Objects as context for detecting their semantic parts

Abel Gonzalez-Garcia
a.gonzalez-garcia@sms.ed.ac.uk

Davide Modolo
davide.modolo@gmail.com

Vittorio Ferrari
vferrari@staffmail.ed.ac.uk

University of Edinburgh

Abstract

We present a semantic part detection approach that effectively leverages object information. We use the object appearance and its class as indicators of what parts to expect. We also model the expected relative location of parts inside the objects based on their appearance. We achieve this with a new network module, called OffsetNet, that efficiently predicts a variable number of part locations within a given object. Our model incorporates all these cues to detect parts in the context of their objects. This leads to considerably higher performance for the challenging task of part detection compared to using part appearance alone (+5 mAP on the PASCAL-Part dataset). We also compare to other part detection methods on both PASCAL-Part and CUB200-2011 datasets.

1. Introduction

Semantic parts play an important role in visual recognition. They offer many advantages such as lower intra-class variability than whole objects, higher robustness to pose variation, and their configuration provides useful information about the aspect of the object. For these reasons, part-based models have gained attention for tasks such as fine-grained recognition [1–10], object class detection and segmentation [11–13], or attribute prediction [14–16]. Moreover, part localizations deliver a more comprehensive image understanding, enabling reasoning about object-part interactions in semantic terms. Despite their importance, many part-based models detect semantic parts based only on their local appearance [1–5, 11, 14, 15, 17]. While some works [6–10, 12, 16] leverage other types of information, they use parts mostly as support for other tasks. Part detection is rarely their focus. Here we take part detection one step further and provide a specialized approach that exploits the unique nature of this task.

Parts are highly dependent on the objects that contain them. Hence, objects provide valuable cues to help detecting parts, creating an advantage over detecting them independently. First, the class of the object gives a firm indication of what parts should be inside it, i.e. only those belonging to that object class. For example, a dark round patch should be more confidently classified as a wheel if it is on a car, rather than on a dog (fig. 1). Furthermore, by looking at the object appearance we can determine in greater detail which parts might be present. For example, a profile view of a car suggests the presence of a car door, and the absence of the licence plate. This information comes mostly through the viewpoint of the object, but also from other factors, such as the type of object (e.g. van), or whether the object is truncated (e.g. no wheels if the lower half is missing). Second, objects also provide information about the location and shape of the parts they contain. Semantic parts appear in very distinctive locations within objects, especially given the object appearance. Moreover, they appear in characteristic relative sizes and aspect ratios. For example, wheels tend to be near the lower corners of car profile views, often in a square aspect ratio, and appear rather small.

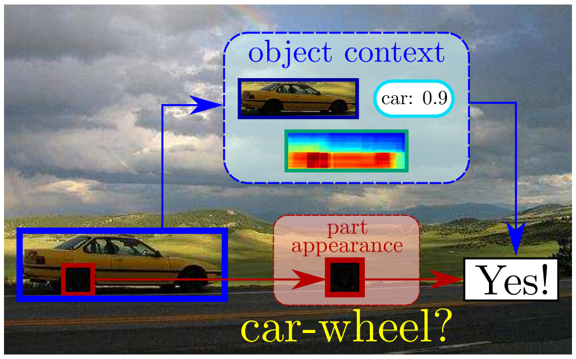


Figure 1. **Motivation for our model.** *Part appearance alone might not be sufficiently discriminative in some cases. Our model uses object context to resolve ambiguities and help part detection.*

In this work, we propose a dedicated part detection model that leverages all of the above object information. We start from a popular Convolutional Neural Network (CNN) detection model [18], which considers the appearance of local image regions only. We extend this model to incorporate object information that complements part appearance by providing context in terms of object appearance, class and the relative locations of parts within the object.

We evaluate our part detection model on all 16 object classes in the PASCAL-Part dataset [11]. We demonstrate that adding object information is greatly beneficial for the difficult task of part detection, leading to considerable performance improvements. Compared to a baseline detection model that considers only the local appearance of parts, our model achieves a +5 mAP improvement. We also compare to methods that report part localization in terms of bounding-boxes [6–8, 11, 14] on PASCAL-Part and CUB200-2011 [19]. We outperform [6, 7, 11, 14] and match the performance of [8]. We achieve this by an effective combination of the different object cues considered, demonstrating their complementarity. Moreover our approach is general as it works for a wide range of object classes: we demonstrate it on 16 classes, as opposed to 1-7 in [1, 3, 5–11, 13–15, 20–28] (only animals and person). Finally, we perform fully automatic object and part detection, without using ground-truth object locations at test time [6, 8, 11, 13]. We released code for our method at [29].

