Context Forest for Object Class Detection

Davide Modolo
davide.modolo@gmail.com

Alexander Vezhnevets
vezhnick@gmail.com

Vittorio Ferrari
vittoferrari@gmail.com

University of Edinburgh
Edinburgh, UK

Abstract

We present Context Forest (ConF) — a technique for predicting properties of the objects in an image based on its global appearance. Compared to standard nearest-neighbour techniques, ConF is more accurate, fast and memory efficient. We demonstrate ConF by predicting three properties: aspects of appearance, location in the image, and class membership. In extensive experiments we show that (i) ConF can automatically select which components of a multi-component detector to run on a given test image, obtaining a considerable speed-up for detectors trained from large sets (10× for EE-SVMs [36] and 2× for DPM [21]); (ii) ConF can improve object detection performance by removing false positive detections at unlikely locations (+2% mAP), and by (iii) removing false positives produced by classes unlikely to be present in the image (+5% mAP on a 200-class dataset [2]).

1 Introduction

Global image appearance carries information about properties of objects in the image. For instance, a picture of a highway taken from a car is more likely to contain cars from the back viewpoint than from the side (fig. 1) and a picture taken under water is more likely to contain a fish than a car. This shows how the global image appearance of images can help understanding what objects are present and what they look like. Moreover, another property that can be inferred from global image appearance is the rough location of object instances [45]. For instance, an urban scene with cars parked in front of a building, shows cars in the bottom half of the image (fig. 2).

In this paper we exploit this observation for object class detection. We propose Context Forest (ConF): a technique for learning the relation between the global appearance of an image and the properties of the objects it contains. Given only the global appearance of a test image, ConF retrieves a subset of training images that contain objects with similar properties. ConF is based on the Random Forest [8, 10] framework, which provides high computational efficiency and the ability to learn complex, non-linear relations between global image appearance and objects properties. It is very flexible and only requires these properties to be defined through a distance function between two object instances, e.g. their appearance similarity or difference in location. We demonstrate ConF by learning to predict three properties: aspects of appearance, location in the image, and class membership.
Multi-component detectors \([4, 14, 15, 17, 21, 26, 36, 60]\) model an object class as a mixture of components, each trained to recognize a different aspect of appearance. For example, different viewpoints (e.g. front and back view of a car \([21, 25]\)) or articulation states (e.g. a person sitting vs standing \([52]\)). When trained on a large set, such a detector has many components \([14, 60]\), which all need to be evaluated on a test image, making it slow. Instead, we use ConF to select a subset of model components which is most relevant to a particular test image. We then run only those components, obtaining a speed-up (2x for DPM \([21]\) and 10x for EE-SVM \([36]\), sec. 4.3). Hence, ConF makes large multi-component detectors practical. This is particularly useful for EE-SVMs, as they have as many components as there are training instances. Interestingly, in some cases we even gain a small improvement in accuracy, by not running some components that would produce false positive detections.

Moreover, we train a second ConF to predict at which positions and scales objects are likely to appear in a given test image, analogue to \([33, 45, 55]\). By incorporating this information in the detector score at test time, we reduce the false positive rate by removing detections at unlikely locations. Experiments show an mAP improvement of 2% (sec. 4.4).

Finally, we train a third ConF to predict what object classes are present in an image, as in \([1, 27, 51]\). We use it at test time by only running detectors of classes predicted to be in the test image. In experiments on a 200-class dataset \([2]\), this enables to run just 10 detectors per image on average, while also improving mAP by 5%, as it removes false positives produced by detectors of classes unlikely to be present in the image (sec. 4.5).

All these experiments demonstrate that ConF is a general technique that can predict various kinds of object properties. We carry out an extensive comparison to standard nearest-neighbour techniques for such context-based predictions \([33, 42, 45, 53, 55]\). It shows that ConF predicts object properties from global image appearance more accurately, while being much faster and memory efficient (sec. from 4.2 to sec. 4.6).

