R-CNN minus R

Karel Lenc
http://www.robots.ox.ac.uk/~karel
Andrea Vedaldi
http://www.robots.ox.ac.uk/~vedaldi

Department of Engineering Science,
University of Oxford,
Oxford, UK.

Abstract

Deep convolutional neural networks (CNNs) have had a major impact in most areas of image understanding. In object category detection, however, the best results have been obtained by techniques such as R(egion)-CNN that combine CNNs with cues from image segmentation, using techniques such as selective search to propose possible object locations in images. However, the role of segmentation in CNN detectors remains controversial. On the one hand, segmentation may be a necessary modelling component, carrying essential geometric information not contained in the CNN; on the other hand, it may be merely a way of accelerating detection, by focusing the CNN classifier on promising image areas. In this paper, we answer this question by developing a detector that uses a trivial region generation scheme, constant for each image. While such region proposals approximate objects poorly, we show that a bounding box regressor using intermediate convolutional features can recover sufficiently accurate bounding boxes, demonstrating that, indeed, the required geometric information is contained in the CNN itself. Combined with convolutional feature pooling, we also obtain an excellent and fast detector that does not require to process an image with algorithms other than the CNN itself. We also streamline and simplify the training of CNN-based detectors by integrating several learning steps in a single algorithm, as well as by proposing a number of improvements that accelerate detection.

1 Introduction

Object detection is one of the core problems in image understanding. Until recently, the best performing detectors in standard benchmarks such as PASCAL VOC were based on a combination of handcrafted image representations such as SIFT, HOG, and the Fisher Vector and a form of structured output regression, from sliding window to deformable parts models. Recently, however, these pipelines have been outperformed significantly by the ones based on deep learning that induce representations automatically from data using Convolutional Neural Networks (CNNs). Currently, the best CNN-based detectors are based on the R(egion)-CNN construction of [10]. Conceptually, R-CNN is remarkably simple: it samples image regions using a proposal mechanism such as Selective Search (SS; [21]) and classifies them as foreground and background using a CNN. Looking more closely, however, R-CNN leaves open several interesting question.

The first question is whether a CNN contains sufficient geometric information to localise objects, or whether the latter must be supplemented by an external mechanism, such as region

Pages 5.1-5.12
DOI: https://dx.doi.org/10.5244/C.29.5

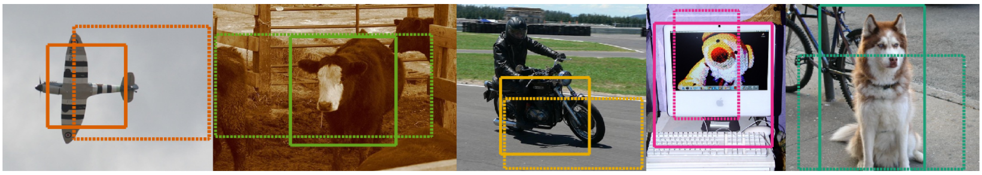

Figure 1: Some examples of the bounding box regressor outputs. The dashed box is the image-agnostic proposal, correctly selected despite the bad overlap, and the solid box is the result of improving it by using the pose regressor. Both steps use the same CNN, but the first uses the geometrically-invariant fully-connected layers, and the last the geometry-sensitive convolutional layers. In this manner, accurate object location can be recovered *without* using complementary mechanisms such as selective search.

proposal generation. There are in fact two hypothesis. The first one is that the only role of proposal generation is to cut down computation by allowing to evaluate the CNN, which is expensive, on a small number of image regions. If this is the case, as other speedups such as SPP-CNN [11] become available, proposal generation becomes less important and could ultimately be removed. The second hypothesis is that, instead, proposal generation provides geometric information which is not represented in the CNN and which is required for accurate object localisation. This is not unlikely, given that CNNs are usually trained to be invariant to large geometric deformations and hence may not be sensitive to an object's location. This question is answered in Section 3.1 by showing that the convolutional layers of standard CNNs contain sufficient information to localise objects (Figure 1).

The second question is whether the R-CNN pipeline can be simplified. While conceptually straightforward, in fact, R-CNN comprises many practical steps that need to be carefully implemented and tuned to obtain a good performance. To start with, R-CNN builds on a CNN pre-trained on an image classification tasks such as ImageNet ILSVRC [6]. This CNN is ported to detection by: i) learning an SVM classifier for each object class on top of the last fully-connected layer of the network, ii) fine-tuning the CNN on the task of discriminating objects and background, and iii) learning a bounding box regressor for each object class. Section 3.2 simplifies these steps, which require running a mix of different software on cached data, by training a single CNN addressing all required tasks. A similar simplification can also be found in a very recent version update to R-CNN, namely Fast R-CNN [9].

