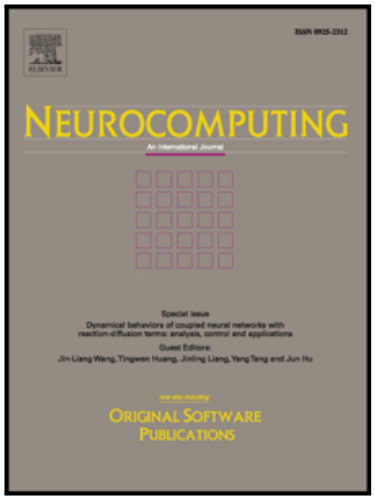
Communicated by Prof. Zidong Wang

Accepted Manuscript



Evaluation of Deep Neural Networks for traffic sign detection systems

Álvaro Arcos-García, Juan A. Álvarez-García, Luis M. Soria-Morillo

PII: S0925-2312(18)30924-X
DOI: https://doi.org/10.1016/j.neucom.2018.08.009
Reference: NEUCOM 19838

To appear in: *Neurocomputing*

Received date: 25 March 2018
Revised date: 21 May 2018
Accepted date: 6 August 2018

Please cite this article as: Álvaro Arcos-García, Juan A. Álvarez-García, Luis M. Soria-Morillo, Evaluation of Deep Neural Networks for traffic sign detection systems, *Neurocomputing* (2018), doi: https://doi.org/10.1016/j.neucom.2018.08.009

Evaluation of Deep Neural Networks for traffic sign detection systems.

Álvaro Arcos-García*, Juan A. Álvarez-García, Luis M. Soria-Morillo

Dpto. de Lenguajes y Sistemas Informáticos, Universidad de Sevilla, 41012, Sevilla, Spain

Abstract

Traffic sign detection systems constitute a key component in trending real-world applications, such as autonomous driving, and driver safety and assistance. This paper analyses the state-of-the-art of several object-detection systems (Faster R-CNN, R-FCN, SSD, and YOLO V2) combined with various feature extractors (Resnet V1 50, Resnet V1 101, Inception V2, Inception Resnet V2, Mobilenet V1, and Darknet-19) previously developed by their corresponding authors. We aim to explore the properties of these object-detection models which are modified and specifically adapted to the traffic sign detection problem domain by means of transfer learning. In particular, various publicly available object-detection models that were pre-trained on the Microsoft COCO dataset are fine-tuned on the German Traffic Sign Detection Benchmark dataset. The evaluation and comparison of these models include key metrics, such as the mean average precision (mAP), memory allocation, running time, number of floating point operations, number of parameters of the model, and the effect of traffic sign image sizes. Our findings show that Faster R-CNN Inception Resnet V2 obtains the best mAP, while R-FCN Resnet 101 strikes the best trade-off between accuracy and execution time. YOLO V2 and SSD Mobilenet merit a special mention, in that the former achieves competitive accuracy results and is the second fastest detector, while the latter, is the fastest and the lightest model in terms of memory consumption, making it an optimal choice for deployment in mobile and embedded devices.

Keywords: deep learning, traffic sign detection, convolutional neural network

1. Introduction

Traffic sign recognition systems (TSRS) form an important component of Advanced Driver-Assistance Systems (ADAS) and are essential in many real-world applications, such as autonomous driving, traffic surveillance, driver safety and assistance, road network maintenance, and analysis of traffic scenes. A TSRS normally concerns two related subjects:

*Corresponding author.

Email addresses: aarcos1@us.es (Álvaro Arcos-García), jaalvarez@us.es (Juan A. Álvarez-García), lsoria@us.es (Luis M. Soria-Morillo)

traffic sign detection (TSD) and traffic sign recognition (TSR). The former focuses on the localisation of the target in a frame, while the latter performs a fine-grained classification to identify the type of the detected target [1, 2].

Traffic signs constitute a fundamental asset within the road network because their aim is to be easily noticeable by pedestrians and drivers in order to warn and guide them both day and night. The fact that signs are designed to be unique and to have distinguishable features, such as simple shapes and uniform colours, implies that their detection and recognition is a constrained problem. Nevertheless, the development of a robust real-time TSRS still presents a challenging task due to the latency in the testing time, which is crucial in making decisions based on the environment and real-world variability, such as scale variations, bad viewpoints, occlusions, and light conditions. Any TSRS must cope well with such issues.

An ADAS relies on LiDAR, onboard RGB cameras, GPS, and IMU sensors. Although traffic signs are normally geo-located and included in navigation maps, they are sometimes replaced or included before the map is updated. The fusion of complementary information acquired from both LiDAR and RGB cameras is a common approach used in TSRS [3, 4]. However, 3D point cloud processing is computationally expensive and the calibration of the sensors and cameras has to be precise since errors of measurement in the order of centimeters may lead to deficient performance.

In recent years, most of the state-of-the-art object-detection algorithms, such as Faster R-CNN [5], R-FCN [6], SSD [7], and YOLO [8], have used convolutional neural networks (CNNs) and can be deployed in mobile devices and consumer products. In order to decide which detector best suits a certain application, not only are standard accuracy metrics important, such as the mean average precision (mAP), but other factors, such as memory consumption and running times, also play a critical role. For instance, autonomous vehicles require good detection accuracy and real-time performance, while mobile devices require lightweight model architectures with low memory usage. In the literature, those detectors are commonly evaluated in object-detection challenges, such as Imagenet [9], PASCAL VOC [10], and Microsoft COCO [11], whose corresponding datasets contain numerous images with common objects, such as cars, planes, people, and bicycles, whereby only accuracy results are reported. However, recent work evaluated the performance of these modern detectors and reported the key metrics, using the Microsoft COCO dataset [12]. Since many of the leading state-of-the-art object-detection approaches have converged on a common methodology that consists of a single CNN that uses sliding-window-style predictions and is trained with a mixed regression and classification objective, the authors implement meta-architectures of Faster R-CNN, R-FCN, and SSD combined with various feature extractors, in order to compare a large number of detection systems in a unified manner.

This paper analyses and compares eight CNN models for object detection that have been previously developed by their corresponding authors and pre-trained on the Microsoft COCO dataset. We fine-tune them on the German Traffic Sign Detection Benchmark dataset (GTSDB) [13] in order to perform traffic sign detection. Considering that the training process of deep CNNs using a very large dataset (e.g. the COCO dataset) requires High Performance Computing (HPC) resources, such as multiple GPUs, and several weeks of continuous training time, we perform transfer learning through fine-tuning to deal with such

issues, which consists of reusing the weights learnt by a trained network on another related network [14]. Evaluated detection models are combinations of meta-architectures (Faster R-CNN, R-FCN, SSD, and YOLO V2) and feature extractors (Resnet V1 50, Resnet V1 101, Inception V2, Inception Resnet V2, Mobilenet V1, and Darknet-19). Such models, pre-trained on the COCO dataset, are publicly available[1][2].

To the best of our knowledge, no other scientific paper analyses several object detectors based on deep learning that are specifically adapted to the domain of the traffic sign detection problem, while evaluating multiple important factors, such as mAP, inference execution time, and memory consumption. The main contributions of this paper are as follows: (1) Presentation of a brief survey of modern object-detection algorithms based on CNNs, namely Faster R-CNN, R-FCN, SSD, and YOLO. (2) Analysis and evaluation of several state-of-the-art object detectors tuned especially for the traffic sign detection task. The evaluation of these models includes key metrics, such as the mAP, memory usage, running time, number of floating point operations (FLOPs), number of parameters of the model, and the effect of traffic sign image sizes. (3) Comparisons and experiments that are made publicly available so that researchers and practitioners can improve their knowledge and fine-tune new models for their comparison with our experimentation. (4) Findings that show that R-FCN strikes the best trade-off between speed and accuracy, SSD models are weak at detecting small traffic signs, and that Mobilenet is the best architecture suited for mobile and embedded devices.

The rest of the paper is organised as follows. Section 2 reviews related work of traffic sign detection systems. Methodology and experiments conducted to analyse several state-of-the-art CNN architectures for object detection are explained in Section 3. In Section 4, the traffic sign detection results obtained are analysed, compared and discussed. Finally, conclusions are drawn and further work is proposed in Section 5.

2. Related work

State-of-the-art research in this field is analysed from two points of view: firstly, traffic sign detection solutions; secondly, deep neural network architectures for object detection.

