Yolo3 pytorch Object detection

```
from __future__ import division

from models import *
from utils.utils import *
from utils.datasets import *

import os
import sys
import time
import datetime
import argparse

from PIL import Image

import torch
from torch.utils.data import DataLoader
from torchvision import datasets
from torch.autograd import Variable

import matplotlib.pyplot as plt
import matplotlib.patches as patches
from matplotlib.ticker import NullLocator


device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

os.makedirs("output", exist_ok=True)

# Set up model
model = Darknet("config/yolov3.cfg", img_size=416).to(device)
# Load darknet weights
model.load_darknet_weights("weights/yolov3.weights")


model.eval()  # Set in evaluation mode

dataloader = DataLoader(
    ImageFolder("data/samples", img_size=416),
    batch_size=1,
    shuffle=False,
    num_workers=0,
)

classes = load_classes("data/coco.names")  # Extracts class labels from file

Tensor = torch.cuda.FloatTensor if torch.cuda.is_available() else torch.FloatTensor

imgs = []  # Stores image paths
```

```
    imgs = []  # Stores image paths
    img_detections = []  # Stores detections for each image index

    print("\nPerforming object detection:")
    prev_time = time.time()
    for batch_i, (img_paths, input_imgs) in enumerate(dataloader):
        # Configure input
        input_imgs = Variable(input_imgs.type(Tensor))

        # Get detections
        with torch.no_grad():
            detections = model(input_imgs)
            detections = non_max_suppression(detections, 0.8, 0.4)

        # Log progress
        current_time = time.time()
        inference_time = datetime.timedelta(seconds=current_time - prev_time)
        prev_time = current_time
        print("\t+ Batch %d, Inference Time: %s" % (batch_i, inference_time))

        # Save image and detections
        imgs.extend(img_paths)
        img_detections.extend(detections)

    # Bounding-box colors
    cmap = plt.get_cmap("tab20b")
    colors = [cmap(i) for i in np.linspace(0, 1, 20)]

    print("\nSaving images:")
    # Iterate through images and save plot of detections
    for img_i, (path, detections) in enumerate(zip(imgs, img_detections)):

        print("(%d) Image: '%s'" % (img_i, path))

        # Create plot
        img = np.array(Image.open(path))
        plt.figure()
        fig, ax = plt.subplots(1)
        ax.imshow(img)

        # Draw bounding boxes and labels of detections
        if detections is not None:
            # Rescale boxes to original image
            detections = rescale_boxes(detections, 416, img.shape[:2])
            unique_labels = detections[:, -1].cpu().unique()
            n_cls_preds = len(unique_labels)
            bbox_colors = random.sample(colors, n_cls_preds)
            for x1, y1, x2, y2, conf, cls_conf, cls_pred in detections:

                print("\t+ Label: %s, Conf: %.5f" % (classes[int(cls_pred)], cls_conf.item()))

                box_w = x2 - x1
```

```
            box_h = y2 - y1

            color = bbox_colors[int(np.where(unique_labels == int(cls_pred))[0])]
            # Create a Rectangle patch
            bbox = patches.Rectangle((x1, y1), box_w, box_h, linewidth=2, edgecolor=color, fa
            # Add the bbox to the plot
            ax.add_patch(bbox)
            # Add label
            plt.text(
                x1,
                y1,
                s=classes[int(cls_pred)],
                color="white",
                verticalalignment="top",
                bbox={"color": color, "pad": 0},
            )

    # Save generated image with detections
    plt.axis("off")
    plt.gca().xaxis.set_major_locator(NullLocator())
    plt.gca().yaxis.set_major_locator(NullLocator())
    filename = path.split("/")[-1].split(".")[0]
    plt.savefig(f"output/{filename}.png", bbox_inches="tight", pad_inches=0.0)
    plt.close()
```

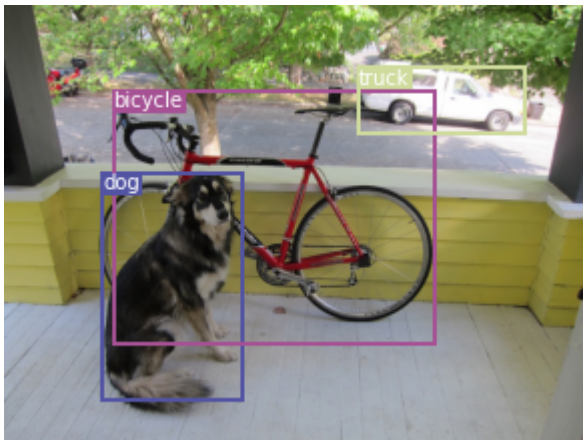
```
        + Batch 0, Inference Time: 0:00:00.049708
        + Batch 1, Inference Time: 0:00:00.034727
        + Batch 2, Inference Time: 0:00:00.031957
        + Batch 3, Inference Time: 0:00:00.033261
        + Batch 4, Inference Time: 0:00:00.035032
        + Batch 5, Inference Time: 0:00:00.059134
        + Batch 6, Inference Time: 0:00:00.033763
        + Batch 7, Inference Time: 0:00:00.032069
        + Batch 8, Inference Time: 0:00:00.036263

Saving images:
(0) Image: 'data/samples/dog.jpg'
        + Label: dog, Conf: 0.99335
        + Label: bicycle, Conf: 0.99981
        + Label: truck, Conf: 0.94229
(1) Image: 'data/samples/eagle.jpg'
        + Label: bird, Conf: 0.99703
(2) Image: 'data/samples/field.jpg'
        + Label: person, Conf: 0.99996
        + Label: horse, Conf: 0.99977
        + Label: dog, Conf: 0.99409
(3) Image: 'data/samples/giraffe.jpg'
        + Label: giraffe, Conf: 0.99959
        + Label: zebra, Conf: 0.97958
(4) Image: 'data/samples/herd_of_horses.jpg'
        + Label: horse, Conf: 0.99459
        + Label: horse, Conf: 0.99352
        + Label: horse, Conf: 0.96845
        + Label: horse, Conf: 0.99478
(5) Image: 'data/samples/messi.jpg'
        + Label: person, Conf: 0.99993
        + Label: person, Conf: 0.99984
        + Label: person, Conf: 0.99996
(6) Image: 'data/samples/person.jpg'
        + Label: person, Conf: 0.99883
        + Label: dog, Conf: 0.99275
(7) Image: 'data/samples/room.jpg'
        + Label: chair, Conf: 0.99906
        + Label: chair, Conf: 0.96942
        + Label: clock, Conf: 0.99971
(8) Image: 'data/samples/street.jpg'
        + Label: car, Conf: 0.99977
        + Label: car, Conf: 0.99402
        + Label: car, Conf: 0.99841
        + Label: car, Conf: 0.99785
        + Label: car, Conf: 0.97907
        + Label: car, Conf: 0.95370
        + Label: traffic light, Conf: 0.99995
        + Label: car, Conf: 0.62254
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
```

```
<Figure size 432x288 with 0 Axes>
```

```
from IPython.display import Image
Image('/content/PyTorch-YOLOv3/data/samples/dog.jpg')
```

```
Image('/content/PyTorch-YOLOv3/output/dog.png')
```


truck
bicycle
dog