The third question is whether R-CNN can be accelerated. A substantial speedup was already obtained in *spatial pyramid pooling* (SPP) by [11] by realising that convolutional features can be shared among different regions rather than being recomputed. However, this does not accelerate training, and in testing the region proposal generation mechanism becomes the new bottleneck. The combination of dropping proposal generation and of the other simplifications are shown in Section 4 to provide a substantial detection speedup – and this for the overall system, not just the CNN part. This is alternative to the very recent Faster R-CNN [17] scheme, which instead uses the convolutional features to construct a new efficient proposal generation scheme. Our findings are summarised in Section 5.

Related work. The basis of our work are the current generation of deep CNNs for image understanding, pioneered by [14]. One of the first frameworks for object detection with CNNs is OverFeat framework [18] which tackles object detection by performing sliding window on a feature map produced by CNN layers followed by bounding box regression (BBR). Even though authors introduce a way how to efficiently increase the number of evaluated locations, their sliding window approach is limited to single aspect ratio bounding boxes (before BBR). For object detection, our method builds directly on the R-CNN approach of [10] as well as the SPP extension proposed in [11]. All such methods rely not only on CNNs, but also on a region proposal generation mechanism such as SS [21], CPMC [3], multi-scale combinatorial grouping [2], and edge boxes [27]. These methods, which are extensively reviewed in [12], originate in the idea of "objectness" proposed by [1]. Interestingly, [12] showed that a good region proposal scheme is essential for R-CNN to work well. Here, we show that this is in fact *not* the case provided that bounding box locations are corrected by a strong CNN-based bounding box regressor, a step that was not evaluated for R-CNNs in [12]. The R-CNN and SPP-CNN detectors build on years of research in object detection. Both can be seen as accelerated sliding window detectors [5, 24]. The two-stage computation using region proposal generation is a form of cascade detector [24] or jumping window [20, 23]. However, they differ in part-based detector such as [8] in that they do not explicitly model object parts in learning; instead parts are implicitly captured in the CNN. As noted above, ideas similar or alternative to Section 3.2 have been recently introduced in [9] and [17].

2 CNN-based detectors

This section summarises the R-CNN (Section 2.1) and SPP-CNN (Section 2.2) detectors.

2.1 The R-CNN detector

The R-CNN method [10] is a chain of conceptually simple steps: generating candidate object regions, classifying them as foreground or background, and post-processing them to improve their fit to objects. These steps are described next.

Region proposal generation. R-CNN starts by an algorithm such as SS [21] or CPMC [3] to extracts from an image $\mathbf{x}$ a shortlist of image regions $R \in \mathcal{R}(\mathbf{x})$ that are likely to tightly enclose objects. These proposals, in the order of a few thousands per image, may have arbitrary shapes, but are converted to rectangles before further processing.

CNN-based features. Candidate regions are described by CNN features before being classified. The CNN itself is *transferred* from a different problem – usually image classification in the ImageNet ILSVRC challenge [6]. In this manner, the CNN can be trained on a very large dataset, as required to obtain good performance, and then applied to object detection, where datasets are usually much smaller. In order to transfer a pre-trained CNN to object detection, its last few layers, which are specific to the classification task, are removed; this results in a "beheaded" CNN ϕ that outputs relatively generic features. The CNN is applied to the image regions R by cropping and resizing the image $\mathbf{x}$, *i.e.* $\phi_{\text{RCNN}}(\mathbf{x};R) = \phi(\text{resize}(\mathbf{x}|_R))$. Cropping and resizing serves two purposes: to localise the descriptor and to provide the CNN with an image of a fixed size, as this is required by many CNN architectures.

SVM training. Given the region descriptor $\phi_{\text{RCNN}}(\mathbf{x};R)$, the next step is to learn a SVM classifier to decide whether a region contains an object or background. Learning the SVM starts from a number of example images $\mathbf{x}_1,\dots,\mathbf{x}_N$, each annotated with ground-truth regions $\bar{R} \in \mathcal{R}_{\text{gt}}(\mathbf{x}_i)$ and object labels $c(\bar{R}) \in \{1,\dots C\}$. In order to learn a classifier for class $\bar{c}$, R-CNN divides ground-truth $\mathcal{R}_{\text{gt}}(\mathbf{x}_i)$ and candidate $\mathcal{R}(\mathbf{x}_i)$ regions into positive and negative. In particular, ground truth regions $R \in \mathcal{R}_{\text{gt}}(\mathbf{x}_i)$ for class $c(R) = \bar{c}$ are assigned a positive label $y(R;\bar{c};\tau) = +1$; other regions R are labelled as ambiguous $y(R;\bar{c};\tau) = \varepsilon$ and ignored if $\text{overlap}(R,\bar{R}) \geq \tau$ with any ground truth region $\bar{R} \in \mathcal{R}_{\text{gt}}(\mathbf{x}_i)$ of the same class $c(\bar{R}) = \bar{c}$. The remaining regions are labelled as negative. Here $\text{overlap}(A,B) = |A \cap B|/|A \cup B|$ is the intersection-over-union overlap measure, and the threshold is set to $\tau = 0.3$. The SVM takes the form $\phi_{\text{SVM}} \circ \phi_{\text{RCNN}}(\mathbf{x};R)$, where ϕ_{SVM} is a linear predictor $\langle w_c, \phi_{\text{RCNN}} \rangle + b_c$ learned using an SVM solver to minimise the regularised empirical hinge loss risk on the training regions.