2.1. Traffic sign detection

Various approaches have been studied for traffic sign detection systems. In [15], a detector composed of two modules is proposed. The former exploits the common properties of sign borders and extracts regions of interest (ROI). The latter performs finer validations over the ROIs and detects traffic signs using a combination of Histograms of Oriented Gradients (HOG) and Support Vector Machines (SVM). A sliding-window detector approach is proposed in [16], where integral channel features classifiers are applied along with the search for traffic signs on different scales and aspect ratios. Wang et al. [17] proposed the winner method for the prohibitory and mandatory signs in the German Traffic Sign Detection

[1] https://github.com/tensorflow/models/blob/master/research/object_detection
[2] https://pjreddie.com/darknet/yolo

Benchmark (GTSDB) [13] challenge. Their system combines a coarse filtering module based on HOG and Linear Discriminant Analysis (LDA) classification on small sliding windows, and a fine filtering module, which includes HOG of larger windows and a SVM classifier. The previous methods are based on the sliding-window schema and feature extraction, which is time-consuming and complex and hence they are not useful for real-time object detection. Recently, Zang et al. [18] combine a local binary pattern (LBP) feature detector with an AdaBoost classifier [19] in order to extract ROIs for coarse selection followed by cascaded CNNs to reduce negative samples of ROI for traffic sign recognition. In 2016, Zhu et al. [20] develop a method to detect and recognize traffic signs based on proposals by the guidance of fully convolutional network. They extent the R-CNN by using an object proposal method, EdgeBox [21] and achieve state-of-the-art results on Swedish Traffic Signs Dataset [22]. Additionally in 2016, Aghdam et al. [23] propose a method that implements the multi-scale sliding window technique within a CNN using dilated convolutions. Dilated convolutions (also known as atrous convolutions) support the exponential expansion of the receptive field without any loss of resolution or coverage and hence they enlarge the field of view of convolutional filters to incorporate a larger context without increasing the amount of computation or the number of parameters [24]. Such an approach locates traffic signs on the GTSDB high-resolution images with an average precision of 99.89% and runs at 37.72 fps. An overview of these revised TSD systems, evaluated on GTSDB, is shown in Table 1.

2.2. Convolutional neural networks for object detection

Since 2013, CNNs, which are able to learn a hierarchy of features by building high-level features from low-level features, have become the standard for object-detection tasks. Examples include are OverFeat [25], which produces bounding boxes and scores using CNNs in a sliding-window fashion, and R-CNN [26], which follows a multi-stage pipeline where object region proposals are extracted from the input image by means of Selective Search [27], whereby feature maps are then computed with a CNN for every region proposal, and finally bounding box regressors and SVM classifiers are applied. R-CNN is expensive both in time and memory because it executes a CNN forward-pass for each object proposal without sharing computation.

To face such issues, Spatial Pyramid Pooling networks (SPPnets) [28] were proposed to improve R-CNN efficiency by sharing computation. SPPnet computes the feature maps from the entire input image only once, and then pools features in sub-images of arbitrary size to generate fixed-length representations to train the detectors. Although the repeated calculation of convolutional feature maps is obviated in SSPnet, it still requires training in a multi-stage pipeline since the fixed-length feature vectors produced by multiple SPP layers are further passed on to fully-connected layers and then, on top of these, bounding box regressors and SVMs are applied. Therefore, the whole process is still slow. Moreover, SSPnet introduces a new problem since parameters below the SPP layer cannot be updated while training.

The more recent Fast R-CNN [29] proposes a new training algorithm that provides solutions to fix the disadvantages of R-CNN and SPPnet, while improving on their speed and accuracy by sharing computation, and by training in a single-stage using a multi-task

Paper	Evaluation (%)				Inference time (FPS)	Hardware	
	Metric	P	D	M		CPU	GPU
Liang et al.(2013)[15]	AUC	100	98.85	92	1-2.5	Intel 4-core 3.7GHz	-
Mathias et al.(2013)[16]	AUC	100	100	96.98	2.5	Intel Core i7 870 3.6GHz	NVIDIA GTX 470
Wang et al.(2013)[17]	AUC	100	99.91	100	0.85	Intel Core i3 3.3GHz	-
Zang et al.(2016)[18]	AUC	99.45	98.33	96.5	*	Intel Core 2 Duo 2.2 GHz	-
Aghdam et al.(2016)[23]	AP	99.89			37.72	-	NVIDIA GTX 980

Table 1: Evaluation results, inference time and hardware utilised in various TSD systems tested on the GTSDB. P refers to prohibitory class, D to danger and M to mandatory. *Time of the full process is not included.

loss and reducing memory consumption. Instead of applying multiple SPP layers as in SSPnets, Fast R-CNN uses a single-level SPP layer, which is called the RoI Pooling layer. Furthermore, the multi-task loss is calculated on top of the network where bounding box regressors and softmax classifiers are applied. The training of the layers below the RoI Pooling layer is possible thanks to these changes, thereby overcoming the original problem of SPPnets. Although SPPnet and Fast-RCNN had reduced the running time of these detection networks, there was a bottleneck exposed in the generation of regions of interest from a proposal method.

In Faster R-CNN [5], in order to overcome such a bottleneck, authors replaced the use of Selective Search with a Region Proposal Network (RPN) that shares convolutional feature maps with the detection network, thus enabling nearly cost-free region proposals. Similar to Faster R-CNN, the Region-based Fully Convolutional Networks (R-FCN) [6] approach applies position-sensitive score maps along with a fully-convolutional region-based detector with shared computation that has no need for the per-region subnetwork to be executed hundreds of times per image. Other approaches, such as Single Shot MultiBox Detector (SSD) [7] and YOLO (You Only Look Once) [8], encapsulate all the computation in a single fully-convolutional neural network instead of having a sequential pipeline of region proposals and object classification. This ability leads to a much faster object detector.

3. Experimentation

The following subsections describe the dataset and the specific configuration used in several CNNs that are fine-tuned for traffic sign detection. Following [12], our experimental setup is composed of four meta-architectures (Faster R-CNN, R-FCN, SSD, and YOLO V2) and six convolutional feature extractors (Resnet V1 50, Resnet V1 101, Inception V2, Inception Resnet V2, Mobilenet V1, and Darknet-19). The feature extractors considered are all well-known convolutional neural networks for image classification that are applied to the input image to obtain high-level features.

Due to time restrictions and computational costs, all experiments presented in this paper use publicly available object-detection models that were pre-trained on the Microsoft COCO dataset [11]. By means of transfer learning [30], we fine-tune these models with the GTSDB dataset in order to detect and classify traffic sign superclasses based on their shapes and colours: mandatory, prohibitory, and danger. At the time of writing this paper, all pre-

	Faster R-CNN	R-FCN	SSD	YOLO V2
Resnet V1 50	✓			
Resnet V1 101	✓	✓		
Inception V2	✓		✓	
Inception Resnet V2	✓			
Mobilenet V1			✓	
Darknet-19				✓

Table 2: Feature extractors vs. architectures. Combinations of CNN architectures and feature extractors evaluated in this paper.

trained models available at the official repositories of Tensorflow Object Detection API [12] and YOLO [8] were used in our experimental setup. The combinations of architectures and feature extractors studied in this work are presented in Table 2. It can be observed that not all possible combinations have been explored. The reason is that each feature extractor must be tailored for use within a meta-architecture. These not trivial adjustments need several experiments and weeks of training, and hence only pre-trained combinations have been selected.

3.1. Datasets

Several publicly available traffic sign datasets have been gathered in countries such as the United States [31], Belgium [32], Germany [13], Croatia [33], Italy [34], Sweden [22], and China [35].

This paper focuses its experimentation on the German Traffic Sign Detection Benchmark (GTSDB) [13] dataset. There are multiple reasons for choosing this dataset over the others, including the fact that it is highly accepted and is widely used for comparing traffic sign detection approaches in the literature. Moreover, its authors and the organization behind them held a public challenge, whereby scientists from different fields contributed their results and tested the GSTDB dataset. Nowadays, a GTSDB website is maintained where submissions of results are still accepted, processed and shown in a leaderboard. Such ranking helps to reveal which state-of-the-art methodologies are utilised for the task of traffic sign detection, although their processing times are not considered. Last but not least, the GTSRB dataset contains natural traffic scenes recorded in various types of roads (highway, rural, urban) during the daytime and at twilight, and numerous weather conditions are featured. This dataset is composed of 900 full images containing 1206 traffic signs that are split into a training set of 600 images (846 traffic signs) and a testing set with 300 images (360 traffic signs). Each of these images contains zero, one, or multiple traffic signs which normally suffer from differences in orientation, light conditions, or occlusions. Signs are grouped in four categories namely mandatory, prohibitory, danger, and other, however, signs labelled as other remain in minority and are not relevant to the challenge itself, and hence are discarded. Consequently, the training set contains 396 prohibitory (59.5%), 114 (17.1%) mandatory and 156 (23.4%) danger traffic sign samples while the testing set comprises 161 prohibitory, 49 mandatory and 63 danger traffic sign images. The following deep neural networks for traffic

Figure 1: Example images from GTSDB dataset.

sign detection are trained and evaluated using this dataset. Figure 1 shows some images from this dataset. The following sections and subsections describe each meta-architecture used and its feature extractors.

