Cardboard Box Detection and Localization using RetinaNet (Keras)

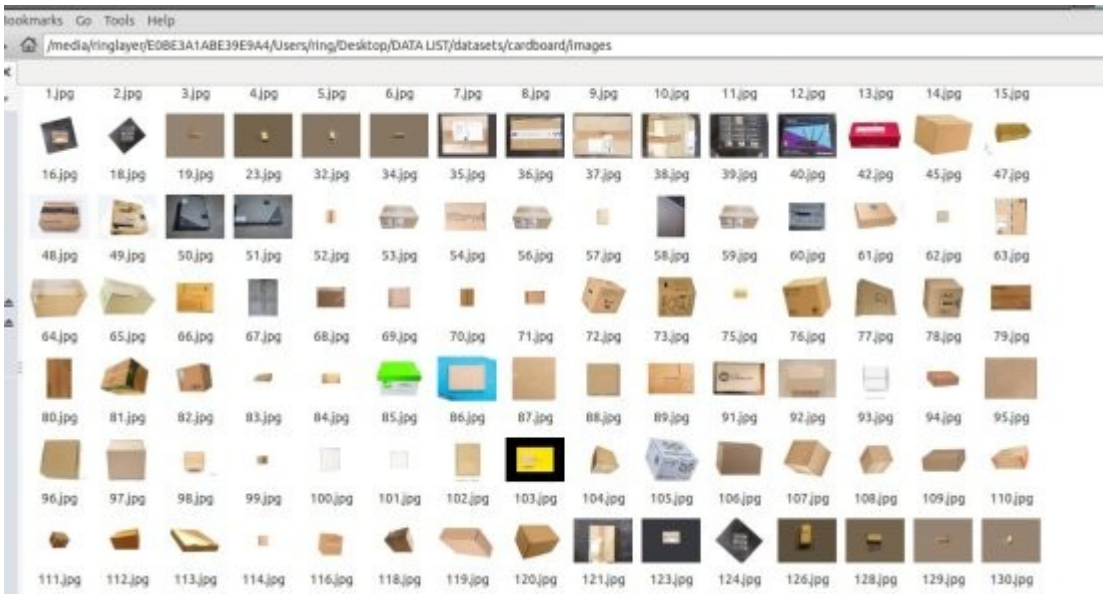
Antonius Robotsoft - www.robotsoft.co.id – https://github.com/antoniusrobotsoft

Keras RetinaNet is keras implementation of RetinaNet object detection as described in Focal Loss for Dense Object Detection Paper by Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, Piotr Dollar (https://arxiv.org/abs/1708.02002).

We are going to train a model using keras retinanet to detect and localize a custom object : “cardboard box” from the image.

Step 1. Dataset Preparation

In order to detect cardboard boxes in the image, I have prepared several cardboard box images :



The dataset can be downloaded from :

