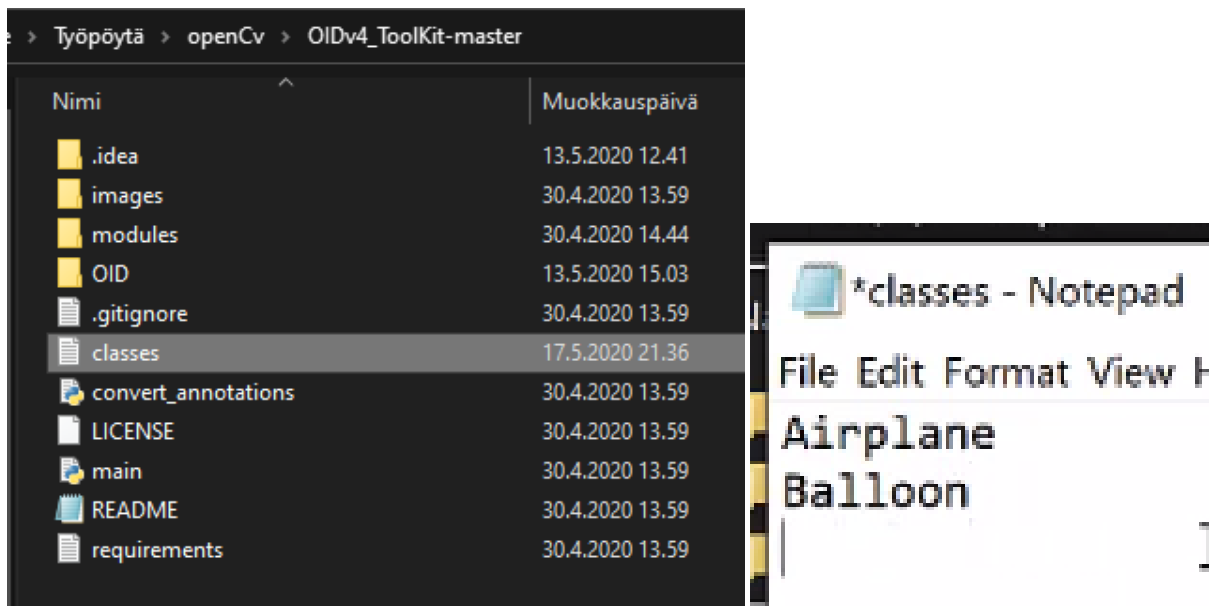
Kun tarvittavat kirjastot on asennettu voit ladata kuvia komennolla

```
python main.py downloader --classes Balloon Airplane --type_csv train --limit 400 --multiclass 1
```

--classes kohdalle mistä asioista haluat kuvia ja --limit kohdalle kuinka monta kuvaa haluat ladata per luokka. --Multiclass 1 komento lataa kaikki kuvat samaan kansioon.

Kuvat latautuvat OID\Dataset\train\”ladattavien luokkien nimet” kansioon.

Lisää classes tekstitiedostoon luokat allekkain yksi luokka per rivi.



Lopuksi aja python scripti `python convert_annotations.py`, tämä komento luo tekstitiedoston jossa on kuvasta havaittavien esineiden ja asioiden koordinaatit. Tekstitiedostot ilmestyvät samaan kansioon jossa kuvat ovat.

Kuvan tunnistuksen opettaminen

Darknet käyttöohje video kuinka luoda oma weights tiedosto:

<https://www.youtube.com/watch?v=zJDUhGL26iU&t=>

1.vie kuvat darknet kansioon

Aluksi kopioi kaikki kuvat ja tekstitiedostot jotka edellisessä vaiheessa ladattiin ja luotiin. Siirrä tiedostot kansioon `\darknet\data\obj`

2. Luo cfg tiedosto

Avaa projekti esimerkiksi visual studiossa tai pycharmissa. Avaa darknet\cfg kansio ja etsi sieltä tiedosto yolov3.cfg ja tallenna uusi kopio tästä tiedostosta(nimellä ei ole väliä se voi olla vaikka yolov3-custom.cfg)

Avaa kopioitu tiedosto. Kommentoi rivi 3,4 ja poista kommentit riviltä 6 ja 7.

