Documentation

PROBLEM STATEMENT

To Detect Whether object in a image or video is cat, dog or human.

INTRODUCTION

YOLO is an abbreviation for the term 'You Only Look Once'. This is an algorithm that detects and recognizes various objects in a picture (in real-time). Object detection in YOLO is done as a regression problem and provides the class probabilities of the detected images.

YOLO algorithm employs convolutional neural networks (CNN) to detect objects in real-time. As the name suggests, the algorithm requires only a single forward propagation through a neural network to detect objects.

This means that prediction in the entire image is done in a single algorithm run. The CNN is used to predict various class probabilities and bounding boxes simultaneously.

The YOLO algorithm consists of various variants. Some of the common ones include tiny YOLO ,YOLOv3 and YOLOv4.

CHALLENGES ENCOUNTER IN OBJECT DETECTION

1. Variable Number of Objects
2. Multiple Spatial Scales and Aspect Ratios
3. Modeling
4. Limited Data
5. Speed for Real-Time detection

VARIOUS APPROACHES

1. R-CNN Family
2. Single Shot MultiBox Detector
3. YOLO

Until now, we saw some very famous and well performing architectures for Object detection. All these algorithms solved some problems mentioned in the Challenges but fail on the most important one — Speed for real-time object detection

The biggest problem with the R-CNN family of networks is their speed — they were incredibly slow, obtaining only 5 FPS on a GPU.

YOLO algorithm gives a much better performance on all the parameters we discussed along with a high fps for real-time usage. YOLO algorithm is an algorithm based on regression, instead of selecting the interesting part of an Image, it predicts classes and bounding boxes for the whole image in **one run of the Algorithm.**

Why YOLO?

As a single-stage detector, YOLO performs classification and bounding box regression in one step, making it much faster than most convolutional neural networks. For example, YOLO object detection is more than 1000x faster than R-CNN and 100x faster than Fast R-CNN.

Machine Learning Pipeline

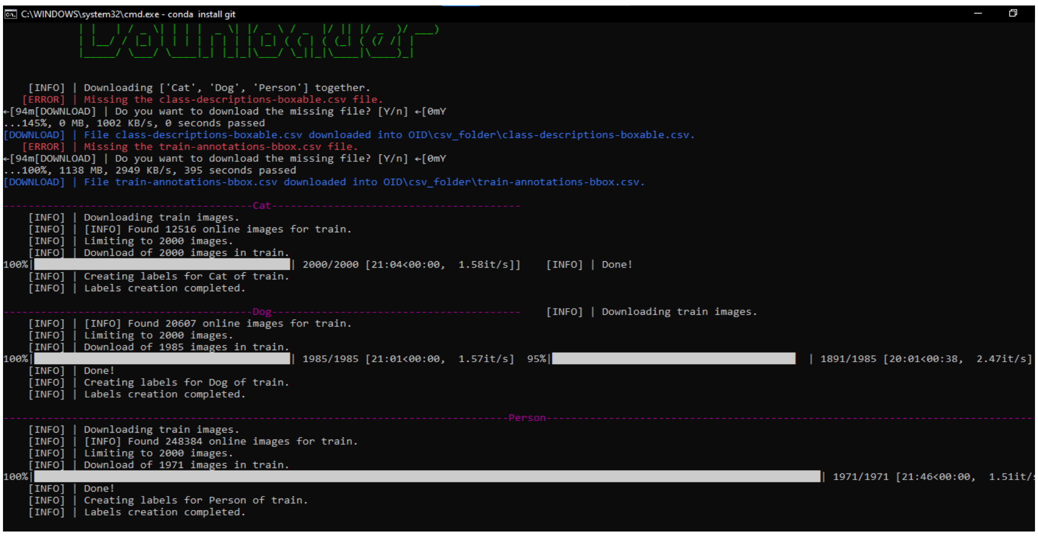
Object detection for your own custom data by applying Transfer Learning using YOLOv4

Step 1: Prepare dataset.

a) Create a dataset of the object for which I want to perform its detection. I may scrape images from **Google Images** to download data or use any other source for your data.

b) Clean your dataset by removing unwanted/irrelevant images. Also, make sure all the images are of the format **.jpg.**

Once I've done with the above two steps, we're good to go for the next step.



