Journal of Advanced Research xxx (xxxx) xxx

Contents lists available at ScienceDirect

Journal of Advanced Research

journal homepage: www.elsevier.com/locate/jare

Review

A review on modern defect detection models using DCNNs – Deep convolutional neural networks

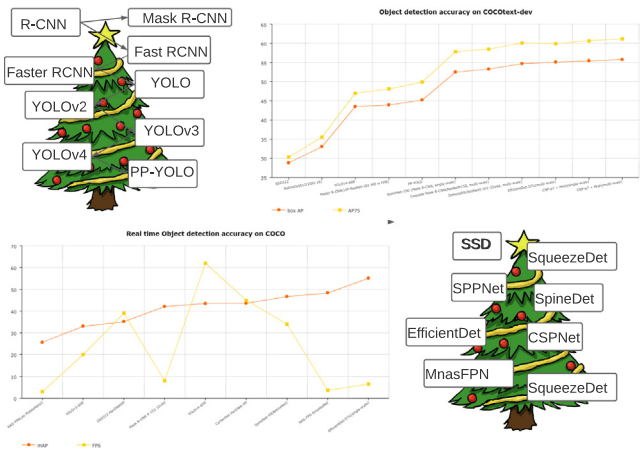
Andrei-Alexandru Tulbure [a,c], Adrian-Alexandru Tulbure [b], Eva-Henrietta Dulf [a,*]

[a] Department of Automation, Technical University of Cluj Napoca, Romania
[b] Department of Engineering, University of Alba Iulia, Romania
[c] Vision Tech Research SRL, Alba Iulia, Romania

HIGHLIGHTS

- A comprehensive analysis of modern object detection models
- Study on models that can be used as detectors for defect detection applications in industry.
- Study on YOLOv4 that can perform good defect detection with not much capital investment.
- Analysis on the correlation between dataset, labeling and the data augmentation steps and accuracy and computations.
- Analysis on the importance of correct data acquiring, augmentation and labeling in low cost applications.
- Analysis of the rate of improvement of the mAP of defect detection and image classification systems in recent years.
- Analysis of model compression and acceleration on embedded applications and smart factories.

GRAPHICAL ABSTRACT



ARTICLE INFO

Article history:
Received 4 February 2021
Revised 10 March 2021
Accepted 31 March 2021
Available online xxxx

Keywords:
Defect detection
Object detection
Image classification
Deeplearning
Deep convolutional neural networks

ABSTRACT

Background: Over the last years Deep Learning has shown to yield remarkable results when compared to traditional computer vision algorithms, in a large variety of computer vision applications. The deeplearning models outperformed in both accuracy and processing time. Thus, once a deeplearning models won the Image Net Large Scale Visual Recognition Contest, it proved that this area of research is of great potential. Furthermore, these increases in recognition performance resulted in more applied research and thus, more applications where deeplearning is useful: one of which is defect detection (or visual defect detection). In the last few years, deeplearning models achieved higher and higher accuracy on the complex testing datasets used for benchmarking. This surge in accuracy and usage is also supported (besides swarms of researchers pouring into the race), by incremental breakthroughs in computing hardware: such as more powerful GPUs(Graphical processing units), CPUs(central processing units) and better computing procedures (libraries and frameworks).
Aim of the review: To offer a structured and analytical overview(stating both advantages and disadvantages) of the existing popular object detection models that can be re-purposed for defect detection: such

☆ Peer review under responsibility of Cairo University.

* Corresponding author.
E-mail addresses: andrei.tulbure@visionrotech.com (A.-A. Tulbure), eva.dulf@aut.utcluj.ro (E.-H. Dulf).

https://doi.org/10.1016/j.jare.2021.03.015

as Region based CNNs(Convolutional neural networks), YOLO(You only look once), SSD(single shot detectors) and cascaded architectures. A further brief summary on model compression and acceleration techniques that enabled the portability of deeplearning detection models is included.

Key Scientific Concepts of Review: It is of great use for future developments in the manufacturing industry that many of the popular, above mentioned models are easy to re-purpose for defect detection and, thus could really contribute to the overall increase in productivity of this sector. Moreover, in the experiment performed the YOLOv4 model was trained and re-purposed for industrial cable detection in several hours. The computing needs could be fulfilled by a general purpose computer or by a high-performance desktop setup, depending on the specificity of the application. Hence, the barrier of computing shall be somewhat easier to climb for all types of businesses.

Introduction

In recent years computer vision has been transformed by the development of deep learning algorithms and by the multitude of GPUs that entered the market, enabling powerful and large scale computations. Pre deep-learning, one needed to have lots of knowledge in feature mapping and feature description, in order to be able to hand craft features and detect object from images. Instead, deep learning enables computational models of multiple layers to represent and learn features and data with lots of layers of abstraction, somewhat copying the brain's operational structure: how it perceives and understands information and how it creates structure. The most important deep-learning based computer vision methods are convolutional neural network(CNN) based ones. Moreover in [1] the details of CNNs are specified: type of layers used, training methods, as well as, more deep-learning methods are presented: Deep Boltzmann machines, deep belief networks, restricted Boltzmann machines and auto-encoders. For a timeline of the history of computer vision, [2] has a great level of summarizing the information, starting from the early trials of original Perceptron of 1957 until the recent revolution of deep-learning, as well as the hardware development implications. The two main applications of deep-learning based computer vision are image classification and object detection, which can or cannot involve semantic segmentation. For a further summary please refer to [3,1], where applications such as: Face recognition, pedestrian detection, activity recognition, human pose estimation and others are described. In image classification one just classifies the object in the image into predefined classes, which is a supervised type of process. In object detection and the subsequent segmentation, one needs to identify the location of the object, maybe track it in the image and then determine the class of the object. A comprehensive review for object detection is [4], which thoroughly discusses the subject mentioning the new industry as a whole, starting from non deep-learning object detection methods such as Histogram of Gradients [5] and also discussing in detail the modern approached such as RCNN(region based CNN) [6] and YOLO(You only look once) [7].

Judging by the fact that object detection became more and more accurate in the latter part of the 2010s, multiple industrial applications evolved from it, one of which is defect detection in industrial processes, mainly by optical inspection. Such a system is composed out of a sub-system that acquires the images (e.g. a camera system) and from the processing sub-system (e.g. a compute on the edge module or a connected workstation)[Fig. 1]. The need for such an inspection process is due to the poor overall reliability of human optical inspection and the high costs for such operations. In order to perform optical inspection every worker needs to be trained intensively and, more importantly, she/he cannot rush the process or else risk the quality reputation of their employer. Also, fatigue can be a big problem when ensuring quality only by the means of trained workers. Hence, the need for a cheaper, faster and more reliable method of performing optical inspection arises. Before the advent of deep learning and especially convolutional neural networks (CNNs), traditional computer vision algorithms (e.g. Viola-Jones [8] or Histogram of oriented gradients [5]) yielded satisfactory results in some processes as optical character recognition (OCR), edge detection, thresholding, color recognition or template matching. The majority of these operations imposed constraints on the manufacturing process and on the object it inspected. The most common constraints were regarding the objects orientation, the lighting conditions and the speed at which the object can travel in the production line (or the lack of speed). As one can see, these constraints are not welcomed when it comes to profitability or the ease of work for human workers in the assembly line. Since 2012 when the work of Krizhevky et al. [9] won the ImageNet Challenge and achieved breakthrough results with their 5 convolution layer and 3 fully connected layer CNN, the applied research in the area of computer vision headed in this direction, yielding better and better results regarding accuracy and processing time. More and more research was done in this area in the last decade, ultimately lots of computer vision applications using neural networks appeared. This paper reviews the modern methods and neural network frameworks and states their advantages and disadvantages. Hence, methods used for embedded computing of neural network applications are highlighted because the goal is to outline the steps necessary for industrial use defect detection algorithms. These methods need to have low computing power requirements, low power usage, being embedded-device compatible and also being resilient and rugged. While the industrial ruggedness is not a characteristic of deep-learning algorithms, having a lean and low cost deep-learning algorithm that performs with high accuracy and high frame per seconds(FPS) rate, helps by enabling the finances to concentrate on that aspect of industrial use. Ultimately everyone should have access to the usefulness of deep-learning algorithms, not just the high capital businesses, who can afford to spend even large sums of money on a solution. Further the paper is structured as follows: the related work is briefly discussed in Section 2, a thorough review of modern CNN models used for object detection is given in Section 3 and a short summary for model compression for defect detection models is stated in Section 4. In Section 5 the performed experiments are stated and the datasets used for benchmarking, while Section 6 concludes the paper. (see Fig. 2).

Related work

Since the second part of the 20th century researchers are more and more interested in the area of image processing and extracting information from images because of the real world utility of these methods. In the last years, the rhythm of research is accelerating,

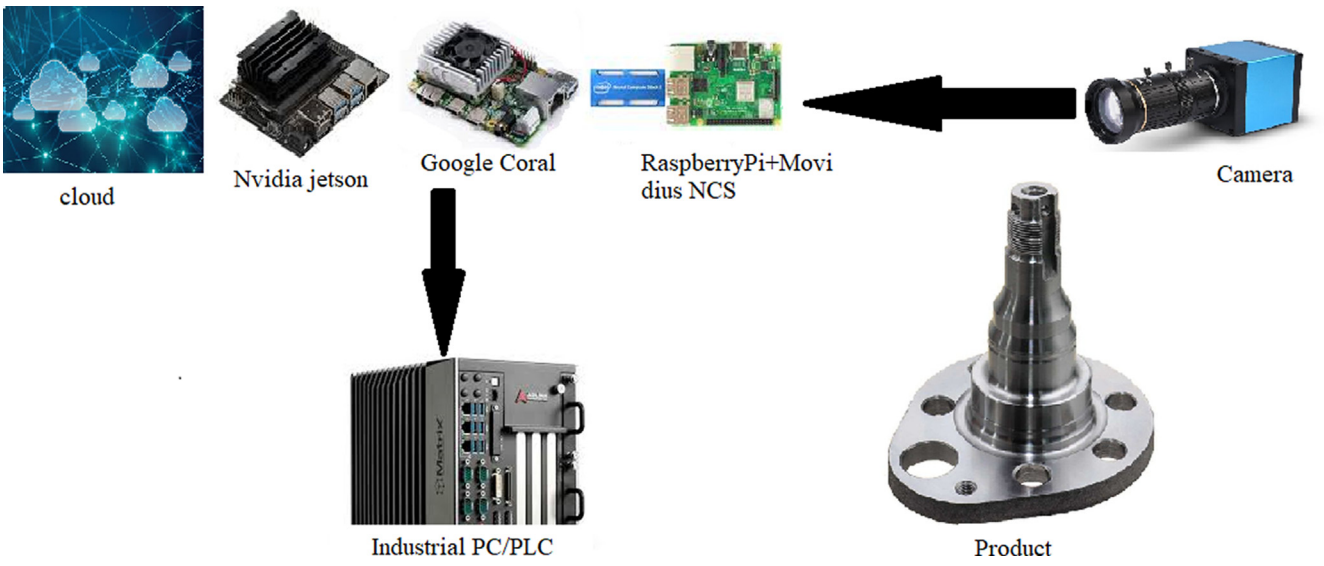


Fig. 1. System overview-concept-of an industrial camera quality check system in a production environment.