2 Related work

Context. The use of context for object detection is a broad research area. Some works \([9, 13, 21, 28, 42]\) model context as the interactions between multiple object class detectors in the same image. In this paper, we model context as the relation between global image appearance and properties of the objects within them, as in \([1, 27, 33, 39, 45, 51, 53, 59]\). These works have shown that global image descriptors give a valuable cue about which classes might be present in an image and where they are located. Many object detectors \([1,\)
employed such global context to remove out-of-context false-positive detections. A similar approach was also used for image parsing [33, 53, 59]. Most of these works have a nearest neighbour core: they first retrieve a small subset of training images which are most globally similar to a test image, and then transfer the relevant statistics of the object properties in this retrieval set to the test image. In our work instead the retrieval set is estimated by ConF, which is explicitly trained to return images containing objects with similar properties to those in the test image. ConF has several advantages over nearest-neighbour approaches: (i) it can learn highly complex non-linear dependencies between the global descriptor and the object property. As a result, it estimates it more accurately; (ii) in large training sets, nearest neighbour becomes very slow, as its complexity is linear in their size. ConF is much faster and more memory efficient; (iii) ConF supports any objective function, which might even be evaluated on a different data representation than the input at test time. This is a crucial feature for our problem, as we want to predict properties of objects, but based on global image features.

Multi-component detectors. These detectors [4, 14, 15, 17, 21, 26, 36, 60] model an object class as mixture of components. They can be slow when trained from large training sets as they need many components to reach peak performance [14, 60]. While we present experiments on DPM [21] and EE-SVM [36], ConF can in principle speed-up any multi-component detector. The problem of speeding-up detection, especially when evaluating many HOG templates, has also been attacked by other works [11, 20, 46, 49]. However, these are specialized to the HOG/DPM detectors, as they exploit its internal structure. Moreover, [11, 49] are multi-class methods and achieve a speed-up only when the number of classes predicted at the same time is large. ConF, instead can speed-up even a single class (sec. 4.3). Finally, we believe ConF can offer an additional, complementary way to speed-up, and could potentially be combined with these works.

EE-SVM. The EE-SVM [36] is an extreme case of multi-component detector, where a separate component is created for each training example. Because of this, EE-SVM benefit the most from dynamically selecting components with ConF. EE-SVM is widely used for applications beyond object detection [6, 11, 18, 24, 30, 37, 47, 48, 50, 54, 57] because it explicitly associates a training example to each object it detects in a test image. This enables transferring meta-data such as segmentation masks [36, 54], 3D models [36], viewpoints [6], GPS locations [24] and part-level regularization [6]. Furthermore, EE-SVM can also be used for discovering objects parts [18, 48], scene classification [30, 48], object classification [16], image parsing [54], image matching [47], automatic image annotation [57] and 3D object detection [50]. All these applications can potentially be accelerated by ConF.

3 Estimating object properties from context

We exploit the observation that global image appearance contains information about properties of the objects inside it. We focus on three properties: aspect, location, and class. We propose a new method (ConF) based on the Random Forest framework [8, 10], which learns the relation between global image features and the properties of the object in that image. Given only the global image appearance of a test image, ConF retrieves a subset of training images that contain objects with similar properties. The properties in this retrieval set can then be used to modify the behaviour of the object detector, e.g. by running only some components or by downgrading the score of false positive detections at unlikely locations. In the following, we first describe ConF in its general form (sec. 3.1), and then specialize it for each of the three properties we consider (sec. 3.2 to 3.4).
3.1 Context Forest (ConF)

Given a training set $T$ the goal of ConF is to map the global appearance $\phi(I_t)$ of a test image $I_t$ into a retrieval set $R \subset T$. We want to construct a mapping, such that properties of objects in images of $R$ are similar to the properties of objects in $I_t$ (e.g. appearance, location, class).

ConF at training time learns an ensemble of decision trees (forest) that operates on global image features $\phi(I)$. We construct each tree by recursively splitting the training set $T$ at each node. We want the leaves of the trees to contain images whose objects properties are compact according to some measure $c(T_l)$, where $T_l$ are the training images in leaf $l$. Each internal node $n$ contains a binary split function $f(\phi(I), \theta_n)$, where $\theta_n$ are its parameters. Let $T_n$ be the training images that reached node $n$, then $f(\phi(I), \theta_n)$ will split $T_n$ into two subsets $T_l$ and $T_r$. We use axis-aligned weak learners as $f$. Following the extremely randomized forest approach, for each node we randomly sample several thousand possible splits $\theta$ and choose the one that maximizes the joint compactness:

$$\theta_n = \arg \max_\theta c(T_l) + c(T_r)$$

subject to $\forall I \in T_l, f(\phi(I), \theta) = 0$, $\forall I \in T_r, f(\phi(I), \theta) = 1$