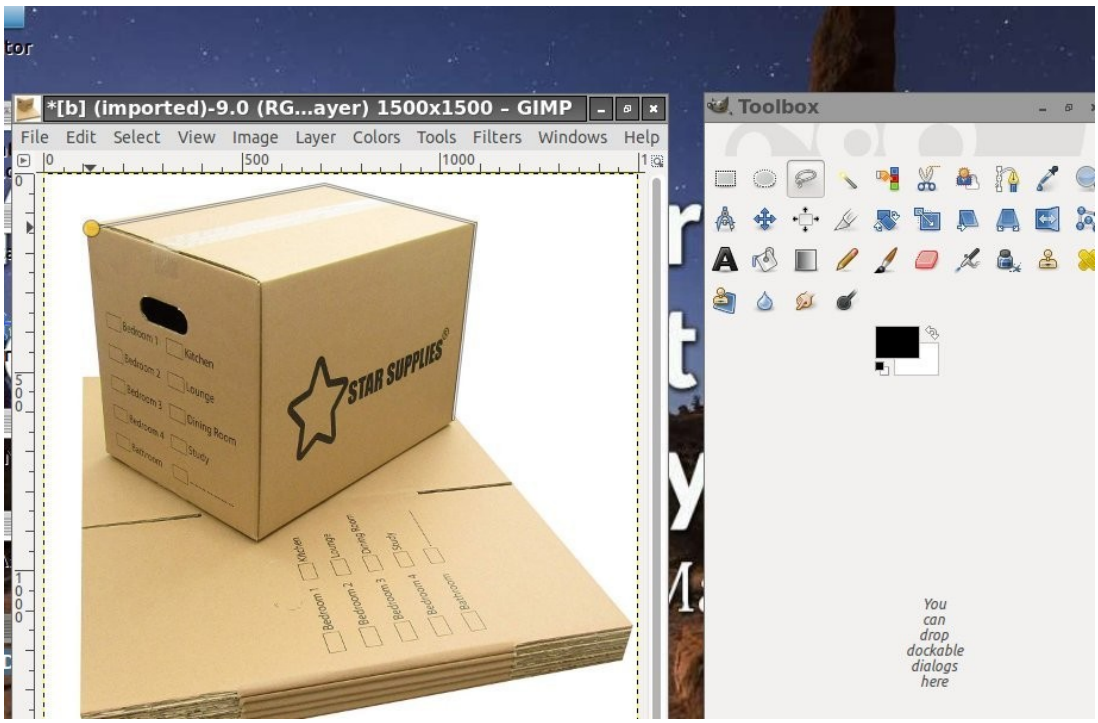
https://jfids.jasaplus.com/repo/cardboard.tar.bz2

https://jfids.jasaplus.com/repo/cardboard2.tar.bz2

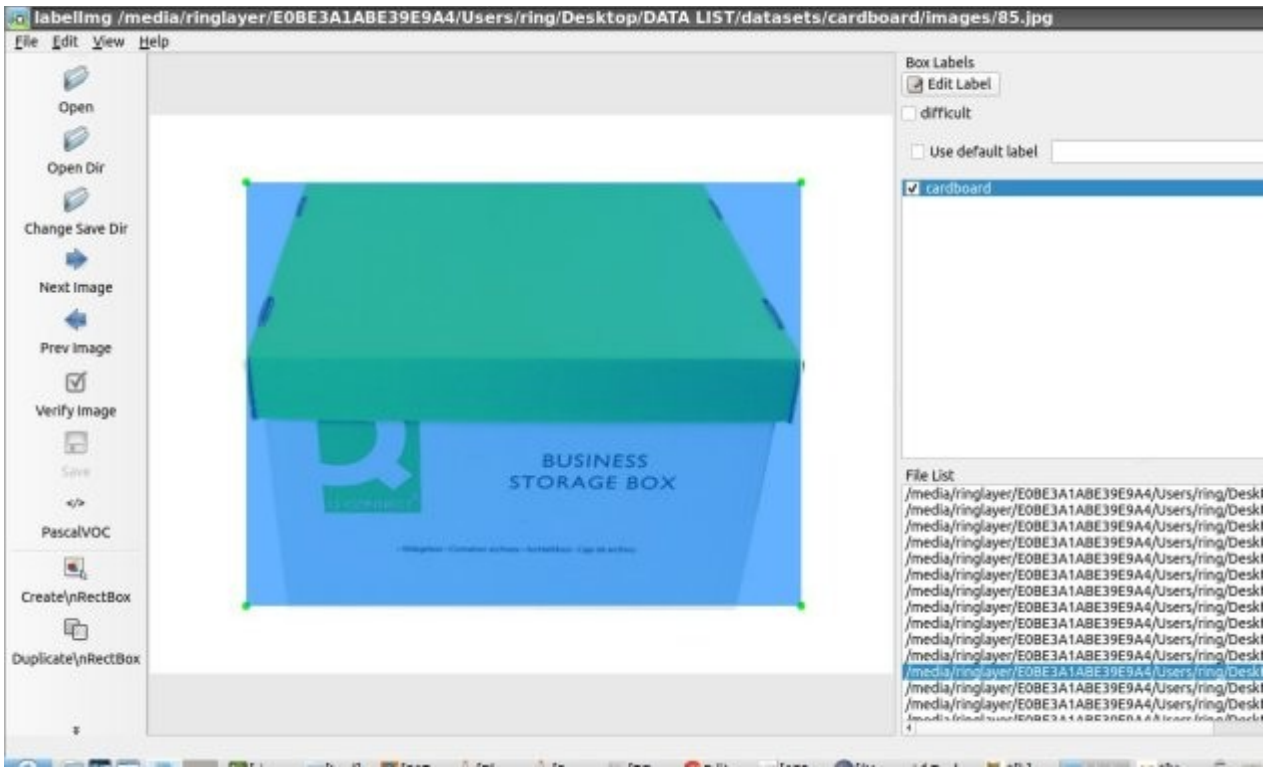
Keras Retinanet will resize any input image before input layer, the input image will be resized into an image with the maximal height 800px. For example, when we give input image 640×480 it will be resized into 1067×800. When we give input image 800×600, it will be resized into 1067×800.

