Journal of Real-Time Image Processing (2022) 19:1199–1210
https://doi.org/10.1007/s11554-022-01255-7

ORIGINAL RESEARCH PAPER

Towards SSD accelerating for embedded environments: a compressive sensing based approach

Imene Bouderbal[1] · Abdenour Amamra[1] · M. El-Arbi Djebbar[1] · M. Akrem Benatia[1]

Received: 15 May 2022 / Accepted: 13 September 2022 / Published online: 28 September 2022

Abstract
Since the rise of convolutional neural networks (CNN), deep learning-based computer vision has been a dynamic field of research. Nevertheless, modern CNN architectures have not given sufficient consideration to real−time applications within limited computation settings and always compromise speed and accuracy. To this end, a novel approach to CNN design, based on the emerging technology of compressive sensing (CS), is proposed. For instance, CS networks function in a compression−reconstruction approach as an encoder−decoder neural network. This approach transforms the computer vision problem into a multioutput learning problem by incorporating the CS network into a recognition network for joint training. As to the deployment phase, images are obtained from a CS−acquisition device and fed directly, without reconstruction, to the new recognition network. Following such an approach considerably improves transmission bandwidth and reduces the computational burden. Furthermore, the redesigned CNN holds fewer parameters than its original counterpart, thus reducing model complexity. To validate our findings, object detection using the Single−Shot Detector (SSD) network was redesigned to operate in our CS−based ecosystem using different datasets. The results show that the lightweight CS network offers good performance at a faster running speed. For instance, the number of FLOPS was reduced by 57% compared to the SSD baseline. Furthermore, the proposed CS_SSD achieves a compelling accuracy while being 30% faster than its original counterpart on small GPUs. Code is available at: https://github.com/Bouderbal-Imene/CS-SSD.

Keywords Computer vision · Real-time · Convolutional neural network · Compressive sensing · SSD · Embedded systems · Robots

1 Introduction

Throughout the past decade, convolutional neural networks (CNN) have acquired great interest in both the industrial and academic fields and have emerged as the leading technique for visual recognition tasks. Despite the vast traction that CNN gained, a gap between expectations and reality persisted. Therefore, practical use of such networks in real-world applications requires an optimal level of performance next to which higher response time is sought. However, to achieve much higher performance, modern neural networks have become very deep and difficult to tune on large datasets. On the other hand, the design of efficient (small and performant) neural networks has become a new research focus that has not matured yet [6]. Even more, the development of vision-based perception capabilities for embedded systems, such as robots or autonomous vehicles, with rapid response time and high accuracy is not easy to achieve. With limited computational power and weight capabilities, the choice of sensors and algorithms able to run onboard in real time is the real challenge. Common computer vision approaches usually employ multiple sensors, including cameras, LASERs, LIDARs, and RADARs. Nevertheless, these sensors have their drawbacks, including the high cost and low spatial resolution, they are havier end consume more, interference between the different sensing modalities, adding to that the preprocessing latency [40].

✉ Imene Bouderbal
imene.bouderbal@yahoo.com

Abdenour Amamra
abdenour.amamra@gmail.com

M. El-Arbi Djebbar
medjebbar@gmail.com

M. Akrem Benatia
akrem.benatia@yahoo.com

[1] Ecole Militaire Polytechnique, Bordj El-Bahri BP 17, Algiers, Algeria

The best compromise of cost and quality at a reasonable preprocessing load is offered by cameras [5]. Therefore, in the present work, we focus on accelerating computer vision models using cameras to achieve image understanding. Most of the proposed architectures in the literature are designed to process *RGB* images even though the latter are compressed before or after acquisition for storage and transmission purposes using either engineered compression algorithms or machine-learning ones (Fig. 1a). Hence, adding an additional decoding step, which is computationally costly and memory demanding, complicates the real-time and embedded deployment. A possible alternative is to design architectures capable of learning with compressed representations (Figure 1b), rather than traditional *RGB* images in order to speed up training and inference of deep networks [14]. Likewise, image compression algorithms have also benefited from the rise of CNN. In addition to their outstanding compression performances, learned compression algorithms can be adapted to specific target domains resulting in better compression rates than engineered ones. One famous learned compression algorithm that has gained significant interest is compressive sensing (CS) tailored to image acquisition applications. A plausible idea would be to develop recognition architectures that work directly on CS compressed representations and bypass the reconstruction phase.

In this paper, we explore CS on adapted CNN architectures to perform computer vision tasks. The chosen computer vision task to validate the proposed approach is object detection based on the Single-Shot Detector (SSD) network since they encompass classification and regression tasks in a single network [25]. The proposed lightweight compressed-sensing architecture is inspired by [33] and is composed of two networks for sampling and reconstruction. Our approach aims at taking the latent space representations resulting from the sampling network and using them for training and inference in the adapted architectures. The major contributions of the paper are the following:

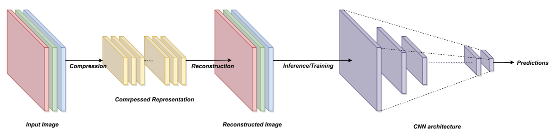
- A lightweight CS network is proposed based on the one contributed in [33] which provides minimal latency for both sampling and reconstruction.
- We propose a new approach that enables recognition networks to operate in CS-based ecosystems. By redesigning the early layers of the recognition network (backbone), this latter will exploit sampled data directly without further reconstruction.
- We validate the proposed approach for the detection task using the SSD object detector upon heavy and light backbones, namely VGG and Mobilenet, respectively.
- Both the redesigned backbones and the SSD head are enhanced by adding Batch Normalization layers to permit training from scratch.
- Our approach is validated on three different datasets on high-performing and small GPUs. The obtained results are promising, achieving interesting accuracies while being up to 30% faster on small GPUs, which is suitable for resource-constrained devices and embedded environments.

We focus on redesigning computer vision architectures for applications in embedded settings and we validate our findings on object detection tasks. The remainder of this paper is organized as follows. The background on object detection, CS paradigm and compressed learning related works are introduced in Sect. 2. In Sect. 3, the details of the proposed approach are presented. Section 4 provides the experimental setup followed by the experimental results in Sect. 5. In Sect. 6, we conclude the paper.

