BABEŞ BOLYAI UNIVERSITY, CLUJ NAPOCA, ROMÂNIA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

PROJECT FARADAY

– MIRPR report –

Team members
Anton Mihai, Info Ro, 231
Avram Vasile-Cosmin, Info Ro, 231
Badita Marin-Georgian, Info Ro, 231

2019

January 14, 2020

1 Introduction

A project that will bring safety down the streets of every city

1.1 Project Purpose and Scope

Traffic participants will benefit from an application that will make their driver's life easier and will increase the degree of safety on the roads around the globe.

The application will successfully detect in real time obstacles such as pedestrian and other cars. It will recognize signs and traffic markings and it will be able to detect the current weather and road conditions. Furthermore, it will be able to detect pedestrians that could create harm, by moving anomalous compared to the vast majority.

1.2 Proposed Solution

Our solution contains a combination of YOLOv3 and DeepSort algorithms for an accurate and fast object detection, combined with anomaly detection algorithms that are able to capture movements that might represent a danger for drivers.

2 Object Detection

Object detection is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos. Well-researched domains of object detection include face detection and pedestrian detection. Object detection has applications in many areas of computer vision, including image retrieval and video surveillance.

It is widely used in computer vision tasks such as face detection, face recognition, video object co-segmentation. It is also used in tracking objects, for example tracking a ball during a football match, tracking movement of a cricket bat, or tracking a person in a video.

2.1 Concept

Every object class has its own special features that helps in classifying the class – for example all circles are round. Object class detection uses these special features. For example, when looking for circles, objects that are at a particular

distance from a point (i.e. the center) are sought. Similarly, when looking for squares, objects that are perpendicular at corners and have equal side lengths are needed. A similar approach is used for face identification where eyes, nose, and lips can be found and features like skin color and distance between eyes can be found.

2.2 Methods

Methods for object detection generally fall into either machine learning-based approaches or deep learning-based approaches. For Machine Learning approaches, it becomes necessary to first define features using one of the methods below, then using a technique such as support vector machine (SVM) to do the classification. On the other hand, deep learning techniques are able to do end-to-end object detection without specifically defining features, and are typically based on convolutional neural networks (CNN).

1. Machine Learning approaches:
    - Viola–Jones object detection framework based on Haar features
    - Scale-invariant feature transform(SIFT)
    - Histogram of oriented gradients(HOG) features
2. Deep Learning approaches:
    - Region Proposals (R-CNN, Fast R-CNN, Faster R-CNN)
    - Single Shot MultiBox Detector (SSD)
    - **You Only Look Once (YOLO)**

2.3 How it works

3 Tools/Frameworks used

For this project we will use Python as a programming language together with some helpful libraries.

- Flask
- OpenCV
- PyTorch
- Numpy
- Keras
- Sklearn
- TensorFlow

3.1 Flask

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries.It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.

3.2 OpenCV

OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. In simple language it is library used for Image Processing. It is mainly used to do all the operation related to Images.

What it can do:

1. Read and Write Images
2. Detection of faces and its features
3. Detection of shapes like circle, rectangle, etc. in a image. E.g. Detection of a coin in images.
4. Text recognition in images. e.g Reading number plates.
5. Modifying image quality and colors
6. Developing Augmented Reality apps.

OpenCV supports the deep learning frameworks TensorFlow, Torch/PyTorch and Caffe.

3.3 PyTorch

PyTorch is an open source machine learning library based on the Torch library, used for applications such as computer vision and natural language processing. It is primarily developed by Facebook's artificial intelligence research group. It is free and open-source software released under the Modified BSD license.

PyTorch provides two high-level features:

- Tensor computing (like NumPy) with strong acceleration via graphics processing units (GPU)
- Deep neural networks built on a tape-based autodiff system

PyTorch Tensors

Tensors, while from mathematics, are different in programming, where they can be treated as multidimensional array data structures (arrays). Tensors in PyTorch are similar to NumPy arrays, but can also be operated on a CUDA-capable Nvidia GPU. PyTorch supports various types of tensors.