3.2. Meta-architectures for object detection

In this subsection, the main features of each meta-architecture (Faster R-CNN, R-FCN, SSD, and YOLO V2) are summarised.

3.2.1. Faster R-CNN

As mentioned in Section 2, Faster R-CNN [5] introduces a Region Proposal Network (RPN), which is a fully convolutional neural network that simultaneously predicts object bounding boxes and objectness scores. It makes the model completely trainable end-to-end since full-image convolutional feature maps are shared with the detection network. Region proposals are generated in a sliding-window fashion, sliding a small network over the output feature map of the latest convolutional layer. The RPN predicts multiple region proposals at each sliding-window location, where k is the maximum number of possible proposals for each location. The k proposals are parameterised relative to k reference boxes called anchors. Each of these anchor boxes are associated with an aspect and scale ratio, and centred at the sliding-window location. In order to reduce redundancy of overlapping RPN proposals, non-maximum suppression (NMS) algorithm is first performed on the proposal regions based on their objectness scores. The NMS algorithm is responsible for merging multiple detections that belong to the same object. Only the $top-N$ ranked proposal regions are then forwarded to the detection network, which finally regresses bounding boxes and classifies each of them in a determined object class.

During experimentation, the number of region proposals to be sent to the box classifier is set to 300 as this is the number of boxes used by the authors in their corresponding papers. Moreover, each feature extractor is trained on images scaled to 600 pixels on their shortest edge using a SGD with momentum (set to 0.9) as the loss-function optimiser [36] along with batch sizes of 1. The initial learning rate is set to 0.0003 and is manually reduced by a factor of 10 twice: after 900,000 iterations and 1,200,000 iterations.

3.2.2. R-FCN

Region-based Fully Convolutional Networks (R-FCN) [6] take the architecture of Faster R-CNN but with only convolutional neural networks. That is, the R-FCN approach applies a fully convolutional region-based detector whose computation is shared across the entire image, thereby obviating the need for the computation of per-region subnetwork to be executed hundreds of times per image. To this end, authors propose position-sensitive score maps to address a dilemma between translation-invariance in image classification (where the shift of an object inside an image should be indiscriminate), and translation-variance in object detection (where the detection task needs meaningful localization representations for the evaluation of how the candidate box overlaps the object). Therefore, R-FCN adopts a sequential two-stage pipeline of region proposal and region classification where candidate regions are extracted by a fully convolutional RPN.

In the same way as for Faster R-CNN, the training configuration as well as the hyper-parameter tuning is exactly the same as was described in the Section 3.2.1 above.

3.2.3. SSD

In comparison with Faster R-CNN and R-FCN architectures, SSD [7] encapsulates all computation in a single feed-forward convolutional neural network to directly infer box offsets and object category scores. Consequently, a stage of bounding box proposal generation and subsequent feature or pixel resampling is not required. SSD uses a set of default boxes (also known as anchors or anchor boxes) that are hand-picked by the developer who has to previously observe the size of the objects to be detected. These default boxes aim to discretise the output space of bounding boxes over different scales and aspect ratios per feature map location. That is, at each feature map cell, SSD predicts the offsets relative to the anchor shapes in the cell, as well as the category scores that indicate the presence of an object class instance in each of those anchors.

Moreover, to handle objects of multiple sizes, SSD combines predictions from feature maps of different resolutions. The early network layers of an SSD model are based on a standard architecture used for high-quality image classification. An auxiliary structure is then added to the network in order to produce multi-scale feature maps for detection purposes. Such a structure is composed of convolutional feature layers whose aim is to decrease the size of these feature maps progressively and allow predictions of detections on multiple scales.

For experimentation, unlike Faster R-CNN and R-FCN, SSD models are trained using RMSprop [37] with a momentum of 0.9 as the loss-function optimiser and batch sizes of 32. The base learning rate is set to 0.004 and is exponentially decayed by a factor of 0.95 for each 800,000 iterations. As regards input image sizes, they are resized to a fixed shape of 300×300 pixels.

3.2.4. YOLO V2

YOLO V2 [8] is inspired by the RPN of Faster R-CNN, which uses hand-picked anchor boxes to predict bounding boxes based on the offsets to these anchors at every location in a feature map. However, on one hand, the YOLO V2 approach runs k-means clustering on

the training-set bounding boxes using a custom distance metric (Eq. 1) in order to find good anchor boxes instead of choosing them by hand. Picking better anchor boxes makes it easier for the network to learn to predict good detection. On the other hand, in order to prevent any anchor box ending up at any point in the image, it predicts the width and height of the box as offsets from cluster centroids and location coordinates relative to the location of the grid cell, by applying a logistic activation to constrain the predictions of the network to fall between 0 and 1.

$$d(box, centroid) = 1 - IoU(box, centroid) \tag{1}$$

The classification model that is used as the base of YOLO V2 is called Darknet-19. Furthermore, YOLO V2 uses batch normalization, which helps to regularize the model and leads to notable improvements in convergence while stabilizing the model [38]. After training, the network is modified and fine-tuned for object detection as described in Section 3.3.5.

In order to improve detection scores, standard data augmentation, such as rotations, random crops and exposure, hue and saturation shifts, are performed along with multi-scale training, which re-sizes the input image size every few iterations, thereby forcing the network to learn to predict detections at different resolutions.

In the same way as for SSD, the loss-function optimiser applied to train the model is RMSprop with a momentum of 0.9 and batch sizes of 64. Moreover, the input image size is 608×608 pixels, and the initial learning rate is set to 0.001, which is decayed by a factor of 10 at steps 400,000 and 450,000.

3.3. Feature extractors

We adopt well-known convolutional neural networks for image classification that will be used as feature extractors to obtain high-level features from input images: Resnet V1 50, Resnet V1 101, Inception V2, Inception Resnet V2, Mobilenet V1, and Darknet-19.

3.3.1. Resnet V1 50 and Resnet V1 101

Resnet V1 101 and Resnet V1 50 are deep residual networks [39] that have succeeded in many challenges, such as ILSVRC, and COCO 2015 (detection, segmentation and classification). To be used as feature extractors of Faster R-CNN and R-FCN meta-architectures, these networks are are split into two stages. The former performs the extraction of RPN features and the latter extracts box classifier features.

Both of these feature extractors are built with four residual blocks: the first three (namely *conv*2_*x*, *conv*3_*x*, and *conv*4_*x* in the original paper) extract RPN features, while the last layer of *conv*4_*x* is used for predicting region proposals. Additionally, box classifier features are extracted by the last layer of the fourth residual block (*conv*5_*x*).

3.3.2. Inception V2

Inception V2 [40] sets the state-of-the-art in the ILSVRC2014 detection and classification challenges. Inception networks make use of Inception units that are able to increase the depth and width of a network without increasing its computational cost.

On one hand, when this feature extractor is used in combination with Faster R-CNN meta-architecture, RPN feature maps are extracted from the $Mixed\_4e$ layer and proposal classifier features from the $Mixed\_5c$ layer. These layers are called respectively $inception(4e)$ and $inception(5b)$ in the network architecture described in [40].

On the other hand, when SDD is applied as a meta-architecture, the feature extraction of region proposals is not required in SSD (as was mentioned in Section 3.2.3), and hence Inception V2 is not split, but instead the whole network model is adopted as the main feature extractor. However, auxiliary convolutional feature maps on multiple scales are needed. The topmost convolutional feature map and a high resolution feature map at a lower level are selected. A sequence of four convolutional layers with batch normalization and depths 512, 256, 256, and 128, is then appended to the previously selected layers to perform the prediction task. Each of these additional layers decay the spatial resolution of feature maps by a factor of 2. For Inception V2, multi-resolution feature maps are generated by the layers $Mixed\_4c$ and $Mixed\_5c$.