Jos kuvantunnistusta kouluttaessa tulee cuda error out of memory on suositeltavaa nostaa subdivisions=32 jos virheilmoitus tulee edelleen niin kokeile subdivisions=64.

```
1 [net]
2 # Testing
3 #batch=1
4 #subdivisions=1
5 # Training
6 batch=64
7 subdivisions=16
```

Seuraavaksi aseta max batches 2000*luokkien määrä mutta ei alle 4000

Ja aseta steps arvot 80% ja 90% max batches luvusta.

```
18 learning_rate=0.001
19 burn_in=1000
20 max_batches = 500200
21 policy=steps
22 steps=400000,450000
23 scales=.1,.1
```

```
max_batches = 4000
policy=steps
steps=3200,3600
```

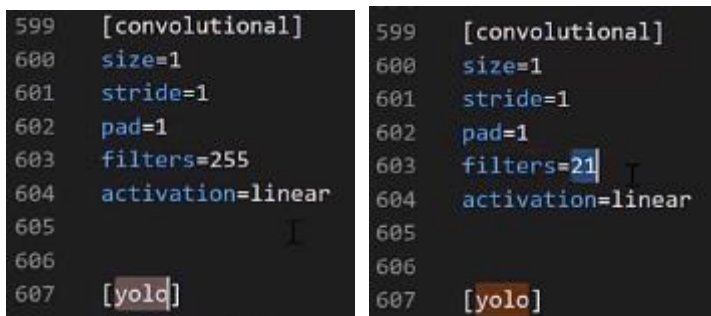
Seuraavaksi CTRL + F ja etsi sanalla "yolo" näitä löytyy 3 kappaletta.

Aseta classes kohdalle luokkiesi määrä ja aseta random=0.

```
607 [yolo]
608 mask = 6,7,8
609 anchors = 10,13, 16,3
610 classes=80
611 num=9
612 jitter=.3
613 ignore_thresh = .7
614 truth_thresh = 1
615 random=1
```

```
607 [yolo]
608 mask = 6,7,8
609 anchors = 10,13, 16
610 classes=2
611 num=9
612 jitter=.3
613 ignore_thresh = .7
614 truth_thresh = 1
615 random=0
```

Ja jokaisen yolo kohdan yläpuolella on [convolutional] osio johon tulee muokata filters osio. Sen tulisi aina olla $(\text{luokkien määrä} + 5) * 3$ eli jos luokkia on 2 niin arvo olisi 21 jos luokkia olisi 4 niin arvo olisi 27.



```
599 [convolutional]
600 size=1
601 stride=1
602 pad=1
603 filters=255
604 activation=linear
605
606 [yolo]

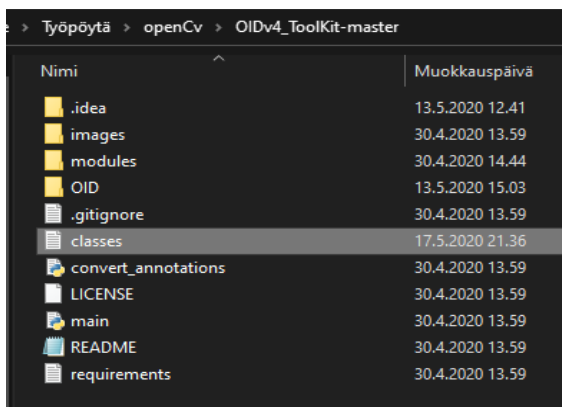
599 [convolutional]
600 size=1
601 stride=1
602 pad=1
603 filters=21
604 activation=linear
605
606 [yolo]
```

cfg tiedoston muokkaaminen on valmis.