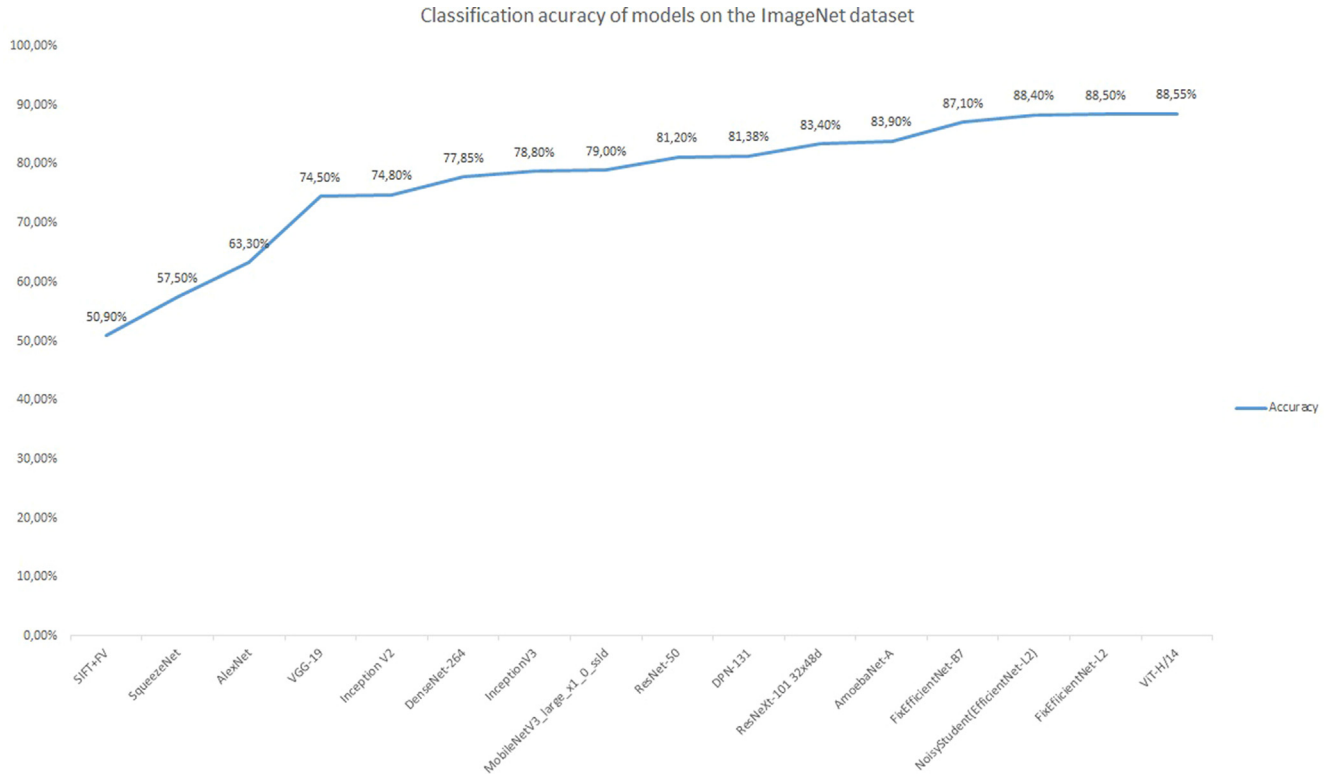


Fig. 2. Evolution of the Top1 accuracy of ImageNet ILSVRC contestants. As one can see, the high performing classification entrants are used as a backbone for detection models. The accuracy for Image classification tasks is higher than the one for detection, thus, one can conclude that detection models can be significantly improved or that the technology did not achieve "enough". The small SqueezeNet model, that fits into any embedded device has a Top1 accuracy of a bit more than 50%. The most accurate FixEfficient model yields almost 90%. Thus, one can easily see that the rate of improvement of Top1 accuracy is slowing down. The exact values can be seen in Table 1.

thus a new pallet of frameworks are presented and better performances are obtained.

Traditional computer vision or shallow methods

One of the earliest algorithms in image segmentation was the Snake algorithm (1987) that used energy minimizing splines guided by external constraint forces and internal image forces, that made localizing important features such as lines or edges more accurate [10]. Furthermore, in [11] a comparison of four texture algorithms (1980) is presented: spatial gray level dependence method(SGLDM) [12], gray level run length method (GLRLM) [13], gray level difference method(GLDM) [14], power spectral method (PSM) [15]. The advantages are compared and the results reflect that the SGLDM was the most powerful algorithm at that time for texture analysis. Another image processing method for feature extraction, advanced in the 1990s, were the discrete and continuous wavelet transforms. A recap of fast methods for

performing these computations was proposed in [16], implemented using Fast Fourier transforms (FFT) [17] and fast finite impulse response (FIR) algorithms. A more modern approach (2009) for wavelet transform is presented in [18]. In [19] a real time defect detection system for textile fabric was conceived. The system was based on multi-resolution decomposition and an artificial neural network(ANN) for defect classification. It yielded satisfactory results on low contrast defects in an industrial environment. Pulsed thermography based images were used for defect detection in a system, together with a perceptron neural network used for computing, in [20]. The point of interest was on aluminum corrosion. The system uses temperature, phase and amplitude data to locate the defects and showed promising results for the application on aluminum with a accuracy of up to 99%, in an restricted setup. Additionally, in [21] an optimisation method for ANNs was proposed. It was based on replacing back-propagation with the bees algorithm (population based search algorithm) and detected defects in wood. Although genetic programming methods are usually used for estimation and optimisation [22], these algorithms are somewhat intersected with classical computer vision methods and more research is needed in the respective direction. Still, the above mentioned methods are not based on deep learning and used shallow networks. The accuracy was around 86% in the test setup proposed by the authors. It needs to be mentioned that these genetic optimisation algorithms [22] did not outperform the classical back-propagation algorithm, hence the justification for its use is scarce. Furthermore, in [23] a cepstral approach for feature extraction (mostly composed of discrete wavelet transform(DWT) or discrete cosine transform(DCT)) was used on radiography images in order to detect welding defects. Their results were based on testing on only 16 images so the results need to be taken with a pinch of salt, as one cannot generalize from such a small dataset. Nonetheless they obtained up to 100% detection accuracy, so the result were promising. In [24] a mathematical method for revealing hidden and defective details in ancient manuscripts was used together with multiple ways of image acquisition (Thermography, X-ray, raman spectroscopy, multi-spectral imaging, laser induced breakdown spectroscopy). Several case studies were presented and the result were promising. An apple defect detection method based on a shallow MLP-Neural Networks was presented in [25]. The main purpose was to detect defect in two classes of apples and the features extracted were color, texture and wavelet features. Also, principal component analysis(PCA) [26] was used for dimensionality reduction and the best accuracy obtained on the scarce dataset provided for the evaluation of the method was of 89.9 %. Another example of a similar method is proposed in [27], where images are acquired by active thermography(temporal thermograms) and the processing is similar to the previous method, as it also uses a shallow network and PCA in order to detect the defects in materials and estimate their depth. Moreover, in [28] a case study is presented for designing a defect detection system using 90s era techniques. A brief survey of these older feature extraction (and image processing) methods can be found in [29]. Furthermore, it needs to be mentioned the fact that traditional images processing methods do not use many standardized test dataset like COCO(Common objects in context) or ImageNet and not all traditional classification algorithms are evaluated on modern benchmarking datasets. So, in order to compare these methods, if benchmarking results are not available, one needs to perform the experiments and benchmarking on their own.

Deep learning era applications

Since the early 2010s, the research in the direction of applying deep learning in order to solve computer vision problems is accelerating. Some vision problems include: object detection, object tracking, image classification, semantic segmentation, feature extraction etc. The main topic of interest for defect detection is object detection, because defects are treated as objects and they need to be localized and subsequently classified. For more details on object detection this above-mentioned review offers a comprehensive overview [4].

A great example of the usefulness of deep learning defect detection algorithms is presented in [30], where one can see that these methods don't only offer an accuracy increase, but they also offer a great speed advantage over traditional methods or even human defect inspection. The authors used the YOLO-v3 network model framework [31] on which they performed the following improvement: they integrated the MobileNet framework [32] to lighten the YOLO-v3 network and the results stated that it performed with similar accuracy as the original YOLO-v3 network, but with a significant speed-up of detection. It outperformed other methods, like SSD [33] and Faster-RCNN [34] for the task of detecting electronic components. Also, it must be stated that in order to make the algorithm perform with a smaller database of pictures, data augmentation techniques were applied: blurring, brightness manipulation, adding noise and contract enhancement. Cropping and rotations were not used for this application. As mentioned in their paper [30], data augmentation techniques show a general improvement in accuracy in every neural network framework. Moreover, by making the network lightweight they achieved both, almost real time processing and portability. Both characteristics are very important for ensuring defect detection for industrial processes at a reduced price. Another example of a low cost defect detection method that replaces human inspection can be found in [35]. The author used a small VGG [36] inspired network model with small convolutional filters to detect faulty bearings. The dataset was small and specific and built with lots of data augmentation. Also, the process is an industrial one, with the bearings coming into the station in the same place every single time.

An up-to-date topic is the defect detection in Lithium-Ion batteries which power the world that we live in today. As one can see in [37], the authors used a deep learning model to inspect the quality of li-ion battery electrodes from light-microscopy images. A lightweight YOLOv3 version named YOLOv3-Lite is presented in [38]. The network's purpose is to detect cracks that formed in an aircraft structure, thus improving the maintenance operations (as one can see it is a common theme for defect detection applications to assist humans in maintenance operation). They combined the basic YOLOv3 detection framework with depthwise separable convolution and feature pyramids. Depthwise separable convolution is responsible for reducing the parameters and extracting crack features effectively and is found in the backbone network. The feature pyramid's role is to join together low and high resolution features in order to obtain more robust classifier semantics. They evaluated the method on fuselage, wing, aircraft tail or engine blades and they obtained 50% better speed than the basic YOLOv3 [31]. A bottleneck that was stated is the fact that the dataset needs to be created and the labeling process can be quite hard, because you need experienced employees to fulfill this part. Their dataset only contains 960 pictures in total, which when compared to other datasets that are used in the community for benchmarking is a very small set.

In [39] the authors developed a deeplearning model for detecting defects in printed circuit boards(PCBs), that got up to an accuracy of 98.79% by training it on a TITAN V GPU. This model is of great help for the industry, because in the case of PCBs the quality inspectors need to be very well trained and cannot process tons of boards a day. Even though it is a particular solution, their dataset is quite big, containing 11000 input images of size 420x420 (which are resized to 416x416 to comply with the detector frames). They chose to implement their solution, by building it on the YOLOv2

[40] framework because of the fact that YOLOv2 has pretty low recall when compared to other public methods at the time of publishing the piece of work. In order to help with collecting all the images necessary, they developed a small applet, that allowed quality inspectors to easily label lots of data efficiently, thus providing a bridge between the quality inspector and the AI engineer. It is of no use to develop complex network models, if the dataset is biased and of poor quality. The final statement suggests that transfer learning may be of future use for defect detection systems.

The authors of [41] presented a method for detecting defects in steel strip surfaces, in real time, by using a modified version of the YOLO framework [7]. The surface defects are hard to detect because of the fact that they are not the same for different production lines and the surface of the steel also presents non-defective interferences, like oil. Thus, the detector needs good generalization. The dataset they compiled is for six types of defects, a total of 4655 images. To evade overfitting and to improve the performances, they used data augmentation techniques. This is a consistent evidence found in detectors: data augmentation improves the accuracy. Furthermore, they obtained great results, on a particular solution they obtained up to 99% detection rate, with 95.86% recall rate, all while their detector is running at 83 FPS. They modified the YOLO model, by making it fully convolutional, hence allowing the network to learn the spatial down-sampling, without pre-defining it. In order to also detect smaller defect, they added higher resolution feature maps(besides the one stated in the YOLO framework) in the feature extracting backbone. The training procedure consists of 5000 iterations, with all the hyper-parameters described in the above mentioned paper. This is not a really expensive solution, judging by the fact that they trained it on a GTX 1080Ti GPU and it took only 12 h. Moreover, they also predicted and detected the location and scale of the defects and thus, they are able to state which production line does not meet the quality criteria and subsequently, improve it.

Also, in [42] a SSD detector is developed in order to detect defects in wood. Even though they used it for a particular case of detecting defects in the Akagi and Pinus sylvestris trees, they obtained up to 96.1% mean average precision, which is great. Their dataset was quite specific, containing only 500 200x200 images of defective parts: wood knots, dead knots etc. They chose to implement their solution based on the SSD detector [33] because the constraints in the wood factories were challenging: 50 meters of wood per minute and greater than 95% accuracy. Hence, they used a new backbone: a DenseNet [43] backbone, used to extract better deep features and avoid the problem of vanishing gradients. Moreover, they also used a feature fusion function of the SSD detector, in order to fuse the multi-layer feature map obtained by the backbone for the regression of the position parameters and classification of the wood defects. Also, they adapted the input the SSD accepts to 200x200 and when multi-layer feature maps were merged, one layer of the previous feature maps was reduced, hence reducing calculations and getting closer to the speed requirements. Their final detection time was 56 ms per image, thus a frame per second (FPS) rate of about 20.