Step 2: Data Annotation.

I've Downloaded annotated images from Open Images Dataset.

a.) A **classes.txt** file is generated that includes the list of all classes I have annotated in my dataset. For every image file annotated, a corresponding **.txt** file is also generated that includes the metadata.

The metadata includes the following – object_id, center_x, center_y, width, height

object_id represents the number corresponding to the object category which we listed in 'classes.txt' earlier.

center_x and **center_y** represent the center point of the bounding box. But they are normalized to range between 0 and 1 by dividing by the width and height of the image.

width and **height** represents the width and height of the bounding box. Again normalized to the range 0 to 1 dividing by the original width and height of the image.

Step 3: Training the model.

Once I have labelled all your data, I may go ahead with the actual training process of the model. **.** I've to make sure that I have a healthy dataset size and have correctly labelled the objects if I want good accuracy.

(i) Check if NVIDIA GPU is enabled for training.
(ii) Clone, configure and compile Darknet.
(iii) Configure yolov4.cfg file.
(iv) Create .names and .data files
(v) Save yolov4_train.cfg and classes.names files in Google Drive.
(vi) Partition the dataset in train and test
(vii) Create train.txt file.
(viii) Download pre-trained weights for the convolutional layers file.
(ix) Start training.

The model will take some time to train depending upon your dataset size and the no. of classes.

It should take about 10-12 hours for training 3 classes, so you can estimate the approximate time required for training your own custom model depending upon your dataset size and no. of classes.

In case the training stopped for some reasons i.e. because of network or power failure or non-availability of GPU resource allocation or for any other reason, don't worry. You can continue the training process from the last saved weights. Simply comment the first line and uncomment the last line in the above cell and rerun it.

Step 4: Testing the model

Once the model is trained completely, atleast three files will be downloaded inside the backup foloder inside yolov4 folder on your Google Drive, depending upon the model size; as shown in the figure below.

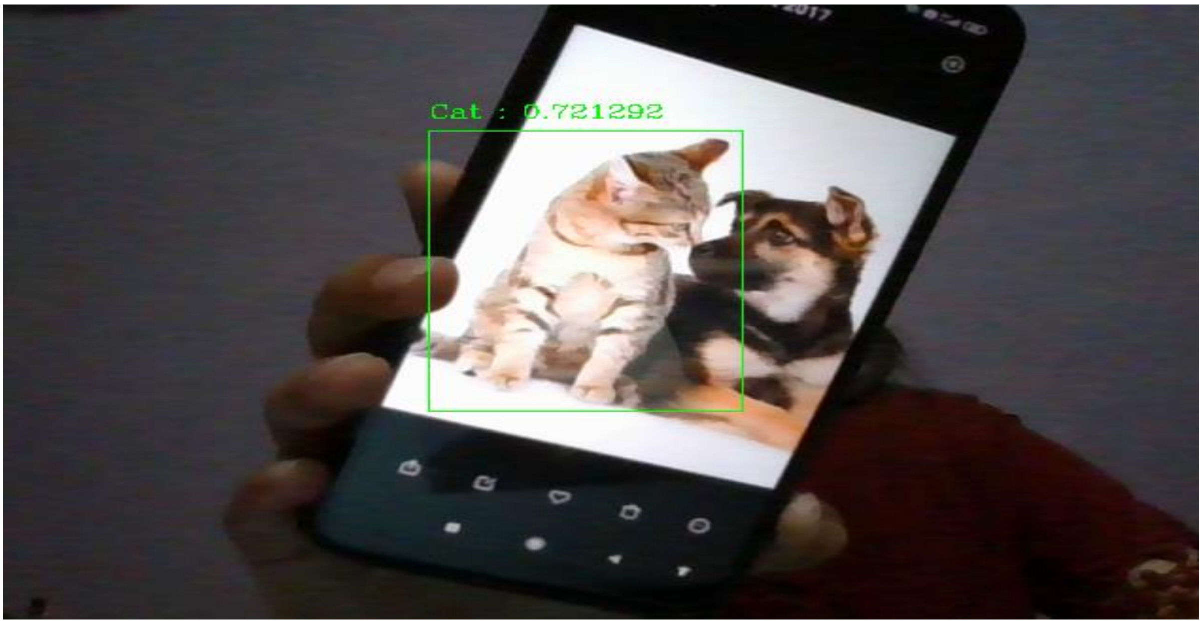
Name ↑	Owner	Last modified	File size
yolov4_train_1000.weights	Ayushi Tripathi	Oct 30, 2021 Ayushi Tripathi	244.2 MB
yolov4_train_2000.weights	Ayushi Tripathi	Oct 31, 2021 me	244.2 MB
yolov4_train_last.weights	me	Oct 31, 2021 Ayushi Tripathi	244.2 MB

