Machine Learning Engineer Nanodegree
Capstone Project Report
Cassandra Carr Sugerman
August 19, 2018

I. DEFINITION

Project Overview

Object detection is an evolving area in the field of machine learning and computer vision. This project seeks to utilize a known object detection algorithm and fine tune the model for a new dataset. Through this implementation, a deeper understanding of convolutional neural networks (CNNs) and computer vision, as applied to object detection, seeks to be established.

The domain of object detection combines localization and classification through the concept of searching an image for a variable number of objects, identifying the objects, and then classifying the objects. There have been several advancements in the area of object detection between 2012 and today. Originally object detection was completed with exhaustive search, where a sliding window algorithm was used to search the entire image for objects, resulting in high computation times (Rey, 2017). In 2014, R-CNN was developed, which used selective search (Uijlings, 2012) as an improvement to exhaustive search. Selective search used grouping methods to produce a series of object proposals. R-CNN took the object proposals and input them into a CNN which output probabilities for each object class (Ouaknine, 2018). Due to selective search being computationally expensive, a few years later, Fast R-CNN and Faster R-CNN were developed to improve selective search and start leading progress towards real time object detection. Faster R-CNN replaced selective search with a Region Proposal Network (RPN) which generates region proposals within the CNN, see Figure 1 below (Shaoqing, 2016).

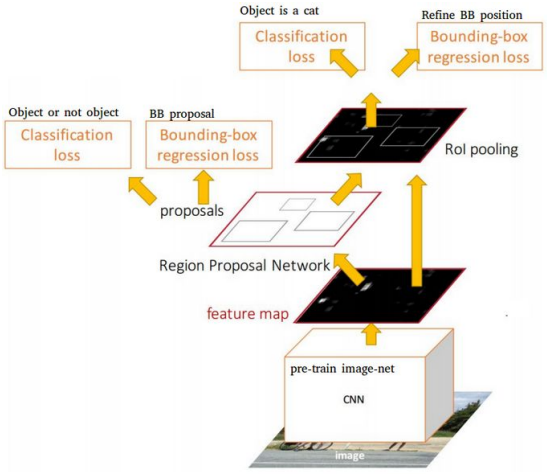


Figure 1: Faster R-CNN architecture

Following Faster R-CNN, other methods were studied to improve speed and accuracy of object detection. One worth noting is YOLO (You only look once), which simplified the model, allowing

for real time object detection, but at the expense of accuracy. The model divides the image into a SxS grid and identifies B bounding boxes for each cell of the grid, with each box having a confidence value. Each cell of the grid is also assigned a class probability. Non-maximum suppression is then used to combine multiple bounding boxes which are identifying the same object (Ouaknine, 2018 & Redmon, 2016).

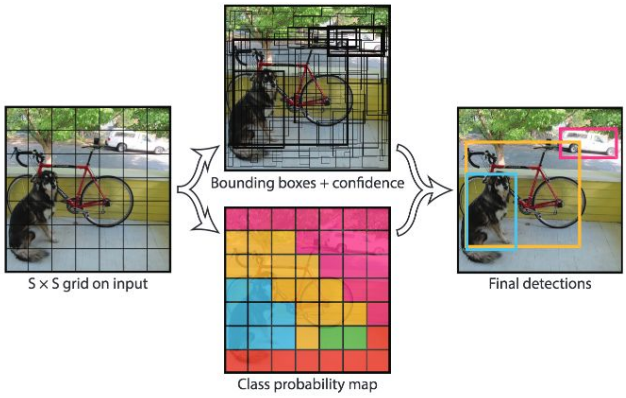


Figure 2: YOLO architecture

Work is still being done in the area of object detection to further improve speed and accuracy. This project aims to take the YOLO architecture and utilize it for a single class, people, in order to see if accuracy can be improved. The classification and detection of people in a video stream is useful in applications such as surveillance and autonomous driving. If successful for one use case, the architecture could then be applied to other objects.