In [44] the authors proposed a system for detecting defects in power line insulators by using images acquired by a drone. They trained a cascaded architecture of CNN based detectors that yielded great results and may be replicated for further aerial quality inspections.

In [45] a method for detecting defects of fasteners on the catenary support device (for electrical trains) using a deep CNN is presented. Fasteners are loosened by the vibration and excitation over the long term operation. The quality inspection is done by specialized operators and it is a long labor intensive process. A solution was needed, so the authors implemented automated process by using a three stage cascaded detection architecture in a coarse-to-fine manner: the cantilever joint detector is implemented using the SSD framework, the localization of fasteners is done by using the YOLO framework and the classifier for diagnosis is based on a medium scale deep CNN, smaller than AlexNet [9], but with similar performance. The results are encouraging and thus, present the opportunity to reduce the human labour needed for maintenance and enable the system to process lots of data accurately. These improvements yield a reduction in the cost of maintenance and also a positive change in the safety of operation of the high speed trains. Hence, when lots of large images are processed a cascaded architecture is suitable, the drawback being the fact that it's very hard to ensure the embedded compatibility and it is not trivial to train such a large network model.

Another application is mentioned in [46], where an automated detection and classification system is presented. It classifies infrastructure surface images that contain defects(cracks, water leakages etc.), thus boosting the maintenance process efficiency. The question raised by the authors is regarded to the fact that structured and reliant datasets are hard to obtain for certain real world applications such as this one. For the classification and detection of surface defects they used a deep residual network. The novelty lies in the fact that they used an active training strategy: the network is trained as soon as labeled images become available and then it is used to select the most informative images. Only for these images, labels are queried from the experts to train the network. The results are encouraging, they obtained 87.5% detection accuracy. When compared to baselines methods, the one developed with active learning and positive sampling is obtains slightly better accuracy and converges faster, thus saving both computation time and cost.

In [47] the authors presented a solution for surface defect detection, solution which is based on a compact CNN. The reasoning behind this choice is the fact that the over-reliance of defect detection algorithms on GPUs for computations hindered the deployment of deep learning in manufacturing processes. Thus, a compact CNN that can run on embedded device's CPUs(central processing units) can be a great solution. They did achieve comparable results to MobileNetV2 [48], while only having 33% of the FLOPs (Floating point operations per second) and 1/8 of the weights. This means it can classify and segment correctly surface defects in 30 ms on an common use CPU like Intel i3-4010U. In order to get to the final solution they used transfer learning, model compression and a compact design. Together with a architecture design based on depth-wise convolution and a light wight backbone they obtained 99.29 % accuracy with only 60 k weights and only 76 million FLOPs, while ResNetV2-50 [49], with 16082 million FLOPs and 23.65 million weights obtained only 99.60%. Also, when taking into account processing time, their model performed the detection and classification in 75 ms, while ResNetV2-50 did it in 893 ms. For comparison, MobileNetV2 [48] did it in 259 ms, while obtaining 99.63% accuracy, with 267 million FLOPs and 480 k trainable weights. All three techniques will be discussed in the next section. For a detail on the architecture please refer to the above mentioned paper.

Defect detection deep learning frameworks

Since 2012, many deep learning detection frameworks were developed, every year bringing new ideas and new developments. The more modern ones, often can use different networks are backbones, leading to a high flexibility scenario. In the Figs. 3–6, one can graphically see the performances of some of the best performing modern models of classification, detection, real time detection and get an overview of them. The intuition that lead to fast development is the fact that the human eye object detection is a

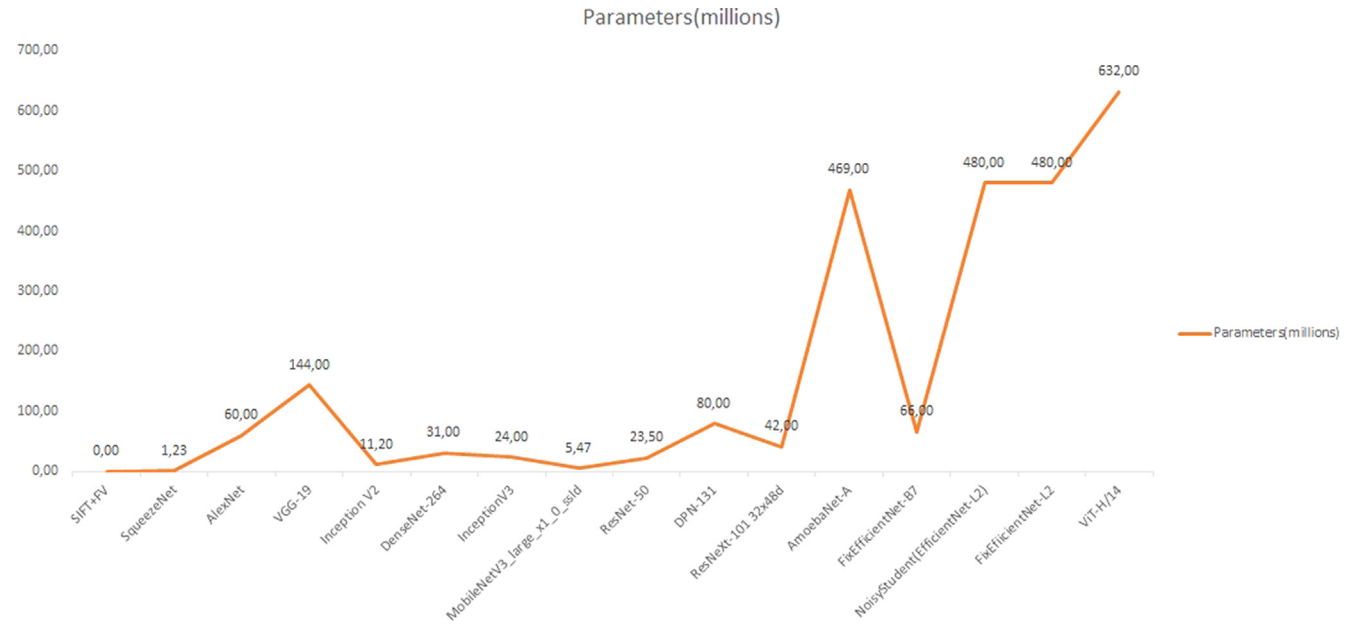


Fig. 3. Graph of the number of parameters(in millions) of some of the best performing Image classification models on ILSVRC. As in contrast with Fig. 2, it's easily picked up the fact that the more parameters one network has, the better its accuracy is. It's not mandatory, but it is a general fact. There are also smaller networks that perform well (eg. ResNet 50 is smaller than VGG19 and performs better. And as a general trend, one can see that more modern models have a parameter number that grew exponentially. The exact values can be viewed in Table 1.

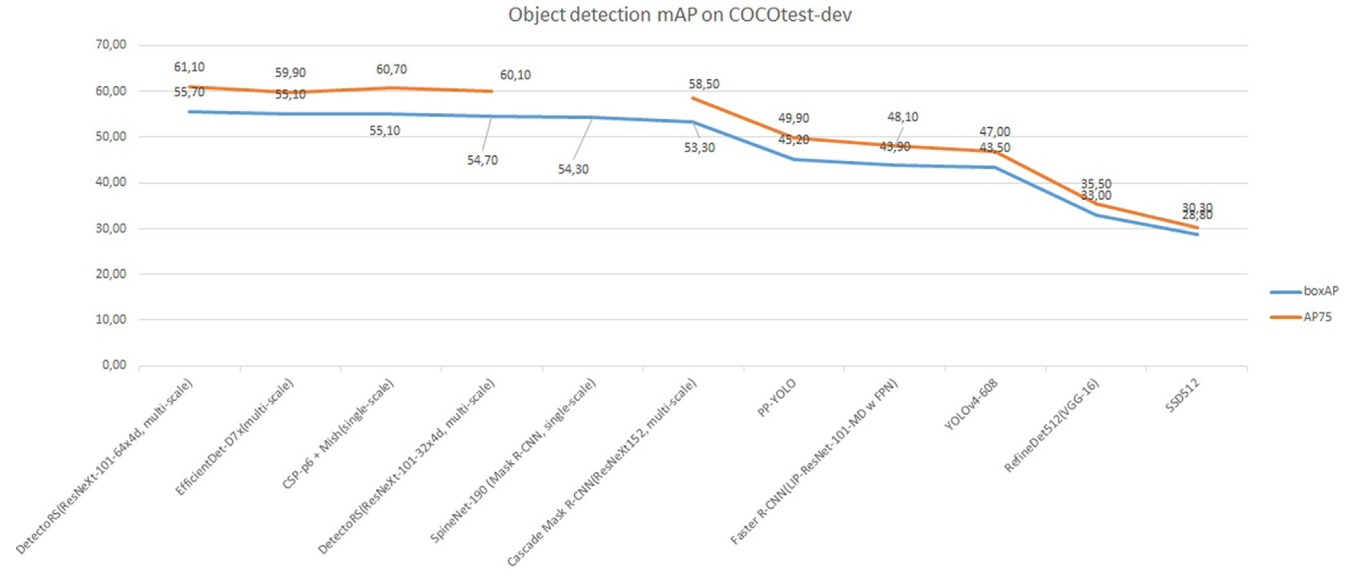


Fig. 4. Graphical representation of some of the better performing detection models on the Microsoft COCO dataset. The values can be found in Table 2. As one can see, that generally the AP values stay the same, no matter the IoU threshold we choose. And another general fact is the rate of improvement of detectors using conventional training and design, slowed down in recent years, passing by the mark of 55% boxAP. The most accurate ones are also the largest ones, thus in order to enable embedded devices like cars, to benefit from the technology, more and more progress needs to be done in the sphere of efficiency. The YOLOv4 detector chosen by us to do some experiments is one of the best performing and also, some of the more accurate.

multi-stage process and not a single step is responsible for the whole detection. Thus, these directions were adopted. In Table 1 the best performing classification neural network models on the ImageNet database are presented. Starting from the pre-deeplearning era of the SIFT model [75] (Scale invariant feature transform) that yielded a Top1 accuracy of 50.9%, up to the most efficient CNN model(as of 25th of October 2020), the FixEfficientNet-L2 [76] that yields 88.5% Top1 accuracy. Also, one can easily see that generally more parameters mean a better accuracy performance of the network, but this is not an inherent rule of thumb for convolutional neural networks! There can be a smaller, more efficient network that displays better accuracy than a larger, more clogged one (see the comparison of ResNet-50 [49] and VGG-19 [36] CNN models). Also, some of the most popular classification models that are used as backbones in many defect detection applications are remarked such as the VGGNet model, the SqueezeNet model, ResNet and ResNeXt and MobileNetv3 [77]. Having a strong backbone is one of the key design choices for a high accuracy defect detection model. Some of the more popular and best performing object detection models are presented in Table 2. AP75