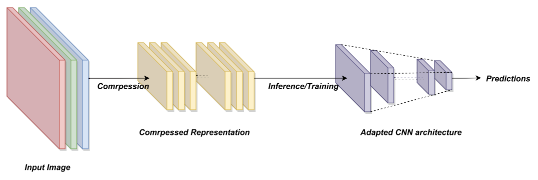
2 Related works

2.1 Object detection

Since the rise of CNN, deep learning-based object detection has been outperforming traditional methods by a significant margin. Existing models are mainly based on RGB images and can be either two-stage or one-stage detectors. Two-stage models rely on region proposal followed by box classification, such as R-CNN [13] and its subsequent improved versions (Fast R-CNN [12], Faster R-CNN [32], Mask RCNN [17] and Cascade RCNN [7]). While models belonging to this category have been proven successful, they remain highly computationally expensive. On the other hand, the current state-of-the-art of fast object detection is mainly driven by one-stage detectors such as YOLO [28]



(a)



(b)

Fig. 1 **a** RGB-based training/inference pipeline; **b** compressed images training/inference pipeline and SSD [25]. By re-framing object detection as a single regression problem, a single network that simultaneously outputs bounding box coordinates and class scores could be used [26]. For YOLO models, the prediction relies on global image features extracted from convolutional layers, which significantly improves the detection speed at the cost of detection precision. Nonetheless, YOLO improvements [4, 29, 30] are one of the fastest and most accurate object detectors by integrating several improvements such as multi-label object class prediction, prediction across scale, the use of K-means clustering to determine box priors, etc. Another popular one-shot detector is SSD. This latter considers a fixed set of default bounding boxes with an associated feature map at different scales and aspect ratios. By coupling the box matching strategy with the multi-scale features, SSD is significantly more accurate than the original YOLO network with a higher detection speed. Also, SSD-based models such as [15] are one of the most accurate and lightweight object detectors. Moreover, many attempts to accelerate existing state-of-the-art models have been carried out. The authors of [23] proposed a general compression pipeline (through pruning, knowledge distillation, and quantization) for one-stage object detection networks to meet the real-time requirements. In [16], a lightweight and fast object detector based on ShuffleNetV2 and YOLO head is proposed. The model has achieved competitive results in accuracy and speed while being lightweight. Likewise, many efforts were made to accelerate object detection tasks for specific applications. The work in [39] proposed a faster version SSD model based on parameter reduction and dilated convolution. The obtained results showed that the proposed model achieves higher speed compared to the original one for specific applications (apple detection, bicycle detection, and vehicle detection). In [31], a real-time traffic sign detection network using DS-DetNet and lite fusion FPN is proposed. The model achieves compelling accuracy with high speed. It should be noted that the methods mentioned above were proposed to speed up the inference stage using RGB-based architectures. The proposed approach herein aims at improving the network's inference speed by considering lightweight compressed inputs. Since this work focuses on embedded settings with limited resources, such as transportation systems and robots, the SSD one-stage object detector was chosen as the baseline architecture for the proposed solution. For instance, the combination of our approach with other RGB-based classifiers, detectors, or segmentation networks could be of interest. Also, the combination of our approach with the above-mentioned ones could be relevant.

2.2 Compressive sensing

Compressive sensing is a powerful sensing paradigm to sample sparse signals with much fewer samples than the Shannon-Nyquist sampling limit [3, 21]. Inherent redundancy present in real signals like images and videos allows significant data compression. CS exploits this redundancy and enables sampling at Sub-Nyquist rates. This makes CS extremely useful for capturing images and videos in systems that cannot afford high data bandwidth. The ability to sample very few data points and still be able to reconstruct the original signal helps to create lower power consumption imaging systems. The existing CS methods are mainly classified into two categories: iterative optimization-based CS methods and neural network-based CS [24]. Refer to [24, 33] for a review on existing solutions. For methods belonging to both categories, the theory behind CS guarantees that a sparse signal (in some domains) can be exactly recovered from many fewer measurements. Concretely, suppose that $x \in R^{N\times 1}$ is a real-valued signal and $\Phi \in R^{M\times N}$ is a sampling matrix, with $M << N$, the CS measurements acquisition is expressed as:

$$y = \Phi x \tag{1}$$

where $y \in R^{M\times 1}$ is the CS measurement. In general, because the number of unknowns is much larger than the number of observations [11], recovering the original signal x from its corresponding measurements, y is not feasible. However, if the signal x is sparse in some domain Ψ, the CS theory shows that an exact recovery of x is possible. The CS reconstruction can be formulated as:

$$\min_x ||\Psi x||_p, \quad \text{subject to :} \quad y = \Phi x \tag{2}$$

where Ψx is the spare coefficients for domain Ψ, and the subscript p is usually set to 1 or 0, characterizing the sparsity of the vector Ψx.

2.3 Neural networks and compression: compressed learning

Compressed learning concepts were first introduced in [1], which showed that direct inference from compressed representations and measurements is feasible with high performances. In the light of this approach and given that training and particularly inference speed is critical, many works have focused on accelerating networks' computations by employing spatial frequency decomposition, and other image compressed representations obtained using engineered codecs [8, 9, 14]. Other works have explored deep learned compressed data to promising effect [1, 34]. In [14], the authors demonstrate the use of DCT coefficients available in the JPEG image format as an effective input representation to CNN. Instead of inputting RGB pixels, a JPEG-compressed image half-decompressed to 8×8 block DCT coefficients is used to input the network. The method can achieve good performance while offering a significant speedup, primarily by replacing the first two blocks with the already-computed JPEG DCT coefficients. Several approaches are evaluated to sample and place the DCT input into the network. However, few examples of architectures exploring features obtained from learned image compression algorithms exist in the literature. A relevant work is [34], where the authors explore the use of learned compressed image representation for solving two computer vision tasks (classification and semantic segmentation) without employing a decoding step. The compressed inputs are acquired using a heavy auto-encoder architecture. The results are compared to those obtained using RGB images and are similar for classification and slightly improved segmentation (especially for aggressive compression rates). The authors also jointly train the compression and classification and show that it can enhance the results. However, the deep encoder constitutes a memory and computation burden and is inefficient. Another work is the one proposed in [41]. The author proposed a compressive convolutional network (CCN), which is a compressive-sensing-enabled CNN. The proposed CCN optimizes and reuses the convolution operations of the first layers of the detector for recoverable data embedding and image compression. Therefore, no extra computational overhead is required for image compression. However, the detection task is performed, as usual, i.e. on RGB images. The approach we propose is most related to [34] since we aim to use deep learned image compressed representations directly to solve vision tasks.

3 Methodology

Traditionally, CNNs deployed in embedded systems such as robots or autonomous vehicles are fed with a flow of RGB images from a high-resolution acquisition device to perceive the surrounding environment better. Frame-by-frame processing of such an enormous amount of data requires hefty computational resources.

Additionally, and motivated by the emergence of deep learning and compressive sensing, different learned sampling mechanisms were developed to incorporate CS in image and video acquisition. In fact, and in contrast to traditional compression algorithms such as JPEG, learned CS does not override the spatial structure of images, as it generates feature maps ready to be explored with adapted architectures. Consequently, using such sampling methods for data acquisition than exploring the acquired data would be more efficient than traditional pipelines. In this scope, the present work proposes and explores an efficient learned compression method, namely learned CS for both real-time sampling and reconstruction, for object detection task and adapts state-of-the-art architectures accordingly. Figure 2 illustrates the proposed pipeline.