2. Related work

DPM-based part-based models. The Deformable Part Model (DPM) [30] detects objects as collections of parts, which are localized by local part appearance using HOG [31] templates. Most models based on DPM [30, 32–39] consider parts as any image patch that is discriminative for the object class. In our work instead we are interested in *semantic parts*, i.e. an object region interpretable and nameable by humans (e.g.'saddle').

Among DPM-based works, [12] is especially related as they also simultaneously detect objects and their semantic parts. Architecturally, our work is very different: [12] builds on DPM [30], whereas our model is based on modern CNNs and offers a tighter integration of part appearance and object context. Moreover, the focus of [12] is object detection, with part detection being only a by-product. They train their model to maximize object detection performance, and thus they require parts to be located only roughly near their ground-truth box. This results in inaccurate part localization at test time, as confirmed by the low part localization results reported (table 5 of [12]). Finally, they only localize those semantic parts that are discriminative for the object class. Our model, instead, is trained for precise part localization and detects all object parts.

CNN-based part-based models. In recent years, CNN-based representations are quickly replacing hand-crafted features [31, 40] in many domains, including semantic part-based models [1, 4, 5, 7, 13, 14, 17, 22, 24, 27, 41–44]. Our work is related to those that explicitly train CNN models to localize semantic parts using bounding-boxes [6–8, 14], as opposed to keypoints [1, 4] or segmentation masks [13, 17, 22, 24, 27, 43]. Many of these works [1, 4, 5, 14, 17] detect the parts used in their models based only on local part appearance, independently of their objects. Moreover, they use parts as a means for object or action and attribute recognition, they are not interested in part detection itself.

Several fine-grained recognition works [8–10] use nearest-neighbors to transfer part location annotations from training objects to test objects. They do not perform object detection, as ground-truth object bounding-boxes are used at both training and test time. Here, instead, at test time we jointly detect objects and their semantic parts.

A few works [7, 13, 28] use object information to refine part detections as a post-processing step. Part-based R-CNN [7] refines R-CNN [45] part detections by using nearest-neighbors from training samples. Our model integrates object information also within the network, which allows us to deal with several object classes simultaneously, as opposed to only one [7]. Additionally, we refine part detections with a new network module, OffsetNet, which is more efficient than nearest-neighbors. Xia et al. [28] refine person part segmentations using the estimated pose of the person. We propose a more general method that gathers information from multiple complementary object cues. The method of [13] is demonstrated only on 5 very similar classes from PASCAL-Part [11] (all quadrupeds), and on fully visible object instances from a manually selected subset of the test set (10% of the full test set). Instead, we show results on 105 parts over all 16 classes of PASCAL-Part, using the entire dataset. Moreover, [13] uses manually defined object locations at test time, whereas we detect both objects and their parts fully automatically at test time.

3. Method

We define a new detection model specialized for parts which takes into account the context provided by the objects that contain them. This is the key advantage of our model over traditional part detection approaches, which detect parts based on their local appearance alone, independently of the objects [1, 4, 14, 17]. We build on top of a baseline part detection model (sec. 3.1) and include various cues based on object class (sec, 3.2.2), object appearance (sec. 3.2.3), and the relative location of parts on the object (sec. 3.3). Finally, we combine all these cues to achieve more accurate part detections (sec. 3.4).

Model overview. Fig. 2 gives an overview of our model. First, we process the input image through a series of convolutional layers. Then, the Region of Interest (RoI) pooling layer produces feature representations from two different kind of region proposals, one for parts (red) and one for objects (blue). Each part region gets associated with a particular object region that contains it (sec. 3.2.1). Features for part regions are passed on to the part appearance branch, which contains two Fully Connected (FC) layers (sec. 3.1). Features for object regions are sent to both the object class (sec. 3.2.2) and object appearance (sec. 3.2.3) branches, with three and two FC layers, respectively.

For each part proposal, we concatenate the output of the part appearance branch with the outputs of the two object branches for its associated object proposal. We pass this refined part representation (purple) on to a part classification