Resources:

Arthur Ouaknine. “Review of Deep Learning Algorithms for Object Detection”. Feb 5, 2018. https://medium.com/comet-app/review-of-deep-learning-algorithms-for-object-detection-c1f3d437b852

Javier Rey. “Object detection: an overview in the age of Deep Learning”. Aug 30, 2017. https://tryolabs.com/blog/2017/08/30/object-detection-an-overview-in-the-age-of-deep-learning/

Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi. “You Only Look Once: Unified, Real-Time Object Detection”. May 9, 2016. https://arxiv.org/pdf/1506.02640.pdf

J.R.R. Uijlings, K.E.A. van de Sande, T. Gevers, and A.W.M. Smeulders. “Selective Search for Object Recognition”. 2012. http://www.huppelen.nl/publications/selectiveSearchDraft.pdf

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. Jan 6, 2016. https://arxiv.org/pdf/1506.01497.pdf

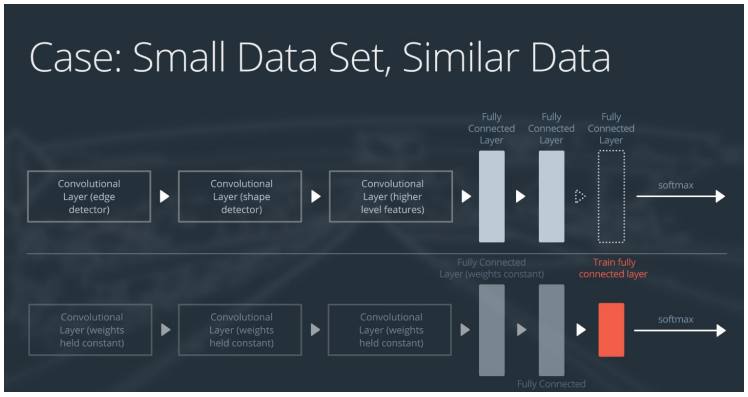
Problem Statement

As described above, accurate and fast object detection is a domain which is continually being improved due to the many machine learning applications where it is applied. These areas encapsulate autonomous driving, video surveillance, and much more. This machine learning capstone project seeks to use object detection to identify people in a video stream. The goal is to develop an understanding of the current work being done in the field and implement the YOLO architecture on a specific problem, people detection. YOLO was documented with a frames per second (FPS) speed of 155 and mean average precision (mAP) of 52.7%. The goal of this project is to improve accuracy when implementing on one class, people, and show that the model can be retrained for a variety of different classes and / or objects.

The dataset used for training was originally noted to be the INRIA Person Dataset (http://pascal.inrialpes.fr/data/human/). This dataset is made up of positive images, which include people, and negative images without people. For all the positive images, there is an associated text file with annotations in Pascal Challenge Format which indicates the bounding box for each person in the image. Unfortunately, the Pascal Challenge Format was found to not be common and it was determined that using Pascal VOC 2007 data (https://pjreddie.com/projects/pascal-voc-dataset-mirror/) would be easier to use initally, as the format and dataset has been widely used.

After the model was successfully trained and tested on the Pascal VOC 2007 dataset (only images labeled with “people” were used), video data was used to test if the model could detect people in a video stream. The stock videos used can be found here:
https://pixabay.com/en/videos/lake-park-people-walking-summer-11571/
https://pixabay.com/en/videos/city-skyscraper-bokeh-blur-6388/
https://pixabay.com/en/videos/street-walk-urban-city-walking-5025/

The model retraining process follows the transfer learning case where we have a small data set with similar data. The pretrained YOLO model weights were loaded, the last fully connected layer removed, and a new fully connected layer was added to create a new model. The model was then retrained, keeping the weights associated with the other layers constant. (Udacity Machine Learning Module & Chu, 2017)



Resources:
Udacity Machine Learning Module: Lesson 2: Convolutional Neural Networks.

Greg Chu. “How to use transfer learning and fine-tuning in Keras and Tensorflow to build an image recognition system and classify (almost) any object”. May 1, 2017. https://deeplearningsandbox.com/how-to-use-transfer-learning-and-fine-tuning-in-keras-and-tensorflow-to-build-an-image-recognition-94b0b02444f2

Navneet Dalal and Bill Triggs. “Histograms of Oriented Gradients for Human Detection”. Dec 20, 2010. https://hal.inria.fr/file/index/docid/548512/filename/hog_cvpr2005.pdf

Metrics

The solution for this project was developing an object detection algorithm for finding and detecting people in a video stream. The model developed will take images from a video stream and output bounding box(es) of all people in the images, then overlay the bounding boxes, generating a new video stream. The classification and localization used to identify the bounding boxes was taught at once through a deep neural network and evaluated with the metric mAP (mean average precision).