This dataset was created using gimp and labelImg. It contains 640×480 images. We are going to train a new model using this dataset to detect image from webcam with 640×480 resolution.

Gimp was used to extract only relevant part of the image using lasso tool :



labelImg was used for labelling specific areas within image that contains our object. Do not use too much margin when labelling the object since the quality of our detection depends on our dataset quality.



Step 2. Converting the Dataset from Pascal VOC Format into CSV Annotation Format for Keras RetinaNet

Next, we need to convert the dataset from voc into csv. We are going to use conv_pascal_to_csv, this utility can be cloned from github url :

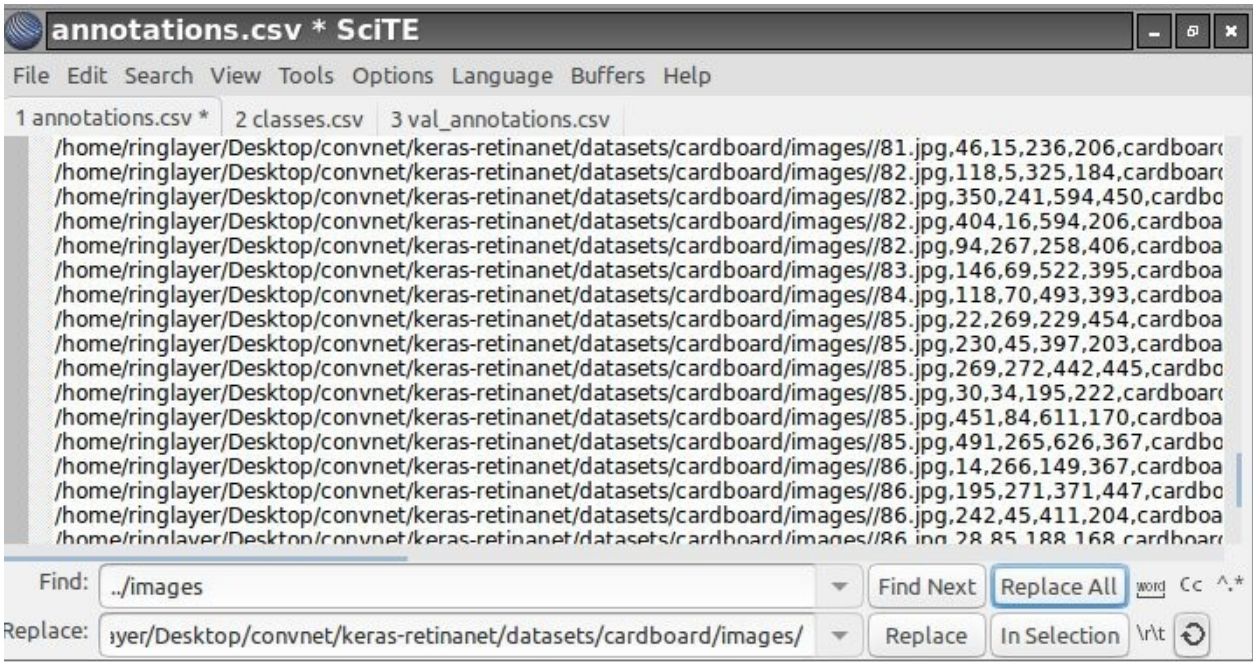
https://github.com/akchan/conv_pascal_to_csv.git