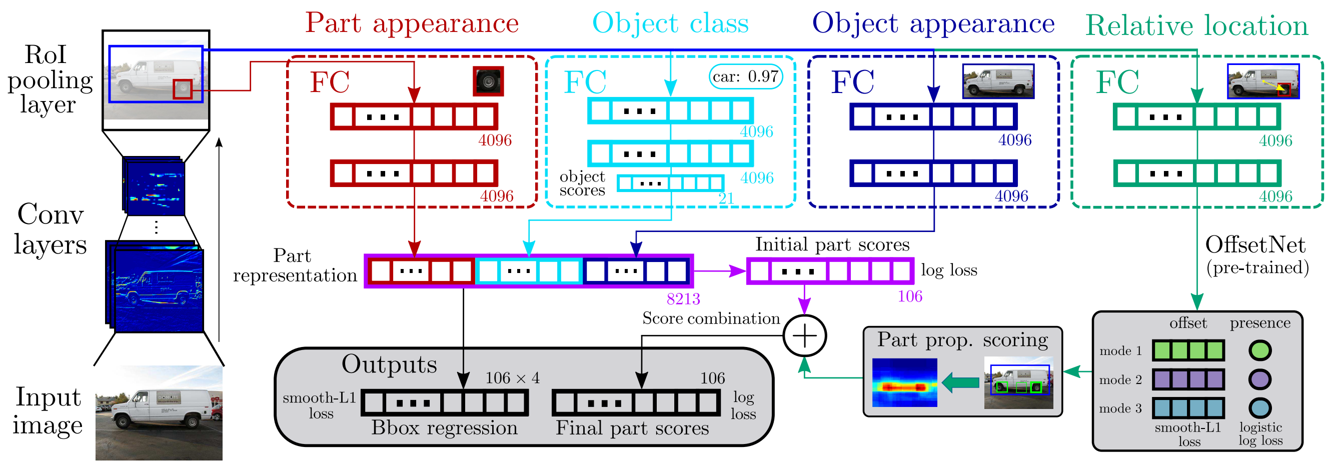


Figure 2. **Overview of our part detection model.** *The model operates on part and object region proposals, passing them through several branches, and outputs part classification scores and regressed bounding-boxes. This example depicts the relative location branch for only object class* car. *In practice, however, it processes all object classes simultaneously. When not explicitly shown, a small number next to the layer indicates its dimension for the PASCAL-Part [11] case, with a total of 20 object classes and 105 parts.*

layer and a bounding-box regression layer (sec. 3.4).

Simultaneously, the relative location branch (green) also produces classification scores for each part region based on its relative location within the object (sec. 3.3). We combine the above part classification scores with those produced by relative location (big + symbol, sec. 3.4), obtaining the final part classification scores. The model outputs these and regressed bounding-boxes.

3.1. Baseline model: part appearance only

As baseline model we use the popular Fast R-CNN [18], which was originally designed for object detection. It is based on a CNN that scores a set of region proposals [46] by processing them through several layers of different types. The first layers are convolutional and they process the whole image once. Then, the RoI pooling layer extracts features for each region proposal, which are later processed by several FC layers. The model ends with two sibling output layers, one for classifying each proposal into a part class, and one for bounding-box regression, which refines the proposal shape to match the extent of the part more precisely. The model is trained using a multi-task loss which combines these two objectives. This baseline corresponds to the part appearance branch in fig. 2.

We follow the usual approach [18] of fine-tuning for the used dataset on the current task, part detection, starting from a network pre-trained for image classification [47]. The classification layer of our baseline model has as many outputs as part classes, plus one output for a generic background class. Note how we have a single network for all part classes in the dataset, spanning across all object classes.

3.2. Adding object appearance and class

The baseline model tries to recognize parts based only on the appearance of individual region proposals. In our first extension, we include object appearance and class information by integrating it inside the network. We can see this as selecting an adequate contextual spatial support for the classification of each proposal into a part class.

3.2.1 Supporting proposal selection

Our models use two types of region proposals (sec. 4). *Part proposals* are candidate regions that might cover parts. Analogously, *object proposals* are candidates to cover objects. The baseline model uses only part proposals. In our models, instead, each part proposal p is accompanied by a *supporting object proposal* $S_{sup}(p)$, which must fulfill two requirements (fig. 3). First, it needs to contain the part proposal, i.e. at least 90% of p must be inside $S_{sup}(p)$. Second, it should tightly cover the object that contains the part, if any. For example, if the part proposal is on a wheel, the supporting proposal should be on the car that contains that wheel. To achieve this, we select the highest scored proposal among all object proposals containing p, where the score is the object classification score for any object class.

Formally, let p be a part proposal and $S(p)$ the set of object proposals that contain p. Let $\phi^k_{obj}(S_n)$ be the classification score of proposal $S_n \in S(p)$ for object class k. These scores are obtained by first passing all object proposals through three FC layers as in the object detector [18]. We select the supporting proposal $S_{sup}(p)$ for p as

$$S_{sup}(p) = \underset{S_n \in S(p)}{\operatorname{argmax}} \left[\max_{k \in \{1,\dots,K\}} \phi^k_{obj}(S_n) \right], \qquad (1)$$

where K is the total number of object classes in the dataset.

