OBJECT DETECTION IN TRAFFIC SCENARIOS - A COMPARISON OF TRADITIONAL AND DEEP LEARNING APPROACHES

Gopi Krishna Erabati, Nuno Gonçalves and Hélder Araújo

Institute of Systems and Robotics, University of Coimbra, Portugal

ABSTRACT

In the area of computer vision, research on object detection algorithms has grown rapidly as it is the fundamental step for automation, specifically for self-driving vehicles. This work presents a comparison of traditional and deep learning approaches for the task of object detection in traffic scenarios. The handcrafted feature descriptor like Histogram of oriented Gradients (HOG) with a linear Support Vector Machine (SVM) classifier is compared with deep learning approaches like Single Shot Detector (SSD) and You Only Look Once (YOLO), in terms of mean Average Precision (mAP) and processing speed. SSD algorithm is implemented with different backbone architectures like VGG16, MobileNetV2 and ResNeXt50, similarly YOLO algorithm with MobileNetV1 and ResNet50, to compare the performance of the approaches. The training and inference is performed on PASCAL VOC 2007 and 2012 training, and PASCAL VOC 2007 test data respectively. We consider five classes relevant for traffic scenarios, namely, bicycle, bus, car, motorbike and person for the calculation of mAP. Both qualitative and quantitative results are presented for comparison. For the task of object detection, the deep learning approaches outperform the traditional approach both in accuracy and speed. This is achieved at the cost of requiring large amount of data, high computation power and time to train a deep learning approach.

KEYWORDS

Object Detection, Deep Learning, SVM, SSD & YOLO

1. INTRODUCTION

Being inspired from nature or human physiology, the science and technology are progressing rapidly to leverage human sophistication and well being. The tools or technology that is being developed is an extension of human faculties, for instance, as humans have the ability of locomotion, there arose the development of vehicles like cars or bikes for faster locomotion. In the past few decades, the automobile industry, research units and academia is focusing on automation of vehicles (self-driving vehicles). One of the significant reasons for automation of vehicles is to trim down the accidents caused by human drivers. Distraction [1] (due to texting, tiredness etc.), speeding [2], drunk driving [3], recklessness are some of the instances the human driver would create problems in traffic environments. The consequences of these actions may vary from damage to property to loss of life. On an average, every minute at least one person dies in a vehicle accident. Not only loss of life but also on an average 10 million people are injured each year. The hospital costs, damage to property and other costs will sum up to 1-3 % of world's gross domestic product [4].

During this era of automation, scientific and technical community has been providing various solutions to these problems. Pre-crash systems is an active area of research to reduce accident severity and injury. The major threat a driver faces in a traffic scenario is from other vehicles. Advanced Driver Assistance Systems (ADAS) [5] which warn the driver about possible

David C. Wyld et al. (Eds): ICAITA, CDKP, SAI, NCO, CMC, SOFT, MLT, AdNLP - 2020
pp. 225-237, 2020. CS & IT - CSCP 2020 DOI: 10.5121/csit.2020.100918

collision with other vehicles has gained lot of attention in research. Robust, reliable and fast obstacle detection is very significant and fundamental step in such systems.

In traffic scenarios, obstacle/object detection is the most important task for autonomous driving, since the relevant scene understanding is essential for driving controls. In computer vision literature, object detection is a twofold process, to classify which category object belongs to (object classification) and to determine the location of object in the scene (object localization). In traffic scenarios, objects include cars, bikes, trucks, buses, pedestrians etc. The object detection algorithms take input in the form of 2D data (like RGB images) and provides the output in the form of 2D bounding boxes which localizes the object along with the class of detected object.

The traditional object detection algorithms extract handcrafted features (like SIFT, SURF etc.) for semantic representation and use shallow networks or statistical pattern recognition to classify the object. On the other hand, deep learning [6] networks learn more complex features and semantic object representations which are leveraged by classifier networks.

This paper presents a comparison of traditional and deep learning approaches for object detection in traffic scenarios. A traditional object detection pipeline with HOG [10] features and SVM [13] classifier is compared with more advanced deep learning object detection approaches like SSD [33] and YOLO [32]. The SSD network is trained and tested with different backbone networks, such as VGG16 [20], MobileNetV2 [8] and ResNeXt50 [9], similarly YOLO with MobileNetV1 [39] and ResNet50 [21]. The three approaches are trained with PASCAL VOC 2007 and 2012 [7] dataset and tested with PASCAL VOC 2007 test set. Both qualitative and quantitative results are presented for comparison. The mAP for five classes, namely, bicycle, bus, car, motorbike and person is calculated for three approaches along with the processing speed.

