Complexity and Accuracy of Hand-Crafted Detection Methods Compared to Convolutional Neural Networks

Valeria Tomaselli[1(✉)], Emanuele Plebani[2], Mauro Strano[1], and Danilo Pau[2]

[1] STMicroelectronics, Catania, Italy
valeria.tomaselli@st.com
[2] STMicroelectronics, Agrate Brianza, Italy
http://www.st.com

Abstract. Even though Convolutional Neural Networks have had the best accuracy in the last few years, they have a price in term of computational complexity and memory footprint, due to a large number of multiply-accumulate operations and model parameters. For embedded systems, this complexity severely limits the opportunities to reduce power consumption, which is dominated by memory read and write operations. Anticipating the oncoming integration into intelligent sensor devices, we compare hand-crafted features for the detection of a limited number of objects against some typical convolutional neural network architectures. Experiments on some state-of-the-art datasets, addressing detection tasks, show that for some problems the increased complexity of neural networks is not reflected by a large increase in accuracy. Moreover, our analysis suggests that for embedded devices hand-crafted features are still competitive in terms of accuracy/complexity trade-offs.

Keywords: Aggregated channel features · Convolutional neural networks · Detection

1 Introduction

The accuracy of object detection algorithms has improved over the years, on one hand thanks to enriched feature representations (multi-channel, multi-resolution, multi-orientation, etc.) and on the other hand due to the adoption of Convolutional neural networks (CNN), at the price of an increased computational cost, especially in the case of neural-based approaches. The complexity and execution time of detection algorithms have a great impact on many visual recognition applications, such as robotics, automotive safety, and human-computer interaction. In these contexts, real-time execution is crucial.

In this work, we perform an analysis and comparison of feature-based versus CNN-based approaches for object detection both in terms of accuracy and execution time. We focus on the automotive use case, where the task consists in the localization and recognition of three main categories: pedestrians, cars and traffic signs. For the hand-crafted approaches we rely on the well performing

S. Battiato et al. (Eds.): ICIAP 2017, Part I, LNCS 10484, pp. 298–308, 2017.
https://doi.org/10.1007/978-3-319-68560-1_27

Aggregated Channel Features (ACF) detector [1], further optimizing it in terms of memory and speed, and we train three different optimized ACF detectors, one for each class. Then, following the approach implemented by Tomé *et al.* [2], the three detectors generate region proposals for fine-tuned AlexNet [3] networks, incrementally trained to classify pedestrians, cars and traffic signs against the background. Finally, we test an approach entirely based on neural networks, You Only Look Once (YOLO) [4] retrained on the same three categories.

The remainder of the paper is structured as follows: Sect. 2, after a brief excursus on state-of-the-art object detection methods, describes in depth the hand-crafted and CNN-based approaches exploited in this work; Sect. 3 reports the experimental evaluation of the detectors, in terms of accuracy and complexity, on a choice of publicly available datasets; finally, Sect. 4 concludes the paper with some remarks and hints on future work.

2 Object Detection

Given the importance of detecting pedestrians, cars, and traffic signs in automotive, a large number of approaches have been tried over the years. Among the three categories of objects analyzed in this paper, the most important and challenging, because of its large intra-class variability, is "pedestrian". For this reason, most of the efforts in developing new approaches have been focused on it.

2.1 Traditional Approaches

Traditional approaches for object detection usually employ a region proposal algorithm which selects regions from the input image at multiple scales. A high-level feature representation is extracted from the region, which is finally sent to a classifier to establish if that region contains the object or not.

Different region proposal algorithms exist: some of them are class-agnostic and hence they can be quickly adapted to any object detection tasks, but they have the drawback of giving to the subsequent stages an excessive number of negative regions that have to be successively rejected. The problem is lessened by designing a region proposal method tailored on the specific object detection task, to reject most of the negative regions early in the pipeline but preserving as many positive regions as possible.

The region proposal stage is followed by features extraction, which has greatly improved over the years thanks to enriched feature representations. The box-shaped filters, proposed by Viola and Jones [5] in 2003, have been superseded by more complex and powerful features, such as Histogram of Oriented Gradients (HOG) [6]. HOG features in turn have been the starting point of even richer and more complex approaches. For example, Felzenszwalb *et al.* [7] have improved accuracy by combining HOG with a Deformable Part Model and Dollár *et al.* [1] have proposed the Aggregated Channel Features (ACF) descriptor, which combines HOG with normalized gradient magnitude and LUV color channels.