3.2.2 Object class

The class of the object provides cues about what part classes might be inside it. For example, a part proposal on a dark round patch cannot be confidently classified as a wheel based solely on its appearance (fig. 1). If the corresponding

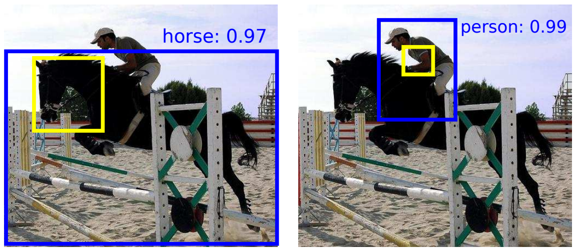


Figure 3. **Examples of supporting proposal selection.** *For each part proposal (yellow), we select as its supporting proposal (blue) the highest scored among the object proposals that contain it.*

supporting object proposal is a car, the evidence towards it being a wheel grows considerably. On the other hand, if the supporting proposal is a dog, the patch should be confidently classified as *not* a wheel.

Concretely, we process convolutional features pooled from the supporting object proposal through three FC layers (fig. 2). The third layer performs object classification and outputs scores for each object class, including a generic background class. These scores can be seen as object semantic features, which complement part appearance.

3.2.3 Object appearance

The appearance of the object might bring even more detailed information about what part classes it might contain. For example, the side view of a car indicates that we can expect to find wheels, but not a licence plate. We model object appearance by processing the convolutional features of the supporting proposal through two FC connected layers (fig. 2). This type of features have been shown to successfully capture the appearance of objects [48, 49].

3.3. Adding relative location

We now add another type of information that could be highly beneficial: the relative location of the part with respect to the object. Parts appear in very distinct and characteristic relative locations and sizes within the objects. Fig. 4a shows examples of prior relative location distributions for some part classes as heatmaps. These are produced by accumulating all part ground-truth bounding-boxes from the training set, in the normalized coordinate frame of the bounding-box of their object. Moreover, this part location distribution can be sharper if we condition it on the object appearance, especially its viewpoint. For example, the *car-wheel* distribution on profile views of cars will only have two modes (fig. 4b) instead of the three shown in fig. 4a.

Overview. Our relative location model is specific to each part class within each object class (e.g. a model for car-wheel, another model for cat-tail). Below we explain the model for one particular object and part class. Given an object proposal o of that object class, our model suggests windows for the likely position of each part inside the object. Naturally, these windows will also depend on the appearance of o. For example, given a car profile view, our model

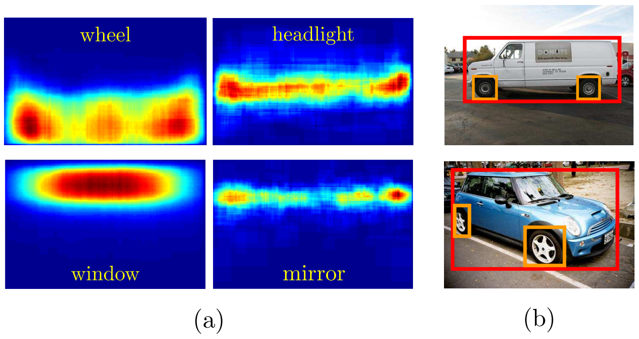


Figure 4. **Prior distributions and examples of part relative locations.** *(a) Heatmaps created using part ground-truth bounding-boxes normalized to the object bounding-box, using all* car *training samples. (b) Examples of part ground-truth bounding-boxes inside object bounding-boxes of class* car*, in different viewpoints.*

suggests square windows on the lower corners as likely to contain wheels (fig. 4b top). Instead, an oblique view of a car will also suggest a wheel towards the lower central region, as well as a more elongated aspect ratio for the wheels on the side (fig. 4b bottom). We generate the suggested windows using a special kind of CNN, which we dub *OffsetNet* (see fig. 2, Relative location branch). Finally, we score each part proposal according to its overlap with the suggested windows. This indicates the probability that a proposal belongs to a certain part class, based on its relative location within the object, and on the object appearance (but it does not depend on part appearance).

OffsetNet Model. OffsetNet directly learns to regress from the appearance of an object proposal o to the relative location of a part class within it. Concretely, it learns to produce a 4D offset vector δv that points to the part inside o. In fact, OffsetNet produces a set of vectors Δv, as some objects have multiple instances of the same part inside (e.g. cars with multiple wheels). Intuitively, a CNN is a good framework to learn this regressor, as the activation maps of the network contain localized information about the parts of the object [50–52].