3.3.3. Inception Resnet V2

In the case of Inception Resnet V2 [41], the computation efficiency of Inception units are combined with the optimization benefits conferred by residual connections. This feature extractor is only combined with Faster R-CNN meta-architecture in our experiments and hence can be split into two stages. On one hand, RPN features are extracted from the $Mixed\_6a$ layer including its associated residual layers (17×17 grid module, known as Inception-ResNet-B in [41]). On the other hand, box classifier features are obtained using the layers located immediately after the Inception-ResNet-B module up to the convolutional layer $Conv2d\_7b\_1x1$, which follows the 8×8 grid module named Inception-ResNet-C in [41]. This feature extractor is operated with dilated convolutions so that the effective output stride size is 8 pixels.

3.3.4. Mobilenet V1

The Mobilenet V1 [42] model is designed for efficient inference in mobile vision applications thanks to the use of depthwise separable convolutions that reduce both the number of parameters and the computational cost. In fact, Mobilenet V1 achieves the same level of accuracy as VGG-16[43] on Imagenet with only 1/30 of the model size and computational cost.

This feature extractor is used in combination with SDD meta-architecture in our experiments, for that reason, its network architecture is not split and auxiliary convolutional feature maps at multiple scales are needed. Analogously to the modifications performed when using Inception V2 with SSD (Section 3.3.2), multi-resolution feature maps are generated by the layers $conv\_11$ and $conv\_13$, and four additional convolutional layers are then appended with decaying resolution and depths 512, 256, 256, and 128 respectively.

3.3.5. Darknet-19

As described in Section 3.2.4, the original object-classification model Darknet-19, which acts as a feature extractor, is modified to perform object detection. Darknet-19 is similar

Layer	Type	# Maps & neurons	Kernel size/Stride
0	Input	3 m. of 608x608 n.	
1	Conv	32 m. of 608x608 n.	3x3/1
2	Max-Pool	32 m. of 304x304 n.	2x2/2
3	Conv	64 m. of 304x304 n.	3x3/1
4	Max-Pool	64 m. of 152x152 n.	2x2/2
5	Conv	128 m. of 152x152 n.	3x3/1
6	Conv	64 m. of 152x152 n.	1x1/1
7	Conv	128 m. of 152x152 n.	3x3/1
8	Max-Pool	128 m. of 76x76 n.	2x2/2
9	Conv	256 m. of 76x76 n.	3x3/1
10	Conv	128 m. of 76x76 n.	1x1/1
11	Conv	256 m. of 76x76 n.	3x3/1
12	Max-Pool	256 m. of 38x38 n.	2x2/2
13	Conv	512 m. of 38x38 n.	3x3/1
14	Conv	256 m. of 38x38 n.	1x1/1
15	Conv	512 m. of 38x38 n.	3x3/1
16	Conv	256 m. of 38x38 n.	1x1/1
17	Conv	512 m. of 38x38 n.	3x3/1
18	Max-Pool	512 m. of 19x19 n.	2x2/2
19	Conv	1024 m. of 19x19 n.	3x3/1
20	Conv	512 m. of 19x19 n.	1x1/1
21	Conv	1024 m. of 19x19 n.	3x3/1
22	Conv	512 m. of 19x19 n.	1x1/1
23	Conv	1024 m. of 19x19 n.	3x3/1
24	Conv	1024 m. of 19x19 n.	3x3/1
25	Conv	1024 m. of 19x19 n.	3x3/1
26	Route(17)	512 m. of 38x38 n.	
27	Reorg	2048 m. of 19x19 n.	-/2
28	Concat(25,27)	3072 m. of 19x19 n.	
29	Conv	1024 m. of 19x19 n.	3x3/1
30	Conv	40 m. of 19x19 n.	1x1/1

Table 3: YOLO V2 network architecture

to the VGG [43] model architecture since it doubles the number of feature maps after every pooling layer and uses chiefly 3×3 kernels. Moreover, it applies a global average pooling to make predictions together with 1×1 kernels to reduce space dimensionality between 3×3 convolutions [44].

This model is first trained on Imagenet using images of 224×224 pixels. The model is then fine-tuned at a larger image size for a few epochs. This gives the network time to adjust its filters to work better on higher resolution inputs. Finally, the network is modified and fine-tuned for detection by removing the last convolutional layer and replacing it by adding on three 3×3 convolutional layers with 1024 filters each followed by a final 1×1 convolutional layer with the number of outputs needed for the detection task.

The result is the YOLO V2 model architecture shown in Table 3. The output feature maps of the last convolutional layer depend on several factors including the number of predicted bounding boxes at each cell $PredB$, which corresponds to the number of anchor boxes, the number of coordinates $CoorB$, and the number of different classes $ClassB$. We set $CoorB = 5$ as the network predicts the center coordinates, width, height and confidence per each bounding box resulting in $(ClassB + CoorB) * PredB = (ClassB + 5) * PredB$ output feature maps. For this experiment, we set 5 anchor boxes computed through the k-means clustering algorithm, and hence the number of filters of the last convolutional layer is 40.

4. Results

In this section we present the performance of the traffic sign detector experiments described in Section 3. The analysis of each of these experiments includes multiple measures,

such as accuracy, number of parameters, floating point operations (FLOPs), memory consumption, and processing time. The models are trained and evaluated on a computer built with an Intel Core i7-4770 CPU, 16 GB of RAM and a NVIDIA Titan Xp discrete GPU, which has 3840 CUDA cores and 12 GB of RAM. As development tools, we used Darknet [3], Darkflow[4] and the Tensorflow Object Detection API [12].

Timings are comprised of both GPU and CPU execution times, and post-processing tasks, such as NMS, are also included. Both execution times and memory demand are reported for a batch size of one and they are averaged over 300 images (GTSDB testing set). The Tensorflow profiler tool[5] was employed to compute these measures as well as the number of parameters and floating point operations (multiply-adds). Our timings are comparable to each other, however, they may not be directly comparable to other reported timing results in the literature since major differences could exist within the computer used, such as software drivers, hardware, framework, and batch size. Nevertheless, factors, such as the total memory allocation of the models during inference, the number of parameters, and the floating point operations, constitute platform-independent measures.

4.1. Accuracy evaluation measure

The mean Average Precision (mAP) quantitative measure from PASCAL VOC 2010 [10] is used to evaluate the performance of the proposed traffic sign detector. First, the interpolated Average Precision (AP), which tracks the precision/recall curve, is computed by setting the precision for recall r to the maximum precision obtained for any recall $r' \geq r$ (Eq. 2), where $p(r')$ is the measured precision at recall r'. The AP measure can then be calculated as the area under this curve by numerical integration that is approximated by the sum of the precision at every k where the recall changes, multiplied by the change in recall $\Delta r(k)$ (Eq. 3), where N is the total number of points where recall changes. Finally, the mAP measure is calculated by taking the average of the APs of all the classes.

$$p(r) = \max_{r':r' \geq r} p(r') \tag{2}$$

$$AP = \sum_{k=1}^{N} p(k)\Delta r(k) \tag{3}$$

In order to determine true and false positive predicted bounding boxes B_p, their respective intersection over union (IoU) with the ground truth bounding boxes B_{gt} are computed. IoU is defined as the area of overlap $B_{gt} \cap B_p$ divided by the area of union $B_{gt} \cup B_p$ (Eq. 4). A prediction is correct when its IoU is greater than 0.5 and it is a false positive otherwise.

[3] `http://pjreddie.com/darknet/` (accessed 11.09.2017)

[4] `https://github.com/thtrieu/darkflow` (accessed 11.09.2017)

[5] `https://github.com/tensorflow/tensorflow/tree/master/tensorflow/core/profiler` (accessed 27.10.2017)

Moreover, ground truth objects with no matching detection are false negatives and multiple detections on the same traffic sign in an image are considered as false positives.

$$IoU = \frac{area(B_{gt} \cap B_p)}{area(B_{gt} \cup B_p)} = \frac{area(B_{gt} \cap B_p)}{area(B_{gt}) + area(B_p) - area(B_{gt} \cap B_p)} \quad (4)$$

4.2. Analyses

Detailed accuracy results per traffic sign superclass are presented in Table 4 along with precision, recall, average precision, and average IoU attained by each detector. On one hand, the worst AP belongs to the mandatory traffic sign category in almost all models, and noticeable differences exist between the AP of the other classes, especially in detectors with lightweight feature extractors, such as SSD variants and YOLO V2. On the other hand, every evaluated model obtains the best AP in its detection of prohibitory traffic sign images. Faster R-CNN Inception Resnet V2 even achieves an accuracy of 100% in this category. These results are highly correlated with the number of training traffic sign samples that are included in the GTSDB dataset described in Section 3.1, where 59.5% are prohibitory and only 17.1% are mandatory. Table 5 includes the FPS, Megabytes of memory, Gigaflops, and millions of parameters of each model sorted by its mAP.