A classifier, such as AdaBoost, Support Vector Machines (SVM), etc. finally decides if the object of interest is in the current region.

2.2 Neural Networks

In the last five years, Convolutional Neural Networks have shown their superiority, in terms of accuracy over hand-crafted features, in a variety of computer vision tasks such as image classification, object detection and semantic segmentation.

In the field of object detection, Regions with CNN (R-CNN) [8] has been widely used due to its generality and fairly good performances. Similarly to traditional approaches, a region proposal algorithm (typically Selective Search [9]) extracts candidate regions, on which CNN features are extracted. Either a SVM classifier is trained on the candidate region features to separate between object classes and background, or the CNN can be directly fine-tuned to discriminate the classes of interest from the background. After classification a non-maxima suppression stage is usually applied to refine the selected bounding boxes.

This complex pipeline is quite slow, especially if the number of proposed regions is high. One possible solution is to drastically reduce the number of regions, by e.g. applying a task-specific region proposal method to reject most of the negative examples. Tomé *et al.* [2] analyze different region proposal methods followed by a CNN-based representation, comparing them in terms of accuracy and efficiency for real-time applications and demonstrating that tailored region proposal algorithms (such as Local Decorrelated Channel Features, LDCF [10], or ACF) consistently outperform general purpose approaches (e.g. sliding window or Selective Search) and they achieve much lower miss rates after the CNN stage. Moreover, LDCF and ACF optimizations further speed up the execution of region proposal.

The running time of R-CNN can be reduced by sharing convolutions across proposals, as done in Spatial Pyramid Pooling [11] and Fast R-CNN [12]. To reduce the execution time of the region proposal stage itself, the Faster R-CNN [13] approach introduces the Region Proposal Network (RPN), which share the same convolutional layers of the classification stage. An even more integrated approach is YOLO [4], where the object detection problem is reformulated as a regression problem matching spatially separated bounding boxes and class probabilities to the ground truth. In this way, a single network is optimized end-to-end directly on detection performances.

2.3 Optimized Aggregated Channel Feature Detection

The Aggregated Channel Features (ACF) detector has been optimized in memory by compressing the classifier parameters, which represent a large part of ACF's memory requirements, with a non-linear scalar quantization.

ACF extracts a set of features from non-overlapped blocks on a multi-resolution pyramid constructed from the input image. The ACF classifier is a boosted cascade of small decision trees: each tree is a set of nodes defined

as $\{(i, v), s\}$ where the tuple (i, v) represents an intermediate node as a feature lookup index i and a comparison threshold v and leaf nodes are represented by a final score s. The index i assumes values between 0 and whc, where w and h are the width and height of the detection window measured in feature blocks and c is the number of features per block. See Table 1 for a summary of the tested models and their parameters, including the minimum amount of bits required to encode the feature index values.

Table 1. Parameters and size of the ACF models.

Model	Trees	Block size	Scores	Thresholds	Index bits	Size (KB)
INRIA	4	2048	8192	6144	13	65.8
Caltech	2	4096	113252	109156	13	1042.0
Compcars	4	2048	8192	6144	12	65.0
Traffic signs	4	400	3200	2800	10	26.9

A trained tree cascade can have a large number of parameters, ranging from 10 s of KB to a few MBs, but the model size can be reduced by employing *scalar quantization* on the parameters of the trees. Thresholds and scores are quantized separately, because they have different ranges and statistics; for the same reason, different centroids for thresholds of different types of features (color channels, gradient magnitude and HOG) are used. If b is the original element size in bits and N_c is the number of centroids in the scalar quantization, the theoretical compression ratio is:

$$r = \frac{\lceil \log_2 N_c \rceil}{b} \tag{1}$$

However, the real compression ratio will be lower, as the index bits cannot be compressed.

There is a notable relation between the centroids of the tree thresholds and the quantization of the features; more precisely, the *centroids* of the quantized thresholds are the *quantization thresholds* for a feature quantization scheme. To see this, consider that already in the uncompressed case the set of all the thresholds in a tree cascade is partitioning the space of the feature values in a number of discrete intervals. If N distinct thresholds are present in the cascade and they are sorted in ascending order, the intervals are $[-\infty, t_1[, [t_1, t_2[, \ldots [t_N, \infty[$ and they implicitly quantize the features in $N + 1$ levels. However, if the thresholds are quantized e.g. in $2^k - 1 \ll N$ levels, there will be only 2^k distinct intervals, which is equivalent to quantize features with 2^k bits. In this case, both features and threshold will be represented by k bits.