OffsetNet generates each offset vector in Δv through a regression layer. To enable OffsetNet to output multiple vectors we build multiple parallel regression layers. We set the number of parallel layers to the number of modes of the prior distribution for each part class (fig. 4). For example, the prior *car-wheel* has three modes, leading to three offset regression layers in OffsetNet (fig. 2). On the other hand, OffsetNet only has one regression layer for *person-head*, as its prior distribution is unimodal.

In some cases, however, not all modes are active for a particular object instance (e.g. profile views of cars only have two active modes out of the three, fig. 4b). For this reason, each regression layer in OffsetNet has a sibling layer that predicts the presence of that mode in the input detection o, and outputs a presence score ρ. This way, even if the network outputs multiple offset vectors, only those with a high presence score will be taken into account. This construction

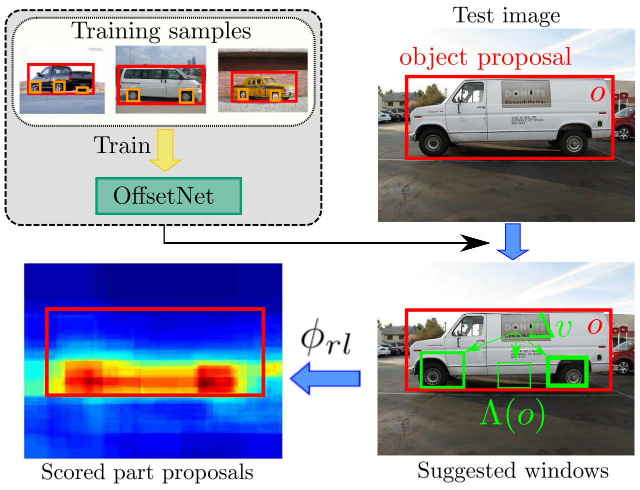


Figure 5. **Example of scoring part proposals based on relative location.** *Part class:* car-wheel. *Each car detection suggests windows likely to contain car-wheel within it. We score part proposals by computing their max IoU with any suggested window, and weighting them by their presence score and the object detection score. We show only one* car *detection for clarity.*

effectively enables OffsetNet to produce a variable number of output offset vectors, depending on the input o.

Training OffsetNet. We train one OffsetNet for all part classes simultaneously by arranging N parallel regression layers, where N is the maximum number of modes over all part classes (4 for PASCAL-Part). If a part class has fewer than N modes, we simply ignore its regression output units in the extra layers. We train the offset regression layers using a smooth-L1 loss, on the training samples described in sec. 4 (analog to the bounding-box regression of Fast R-CNN [18]). We train the presence score layer using a logistic log loss: $L(x, c) = \log(1 + e^{-cx})$, where x is the score produced by the network, and c is a binary label indicating whether the current mode is present ($c = +1$) or not ($c = -1$). We generate c using annotated ground-truth bounding-boxes (sec. 4). This loss implicitly normalizes score x using the sigmoid function. After training, we add a sigmoid layer to explicitly normalize the output presence score: $\rho = 1/(1 + e^{-x}) \in [0, 1]$.

Generating suggested windows. At test time, given an input object proposal o, OffsetNet generates M pairs $\{(\delta v_i, \rho_i)\}_{i=1}^M$ of offset vectors $\delta v_i \in \Delta v$ and presence scores ρ_i for each part class, where M is the number of modes in the prior distribution. We apply the offset vectors δv_i to o, producing a set of suggested windows $\Lambda(o) = \{o + \delta v_i\}_{i=1}^M = \{w_i\}_{i=1}^M$.

Scoring part proposals. At test time, we score all part proposals of an image by generating windows with OffsetNet for all the detected objects in the image. Let O be a set of object detections in the image, i.e. object proposals with high score after non-maxima suppression [30]. We produce these automatically using standard Fast R-CNN [18]. Let

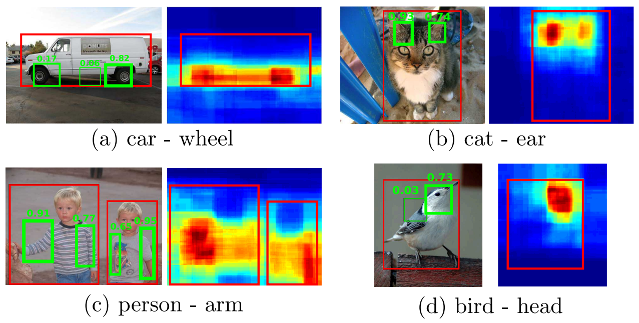


Figure 6. **OffsetNet results.** *Examples of windows suggested by OffsetNet (green) for different part classes, given the input object detections (red). We also show the heatmap generated by scoring all part proposals, showing how highly scored proposals occupy areas likely to contain part instances. The presence scores clearly indicate which suggested windows should be relied on (e.g. the second head of the bird and the middle wheel of the van have very low scores and are discounted in ϕ_{rl}).*