The execution time is really a critical factor for real-time TSD systems. The overall mAP achieved by each model configuration together with its processing time are drawn in Figure 2. Three groups are observed. The first group is comprised of the fastest models, YOLO V2 and SSD, which do not perform region proposal generation. YOLO V2 outperforms the SSD models in terms of mAP, although they do have similar running times. SSD Mobilenet is the fastest of all the models, with an execution time of 15.14 ms per image (66 fps) although its accuracy is slightly worse than that of SSD Inception V2. The second cluster is composed of the Faster R-CNN models with lightweight feature extractors and R-FCN Resnet 101. These models are more accurate and require approximately 100 ms per image on average. In fact, the accuracies obtained by R-FCN and Faster R-CNN when the feature extractor is a Resnet 101 network, are very close to the Faster R-CNN Inception Resnet V2 model (third group), which attains the best mAP: 95.77%. However, it is by far the slowest model due to its processing time, which is almost half a second. Consequently, the R-FCN Resnet 101 model strikes the best balance between accuracy and speed among the model configurations studied, since it achieves an mAP of 95.15% and takes 85.45 ms per image (11.7 fps). A faster option with still good accuracy is that of YOLO V2, which runs at 21.48 ms (46.55 fps).

Additionally, we notice that traffic sign image sizes have negative effects on accuracy. Ground truth traffic sign samples that belong to the validation set are divided into three groups regarding their width. The first group contains 89 images, whose width is in the range [0,32). The second group has 93 samples, and their width is included in the range [32,46). The third group clusters 91 images, which width is greater than 45 pixels. All detectors perform better on larger traffic sign images, as can be seen in Figure 3. One possible reason that explains this fact is that the initial convolutional weights of the networks evaluated were pre-trained on the Microsoft COCO dataset and most of its images have large objects in the

Model	Class	Avg. IoU	Precision	Recall	AP
Faster R-CNN Resnet 50	Prohibitory	82.52	91.38	98.75	98.62
	Mandatory	81.21	70.00	85.71	85.15
	Danger	85.07	79.45	92.06	90.78
Faster R-CNN Resnet 101	Prohibitory	87.29	90.29	98.14	98.13
	Mandatory	85.58	67.65	93.88	93.46
	Danger	87.05	85.51	93.65	93.64
Faster R-CNN Inception V2	Prohibitory	82.73	81.22	99.38	99.36
	Mandatory	79.66	62.50	81.63	80.47
	Danger	85.62	81.69	92.06	92.03
Faster R-CNN Inception Resnet V2	Prohibitory	91.37	96.99	100	100
	Mandatory	89.16	79.31	93.88	93.66
	Danger	90.11	92.19	93.65	93.65
R-FCN Resnet 101	Prohibitory	87.93	84.66	99.38	99.37
	Mandatory	85.37	76.67	93.88	92.58
	Danger	86.95	86.76	93.65	93.52
SSD Inception V2	Prohibitory	81.76	96.95	78.88	78.77
	Mandatory	80.85	90.00	55.10	54.46
	Danger	85.76	93.18	65.08	65.05
SSD Mobilenet	Prohibitory	80.49	92.50	68.94	67.03
	Mandatory	78.51	89.65	53.06	52.01
	Danger	81.11	79.63	68.25	65.85
YOLO V2	Prohibitory	73.96	92.31	89.44	88.73
	Mandatory	74.66	79.07	69.39	65.70
	Danger	75.82	94.55	82.54	82.06

Table 4: GTSDB accuracy results (in %) as attained by each traffic sign detector model. Average IoU only takes IoU values of true positive bounding boxes.

Model	mAP	FPS	Memory (MB)	GigaFLOPS	Parameters (10^6)
Faster R-CNN Inception Resnet V2	95.77	2.26	18250.45	1837.54	59.41
R-FCN Resnet 101	95.15	11.70	3509.75	269.90	64.59
Faster R-CNN Resnet 101	95.08	8.11	6134.71	625.78	62.38
Faster R-CNN Resnet 50	91.52	9.61	5256.45	533.58	43.34
Faster R-CNN Inception V2	90.62	17.08	2175.21	120.62	12.89
YOLO V2	78.83	46.55	1318.11	62.78	50.59
SSD Inception V2	66.10	42.12	284.51	7.59	13.47
SSD Mobilenet	61.64	66.03	94.70	2.30	5.57

Table 5: Models' properties ordered by mAP.

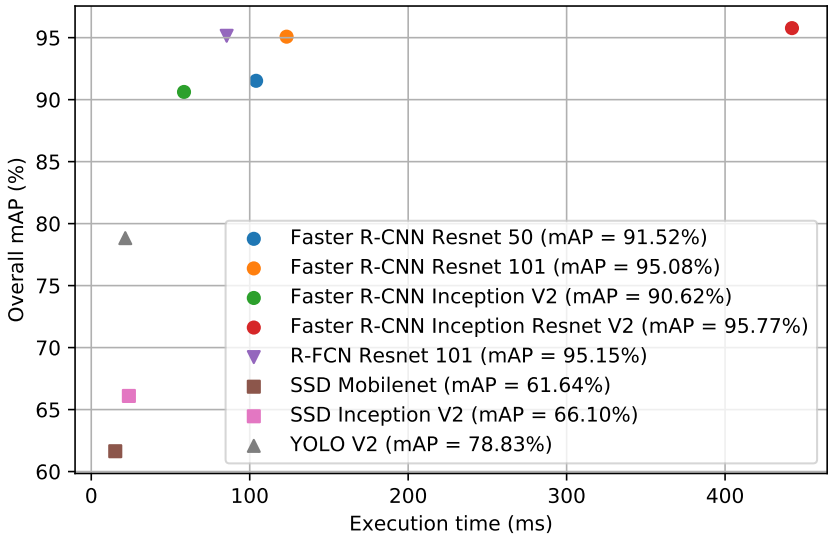


Figure 2: Accuracy vs. execution time.

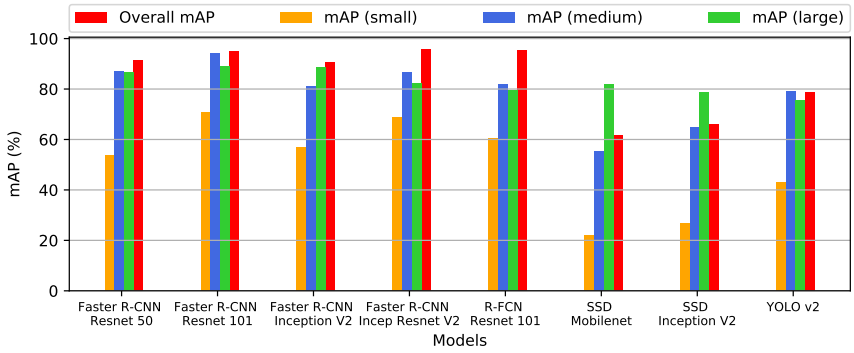


Figure 3: Accuracy classified by traffic sign size for 8 different detectors.

centre of the image. However, traffic signs are generally localised towards the edges of the image and are smaller. YOLO V2 and SSD models show poor performance on small traffic sign images despite reaching accuracy scores better than or similar to Faster R-CNN and R-FCN models on large traffic sign samples. This insight can also be observed in the papers where the models were originally described by their authors [5, 6, 7, 8] and in [12]. For instance, the YOLO V2 model, trained on the PASCAL VOC 2012 dataset [8], achieves a lower performance in detecting small objects, such as plants (49.1% AP) and bottles (51.8% AP) in comparison with its performance in detecting other kinds of larger objects, such as bikes (82% AP), airplanes (86.3% AP), and cars (76.5% AP).

Figure 4 represents the FLOP count against execution time. The number of FLOPs computed by each model is a platform-independent measure. On one hand, the use of denser blocks in residual networks leads to higher FLOPs and computation time for both Faster R-CNN and R-FCN detectors. On the other hand, SSD Mobilenet is the model with the fewest FLOPs and shortest running time. It should be borne in mind that the FLOP counter may not be linear with respect to actual execution times, due to multiple factors, such as hardware optimization, and memory I/O. This fact can be observed in the comparison of YOLO V2 and SSD Inception V2 models. The former executes 62.78 billion FLOPs in less time than the latter, which performs 7.59 billion FLOPs. Moreover, the number of parameters that each neural network has to learn (weights and bias) is not directly related with their running time, as shown in Figure 5. It can be seen that models whose feature extractor is a Resnet 101 contain millions more parameters than detectors with higher (Faster R-CNN Inception Resnet V2) or similar (Faster R-CNN Resnet 50) execution times. YOLO V2 is an analogous case since having approximately 50 million learnable parameters, its computation time is shorter than or nearly equal to that of lightweight models, such as SSD Mobilenet, SSD Inception V2, and Faster R-CNN Inception V2.