Bounding box regression. Candidate bounding boxes are refitted to detected objects by using a CNN-based regressor as detailed in [10]. Given a candidate bounding box $R = (x,y,w,h)$, where (x,y) are its centre and (w,h) its width and height, a linear regressor estimates an adjustment $\mathbf{d} = (d_x,d_y,d_w,d_h)$ that yields the new bounding box $\mathbf{d}[R] = (wd_x + x, hd_y + y, we^{d_w}, he^{d_h})$. In order to train this regressor, one collects for each ground truth region R^* all the candidates R that overlap sufficiently with it (with an overlap of at least 0.5). Each pair (R^*,R) of regions is converted in a training input/output pair $(\phi_{\text{cnv}}(\mathbf{x},R),\mathbf{d})$ for the regressor, where $\mathbf{d}$ is the adjustment required to transform R into R^*, i.e. $R^* = \mathbf{d}[R]$. The pairs are then used to train the regressor using ridge regression with a large regularisation constant. The regressor itself takes the form $\mathbf{d} = Q_c^\top \phi_{\text{cnv}}(\text{resize}(\mathbf{x}|_R)) + t_c$ where ϕ_{cnv} denotes the CNN restricted to the convolutional layers, as further discussed in Section 2.2. The regressor is further improved by retraining it after removing the 20% of the examples with worst regression loss – as found in the publicly-available implementation of SPP-CNN.

Post-processing. The refined bounding boxes are passed to non-maxima suppression before being evaluated. Non-maxima suppression eliminates duplicate detections prioritising regions with higher SVM score $\phi_{\text{SVM}} \circ \phi_{\text{RCNN}}(\mathbf{x};R)$. Starting from the highest ranked region in an image, other regions are iteratively removed if they overlap by more than 0.3 with any region retained so far.

CNN fine-tuning. The quality of the CNN features, ported from an image classification task, can be improved by *fine-tuning* the network on the target data. In order to do so, the CNN $\phi_{\text{RCNN}}(\mathbf{x};R)$ is concatenated with additional layers ϕ_{sftmx} (a linear projection followed by softmax normalisation) to obtain a predictor for the $C+1$ object classes. The new CNN $\phi_{\text{sftmx}} \circ \phi_{\text{RCNN}}(\mathbf{x};R)$ is then trained as a classifier by minimising its empirical logistic risk on a training set of labelled regions. This is analogous to the procedure used to learn the CNN in the first place, but with a reduced learning rate and a different (and smaller) training set similar to the one used to train the SVM. In this dataset, a region R, either ground-truth or candidate, is assigned the class $c(R;\tau_+,\tau_-) = c(\bar{R}^*)$ of the closest ground-truth region $\bar{R}^* = \text{argmax}_{\bar{R} \in \mathcal{R}_{\text{gt}}(\mathbf{x})} \text{overlap}(R,\bar{R})$, provided that $\text{overlap}(R,\bar{R}^*) \geq \tau_+$. If instead $\text{overlap}(R,\bar{R}^*) < \tau_-$, then the region is labelled as $c(R;\tau_+,\tau_-) = 0$ (background), and the remaining regions as ambiguous and ignored. By default τ_+ and τ_- are both set $1/2$, resulting in a much more relaxed training set than for the SVM. Since the dataset is strongly biased towards background regions, during CNN training it is rebalanced by sampling with 25% probability regions such that $c(R) > 0$ and with 75% probability regions such that $c(R) = 0$.

2.2 SPP-CNN detector

A significant disadvantage of R-CNN is the need to recompute the whole CNN from scratch for each evaluated region; since this occurs thousands of times per image, the method is slow. SPP-CNN addresses this issue by factoring the CNN $\phi = \phi_{\text{fc}} \circ \phi_{\text{cnv}}$ in two parts, where ϕ_{cnv} contains the so-called *convolutional* layers, pooling information from local regions, and ϕ_{fc} the *fully connected* (FC) ones, pooling information from the image as a whole. Since the convolutional layers encode *local information*, this can be selectively pooled to encode the appearance of an image subregion R instead of the whole image [4, 11]. In more detail, let $\mathbf{y} = \phi_{\text{cnv}}(\mathbf{x})$ the output of the convolutional layers applied to image $\mathbf{x}$. The feature field $\mathbf{y}$ is a $H \times W \times D$ tensor of height H and width W, proportional to the height and width of the input image $\mathbf{x}$, and D feature channels. Let $\mathbf{z} = SP(\mathbf{y};R)$ be the result of applying the *spatial pooling* (SP) operator to the feature in $\mathbf{y}$ contained in region R. This operator is defined as:

$$z_d = \max_{(i,j):g(i,j)\in R} y_{ijd}, \qquad d = 1,\dots,D \tag{1}$$

where the function g maps the feature coordinates (i,j) back to image coordinates $g(i,j)$. The SP operator is extended to *spatial pyramid pooling* (SPP; [15]) by dividing the region R into subregions $R = R_1 \cup R_2 \cup \dots R_K$, applying the SP operator to each, and then stacking the resulting features. In practice, SSP-CNN uses $K \times K$ subdivisions, where K is chosen to match the size of the convolutional feature field in the original CNN. In this manner, the output can be concatenated with the existing FC layers: $\phi_{\text{SPP}}(\mathbf{x};R) = \phi_{\text{fc}} \circ \text{SPP}(\cdot;R) \circ \phi_{\text{cnv}}(\mathbf{x})$. Note that, compared to R-CNN, the first part of the computation is shared among all regions R.

Next, we derive the map g that transforms feature coordinates back to image coordinates as required by (1) (this correspondence was approximated in [11]). It suffices to consider one spatial dimension. The question is which pixel $\mathbf{x}_0(i_0)$ corresponds to feature $\mathbf{x}_L(i_L)$ in the L-th layer of a CNN. While there is no unique definition, a useful one is to let i_0 be the *centre of the receptive field* of feature $\mathbf{x}_L(i_L)$, defined as the set of pixels $\Omega_L(i_L)$ that can affect $\mathbf{x}_L(i_L)$ as a function of the image (i.e. the support of the feature seen as a function). A short calculation leads to

$$i_0 = g_L(i_L) = \alpha_L(i_L - 1) + \beta_L, \quad \alpha_L = \prod_{p=1}^{L} S_p, \quad \beta_L = 1 + \sum_{p=1}^{L} \left(\prod_{q=1}^{p-1} S_q \right) \left(\frac{F_p - 1}{2} - P_p \right),$$

where the CNN layers are described geometrically by: padding P_l, downsampling factor S_l, and filter width F_l. The meaning of these parameters is obvious for linear convolution and spatial pooling layers; most of other layers can also be thought of as "convolutional" (*e.g.* ReLU) but with null padding, no (unitary) subsampling, and unitary filter width.

Given the definition of g, similarly to [11] equation (1) pools the features whose receptive field centre is contained in the image region R

3 Simplifying and streamlining R-CNN

This section describes the main technical contributions of the paper: removing region proposal generation from R-CNN (Section 3.1) and streamlining the pipeline (Section 3.2).

3.1 Dropping region proposal generation

While the SPP method of [11] (Section 2.2) accelerates R-CNN evaluation by orders of magnitude, it does not result in a comparable acceleration of the detector as a whole; in fact, proposal generation with SS is about ten time slower than SPP classification. Much faster proposal generators exist, but may not result in very accurate regions [26]. However, this might not be a problem if accurate object location can be recovered by the CNN. Here we take this idea to the limit: we drop $\mathcal{R}(\mathbf{x})$ entirely and to use instead an image-independent list of candidate regions $\mathcal{R}_0$, relying on the CNN for accurate localisation *a-posteriori*.

