

The difference between yolov4 and yolov4-tiny is: tiny is a compressed version of yolov4, which mainly runs the core version of small computing power cpu. The frame rate of the tiny version on jetso nano will be more than ten times higher than that of yolov4. We recommend using yolov4-tiny.

### 1. Insall CUDA, OpenCV, cuDNN

Input following command to download

```
git clone https://github.com/AlexeyAB/darknet.git
```

### 2. Input following command

```
cd darknet
```

```
sudo vim Makefile #modify Makefile
```

### 3. Modify the content as shown below

```
GPU=1
```

```
CUDNN=1
```

```
OPENCV=1
```

```
File Edit Tabs Help
GPU=1
CUDNN=1
CUDNN_HALF=0
OPENCV=1
AVX=0
OPENMP=0
LIBSO=0
ZED_CAMERA=0
ZED_CAMERA_v2_8=0

# set GPU=1 and CUDNN=1 to speedup on GPU
# set CUDNN_HALF=1 to further speedup 3 x times (Mixed-precision on Tensor Cores)
# GPU: Volta, Xavier, Turing and higher
# set AVX=1 and OPENMP=1 to speedup on CPU (if error occurs then set AVX=0)
# set ZED_CAMERA=1 to enable ZED SDK 3.0 and above
# set ZED_CAMERA_v2_8=1 to enable ZED SDK 2.X

USE_CPP=0
DEBUG=0

ARCH= -gencode arch=compute_30,code=sm_30 \
      -gencode arch=compute_35,code=sm_35 \
      -gencode arch=compute_50,code=[sm_50,compute_50] \
"Makefile" 1911 5663C 4 8 Top
```

### 4. Compile

Input following command in darknet directory.

```
make -j4
```

### 5. Copy the weight files yolov4.weights and yolov4-tiny.weights to the darknet directory.

```
nano@nano-desktop:~/darknet$ ls
3rdparty  cmake  darknet_video.py  LICENSE  README.md  yolov4-tiny.weights
backup    CMakeLists.txt  data              Makefile  results    yolov4.weights
build     darknet         image_yolov3.sh  net_cam_v3.sh  scripts   video_yolov3.sh
build.ps1 DarknetConfig.cmake.in  image_yolov4.sh  net_cam_v4.sh  src       video_yolov4.sh
build.sh  darknet_images.py  include          obj          predictions.jpg
cfg       darknet.py         json_mjpeg_streams.sh
```

## 6. Test

### Yolov4 image detection

```
./darknet detect cfg/yolov4.cfg yolov4.weights data/dog.jpg
```

```
./darknet detector test cfg/coco.data cfg/yolov4.cfg yolov4.weights data/dog.jpg
```

### Yolov4-tiny image detection

```
./darknet detect cfg/yolov4-tiny.cfg yolov4-tiny.weights data/dog.jpg
```

```
./darknet detector test cfg/coco.data cfg/yolov4-tiny.cfg yolov4-tiny.weights data/dog.jpg
```

# Change the detection threshold

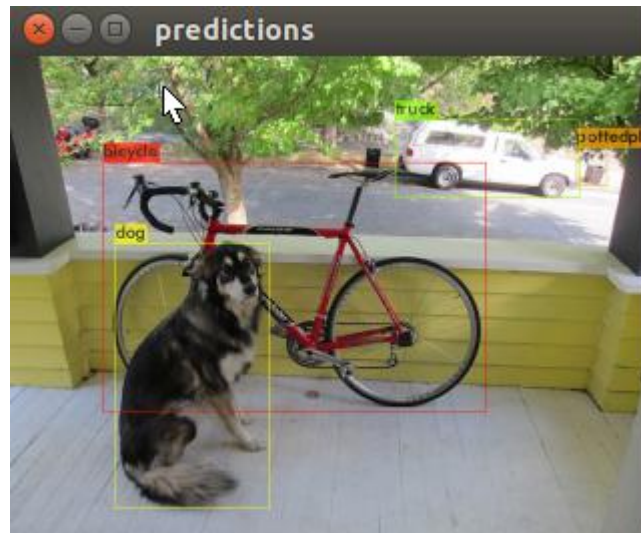
# By default, YOLO only displays detected objects with a confidence of .25 or higher.

You can modify this setting with the following command.