The proposed solution transforms the recognition problem into a multiple output learning problem [37]. The proposed scheme learns simultaneously to predict two outputs given an input image. The first is the reconstructed image following the CS branch, and the second is the bounding boxes and scores following the detection branch. It offers in one hand two training procedures:

- The entire network can be trained jointly from scratch or from a previous trained state of the CS network as an initialization;
- The detection network can be trained using a fixed previously trained sampling network.

However, in this work, only the first training procedure is explored. Furthermore, during deployment and inference, the reconstruction part can be omitted since the compressive

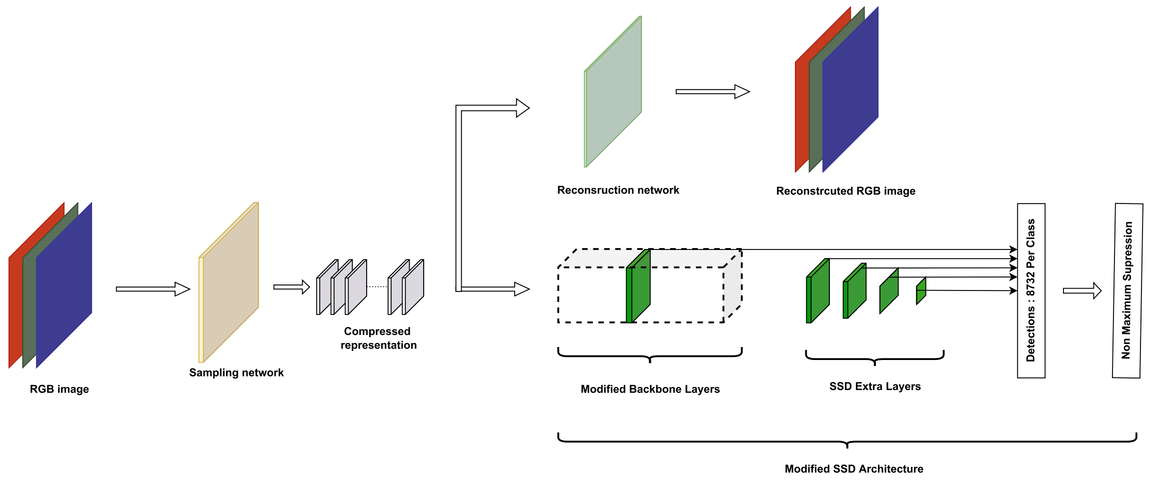


Fig. 2 The proposed architecture for joint object detection and image sampling

camera using the sampling learned weights deliverers representations ready to be explored by the modified detection network. Hence, the gain for such architectures mainly stems from the reduced data transfer between CPU and GPU due to image compression and decompression. As it can be seen in Fig. 2, after the generation of the compressed representation by the sampling network, this latter is fed to the reconstruction network to optimize for mean-squared reconstruction error and to the new detection network (named CS_D) to optimize for both classification and bounding box regression losses. The total loss function is:

$$L_c = \beta L_{CS} + L_{OD} \tag{3}$$

where L_{CS} is the loss term for the compressive sensing network (see Eq. 4), and L_{OD} is the CS_D loss (see Eq. 5). β controls the trade-off between compression loss and detection loss.

$$L_{CS} = \mathrm{MSE}(x, \hat{x}) \tag{4}$$

where x is the input image and $\hat{x}$ is the reconstructed image such as: $\hat{x} = R(S(x))$.

$$L_{OD} = L(x, c, l, g) = \frac{1}{N}\left(L_{\mathrm{conf}}(x, c) + \alpha L_{\mathrm{loc}}(x, l, g)\right) \tag{5}$$

Similarly to [25], the CS_D loss function consists of two terms: L_{conf} and L_{loc} where N is the matched default boxes. L_{loc} is the localization loss which is the Smooth $L1$ loss between the predicted box l and the ground-truth box g. L_{conf} is the confidence loss which is the Softmax loss over multiple classes confidences c (α is set to 1 by cross validation).

3.1 Lightweight CNN for image compressive sensing: L_CSnet

The proposed lightweight CNN for image CS is inspired by [33]. It has a sampling network and a reconstruction network. The sampling network is used to learn the sampling matrix and acquire CS measurements. The reconstruction network, which is linear, learns an end-to-end mapping from the CS measurements to the reconstructed images (Fig. 3). In contrast to [33], the deep nonlinear reconstruction network used for quality refinement is removed.

3.1.1 Sampling network

First, the image is divided into non-overlapping blocks of size $B \times B \times l$ (l is the number of channels). Then, using a sampling matrix Φ_B of size $n_B \times lB^2$, the CS measurements are acquired. A convolution layer is used to imitate the compressed sampling process (while considering each row of the sampling matrix Φ_B as a filter). The size of each filter in the sampling layer is $B \times B \times l$ (according to the size of the image blocks) so that each filter outputs one measurement. For the sampling ratio $\frac{M}{N}$, there are $n_B = \frac{M}{N} lB^2$ rows in the sampling matrix Φ_B to obtain n_B CS measurements. Therefore, there are n_B filters of size $B \times B \times l$ in this network. Formally, the sampling process S can be expressed as :

$$y = S(x) = W_s * x \tag{6}$$

where $*$ represents convolution operation, x is the input image, y is the CS measurement, W_s corresponds to n_B filters of size $B \times B \times l$. The output of the sampling network to an image block is a $1 \times 1 \times n_B$ vector. The learned sampling matrix can efficiently utilize the characteristic of images and make the CS measurements retain more structural information for better reconstruction. In the application phase, the learned sampling matrix is used as an encoder to generate CS measurements.

3.1.2 Reconstruction network

To reconstruct the image, the pseudo-inverse matrix of the sampling network Φ_B is used, following the works in [33, 38]. Given CS measurements y_i of the jth block, its reconstruction result is $\underset{\sim}{x_j} = \Phi y_j$. Φ_B is a matrix of size $lB^2 \times n_B$. Similar to the sampling process, a convolution layer with special kernel size and stride could be used to implement the reconstruction process. Similar to [33], the matrix Φ_B is optimized instead of the pseudo-inverse matrix of Φ_B. The reconstruction process R can be expressed as:

$$\tilde{R}(y) = W_{\mathrm{int}} * y \tag{7}$$

where y is the CS measurement, and W_{int} is the filters. The size of each one of the lB^2 convolution filter in the reconstruction layer is $1 \times 1 \times n_B$. The stride of this convolution layer is set to 1×1 to reconstruct each block and the bias is ignored. Each column of $R(y)$ is a $1 \times 1 \times lB^2$ vector corresponding to an image block of size $B \times B \times l$. A combination layer, which contains a reshape and a concatenation function, is used to obtain the final reconstructed image.

