Machine Learning Project Report: Pedestrian Detection for Autonomous Driving Applications

Abstract

TODO: write a summary, emphasize the contributions

1. Introduction

TODO: split it up, rewrite

Very successful approach from 2005 used a feature representation called Histogram of Oriented Gradients (HOG) with linear classifier to detect pedestrians. In 2010 they introduced Deformable Parts Model (DPM) which allows deformity and replaced linear classifier with latent SVM to achieve a lot better results. They used a dynamic programming algorithm to do this fast. This was a state of the art solution until the era of deep learning using Convolutional Neural Networks (CNN). In 2015 they published a paper TODO: cite, where they argued that DPM is a kind of CNN, since it extracts edges in a similar way and uses histograms instead of pooling layers. Using CNN gave a lot better performance but it is too slow to use it for localization by sliding window approach. They introduced a region proposal method of guessing where to apply CNN. This region detector is looking for regions of interest (RoI) and returns bounding boxes around blob-like objects. Variations of this idea:

- R-CNN uses this idea directly, but results in a complicated training pipeline. It post-hoc trains binary SVM's to classify regions. It is also very slow at test time.
- Fast R-CNN basically combines tasks of R-CNN into a simpler pipeline using RoI pooling and minimizing log loss + smooth L1 loss. Problem is it is highly technical, it takes very long to train (10h on powerful GPUs on small Pascal dataset). Also it is still slow at test time approx 2sec per image, which still doesn't allow real time use. Aslo to demanding on memory, produces 200GB dataset when training.
- Faster R-CNN puts the whole thing in one giant network, replacing sliding window operation with a convolution layer. Messy training, implemented in Matlab or Caffe + Python. Achieves .2 sec per image.
- YOLO, directly predicts regression tensor, which makes it a lot faster than all other methods, but worse performance. Good when no access to powerful GPUs, could maybe work even on phones at reasonable Hz.
- Current state of the art solutions use mostly: ResNet (residual NN architecture) trained backbone for image feature extraction in Faster R-CNN pipeline and some extra crazy things to achieve high performance on particular dataset.

Datasets for image recognition and detection:

- Pascal - small 20 classes, about 2.5 objects per image
- ImageNet - 200 classes, half a million images, 1 object per image
- Coco - 80 classes, 120k images, more objects per image

For evaluation, they mostly use mean average precision (mAP) that uses intersection over union (Iou).

Backbone CNN (feature extractor network):

- AlexNet 2012 winner on ImageNet. Classic network. They also used data augmentation, such as deforming images and hallucinating other training data. Applied ensemble of NNs for 2% improvement.
- ZFNet
- VGGNet won the 2014 localization challenge! Very simple architecture. But deep, requires around 200MB of memory for backpropagation to work. Produces powerful image features.
- GoogleNet, ResNet - big improvements but crazy architectures and long training time 2-3 weeks on 8 powerful GPUs.

1.1. Related Work

CityPersons dataset [**?**], Elephant [**?**]

2. Our Approach

TODO: decribe Faster R-CNN and our modifications