This paper has the following structure: related work of object detection is presented in Section 2, an overview of compared methods is discussed in Section 3, training details are presented in Section 4 followed by results and discussion in Section 5, and finally conclusion in Section 6.

2. RELATED WORK

The topic of object detection has been widely researched in the last two decades. The aim of object detection algorithms using 2D data (RGB images) is to classify and locate the objects in the images, with 2D rectangular bounding boxes and also show their confidence of existence. This task is performed by the handcrafted traditional methods and new age deep learning methods. Depending on the methods to extract semantic features from the images to classify the objects, we divide the object detection methods into two: traditional methods and deep learning methods. The workflow of the two methods is shown in Figure 1.

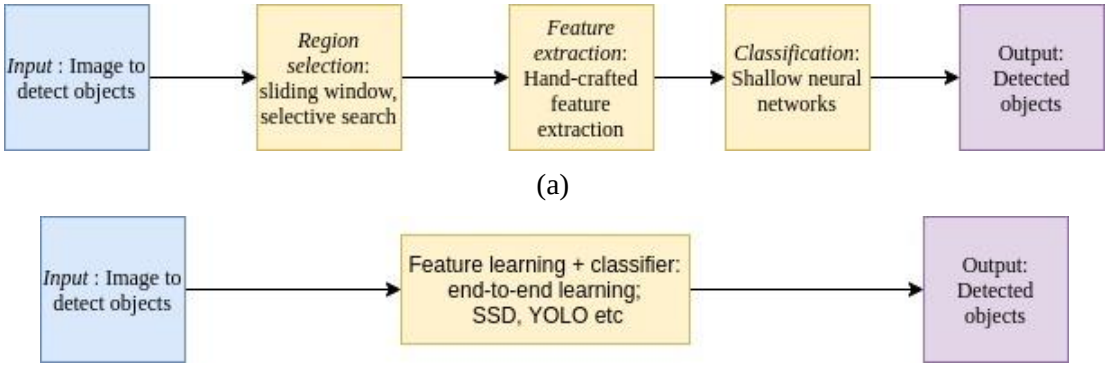

Figure 1. Workflow of (a) traditional methods and (b) deep learning methods

Features are compact, yet descriptive and distinctive attributes of the objects in the images, which are classified with classifiers. In traditional methods the features are handcrafted using established computer vision feature descriptors like SIFT [11], SURF [12] etc. The features are extracted as many as possible practically and form a definition known as bag of words for each object class. But there is always a difficulty to choose the priority of features to be extracted from the given data. So, it is up to the designer's judgement which features would best describe the different classes of objects in the data. The features are passed on to a shallow network classifier for classification of objects. In deep learning methods, the features are extracted by the network from the data, by discovering underlying patterns in the classes of data and automatically working best to provide compact, yet salient features for each class of object.

2.1 Traditional methods

The traditional object detection pipeline can be divided into three stages: Region selection, Feature description and Classification. *Region selection*: The different objects which are to be detected may appear at any position in the image and have variety of sizes or aspect ratios. So, it is a natural choice to search the entire image with multi-scale sliding windows for the region selection. This approach may result in finding all possible positions of objects, but it is very time consuming and computationally expensive due to a large number of candidate windows. However, if we apply only a limited number of template windows, the results would be unsatisfactory. This technique is known as sliding window technique to select Region of Interest (RoI). *Feature description*: Once the RoI is selected, we need to compute feature descriptors which provide robust and semantic representation of object. HOG [10], SIFT [11] and SURF [12] features are representative ones. However, due to the different appearance of various objects, illumination changes and occlusions, it is very difficult to manually design a feature extractor to perfectly represent all objects. *Classification*: Besides feature description, a classifier is required to distinguish an object class from other object classes. For instance, Support Vector Machine (SVM) [13] and Deformable Parts Model (DPM) [14] are some of the possible choices.

Papageorgiou *et al* [16] uses rectified Haar wavelets as feature descriptors along with SVM classifier for person detection. A direct approach of extracting edge images and matching them to a set of learned exemplars is given in [17]. Viola and Jones proposed boosted cascade of simple features to detect objects in [15]. The images are represented in a new representation called `integral image', which is a fast way of calculating Haar-like features. Ada-boost algorithm is used in choosing features and improving performance. This technique is used to build a strong classifier with a cascade of many weak classifiers.

2.2 Deep learning methods

With the advent of deep learning, some of the most difficult problems in computer vision, those for which no formal models exist, started to have realistic solutions. Among those problems, object detection is also being treated with deep learning techniques [6]. Deep learning models are neural networks with deep architecture. Neural networks are inspired by the neural structure of the human brain and aim at solving learning problems in a systematic way. The emergence of the ImageNet database [18], increase in computational capabilities with GPUs and important advances in design of networks and training strategies has led to the development of deep learning in present millennium. Dropout and data augmentation have helped to decrease the problem of over fitting. Batch normalization (BN) [19] allowed to realize and effectively train very deep neural networks. Meanwhile, various network structures like AlexNet [6], VGG [20], ResNet [21], GoogleLeNet [22] have been studied to improve the performance of the feature extraction networks.