$\phi_{obj}(o)$ be the score of detection $o \in O$ for the considered object class. We compute the relative location score $\phi_{rl}(p)$ for part proposal p using its overlap with all windows suggested by OffsetNet

$$\phi_{rl}(p) = \max_{w_i \in \Lambda(o), o \in O} (\mathrm{IoU}(p, w_i) \cdot \rho_i \cdot \phi_{obj}(o)), \quad (2)$$

where we use Intersection-over-Union (IoU) to measure overlap. Here, $\Lambda(o) = \{w_i\}_{i=1}^M$ is the set of suggested windows output by OffsetNet for object detection o, and ρ_i is the associated presence score for each individual window w_i. Suggested windows with higher presence score have higher weight in the overall relative location score ϕ_{rl}. The score of the object detection $\phi_{obj}(o)$ is also used to weight all suggested windows based on it. Consequently, object detections with higher score provide stronger cues through higher relative location scores ϕ_{rl}. Fig. 5 depicts this scoring procedure.

Fig. 6 shows examples of windows suggested by OffsetNet, along with their presence score and a heatmap generated by scoring part proposals using eq. (2). We can see how the suggested windows cover very likely areas for part instances on the input objects, and how the presence scores are crucial to decide which windows should be relied on.

3.4. Cue combination

We have presented multiple cues that can help part detection. These cues are complementary, so our model needs to effectively combine them.

We concatenate the output of the part appearance, object class and object appearance branches and pass them on to a part classification layer that combines them and produces initial part classification scores (purple in fig. 2). Therefore, we effectively integrate object context into the network, resulting in the automatic learning of object-aware part representations. We argue that this type of context integration has greater potential than just a post-processing step [7, 13].

The relative location branch, however, is special as its features have a different nature and much lower dimensionality (4 vs 4096). To facilitate learning, instead of directly concatenating them, this branch operates independently of the others and computes its own part scores. Therefore, we linearly combine the initial part classification scores with those delivered by the relative location branch (big $+$ in fig. 2). For some part classes, the relative location might not be very indicative due to high variance in the training samples (e.g. *cat-nose*). In some other cases, relative location can be a great cue (e.g. the position of *cow-torso* is very stable across all its instances). For this reason, we learn a separate linear combination for each part class. We do this by maximizing part detection performance on the training set, using grid search on the mixing weight in the $[0, 1]$ range. We define the measure of performance in sec. 5.

4. Implementation details

Proposals. Object proposals [46, 53, 54] are designed to cover whole objects, and sometimes fail to find small parts. To alleviate this issue, we changed the standard settings of Selective Search [46], by decreasing the minimum box size to 10. This results in adequate proposals even for parts: reaching 71.4% recall with $\sim$3000 proposals (IoU $>$0.5). For objects, we keep the standard settings (minimum box size 20), resulting in $\sim$2000 proposals.

Training the part detection network. Our networks are pre-trained on ILSVRC12 [55] for image classification and fine-tuned on PASCAL-Part [11] for part detection, or on PASCAL VOC 2010 [56] for object detection, using MatConvNet [57]. Fine-tuning for object detection follows the Fast R-CNN procedure [18]. For part detection fine-tuning we changed the following settings. Positive samples for parts overlap any part ground-truth > 0.6 IoU, whereas negative samples overlap < 0.3. We train for 12 epochs with learning rate 10^{-3}, and then for 4 epochs with 10^{-4}.

We jointly train part appearance, object appearance, and object class branches for a multi-task part detection loss. We modify the RoI pooling layer to pool convolutional features from both the part proposal and the supporting object proposal. Backpropagation through this layer poses a problem, as (1) is not differentiable. We address this by backpropagating the gradients only through the area of the convolutional map covered by the object proposal selected by the argmax. We obtain the object scores used in (1) from the object class branch, which is previously initialized using the standard Fast R-CNN object detection loss, in order to provide reliable object scores when joint training starts.

Training OffsetNet. We need object samples and part samples to train OffsetNet. Our object samples are all object ground-truth bounding-boxes and object proposals with IoU $\geq$ 0.7 in the training set. Our part samples are only part ground-truth bounding-boxes. We split the horizontal axis in M regions, where M is the number of modes in the part class prior relative location distribution. We assign each part ground-truth bounding-box in the object to the closest mode. If a mode has more than one part bounding-box assigned, we pick one at random. In case a mode has no instance assigned (e.g. occluded wheel) for a particular training sample, the loss function omits the contribution of that mode. All layers except the top ones are initialized with a Fast R-CNN network trained for object detection. Similarly to the other networks, we train it for 16 epochs, but with learning rates 10^{-4} and 10^{-5}.