Memory consumption is also a critical factor. It helps to make decisions, such as whether a certain model can be trained on a single GPU or whether it is necessary to use a cluster of these computation units, and to decide whether a determined neural network architecture can be deployed in mobile and embedded devices. Figure 6 plots total memory usage against the running time of the models studied. A high linear correlation exists between execution

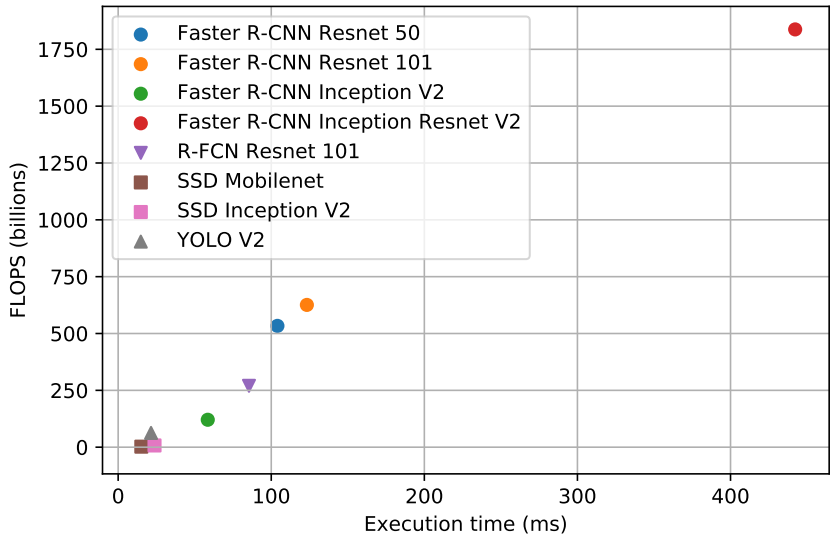


Figure 4: FLOPS vs. execution time.

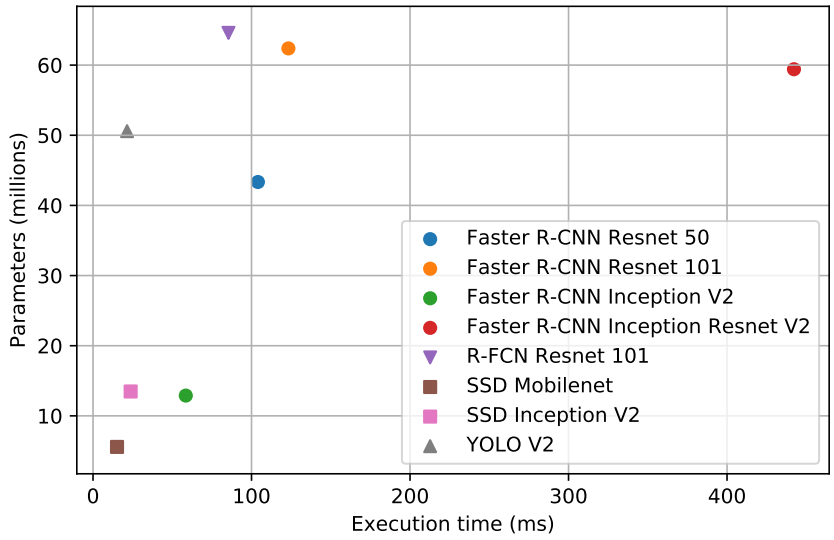


Figure 5: Parameters vs. execution time.

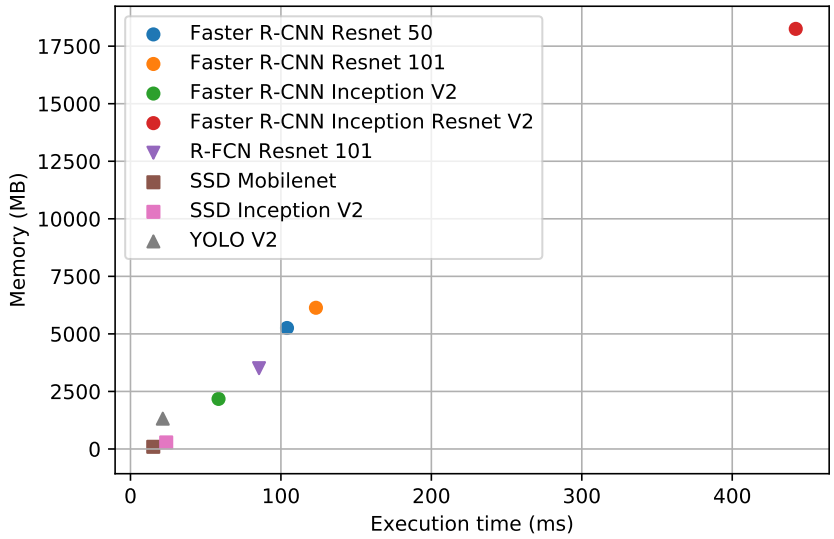


Figure 6: Memory vs. execution time.

time and larger and more powerful feature extractors that require much more memory. Again, the models based on residual networks occupy the top positions in terms of memory usage, while SSD Mobilenet and SSD Inception V2 models are the cheapest in that they require 94.70 MB and 284.51 MB, respectively.

Finally, a radar chart is plotted in Figure 7 whose spokes represent the five measured factors as described above: mAP, execution time, FLOPs, parameters, and memory usage. The minimum value of each measure was considered as the best, except for mAP, where the maximum value was taken as the best. Moreover, for each factor, all values were converted to the range [0,10]. It should be borne in mind that mAP, running time, and memory consumption constitute the most critical factors. Consequently, we observe that the best overall models are R-FCN Resnet 101 and Faster R-CNN Inception V2.

4.3. Traffic sign detections in real-world scenarios

In Figures 8 to 10, a side-by-side comparison is presented of the traffic signs detected in images from the GTSDB dataset using the eight detectors evaluated in this paper. The visualised detections have a score value greater than a threshold of 0.5. Three common scenarios are represented in these figures: Firstly, a road scene that contains small, medium-sized, and large traffic signs of different categories (Figure 8); Secondly, an image where small grouped traffic signs located on both sides of the road can be visualised (Figure 9); Thirdly, a scene where multiple large traffic signs of various categories are grouped and localised on both sides of the road (Figure 10). We observe that all of the detectors perform well on large traffic signs. However, YOLO V2 and SSD models are weak at detecting small traffic signs, especially when these signs appear in groups. Additionally, detection scores are generally lower than those of the remaining detectors. Furthermore, YOLO V2 has certain limitations. It imposes strong spatial constraints on bounding box predictions since

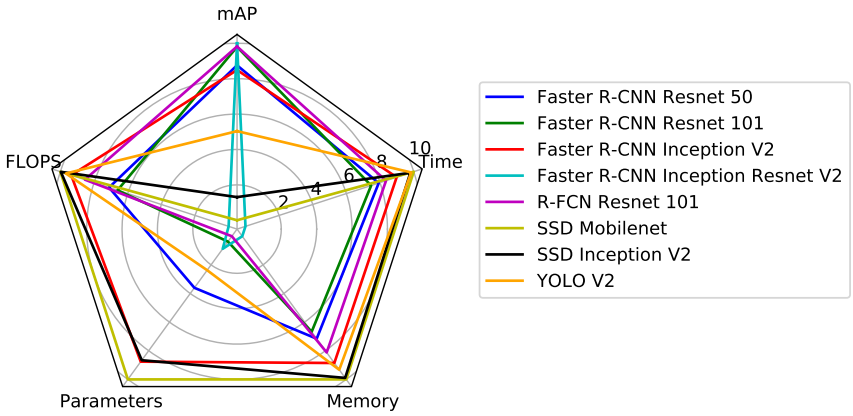


Figure 7: Analysis overview of the traffic sign detector models.

each grid cell can only have one class. This restricts the set of possible predictions in the case where there are many nearby objects, which is the case represented in Figure 9, where multiple small traffic signs appear in groups as mentioned above. Other models, such as R-FCN Resnet 101, and Faster R-CNN Inception V2, also present difficulties in detecting signs in this image because they have some false positives localised very near to the real true positives. It is remarkable that only Faster R-CNN Inception Resnet V2 and Faster R-CNN Resnet 101 models are able to detect every traffic sign included in the scene shown in Figure 8. With scores near to 100%, they even manage to detect a prohibitory traffic sign (the smallest sign) on the left-hand size of the image, which was not annotated in the ground truth.