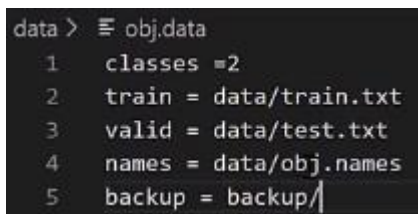
3.Kuvien sijaintien luonti

Luo darknet\data kansioon tiedosto obj.names

Tämän tiedoston tulisi olla sama kuin classes tiedosto. Eli tiedostossa on ladattujen kuvien luokkien nimet allekkain.

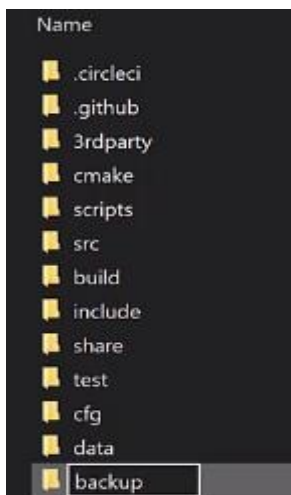


Luo darknet\data kansioon tiedosto nimeltä obj.data ja kirjoita tiedostoon teksti alla olevan kuvan mukaisesti mutta laita classes kohtaan kuinka monta eri kuva luokkaa sinulla on.

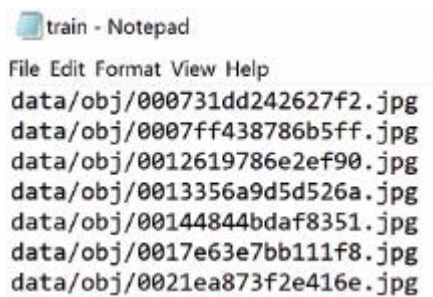


```
data > obj.data
1 classes = 2
2 train = data/train.txt
3 valid = data/test.txt
4 names = data/obj.names
5 backup = backup/
```

Luo darknet kansion juureen uusi kansio backup



Lataa python scripti <https://github.com/theAIGuysCode/YoloGenerateTrainingFile> ja liitä se darknet kansioon. Aja ladattu python scripti `python generate_train.py` jos ei tule virhe ilmoituksia niin darknet\data kansiossa pitäisi olla nyt tiedosto nimeltä "train" ja tässä tekstitiedostossa on kaikkien kuviesi tiedostosijainnit.



4.Opetuksen aloittaminen

Lataa vielä sivustolta <https://pjreddie.com/darknet/yolo/> "pretrained convolutional weights" tiedosto joka toimii pohja weights tiedostona opetukselle.

Download Pretrained Convolutional Weights

For training we use convolutional weights that are pre-trained on Imagenet. We use weights from the darknet53 model. You can just download the weights for the convolutional layers [here \(76 MB\)](#).

Kun weights tiedosto on ladattu liitä se darknet kansioon.