3.4 Numpy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents. NumPy addresses the slowness problem partly by providing multidimensional arrays and functions and operators that operate efficiently on arrays, requiring rewriting some code, mostly inner loops using NumPy.

Python bindings of the widely used computer vision library OpenCV utilize NumPy arrays to store and operate on data. Since images with multiple channels are simply represented as three-dimensional arrays, indexing, slicing or masking with other arrays are very efficient ways to access specific pixels of an image. The NumPy array as universal data structure in OpenCV for images, extracted feature points, filter kernels and many more vastly simplifies the programming workflow and debugging.

3.5 Keras

Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, R, Theano, or PlaidML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. It was developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System), and its primary author and maintainer is François Chollet, a Google engineer. Chollet also is the author of the XCeption deep neural network model.

Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing Deep Neural Network code. The code is hosted on GitHub, and community support forums include the GitHub issues page, and a Slack channel.

In addition to standard neural networks, Keras has support for convolutional and recurrent neural networks. It supports other common utility layers like dropout, batch normalization, and pooling.

Keras allows users to productize deep models on smartphones (iOS and Android), on the web, or on the Java Virtual Machine. It also allows use of distributed training of deep-learning models on clusters of Graphics Processing Units (GPU) and Tensor processing units (TPU) principally in conjunction with CUDA.

3.6 TensorFlow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache License 2.0 on November 9, 2015.

Among the applications for which TensorFlow is the foundation, are automated image-captioning software, such as DeepDream. RankBrain now handles a substantial number of search queries, replacing and supplementing traditional static algorithm-based search results.

3.7 YOLO: Real-Time Object Detection

You only look once (YOLO) is a state-of-the-art, real-time object detection system.

YOLO is popular because it achieves high accuracy while also being able to run in real-time. The algorithm "only looks once" at the image in the sense that it requires only one forward propagation pass through the neural network to make predictions. After non-max suppression (which makes sure the object detection algorithm only detects each object once), it then outputs recognized objects together with the bounding boxes.

With YOLO, a single CNN simultaneously predicts multiple bounding boxes and class probabilities for those boxes. YOLO trains on full images and directly optimizes detection performance. This model has a number of benefits over other object detection methods:

- YOLO is extremely fast
- YOLO sees the entire image during training and test time so it implicitly encodes contextual information about classes as well as their appearance.
- YOLO learns generalizable representations of objects so that when trained on natural images and tested on artwork, the algorithm outperforms other top detection methods.

Further research was conducted resulting in the December 2016 paper "YOLO9000: Better, Faster, Stronger," by Redmon and Farhadi, both from the University of Washington, that provided a number of improvements to the YOLO detection method including the detection of over 9,000 object categories by jointly optimizing detection and classification.

Even more recently, the same researchers wrote another paper in April 2018 on their progress with evolving YOLO even further, "YOLOv3: An Incremental Improvement".

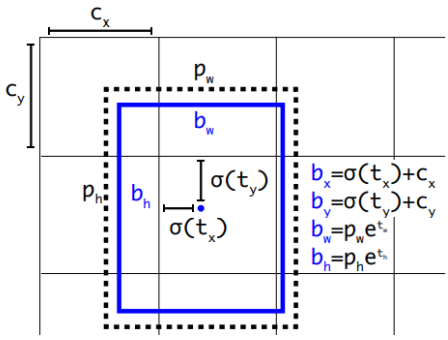

Figure 1: Bounding boxes with dimension priors and location prediction. We predict the width and height of the box as offsets from cluster centroids. We predict the center coordinates of the box relative to the location of filter application using a sigmoid function.

Class Prediction

Each box predicts the classes the bounding box may contain using multilabel classification. We do not use a softmax as we have found it is unnecessary for good performance, instead we simply use independent logistic classifiers. During training we use binary cross-entropy loss for the class predictions

This formulation helps when we move to more complex domains like the Open Images Dataset. In this dataset there are many overlapping labels (i.e. Woman and Person). Using a softmax imposes the assumption that each box has exactly one class which is often not the case. A multilabel approach better models the data.