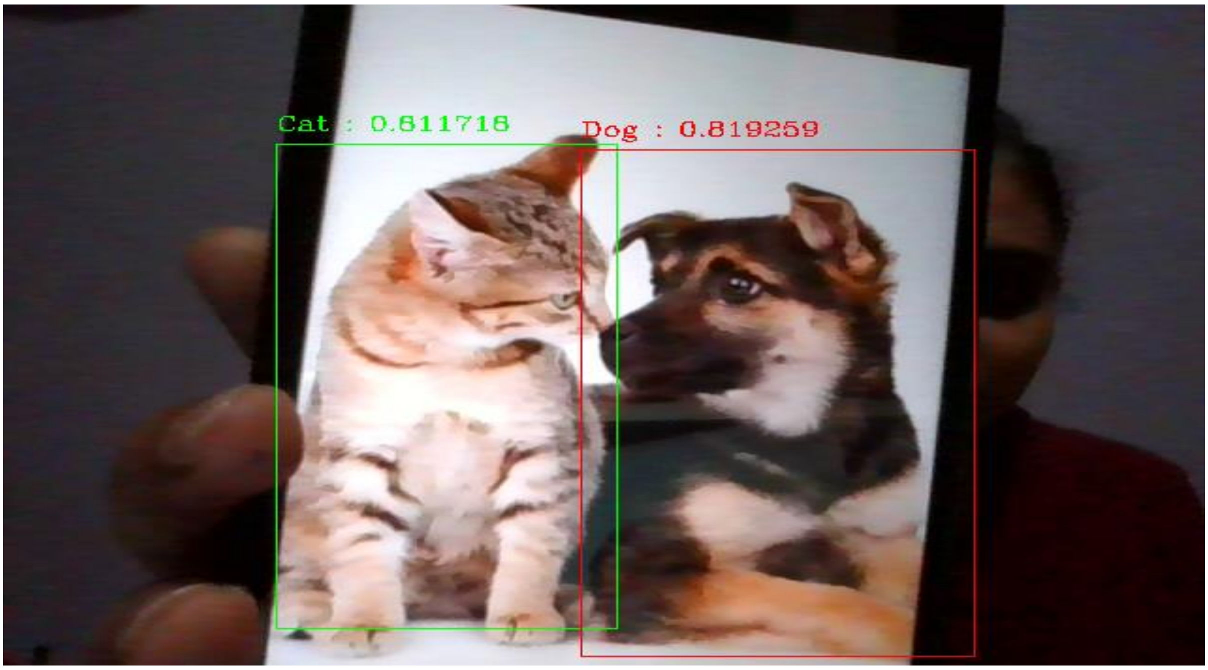
Step 5 Check the model performance using map function. Performance of my model is shown below.