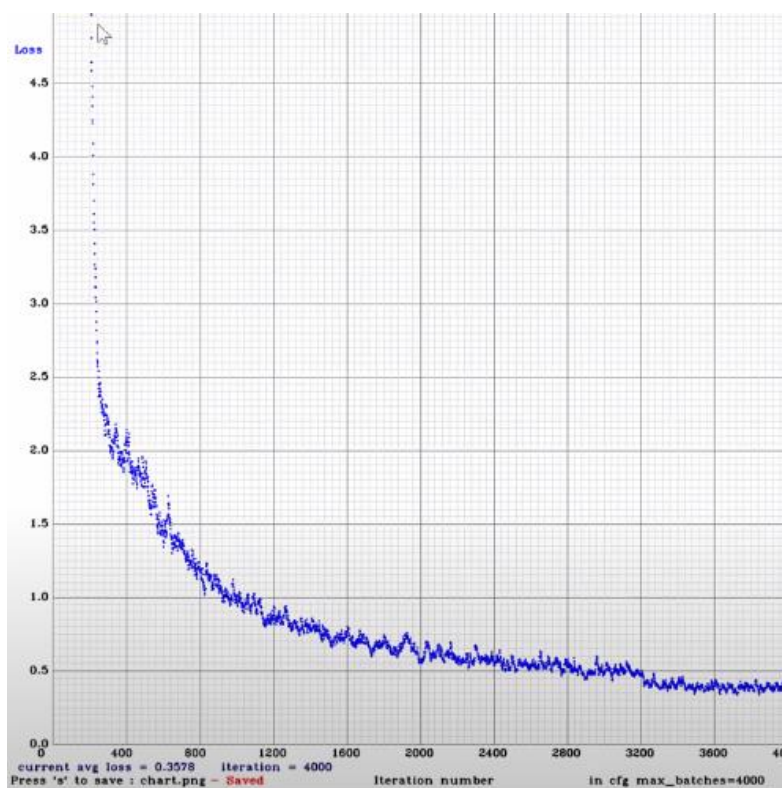
Ja nyt kaiken pitäisi olla valmiina oman weights tiedoston kouluttamisen aloittamista varten.

Avaa darknet kansio windows komentokehottessa.

Ja aja alla oleva komento, MUTTA vaihda cfg/yolov3-custom.cfg tilalle muokkaamasi cfg tiedoston polku.

```
darknet.exe detector train data/obj.data cfg/yolov3-custom.cfg darknet53.conv.74
```

Tämä avaa ikkunan jossa näkyy kuinka mones iteraatio ja kuinka suuri average loss on(mitä matalampi sitä parempi).



Ohjelma tallentaa Weights tiedostot darknet/backup kansioon ja jos haluaa jatkaa kouluttamista edellisestä weights tiedostoista niin korvaa "darknet53.conv.74" komentorivillä weights tiedostolla josta haluat jatkaa kouluttamista.

Darknet tallentaa joka iteraation jälkeen päivitetyn version weights tiedostosta. Tilanteessa jossa esimerkiksi tietokone on sammunut voi kouluttamista jatkaa komennolla

```
Darknet.exe detector train data/obj.data cfg/yolov3-custom.cfg backup/yolov3-custom_last.weights
```

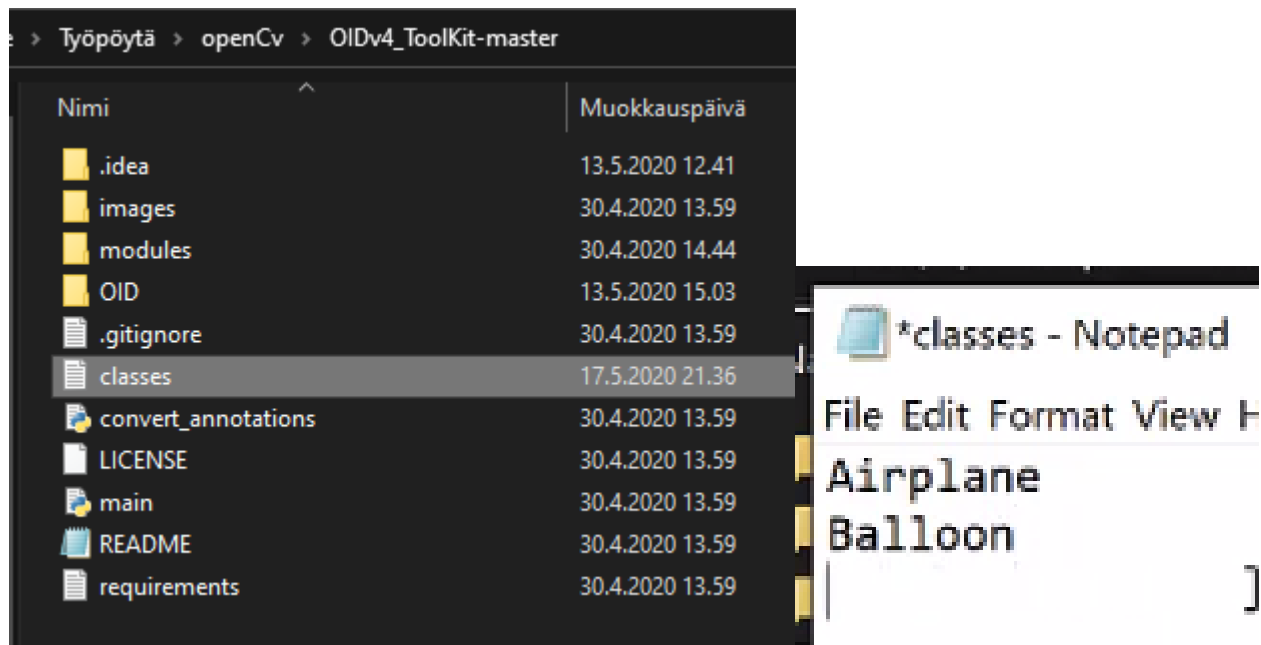
Muistilista jos kouluttaa uusia weights tiedostoja uusilla luokilla ja kuvilla