$ ruby conv_pascal_to_csv.rb –help
Usage: conv_pascal_to_csv [options]
–annotation-path PATH path to xml annotations directory
–image-path PATH path to jpeg images directory
–val-ratio float sample ratio for validation (0.0 – 1.0). default=0.1
$ ruby conv_pascal_to_csv.rb –annotation-path /home/ringlayer/Desktop/convnet/keras-retinanet/datasets/cardboard2/voclabels –image-path /home/ringlayer/Desktop/convnet/keras-retinanet/datasets/cardboard2/images

Once the conversion finished, we will get 3 files in csv directory :

$ ls csv
annotations.csv classes.csv val_annotations.csv

I replaced image paths at annotations.csv and val_annotations.csv with absolute paths of my dataset location :



Step 3. Start Training New Model

In order to train new model, we can do **transfer learning** by using previously trained weight. Here, I use Resnet50 as backbone to train new model :

python3 keras_retinanet/bin/train.py –freeze-backbone –random-transform –weights weights/ResNet-50-model.keras.h5 –batch-size 8 –steps 500 –epochs 15 csv csv/annotations.csv csv/classes.csv

These parameters required since we were not feeding the neural net at once with all training images, instead we will divide it into several epochs (each epoch will be divided into 500 steps)

–batch-size 8

We used batch size 8, The batch size defines the number of images that will be propagated through the network.

–epochs 15

One epoch means an entire dataset is passed forward and backward through the neural network. Since we used 15 epochs, it means it will need 15 epochs to complete the entire dataset provided at annotations.csv.

–steps 500

We use 500 steps on each epoch. This is the number of batch iterations before a training epoch is considered finished.