where compactness is defined as

$$c(T) = \frac{1}{N(T)^2} \frac{1}{\sigma^2 \sqrt{2\pi}} \sum_{w_i \in T} \sum_{w_j \in \{T\setminus w_i\}} e^{-\frac{1}{2} \frac{D(w_i, w_j)^2}{\sigma^2}}$$

where $N(T)$ is the number of ground-truth object bounding-boxes in $T$ and $D(w_i, w_j)$ is a distance measure between the properties of two object bounding-boxes $w_i$ and $w_j$. Note how the inner summation in (2) is an estimation of the density of the distribution induced by all bounding-boxes in $\{T\setminus w_i\}$, evaluated at $w_j$. This value is high if $w_j$ has other bounding-boxes nearby. The estimate is done with a Gaussian Kernel Density estimator (KDE). We estimate the standard deviation $\sigma$ from the entire training set once before training the forest. This determines the scale of the problem, i.e. at which range two bounding-boxes should be considered close. For each training bounding-box $w_i$ we compute its k-nearest neighbours in the whole training set and compute the standard deviation over them. Finally, we set $\sigma$ as the median of these standard deviations over all bounding-boxes.

By employing different compactness measures $c$, we can use ConF to learn relations between different object properties and global image features. Later we show how to use it for
selecting components relevant for a test image (sec. 3.2), for estimating likely object locations in it (sec. 3.3), and for predicting which object classes it is likely to contain (sec. 3.4).

**ConF at test time** operates in two phases (fig. 1, 2 and 3). First, test image $I_t$ is passed through the forest, reaching a leaf in each tree. Thereby, each tree selects the subset of training images contained in that leaf. For each training image $I_i$, we count how many trees have selected it, forming the score $\eta(I_i, I_t)$. We now construct the retrieval set $R$ to contain the $k$ most frequently selected training images. In our experiments $k = 20$.

Note how the split function $f$ and the compactness measure $c$ operate in structurally different spaces. While $f$ operates on the global image features $\phi(I)$, $c$ is measuring the similarity of properties of objects inside the images. In this way ConF learns the relation between the two. Importantly, $c$ is neither convex nor differentiable in $\phi(I)$ and we are only able to learn this inter-space relation thanks to the unique advantages of Random Forests.

### 3.2 ConF for component selection

Here we assume that each component of an object detector has been previously trained from a visually compact set of object instances, and that we know the component id $\xi_j$ of each training instance $j$. In EE-SVMs each training instance leads to a different component. In DPM the component id of a training instance can be inferred by the output of the training procedure [22]. Based on this information, we train a ConF to select a small subset of components to run on a given test image $I_t$.

**Training.** To train ConF for components selection we define the distance $D(w_i, w_j)$ in (2) as the L2 distance between the HOG descriptors of bounding-boxes $w_i$ and $w_j$.

**Test.** We pass the test image $I_t$ through ConF obtaining a retrieval set $R$. We then estimate a posterior distribution $p(\xi_j | I_t)$ over components $\xi_j$ given $I_t$. As a training image $I$ might contain multiple instances from different components, each training image is ‘labelled’ by a distribution over components $p(\xi_j | I)$. We estimate the component distribution for the test image $I_t$ as the average over the training images in the retrieval set $R$:

$$p(\xi_j | I_t) = \frac{1}{|R|} \sum_{I \in R} p(\xi_j | I)$$

Based on this distribution, we can now select which components to run on $I_t$. We rank components by their probability and iteratively pick them until their combined probability mass exceeds a threshold $\gamma$. This threshold controls a trade-off between running few components and getting high detection performance. An interesting aspect of our formulation is that the number of selected components changes depending on the test image. A test image with a characteristic appearance matching training images with a systematic recurrence of a few components will lead to a peaky $p(\xi_j | I_t)$. In this case it is safe to run only a few components and we obtain a substantial speed-up. On the other hand, if the ConF is uncertain about the contents of the test image, then the entropy of $p(\xi_j | I_t)$ will be high, and many components will be selected. In the extreme case, for a very difficult test image, our procedure naturally degenerates to the default case of running all components (assuming a high threshold $\gamma$).

### 3.3 ConF for object location

At test time, a typical detector scores windows in the test image $I_t$, based on their appearance only. We propose here to augment the detector’s scores by adding knowledge about likely positions and scales of the object class, derived purely from the global appearance of $I_t$. 
Figure 3: Illustration of ConF predicting class membership (sec. 3.4).