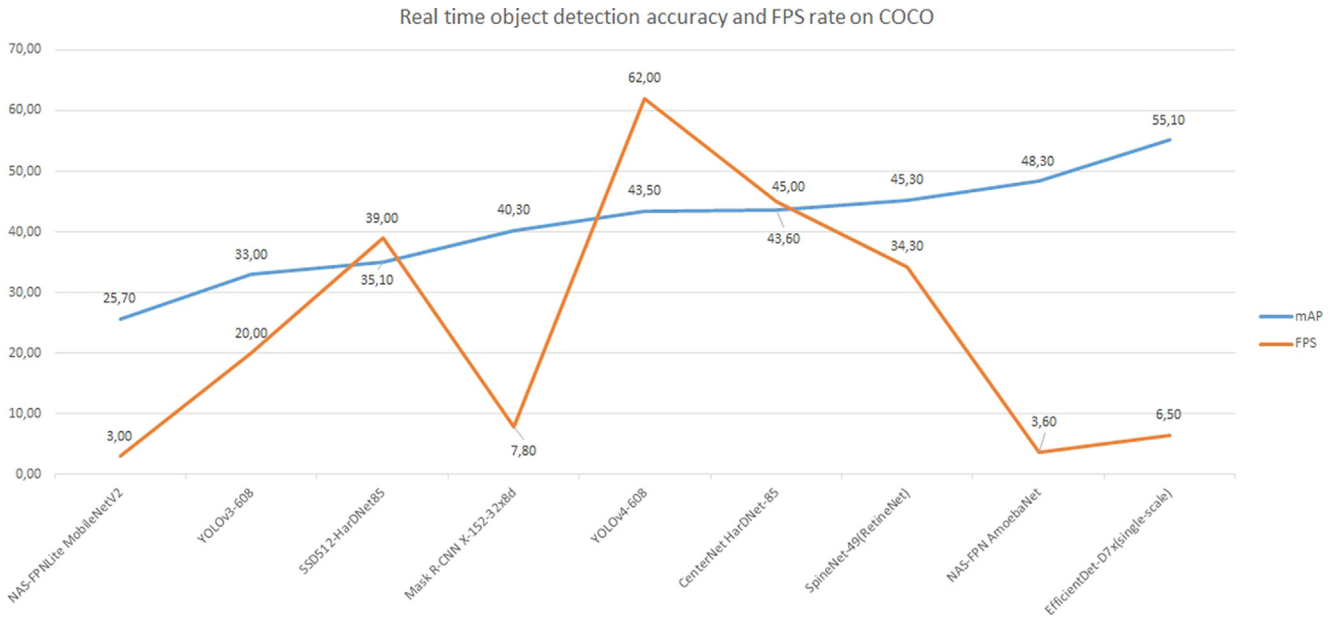


Fig. 5. Most important for embedded devices is the real time performance of an object detector, as it is trivial to why. Encouraging is the fact that, the performance of real time object detectors does not seem to have significantly slowed down and a great chunk of progress is still possible. We could consider real time everything that passes 30FPS, such that only a few models from the graph will be considered as such. YOLOv4 is the fastest model and it's accuracy is more than decent and given the easy training procedure it is a top candidate for defect detection models(as were their previous versions v2,v3 etc.). The values can be found in Table 3.

meaning the average precision when the IoU is 0.75, meaning that 75% of the prediction overlaps with the ground truth. Most of these models can be skewed in order to yield great defect detectors. Also, of interest is the frame rate of these detectors, in order to be able to pair the correct object detector with the correct application. The YOLO framework detectors are currently the fastest detectors with frame rates up to 120 FPS. As one can see, in Table 3, only real time detectors are presented and their performances listed. A "TRUE" Real time object detector means that the detectors runs at more than 30 FPS, but also slower ones are evaluated, as accuracy plays an important role too. A general consensus may the that, the slower the detectors, the more accurate it is, which can be categorized as a half-truth, as it is not always the case. mAP means the "mean average precision" and it states if the object is correctly detected. More precisely, it is the area under the precision-recall curve. Next, some discussion of the detection models is needed in order to state the differences, evolution and the practicality of each solution.

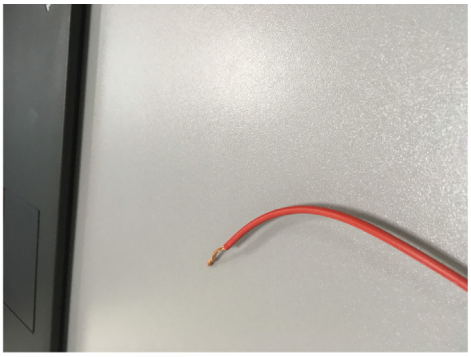
RCNN

A straightforward and scalable framework for detection is presented in paper [6]. At the time of publishing, it improved upon the state-of-the-art detection algorithm by more than 30%, getting up to an accuracy of 53.3% on the PASCAL VOC 2012 dataset and 31.4% on the ILSVRC2013 detection dataset. In order to achieve this performance, the method made use of two important insights. First, to localize and segment objects, a high-capacity CNN is applied to bottom-up region proposals (which are extracted before; in this case around 2000) and then the features were computed for each proposal. Moreover, a type of effective "pretraining" is presented, where the network is trained in a supervised manner on abundant (labeled) data and then after that, the net is fine-tuned for the specific task, which in this case is object detection. Finally, the classification part for the regions is handled by a class-specific linear SVM(support-vector-machine) [78]. For a more simplistic explanation, in order to detect a specific defect in an industrial process, one can simply train a CNN on a large labeled classification dataset that is available online and after completing this step, the fine-tuning on the target task is performed with the scarce labeled data from the industrial process in question. After it was proved that this is an effective approach, the further developed frameworks, all encapsulated some kind of pre-training.

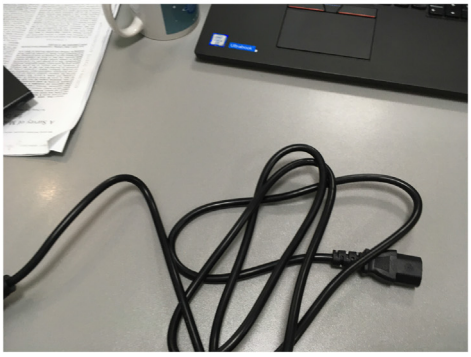
SPPNet

In [79] the authors presented a new framework for deep convolutional neural networks aimed at the task of object detection. The intuition behind developing this model was the fact that previous frameworks needed fixed size input images, a constraint that was artificial. Thus, they developed a new pooling strategy called Spatial pyramid pooling to tackle the above mentioned problem. It can create a fixed length representation, no matter the image size or image scale. Additionally, it is robust to object deformations or pose modifications. For detection, the feature maps of the image are only computed once and then features of random regions(ROIs) are pooled to generate the fixed length representation for detector training. Thus, the repeated computation of convolutional features is eliminated and it improves the overall speed performance. The novelty being the spatial pooling layer(which generally replaced the last pooling layer), it is a layer that improves the Bag-of-Words approach (that generates fixed length vectors for the softmax classifier) by the fact that it can maintain spatial information by pooling in local spatial bins. Moreover, these spatial bins have sizes proportional to the input image size, such that the number of bins is fixed regardless of the size of the image. Hence, one can see that the performance improvements are straight-forward, since older methods relied of sliding windows that were not limited number wise. The larger the image, the more windows the method presented. The outputs of the pooling layer are vectors that have the dimensions of the number of filters times the number of bins. These vectors feed into the last fully connected layer and the classifier that performs the final step. The accuracy obtained on the PASCAL VOC 2007 dataset was 60.9% mAP (best one), while on ISLVRC 2014 it obtained 35.11% mAP, coming in second.

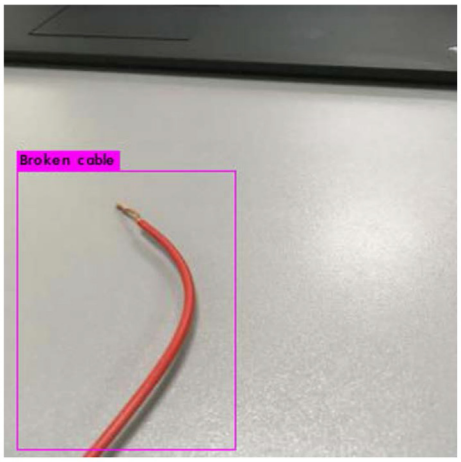
(a) Sample in the dataset for a red broken cable used for electrical circuits.



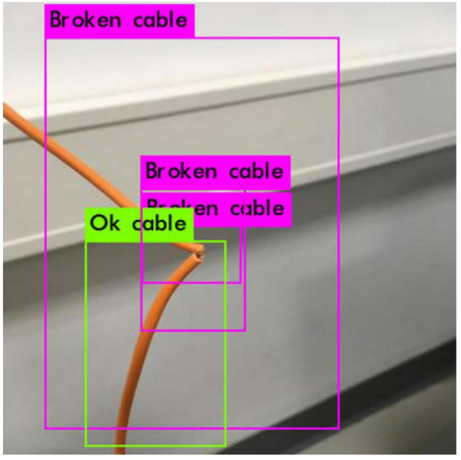
(b) Sample in the dataset for a red broken cable used for electrical circuits.



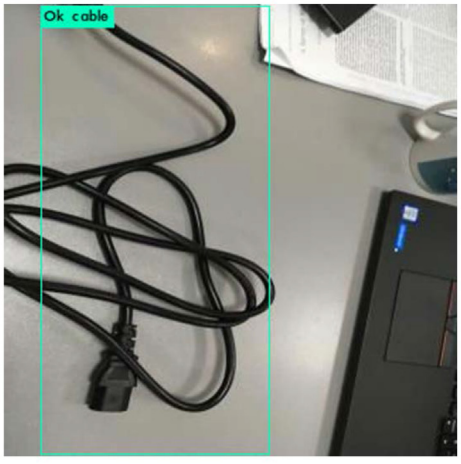
(c) A sample in the dataset for a screen socket connection cable. Multiple types of cables were added in the dataset because of the unknown variables of factory cables. So it's better to have more types of cables than less types.



(d) Result of the detection model for detecting faulty cables. As one can see this is a easy case because the cable is not covered by another object in the images and it is pretty close to the camera.



(e) Result of the detection model when labeling multiple sub-parts of the object. Hence, this type of actions can reduce the accuracy and are not good practice



(f) Result of the detection model for an unusual type of cable.

Fig. 6. Results from the experiments using YOLOv4 as a defect detector for irregular shaped object that can not be found in a benchmarking dataset like COCO.

Fast-RCNN

In [80] the author presented an incremental improvement of the above mentioned method, speeding up the training and detection process: training the VGG16 network 9 times faster than RCNN, detection speed is up 213 times and the mAP(mean average precision) is higher, when evaluated on the PASCAL VOC 2012 dataset. The main intuition behind this improvement is the fact

Table 1
Table for performance values for the network models evaluated on ImageNet database, starting from 2011 with traditional image classification algorithms and continuing with deeplearning based classification algorithms from 2012 on-wards until 25th of October 2020. Thus, one can easily see the Top1-accuracy(correct class predicted) increases with every single iteration of an improvement or a new model framework. Also, the number of parameters of a network, which directly implies the number of computations and the memory needed is represented in column number 3. Usually, when a network has more parameters, it's accuracy tends to go up, but it is not mandatory as inefficiencies can easily occur.

Network models evaluated on ImageNet database (2011-)		
Network model name	Top1-Accuracy	Nr of parameters (millions)
SIFT + FV [50]	50.9%	0
SqueezeNet [51]	57.5%	1.23
AlexNet [9]	63.3%	60
VGG-19 [36]	74.5%	144
Inception V2 [52]	74.8%	11.2
DenseNet-264 [53]	77.85%	31
InceptionV3 [54]	78.8%	24
MobileNetV3-large-x1-0-ssld [55]	79.0%	5.47
ResNet-50 [49]56	81.2%	25
DPN-131 [57]	81.38%	80
ResNeXt-101 32x4d [56]	83.4%	42
AmoebaNet-A [58]	83.9%	469
FixEfficientNet-B7 [59]	87.1%	66
NoisyStudent(EfficientNet-L2) [60]	88.4%	480
FixEfiicientNet-L2 [59]	88.5%	480
ViT-H/14 [61]	88.55%	632

Table 2
Table for object detection performance values for the network models evaluated on COCOtest-dev database (starting from 2016 with SSD512). Thus, one can easily see the box average precision(correct class predicted and bounding box that encapsulates the class is correct) increases with every single iteration of an improvement or a new model framework, which is party enabled by the increases in performance of image classification algorithms. Generally, when a network has more parameters, it's accuracy tends to increase, but it is not mandatory as inefficiencies can easily occur. AP75 is the average precision of the method when the Intersection over Union value of the detection is 0.75. This means that at least 75% of the prediction overlaps the ground truth. We need this kind of accuracy for defect detection because the process needs high accuracy and reliability in order for it to replace traditional industrial inspection methods.