The difference between feature values inside an interval has no impact on the result of the classifier, and thus after threshold quantization, the additional feature quantization has no further penalty on accuracy. Moreover, the comparison may be performed directly in the compressed domain if threshold and

feature centroids are coded in ascending order. Let the feature centroid be n_f ($n \in \{0, \ldots, 2^n - 1\}$) and the code of the threshold in the current node be n_t: the left branch in the tree will be selected if $n_f \leq n_t$ and the right branch otherwise. The only operation required is an integer magnitude comparator, which can be efficiently implemented on specialized hardware.

2.4 Convolutional Neural Network Detection

We decided to compare two different CNN-based approaches, a region-based one (R-CNN) and one completely based on neural networks (YOLO), trying to address low complexity target platforms and real time applications.

In R-CNN, we decided to exploit the already trained ACF detectors in the proposal stage to discard most false positives in the early stages of the pipeline. Since learning the parameters of a CNN from scratch requires large annotated datasets, we start from the general-purpose AlexNet neural network, trained on the Imagenet dataset [17] and we fine-tuned it for a few epochs on the target dataset using a small learning rate to adapt the network parameters to the new task. Moreover, we trained three models to incrementally classify pedestrians, pedestrians and cars and finally all the three classes against the background. For training we used the well-known Caffe framework [23].

The CNN has been trained on windows cropped from the images in the dataset; the windows have been generated by the ACF detectors for pedestrians, cars and traffic signs ran with a low classification threshold. The ground truth annotations (described in Sect. 3.1) have been used to assign the windows to the right category, using the "background" class for false positives. By doing so, the CNN classifier learns to reject most of the false positives generated by the region proposal algorithm and it increases accuracy. As done by Tomé *et al.* in [2], the regions identified by the detectors have been enlarged with padding pixels to mitigate the issue of imprecise localization. In addition to the false positives generated by the ACF detectors, the background category has been populated with random negative regions and the final dataset is further refined by a quick visual inspection.

To assess the performances of a fully CNN-based detection approach we selected YOLO, because its low complexity is well suited for a real-time application. In particular, we decided to exploit a low-complexity version, tiny-YOLO, which is much faster than the original YOLO model but less accurate. This model achieves in classification mode the same top-1 and top-5 accuracy as AlexNet but with 1/10th of the parameters, since it lacks the large fully connected layers at the end. Starting from the pre-trained model on Imagenet, we fine-tune the network on the *Cityscapes* dataset (see Sect. 3.1) using the Darknet framework [24].

3 Experimental Evaluation

3.1 Datasets

We have have exploited different state of the art datasets for object detection and in particular we have trained three ACF detectors on object-specific dataset.

The pedestrian detector has been trained on the *INRIA* dataset and the more challenging *Caltech Pedestrians* dataset. In Caltech, we used the "Reasonable" setting for the train and test sets, which subsamples the original sequences by 30×. The car detector has been trained on the front/rear views in the Comprehensive Car (*CompCars*) dataset. The restriction on the viewpoint is justified by the fact that a single ACF detector does not recognize well both front and lateral views due to their different aspect ratio, and thus an additional detector must be specifically trained on lateral views. Of the original 136,726 images, we selected around 1500 rear views for training and test.

Following Mathias *et al.* [18], the traffic sign detector has been trained on the German Traffic Signs Detection (GTSD) [19] and the Belgian Traffic Signs Detection (BTSD) [20] datasets, two large image datasets captured in different German and Belgian cities and containing a variety of light conditions. We have merged the training and test sets of the two datasets to obtain a more robust detector. We use the three annotated super-classes: *mandatory* (M), *danger* (D) and *prohibitory* (P) and we have disregarded traffic signs which do not belong to those super-classes.

Training and test splits for all datasets are shown in Table 2.

Table 2. Training and test splits for INRIA, Caltech, Compcars, the merged GTSD and BTSD and Cityscapes datasets.

	Training		Testing	
	Pos.	*Neg.*	*Pos.*	*Neg.*
INRIA	614	1218	288	453
Caltech	4250		4024	
Compcars	968		484	
GTSD+BTSD	2915	3594	1804	648
Cityscapes	5000		500	

To evaluate the R-CNN detector, we trained and tested the pedestrian networks separately on the INRIA and Caltech dataset; we trained the car network on the CompCars dataset restricted to rear views and the traffic sign detector on *Cityscapes* [21]. The *Cityscapes* dataset contains urban street scenes exhibiting a high variability, in terms of places (50 cities), weather conditions, seasons and daytime light, hence it is suitable to mimic the behavior of the trained model in real scenarios. Only segmentation annotations are currently available and thus we generated object bounding boxes by extracting the rectangles enclosing the segmentation polygons, which are annotated in Javascript Object Notation (JSON) format.