Training. We train two ConFs to predict object positions and scales, respectively. To do so, we employ a different measure of compactness, substituting the distance between two bounding-boxes in (2) with $D_{\text{POS}}$ (or $D_{\text{SCALE}}$). We define $D_{\text{POS}}(w_i, w_j)$ as the L2 distance between $w_i$ and $w_j$. We define $D_{\text{SCALE}}(w_i, w_j) = \max(H_i, H_j) \cdot \max(W_i, W_j)$ as the difference in their scale ($W$ and $H$ refer to width and height).

Test. We first pass the test image $I_t$ through ConF obtaining a retrieval set $\mathcal{R}$, and then compute the following score for each window $w$ in the test image:

$$\frac{1}{N(\mathcal{R})} \sum_{w_j \in \mathcal{R}} \frac{1}{\sigma^2 \sqrt{2\pi}} e^{-\frac{1}{2} \frac{D(w_i, w_j)^2}{\sigma^2}}$$

(4)

where $N(\mathcal{R})$ is the number of object instances in $\mathcal{R}$, and $D$ is either $D_{\text{POS}}$ or $D_{\text{SCALE}}$. We learn $\sigma$ from the entire training set as in sec. 3.1. This score captures how likely a window is to cover an object based on its position/scale. Finally, we linearly combine the position/scale scores with the detector’s score of $w$. The usual non-maxima suppression stage follows.

3.4 ConF for class selection

In multi-class problems, a typical system would run detectors for all classes on all test images [43]. Instead, here we use ConF to predict what classes are present in each image, and run only the corresponding detectors. This greatly reduces the number of detectors run, and removes some false-positives.

Training. To train ConF to predict what classes are present in an image, we use the following distance between two bounding-boxes in the compactness function (2):

$$D_{\text{CLASS}}(w_i, w_j) = \begin{cases} 0 & \text{if } w_i \text{ and } w_j \text{ are objects of the same class} \\ \infty & \text{otherwise} \end{cases}$$

(5)

This definition simplifies (2) considerably, as $e^{-\frac{1}{2} \frac{D(w_i, w_j)^2}{\sigma^2}}$ can only be 0 or 1. Note how the inner summation in (2) now simply counts the number of objects $w_j \in \{\mathcal{T} \setminus w_i\}$ of the same class as $w_i$. We can rewrite (2) equivalently as:

$$c(\mathcal{T}) = \frac{1}{N(\mathcal{T})^2} \frac{1}{\sigma^2 \sqrt{2\pi}} \sum_{c \in \mathcal{C}} N(\mathcal{T}, c) \cdot (N(\mathcal{T}, c) - 1)$$

(6)

where $N(\mathcal{T}, c)$ counts how many objects in $\mathcal{T}$ belong to class $c$. Note how (6) allows an important speed up over (2), as it avoids computing the KDE. Evaluating (6) takes time $O(|\mathcal{T}|)$, compared to $O(|\mathcal{T}|^2)$ for (2).
Test. We pass the test image $I_t$ through ConF obtaining a retrieval set $\mathcal{R}$. We then estimate a posterior distribution $p(c_j|I_t)$ over object classes $c_j$ given $I_t$, analog to what done for components of one class in sec. 3.2. Based on this distribution, we can now select which detectors to run on $I_t$. In our experiments we run all detectors with $p(c_j|I_t) > 0$.

4 Experiments and conclusions

We present a series of experiments on two datasets (sec. 4.1). In sec. 4.2 we first evaluate how good the retrieval sets generated by ConF are. Then, we show how ConF for component selection (sec. 4.3), object location (sec. 4.4) and class selection (sec. 4.5) can be used to improve object detection. Finally, we show how ConF is more computation and memory efficient then a nearest neighbour approach (sec. 4.6).

4.1 Datasets

We present experiments on two datasets: a 2-class dataset we call BigCH, and the 200-class val subset of ILSVRC2014 [2]. The first one has few classes, but many training instances per class (on average $\sim 21k$). We use it to evaluate ConF for selecting model components (sec. 4.3) and predicting object location (sec. 4.4). The second dataset has many classes, but much fewer training instances per class (on average $\sim 530$). We use it to evaluate ConF for class selection (sec. 4.5).

BigCH is a 2-class dataset (Car and Horse). It combines 6 existing datasets: PASCAL VOC 2012 [20], ImageNet [12], LabelMe [44], SUN 2012 [58], UIUC [3] and PASCAL-10x [60]. We collected all images with bounding-box annotations and removed duplicate images and incorrect bounding-boxes. In total, the dataset has 15766 images containing 28548 car instances and 10107 images containing 13071 horse instances. Finally, we collected negative images so that each source dataset contributes an equal number of positive and negative images. We split the dataset into training (90%) and test (10%) sets. The training set contains 25774 cars and 9097 horses, and the test set contains 2774 cars and 1010 horses.