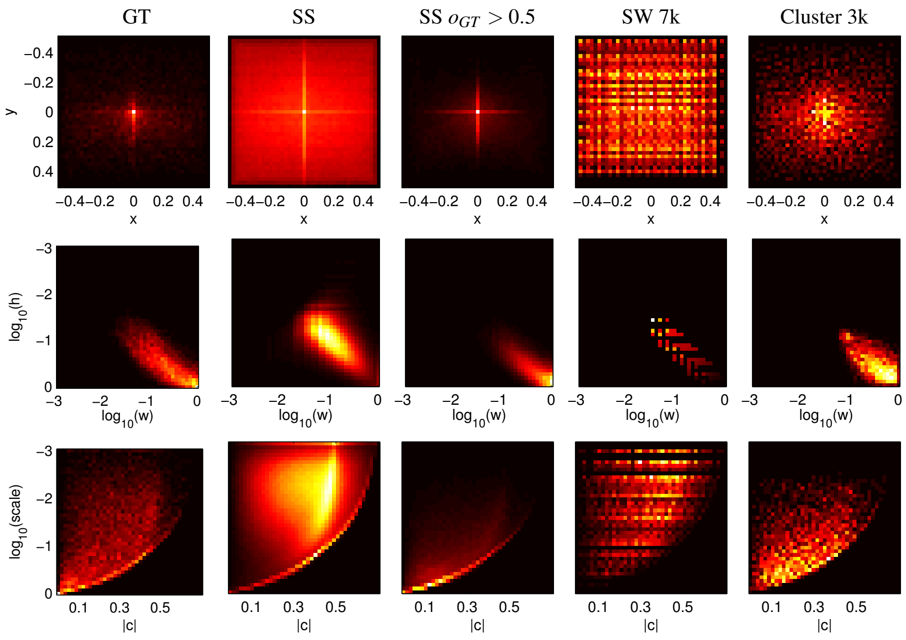

Figure 2: Bounding box distributions using the normalised coordinates of Section 3.1. Rows show the histograms for the bounding box centre (x,y), size (w,h), and scale vs distance from centre $(s,|c|)$. Column shows the statistics for ground-truth, selective search, restricted selective search, sliding window, and cluster bounding boxes (for $n = 3000$ clusters).

Constructing R_0 starts by studying the distribution of bounding boxes in a representative object detection benchmark, namely the PASCAL VOC 2007 data [7]. A box is defined by the tuple (r_s, c_s, r_e, c_e) denoting the upper-left and lower-right corners coordinates (r_s, c_s) and (r_e, c_e). The bounding box centre is then given by $(x,y) = \frac{1}{2}(c_e + c_s, r_e + r_s)$. Given an image of size $H \times W$, we define the normalised width and height as $w = (c_e - c_s)/W$ and $h = (r_e - r_s)/H$ respectively; we define also the scale as $s = \sqrt{wh}$ and distance from the image centre as $|c| = \| [(c_s + c_e)/2W - 0.5, (r_s + r_e)/2H - 0.5)] \|_2$.

The first column of Figure 2 shows the distribution of such parameters for the GT boxes in the PASCAL data. It is evident that boxes tend to appear close to the image centre and to fill the image. The statistics of SS regions differs substantially; in particular, the $(s,|c|)$ histogram shows that SS boxes tend to distribute much more uniformly in scale and space compared to the GT ones. If SS boxes are restricted to the ones that have an overlap of at least 0.5 with a GT BB, then the distributions are similar again, with a strong preference for centred and large boxes.

The fourth column shows the distribution of boxes generated by a *sliding window* (SW; [5]) object detector. For an "exhaustive" enumeration of boxes at all location, scales, and aspect ratios, there can be hundred of thousands boxes per image. Here we subsample this set to 7K in order to obtain a candidate set with a size comparable to SS. This was obtained by sampling the width of the bounding boxes as $w = w_0 2^l, l = 0, 0.5, \dots 4$ where $w_0 \approx 40$ pixels is the width of the smallest bounding box considered in the SSP-CNN detector. Similarly, aspect ratios are sampled as $2^{\{-1,-0.75,\dots 1\}}$. The distribution of boxes, visualised in the fourth column of Figure 2, is similar to SS and dissimilar from GT. This is much denser sampling than in the OverFeat framework [18] which evaluates approximately 1.6K boxes per image with a single aspect ratio only.

A simple modification of sliding window is to bias sampling to match the statistics of the GT bounding boxes. We do so by computing n K-means clusters from the collection of vectors (r_s, c_s, r_e, c_e) obtained from the GT boxes in the PASCAL VOC training data. We call this set of boxes $\mathcal{R}_0(n)$; the fifth column of Figure 2 shows that, as expected, the corresponding distribution matches nicely the one of GT even for a small set of $n = 3000$ cluster centres. Section 4 shows empirically that, when combined with a CNN-based bounding box regressor, this proposal set results in a very competitive (and very fast) detector.

3.2 Streamlined detection pipeline

This section proposes several simplifications to the R/SPP-CNN pipelines complementary to dropping region proposal generation as done in Section 3.1. As a result of all these changes, the whole detector, including detection of multiple object classes and bounding box regression, *reduces to evaluating a single CNN*. Furthermore, the pipeline is straightforward to implement on GPU, and is sufficiently memory-efficient to process multiple images at once. In practice, this results in an extremely fast detector which still retains excellent performance.

Dropping the SVM. As discussed in Section 2.1, R-CNN involves training an SVM classifier for each target object class as well as fine-tuning the CNN features for all classes. An obvious question is whether SVM training is redundant and can be eliminated.

Recall from Section 2.1 that fine-tuning learns a softmax predictor ϕ_{sftmx} on top of R-CNN features $\phi_{\text{RCNN}}(\mathbf{x}; R)$, whereas SVM training learns a linear predictor ϕ_{SVM} on top of the same features. In the first case, $P_c = P(c|\mathbf{x}, R) = [\phi_{\text{sftmx}} \circ \phi_{\text{RCNN}}(\mathbf{x}; R)]_c$ is an estimate of the class posterior for region R; in the second case $S_c = [\phi_{\text{SVM}} \circ \phi_{\text{RCNN}}(\mathbf{x}; R)]_c$ is a score that discriminates class c from any other class (in both cases background is treated as one of the classes). As verified in Section 4 and Table 1, P_c works poorly as a score for an object detector; however, and somewhat surprisingly, using as score the ratio $S'_c = P_c/P_0$ (where P_0 is the probability of the background class) results in performance nearly as good as using an SVM. Further, note that ϕ_{sftmx} can be decomposed as $C+1$ linear predictors $\langle w_c, \phi_{\text{RCNN}} \rangle + b_c$ followed by exponentiation and normalisation; hence the scores S'_c reduces to the expression $S'_c = \exp\left(\langle w_c - w_0, \phi_{\text{RCNN}} \rangle + b_c - b_0\right)$.