The framework for object detection can be classified into three types: 1) Sliding window detectors with neural networks [23]. 2) Traditional object detection pipeline, generating region

proposals at first and then classifying each region into object category. The region proposal methods include R-CNN [24], Fast R-CNN [25], Faster R-CNN [26], R-FCN [27], FPN [28], Mask R-CNN [29]. 3) Object detection as a classification problem with unified architecture to both classify and localize the objects. The classification based one stage methods include YOLO [30], YOLOv2 [31], YOLOv3 [32], SSD [33], DSSD [34].

The framework for object detection can be mainly classified into two types. One follows a two stage process of proposing regions and classification and other follows a single stage process as a unified architecture to both classify and localize.

2.2.1 Two-stage methods

Girshick *et al* [24] adopts a selective search [35] method to propose 2k RoIs in the image. A Convolutional Neural Network (CNN) is run on top of the proposed regions to extract more semantic and high-level feature representation of RoIs. A pre-trained SVM classifier is used to discern among the classes of objects and a linear bounding box regression is used to obtain tighter bounding boxes for localising the objects. Although selective search is computationally more economical than brute-force search by sliding window, it is still costly, as selective search takes 2 seconds to propose 2k RoIs. It is very costly to run CNN on top of each of 2k regions. The inference time is about 40 - 50 seconds. The R-CNN [24] is improved by Fast-RCNN [25] by incorporating mainly two augmentations. One is performing the feature extraction over image by CNN before proposing regions, thus only running one CNN over entire image instead of 2k CNNs. The other is to replace SVM with softmax layer, thus extending the neural network for prediction instead of creating new model. Except for region proposals, all parameters in this architecture are optimized via a multi task loss in an end-to-end way. One bottleneck still remaining in Fast R-CNN is selective search for RoIs generation, which is costly. The inference time for this approach is 2 seconds. Faster-RCNN [26] replaces the slow region proposal method of selective search by an internal deep network called Region Proposal Network (RPN), to improve the speed, thus the name Faster R-CNN. RPN is achieved with a fully convolutional neural network (which shares layers with object detection network) which has the ability to predict object boundaries and scores at each position simultaneously. The concept of anchor boxes or default bounding boxes is presented in [26]. The inference time for this approach is 0.2 seconds.

2.2.2 Single-stage methods

Two stage frameworks comprise of region proposal generation and classification (feature extraction with CNN, classification and bounding box regression), which are usually trained separately. Even in the end-to-end Faster R-CNN framework an alternative training is required to obtain shared parameters between RPN and detection network. As a consequence, the framework remains as a bottleneck for real time applications. Single stage frameworks based on global classification/regression, can directly map from image pixels to bounding box coordinates and class probabilities, thus reduces the time expense and work in real time applications. Redmon *et al* [30] proposed a network that looks at the image only once to detect multiple objects, thus the name YOLO. This model has a unified network to perform classification and localization of objects at once, thus end-to-end training of network can be achieved. Liu *et al* [33] formulated a network that predicts and localizes the objects in single shot. Multi scale feature maps with anchor boxes are used to obtain the predictions of objects of different sizes and aspect ratios.

3. AN OVERVIEW OF COMPARED METHODS

This section provides an overview of traditional (HOG and SVM) and deep learning methods (SSD and YOLO) for object detection.

3.1 Object detection with HOG and SVM

The general pipeline in traditional object detection is: RoIs selection, feature extraction and classification. There are many feature descriptors like SIFT [11], SURF [12], Haar-like [15], HOG [10] which can be used to extract features for classification. Dalal and Triggs [10] proposed HOG feature descriptor which has gained wider recognition as a successful feature descriptor for object detection. The HOG features provide robust, reliable and high-level semantic representation of image regions.

3.1.1 Region selection - Selective search

Given an image to detect the objects, firstly we need to select the image windows with potential possibility of object, also called as RoIs. Sliding window technique is one of the brute force approach to select image patches for object detection. Selective search [35] is a region proposal algorithm used in object detection. Selective search starts with considering individual pixels as their own group. Next, the similarity measure based on colour, texture, size and shape compatibility is calculated for each group and two closer ones are combined. The merging of the regions is continued until everything is combined. This method is designed to be faster than sliding window technique with a very high recall. The method of selective search is used for region proposal.