3.2 Recognition network

Our goal is to design a CNN for object detection starting from compressed representations of images obtained via the sampling network of the deep CS model described in the previous subsection. As mentioned earlier, the chosen baseline to design such a network is the state-of-the-art object detector SSD.

3.2.1 RGB baselines

Like most CNN object detectors (except YOLO and its variants), SSD relies on existing classification modules.

Originally, SSD was designed to use the VGG network as a backbone. However, object detection networks are not restricted to a specific classification backbone and can easily use others, provided that the dimensions of input images and network outputs are compatible. The proposed approach was implemented using two backbones, namely VGG and MobileNet [19], to evaluate its impact on both heavy and lightweight architectures. Other computer vision tasks, such as classification and segmentation, can also be used alongside the proposed approach. We first use the original SSD framework (based on VGG), then we study the effectiveness of BatchNorm for training SSD from scratch following the approach proposed in [42] (denoted SSD_BN). We incorporate BatchNorm layers after each convolutional layer in the VGG backbone and head (MobileNet already has BatchNorm layers). According to the authors [42], BatchNorm renders the optimization landscape remarkably smoother, inducing a more predictable and steady behavior of the gradients to permit bigger searching space and prompter convergence.

3.2.2 CS based-architectures

The use of compressed representation requires the design of adequate CNN. Figure 2 illustrates the proposed architecture where the global model is composed of two subnetworks, the CS one and the SSD one. The SSD subnetwork (i.e. backbone and head) used in the CS-based models is the SSD_BN. This latter is further adapted to allow compressed data processing and is denoted: CS_D_i_j where i stands for the block size B and j for the sampling rate $\frac{M}{N}$ used in the CS sub-network.

It is worth mentioning that the block sizes were chosen to produce compressed representations that could be fed to CS_D models. In the literature [11, 27, 33], a block size of 32 was used, but in this work, other values were explored. For input images with spatial dimension $h \times w \times l$, the sampling network of the CS model outputs a compressed representation with dimensions $B \times B \times n_B$, where n_B is the number of produced feature maps (corresponding to n_B filters, as described in Sect. 3.1.1). Therefore, variants of the backbones architecture are proposed to use these latent space representations as input. Similar to [34], these networks are designed by cutting off the front of the regular RGB models that have a larger spatial dimension than $B \times B$.

The spatial data inputs that will be investigated are 75×75 and 38×38. The first one is obtained using a block size of $B = 4$ and input images of 300×300, while for the second one, a block size of $B = 8$ and input images of 304×304. For the VGG backbone, the first and second convolutional blocks are removed for the former inputs, and the compressed representations are fed to the third convolutional block. Similarly, the fourth convolutional block is used directly after removing previous blocks for the later inputs. As for the Mobilenet network, only inputs of 75×75 are used while removing the early three convolutional blocks.

4 Experimental setup

4.1 Evaluation metrics

The image quality assessment we use herein to evaluate the proposed CS network is a full reference one based on Peak Signal-to-Noise Ratio [18] (PSNR) and Structural Similarity Index [36] (SSIM) metrics. The former is used as it correlates with the pixel-based loss used in the optimization of the CS branch (see Eq. 4). The second is used to measure better the similarity of images as perceived by humans. Also, sampling and reconstruction latency are analyzed. For the object detection task, the mean average precision (mAP) at different IOU thresholds along with the number of Frames Per Seconds (FPS) that can be processed by the network are used [27]. Besides, the number of floating-point operations [20] (FLOPs) that represents the amount of calculation of a model is used to measure models complexity.

4.2 Datasets

The experimentations were carried out on the YYmnist, the Pascal VOC datasets and the Mask dataset (Table 1). It should be mentioned that the first one has similar classes, as all of them are black sharing a white background. As for the pascal VOC, the classes belong to sparse and heterogenous categories (cow vs. tv monitor, for example), and the last used dataset has only one class (with mask).

4.3 Training procedure

On the Pascal VOC dataset, we used the same training settings as the original SSD (the baseline for this study), including data augmentation and anchor settings for all models (RGB-based SSD and CS-based SSD). When training SSD_BN variants, obtained by adopting the approach proposed in [42], we use their proposed configuration (learning rate, batch size, etc.), to ascertain the effectiveness of Batchnorm layers in training from scratch. For the YYmnist dataset and the Mask dataset, a simple data augmentation pipeline is used to accelerate the training process.

After hyperparameter tuning on NVIDIA Tesla V100 GPUs, all generated models are trained from scratch. A large batch size (of 128 images) is used for training to ensure the stable statistical results of BatchNorm in the training phase. All models are trained for a fixed number of epochs with no early stopping to ensure a fair test comparison between their results. The loss function to minimize is described in Eq. (3).

5 Experimental results

5.1 Models complexity

Execution time required for a forward pass through a neural network depends on the number of floating-point operations (FLOPs). From Fig. 4, we can see that by applying our approach:

- VGG backbone: for the set of models where a block size of 4 is used, the gain in terms of FLOPS is 31%, and for those where a block size of 8 is up to 57%;
- Mobilenet backbone: the number of FLOPs is reduced by 23% using a block size of 4.

5.2 Evaluation of the proposed lightweight CS network

Herein, we investigate the performance of L_CSnet network in terms of both image reconstruction quality and running speed (Table 2). We compare it with the JPEG standard [35], as it is one of the most popular and effective compression algorithms, and two state-of-the-art deep learning based CS method, namely ReconNet [22] and CSnet [33]. For this part of the evaluation, our model was trained using the DIV2k dataset [2] using 256×256 grayscale images. For a fair comparison, we follow [22, 33] to use a block size of 32×32 and Set11 [2] as the evaluation dataset. Refer to section S1 (Supplementary Information) for a deeper full reference evaluation of the proposed CS network based on [10].

In contrast to the JPEG standard, the deep learning-based CS methods are much faster (95.24%, 97.34% and 99.76% for Reconet, CSnet and L_CSNet, respectively). However, the JPEG standard performs better in image quality metrics (PSNR and SSIM). Comparing the learned CS methods, our model is faster, offering acceptable reconstruction performances (gained 1.56 dB over ReconNet and lost 2.53 dB over CSnet). We explain this gain in speed by the linearity of the reconstruction branch of the proposed CS network (no enhancement step as in [33]). Yet, it is also the reason for the loss in image quality. Still, similar to [22, 33], the proposed solution can easily be adapted to specific target domains since it is learned, such as stereo, medical, and aerial imaging, leading to even better compression performances. Figure 5 and section S2 illustrate qualitative results of the proposed compressive sensing model.