For example, to display all detections, you can set the threshold to 0.1.

```
./darknet detect cfg/yolov4-tiny.cfg yolov4-tiny.weights data/dog.jpg -thresh 0.1
```

```
21 conv 128 3 x 3/ 1 26 x 26 x 128 -> 26 x 26 x 128 0.199 BF
22 route 21 20 -> 26 x 26 x 256
23 conv 256 1 x 1/ 1 26 x 26 x 256 -> 26 x 26 x 256 0.089 BF
24 route 18 23 -> 26 x 26 x 512
25 max 2x 2/ 2 26 x 26 x 512 -> 13 x 13 x 512 0.000 BF
26 conv 512 3 x 3/ 1 13 x 13 x 512 -> 13 x 13 x 512 0.797 BF
27 conv 256 1 x 1/ 1 13 x 13 x 512 -> 13 x 13 x 256 0.044 BF
28 conv 512 3 x 3/ 1 13 x 13 x 256 -> 13 x 13 x 512 0.399 BF
29 conv 255 1 x 1/ 1 13 x 13 x 512 -> 13 x 13 x 255 0.044 BF
30 yolo
[yolo] params: iou loss: ciou (4), iou_norm: 0.07, obj_norm: 1.00, cls_norm: 1.00, delta_norm: 1.00, scale_x_y: 1.05
nms kind: greedynms (1), beta = 0.600000
31 route 27 -> 13 x 13 x 256
32 conv 128 1 x 1/ 1 13 x 13 x 256 -> 13 x 13 x 128 0.011 BF
33 upsample 2x 13 x 13 x 128 -> 26 x 26 x 128
34 route 33 23 -> 26 x 26 x 384
35 conv 256 3 x 3/ 1 26 x 26 x 384 -> 26 x 26 x 256 1.196 BF
36 conv 255 1 x 1/ 1 26 x 26 x 256 -> 26 x 26 x 255 0.088 BF
37 yolo
[yolo] params: iou loss: ciou (4), iou_norm: 0.07, obj_norm: 1.00, cls_norm: 1.00, delta_norm: 1.00, scale_x_y: 1.05
nms kind: greedynms (1), beta = 0.600000
Total BFLOPS 6.910
avg outputs = 310203
Allocate additional workspace_size = 26.22 MB
Loading weights from yolov4-tiny.weights...
seen 64, trained: 32012 K-images (500 Kilo-batches_64)
Done! Loaded 38 layers from weights-file
Detection layer: 30 - type = 28
Detection layer: 37 - type = 28
data/dog.jpg: Predicted in 1073.680000 milli-seconds.
bicycle: 29%
person: 25%
dog: 72%
cat: 16%
truck: 82%
car: 46%
person: 11%
```



### Yolov4 video detection

(Users need to upload the video files to be detected to the data folder)

`./darknet detector demo cfg/coco.data cfg/yolov4.cfg yolov4.weights data/123.mp4`

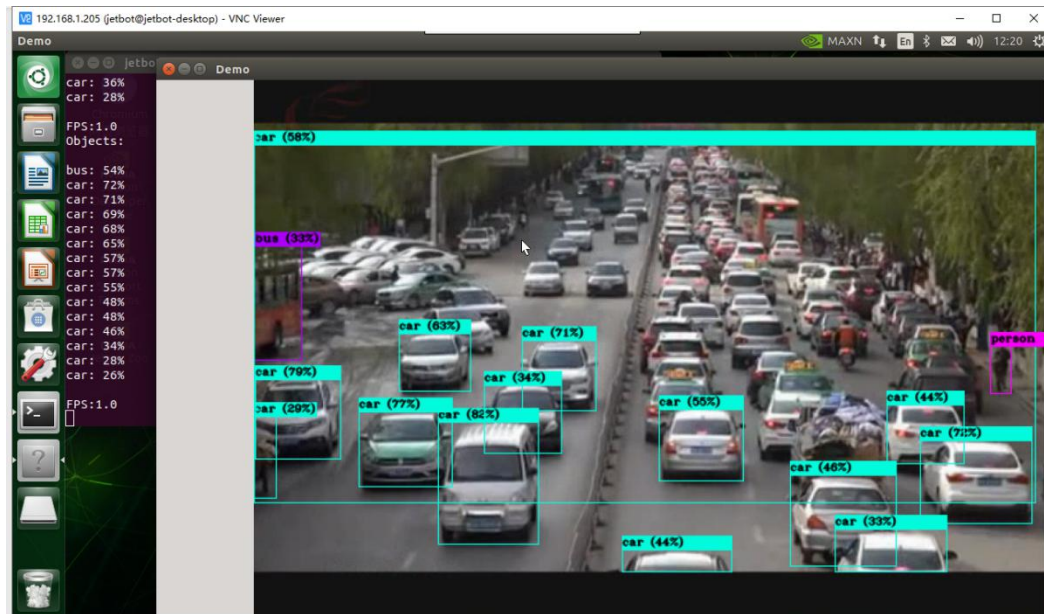
### Yolov4-tiny video detection

(Users need to upload the video files to be detected to the data folder)