3.1.2 Feature extraction - modified HOG

HOG is a gradient based feature descriptor and it captures the information about the object's shape well. The local object shape within an image is characterized by the distribution of gradient magnitude and direction. In order to detect the objects in the image, we need to run and classify HOG descriptors for a large number of image patches. To accomplish this with less computation effort, we slightly modified the original computation of HOG [10]. The concept of integral image proposed by Viola and Jones [15] is used to leverage the computation speed for the calculation of HOG features. Firstly, we compute the gradients in horizontal and vertical directions of an image (of shape m x n) and calculate the magnitude and direction of gradients of image. Secondly, we discretize the gradient orientations into q orientation bins and form a histogram of oriented gradients for each pixel, resulting in matrix of shape m x n x q. Finally, we compute the integral image of the computed HOG, so that we can compute the HOG for any image patch efficiently in constant time, this we call integral gradient image (IntGradImg). For instance, given a rectangular image patch represented by points (p11, p12, p21, p22) we can compute HOG descriptor as given in (1).

$$\text{hog} = \text{IntGradImg}(p11) - \text{IntGradImg}(p12) - \text{IntGradImg}(p21) + \text{IntGradImg}(p22) \quad (1)$$

For any given image patch, we subdivide the image patch into equal spatial cells (nx, ny). For each cell, we calculate the HOG descriptor which is discretized into q orientation bins as stated above. We finally concatenate the histograms of all the cells and normalize the resulting descriptor vector with respect to its L2-norm.

3.1.3 Classification - SVM

SVM trained on HOG descriptors is considered as a de facto standard in many visual perception tasks [36]. The idea of SVM is simple, to build an optimal hyperplane which separates the data into classes. SVM is built on the concept of margins, where margin is the separation between the closest class points called support vectors. The goal of the SVM classifier is to maximize the margins to result an optimal hyperplane to classify the data.

3.2 Object detection with SSD

SSD [33] detects the objects by passing through input image only once in a single shot, to predict object class for classification and bounding box for localization of objects. Concretely, for a given input image, firstly, the input image is passed through a series of convolutional layers yielding feature maps at different scales. Secondly, for each location of feature map a small (of shape 3x3) convolutional filter is used to evaluate a small set of default bounding boxes called anchor boxes. Finally, for each anchor box, the bounding box offsets and class probabilities are simultaneously predicted.

The architecture of SSD consists of feature map extraction network, additional feature layers and prediction layers. VGG16 [20] is used as feature map extraction network (also called backbone network) in [33]. In addition to *conv4_3* layer of VGG16, five additional feature layers as a part of multi-scale feature maps are used to detect objects. At each location of feature map, a certain number of anchor boxes are evaluated to predict the objects, and each prediction comprises of *(C+4)* parameters, where *C* is the number of classes of objects and *4* are rectangular bounding box parameters. This type of prediction in multiple boxes is known as *multibox* detection. Instead of RPN as in Faster R-CNN [26], SSD uses 3x3 convolutional filters with *k*(C+4)* number of filters, where *k* is number of anchor boxes for each layer, to predict class and bounding box parameters. The predictions from multi scale feature layers are concatenated to obtain final predictions. Alongside VGG16, MobileNetV2 [8] and ResNeXt50 [9] are also used as backbone networks to evaluate SSD.

The objects of different sizes and aspect ratios are detected by SSD by leveraging the use of multi scale feature maps and default bounding boxes. Lower resolution feature maps with higher receptive field are responsible to detect bigger objects, on the other hand, higher resolution feature maps with lower receptive field are responsible for the detection of smaller objects. SSD uses non-max suppression to filter the duplicate predictions.

3.3 Object detection with YOLO

The main idea of YOLO [30] is that the input image is divided into *S*x*S* grid, if the centre of object falls into a grid cell, that grid cell is responsible for detecting that object. Each grid cell predicts *k* bounding boxes, confidence scores and conditional class probabilities. At inference time, in order to get class specific confidence scores for each box, we multiply conditional class probability and individual box confidence predictions. The predictions are encoded as an array of shape *S*x*S*x*k**(5+*C*), where *k* is number of anchor boxes and *C* is classes. Classification and localization is done by single network, thereby end-to-end training is possible for better accuracy. YOLOv2 is proposed in [31] to make YOLO better, stronger and faster. YOLOv2 proposes certain improvements to YOLO with the objective to improve accuracy of detections, like employing batch normalization, high resolution classifier and anchor boxes. YOLOv3 [32] employs a feature pyramid like prediction network at 3 different scales to cope with objects of different sizes.