5. Results and conclusions

5.1. Validation of our model

Dataset. We present results on PASCAL-Part [11], which has pixel-wise part annotations for the images of PASCAL VOC 2010 [56]. For our experiments we fit a bounding-box to each part segmentation mask. We pre-process the set of part classes as follows. We discard additional information on semantic part annotations, such as 'front' or 'left' (e.g. both "car wheel front left" and "car wheel back right" become *car-wheel*). We merge continuous subdivisions of the same semantic part ("horse lower leg" and "horse upper leg" become *horse-leg*). Finally, we discard tiny parts, with average width and height over the training set $\leq$15 pixels (e.g. "bird eye"), and rare parts that appear < 10 times (e.g. "bicycle headlight"). After this pre-processing, we obtain a total of 105 part classes for 16 object classes. We train our methods on the `train` set and test them on the `val` set (the `test` set is not annotated in PASCAL-Part). We note how we are the first work to present fully automatic part detection results on the whole PASCAL-Part dataset.

Performance measure. Just before measuring performance we remove duplicate detections using non-maxima suppression [30]. We measure part detection performance using Average Precision (AP), following the PASCAL VOC protocol [56]. We consider a part detection to be correct if its IoU with a ground-truth part bounding-box is > 0.5.

Baseline results. Tab. 1 presents part detection results. As base network in the top 10 rows we use AlexNet [47] (convolutional layers on the leftmost column of fig. 2). As a baseline we train a Fast R-CNN model to directly detect parts, using only their own appearance (sec. 3.1). This achieves only 22.1 mAP (without bounding-box regression). As a reference, the same model achieves 48.5 mAP, when trained and evaluated for object class detection on PASCAL VOC 2010 [56] (same object classes as PASCAL-Part). This massive difference in performance demonstrates the inherent difficulty of the part detection task.

Adding object appearance and class. By adding object appearance (sec. 3.2.3), performance increases by +3 mAP, which is a significant improvement. Adding object class

	Model	Obj. App	Obj. Cls	Rel Loc	mAP
AlexNet	Baseline [18] (part appearance only)				22.1
	Obj. appearance	✓			25.1
	Obj. class		✓		23.0
	Obj. app + cls	✓	✓		25.7
	OffsetNet ($M = 1$)			✓	24.3
	OffsetNet			✓	24.7
	Obj. app + OffsetNet	✓		✓	26.8
	Full (Obj. app + cls + OffsetNet)	✓	✓	✓	**27.4**
	Baseline [18] (bbox-reg)				24.5
	Full (bbox-reg)	✓	✓	✓	**29.5**
VGG16	Baseline [18] (bbox-reg)				35.8
	Obj. appearance (bbox-reg)	✓			38.2
	Obj. class (bbox-reg)		✓		35.4
	Obj. app + cls (bbox-reg)	✓	✓		38.7
	OffsetNet (bbox-reg)			✓	38.5
	Obj. app + OffsetNet (bbox-reg)	✓		✓	39.4
	Full (bbox-reg)	✓	✓	✓	**40.1**

Table 1. **Part detection results on PASCAL-Part.** *The first 10 rows use AlexNet, the last 7 use VGG16. The baseline model uses only part appearance. All other models include it too.*

(sec. 3.2.2) also helps, albeit less so (+0.9 mAP). This indicates that the appearance of the object contains extra knowledge relevant for part discrimination (e.g. viewpoint), which the object class alone cannot provide. Furthermore, the combination of both types gives a small additional boost (+0.6 mAP compared to using only object appearance). Although in principle object appearance subsumes its class, having a more explicit and concise characterization of the class is beneficial for part discrimination.

Adding relative location. Our relative location model (OffsetNet, sec. 3.3) also brings improvements. We present results for two OffsetNet versions. In the first one, we fix the number of modes M to 1 for all part classes, regardless of the complexity of the prior distribution. In this simplified setting, OffsetNet already benefits part detection (+2.2 mAP over the baseline). When setting M based on the prior distribution as explained in sec. 3.3, the improvement further rises to +2.6 mAP.

Combining cues. Finally, we combine all our cues as in sec. 3.4 (always using also part appearance). First, we combine our relative location model with object appearance. This combination is beneficial and surpasses each cue alone.