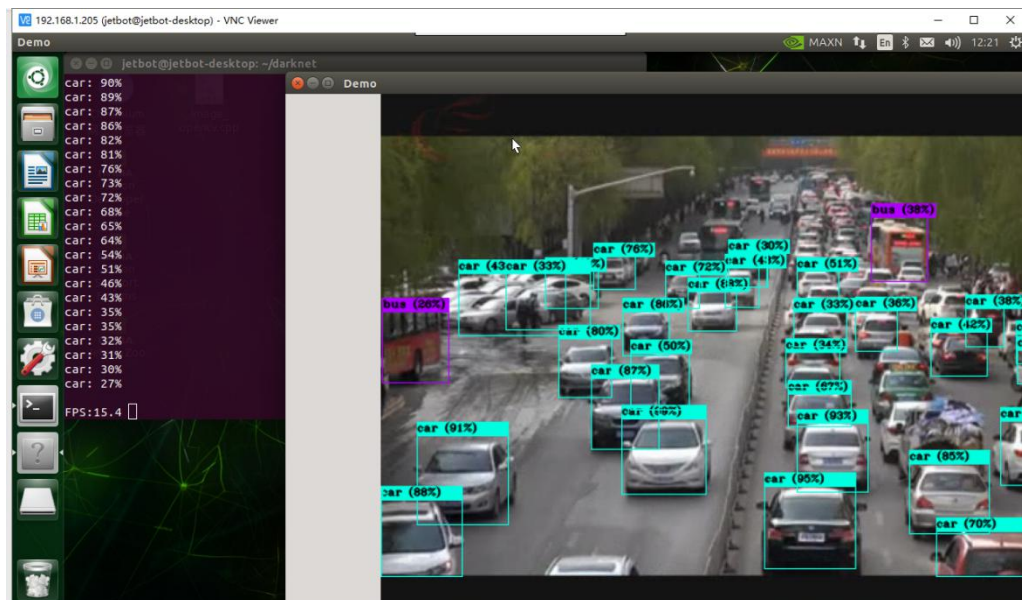
`./darknet detector demo cfg/coco.data cfg/yolov4-tiny.cfg yolov4-tiny.weights data/xxx.mp4`

```
nx@nx-desktop:~/darknet-master$ ./darknet detector demo cfg/coco.data cfg/yolov4.cfg yolov4.weights data/123.mp4
CUDA-version: 10020 (10020), cuDNN: 8.0.0, GPU count: 1
OpenCV version: 4.1.1
Demo
0 : compute_capability = 720, cudnn_half = 0, GPU: Xavier
net.optimized_memory = 0
mini_batch = 1, batch = 1, time_steps = 1, train = 0
layer  filters  size/strd(dil)  input  output
0 conv  32  3 x 3/ 1  608 x 608 x 3 -> 608 x 608 x 32 0.639 BF
1 conv  64  3 x 3/ 2  608 x 608 x 32 -> 304 x 304 x 64 3.407 BF
2 conv  64  1 x 1/ 1  304 x 304 x 64 -> 304 x 304 x 64 0.757 BF
3 route  1  -> 304 x 304 x 64
4 conv  64  1 x 1/ 1  304 x 304 x 64 -> 304 x 304 x 64 0.757 BF
5 conv  32  1 x 1/ 1  304 x 304 x 64 -> 304 x 304 x 32 0.379 BF
6 conv  64  3 x 3/ 1  304 x 304 x 32 -> 304 x 304 x 64 3.407 BF
```

Yolov4 result as shown below.



Yolov4-tiny result as shown below.



Yolov4 camera detection in real time.

`./darknet detector demo cfg/coco.data cfg/yolov4.cfg yolov4.weights /dev/video1`

Yolov4 camera detection in real time.

`./darknet detector demo cfg/coco.data cfg/yolov4-tiny.cfg yolov4-tiny.weights /dev/video1`

**Tip:** Select the number corresponding to the USB camera for the video device.