4. IMPLEMENTATION DETAILS

4.1 About the dataset

The traditional and deep learning object detection methods are compared by evaluating the methods on PASCAL VOC 2007 and 2012 dataset [7]. The dataset consists of 16,551 images in the PASCAL VOC 2007 and 2012 training and validation dataset and 4,925 images in the PASCAL VOC 2007 test dataset. The dataset consists of 20 classes of objects. For the evaluation of methods, we consider five classes namely, bicycle, bus, car, motorbike and person which are relevant for traffic scenarios and all other objects are treated as 'other' class.

4.2 Training details - HOG and SVM

In order to train a SVM classifier, we need to collect positive and negative samples for different classes. The positive training and validation samples of all classes are collected from the annotated training and validation images subset of PASCAL VOC 2007 and 2012 dataset respectively. The samples are described with HOG descriptor as specified in Section 3.1.2.

The SVM classifier is trained with training samples for six different classes (bicycle, bus, car, motorbike, person, other). The support vector classification with a linear kernel from Scikit-learn [37] is used, whose implementation is based on libsvm [38]. We use one-vs-rest multi class strategy to train the classifier. This strategy tries to fit one classifier per class. For each classifier, the class is fitted against all other classes. The main advantage of such strategy is its interpretability, where we can gain knowledge about the class by inspecting its corresponding classifier.

The classifier is validated on the validation samples and by experimentation the regularization parameter (C) is set to 0.92 to reduce over fitting of data. To obtain a better classifier, good negative samples are important. A technique known as bootstrapping is applied, to select hard negative samples for next round of training by evaluating the current classifier on negative samples and selecting high confident negative samples. The classifier is re-trained using original positive samples and original + new negative samples.

To detect objects in a given image, the image is passed through a detection module. The detection module consists of sub modules such as, selecting RoIs using selective search, computing integral gradient image as described in Section 3.1.2, describing each RoI using HOG descriptor, classification of the descriptor using the trained classifier. The detection module may predict multiple bounding boxes for a same object. The filtering of predictions is performed by applying per class confidence thresholding and non-maximum suppression. The method is implemented in Python with the aid of open-source Python libraries like Numpy [40] and Scikit-learn [37].

4.3 Training details - SSD

SSD is trained with 16,551 images with ground truth annotations from the PASCAL VOC 2007 and 2012 training and validation dataset on a NVIDIA GeForce RTX 2080 Ti GPU. The network is trained with VGG16 [20], MobileNetV2 [8] and ResNeXt50 [9] as backbones, that are originally trained on the ImageNet dataset [18]. In order to train the network, we need to choose the anchor boxes manually, depending on our dataset. Four or six anchor boxes are attached to each prediction layer of the network. We define scale value and aspect ratios of anchor boxes for each prediction layer of the network. The scale values range from 0.1 to 0.9 for higher resolution layers to lower resolution layers respectively. The aspect ratios for layers with six anchor boxes are: 1, 2, 3, 1/2 and 1/3. The width and height of anchor boxes are calculated using (2).

$$\text{width} = \text{scale}\sqrt{\text{aspect ratio}}$$
$$\text{height} = \text{scale}/\sqrt{\text{aspect ratio}} \quad (2)$$

The assignment of anchor boxes per prediction layer results in large number of anchor boxes. The anchor boxes are classified as positive matches and negative matches. The SSD penalizes localization loss from only positive matched anchor boxes. An anchor box is considered positive match, if the Intersection over Union (IoU) of that corresponding anchor box with ground truth box is greater than 0.5. An anchor box whose IoU with ground truth box is greater than 0.3 but less than 0.5 are considered as neutral boxes and are not considered for computation of loss, as these boxes are 'too close' to a ground truth box to be a valid negative background box. The loss function is a weighted sum of localization and classification loss. The large number of anchor

boxes results in higher negative matches than positive matches, this results in class imbalance which hurts the training. So, hard negative mining is applied, where instead of using all negatives we sort the negatives by their confidence loss and pick the negatives with top loss, such that negative to positive matches ratio is kept at most as 3:1. This results in faster and stable training. Batch normalization and data augmentation schemes, such as flipping, cropping, colour distortions are applied to avoid over fitting. The SSD method is implemented in Python using TensorFlow library [41]. The network is trained for 120 epochs with 1000 steps per epoch, with a batch size of 32. Adam optimizer is used with a learning rate of $1e^{-3}$ for first 80 epochs and $1e^{-4}$ for rest of epochs.

4.4 Training details - YOLO

YOLO is trained with 16,551 images with ground truth annotations from the PASCAL VOC 2007 and 2012 training and validation dataset on a NVIDIA GeForce RTX 2080 Ti GPU. The network is trained with MobileNetV1 [39] and ResNet50 [12] as backbones, that are originally trained on the ImageNet dataset [18]. Three anchor boxes are attached to each of the three feature map layers of the network. k-means clustering technique is used to find the anchor boxes. Batch normalization and data augmentation schemes, such as flipping, cropping, colour distortions are applied to avoid over fitting. The YOLO method is implemented in Python using TensorFlow library [41]. The network is trained for 100 epochs with 1000 steps per epoch, with a batch size of 16. Adam optimizer is used with a learning rate of $1e^{-3}$ for initial 30 epochs and $1e^{-5}$ for rest of epochs.