Even when employed in classical training and inference pipelines, L_CSNet is more interesting since it offers shorter encoding/decoding time, mainly due to the linear reconstruction branch (Figure S2.4).

5.3 Analysis of Batch-normalization when training from scratch

To ascertain the contribution of Batch-normalization layers, we train SSD from scratch without BatchNorm as our baseline. As mentioned before, BatchNorm induces a remarkably smoother optimization landscape, permitting a bigger searching space and prompter convergence. When we use the original SSD configuration, our baseline produces 63.4% mAP on VOC 2007, which is 12.5% worse than the performance reached by the detector when it is initialized with a pre-trained classification network (i.e. 75.9%). As for the SSD_BN model, an equivalent mAP to the detector initialized with the pre-trained VGG backbone is reached (i.e. 75.8%). Refer to Table 3 for more details.

5.4 Detection results

After hyperparameter tuning of the newly designed multi-output learning networks (CS_D), Adam optimizer was used since the network did not converge using the SGD optimizer, even though it was the one that permitted the SSD_BN network to converge from scratch. Also, higher learning rate values adversely affect the CS part of the network, leading to divergence (0.001 is used after tuning). Furthermore, the SSD_BN is retrained from scratch using this configuration for a fair comparison (refer to Table 3 for more details).

5.4.1 Detection results on the YYmnist dataset

Extensive experiments were conducted on the YYmnist dataset. For both used backbones, similar mAP is obtained using compressed representations in comparison with the networks using the full input (see Fig. 6). Consequently, it appears that full images are not critical to correctly detecting objects within images (refer to figure S2.5 for a sample image alongside the 4 highest entropy channels of the compressed representation and predicted boxes from this latter). The obtained results aligned with the results of [5, 14, 34], which claims that CNNs are resilient to image compression given that its level is sufficient. Moreover, from Table 4, we can see that the loss in accuracy is negligible compared to the gain in FPS and can be improved by using more sophisticated augmentation pipelines. Also, as mentioned before, the classes of this dataset (MNIST classes) share many characteristics and are not sparse, causing the results of the CS-based SSD models to be close to their original counterparts. After further training of the best performing configurations, the CS_D models reached the baseline one in terms of accuracy (refer to Table 4).

For the VGG-based models, using a block size of $B = 8$ resulted in better improvements in the speed of the models. All different used aspect ratios resulted in almost the same speed with slightly different drops in mAP. The worst performance is obtained for the sampling ratio of 0.01 (compression rate = 99%) with a decline in mAP of 18.6%. In fact, the CS_D_8_5 network is 30% faster then the original SSD while being only 0.017% less accurate on the mAP metric. This speed up gain is due to the fact that the CS_D_8_5 branch used for detection has three convolutional blocks less than the original SSD and thus requires less time to process its inputs and produce predictions. Using a block size of $B = 4$ resulted in a slight improvement in the FPS of models compared to using a block size of$B = 8$ because, first their is an additional convolutional block in the CS_D_4_j models compared to the CS_D_8_j models and second because the shape of the CS measurement using a block size of 4 is larger ($75 \times 75 \times c$ for $B = 4$ vs $38 \times 38 \times c$ for $B = 8$, c is the number of channels and depend on the sampling ratio $\frac{M}{N}$) and thus requiring more time to flow through the detection branch. The most significant improvement is for the sampling ratio of 0.01 with a drop in mAP of 4.1%. From the reported results in Table 4, even though data are compressed using the sampling ratios 0.5, 0.25, 0.1 resulting in more compressed representations (less channels with smaller sampling ratio), the FPS that the models can process seem to saturate with an identical or small drop in performance. In general, the FPS ratio increased when reducing the sampling ratio (increasing the compression rate) and increasing the block size, except for the value 0.5 with a block size of 4. A possible explanation for such a performance behaviour could be that hyper-parameters tuned for a certain configuration is not the best for all configurations. Thus, block size, compression ratio, and image resolution are new hyper-parameters to tune. Similarly for the Mobilenet-based models, the CS_D model is 32.1% faster than the RGB one while being only 3.8% less accurate. Training for an additional 20 epochs reduces this gap to only 1%.

5.4.2 Detection results on the PASCAL VOC dataset

The preliminary results obtained in the first experiments permitted us to choose the best configurations to validate the proposed approach on the Pascal VOC dataset. Four VGG-based configurations are selected (CS_D_4_01, CS_D_8_5, CS_D_8_25, and CS_D_8_1) and will be compared to the SSD_BN network. The chosen models are those that delivered the best results in term of speed-up and accuracy. The results obtained on the PASCAL VOC dataset follow the findings of the first experiments on the YYmnist dataset (Table 5). However, for the pascal VOC dataset, the loss in accuracy is more critical (9.1% for the best configuration). There may be two reasons for this fact, the first one being the sparsity and heterogeneity of PASCAL VOC classes and the second one being the inability of the used CS network to encode relevant features of such dataset.

As it was mentioned before, the CS network proposed to validate our approach is single-scale, lightweight and linear Fig. 3. Therefore, its ability to learn relevant features of many classes is restricted/limited (Figures S4.7 and S4.8 illustrates some detection results on the PASCAL VOC test set). To overcome these limits, we propose a variant of our approach using a multi-layer CS network, described in detail in S3. The rational idea behind this step is the assumption that multi-layer sampling would emphasize the ability of the CS network to represent sparse features. However, the results show that single-scale and multi-scale sampling networks perform equivalently when used in our approach (refer to S3 for detailed results). We believe the reason behind such performance is the absence of both nonlinearity and bias in the sampling network (to maintain compatibility with the conventional CS). According to [34], the recognition network would perform better if an autoencoder architecture is used to obtain the compressed representations. However, the drawback of using an autoencoder is the memory complexity and time the encoder needs to generate the feature maps, which is neither suitable for embedded systems nor real-time applications. Thus, to improve the performance focus should be on the recognition branch. Furthermore, from the obtained results (Fig. 7), we conclude that the detection branch performance is not affected by the image quality metrics, SSIM and PSNR, confirming the no need for the enhancement block in the reconstruction branch in our approach.

As to speed-up gain, we have found that the proposed solution delivers more interesting accelerations on small GPUs. The same implementation is 41.66% faster on Nvidia GTX 950M and 21.62% faster on Tesla V100-SXM2. A possible explanation for this is that powerful GPUs have more RAM (4GB for Nvidia GTX 950M vs. 32GB for Tesla V100-SXM2) and thus can store input data, weight parameters and activations as an input propagates through the network. Worth mentioning that our implementation was not optimized for GPUs; hence, it will deliver better results in terms of speed-up.