Network models evaluated on COCOtest-dev object detection database (2013-)		
Network model name	box AP	AP75
SSD512 [33]	28.8%	30.3%
RefineDet512(VGG-16) [62]	33.0%	35.5%
YOLO-v4-608 [63]	43.5%	47.0%
Faster R-CNN(LIP-ResNet-101-MD w FPN) [64]	43.9%	48.1%
PP-YOLO [65]	45.2%	49.9%
Cascade Mask R-CNN(ResNeXt152, multi-scale) [66]	53.3%	58.5%
SpineNet-190 [67]	54.3	
DetectoRS(ResNeXt-101-32x4d, multi-scale) [68]	54.7%	60.1%
EfficientDet-D7x(multi-scale) [69]	55.1%	59.9%
CSP-p6 + Mish(multi-scale) [70]	55.2%	60.7%
DetectoRS(ResNeXt-101-64x4d, multi-scale) [68]	55.7%	61.1%

that the "multi-stage" pipeline is slow, thus the object detection is also slow, so it is rational to improve this aspect by developing a "single-stage" pipeline. Fast R-CNN is capable of classifying objects and finding their location in the frame within the same pipeline. The image (frame) and ROI(region of interest) proposals are funneled as inputs for the processing algorithm. Every region is then passed through a feature map of fixed dimensions and is then mapped intro a feature vector by the means of fully connected layers. As outputs, the method returns two vectors, for every ROI there are the bounding box offsets and for every class there are the class probabilities computed through a softmax function. At training, this framework takes into consideration a multi-task loss that tries

Table 3
Table for real time object detection performance values for the network models evaluated on COCO(real time) database (starting with MASK-RCNN in 2017). Thus, one can easily see the mean average precision(correct class predicted and bounding box that encapsulates the class is correct) increases over time, which is mostly enabled by the increases in performance of image classification algorithms and object detection algorithms. Generally, as previously specified, when a network has more parameters, it's accuracy tends to increase, but it is not mandatory as inefficiencies can easily occur. Thus, the FPS rate is usually mirrored against the mean AP. For defect detection in real time (e.g.: for Airplane inspections) we need this kind of accuracy for defect detection because the process needs high accuracy and reliability in order for it to replace traditional industrial inspection methods and also, we need high speed of processing because maintenance operations cannot take more than a couple of hours when flights are scheduled daily.

Network models evaluated on COCO real time object detection database (2017-)		
Network model name	mAP	FPS
NAS-FPNLite MobileNetV2 [71]	25.7%	3
YOLOv3-608 [31]	33.0%	20
SSD512-HarDNet85 [72]	35.1%	39.0
Mask R-CNN X-152-32x8d [73]	40.3%	3
YOLOv4-608 [63]	43.5%	62.0
CenterNet HarDNet-85 [72]	43.6%	45.0
SpineNet-49 [74]	45.3%	29.1
NAS-FPN AmoebaNet [71]	48.3%	3.6
EfficientDet-D7x(single-scale) [69]	55.1%	6.5

to optimize not only by taking into account accuracy, but also other aspects such as speed. Next, it makes use of fine-tuning for detection and, in order to comply with the findings of R-CNN [6], pre-trained networks are heavily used. Comparing it with other methods developed in the same year, one can state that the speed gains are great and despite this fact, the accuracy is also better. On the MS-COCO, the method returned 19.7% mAP with the IoU threshold mandate, while on the PASCAL VOC 2012 dataset, it obtained state-of-the-art 68.4% mAP. gether with the region based CNN approach, that got us closer to high accuracy real time detection.

Faster-RCNN

A new and faster incremental improvement of the Region based CNN framework is presented by the authors in [34]. The new method states that the previous method's bottleneck in the region proposal part(the algorithm was slow) was at fault for the "lack" of performance and not the CNN computations. Such that the concept of Region proposal network(RPN) is introduced, which is a network that is responsible for the region proposal computation(via an "attention mechanism") and also for the object detection. Thus, an almost zero supplemental cost is obtained for the region proposal part. The RPN returns the objects limit coordinates and also its objectness score, both of which are of high quality. An RPN is a fully convolutional network(FCN) that is trained end-to-end. The anchors used are both translation invariant(which reduces the model size) and multi-scale(by using pyramids of reference boxes in the regression functions), such that the algorithm is more robust to multiple scales and multiple aspect ratios. An almost real time detection performance was obtained, the detector being able to run at 5 FPS and with a state-of-the-art accuracy in the year of publishing on both ILSVRC and COCO datasets, while also achieving state-of-the-art on the PASCAL VOC 2007, 2012 and MS COCO (combined) datasets with only 300 proposals per image. The mAP is 75.9% with the VGG16 as a backbone network. When only testing on PASCAL VOC 07 + 12, the mAP is only around 70.4%, which means that training also on the COCO dataset increases mAP (this is intuitively true as a large dataset used at training time should increase accuracy if over-fitting is not attained).

YOLO

In [7] the authors propose a new method that achieves real time object detection, the detector running at 45 frames per second. They also have a smaller detection network running at 155 frames per second. When taking in account the accuracy/speed trade-off, the full-size detector is the best bet for object detection. The main intuition behind this framework is the fact that previous detectors re-purposed classifiers to perform detection and that made it harder and harder to optimize for accuracy and speed, thus a new method that frames object detection as a regression problem to spatially separated bounding boxes and class probabilities was considered. In this method, a single network predicts the bounding boxes and the probabilities directly in just one evaluation. This enabled to take into account the direct detection performance and made it easier to optimize the whole process, such that impressive speed gains were possible. But there is no gain without drawbacks and some of the YOLO methods limitations are: the model has a hard time detecting small object, especially the ones that appear in groups (eg. birds, fish) because of the strong spatial constraints applied on bounding box predictions because each grid cell predicts just 2 boxes and can have just 1 class; another drawback is the fact that the model struggles to generalize to object in a new setting, a new configuration or with an unusual aspect ratio because it learns to predict bounding boxes directly from data; and maybe the most important drawback is the fact that the loss function treats errors the same, no matter the size of the boxes, but generally a small error in a large box isn't anything to complain about, whereas a small error in a small box is something critical. When compared with other detectors on the PASCAL VOC 2012 dataset, one can see that when YOLO was combined with FAST R-CNN, the mAP was 70.7%, while the YOLO solo detectors had 57.9%. Not the best result in terms of accuracy, but when looking at the VOC2007 results, one can easily see that YOLO performs at a much higher FPS rate than other detectors.

SSD-Single shot detector

In [33] the authors present a novel method in the year of publication for defect detection. SSD is a simple framework that utilizes a single deep neural network and gets rid of the object proposal generation step and the additional feature resampling stages that other comparable state-of-the-art methods use, thus it is faster with great accuracy. It discretizes the output space of the subsequent bounding boxes into a set of default boxes over multiple aspect ratios and different scales per feature map locations and thus, at run-time it generates objectness scores for the presence of each object category in every default box. In order to improve the accuracy, the network produces adjustments to each box to better fit the shape of the presumpted object in the respective area. Moreover, the DNN combines predictions from multiple feature maps with different resolutions (ratios) to better handle objects of different sizes. The main benefits of this single net approach are speed, coupled with a resulting easy to train approach. Some of the design choices that best improves the accuracy were: data augmentation, using atrous convolution and including different aspect ratios for anchor boxes. The backbone network used is the same VGG16 net. As the results show, for a 300x300 image size, it achieves 74.3% mAP on the PASCAL VOC 2007 dataset @ 59 FPS, while on PASCAL VOC 2012 the same network achieves 77.5% mAP. When increasing the image size to 512x512 and using image expansion for augmentation purposes, one can see a slight bump in the accuracy, on VOC 2012 the accuracy increases up to 82.2%. On the COCO test-dev2015 dataset, the model obtained better results than the state-of-the-art Faster R-CNN on the IoU precision [0.5:0.95], getting a mAP of 28.8% and 48.5% @0.5, while on IoU 0.75, 30.3%. The conclusion is the fact that data augmentation methods help a lot in improving detection accuracy, while not costing any computation time as it is done a priori. This framework proved a concept and paved the way for further improvements, to.

SqueezeDet

When taking into account the real world applications of object detection, one cannot ignore autonomous driving. For this application to run smoothly and safely, the NN model needs to realize at least real time inference speed to ensure the vehicle can respond to any problems fast and safely. Furthermore, the model needs to have a small size (because of memory constraints), while also being very energy efficient(3 to 10 W range). Not to forget, even though it needs to run fast, the accuracy needs to be as close to state-of-the-art as possible. For all these, the authors propose in [81] a novel single stage method, called SqueezeDet, that is a fully convolutional network(FCN). This means that they not only use convolutional layers for feature extraction, but also in the output layer to compute bounding boxes and class probabilities. This leads to a smaller size and a more energy efficient run time performance. It is also very fast, because it only has one single forward pass of the neural network. The convolutional neural network first takes an image as an input and extracts low resolution, high dimensional features which are then fed into the ConvDet layer(which is a convolutional layer that works as a sliding window that moves through each spatial position on the feature map) to compute the bounding boxes. Each box has one confidence score for positioning and multiple class probabilities. The key is here to perform non max suppression(NMS) and to only keep the top bounding boxes with the highest confidence. Confidence is computed as the probability that there is an object in the box times the IoU of the prediction over the ground truth. This method of computing confidence is similar to other methods like YOLO [7]. The backbone of the method is the SqueezeNet [82] network (built upon the Fire module to ensure a small model size), because of its small size and energy efficiency. The detection performance attained on the KITTI dataset is of 80.4% mAP, with a model size of only 26.8 MB and a speed of 32.1 FPS. When comparing to other methods, this one is 19x faster than Faster-RCNN(with a VGG 16 backbone) and 61x smaller. The accuracy is on part with the above mentioned method. The KITTI dataset was chosen instead of the more used ISLVRC or PASCAL VOC or COCO because of the fact that the focus of this method is object detection for autonomous cars.

YOLOv2

In [40] the authors presented an improved version of the YOLO detector with the same speed advantage, but this time, at the time of publishing it had state-of-the-art performance on standard detection datasets such as PASCAL VOC and COCO. By the means of the multi-scale training method(that trains both on correct detection and classification) that they presented, the YOLOv2 model can run at different trade-offs: 76.8 mAP @ 67 FPS or 78.6% @ 40 FPS on the PASCAL VOC 2007 dataset. YOLOv2 has some elegant solutions for some of the original YOLO shortcoming, such as: Batch normalization, in all the convolutional layers, which helps with better convergence; increased resolution, such that for 10 epochs YOLOv2 trains at 448x448 resolution on ImageNet and the filters can adjust to high-res inputs; dimension clustering, instead of choosing priors for the boxes by hand, standard Euclidean k-means clustering is run on the training set bounding boxes to automatically find priors; direct location predictions, as YOLOv2 predicts five bounding boxes at each cell in the output feature map and five coordinates for each box(x,y,o,width and height) - center

coordinates and dimensions, instead of following the YOLO approach and predicting location coordinates relative to the location of the grid cell; Fine-Grained features, which means that YOLOv2 predicts detections on a 13x13 feature map, this being sufficient for large object and being helpful when localizing smaller object, helping with YOLO's drawbacks in this area; multi-scale training, at every 10 batches, the network chooses new input image sizes such that there are multiples of 32 because of the network downsampling factor of 32. In order to maximize also for speed, a new backbone is used called Darknet-19(19 conv layers and 5 maxpooling layers) that is similar to the VGG model as 3x3 convolutions are used and the number of channels are doubled after every pooling step. In order to make the network classify better, labels from WordNet are used as this implies a Hierarchical classification: WordNet is structured as a directed graph, not a tree and this means that a passenger car is a vehicle, as well as a truck. And this kind of structure is very helpful, such that a hierarchical tree was built out of this data. The datasets were combined with this hierarchical tree and a rich and powerful representation for training and detection was obtained.

Mask-RCNN

Further extending the region based CNN approach, in [73] the authors presented a new 2-stage method for object detection and instance segmentation. It is more straight-forward to train a CNN architecture than performs these both tasks in parallel(predicts the class and the box offsets, plus a binary mask) and obtains great results. It also generalizes well on other tasks like human pose estimation. It can run at 5 FPS and obtained for object detection on the COCO test-dev dataset 39.8% boxAP and 43.4% AP75. This is not real time performance, but it is good enough for a detector needed for industrial purposes or as a worker assisting tool. Mask RCNN extends the Faster RCNN framework by additionally including a branch for prediction an object mask in parallel on each region of interest. This model is simple to train and it incorporates a three part loss that takes into account the detection loss, as well as the mask loss. This leads to better performance. The main change is the fact that instead of the RoIPool operation for extracting small feature maps from each region of interest(RoI) in the Fast RCNN architecture [80], they present a new operation called RoIAlign that remove the hard quantization of RoIPool, aligning the extracted features with the input. For a more detailed explanation refer to [73]. This model used as backbones the ResNet 101-FPN backbone and the ResNeXt-101-FPN backbone, the latter one being more accurate(ResNet: 38.2% mAP and ResNeXt: 39.8% mAP).