5. RESULTS AND DISCUSSION

This section provides a comparison of qualitative and quantitative results of the object detection task by traditional and deep learning approaches and discusses about the critical aspects of the two approaches.

5.1 Qualitative Results

The results of object detection using traditional and deep learning approaches is shown in Figure 2. As shown in Figure 2b, SSD is able to detect small objects (persons and cars) in the background, but in Figure 2e YOLO is unable to detect the small objects. SSD predicts the objects using multi scale feature maps (6 scales), so the receptive fields of multi layer feature maps are able to detect objects of small and big sizes. The SSD with VGG16 network predicts the persons in the background as shown in Figure 2b(top) but SSD with MobileNetV2 network is unable to detect the persons at the background as shown in Figure 2a(top). MobileNetV2 [8] uses depthwise separable convolutions, that consists of a depthwise and pointwise convolutions one after another as opposed to normal convolutions in VGG16. This drastically reduces the number of parameters of the network, thereby reducing model size and complexity, resulting in increase of processing speed but at the cost of small reduction in accuracy as shown in Figure 2a (top) and Figure 2b(top).

The traditional object detection approach (HOG+SVM) could not detect objects with good accuracy as compared to deep learning approaches. As shown in Figure 2f, the selective search which is used to generate regions is not good enough to provide more regressed bounding boxes like in deep learning approaches. The traditional approach which uses SVM as a classifier is unable to provide better classification results when compared to deep learning approaches.

5.2 Quantitative Evaluation

The evaluation of traditional and deep learning object detection approaches is performed on PASCAL VOC 2007 test set. The mAP is considered as an evaluation metric for object detection. mAP is the average of precision over all categories. Average Precision (AP) is the

area under Precision-Recall curve. A prediction is considered positive, if its IoU with the ground truth is greater than 0.5.

(a) (b) (c)

(d) (e) (f)

Figure 2. Object detection results: (a) SSD-MobileNetV2 (b) SSD-VGG16 (c) SSD-ResNeXt50 (d) YOLO-MobileNetV1 (e) YOLO-ResNet50 (f) SVM-HOG

The mAP, per-class average precision and speed (in frames per second - fps) of traditional and deep learning approaches is shown in Table 1.

The deep learning approaches like SSD and YOLO outperform the traditional approach like HOG and SVM by huge margin in terms of mAP as shown in Table 1. The inference of deep learning approaches is performed on a NVIDIA GeForce RTX 2080 Ti GPU. The processing speeds of both the deep learning approaches are above par real time speeds, which is very crucial for self-driving vehicles in traffic scenarios. Although SSD with ResNeXt50 network gives higher mAP than other SSD backbone architectures, the processing speed for this approach reduces to almost half as compared with SSD with MobileNetV2 architecture. As a trade off between accuracy and speed, we can consider SSD with VGG16 as a better approach for object detection.

Table 1. Mean Average Precision (mAP), per-class Average precision (AP) and processing speed of different object detection approaches on PASCAL VOC 2007 test set

Method	Backbone	Parameters	Per-class AP					mAP	Speed (in fps)
			Bicycle	Bus	Car	Motorbike	Person		
SSD	VGG16	~26.3M	84.7	84.5	81.6	81.9	75.9	81.7	71
	MobileNetV2	~7.5M	81.4	81.1	78.1	78.6	72.7	78.4	**95**
	ResNeXt50	~29.5M	86.2	84.2	82.2	81.7	77.3	**82.3**	48
YOLOv3	MobileNetV1	~24.2M	75.9	75.7	73.5	73.8	68.1	73.4	73
	ResNet50	~45.3M	79.8	79.4	76.4	77.1	71.3	76.8	33
SVM	HOG	-	24.5	23.7	21.2	20.7	17.8	21.6	2*

*on Intel Core i9 CPU

5.3 Discussion

Deep learning is pushing its limits to obtain a super human accuracy in object detection when compared to traditional object detection methods. There are benefits and drawbacks to deep learning approaches as compared to traditional computer vision techniques. Deep learning brings in many challenges such as computation power, big data, training time etc. But rapid progressions in device capability in terms of computation power, memory capacity has improved the performance and cost effectiveness of deep learning approaches. The rise of data, where in the era itself is being called as `era of data', has also paved way for the up rise of deep learning approaches.