YOLOv3 predicts boxes at 3 different scales. Our system extracts features from those scales using a similar concept to feature pyramid networks Feature Pyramid Networks For Object Detection. From our base feature extractor we add several convolutional layers. The last of these predicts a 3-d tensor encoding bounding box, objectness, and class predictions. In our experiments with COCO Microsoft COCO: Common Objects in Context we predict 3 boxes at each scale so the tensor is N x N x [3 * (4 + 1 + 80)] for the 4 bounding box offsets, 1 objectness prediction, and 80 class predictions

Each network is trained with identical settings and tested at 256x256, single crop accuracy. Run times are measured on a Titan X at 256 x 256. Thus Darknet-53 performs on par with state-of-the-art classifiers but with fewer floating point operations and more speed. Darknet-53 is better than ResNet-101 and 1.5x faster. Darknet-53 has similar performance to ResNet-152 and is 2x faster. Darknet-53 also achieves the highest measured floating point operations per second. This means the network structure better utilizes the GPU, making it more efficient to evaluate and thus faster. That's mostly because ResNets have just way too many layers and aren't very efficient.

Backbone	Top-1	Top-5	Bn Ops	BFLOP/s	FPS
Darknet-19 [15]	74.1	91.8	7.29	1246	**171**
ResNet-101[5]	77.1	93.7	19.7	1039	53
ResNet-152 [5]	**77.6**	**93.8**	29.4	1090	37
Darknet-53	77.2	**93.8**	18.7	**1457**	78

Figure 2: Comparison of backbones. Accuracy, billions of operations, billion floating point operations per second, and FPS for various networks.

Training

4 Project steps

The aim of the project is building a pipeline that is able to detect anomalies in pedestrian movement. For this, a 3 step pipeline is proposed. Firstly, the pedestrian and the context (other cars, traffic signs, etc) need to be detected. Next, the movement of each individual pedestrian needs to be recorded as precise as possible. For this, a tracking algorithms, namely DeepSort is used. Finally, statistical methods have to be applied over pedestrian movement patterns, in order to detect which ones are the different from the majority.

1. Pedestrian recognition
   The approach is to use YOLOv3 for object recognition, retrained in order to better detect pedestrians in coparison to other objects.
2. Pedestrian tracking
   For this, DeepSort is going to be used. To make it clear before getting into detail, the 'SORT' in the name stands for Simple Online and Real-Time Tracking.
3. Action prediction
   Having already recorded the pattern of movement for each tracked object, clustering or anomaly detection techniques can be used in order to determine which are the pedestrians that might be dangerous for traffic.

4.1 Pedestrian recognition

Available methods

The standard computer vision techniques involve scanning the image with a predefined sliding window while performing prediction over each step. Since there are so many steps to be done, this is obviously a slow technique, far from real time demands.

Over the time, various algorithms have been developed, like R-CNN and it's faster versions, but no one compares to YOLO, which has been on an ascending popularity slope since 2016. Using a proper GPU, this method can be used in

real-time multi object detection, since, as the name states, it only looks at the image once, compared to a few dozen steps other algorithms take.

Ups and downs of each method
When it comes to analysing images, time and performance constraints re the firs ones that matter. Working with different techniques, we have observed that when fast prediction is needed, powerful hardware has to be used, while on the other side, slow video prediction worked on almost every device configuration.But performance of production ready algorithms (known also as inference time) isn't the only concern when working with this kind of methods, but also the training time is very resource consuming. One of the biggest challenged that we have encountered is the complicated training steps of Yolo. Not only it needed a big amount of labeled data, but also the hardware it ran on made a huge difference in training times.

Method used
After experimenting with various open source algorithms, we have started working with a pre-trained Yolo network, using the darknet infrastructure. After a few tests on the 'out of the box' weights, we have gone towards a retraining approach. We have modified the classes in order for pedestrians to have a greater bias when training with various objects. The reason of increasing the importance of pedestrian objects withing our networks is because of the nature of the problem we are trying to solve, namely pedestrian action prediction.