The *Cityscapes* dataset with generated object annotations has also been exploited to fine-tune the tiny-YOLO CNN, by including only the three chosen classes (pedestrians, cars and traffic signs) and with the addition of the

"car-group" and "person-group" categories. We choose this dataset for YOLO as it is the only one to include annotations for all the three classes.

3.2 Metrics for Complexity and Accuracy

To assess computational complexity we compared the analyzed approaches on an NVIDIA Jetson TK1, a development platform equipped with a 192-core NVIDIA Kepler GPU, an NVIDIA quad-core ARM Cortex-A15 CPU and 2 GB of memory. We ported the detectors on this platform and we measured the average time to process all the frames of a reference VGA video containing objects from the three categories; then, we estimated the average frame rate.

More precisely, the ACF detector has been ported on ARM using the NEON Single Instruction, Multiple Data (SIMD) instructions and multi-threading. For the neural network approaches, the Caffe framework [23] has been compiled on the platform with CUDA support and used to test both R-CNN and tiny-YOLO. As the tiny-YOLO model is implemented in the Darknet framework [24], which is not optimized for ARM platforms, we converted the trained model in the Caffe format using our own version of the *Caffe-yolo* project [25] to support a wider range of network architectures.

To assess accuracy, we chose the Log Average Miss Rate (LAMR) evaluation metric proposed by Dollár *et al.* [22]. This metric summarizes detector performance by averaging in the range 10^{-2} to 1 the miss rate at nine points in the False Positives Per Image (FPPI) axis, evenly spaced in log-space. If the curves are approximately linear in this range, the LAMR metric is a smoothed estimate of the miss rate at 10^{-1} FPPI.

3.3 Accuracy Results

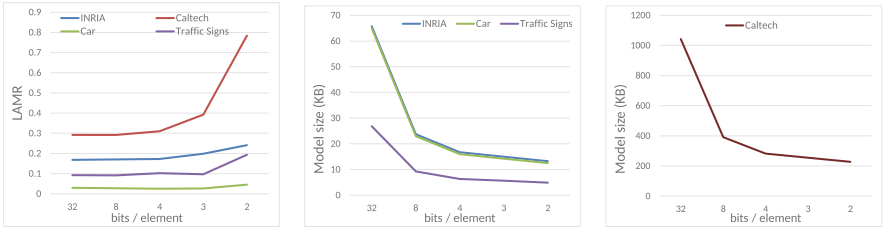
Table 3 reports the results in terms of LAMR and frames per second (*fps*) for the ACF, R-CNN and YOLO detectors trained to recognize different objects (pedestrians, cars and traffic signs) on different datasets. Performances are heavily dependent on the specific dataset, as shown e.g. by the fact that the LAMR of the ACF pedestrian detector is lower for the simpler *INRIA* dataset than it is for the more challenging *Caltech* dataset. Moreover, the ACF traffic sign detector is sensitive to the choice of training set: initially, we trained it only on the GTSD dataset, achieving 9.21% on its test set; however, when the same model was tested on the BTSD dataset to assess its generalization capability, performances dropped to 16.52%. This large difference in performances can be ascribed to the big discrepancy between the two datasets. To obtain a more robust detector which can localize traffic signs even in adverse conditions (e.g. back-light), we merged the two training sets, increasing data variability. By doing so, the performances, assessed on the merged test sets, improved back to 9.21% LAMR, now on a much challenging dataset.

As explained in Sect. 2.3, the ACF models have been optimized in memory, by compressing thresholds and scores. Figure 1(a) shows the change in accuracy of ACF models with increasing compression, measured in bits per parameter (both

Table 3. Detection accuracy an speed on different datasets. The frames per second (*fps*) measure is cumulative, that is, for traffic signs is the speed of running a detector recognizing also cars and pedestrians (ACF trained on INRIA dataset).