1.lataa kuvat(jos käyttää googlen tarjoamaa kuva apia)

OIDv4 toolkit aja python scripti main.py

```
python main.py downloader --classes Balloon Airplane --type_csv train --limit 400 --multiclass 1
```

Vaihda classes tiedostoon luokkien nimet allekkain



Aja scriptti: `OIDv4_ToolKit> python convert_annotations.py`

2.Kuvien sijainti ja cfg tiedosto

Siirrä OID\Dataset\train\ kansiota kuvat ja tekstitiedostot darknet\data\obj kansioon

Muokkaa darknet\data\obj.names tiedosto niin että se on sama kuin OIDv4 toolkit classes tiedosto. Ja muokkaa darknet\data\obj.data tiedostoon luokkiesi määrä.

Muokkaa kopioitua tiedostoa joka luotiin tiedostosta darknet\cfg\yolov3.cfg tai luo uusi kopio jos haluat säilyttää edellisen tiedoston asetukset.

Jos muokkaat edellistä cfg tiedostoa jonka loit niin pakolliset muutokset ovat

Max_batches = 2000*luokkien määrä ei alle 4000

Steps = maxbatches *0.80 , maxbatches *0,90 (kirjoita pelkät numerot)

3 eri [yolo] kohtaan luokkien määrä ja [yolo] kohtien yläpuolella oleviin [convolutional] kohtien filters muuttujan kohdalle (luokkien määrä+5)*3 (pelkkä numero)

Seuraavaksi aja darknet kansiossa oleva generate_train.py scripti

Kaiken pitäisi olla nyt valmista jotta voit aloittaa uuden weights tiedoston kouluttamisen.

Huomautuksia ja mietteitä jatkoa varten

<https://github.com/AlexeyAB/darknet> Sivulta löytyy apua ongelmiin ja mahdollisiin säätöihin joita voit tehdä kun koulutat uusia weights tiedostoja, voit myös yrittää muitakin malleja kuin yolov3 mutta omalla kohdalla se on ollut ainoa joka on ollut onnistunut malli.

Testaamatta vielä että ladataan google apista esim 6000 kuvaa 1000 per luokka ajetaan 12000 iteraatiota ja sen jälkeen ladataan uudet 6000 kuvaa 1000 per luokka ja jatkettaisiin kouluttamista. (lisäisikö tämä tarkkuutta?)

Testaamatta myös omien kuvien pohjalta tehdä kuvantunnistus johtuen siitä että kuvia tarvitaan satoja ellei tuhansia jotta kuvantunnistuksesta saadaan tarkka ja jos kuvia ei ole valmiiksi merkattu ne tulisi käydä yksi kerrallaan läpi ja merkata asiat jotka halutaan tunnistaa.