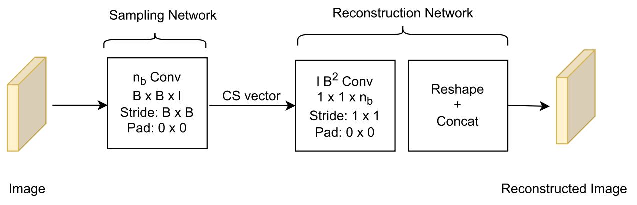


Fig. 3 Proposed light compressive sensing CNN architecture

Table 1 Datasets characteristics

Dataset	Classes	Dataset samples	Observation
PASCAL VOC	20	16,551 (training 07+12); 4,952 (test 07)	Sparse and heterogeneous classes
YYmnist	10	1000 (training/test)	Localize digits from 0 to 9 with different aspect ratios
Mask	1	134 (training); 15 (test)	

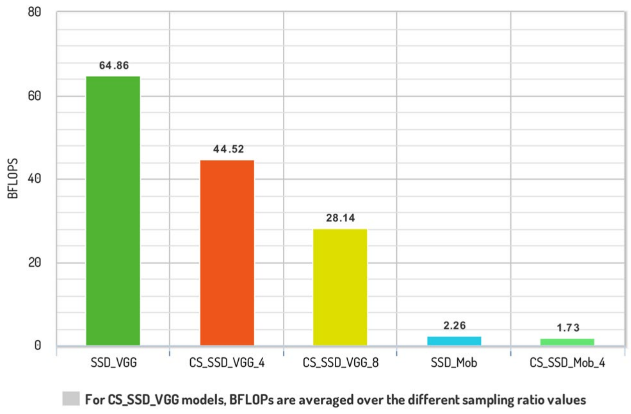


Fig. 4 Model complexity (BFLOPS) for the different networks

Table 2 Average running time and image quality metrics for the lightweight_CSNet, the JPEG standard, ReconNet [22] and CSnet [33] for a sampling ratio of $\frac{M}{N} = 0.1$ on Set11 test images

Method	Avg. running time	SSIM	PSNR	Programming Language
L_CSNet	0.00138	0.85	25.84	Python + TensorFlow
JPEG	0.584	0.978	39.85	/
ReconNet	0.0278	0.662	24.28	Matlab + Matconvenet
CSnet	0.0155	0.815	28.37	Matlab + Matconvenet

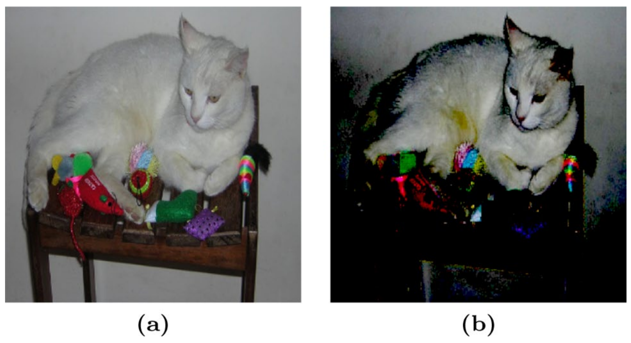


Fig. 5 Sample from PASCAL VOC dataset: (left) original image, (right) reconstructed image using the sampling/reconstruction branch of the CS_D model

Table 3 Analysis of BatchNorm for SSD trained from scratch on VOC 2007 test set

Network	Training type	Configuration	mAP(%)
SSD	From scratch	$lr_{adam} = 0.001$, batchsize =32	63.4%
	Pretrained	$lr_{adam} = 0.001$, batchsize = 32	75.9%
SSD_BN	From scratch	$lr_{sgd} = 0.05$, batchsize = 128	75.8%
	From scratch	$lr_{adam} = 0.001$, batchsize = 128	69.5%

All the networks are based on the truncated VGG backbone network, with and without BatchNorm layers

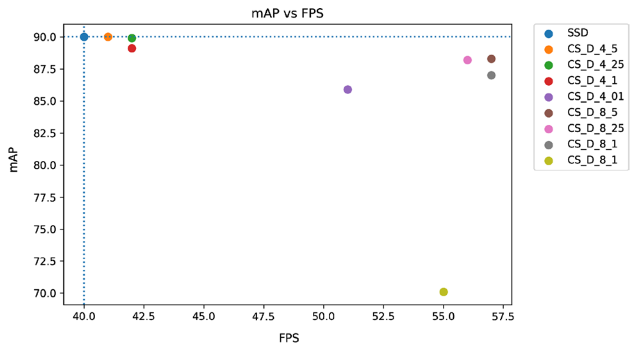


Fig. 6 mAP vs FPS for the CS-based detection networks

5.4.3 Detection results on the Mask dataset

We have also tested one of our best performing models on the Mask dataset, using both the VGG and Mobilenet backbones. For the CS-based models a high compression rate of 75% which correspond to a sampling rate of $\frac{M}{N} = 0.25$. Refer to Table 6 for the obtained results (figure S4.9 shows some detection results obtained using compressed data).

5.5 Approach limitations

The proposed approach is particularly suited for homogenous datasets and wherever the memory constraints and storage are critical. However, when applied to a sparse dataset it fails to learn relevant features for the recognition branch. This weakness arises from the linearity of the CS network,

Table 4 FPS, mAP and image quality metrics for different backbones, sampling rates $\frac{M}{N}$ (averaged over test images) and block sizes

Network	Backbone	B	$\frac{M}{N}$	FPS	mAP_{50}	mAP_{70}	Encoding time (s)	Reconstruction time (s)
SSD	VGG	/	/	40	**90**	/	/	/
CS_D_4_5		4	0.5	41	**90**		0.0027	0.002
CS_D_4_25		4	0.25	42	89.9		0.0027	0.0018
CS_D_4_1		4	0.1	42	89.1		0.0026	0.0017
CS_D_4_01		4	0.01	**51**	85.9		0.0022	0.0013
CS_D_8_5		8	0.5	**57**	**88.9**	89.1	0.0021	0.00143
CS_D_8_25		8	0.25	56	88.8	89	0.0021	0.00134
CS_D_8_1		8	0.1	**57**	88.5	88.7	0.0021	0.00136
CS_D_8_01		8	0.01	55	84.8	84.9	0.0022	0.00146
SSD	Mobilenet	/	/	55	**55.8**	/	/	/
CS_D_4_25		4	0.25	81	52	54.8	0.0020	0.0017