II.ANALYSIS

Data Exploration

It was determined that images containing people from the Pascal VOC 2007 dataset would be used for training. The dataset contains 1070 images for training and 1025 images for validation containing people. Below is an example of the images found in the dataset.

Exploratory Visualization

For each image in the Pascal VOC 2007 dataset, there is an annotation for that image which contains the bounding box information for all items in that image. An example of this file is below. The orange text shows information about the file, including the file name and size of the image. The purple text shows the bounding box for each labeled item in the image. This example has one object, labeled “person”, but there could be a number of different objects with the same or different labels.

```
<annotation>
        <folder>VOC2007</folder>
        <filename>000229.jpg</filename>
        <source>
                <database>The VOC2007 Database</database>
                <annotation>PASCAL VOC2007</annotation>
                <image>flickr</image>
                <flickrid>337948768</flickrid>
        </source>
        <owner>
                <flickrid>mangeek</flickrid>
                <name>Marc Doughty</name>
        </owner>
        <size>
                <width>500</width>
                <height>375</height>
                <depth>3</depth>
        </size>
        <segmented>0</segmented>
        <object>
                <name>person</name>
                <pose>Unspecified</pose>
                <truncated>1</truncated>
                <difficult>0</difficult>
                <bndbox>
                        <xmin>399</xmin>
                        <ymin>219</ymin>
                        <xmax>486</xmax>
                        <ymax>371</ymax>
                </bndbox>
        </object>
</annotation>
```

Algorithms and Techniques

The algorithm used in this project follows the version 2 YOLO algorithm (Redmon). The YOLO model is one of the leading object detection algorithms and has been frequently used to achieve positive and fast results for object detection of various classes.

The YOLO model consists of 23 convolutional layers, and for this project, pretrained weights were loaded from the original model. For the first 22 layers, the weights were kept frozen, while the last layer was randomized and retrained.

Joseph Redmon, Ali Farhadi.”YOLO9000: Better, Faster, Stronger.” https://pjreddie.com/media/files/papers/YOLO9000.pdf

Benchmark

As mentioned above, the benchmark model used for this project was the YOLO version 2 model. The goal was to take the same deep neural network used for object detection and fine tune the model for people identification. After fine tuning, mAP (mean average precision) was evaluated in an attempt to improve upon what was achieved with the YOLO model, 52.7%. Since FPS (frames per second) was difficult to optimize due to computing power available, this will not be evaluated against the original YOLO metrics.

III.METHODOLOGY

Data Preprocessing

First, it is important to note that the keras-yolo2 repository was used to lay the groundwork for this project (https://github.com/experiencor/keras-yolo2). It was a similar project that used keras to recreate the yolo model and retrain the model for a new dataset.

For data preprocessing, the images were first normalized by dividing the pixel arrays by 255. The images and associated annotations (xml files) were then appropriately parsed. A batch generator was then used during training. The batch generator first applied an augmentation algorithm to each image in the batch, which performed random scaling, translation, and flip to the image. The object information, parsed from the annotations, were then looped through in order to get the bounding box information for each object labeled in the image. The image and true bounding boxes are then passed to the model.

Implementation

The model design for this application is based on the YOLOv2 deep learning model. For this project, the model was built in Keras and precisely followed the YOLOv2 architecture, which consists of 23 convolutional layers. The pretrained YOLO model weights were loaded into the model and the weights for the first 22 layers were kept frozen. The 23rd layers was changed to output the correct size based on the new number of classes (in this case there is only one class, people). The weights for the 23rd layer were then randomized and the model was retrained.

Refinement

The model was trained with various different parameters until an acceptable model was found. First, a various number of epochs were trialed, eventually determining that 16 was optimal. Then, parameters NO_OBJECT_SCALE, COORD_SCALE, and CLASS_SCALE were adjusted to see their effects; as the model was producing many false positives. These parameters affect how much to penalize wrong predictions of non-objects, how much to penalize wrong position and size predictions, and how much to penalize wrong class predictions. Although adjusting these factors helped in the false positives, many still remained. It was changing the optimizer from Adam to RMSprop that led to a functional model.

IV.RESULTS

Model Evaluation and Validation

Overall, the model developed performed well on the trial images and videos containing people. The mean average precision (mAP) was calculated on the validation set and was found to be 65.96%.