ILSVRC2014 [2] is a 200-class dataset annotated by bounding-boxes. Following [23], we perform experiments on the val set and consider two disjoint subsets: val1 (10k images) and val2 (10k images). We use val1 for training and test on val2.

4.2 Quality of retrieval sets

We quantify how similar object bounding-boxes from a test image $I_t$ are to those in the retrieval set $\mathcal{R}$ returned by the method using the average density of $\mathcal{R}$ evaluated at the object properties in $I_t$

\[
\frac{1}{Z\sigma^2\sqrt{2\pi}} \sum_{w_i \in I_t} \sum_{w_j \in \mathcal{R}(I_t)} e^{-\frac{1}{2} \frac{D(w_i,w_j)^2}{\sigma^2}}
\]

where $Z$ is the number of pairs of bounding-boxes in $I_t$ and $\mathcal{R}$. The distance $D$ and standard deviation $\sigma$ vary depending on the property, as defined in sec. 3.2-3.4. We extract different features as global image descriptors $\phi(I)$, depending on the property. For the appearance and location properties, we extract SURF [7], LAB and SIFT [34] features on a dense grid at multiple scales. For each feature type and object class we train a codebook of 1000 visual words and construct a 2-level spatial pyramid [32]. Additionally, we also extract a GIST [40] descriptor for the image. Overall, we train ConF on a 16000 dimensional feature space. For the class membership property, we extract state-of-the-art convolutional neural network (CNN) descriptors of 4096 dimensions [29]. These are the output of a CNN pre-trained specifically for image classification [29] and so they are very suited to the task.
Table 1 show results averaged over the test set, higher values are better. For appearance and location, we also average results over the two classes (car and horse). As a baseline, we return the whole training set as the retrieval set. This leads to a generic prior on image properties, independent of the test image. Moreover, we compare to the traditional way of building retrieval sets by k-nearest neighbours (kNN) \([13, 14, 15, 16, 17]\), defined on the same features as ConF. Both kNN and ConF greatly outperform the baseline, proving they return meaningful retrieval sets. This confirms the observation that the global image appearance conveys information about the objects properties inside the images. Moreover, ConF returns better retrieval sets than kNN across all object properties and retrieval set sizes evaluated. In the next sections we show how more accurate retrieval sets result in better detection performance and greater speed ups.

### 4.3 ConF for automatic component selection

We now use ConF to select object detector components relevant for a given test image on the BigCH dataset. We present experiments on two detectors: DPM \([18]\) and EE-SVM \([19]\).

DPM trains a mixture of components, each on a subset of the data with compact HOG appearance \([14, 15]\). We use the publicly available implementation \([19]\) and train 16 components for the class ‘car’ and 10 components for ‘horse’ (these number of components lead to best performance on the large BigCH dataset).

The EE-SVM model is composed of a separate linear SVM for every training instance (exemplar), trained using the exemplar as the only positive against all negatives in the training set (here on HOG features). We refer to a single exemplar SVM (E-SVM) as a component of the EE-SVM model, by analogy with DPM components. We use the publicly available implementation \([19]\).

Fig. 4 shows the evolution of AP while increasing the percentage of components used (higher is better). We compare to building retrieval sets by kNN, and to a baseline which randomly selects components without looking at the test image. ConF outperforms the baseline and kNN for both object classes, for both detectors, and over the whole range of the plots. By employing ConF, we closely match the performance of the full DPM model by running roughly half of the components. We match the performance of a full EE-SVM when running
less than 10% of the components. Even in the extreme case of running just one EE-SVM component, the AP is about 90% of that of the full model. Interestingly, for EE-SVM on the horse class, ConF improves AP by 3% over the full ensemble using all components, when running $10 \times$ fewer components. There is also a minor improvement in AP for EE-SVM on the car class. This comes from dropping some components that lead to false positives.

These experiments demonstrate the ability of ConF to select relevant components given just global image appearance. This makes EE-SVMs practical even when trained from large sets with tens of thousands of exemplars. This is crucial for applications that exploits the useful ability of EE-SVM to associate training exemplars to objects detected in test images. Fig. 5 shows some example results.