Object	Dataset	ACF		R-CNN		YOLO	
		LAMR	fps	LAMR	fps	LAMR	fps
Pedestrians	*INRIA*	16.82	11	30.92	5.2	62.02	1.4
	Caltech	29.2	2.6	28.3	1.98	84.7	
Cars	*CompCars*	2.93	10	2.06	4.11	12.55	
Traffic signs	*GTSD+BTSD*	9.21	8.5	10.97	3.99	27.35	

for thresholds and scores). Up to moderate compression levels (4 bits/element) the impact on accuracy is small, and higher compression rates affect mostly models trained on difficult datasets (Caltech). Moreover, for compression rates of 3 bits/element or higher, the relative reduction in size is smaller, as the indexing bits (term i in Sect. 2.3) starts to dominate the total size, as shown by Fig. 1(b) and (c). The optimal compression level is thus 4 bits/element, which allows a model reduction of around 4× with a loss in LAMR of less than 2% over different datasets and object classes.



(a) (b) (c)

Fig. 1. Results of compression: classification error (LAMR) and model size. In (a) classification error (LAMR) vs bits per element; in (b), model size for the INRIA pedestrian, car and traffic sign models; in (c), model size for the Caltech pedestrian model.

Table 3 also shows LAMR results for R-CNN. R-CNN LAMR is lower than the ACF one both for pedestrians and cars categories, with the notable exception of the INRIA dataset. The annotations in INRIA are incomplete [15] and a large number of pedestrians in the background or partially occluded are not labeled. A quick inspection of the negative examples selected from ACF proposals shows significant overlap with the image of a person in a large fraction of them; as CNN performance is sensitive to label noise, the mislabeled examples end up decreasing the R-CNN accuracy.

For traffic signs the LAMR of the R-CNN approach (10.97%) is slighlty greater than the corresponding ACF detector (9.21%), because the CNN was fine-tuned on the *Cityscapes* and has been evaluated on the merged German and Belgian traffic signs test sets, by running the network on the regions proposed by the ACF traffic signs detector. The network only discards false positives and it cannot increase recall beyond the ACF one.

The tiny-YOLO detector has been fine-tuned on the challenging *Cityscapes* dataset and its accuracy performances have been evaluated on the available test set, obtaining high LAMR values for pedestrians, cars and traffic signs categories (74.92%, 32.66% and 72.48%, respectively). These results mostly depend on the high complexity of the dataset, which contains cars, pedestrians and traffic signs in a variety of views (e.g. front/back/lateral), and the traffic signs category includes many different kinds of street signs, not limited to the mandatory/prohibitory/danger sub-classes. In order to have a fairer comparison between ACF/R-CNN on one side and tiny-YOLO on the other, we have tested the latter on the *INRIA*, *Caltech*, *CompCars* and *GTSD+BTSD* datasets, obtaining the results reported in Table 3. Despite having been trained on a completely different dataset, tiny-YOLO shows acceptable performances in detecting cars, but performances drop in detecting traffic signs and especially pedestrians. This is line with the well-known structural limit of YOLO and tiny-YOLO networks in recognizing small objects.

3.4 Complexity Results

As already explained, the complexity of ACF, RCNN and tiny-YOLO has been evaluated by measuring the average frames per second on a reference VGA video on the NVIDIA Jetson TK1 platform. Results are again reported in Table 3. Since the tiny-YOLO model has been trained on the 3 categories as a whole, complexity figures for one and two categories are not available.

4 Conclusion and Future Work

Our analysis on object detection based on convolutional neural networks reveals that, when compared to an approach based on aggregation of hand crafted features followed by a cascade-of-trees classifier, the latter can provide an accurate, low memory and computational light detector in the automotive applications modelized by the adopted datasets.

Not surprisingly, YOLO shows the worst performances in term of LAMR and fps across all datasets and we had to go through multiple iterations to achieve satisfying results, as the performances we obtained initially were worse than the ones reported. Excluding YOLO, R-CNN on pedestrians decremented LAMR by only 3% compared to ACF, while on traffic signs LAMR incremented by 19% and by 84% on pedestrians/Inria. These are considered poor performances from an implementation point of view, since R-CNN frame rate ranges between 41% on cars and 76% on Caltech Pedestrians when compared to the ACF frame rate

achieved on the NVIDIA Jetson TK1. This result is further exacerbated by the fact that R-CNN is also exploiting the computational power of the embedded GPU on top of the optimized ACF detector we developed, increasing the costs in power consumption and silicon area required to implement the detector.

Our experiments confirmed the initial hypothesis that the increased complexity of neural networks as implemented on the embedded systems under consideration is not justified by a remarkable increase in accuracy, and in some cases, such as traffic sign detection, neural networks even increased the miss ratio. We are also aware that new neural network accelerators are designed and implemented on non-GPU architectures for future smart sensors in order to overcome the aforementioned issues: these architectures will certainly exploit the massive parallelism of multiply and accumulate operations which dominates CNNs. In this direction, further investigation of low-power and low-precision implementations may be promising, such as the binary neural networks approaches [26] aimed at dramatically reducing memory and complexity costs while maintaining adequate accuracy and robustness to noise.