Traditional methods of computer vision requires expert analysis, for instance, choosing a feature descriptor for classification or correspondence matching. It is difficult to choose which features are significant for a specific data. For example, in case of classification, as the number of classes increases the feature extraction will become more cumbersome. The handcrafted feature design remains a bottleneck in traditional methods, as it is very subjective in nature. On the other hand, deep learning algorithms are trained end-to-end to extract features ranging from low to high level from the data. The deep learning network discovers underlying semantic information and automatically learns the most salient features, which provides a better accuracy compared to handcrafted feature descriptors.

Albeit as shown in Table 1, it is established that deep learning methods perform well than traditional methods for object detection, there are trade-offs with respect to computing requirements, availability of data and training time. The training of deep neural networks require lots of computation power and training time. In our case, to train a SSD model, it took nearly 24 hours on a NVIDIA GeForce RTX 2080 Ti GPU. The deep learning methods require lots of data to train the network otherwise the network may overfit to training data and may not generalize well to other data. The features learned from a neural network is specific to the trained dataset, if not trained well, probably won't perform well for other images. Whereas, some traditional algorithms like SURF can be used for applications such as correspondence matching which don't require class specific knowledge. So, deep learning is not a unique solution for every problem but in our case of object detection, it outperforms the traditional techniques.

6. CONCLUSION

This work presents a comparison of traditional and deep learning approaches for the task of object detection in traffic scenarios, in terms of mAP and processing speed. The traditional

method of handcrafted feature descriptor like HOG and a linear SVM classifier is compared with deep learning algorithms like SSD and YOLO. SSD algorithm is implemented with different backbone architectures like VGG16, MobileNetV2 and ResNeXt50, similarly YOLO algorithm is implemented with MobileNetV1 and ResNet50 architectures to compare the performance of the approaches. The training of the algorithms is performed on PASCAL VOC 2007 and 2012 training and validation datasets and inference is performed on PASCAL VOC 2007 test set. For the calculation of mAP, we consider five classes namely, bicycle, bus, car, motorbike and person, which are relevant for traffic scenarios. The training and inference is performed on a NVIDIA GeForce RTX 2080 Ti GPU.

We presented both qualitative and quantitative results of the object detection for comparison of traditional and deep learning approaches in Section 5. Qualitatively and quantitatively, the deep learning approaches outperform the traditional approach both in accuracy and speed, as shown in Figure 2 and Table 1 respectively. The deep learning approaches obtain above par real time processing speed, which is significant for self-driving vehicles in traffic scenarios. SSD algorithm is able to detect the small objects in the background (persons and cars in Figure 2b), which is not the case with YOLO, this is due to the fact that SSD uses multi scale feature maps which leverages small and large receptive fields to detect small and large objects. SSD with MobileNetV2 architecture obtains higher processing speed due to depthwise separable convolutions which drastically reduces the number of parameters in the network, but this reduces the mAP compared to ResNeXt50 architecture. As a trade off between accuracy and speed, one can consider SSD with VGG16 architecture to perform well.

In the traditional approaches, we use handcrafted features like SIFT, HOG etc. for applications related to visual perception. The selection of handcrafted features is very subjective in nature and it becomes a cumbersome task to select the feature descriptors when a large variety of subjects are involved in the scene. On the other hand, deep learning architectures learn the underlying patterns in the data and automatically work most relevant and salient features from the data, which leverages the accuracy in certain computer vision applications (like object detection). But, this is achieved at the cost of requiring lot of data and time to train the network alongside with high computational power.

ACKNOWLEDGEMENTS

This project is funded by the European Union's Horizon 2020 research and innovation program under the Marie Sklodowska-Curie grant agreement No. 765866.

REFERENCES

[1] Pettitt Michael, Burnett Gary and Stevens Alan., "Defining Driver Distraction," in *Intelligent Transportation Society of America - 12th World Congress on Intelligent Transport Systems*, 2005.

[2] Singh J, Sahni MK, Bilquees S, Khan SMS, Haq I, "Reasons for road traffic accidents - victims' perspective," in *Int. J. Med. Sci. Public Health*, 2016;5:814-818.

[3] Zhang Xingjian, Zhao Xiaohua, Du Hongji and Rong Jian, "A Study on the Effects of Fatigue Driving and Drunk Driving on Drivers' Physical Characteristics. Traffic injury prevention," 15. 10.1080/15389588.2014.881996, 2004.

[4] W. Jones, "Keeping Cars from Crashing," in *IEEE Spectrum*, vol. 38, no 9, pp. 40-45, 2001.

[5] L. Li, D. Wen, N.-N. Zheng and L.-C. Shen, "Cognitive cars: A new frontier for ADAS research," in *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 1, pp. 395-407, Mar. 2012.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012.