mAP_{50}, mAP after 50 epochs; mAP_{70}, mAP after 70 epochs

FPS is averaged for a batch size of 1

The best performance is marked in bold font

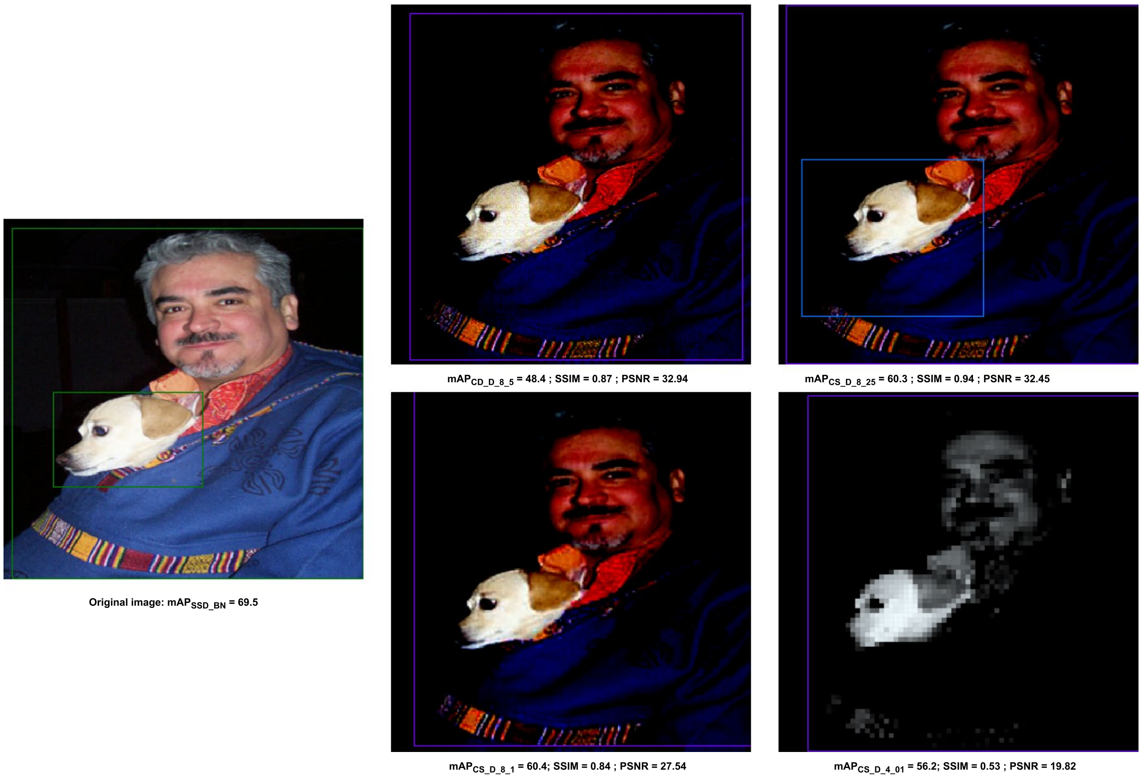

mAP$_{CD\_D\_8\_5}$ = 48.4 ; SSIM = 0.87 ; PSNR = 32.94
mAP$_{CS\_D\_8\_25}$ = 60.3 ; SSIM = 0.94 ; PSNR = 32.45
Original image: mAP$_{SSD\_BN}$ = 69.5
mAP$_{CS\_D\_8\_1}$ = 60.4; SSIM = 0.84 ; PSNR = 27.54
mAP$_{CS\_D\_4\_01}$ = 56.2; SSIM = 0.53 ; PSNR = 19.82

Fig. 7 Detection results, mAP, SSIM and PSNR values over the different trained SSD based models on the Pascal VOC dataset

which is also the strong point of our approach. Subsequent research will focus on enhancing the detection branch outputs while exploring CS-sampled data.

6 Conclusion

This paper proposes a new efficient approach to design CNN for embedded environments with limited resources for real-time applications. Although the existing CNN models achieved state-of-the-art performances, they still

Table 5 mAP, image quality metrics, FPS, encoding and decoding times (averaged over test images) for the different cs based SSD models on different GPUs

				GTX 950M			Tesla V100-SXM2		
Network	mAP	SSIM	PSNR	FPS	Enc. time	Rec. time	FPS	Enc. time	Rec. time
SSD_BN (adam)	**69.5%**	/	/	7	/	/	29		
CS_D_8_5	48.4%	0.87	32.94	11	0.0036	0.0036	34	0.0022	0.0021
CS_D_8_25	60.3%	0.94	32.45	**12**	0.0036	0.0036	**37**	0.0026	0.0021
CS_D_8_1	60.4%	0.84	27.54	**12**	0.0034	0.0035	**37**	0.0022	0.0021
CS_D_4_01	56.2%	0.53	19.82	9	0.0036	0.0037	33	0.0022	0.002

FPS is averaged for batch size of 1

The best performance is marked in bold font

Table 6 FPS, mAP, encoding and decoding times for different models on the Mask dataset after 40 epochs

Network	Backbone	B	$\frac{M}{N}$	FPS	mAP	Enc. time	Rec. time
SSD	VGG			36	86.4		
CS_D_4_25		8	0.25	53	72.8	0.0024	0.0021
SSD	Mobilenet						
CS_D_4_25		4	0.25	66	63	0.0021	0.0019

FPS is averaged for a batch size of 1

suffer from efficiency. To cope with this limit and validate our approach, the single-shot object detector was redesigned using compressed data according to the proposed method. During training, a lightweight CS network is merged with a truncated recognition network for joint learning of sampling /reconstruction and detection weights. During deployment, only the truncated backbone and the detection neck are used to predict from compressed data delivered by a compressive device that uses the sampling network of the designed lightweight CS model. Through our experiments, we showed that the detection models took a parameter reduction in channel deletion and convolution reduction, respectively. Also, the proposed training workflow permitted the augmentation of the dataset during training, which adversely affected the approach proposed in [14]. Our approach is particularly well suited for embedded use, as demonstrated by our tests on the Nvidia GTX 950M. In the future, we think some topics need to be deeply investigated to improve the proposed approach. First, combine our approach with those of the literature that permits the improvement of performance. Also, the optimization of our implementation for GPUs to achieve better results. Furthermore, the combination of our approach with lighter and more efficient backbones such as the one proposed in [39]. Finally, validate the proposed approach using other computer vision tasks, such as segmentation and tracking.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s11554-022-01255-7.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Adler, A., Elad, M., Zibulevsky, M.: Compressed learning: a deep neural network approach. arXiv:1610.09615 (2016)
2. Agustsson, E., Timofte, R.: Ntire 2017 challenge on single image super-resolution: Dataset and study. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops, pp. 126–135 (2017)
3. Bethi, Y.R.T., Narayanan, S., Rangan, V., Thakur, C.S.: Real-time object detection and localization in compressive sensed video on embedded hardware. arXiv:1912.08519 (2019)
4. Bochkovskiy, A., Wang, C.Y., Liao, H.Y.M.: Yolov4: Optimal speed and accuracy of object detection. arXiv:2004.10934 (2020)
5. Bouderbal, I., Amamra, A., Benatia, M.A.: How would image down-sampling and compression impact object detection in the context of self-driving vehicles? In: CSA, pp. 25–37 (2020)
6. Bouderbal, I., Amamra, A., Benatia, M.A.: An analytical study of efficient cnns tuning and scaling for traffic signs recognition. In: 2021 international conference on recent advances in mathematics and informatics (ICRAMI), pp. 1–6, IEEE (2021)
7. Cai, Z., Vasconcelos, N.: Cascade r-cnn: High quality object detection and instance segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence (2019)
8. Deguerre, B., Chatelain, C., Gasso, G.: Fast object detection in compressed jpeg images. In: 2019 IEEE intelligent transportation systems conference (ITSC), pp. 333–338, IEEE (2019)
9. Deguerre, B., Chatelain, C., Gasso, G.: Object detection in the dct domain: is luminance the solution? In: 2020 25th international conference on pattern recognition (ICPR), pp. 2627–2634, IEEE (2021)
10. Ding, K., Ma, K., Wang, S., Simoncelli, E.P.: Comparison of full-reference image quality models for optimization of image processing systems. Int. J. Comput. Vis. **129**(4), 1258–1281 (2021)