Integrating SPP and bounding box regression. While in the original implementation of SPP [11] the pooling mechanism is external to the CNN software, we implement it directly as a layer $\text{SPP}(\cdot; R_1, \dots, R_n)$. This layer takes as input a tensor representing the convolutional features $\phi_{\text{cnv}}(\mathbf{x}) \in \mathbb{R}^{H \times W \times D}$ and outputs n feature fields of size $h \times w \times D$, one for each region $R_1, \dots, R_n$ passed as input. These fields can be stacked in a 4D output tensor, which is supported by all common CNN software. Given a dual CPU/GPU implementation of the layer, SPP integrates seamlessly with most CNN packages, with substantial benefit in speed and flexibility, including the possibility of training with back-propagation through it.

Similar to SPP, bounding box regression is easily integrated as a bank of filters $(Q_c, b_c), c = 1, \dots, C$ running on top of the convolutional features $\phi_{\text{cnv}}(\mathbf{x})$. This is cheap enough that can be done in parallel for all the object classes in PASCAL VOC.

Scale-augmented training, single scale evaluation. While SPP is fast, one of the most time consuming step is to evaluate features at multiple scales [11]. However, the authors of [11] also indicate that restricting evaluation to a single scale has a marginal effect in performance.

Evaluation method	Single scale		Multi scale	
BB regression	no	yes	no	yes
S_c (SVM)	54.0	58.6	56.3	59.7
P_c (softmax)	27.9	34.5	30.1	38.1
P_c/P_0 (modified softmax)	54.0	58.0	55.3	58.4

Table 1: Evaluation of SPP-CNN with and without the SVM classifier. The table report mAP on the PASCAL VOC 2007 test set for the single scale and multi scale detector, with or without bounding box regression. Different rows compare different bounding box scoring mechanism of Section 3.2: the SVM scores S_c, the softmax posterior probability scores P_c, and the modified softmax scores P_c/P_0.

method	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
SVM MS	**59.68**	66.8	75.8	55.5	43.1	38.1	66.6	73.8	70.9	29.2	71.4	58.6	65.5	76.2	73.6	57.4	29.9	60.1	48.4	66.0	66.8
SVM SS	**58.60**	66.1	76.0	54.9	38.6	32.4	66.3	72.8	69.3	30.2	67.7	63.7	66.2	72.5	71.2	56.4	27.3	59.5	50.4	65.3	65.2
FC8 MS	**58.38**	69.2	75.2	53.7	40.0	33.0	67.2	71.3	71.6	26.9	69.6	60.3	64.5	74.0	73.4	55.6	25.3	60.4	47.0	64.9	64.4
FC8 SS	**57.99**	67.0	75.0	53.3	37.7	28.3	69.2	71.1	69.7	29.7	69.1	62.9	64.0	72.7	71.0	56.1	25.6	57.7	50.7	66.5	62.3
FC8 C3k MS	**53.41**	55.8	73.1	47.5	36.5	17.8	69.1	55.2	73.1	24.4	49.3	63.9	67.8	76.8	71.1	48.7	27.6	42.6	43.4	70.1	54.5
FC8 C3k SS	**53.52**	55.8	73.3	47.3	37.3	17.6	69.3	55.3	73.2	24.0	49.0	63.3	68.2	76.5	71.3	48.2	27.1	43.8	45.1	70.2	54.6

Table 2: Comparison of different variants of the SPP-CNN detector. First group of rows: original SPP-CNN using Multi Scale (MS) or Single Scale (SS) detection. Second group: the same experiment, but dropping the SVM and using the modified softmax scores of Section 3.2. Third group: SPP-CNN *without region proposal generation*, but using a fixed set of 3K candidate bounding boxes as explained in Section 3.1.

Here, we maintain the idea of evaluating the detector at test time by processing each image at a single scale. However, this requires the CNN to *explicitly learn* scale invariance, which is achieved by fine-tuning the CNN using randomly rescaled versions of the training data.

4 Experiments

This section evaluates the changes to R-CNN and SPP-CNN proposed in Section 3. All experiments use the Zeiler and Fergus (ZF) small CNN [25] as this is the same network used by [11] that introduce SPP-CNN. While more recent networks such as the very deep models of Simonyan and Zisserman [19] are likely to perform better, this choice allows to compare directly [11]. The detector itself is trained and evaluated on the PASCAL VOC 2007 data [7], as this is a default benchmark for object detection and is used in [11] as well.