4.2 Pedestrian tracking

DeepSort
After a successful pedestrian detection, each person in the image has to be tracked individually in order for an algorithm to learn the pattern of his movement. DeepSort , which basically means using deep learning techniques for Simple Online and Real-time tracking for object follows in the pipeline. Our aim is to assign a unique id for each pedestrian in order to further track it's movement.

How it works
The way this algorithm works is simple on a high level. First it assigns an unique id to every bounding box in the image. Then, on each 2 consecutive frames it computes the centroids for each bounding box, then for a controid C in the first frame it searches for the closest centroid in the second frame. Afterwards it links each 2 closest centroids and this is how tracking is performed.

As with this technique objects might get wrong id's when too close to each other, the author of this algorithm introduced an improvement to the base technique, namely Kalman Filter — Kalman Filter is an improvement over simple centroid based tracking. It models future position and velocity using gaussians. When it receives a new reading it can use probability to assign the measurement to its prediction and update itself. It is light in memory and fast to run. And since it uses both position and velocity of motion, it has better results than the centroid based tracking.

Drawbacks

This algorithm generally works decent, with a few exceptions. If the bounding boxes in one image are too big, then it reduces the effectiveness of the algorithm because it captures too much of the background. Secondly, when people are dressed similarly, for example in sports, it becomes hard to make a difference between objects.

4.3 Action prediction

Our aim is to detect movement patterns that stand out from the crowd. For example, if people are normally walking along the sidewalk and one pedestrian is walking in a perpendicular direction, towards the road, this may be a safety issue and we would like our algorithm to detect it.

Out approach is to use some anomaly detection method in order to detect the anomaly cluster of people from the whole crowd. In other words, we want to detect those pedestrians that do not respect the general moving pattern. Out proposal is to use LSTM networks and autoencoders. The LSTMs will learn from videos without anomalies, while trying to reconstruct the near future. Here the problem splits into two branches. If what we feed to our trained network is a sequence of normal moving people, it will be able to accurately reconstruct the near future, being almost no difference between the prediction and the ground truth. On the other side, when we feed an anomaly to the network, there will be significant differences between what is predicted and what actually is recorded.

As the previously presented method learns from videos, namely sequences of pictures, a different network can learn from movement vectors, where each recognised pedestrian has it's own list of positions. Whenever a new object has a movement vector different from what is considered to be normal, it will be signaled as an anomaly.

5 Improvements

5.1 Retraining with custom classes

While experimenting with the darknet framework, we have also tried the "Hello world" of object recognition, namely trying to recognise cats and dogs in custom pictures. We have labeled our own dataset and trained overnight with that pictures. Since yolo has proven to be hard to train from scratch, the results we have received were not the most accurate, due to lack of powerful training hardware.

5.2 Retraining for better pedestrian detection

After the 'not so successful' approach presented above, we have gone towards a transfer learning approach, where we have downloaded a pretrained set of weights, which we have further trained with a higher bias towards pedestrians.

5.3 Metrics

Intersection over union: IoU is an evaluation metric used to measure the accuracy of an object detector on a particular dataset. This metric is often met in object detection algorithms like the ones named above. On a high level, it tells us how far one bounding box is compared to the ground truth. In other words,the metric measures the degree of overlapping between the prediction and the reality.

6 Problems encountered

Long training times

One of the biggest problems we have encountered during the project was the overhead presented by training the model. Not only was it hard due to operating system requirements and hard to setup frameworks but also due to long training times due to lack of powerful hardware.

Lack of accurate resources

Since the methods described above, namely Yolo and DeepSort, are fairly new techniques, research on them starting in 2016, the online resources to be found are scarce, thus taking a long time to get informed.

7 Results/Tests

Since we have not been able to complete the whole project, we have conducted experiments on the first two steps, namely Yolo and DeepSort. We have retrained the models for the first one and we have a working prototype for object recognition and tracking.

- https://pjreddie.com/media/files/papers/YOLOv3.pdf
- https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/
- https://towardsdatascience.com/people-tracking-using-deep-learning-5c90d43774be