[7] Everingham M, Van Gool L, Williams C K I, Winn J. and Zisserman A., "The PASCAL Visual Object Classes (VOC) Challenge," in *International Journal of Computer Vision*, 88(2), 303-338, 2010.

[8] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov and L. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, 2018, pp. 4510-4520.

[9] S. Xie, R. Girshick, P. Dollár, Z. Tu and K. He, "Aggregated Residual Transformations for Deep Neural Networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, 2017, pp. 5987-5995.

[10] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, 2005.

[11] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," in *Int. J. of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[12] Herbert Bay, Tinne Tuytelaars and Luc Van Gool, "SURF : Speeded Up Robust Features," in *ECCV*, 2006.

[13] C. Cortes and V. Vapnik, "Support vector machine," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[14] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," in *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, pp. 1627–1645, 2010.

[15] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR* 2001, Kauai, HI, USA, 2001, pp. I-I.doi: 10.1109/CVPR.2001.990517.

[16] C. Papageorgiou and T. Poggio, "A trainable system for object detection," in *IJCV*, 38(1):15–33, 2000.

[17] D. M. Gavrila and V. Philomin, "Real-time object detection for smart vehicles," in *CVPR*, Fort Collins, Colorado, USA, pages 87–93, 1999.

[18] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009.

[19] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015.

[20] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv:1409.1556*, 2014.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[22] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *CVPR*, 2015.

[23] H. Nakahara, H. Yonekawa, S. Sato, "An object detector based on multiscale sliding window search using a fully pipelined binarized CNN on an FPGA," in *Proc. IEEE Int. Conf. Field Program. Technol. (ICFPT)*, pp. 168-175, Dec. 2017.

[24] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *CVPR*, 2014.

[25] R. Girshick, "Fast r-cnn," in *ICCV*, 2015.

[26] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards realtime object detection with region proposal networks," in *NIPS*, 2015, pp. 91–99.

[27] Y. Li, K. He, J. Sun et al., "R-fcn: Object detection via region-based fully convolutional networks," in *NIPS*, 2016, pp. 379–387.

[28] T.-Y. Lin, P. Dollar, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, “Feature pyramid networks for object detection,” in *CVPR*, 2017.

[29] K. He, G. Gkioxari, P. Dollar, and R. B. Girshick, “Mask r-cnn,” in *ICCV*, 2017.

[30] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *CVPR*, 2016.

[31] J. Redmon and A. Farhadi, “Yolo9000: better, faster, stronger,” *arXiv:1612.08242*, 2016.

[32] Joseph Redmon and Ali Farhadi, "YOLOv3: An Incremental Improvement," *arXiv:1804.02767*, 2018.

[33] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in *ECCV*, 2016.

[34] C. Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg, “Dssd: Deconvolutional single shot detector,” *arXiv:1701.06659*, 2017.

[35] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, “Selective search for object recognition,” in *Int. J. of Comput. Vision*, vol. 104, no. 2, pp. 154–171, 2013.

[36] H. Bristow and S. Lucey, “Why do linear SVMs trained on HOG features perform so well?” in *arXiv preprint arXiv:1406.2419*, 2014.

[37] Pedregosa et al., "Scikit-learn: Machine Learning in Python," in *JMLR 12*, pp. 2825-2830, 2011.

[38] Chih-Chung Chang and Chih-Jen Lin, "LIBSVM : a library for support vector machines," in *ACM Transactions on Intelligent Systems and Technology*, 2:27:1--27:27, 2011.

[39] Andrew G. Howard, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *arXiv:1704.04861v1*.

[40] 1. Oliphant TE, "A guide to NumPy," Vol. 1. *Trelgol Publishing USA*, 2006.

[41] Martin abadi et al., "TensorFlow: Large-scale machine learning on heterogeneous systems", 2015.

Authors

Gopi Krishna Erabati is a Marie-Curie early stage researcher and doctoral student at Institute of Systems and Robotics (ISR), University of Coimbra, Portugal. He received his bachelor's degree in 2013 from Kakatiya University, India and MSc in 2018 from University of Dijon, France. His research interests include computer vision, machine learning, visual perception and autonomous navigation.

Nuno Gonçalves is a researcher at the ISR, University of Coimbra and Tenured Assistant Professor at the Dept. of Electrical and Computers Engineering of the University of Coimbra. He received his MSc and PhD degrees in 2002 and 2008, respectively from the University of Coimbra. His main research areas are computer vision, computer graphics and machine learning, with special emphasis to geometric problem in vision systems.

Hélder Araújo is a Full Professor at the Dept. of Electrical and Computer Engineering of the University of Coimbra and senior researcher at the ISR, University of Coimbra. His research interests include robot vision, robot navigation, computer vision, sensor fusion and cognitive robotics.