Machine Learning Engineer Nanodegree
Capstone Proposal

Cassandra Carr Sugerman

July 4, 2018

Domain Background

The domain of object detection seeks to combine 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 was developed to improve selective search and start to 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).

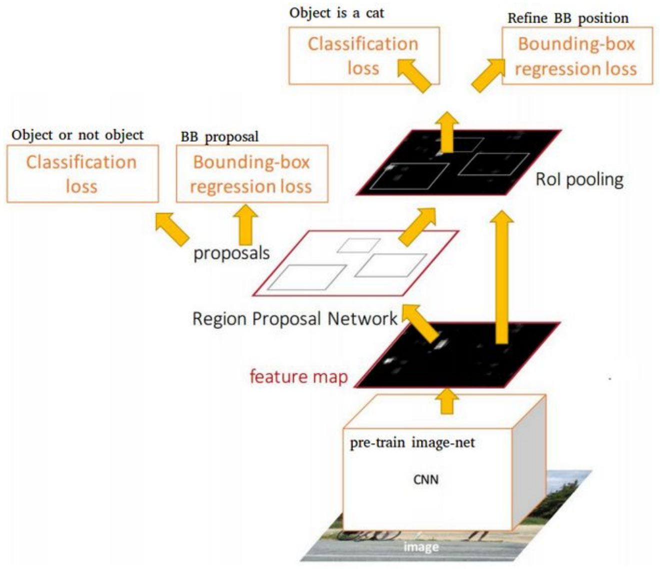


Figure 1: Faster R-CNN architecture

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).

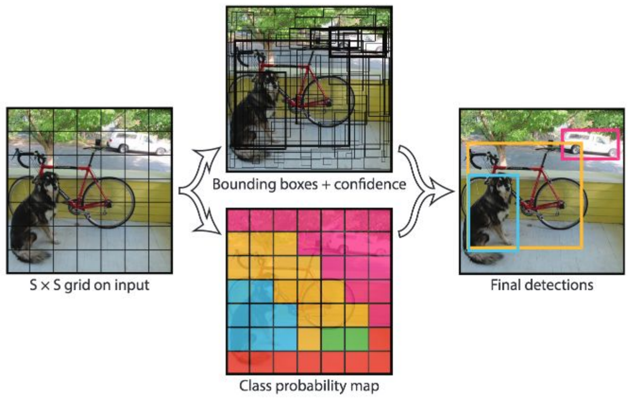


Figure 2: YOLO architecture

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.

Datasets and Inputs

The dataset used for training will 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. This dataset will be used to train a neural network mimicking the YOLO model. A portion of the dataset will be saved back for testing purposes. The dataset has about 600 positive images and 1200 negative images for training, and 300 positive images and 500 negative images for testing. The dataset seems widely varied and should produce generalized results. It will be a good starting place to train the YOLO model to identify people in various images.

After the model is successfully trained and tested on the INRIA Person Dataset, video data will be collected of people walking in high pedestrian areas and the model will be used to identify each person in the video stream. If additional data is found to be needed, it will be generated from the videos and used to fine tune the model.