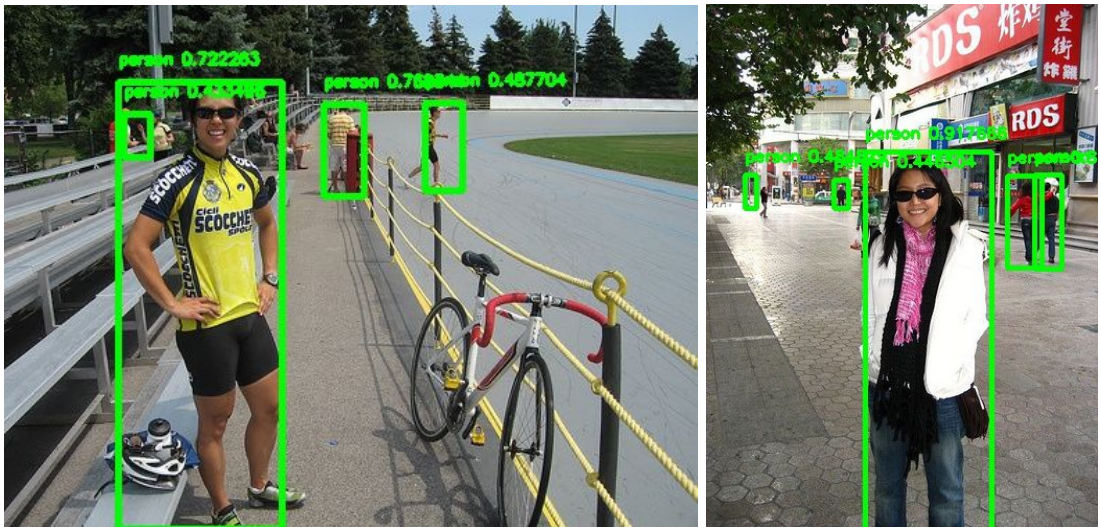
Justification

As noted above, the model developed in the project to identify people showed a mean average precision of 65.96% on the validation set. This is an improvement over the original YOLO model, which achieved 52.7%. It is important to note that the YOLO model identified multiple classes, where in this case, only one class was trained. However, overall, we can state that the model is satisfactory compared to the benchmark.

V.CONCLUSION

Free-Form Visualization

The model developed in this project was first tested on a series of images containing people and it was found to identify people in the images accurately. See below for examples.



The model was also used to find people in multiple video streams. One of the results can be found here: https://youtu.be/6I3_GQn-evQ.

Reflection

This project aimed to develop an object detection algorithm for identifying people in a video stream. Based on object detection research, it was determined that the YOLO model would be a good candidate for this application. The YOLO deep neural network was recreated and weights from the original model were loaded. The last layer of the model was then modified, weights randomized, and the model was retrained on a new dataset containing only people. Originally the model gave too many false positives, but after refinement, the model proved to be successful in identifying people in a video stream.

During the project, a number of barriers were overcome. It was concluded that keras would be an acceptable platform to recreate and run the model, so a benchmark repository was found which used keras to recreate the YOLOv2 model (https://github.com/experiencor/keras-yolo2). This code was the baseline for this project and was modified for the dataset at hand. This was the first major challenge faced, as many iterations were performed before a working model was established. This ended up being a good learning experience because the code was deep dived and better understood. After the code was working and giving results, there was still fine tuning that needed to occur in order to develop an accurate model. This was the second major challenge faced. AWS (Amazon Web Services) was used to speed up the training process and allow for an appropriate number of iterations, however, training time was still very long. Multiple parameters were changed and eventually an accurate model was developed.

Improvement

There is still room for improvement to be made on this model. However, due to the long training time, it was difficult to do a lot of experimenting. It would be beneficial to change the model to the YOLO tiny model and see if similar, or even better, results can be accomplished. The tiny YOLO model has fewer layers and would take less time to train. In addition, improvement could be made to the model by looking at other optimizers, increasing the number of epochs, adding more pre-processing, adding dropout, and changing the optimizer parameters such as learning rate. In addition, it would be interesting to make adjustments to the code to easily train on a variety of other datasets and see how it performs at identifying different objects. The YOLO model has great potential for being a useful object detection algorithm in many instances.