### 4.4 ConF for object locations

Here we demonstrate how ConF trained to estimate the location of objects from global image features can improve detection performance by downgrading the score of false positives at unlikely locations. We experiment with DPM and EE-SVM on BigCH, as in sec. 4.3. As tab. 2 shows, ConF improves AP for both classes and both detectors (+2% for cars and +1% for horses). Instead, kNN does not bring any improvement. Fig. 6 shows example results.

### 4.5 ConF for class selection

Here we experiment on the ILSVRC2014 dataset, which has a large number of classes (200). In this scenario, we can use ConF to to predict what classes are present in an image, and then only run detectors for those classes (sec. 3.4). We use the EE-SVM detector, but this time based on object proposals [56] and state-of-the-art R-CNN features [23]. These are produced by CNN pre-trained for image classification [29, 31] and then fine-tuned for object detection [23]. For training an E-SVM, we set $C = 10^{-3}$ and we mine hard negatives from 2000 random training images (using more did not bring any improvement). Both fine-tuning and E-SVM training are done on the validation set. We measure test performance on val2, as AP averaged over the 200 classes (mAP).

Without any context, EE-SVM achieves an mAP of 16.3%. Selecting classes based on kNN retrieval sets improves performance by +3.3% (mAP 19.6%), while ConF delivers a larger improvement of +4.8% (mAP 21.1%). The improvements are due to removing false positives produced by detectors of classes unlikely to be present in the image. Interestingly, ConF selects less than 10 classes per image on average, and therefore runs $20 \times$ fewer EE-SVM detectors than the context-free baseline.

<table>
<thead>
<tr>
<th>DPM - Car</th>
<th>EE-SVM - Car</th>
</tr>
</thead>
<tbody>
<tr>
<td>None 69.1</td>
<td>NN 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DPM - Horse</th>
<th>EE-SVM - Horse</th>
</tr>
</thead>
<tbody>
<tr>
<td>None 64.6</td>
<td>NN 0</td>
</tr>
</tbody>
</table>

Table 2: Results of augmenting the detector score with the location model derived by ConF (sec. 3.3) and NN compared to not using location model at all.
Figure 6: Detection obtained before (top row) and after (bottom row) applying ConF as location model. Green bounding-boxes highlight correct detections, while red ones show false positives.

4.6 Computational and memory efficiency
ConF does not only offer better performance than kNN, but is also more memory and computationally efficient. kNN requires a number of distances computations linear in the number of training images, whereas ConF requires only a logarithmic number of thresholding operations. In practice this makes a big difference in runtime, e.g. ConF for component selection takes on average 0.5s per image vs 9.9s for kNN (on a 4-cores Intel Core i5 2.0GHz).

In terms of memory, kNN stores all feature vectors of all images in the training set. For cars in BigCH, this amounts to 1.7 GB. For each internal node ConF stores a threshold, a feature id and the ids of its children, amounting to 16 bytes. The leaves store the indices of the training images they contain, for a total of exactly the number of training images × 2 bytes overall (per tree). For cars in BigCH, there are < 900 internal nodes on average per tree. As we store 750 trees per class, the grand total is only 27 MB (60 × less than kNN).

4.7 Conclusions
We presented Context Forest (ConF), a technique for predicting properties of objects in an image based on its global appearance. We trained ConF to predict three properties: aspects of appearance, location in the image, and class membership. We presented and extensive comparison to standard nearest-neighbour techniques for such context-based predictions and results showed that ConF predicts object properties from global image appearance more accurately than kNN, while being more memory efficient and much faster.

We also used ConF to speed-up and improve object class detection (with DPM and EE-SVM). We first showed how ConF can be employed to dynamically select a small subset of model components which is most relevant for a test image. Running only the selected components led to a speed-up (2 × for DPM and 10 × for EE-SVM). Interestingly, in some cases we gained an improvement in accuracy as well, by not running the components that led to false detections. Then, we trained a second ConF to predict at which positions and scales objects are likely to appear in a given test image. By incorporating this location information in the detector score, we reduced the false positive rate by removing detections in unlikely locations and improved detection performance (AP +1-2%). Finally, we trained a third ConF to predict what classes are present in each image, and run only the corresponding detectors. This greatly reduced the number of detectors run (20 ×), and removed some false-positives, achieving an improvement of +4.8%.

Acknowledgement. We gratefully acknowledges support by SNSF fellowship PBEZP-2142889 and ERC Starting Grant VisCul.
References