Dropping the SVM. The first experiment evaluates the performance of the SPP-CNN detector with or without the linear SVM classifier, comparing the bounding box scores S_c (SVM), P_c (softmax), and S'_c (modified softmax) of Section 3.2. As can be seen in Table 1 and Table 2, the best performing method is SSP-CNN evaluated at multiple scales, resulting in 59.7% mAP on the PASCAL VOC 2007 test data (this number matches the one reported in [11], validating our implementation). Removing the SVM and using the CNN softmax scores directly performs really poorly, with a drop of 21.6% mAP point. However, adjusting the softmax scores using the simple formula P_c/P_0 restores the performance almost entirely, back to 58.4% mAP. While there is still a small 1.3% drop in mAP accuracy compared to using the SVM, removing the latter dramatically simplifies the detector pipeline, resulting in particular in significantly faster training as it removes the need of preparing and caching data for the SVM (as well as learning it).

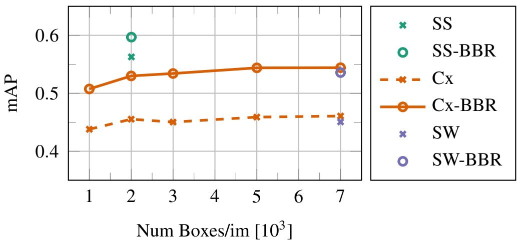

Figure 3: mAP on the PASCAL VOC 2007 test data as a function of the number of candidate boxes per image, proposal generation method, and using or not bounding box regression. In all cases, the CNN is fine-tuned for the particular bounding-box generation algorithm.

Impl. [ms]		SelS	Prep.	Move	Conv	SPP	FC	BBR	$\Sigma -$ SelS	
SPP	MS	$1.98 \cdot 10^3$	23.3	67.5	186.6	211.1	91.0	39.8	**619.2**	$\pm$118.0
OURS			23.7	17.7	179.4	38.9	87.9	9.8	**357.4**	$\pm$34.3
SPP	SS		9.0	47.7	31.1	207.1	90.4	39.9	**425.1**	$\pm$117.0
OURS			9.0	3.0	30.3	19.4	88.0	9.8	**159.5**	$\pm$31.5

Table 3: Timing (in *ms*) of the original SPP-CNN and our streamlined full-GPU implementation, broken down into selective search (SS) and preprocessing: image loading and scaling (Prep), CPU/GPU data transfer (Move), convolution layers (Conv), spatial pyramid pooling (SPP), fully connected layers and SVM evaluation (FC), and bounding box regression (BBR). The performance of the tested classifiers is referred in the first two rows of Table 2.

Multi-scale evaluation. The second set of experiments assess the importance of performing multi-scale evaluation of the detector. Results are reported once more in Tables 1 and 2. Once more, multi-scale detection is the best performing method, with performance up to 59.7% mAP. However, single scale testing is very close to this level of performance, at 58.6%, with a drop of just 1.1% mAP points. Just like when removing the SVM, the resulting simplification and in this case detection speedup make this drop in accuracy more than tolerable. In particular, testing at a single scale accelerates detection roughly five-folds.

Dropping region proposal generation. The next experiment evaluates replacing the SS region proposals $\mathcal{R}_{\mathrm{SS}}(\mathbf{x})$ with the fixed proposals $\mathcal{R}_0(n)$ as suggested in Section 3.1 (fine-tuning the CNN and retraining the bounding-box regression algorithm for the different region proposals in the training set). Table 2 shows the detection performance for $n = 3{,}000$, a number of candidates comparable with the 2,000 extracted by selective search. While there is a drop in performance compared to using SS, this is small (59.68% vs 53.41%, i.e. a 6.1% reduction), which is surprising since bounding box proposals are now oblivious of the image content.

Figure 3 looks at these results in greater detail. Three bounding box generation methods are compared: selective search, sliding windows, and clustering (see also Section 3.1), with or without bounding box regression. Neither clustering nor sliding windows result in an accurate detector: even if the number of candidate boxes is increased substantially (up to $n = 7K$), performance saturates at around 46% mAP. This is much poorer than the ~56% achieved by selective search. Bounding box regression improves selective search by about 3% mAP, up to ~59%, but it has a much more significant effect on the other two methods, improving performance by about 10% mAP. Note that clustering with 3K candidates performs as well as sliding window with 7K.