5. Conclusions and future work

In this paper, a experimental comparison of eight traffic sign detectors based on deep neural networks is presented. We analyse the main aspects of these detectors, such as accuracy, speed, memory consumption, number of floating point operations, and number of learnable parameters within the CNN. All of the models studied in this work were pre-trained on the Microsoft COCO dataset and fine-tuned afterwards with the GTSDB dataset in order to detect and classify traffic sign superclasses based on their shapes and colours: mandatory, prohibitory, and danger.

Accuracy results are evaluated following the mAP quantitative measure from PASCAL VOC 2010. We found that Faster R-CNN Inception Resnet V2 obtains the best mAP (95.77%), while R-FCN Resnet 101 holds the best trade-off between accuracy (95.15%) and execution time (85.45 ms per image). Special mentions are deserved by YOLO V2 and SSD Mobilenet. The former achieves competitive accuracy results (78.83%) and is the second-fastest detector with running times of 21.48 ms per image on average. The latter is the fastest model of all of the detectors and also the least demanding in terms of memory consumption. These key factors make SSD Mobilenet an optimal choice for deployment in

(a) Faster R-CNN Inception Resnet V2

(b) Faster R-CNN Inception V2

(c) Faster R-CNN Resnet 50

(d) Faster R-CNN Resnet 101

(e) R-FCN Resnet 101

(f) YOLO V2

(g) SSD Inception V2

(h) SSD Mobilenet

Figure 8: Example detections from 8 different models in a road scene with small, medium-sized and large traffic signs of multiple categories. All detections are correct in examples *a* and *d*. In *b*, *c*, *e* and *f*, the smallest traffic sign is not recognised. Finally, in *g* and *h*, two traffic signs are not localised.

(a) Faster R-CNN Inception Resnet V2

(b) Faster R-CNN Inception V2

(c) Faster R-CNN Resnet 50

(d) Faster R-CNN Resnet 101

(e) R-FCN Resnet 101

(f) YOLO V2

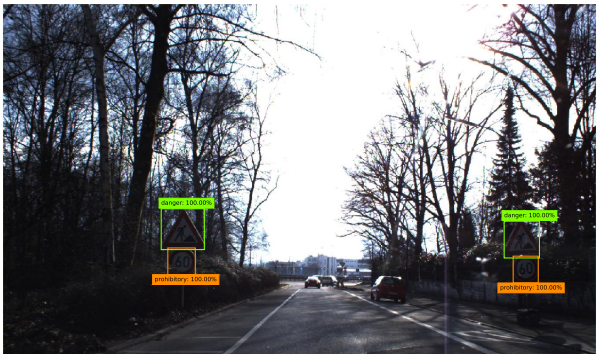
(g) SSD Inception V2

(h) SSD Mobilenet

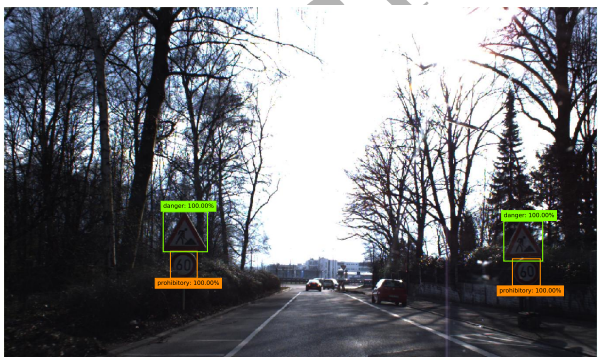
Figure 9: Example detections from 8 different models in a road scene with small traffic signs of the same type grouped on both sides of the road. All detections are correct in examples *a*,*b*,*c* and *d*. In *e*, there are two false positives. Two traffic signs remain undetected in *f*. Finally, in *g* and *h* three traffic signs are not recognised.

(a) Faster R-CNN Inception Resnet V2

(b) Faster R-CNN Inception V2



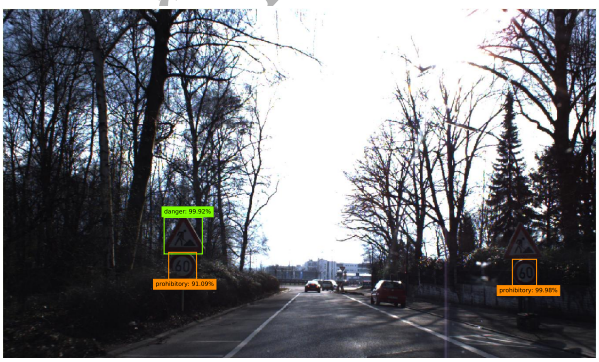
(c) Faster R-CNN Resnet 50



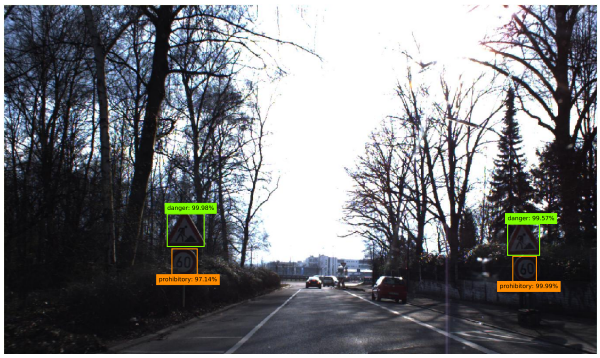
(d) Faster R-CNN Resnet 101

(e) R-FCN Resnet 101

(f) YOLO V2



(g) SSD Inception V2



(h) SSD Mobilenet

Figure 10: Example detections from 8 different models in a road scene with large traffic signs of various categories grouped on both sides of the road. Only in g is there an undetected traffic sign.

mobile and embedded devices. Nevertheless, we observed that SSD models remain very weak at detecting small traffic signs despite the fact that it is critical for any TSDS to perform well at detecting signs in advance so that correct decisions can be made as soon as possible. In general, all of the models present good results at detecting large traffic signs (mAP above 75%). It is also very interesting that only the YOLO V2 and SSD models achieve more than 30 FPS using a NVIDIA Titan Xp, which makes them feasible for real-time traffic sign detection. Another conclusion is that the application of transfer learning to pre-trained models leads to results close to those obtained with the state-of-the-art methods in a specific domain, such as traffic sign detection, where the best results are achieved using a CNN with dilated convolutions on 5 different image scales [23].

It should be borne in mind that the evaluation of the detectors was performed on isolated traffic scene images recorded on various types of roads. Hence, the images are not continuous in time and, consequently, tracking systems could not be used. Such tracking systems could improve the performance of the detectors if the source of the images were made up of consecutive frames extracted from a video.

In future work, we plan to research other neural network architectures that have been proven to work well detecting or classifying general-purpose objects, such as DenseNet [45], and to adapt them to the traffic sign recognition domain. Moreover, advanced embedded platforms, such as NVIDIA Jetson TX2[6] and NVIDIA Drive Px[7], have recently been released: the detectors presented in this paper should be evaluated using these new platforms in order to reveal valuable insights that help practitioners choose and deploy an appropriate traffic sign detector in the real world.

Acknowledgements

This work has been partially supported by the projects "Hermes-Smart Citizen" and "VICTORY" (both MINECO/FEDER R&D, UE) (Grant Nos.: TIN2013-46801-C4-1-R and TIN2017-82113-C2-1-R). We would like to thank NVIDIA for the Titan Xp GPU donated to our research team via the academic GPU seeding program.

References

[1] A. De La Escalera, L. E. Moreno, M. A. Salichs, J. M. Armingol, Road traffic sign detection and classification, IEEE Transactions on Industrial Electronics 44 (6) (1997) 848–859. doi:10.1109/41.649946.

[2] Á. Arcos-García, J. A. Álvarez-García, L. M. Soria-Morillo, Deep neural network for traffic sign recognition systems: An analysis of spatial transformers and stochastic optimisation methods, Neural Networks 99 (2018) 158 – 165.

[3] L. Zhou, Z. Deng, Lidar and vision-based real-time traffic sign detection and recognition algorithm for intelligent vehicle, in: Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on, IEEE, 2014, pp. 578–583.