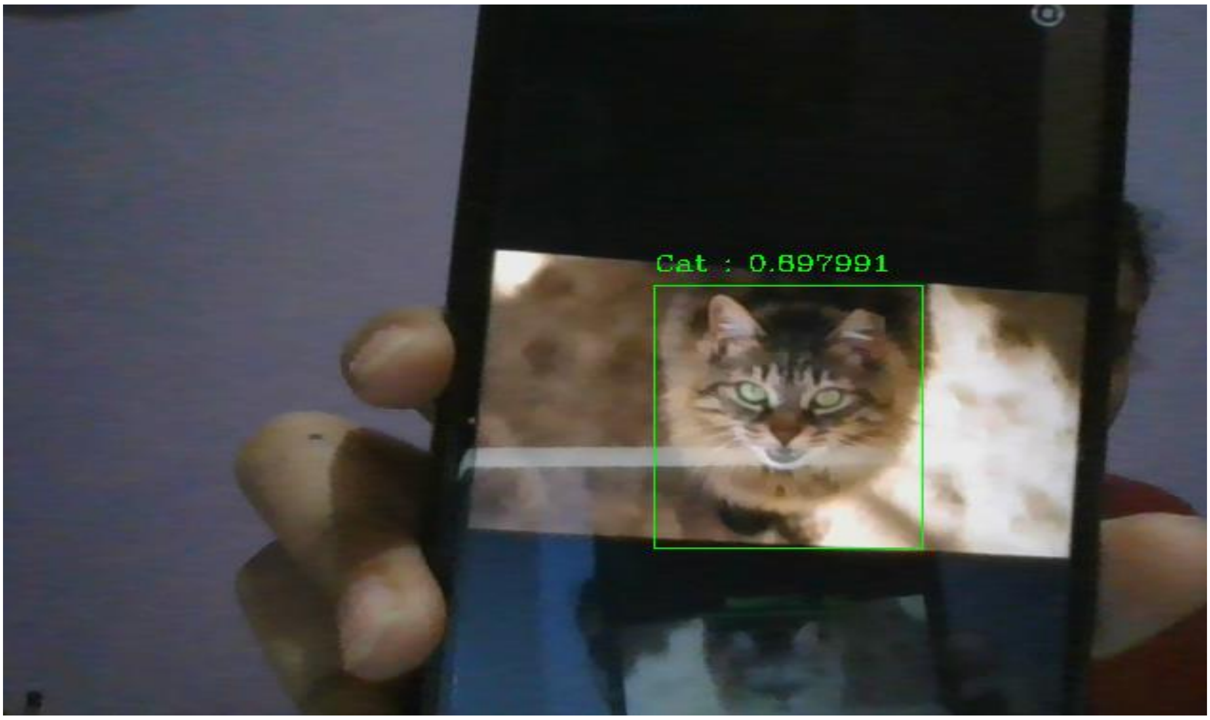
```
 detections_count = 10486, unique_truth_count = 2755
class_id = 0, name = Cat, ap = 87.46%          (TP = 394, FP = 42)
class_id = 1, name = Dog, ap = 81.83%          (TP = 416, FP = 82)
class_id = 2, name = Person, ap = 51.95%       (TP = 711, FP = 311)

 for conf_thresh = 0.25, precision = 0.78, recall = 0.55, F1-score = 0.65
 for conf_thresh = 0.25, TP = 1521, FP = 435, FN = 1234, average IoU = 60.83 %

 IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
 mean average precision (mAP@0.50) = 0.737467, or 73.75 %
Total Detection Time: 317 Seconds
```

Step 6 Test your model on test images and for that I have used opencv-python module.

Given Below are the results for real time web stream


Cat : 0.721292

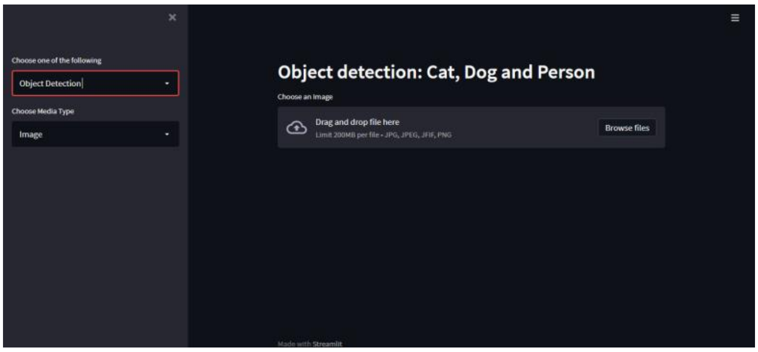

Cat : 0.811718
Dog : 0.819259



Step 7: Model Deployment

I've deployment my model on python's streamlit app. Below image shows the UI for the app.

a.) To browse images



b.) To browse videos