YOLOv3

This one is the third iteration of the YOLO method, presented by the authors in [31]. It makes use of some small changes that are presented in the form of a technical update/report with respect to the original YOLO [7] method. With these changes they obtained similar accuracy to the state of the art methods, but run 3-8x faster, such that it still runs real-time detection. At 320x320 image size, it runs at 28.2 mAP in 22 ms, while SSD runs at 31.2 mAP at 125 ms. The state-of-the-art RetinaNet-101–800 runs at 37.8 mAP but only in 200 ms (9–10 times slower). The method is improved incrementally by: bounding box prediction by using logistic regression, such that if there is an IoU equal to 0, the detector doesn't get penalized for accuracy, even if the detection is false; and only the ones where an IoU = 1(or $\geqslant 0.5$) with a ground truth are penalized). That is great for the fact that if you do not have a pairing with a ground truth you do not lose accuracy. Thus, there is only 1 bounding box prior allocated for every ground truth and the possibilities of false positives that are a drawback on model's accuracy are reduced. Another improvement (for class prediction) is the usage of independent logistic classifiers instead of classical soft-max with binary cross-entropy at training time. Moreover, there are three predictions of bounding boxes at three different dimensions. This improves accuracy because of the improvement of aspect ratios covered. Finally, shortcut connections are introduced in the model, in the same way that residual networks are thought of [49].

EfficientDet

In [69] the authors systemically introduce a new detection framework design that brings efficient object detection called EfficientDet. It relies on a efficient backbone called EfficientNet [83], a weighted bidirectional feature pyramid network (BiFPN) and a custom compound scaling method. By employing these 3 optimizations, EfficientDet-D7 obtains state-of-the-art 55.1 mAP accuracy on MS COCO test-dev dataset with 77 M parameters and 410B FLOPs. Thus this framework is also smaller and with a higher accuracy than previous competitors. The D7(7 comes from the compound coefficient which controls all the scaling) version has an input size of 1536, a B6 EfficientNet backbone, the nr of layers for the BiFPN is 8, with 384 channels and the nr of layers of the box/class prediction part are 5. The backbone is an efficient network that obtains a high accuracy on the ImageNet dataset. For an in-depth analysis on the backbone, please refer to [83]. This is combined with a bi-directional feature pyramid network for multi-scale feature fusion (top-down and bottom-up). It aims to aggregate multiple features found at distinct resolutions, usually from levels three to seven in a top-down manner and from level seven to three in a bottom-up manner. These need to be weighted in order to enable the network to learn, which features are more important. The last piece is the compound scaling method that is needed when aiming to optimize for both speed and accuracy and a large pool of resource constraints. This scaling aims at, starting from the baseline EfficientDet model, increasing the resolution, depth and width for all: the backbone net, the FPN and the box/-class predictions, at the same time. When compared to other scaling methods, such as single dimension scaling, the first approach is more efficient than the latter, such that it will be employed.

CSPNet

In [84] the authors tried to tackle the problem of ever increasing computations of modern neural networks used for computer vision problems and came up with a slightly improved backbone that can enhance the learning and the performances of CNNs called CSPNet. Some approaches are developed already for mobile devices, but there is one problem, depth-wise separable convolution is not compatible with the industrial integrated circuit designs such as ASICs for edge computing. The proposed network model integrates feature maps from the beginning and the end of a network stage, which results in a reduction of computations by 20% for the following backbone nets: ResNet [49], ResNeXt [85] and DenseNet [43] and enable them to be deployed efficiently on mobile devices or CPU-only devices. First of all, the training process needs to be more efficient, as neural networks usually degrade in accuracy when they are transformed into lightweight models. Secondly, the computational bottlenecks need to be removed, as the goal is to evenly delegate and distribute computations at each of the CNN layers. This leads to a more efficient utilization rate of each computation unit and reduces energy consumption. The last challenge tackled by this model, was the fact that memory requirements are running sky-high for CNNs and also, accessing RAM is a very energy consuming operation when compared to only accessing on-chip Flash. For this, cross-channel pooling (or Maxout operation) was used to

compress the feature maps when generating the feature pyramids. By the means of the above mentioned processes, CSPNet model can enhance the accuracy of smaller models, by promoting learning capabilities and also increasing the efficiency of energy usage by the fact that the redundant gradient information is reduced, hence less costly inference computations. Experimentally, this model obtains state of the art performances when evaluated on the MS COCO object detection database, reaching 38.4 % AP with a ResNeXt50 backbone, while running at 35FPS on a 608x608 input image size. For more details on the structure of their cross stage layers, please refer to [84].

Mnas-FPN MobileNets for detection

A novel method is presented in the paper [86] that underlines the benefits of directly searching for object detection architectures in certain search spaces and not just transfer learned backbones from classification to detection. The proposed search space(it can be viewed as a detection framework, as it searches for detection heads and not backbones) is named MnasFPN and it is a mobile compatible one. MnasFPN constructs a detection net from a feature extractor backbone (a mobile friendly one in this case) and one repeatable cell structure that generates new features by combining and merging pairs of existing ones. Each cell is composed out of multiple blocks that merge two separate feature maps at different resolutions into an intermediate feature. This intermediate feature is then processed by a separable convolution and output by the block. The key characteristic is the fact that each cell shall use a collection of feature maps at different and various resolutions as an input and output it at the same resolution, hence this structure can be easily repeatable. MnasFPN differs from other NAS approaches mainly at the block level. For an in-depth analysis please refer to the paper above. MnasFPN makes use of inverted residual blocks at the detection head level which help with mobile CPU execution and it restructures convolution operations in the head and reshaping operations to enhance efficient merging of information across different scales and resolutions. Ablations studies prove that both of these actions are needed to acquire the stated performance indicators. When used together with a mobile compatible backbone as MobileNet [48] or [77], this method yields a state-of-the-art for mobile devices mAP of 26.1 % on the COCO test-dev dataset and uses 0.92B MAdds and 2.5 M parameters. The latency is up to 183 ms per detection. It is up to 1.8 mAP more than previous tries for mobile compatible devices using MobileNets and SSD detection head.

YOLOv4

The last improvement of the YOLO one stage detection framework was presented by the authors in [63]. The main goal was to obtain a more robust detector with a high accuracy, but maintaining the ease of training and the real time usage. The performance is of a high level, getting 38% mAP @ 120 FPS or 43% mAP @ 80 FPS on PASCAL VOC 2012, thus enabling real time detection with a high degree of accuracy. These kind of performances are needed for enabling real time connected factories or autonomous cars. On MS COCO the accuracy was 43.5% mAP and 65.7% AP50 @ 65 FPS on a Tesla V100. By the performance it has on different datasets we can assume that it is a more general kind of detector, being also applicable to general detection(not just standardized benchmark datasets). The framework has an easy training procedure, being able to be trained on a conventional GPU(1080Ti or 2080Ti) getting rid of some of the constraints that bigger models have: distributed training in parallel or large mini-batches of data. Quality detectors have a backbone that is trained on ImageNet dataset and a head which is used to predict classes and bounding boxes, thus this one is no different. Its backbone is CSPDarknet53 and the "neck" layer which collects feature maps from different stages is build from SPP (spatial pyramid pooling) and PAN(path aggregation network). The head is the YOLO v3 presented in a previous subsection. In order to improve on the previous iteration it makes use of the following procedures for the detector: CJoU loss (complete Intersection over Union loss), CmBN(cross mini batch normalization), DB regularization, mosaic data augmentation, self adversarial training, it eliminates grid sensitivity, multiple anchors for one ground truth, cosine annealing scheduler, optimal hyper parameters, random training shapes, mish activation(self regularized non-monotonic activation function), SPP block, SAM block(spatial attention module), PAN aggregation block, DIoU-NMS(Distance IoU non max suppression). For the backbone it uses: cutmix and mosaic augmentation, drop-block regularization, class label smoothing, mish activation, cross stage partial connection and multi-input weighted residual connections. For an in depth ablation study, more details and further experiments, please refer to the above-cited paper.

SpineNet used for detection

In [67] authors present a more robust and higher performing method of the original SpineNet model [74]. On the COCO test-dev dataset their model (SpineNet190 with self-training) obtained 54.3 % mAP with 164 M parameters and 1885B FLOPs. At a first glance one can see that this kind of model cannot be easily implemented on a mobile device, but there is no need to. The main characteristic of this model that the authors try to underline is the fact that sometimes pre-training(commonly used) and data augmentation can hurt accuracy. Replacing pre-training with self-training can yield a bump of up to + 3.4 mAP on the detection datasets across all sizes. Also, stronger data augmentation and better labels for the data, diminishes the inherent value of pre-training on ImageNet(or other dataset), thus caution needs to be exercised. But, unlike pre-training, self-training is helpful even when using a strong data augmentation, no matter the quantity of data. By their ablation studies, in the case when pre-training is helpful, self-training improves upon the initial results, thus the case for self-training initialization of detection models is made. These benefits pass also to semantic segmentation, where their SpineNet model achieved a state-of-the-art 90.5 mIoU. Their experiments also show that classification and self-supervised tasks are limited when it comes to learning universal representations from data, that may enable solving multiple tasks. The intuition lies in the fact that pre-training is not aware of the task of interest and hence fails to adapt. This awareness is very important when switching from classification to detection. A case is also made for joint-training, because in certain circumstances pre-training, self-training and joint-training can have additive effects on the performances of the trained model. Even though these approaches may require more compute power and more calculations, it is worth it when the accuracy needs to be maximized no matter the cost. For a more detailed analysis on the original architecture, but slower performing SpineNet model refer to [74].

PP-YOLO

3 aug 2020 In [65] the authors provide and propose a new and efficient implementation of the popular YOLO detection framework. It is based on the older YOLO-v3 [31] detector, but with a few key modifications that enhance performance and do not hurt inference time. The backbone used by the authors is ResNet-50-vd-dcn, because of the fact that many libraries and framework are optimized to process data with ResNet series and it is also one of the most commonly used, stating that these improvements

are independent of the backbone used. There needs also to be specified the fact that, they did not use NAS to perform hyper parameter search, because is often consumes more computing resources than it's actual provided benefits are worth it. PP-YOLO is a one stage anchor based detector and is made up out of a backbone, a detection neck and a detection head. The backbone is specified above and replaces the original DarkNet53 of YOLOv4, although also the ResNet architecture is a little bit modified as last stage 3x3 conv layers are replaced with deformable conv layers. As a detection neck, a FPN(Feature pyramid network) is used to build the feature pyramid with lateral connectionz between feature maps. The detection head consists of 2 conv layers, a 3x3 size followed by a 1x1 size convolution layer. This outputs the bounding box coordinates, the probability and the name of the class. The main "tricks" used in order to improve the YOLO detector are: using a larger batch size for stability of training, hence increasing it from 64 to 192; an evaluation based on the exponential moving average (EMA) during training; Drop-block instead of the classical dropout; IoU loss instead of L1 loss for the bounding box regression; making the detectors IoU aware, instead of only relying on class probability and objectness score for final confidence; Grid sensitivity; Matrix NMS, for a faster non max suppression; CoordConv, that helps with learning translation invariance or translation dependence; SPP [79] and using a better pre-train model (with higher accuracy on ImageNet). The PP(Paddle-Paddle) YOLO model yields 45.2 % mAP on the COCO test-dev dataset, at a faster speed than YOLO-v4 [63]. On a Tesla V100 GPU the frames per second for PP-YOLO are 72.9 FPS @45.2% mAP or 132 FPS @39 % mAP. Comparing it with the novel YOLO-v4 or the older YOLO-v3 [31] we can see that it outperform both in terms of accuracy and speed: the best performing YOLOv3 [31] runs at 60 FPS @38% mAP, while the best performing YOLO-v4 [63] runs at 123 FPS @38% mAP.