11. Fowler, J.E., Mun, S., Tramel, E.W.: Multiscale block compressed sensing with smoothed projected landweber reconstruction. In: 2011 19th European signal processing conference, pp. 564–568, IEEE (2011)
12. Girshick, R.: Fast r-cnn. In: Proceedings of the IEEE international conference on computer vision, pp. 1440–1448 (2015)
13. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 580–587 (2014)
14. Gueguen, L., Sergeev, A., Kadlec, B., Liu, R., Yosinski, J.: Faster neural networks straight from jpeg. Adv. Neural Inf. Process. Syst. **31**, 3933–3944 (2018)
15. Guo, S., Liu, Y., Ni, Y., Ni, W.: Lightweight ssd: real-time lightweight single shot detector for mobile devices. In: VISIGRAPP (5: VISAPP), pp. 25–35 (2021)
16. Han, J., Yang, Y.: L-net: lightweight and fast object detector-based shufflenetv2. J. Real-Time Image Process. **18**(6), 2527–2538 (2021)
17. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision, pp. 2961–2969 (2017)
18. Hore, A., Ziou, D.: Image quality metrics: Psnr vs. ssim. In: 2010 20th international conference on pattern recognition, pp. 2366–2369. IEEE (2010)
19. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: efficient convolutional neural networks for mobile vision applications. arXiv:1704.04861 (2017)
20. Justus, D., Brennan, J., Bonner, S., McGough, A.S.: Predicting the computational cost of deep learning models. In: 2018 IEEE international conference on big data (Big Data), pp. 3873–3882, IEEE (2018)
21. Khosravy, M., Gupta, N., Patel, N., Duque, C.A.: Recovery in compressive sensing: a review. In: Compressive sensing in healthcare, pp. 25–42 (2020)
22. Kulkarni, K., Lohit, S., Turaga, P., Kerviche, R., Ashok, A.: Reconnet: non-iterative reconstruction of images from compressively sensed measurements. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 449–458 (2016)
23. Li, Z., Sun, Y., Tian, G., Xie, L., Liu, Y., Su, H., He, Y.: A compression pipeline for one-stage object detection model. J. Real-Time Image Process. **18**(6), 1949–1962 (2021)
24. Liao, L., Li, K., Yang, C., Liu, J.: Low-cost image compressive sensing with multiple measurement rates for object detection. Sensors **19**(9), 2079 (2019)
25. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: single shot multibox detector. In: European conference on computer vision, pp. 21–37. Springer (2016)
26. Ophoff, T., Van Beeck, K., Goedemé, T.: Exploring rgb+ depth fusion for real-time object detection. Sensors **19**(4), 866 (2019)
27. Padilla, R., Passos, W.L., Dias, T.L., Netto, S.L., da Silva, E.A.: A comparative analysis of object detection metrics with a companion open-source toolkit. Electronics **10**(3), 279 (2021)
28. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 779–788 (2016)
29. Redmon, J., Farhadi, A.: Yolo9000: better, faster, stronger. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 7263–7271 (2017)
30. Redmon, J., Farhadi, A.: Yolov3: an incremental improvement. arXiv:1804.02767 (2018)
31. Ren, K., Huang, L., Fan, C., Han, H., Deng, H.: Real-time traffic sign detection network using ds-detnet and lite fusion fpn. J. Real-Time Image Process. **18**(6), 2181–2191 (2021)
32. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: towards real-time object detection with region proposal networks. Adv. Neural Inf. Process. Syst. **28**, 91–99 (2015)
33. Shi, W., Jiang, F., Liu, S., Zhao, D.: Image compressed sensing using convolutional neural network. IEEE Trans. Image Process. **29**, 375–388 (2019)
34. Torfason, R., Mentzer, F., Agustsson, E., Tschannen, M., Timofte, R., Van Gool, L.: Towards image understanding from deep compression without decoding. arXiv:1803.06131 (2018)
35. Wallace, G.K.: The jpeg still picture compression standard. IEEE Trans. Consumer Electron. **38**(1), 5 (1992)
36. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Process. **13**(4), 600–612 (2004)
37. Xu, D., Shi, Y., Tsang, I.W., Ong, Y.S., Gong, C., Shen, X.: Survey on multi-output learning. IEEE Trans. Neural Netw. Learn. Syst. **31**(7), 2409–2429 (2019)
38. Zhang, J., Zhao, D., Gao, W.: Group-based sparse representation for image restoration. IEEE Trans. Image Process. **23**(8), 3336–3351 (2014)
39. Zhang, X., Xie, H., Zhao, Y., Qian, W., Xu, X.: A fast ssd model based on parameter reduction and dilated convolution. J. Real-Time Image Process. **18**(6), 2211–2224 (2021)
40. Zhao, Z.Q., Zheng, P., Xu, S.T., Wu, X.: Object detection with deep learning: a review. IEEE Trans. Neural Netw. Learn. Syst. **2019**, 5 (2019)
41. Zhou, X., Xu, L., Liu, S., Lin, Y., Zhang, L., Zhuo, C.: An efficient compressive convolutional network for unified object detection and image compression. In: Proceedings of the AAAI conference on artificial intelligence, vol. 33, pp. 5949–5956 (2019)
42. Zhu, R., Zhang, S., Wang, X., Wen, L., Shi, H., Bo, L., Mei, T.: Scratchdet: Training single-shot object detectors from scratch. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 2268–2277 (2019)

Imene Bouderbal is currently a Ph.D. candidate at the Ecole Militaire Polytechnique (EMP), Algeria. Her work focuses on deep-learning based computer vision for real-time applications. She has obtained a Master's degree in computer science and a computer engineering degree from EMP in 2017.