[6] https://www.nvidia.com/en-us/autonomous-machines/embedded-systems-dev-kits-modules/

[7] https://www.nvidia.com/en-us/self-driving-cars/drive-px/

[4] Á. Arcos-García, M. Soilán, J. A. Álvarez-García, B. Riveiro, Exploiting synergies of mobile mapping sensors and deep learning for traffic sign recognition systems, Expert Systems with Applications 89 (2017) 286–295.

[5] S. Ren, K. He, R. Girshick, J. Sun, Faster r-cnn: Towards real-time object detection with region proposal networks, in: Advances in neural information processing systems, 2015, pp. 91–99.

[6] J. Dai, Y. Li, K. He, J. Sun, R-fcn: Object detection via region-based fully convolutional networks, in: Advances in neural information processing systems, 2016, pp. 379–387.

[7] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, A. C. Berg, SSD: Single shot multibox detector, in: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. 9905 LNCS, 2016, pp. 21–37. arXiv:1512.02325, doi:10.1007/978-3-319-46448-0_2.

[8] J. Redmon, A. Farhadi, Yolo9000: better, faster, stronger, arXiv preprint 1612.

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, IEEE, 2009, pp. 248–255.

[10] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, A. Zisserman, The pascal visual object classes (VOC) challenge, International Journal of Computer Vision 88 (2) (2010) 303–338. doi:10.1007/s11263-009-0275-4.

[11] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C. L. Zitnick, Microsoft COCO: Common objects in context, in: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. 8693 LNCS, 2014, pp. 740–755. arXiv:1405.0312, doi:10.1007/978-3-319-10602-1_48.

[12] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, et al., Speed/accuracy trade-offs for modern convolutional object detectors, arXiv preprint arXiv:1611.10012.

[13] S. Houben, J. Stallkamp, J. Salmen, M. Schlipsing, C. Igel, Detection of traffic signs in real-world images: The german traffic sign detection benchmark, in: Neural Networks (IJCNN), The 2013 International Joint Conference on, IEEE, 2013, pp. 1–8.

[14] S. J. Pan, Q. Yang, A survey on transfer learning, IEEE Transactions on knowledge and data engineering 22 (10) (2010) 1345–1359.

[15] M. Liang, M. Yuan, X. Hu, J. Li, H. Liu, Traffic sign detection by roi extraction and histogram features-based recognition, in: Neural Networks (IJCNN), The 2013 International Joint Conference on, IEEE, 2013, pp. 1–8.

[16] M. Mathias, R. Timofte, R. Benenson, L. Van Gool, Traffic sign recognitionhow far are we from the solution?, in: Neural Networks (IJCNN), The 2013 International Joint Conference on, IEEE, 2013, pp. 1–8.

[17] G. Wang, G. Ren, Z. Wu, Y. Zhao, L. Jiang, A robust, coarse-to-fine traffic sign detection method, in: The 2013 International Joint Conference on Neural Networks (IJCNN), 2013, pp. 1–5. doi:10.1109/IJCNN.2013.6706812.

[18] D. Zang, J. Zhang, D. Zhang, M. Bao, J. Cheng, K. Tang, Traffic sign detection based on cascaded convolutional neural networks, in: 2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2016, pp. 201–206. doi:10.1109/SNPD.2016.7515901.

[19] Y. Freund, R. E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, Journal of computer and system sciences 55 (1) (1997) 119–139.

[20] Y. Zhu, C. Zhang, D. Zhou, X. Wang, X. Bai, W. Liu, Traffic sign detection and recognition using fully convolutional network guided proposals, Neurocomputing 214 (2016) 758–766.

[21] C. L. Zitnick, P. Dollár, Edge boxes: Locating object proposals from edges, in: European Conference on Computer Vision, Springer, 2014, pp. 391–405.

[22] F. Larsson, M. Felsberg, Using fourier descriptors and spatial models for traffic sign recognition, in: Scandinavian Conference on Image Analysis, Springer, 2011, pp. 238–249.

[23] H. H. Aghdam, E. J. Heravi, D. Puig, A practical approach for detection and classification of traffic signs using convolutional neural networks, Robotics and autonomous systems 84 (2016) 97–112.

[24] F. Yu, V. Koltun, Multi-scale context aggregation by dilated convolutions, arXiv preprint arXiv:1511.07122.

[25] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, Y. LeCun, OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks, arXiv preprint arXiv (2013) 1312.6229arXiv:1312.6229.
URL http://arxiv.org/abs/1312.6229

[26] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 580–587.

[27] J. R. Uijlings, K. E. Van De Sande, T. Gevers, A. W. Smeulders, Selective search for object recognition, International journal of computer vision 104 (2) (2013) 154–171.

[28] K. He, X. Zhang, S. Ren, J. Sun, Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence 37 (9) (2015) 1904–1916. arXiv:1406.4729, doi:10.1109/TPAMI.2015.2389824.

[29] R. Girshick, Fast R-CNN, arXiv preprint arXiv:1504.08083arXiv:1504.08083.
URL http://arxiv.org/abs/1504.08083

[30] J. Yosinski, J. Clune, Y. Bengio, H. Lipson, How transferable are features in deep neural networks?, in: Advances in neural information processing systems, 2014, pp. 3320–3328.

[31] A. Mogelmose, M. M. Trivedi, T. B. Moeslund, Vision-based traffic sign detection and analysis for intelligent driver assistance systems: Perspectives and survey, IEEE Transactions on Intelligent Transportation Systems 13 (4) (2012) 1484–1497.

[32] R. Timofte, K. Zimmermann, L. Van Gool, Multi-view traffic sign detection, recognition, and 3D localisation, Mach. Vis. Appl. 25 (3) (2011) 633–647. doi:10.1007/s00138-011-0391-3.
URL http://dx.doi.org/10.1007/s00138-011-0391-3

[33] F. Jurišić, I. Filković, Z. Kalafatić, Multiple-dataset traffic sign classification with onecnn, in: Pattern Recognition (ACPR), 2015 3rd IAPR Asian Conference on, IEEE, 2015, pp. 614–618.

[34] A. Youssef, D. Albani, D. Nardi, D. D. Bloisi, Fast traffic sign recognition using color segmentation and deep convolutional networks, in: International Conference on Advanced Concepts for Intelligent Vision Systems, Springer, 2016, pp. 205–216.

[35] Z. Zhu, D. Liang, S. Zhang, X. Huang, B. Li, S. Hu, Traffic-sign detection and classification in the wild, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2110–2118.

[36] N. Qian, On the momentum term in gradient descent learning algorithms (1999). doi:10.1016/S0893-6080(98)00116-6.

[37] T. Tieleman, G. Hinton, Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude, COURSERA: Neural Networks for Machine Learning (2012).

[38] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, arXiv preprint arXiv:1502.03167.

[39] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[40] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: International Conference on Machine Learning, 2015, pp. 448–456.

[41] C. Szegedy, S. Ioffe, V. Vanhoucke, A. A. Alemi, Inception-v4, inception-resnet and the impact of residual connections on learning., in: AAAI, 2017, pp. 4278–4284.

[42] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: Efficient convolutional neural networks for mobile vision applications, arXiv preprint arXiv:1704.04861.

[43] K. Simonyan, A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, International Conference on Learning Representations (ICRL) (2015) 1–14arXiv:1409.1556, doi:10.

1016/j.infsof.2008.09.005.
URL http://arxiv.org/abs/1409.1556
[44] M. Lin, Q. Chen, S. Yan, Network In Network, arXiv preprint (2013) 10arXiv:1312.4400, doi:10.1109/ASRU.2015.7404828.
URL http://arxiv.org/abs/1312.4400
[45] G. Huang, Z. Liu, K. Q. Weinberger, L. van der Maaten, Densely connected convolutional networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, Vol. 1, 2017, p. 3.

Author biographies

Álvaro Arcos García received his Computer Science Masters degree from University of Seville (Spain) in 2013 and is currently pursuing his Ph.D. in Computer Science at the University of Seville, Spain. His research interests include deep learning, computer vision, ubiquitous computing and cloud computing.

Dr. Juan Antonio Álvarez García is an Associate Professor at the Department of Languages and Computer Systems at the University of Seville. His research focuses in computer vision, deep learning and human activity recognition. He holds a Ph.D. degree in Software Engineering from the University of Seville.

Dr. Luis Miguel Soria Morillo is a Lecturer and Researcher at the Department of Languages and Computer Systems at the University of Seville. His research focuses in activity recognition, human computer interaction and healthcare solutions. He holds a Ph.D. degree in Software Engineering from the University of Seville.