Model compression and acceleration

Model compression and acceleration techniques have been of great benefit for modern deep neural networks, thus we'll lay out a quick summary because of the importance in todays industrial landscape. These networks have achieved lots of breakthroughs and performed very well, but their computation and memory requirements have sky-rocketed. Thus, the result was a slowdown in their timeline of deployment in low power, low memory or strictly low latency devices such as micro-processors(in autonomous cars), micro-controllers(in smart factories) etc. So, a new problem needed a new solution: if deep learning aspires to be widely used, one needs to be able to fit deep learning algorithm in embedded and connected devices, because they are a great test for efficiency. The main idea is to perform these acceleration and compression techniques without significantly hurting the accuracy, as a main reference. In Table 4, a snapshot of these methods is outlined. There are four main categories of compression and acceleration techniques [87]:

Experiments and datasets used for benchmarking

A large emphasis needs to be put on the dataset on which the experiments are performed on. A wrongly sampled dataset, with poor quality images and poor balance between the objects labeled, can return bad results even if the model used is state-of-the-art. A quick experiment we have done(using free online tools) is to label 2 cable datasets(Fig. 6 a,b,c) composed out of 164(400 after augmentation) and 342 images(806 after augmentation) that may contain and may not contain cables, most of them do. The accuracy of the detector is impacted by the labeling technique: if you label a long cable twice or more and label multiple segments of a cable (especially is you label one part good and one part faulty), the mAP will decrease, both because of localization errors and because of classification errors. Also, if you don't label every single instance of the object in the sample images, the mAP will decrease. In order to assure oneself that every sample is labeled and correctly labeled, there are online tools that can check the health of the dataset. Moreover, the balancing of the dataset needs to be considered, as too many samples of an object or too few, may further hurt overall test time accuracy. Thus, performing a health check of the dataset is a mandatory step in training a defect detector. Finally, a small dataset performs significantly worse, than a more comprehensive dataset, which includes lots of edge cases in which we can find the suggested defect. So, in order to enable higher ceiling of performance, the dataset needs to be as large as possible (this is a general recommendation because there are cases when you have to detect defects in objects that are in the exact same place every time and for this kind of detection, there is no need for lots of data eg. 1000 samples per case should be enough). The ones used by us to experiment with the settings, are quite small, but take hours and hours of work to acquire and to label correctly. Thus, one can summarize that, for more general applications you need a large dataset, which is correctly balanced and without bias as much as possible and for more particular applications (generally found on assembly posts on the assembly line which perform repetitive action) the datasets can be smaller, but with sufficient information. The datasets we have used can be found at: github.com/tulbureandreit/Cabledata set. The largest dataset could not be uploaded.

Furthermore, a big concern shall also be the different augmentation steps used for improving the dataset diversity and trying to tackle the overfit problem. In [63] the authors specify multiple augmentation methods, the so called Bag of Freebies (because they do not impact training time and training resources): mosaic, mixup and cutmix, which are more advanced and the more basic ones like Photometric Distortions (eg: Brightness, Contrast, Hue, Saturation, Noise) and Geometric Distortions (eg: random scaling, cropping, flipping, rotation). In our little cable experiment we only used horizontal and vertical Flip, Rotations between −20 and + 20 degrees, Blur up to 6.5px and adding random Noise to the samples, up to 10% of pixels. In terms of mAP it did not had a significant effect on the results, mainly because we had small sample datasets, but

Table 4
Table for briefly summarizing the four main types of model compression and acceleration.

Summary of model compression and acceleration approaches				
Name	Short description	Type of layers	Details	Extraction method
Parameter pruning and quantization	Eliminate redundant parameters which are not important for the performance	Conv and FC layers	High performance, robust to different settings	Pre-trained or trained from scratch
Low-rank factorization	Use tensor decomposition to estimate important parameters	Conv and FC layers	easy to implement, robust pipeline	Pre-trained or trained from scratch
Transferred/compact conv filter	Specific structural conv filters to save parameters	conv layers	algos dependent on applications, higi performance	train from scratch
Knowledge distillation	train the outputs of a compact NN with the knowledge of a large NN	conv and FC layers	performance is subjective to application type and structure	training from scratch

it did improve. Thus, a mix of data augmentation techniques must be considered when training a defect detector. Moreover, the weaker the performance of the hardware setup (a small number of TFOPS or GPU RAM), the better the offset performance gain relative to the basis performance will be. The training procedure shall be straightforward and attention needs to be paid regarding the model chosen for action. Every single model has certain particularities and in our experiments, we used YOLOv4 model to detect defective cables, so the training procedure follows the steps and settings mentioned in [63]. There were two hardware setups used: one was relatively low-cost(700E) Nvidia GF750, hence our emphasis on enabling every single small business or small university to use our review as a crash course for building their own defect detection system, and the other one was via a virtual connection to a training station equipped with 4 nvidia P106-100 graphic cards, with 6 GB of RAM and Ubuntu. The remote training was performed via SSH. This second way of training is useful for large scale applications and usually involved a lot more cost than the first one, but for applications that are not specific, it's the necessary way of doing it, because cheaper computers will run out of GPU memory needed for training. All in all, a key aspect that the YOLO detection model provides is the fact that you can penalize more, either accuracy or localization, thus there is some freedom with the training process. This is important because in many cases there is a need for penalizing accuracy more than localization and in other cases there needs to be a balance. This depends on the application type that one develops. Usually, in smaller FPS applications, accuracy should be of larger concern than detection and vice versa. Our results are exemplified in Figs. 6(d,e,f).

Conclusions

Defect detection using computer vision models started to pick up popularity in the 21st century, as the object detection models became more and more popular. The exponential growth in popularity of deep learning methods for defect detection and other computer vision related applications in recent years is fueled by lots of researchers plunging into this sector, as well as by hardware and data breakthroughs. Lots of areas of defect detection solutions were reviewed in this paper and as demonstrated deep learning methods achieve state-of-the-art performance in defect detection, while also having great generalization properties. The general accepted idea is that the dataset, as well as the chosen model lead to great performances, thus both need to receive attention from the developer. The training hardware doesn't need to be expensive, if the application does not mandate it. Smaller, more specific applications can yield great results and thus, improve the workings of a small lab or business just by using general purpose laptops and generic detection models which will be tweaked for the defects we look into. While general applications that have a target as to detect lots of defects, need very large and balanced datasets, a hardware setup with lots of computational power and a specific detection model that is not just tweaked for defect detection, but built from the ground up for the specific action that we want it to perform. Thus, hardware setup and availability(that's why recommended is using virtual machines on a desktop machine learning station for general solutions) plays a role in the performance of the models. All in all, there are multiple factors that need to be taken into consideration when planning to design defect detection algorithms. There is no rule of thumb when choosing which object detection model shall generalize the best on the particular dataset of interest. First of all, the developer needs to be in constant contact with the system technician or quality inspector, in order to assure that the dataset is of great quality and is reliable. Furthermore, the constraints shall dictate which framework one chooses and if he starts developing his network from scratch or tweaks an existing one. If the temporal constraint is an important one, then models such as YOLO [7] or SqueezeDet [81] which perform at a high FPS rate are a great choice for a starting point. Then after performing a couple of proof of concept tests, one can see in which part the model lags and where to improve it (or where to improve the data). Moreover, one can apply different model compression techniques on the neural network model in order to make it run faster if the accuracy doesn't get hit too hard. If the accuracy is the leading constraint, than a more robust and larger model such as the EfficientDet [69] shall be chosen. Generally, by starting with the YOLO framework, especially the newer ones such as YOLOv4 [63] or PPYOLO [65] one cannot make a huge mistake. These are network models with great generalization properties and could be really helpful when trying to get a proof of concept as quickly as possible, hence we also tried out some experiments with the YOLOv4 framework. Our results show that, one can quickly get the proof of concept and can easily see in which direction to look in order to improve the performances. All this, without a large expense in the name of workstation equiped with lots of GPUs.

Further developments

A key trend in the sector is to obtain higher and higher box accuracy for detection. This is straightforward because in order to be able to rely on these algorithms, you would like eventually to get to 100% precision. Also, in order to be deployed in multiple applications like autonomous cars, these models need to run at a high frame rate, greater than 30 FPS, in order to be considered real time and in order to make the decisions faster than humans. A good example here would be a car approaching a roundabout: "shall the car enter or wait for a possibility to enter?". While a human can make these split-second decision with quite high accuracy (an accident happens only one out of every thousands or tens of thousands of times), a machine needs to be able to compute the scene and understand efficiently everything fast, in tens of millisecond at most. Moreover, these is a real need for model compression and acceleration developments, in order to be able to fit in the embedded devices that are found on an autonomous car or in a factory. These devices are powerful for their class of electronics, being able for a couple of TFLOPS (Tera floating point operations) at a reduced energy consumption(for the latest autonomous capable cars) or at GFLOPS(Giga floating point operations) - for the latest factory tech, but are not at the level of new GPUs and distributed systems. Factoring in that optimizing for consumption is also a mandatory requirement for vehicles, one can easily grasp the need for further compression and acceleration, while maintaining accuracy high. For factory equipment, the main constraint for the deployment of deep learning defect detection algorithms is the fact that for industrial computers the computing power is not that great and the memory available is also questionable. While the power consumption requirement is not "a hard constraint", memory is a problem, thus compression is justified also for factory applications. As a summary of the needs for defect detection: the models needs to be high accuracy, run at high speed, while needing a few MBs of memory or a few tens of MBs for operation and also consuming as little power as possible per every GFLOPS of computation. Furthermore, an upcoming trend in detection is to limit the data transfers between the processor responsible for data acquisition(field device) and the cloud system. The main intuition behind this is the fact that you lose valuable time (and also money), by sending every bit of data to the cloud for processing and then sending back the results of the processing operation to the field-device. Thus, more and more computing on the edge devices are developed such as: the Intel Neural Compute Stick or the Nvidia Jetson family of development boards. These are custom built and optimized for

deep learning applications, thus even though they lack in processing power and memory size, they are able to comfortably run these network models. Training models will be still performed on large distributed systems or via the cloud, but deployment and operation will be performed "cloud free". Also, data security and storage needs to be taken into consideration!.

Compliance with Ethics Requirements

This article does not contain any studies with human or animal subjects.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This research was funded by a grant of the Romanian National Authority for Scientific Research, CNDI-UEFISCDI, project number PN-III-P2-2.1-PED-2019–1660, contract number 387PED2020.