References

1. Dollár, P., Appel, R., Belongie, S., Perona, P.: Fast feature pyramids for object detection. IEEE Trans. Pattern Anal. Mach. Intell. **36**(8), 1532–1545 (2014)
2. Tomé, D., Monti, F., Baroffio, L., Bondi, L., Tagliasacchi, M., Tubaro, S.: Deep convolutional neural networks for pedestrian detection. Technical report, Politecnico di Milano (2015)
3. Krizhevsky, A., Sutskever, I., Hinton, G.: Imagenet classification with deep convolutional neural networks. In: NIPS, pp. 1–9 (2012)
4. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 779–788 (2016)
5. Viola, P., Jones, M., Snow, D.: Detecting pedestrians using patterns of motion and appearance. Int. J. Comput. Vis. **2**, 734–741 (2003)
6. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. **1**, 886–893 (2005)
7. Felzenszwalb, P., Girshick, R., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part-based models. IEEE Trans. Pattern Anal. Mach. Intell. **32**(9), 1627–1645 (2010)
8. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition (2014)
9. Uijlings, J., van de Sande, K., Gevers, T., Smeulders, A.: Selective search for object recognition. Int. J. Comput. Vis. **104**, 154–171 (2013)
10. Nam, W., Dollár, P., Han, J.H.: Local decorrelation for improved pedestrian detection. In: 28th Annual Conference on Neural Information Processing Systems (2014)
11. He, K., Zhang, X., Ren, S., Sun, J.: Spatial pyramid pooling in deep convolutional networks for visual recognition. arXiv:1406.4729v4 (2014)
12. Girshick, R.: Fast R-CNN. arXiv:1504.08083 (2015)
13. Ren, S., Ross, K.H., Sun, G.J.: Faster R-CNN: towards real-time object detection with region proposal networks. arXiv:1506.01497v2 (2015)

14. Dalal, L., Triggs, B.: Histograms of oriented gradients for human detection. In: IEEE Conference on Computer Vision and Pattern Recognition (2005)
15. Taiana, M., Nascimento, J.C., Bernardino, A.: An improved labelling for the INRIA person data set for pedestrian detection. In: Sanches, J.M., Micó, L., Cardoso, J.S. (eds.) IbPRIA 2013. LNCS, vol. 7887, pp. 286–295. Springer, Heidelberg (2013). doi:10.1007/978-3-642-38628-2_34
16. Yang, L., Luo, P., Loy, C.C., Tang, X.: A large-scale car dataset for fine-grained categorization and verification. In: IEEE Conference on Computer Vision and Pattern Recognition (2015)
17. Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L.: ImageNet: a large-scale hierarchical image database. IEEE Conference on Computer Vision and Pattern Recognition (2009)
18. Mathias, M., Timofte, R., Benenson, R., Van Gool, L.: Traffic sign recognition - how far are we from the solution? In: International Joint Conference on Neural Networks (IJCNN), Dallas, USA, (2013)
19. Houben, S., Stallkamp, J., Salmen, J., Schlipsing, M., Igel, C.: Detection of traffic signs in real-world images: the German traffic sign detection benchmark. In: International Joint Conference on Neural Networks (2013)
20. Timofte, R., Zimmermann, K., Van Gool, L.: Multi-view traffic sign detection, recognition, and 3D localisation. Mach. Vis. Appl. **25**, 633–647 (2011)
21. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: IEEE Conference on Computer Vision and Pattern Recognition (2016)
22. Dollár, P., Wojek, C., Schiele, B., Perona, P.: Pedestrian detection: an evaluation of the state of the art. Pattern Anal. Mach. Intell. **34**(4), 743–761 (2012)
23. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: convolutional architecture for fast feature embedding. In: Proceedings of the 22nd ACM International Conference on Multimedia, pp. 675–678 (2014)
24. Redmon, J.: Darknet: Open Source Neural Networks in C (2013–2016). https://pjreddie.com/darknet/
25. Xing, W., Plebani, E.: YOLO (Real-Time Object Detection) in caffe. https://github.com/Banus/caffe-yolo
26. Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R., Bengio, Y.: Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or −1 (2016). arXiv preprint arXiv:1602.02830