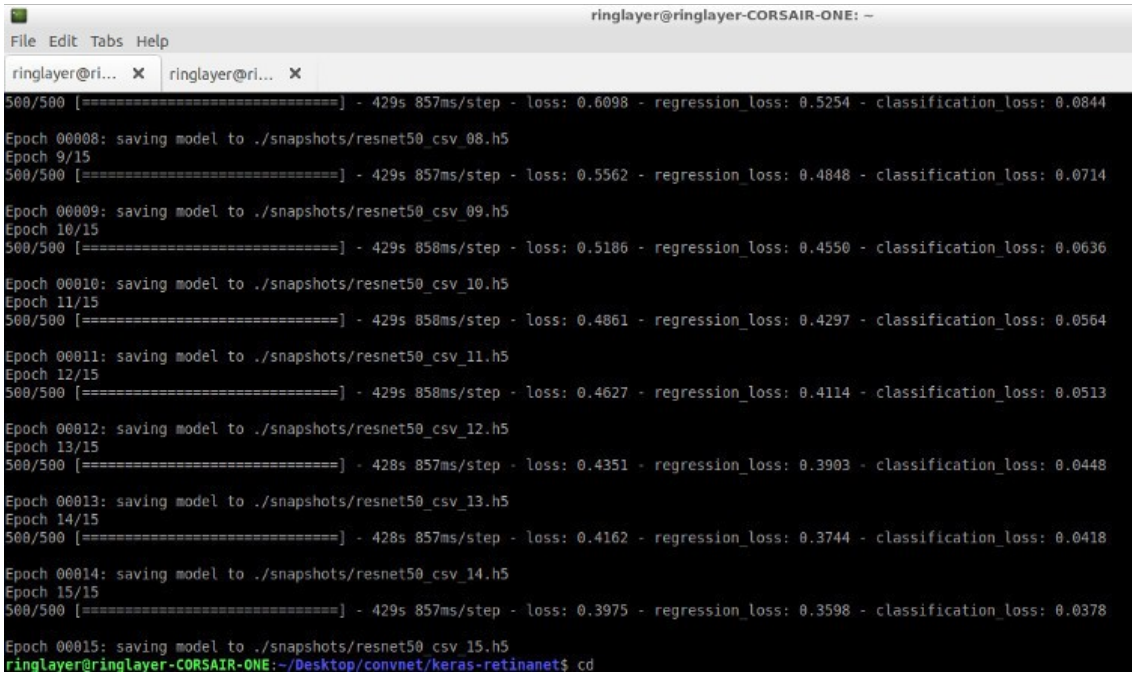
Here, I trained the model using gtx1080 with 8 gb vram :

```
ringlayer@ringlayer-CORSAIR-ONE:~/Desktop/convnet/keras-retinanet$ ./train
Using TensorFlow backend.
2019-07-02 01:30:34.978905: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:964] successful NUMA node read
t least one NUMA node, so returning NUMA node zero
2019-07-02 01:30:34.979291: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1432] Found device 0 with properti
name: GeForce GTX 1080 major: 6 minor: 1 memoryClockRate(GHz): 1.7335
pciBusID: 0000:01:00.0
totalMemory: 7.93GiB freeMemory: 7.75GiB
2019-07-02 01:30:34.979306: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1511] Adding visible gpu devices:
2019-07-02 01:30:35.155345: I tensorflow/core/common_runtime/gpu/gpu_device.cc:982] Device interconnect StreamExe
2019-07-02 01:30:35.155374: I tensorflow/core/common_runtime/gpu/gpu_device.cc:988]      0
2019-07-02 01:30:35.155383: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1001] 0:   N
2019-07-02 01:30:35.155506: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1115] Created TensorFlow device (/
 memory) -> physical GPU (device: 0, name: GeForce GTX 1080, pci bus id: 0000:01:00.0, compute capability: 6.1)
2019-07-02 01:30:35.155983: I tensorflow/core/common_runtime/process_util.cc:69] Creating new thread pool with de
sm_threads for best performance.
path : csv/classes.csv
path : csv/annotations.csv
Creating model, this may take a second...

Layer (type)                 Output Shape              Param #     Connected to
==================================================================================================
input_1 (InputLayer)         (None, None, None, 3 0
__________________________________________________________________________________________________
conv1 (Conv2D)               (None, None, None, 6 9408        input_1[0][0]
__________________________________________________________________________________________________
bn_conv1 (BatchNormalization (None, None, None, 6 256         conv1[0][0]
__________________________________________________________________________________________________
conv1_relu (Activation)       (None, None, None, 6 0           bn_conv1[0][0]
__________________________________________________________________________________________________
pool1 (MaxPooling2D)          (None, None, None, 6 0           conv1_relu[0][0]
__________________________________________________________________________________________________
res2a_branch2a (Conv2D)       (None, None, None, 6 4096        pool1[0][0]
__________________________________________________________________________________________________
bn2a_branch2a (BatchNormalizati (None, None, None, 6 256         res2a_branch2a[0][0]
__________________________________________________________________________________________________
res2a_branch2a_relu (Activation (None, None, None, 6 0           bn2a_branch2a[0][0]
```

This configuration takes about 7gb vram

```
ringlayer@ringlayer-CORSAIR-ONE:~/Desktop/convnet/darknet$ nvidia-smi
Tue Jul  2 01:47:04 2019
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 410.48                 Driver Version: 410.48                    |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  GeForce GTX 1080    Off  | 00000000:01:00.0  On |                  N/A |
| 54%   66C    P2   165W / 180W |   7845MiB /  8116MiB |     99%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|    0      1135      G   /usr/lib/xorg/Xorg                            52MiB |
|    0      6465      C   python3                                     7781MiB |
+-----------------------------------------------------------------------------+
```

In order to fine tune new model training we can use a gpu with 11 gb vram.



The trained weight on each epoch will be saved to snapshots directory. We are going to use resnet50_csv_15.h5 for our final model to detect cardboard (I renamed resnet50_csv_15.h5 into cb20.h5).

The source code for testing this model on webcam and video can be fetched from github :

https://github.com/antoniusrobotsoft

Testing our model to detect cardboard in video and image :