References

[1] Voulodimos A, Doulamis N, Doulamis A, Protopapadakis E. Deep learning for computer vision: A brief review. Comput Intell Neurosci 2018.
[2] LeCun Y. 1.1 deep learning hardware: Past, present, and future. 2019 IEEE International Solid-State Circ Conf - (ISSCC) 2019:12–9.
[3] LeCun Y, Kavukcuoglu K, Farabet C. Convolutional networks and applications in vision. In: Proceedings of 2010 IEEE international symposium on circuits and systems. p. 253–6.
[4] Murthy CB, Hashmi MF, Bokde N, Geem ZW. Investigations of object detection in images/videos using various deep learning techniques and embedded platforms–a comprehensive review. Appl Sci 2020;10:3280.
[5] Dalal N, Triggs B. Histograms of oriented gradients for human detection. In: 2005 IEEE Computer society conference on computer vision and pattern recognition (CVPR'05) 2005; vol. 1: 886–93.
[6] Girshick RB, Donahue J, Darrell T, Malik J. Rich feature hierarchies for accurate object detection and semantic segmentation. 2014 IEEE conference on computer vision and pattern recognition 2014:580–7.
[7] Redmon J, Divvala S, Girshick RB, Farhadi A. You only look once: Unified, real-time object detection. In: 2016 IEEE conference on computer vision and pattern recognition (CVPR). p. 779–88.
[8] Viola P, Jones MJ. Rapid object detection using a boosted cascade of simple features. In: Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition, CVPR 2001 1. p. I-I..
[9] Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. Commun ACM 2012;60:84–90.
[10] Kass M, Witkin A, Terzopoulos D. Snakes: Active contour models. Int J Comput Vision 2004;1:321–31.
[11] Conners R, Harlow C. A theoretical comparison of texture algorithms. IEEE Trans Pattern Anal Mach Intell PAMI-2 1980:204–22.
[12] Sassi OB, Slima M, Chtourou K, Hamida A. Improved spatial gray level dependence matrices for texture analysis. Int J Comput Sci Inform Technol 2012;4:209–19.
[13] Wang X, Albregtsen F, Foyn B. Texture features from gray level gap length matrix. MVA 1994.
[14] Yang J, Guo J. Image texture feature extraction method based on regional average binary gray level difference co-occurrence matrix. In: 2011 International conference on virtual reality and visualization. p. 239–42.
[15] Shabat AM, Tapamo J. A comparative study of local directional pattern for texture classification. In: 2014 World symposium on computer applications & research (WSCAR). p. 1–7.
[16] Rioul O, Duhamel P. Fast algorithms for discrete and continuous wavelet transforms. IEEE Trans Inf Theory 1992;38:569–86.
[17] Oberst U. The fast fourier transform. SIAM J Control Optim 2007;46 (2):496–540. doi: https://doi.org/10.1137/060658242.
[18] Ghazvini M, Monadjemi S, Movahhedinia N, Jamshidi K. Defect detection of tiles using 2d-wavelet transform and statistical features. Int J Electr Comput Eng 2009;3:89–92.
[19] Karayiannis Y, Stojanovic R, Mitropoulos P, Koulamas C, Stouraitis T, Koubias S, Papadopoulos G. Defect detection and classification on web textile fabric using multiresolution decomposition and neural networks, ICECS'99. In: Proceedings of ICECS '99. 6th IEEE international conference on electronics, circuits and systems (Cat. No.99EX357) vol. 2;1999: 765–8.
[20] Vallerand S, Maldague X. Defect characterization in pulsed thermography: a statistical method compared with kohonen and perceptron neural networks. Ndt & E Int 2000;33:307–15.
[21] Pham DT, Soroka A, Ghanbarzadeh A, Koç E, Otri S, Packianather M. Optimising neural networks for identification of wood defects using the bees algorithm. In: 2006 4th IEEE international conference on industrial informatics; 2006. p. 1346–51.
[22] Gao W, Chen X, Chen D. Genetic programming approach for predicting service life of tunnel structures subject to chloride-induced corrosion. J Adv Res 2019;20:141–52.
[23] Kasban H, Zahran O, Arafa H, El-Kordy M, El-Araby S, El-Samie FE. Welding defect detection from radiography images with a cepstral approach. Ndt & E Int 2011;44:226–31.
[24] Tonazzini A, Salerno E, Abdel-Salam Z, Harith MA, Marras L, Botto A, Campanella B, Legnaioli S, Pagnotta S, Poggialini F, Palleschi V. Analytical and mathematical methods for revealing hidden details in ancient manuscripts and paintings: A review. J Adv Res 2019;17:31–42.
[25] Unay D, TCTS BG. Apple defect detection and quality classification with mlp-neural networks, 2002.
[26] Mishra S, Sarkar U, Taraphder S, Datta S, Swain D, Saikhom R, Panda S, Laishram M. Principal component analysis. Int J Livestock Res 2017:1. doi: https://doi.org/10.5455/ijlr.20170415115235.
[27] Dudzik S. Characterization of material defects using active thermography and an artificial neural network. Metrol Meas Syst 2013;20:491–500.
[28] Brzakovic D, Vujovic N. Designing a defect classification system: A case study. Pattern Recognit 1996;29:1401–19.
[29] Newman T, Jain AK. A survey of automated visual inspection. Comput Vis Image Underst 1995;61:231–62.
[30] Rui H, Ji-nan G, Xiao-hong S, Yong-tao H, Uddin S. A rapid recognition method for electronic components based on the improved yolo-v3 network. Electronics 2019;8:825.
[31] Redmon J, Farhadi A. Yolov3: An incremental improvement, ArXiv abs/1804.02767.
[32] Howard A, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications, ArXiv abs/1704.04861.
[33] Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu CY, et al. Ssd: Single shot multibox detector. In: ECCV; 2016.
[34] Ren S, He K, Girshick RB, Sun J. Faster r-cnn: Towards real-time object detection with region proposal networks. IEEE Trans Pattern Anal Mach Intell 2015;39:1137–49.
[35] Andrei-Alexandru T, Henrietta DE. Low cost defect detection using a deep convolutional neural network. 2020 IEEE International conference on automation, quality and testing, robotics (AQTR) 2020:1–5.
[36] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition, CoRR abs/1409.1556.
[37] Badmos O, Kopp A, Bernthaler T, Schneider G. Image-based defect detection in lithium-ion battery electrode using convolutional neural networks. J Intell Manuf 2020;31:885–97.
[38] Li Y, Han Z, Xu H, Liu L, Li X, Zhang K. Yolov3-lite: A lightweight crack detection network for aircraft structure based on depthwise separable convolutions. Appl Sci 2019;9:3781.
[39] Adibhatla VA, Chih H-C, Hsu C-C, Cheng J, Abbod M, Shieh J. Defect detection in printed circuit boards using you-only-look-once convolutional neural networks. Electronics 2020;9:1547.
[40] Redmon J, Farhadi A. Yolo9000: Better, faster, stronger. In: 2017 IEEE conference on computer vision and pattern recognition (CVPR); 2017. p. 6517–25.
[41] Li J, Su Z, Geng J, Yin Y. Real-time detection of steel strip surface defects based on improved yolo detection network. IFAC-PapersOnLine 2018;51:76–81.
[42] Ding F, Zhuang Z, Liu Y, Jiang D, Yan X, Wang Z. Detecting defects on solid wood panels based on an improved ssd algorithm, Sensors (Basel, Switzerland) 20.
[43] Huang G, Liu Z, Weinberger KQ. Densely connected convolutional networks. 2017 IEEE conference on computer vision and pattern recognition (CVPR) 2017:2261–9.
[44] Tao X, Zhang D, Wang Z, Liu X, Zhang H, Xu D. Detection of power line insulator defects using aerial images analyzed with convolutional neural networks. IEEE Trans Syst, Man, Cybernet: Syst 2020;50:1486–98.
[45] Chen J, Liu Z, Wang H, Núñez A, Han Z. Automatic defect detection of fasteners on the catenary support device using deep convolutional neural network. IEEE Trans Instrum Meas 2018;67:257–69.
[46] Feng C, Liu M-Y, Kao C-C, Lee T-Y. Deep active learning for civil infrastructure defect detection and classification, 2017.
[47] Huang Y, Qiu C, Wang X, Wang S, Yuan K. A compact convolutional neural network for surface defect inspection. Sensors (Basel, Switzerland) 20.
[48] Sandler M, Howard A, Zhu M, Zhmoginov A, Chen L-C. Mobilenetv 2: Inverted residuals and linear bottlenecks. 2018 IEEE/CVF conference on computer vision and pattern recognition 2018:4510–20.
[49] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. 2016 IEEE conference on computer vision and pattern recognition (CVPR) 2016:770–8.
[50] Duy-Dinh Le SS. ilsvrc2011.nsc.bow.dense4mul.sift.soft-500-vl2.ils.
[51] Iandola FN, Moskewicz MW, Ashraf K, Han S, Dally W, Keutzer K. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size, ArXiv abs/1602.07360.

[52] Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift, ArXiv abs/1502.03167.
[53] Huang G, Liu Z, Weinberger KQ. Densely connected convolutional networks. In: 2017 IEEE conference on computer vision and pattern recognition (CVPR). p. 2261–9.
[54] Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z. Rethinking the inception architecture for computer vision. In: 2016 IEEE conference on computer vision and pattern recognition (CVPR). p. 2818–26.
[55] Cui C, Ye Z, Li Y, Li X, Yang M, Wei K, et al. Semi-supervised recognition under a noisy and fine-grained dataset, ArXiv abs/2006.10702.
[56] Yalniz IZ, Jégou H, Chen K, Paluri M, Mahajan D. Billion-scale semi-supervised learning for image classification, ArXiv abs/1905.00546.
[57] Chen Y, Li J, Xiao H, Jin X, Yan S, Feng J. Dual path networks. In: NIPS; 2017.
[58] Real E, Aggarwal A, Huang Y, Le QV. Regularized evolution for image classifier architecture search. In: AAAI.
[59] Touvron H, Vedaldi A, Douze M, Jégou H. Fixing the train-test resolution discrepancy: Fixefficientnet, ArXiv abs/2003.08237.
[60] Xie Q, Hovy E, Luong M-T, Le QV. Self-training with noisy student improves imagenet classification. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). p. 10684–95.
[61] Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, et al. An image is worth 16x16 words: Transformers for image recognition at scale, ArXiv abs/2010.11929.
[62] Zhang S, Wen L, Bian X, Lei Z, Li S. Single-shot refinement neural network for object detection. In: 2018 IEEE/CVF conference on computer vision and pattern recognition. p. 4203–12.
[63] Bochkovskiy A, Wang CY, Liao H. Yolov4: Optimal speed and accuracy of object detection, ArXiv abs/2004.10934.
[64] Gao Z, Wang L, Wu G. Lip: Local importance-based pooling. In: 2019 IEEE/CVF international conference on computer vision (ICCV). p. 3354–63.
[65] Long X, Deng K, Wang G, Zhang Y, Dang Q, Gao Y, et al. Pp-yolo: An effective and efficient implementation of object detector, ArXiv abs/2007.12099.
[66] dong Liu Y, Wang Y, Wang S, Liang T, Zhao Q, Tang Z, et al. Cbnet: A novel composite backbone network architecture for object detection. In: AAAI; 2020.
[67] Zoph B, Ghiasi G, Lin TY, Cui Y, Liu H, Cubuk ED, et al. Rethinking pre-training and self-training, ArXiv abs/2006.06882.
[68] Qiao S, Chen LC, Yuille A. Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution, ArXiv abs/2006.02334.
[69] Tan M, Pang R, Le QV. Efficientdet: Scalable and efficient object detection. In: 2020 IEEE/CVF conference on computer vision and pattern recognition (CVPR); 2020. p. 10778–87.
[70] Misra D. Mish: A self regularized non-monotonic activation function. In: BMVC; 2020.
[71] Ghiasi G, Lin T-Y, Pang R, Le QV. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In: 2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR). p. 7029–38.
[72] Chao P, Kao CY, shan Ruan Y, Huang CH, Lin Y. Hardnet: A low memory traffic network. In: 2019 IEEE/CVF international conference on computer vision (ICCV); 2019. p. 3551–60.
[73] He K, Gkioxari G, Dollár P, Girshick RB. Mask r-cnn. IEEE Trans Pattern Anal Mach Intell 2020;42:386–97.
[74] Du X, Lin T-Y, Jin P, Ghiasi G, Tan M, Cui Y, Le QV, Song X. Spinenet: Learning scale-permuted backbone for recognition and localization. In: 2020 IEEE/CVF conference on computer vision and pattern recognition (CVPR). p. 11589–98.
[75] LoweDavid G. Distinctive image features from scale-invariant keypoints. Int J Comput Vision.
[76] Touvron H, Vedaldi A, Douze M, Jégou H. Fixing the train-test resolution discrepancy: Fixefficientnet, ArXiv abs/2003.08237.
[77] Howard A, Sandler M, Chu G, Chen LC, Chen B, Tan M, et al. Searching for mobilenetv3. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV); 2019. p. 1314–24.
[78] Evgeniou T, Pontil M. Support vector machines: Theory and applications. In: Machine Learning and Its Applications.
[79] He K, Zhang X, Ren S, Sun J. Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE Trans Pattern Anal Mach Intell 2015;37:1904–16.
[80] Girshick RB. Fast r-cnn. In: 2015 IEEE International conference on computer vision (ICCV); 2015. p. 1440–8.
[81] Wu B, Iandola FN, Jin P, Keutzer K. Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving. In: 2017 IEEE conference on computer vision and pattern recognition workshops (CVPRW). p. 446–54.
[82] Iandola FN, Moskewicz MW, Ashraf K, Han S, Dally W, Keutzer K. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size, ArXiv abs/1602.07360.
[83] Tan M, Le QV. Efficientnet: Rethinking model scaling for convolutional neural networks, ArXiv abs/1905.11946.
[84] Wang CY, Liao H, Yeh IH, Wu YH, Chen PY, Hsieh JW. Cspnet: A new backbone that can enhance learning capability of cnn. In: 2020 IEEE/CVF conference on computer vision and pattern recognition workshops (CVPRW); 2020. p. 1571–80.
[85] Xie S, Girshick RB, Dollár P, Tu Z, He K. Aggregated residual transformations for deep neural networks. In: 2017 IEEE conference on computer vision and pattern recognition (CVPR). p. 5987–95.
[86] Chen B, Ghiasi G, Liu H, Lin T-Y, Kalenichenko D, Adam H, Le QV. Mnasfpn: Learning latency-aware pyramid architecture for object detection on mobile devices. In: 2020 IEEE/CVF conference on computer vision and pattern Recognition (CVPR). p. 13604–13.
[87] Cheng Y, Wang D, Zhou P, Zhang T. A survey of model compression and acceleration for deep neural networks, ArXiv abs/1710.09282.