We can draw several interesting conclusions. First, for the same low number of candidate boxes, selective search is much better than any fixed proposal set; less expected is that performance does not increase even with 2× more candidates, indicating that the CNN is *unable to tell which bounding boxes wrap objects better* even when tight boxes are contained in the shortlist of proposals. This can be explained by the high degree of geometric invariance in the CNN. At the same time, the CNN-based bounding box regressor can make loose bounding boxes significantly tighter, which requires geometric information to be preserved by the CNN. This apparent contradiction can be explained by noting that bounding box classification is built on top of the FC layers of the CNN, whereas bounding box regression is built on the convolutional ones. Evidently, geometric information is removed in the FC layers, but is still contained in the convolutional layers (see also Figure 1).

Detection speed. The last experiment (Table 3) evaluates the detection speed of SPP-CNN (which is already orders of magnitude faster than R-CNN) and our streamlined implementation using the *MatConvNet* [22] (the original SPP detector is using *Caffe* [13] with identical GPU kernels). Not counting SS proposal generation, the streamlined implementation is between 1.7× (multi-scale) to 2.6× (single-scale) faster than original SPP, with the most significant gain emerging from the integrated SPP and bounding box regression implementation on GPU and consequent reduction of data transfer cost between CPU and GPU.

As suggested before, however, the bottleneck is selective search. Compared to the slowest MS SPP-CNN implementation of [11], using all the simplifications of Section 3, including removing selective search, results in an overall detection speedup of more than 16×, from about 2.5s per image down to 160ms (this at a reduction of about 6% mAP points).

5 Conclusions

Our most significant finding is that current CNNs do not require to be supplemented with accurate geometric information obtained from segmentation based methods to achieve accurate object detection. The necessary geometric information is in fact contained in the CNN, albeit in the intermediate convolutional layers instead of the deeper fully-connected ones (this finding is independently corroborated by visualisations such as the ones in [16]). This does not mean that proposal generation is not useful; in particular, in datasets such as MSR COCO that contain many small objects a fixed list of proposal might not work as well as it does for PASCAL VOC; however, our findings mean that fairly coarse proposals are sufficient as geometric information can be extracted frogit m the CNN.

These findings open the possibility of building state-of-the-art object detectors that rely exclusively on CNNs, removing region proposal generation schemes such as selective search, and resulting in integrated, simpler, and faster detectors. Our current implementation of a proposal-free detector is already much faster than SPP-CNN, and very close, but not quite as good, in term of mAP. However, we have only begun exploring the design possibilities and we believe that it is a matter of time before the gap closes entirely. In fact, papers recently appeared in arXiv, such as [17], appear to be heading in this direction.

6 Acknowledgements

Financial support for Karel Lenc was provided by a BP industrial grant and by a DTA of the Engineering Science Department of the University of Oxford.

References

[1] B. Alexe, T. Deselaers, and V. Ferrari. What is an object? In *Proc. CVPR*, 2010.

[2] P. Arbeláez, J. Pont-Tuset, J. T. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *Proc. CVPR*, 2014.

[3] J. Carreira and C. Sminchisescu. Cpmc: Automatic object segmentation using constrained parametric min-cuts. In *PAMI*, 2012.

[4] M. Cimpoi, S. Maji, and A. Vedaldi. Deep convolutional filter banks for texture recognition and segmentation. In *Proc. CVPR*, 2015.

[5] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proc. CVPR*, 2005.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *Proc. CVPR*, 2009.

[7] M. Everingham, A. Zisserman, C. Williams, and L. Van Gool. The PASCAL visual obiect classes challenge 2007 (VOC2007) results. Technical report, Pascal Challenge, 2007.

[8] P. F. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *Proc. CVPR*, 2008.

[9] R. Girshick. Fast RCNN. In *arXiv*, number arXiv:1504.08083, 2015.

[10] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proc. CVPR*, 2014.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *Proc. ECCV*, 2014.

[12] J. Hosang, R. Beneson, P. Dollár, and B. Schiele. What makes for effective detection proposals? *arXiv:1502.05082*, 2015.

[13] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proc. NIPS*, 2012.

[15] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bag of features: Spatial pyramid matching for recognizing natural scene categories. In *Proc. CVPR*, 2006.

[16] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Proc. CVPR*, 2015.

[17] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *arXiv:1506.01497*, 2015.

[18] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *CoRR*, abs/1312.6229, 2013.

[19] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[20] J. Sivic, B. C. Russel, A. A. Efros, A. Zisserman, and W. T. Freeman. Discovering objects and their location in images. In *Proc. ICCV*, 2005.

[21] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. Selective search for object recognition. *IJCV*, 2013.

[22] A. Vedaldi and K. Lenc. Matconvnet – convolutional neural networks for MATLAB. *CoRR*, abs/1412.4564, 2014.

[23] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *Proc. ICCV*, 2009.

[24] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. CVPR*, 2001.

[25] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *Proc. ECCV*, 2014.

[26] Q. Zhao and Z. Liu an B. Yin. Cracking bing and beyond. In *Proc. BMVC*, 2014.

[27] C. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *Proc. ECCV*, 2014.