Resources:
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

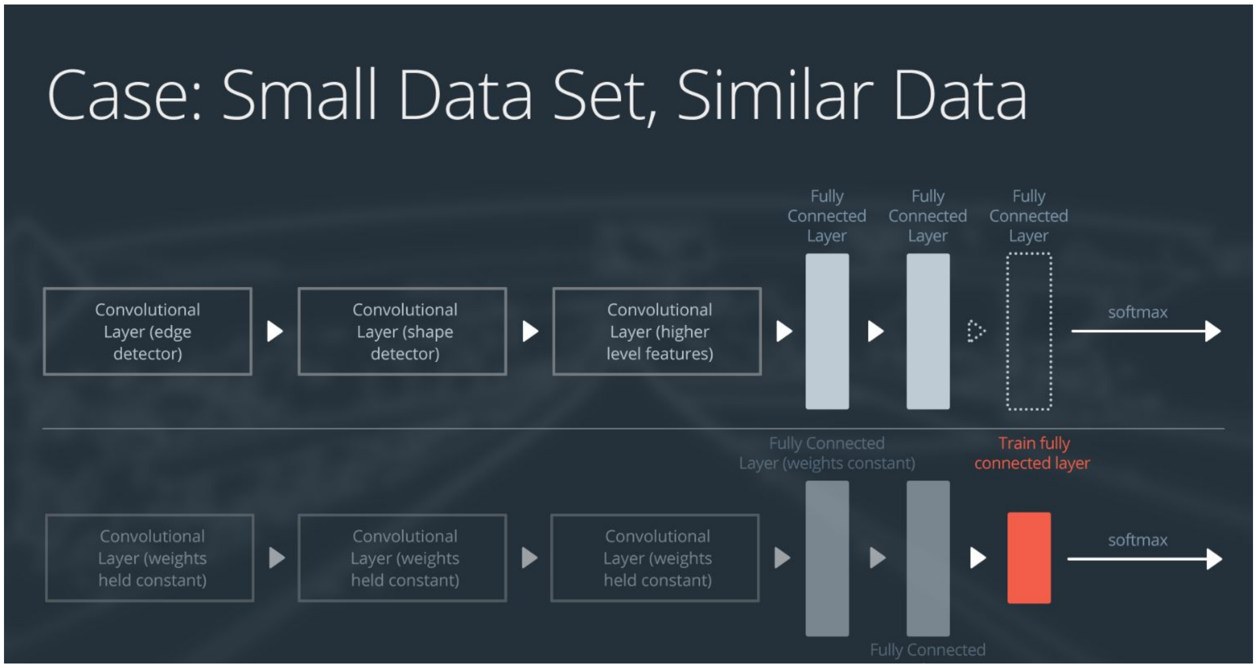
Solution Statement

The solution for this project is 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 the bounding box(es) of all people in the images, then piece together the bounding box images back together, generating a new video stream. The classification and localization used to identify the bounding boxes will be taught at once through a deep neural network. The model will be evaluated with the metric mAP.

Benchmark Model

The benchmark model for this project is the YOLO model. This model has been noted for its simplicity and ability to detect objects in close to real time. The goal is to take the same deep neural network used for object detection and fine tune the model for people identification. After fine tuning, FPS and mAP will be evaluated and an attempt will be made to improve upon what was achieved with the YOLO model.

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


Case: Small Data Set, Similar Data
Convolutional Layer (edge detector)
Convolutional Layer (shape detector)
Convolutional Layer (higher level features)
Fully Connected Layer
Fully Connected Layer
Fully Connected Layer
softmax
Fully Connected Layer (weights constant)
Train fully connected layer
Convolutional Layer (weights held constant)
Convolutional Layer (weights held constant)
Convolutional Layer (weights held constant)
softmax
Fully Connected

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

Evaluation Metrics

The evaluation metric for the model will be mAP (mean average precision). The goal will be to get similar or better results to the metrics used in evaluating the original YOLO model and show how the model can be used for new datasets and / or objects.

Project Design

Example used:
https://github.com/experiencor/keras-yolo2/blob/master/Yolo%20Step-by-Step.ipynb

Step 1: Preprocessing

- For both the training and validation datasets:
  - Process images in INRIA Person Dataset to be the correct size for training
  - Process annotations for each image of the INRIA Person Dataset to be the correct format for training
  - Save processed images and labels in pickle for ease of use during training

Step 2: Prepare the model for training

- Convert the YOLO weights and cfg files to a .h5 file to be used with Keras using the Keras-Yolo3 repository.
- Load the model and weights into Keras
  ```
  yolo_model = load_model(weights_path)
  ```
- Remove the last layer from the model and replace with a new layer for only 1 class (People)
  ```
  yolo_model.layers.pop()
  new_yolo_model = Sequential()
  new_yolo_model.add(yolo_model...)
  new_yolo_model.add(Conv2D(...))
  new_yolo_model.add(Activation('linear'))
  new_yolo_model.add(Reshape(...)
  ```
- Randomize weights of the last layer, freeze all other weights
  ```
  for layer in model.layers[:num_frozen_layers]:
        layer.trainable = False
  ```

Step 3: Train the new model

- Use Keras to train the new model with the preprocessed training dataset
  ```
  new_yolo_model.compile(...)
  new_yolo_model.fit(...)
  ```
- Fine tune the model to optimize mAP based on the validation dataset

Step 4: Test the newly trained model on new images

- Take some images of people and see how the bounding boxes perform

Step 5: Implement on video

- Take test video of pedestrians and overlay the bounding boxes from the new model