Our best model (full) combines all cues and achieves +5.3 mAP over the baseline. This as a substantial improvement when considering the high difficulty of the task, as demonstrated by the rather low baseline performance. This result shows that all cues we propose are complementary to part appearance and to each other; when combined, all contribute to the final performance.

We also tested the baseline and our model using bounding-box regression (bbox-reg in tab. 1). This helps in all cases. Importantly, the improvement brought by our full model over the baseline (+5 mAP) is similar to the one without bounding-box regression (+5.3 mAP).

Partial object information. We explore here whether our method still helps when the object is only partially visible. We evaluate the baseline and our full method only on occluded objects, as indicated in the annotations of PASCAL-Part. The baseline gives 13.7 mAP, whereas our full method achieves 17.2 mAP. This demonstrates that integrating object cues benefits part detection even on partially occluded objects. One reason for this desirable behaviour is that our model learns to deal with occluded objects during training, as the training set of PASCAL-Part also includes such cases.

Comparison to	Chen et al. [11]		Fine-grained [7] [6] [8]		Gkioxari et al. [14]	
Dataset	PASCAL-Part		CUB200-2011		PASCAL VOC09	
Measure	POP	PCP	PCP		AP (0.3)	AP (0.5)
Obj GT at test	✓	✓		✓		
Theirs	44.5	70.5	66.1 [7]	74.0 [7] 85.0 [6] **94.2** [8]	38.7	17.1
Ours	**51.3**	**72.6**	**91.9**	92.7	**53.6 (65.5)**	**21.6 (44.7)**

Table 2. **Comparison to other methods.** *We compare to methods that report bounding-box part detection results, using their settings and measures.*

Example detections. Fig. 7 shows some part detection examples for both the baseline and our full model (without bounding-box regression). In general, our model localizes parts more accurately, fitting the part extent more tightly (fig. 7a,7e). Moreover, it also finds some part instances missed by the baseline (fig. 7b, 7c). Our method uses object detections automatically produced by Fast R-CNN [18]. When these are inaccurate, our model can sometimes produce worse part detections than the baseline (fig. 7f).

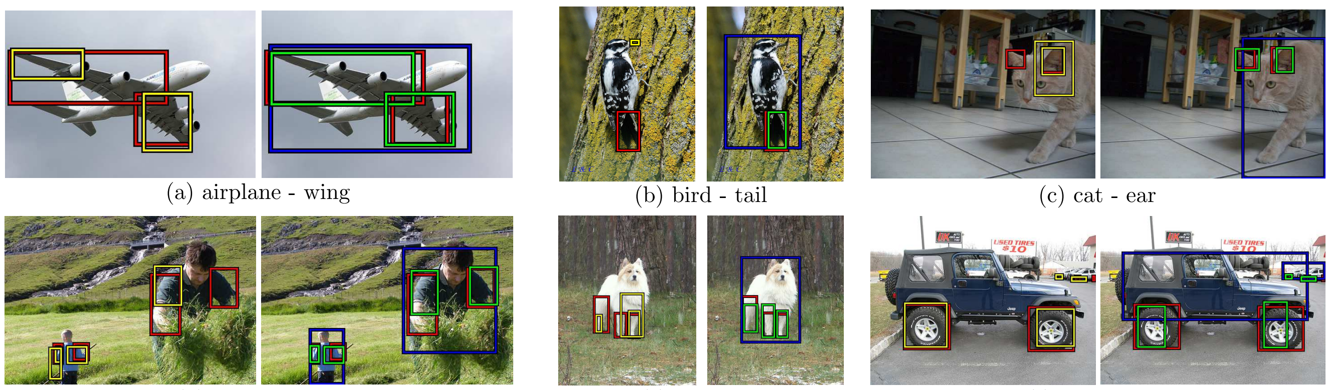
Runtime. We report runtimes on a Titan X GPU. The baseline takes 4.3s/im, our model 7.1s/im. Note how we also output object detections, which the baseline does not.

Results for VGG16. We also present results for the deeper VGG16 network [58] (last 7 rows in tab. 1). The relative performance of our model and the baseline is analog to the AlexNet case, but with higher mAP values. The baseline achieves 35.8 mAP with bounding-box regression. Our full model achieves 40.1 mAP, i.e. an improvement of magnitude comparable to the AlexNet case (+4.3 mAP).

5.2. Comparison to other part detection methods

We compare here our full (bbox-reg) model (tab. 1) to several prior works on detecting parts up to a bounding-box [7, 11, 14]. We use AlexNet, which is equivalent to the networks used in [7, 8, 14]. Tab. 2 summarizes all results.

Chen et al. [11]. We compare to [11] following their protocol (sec. 4.3.3 of [11]). They evaluate on 3 parts (head, body, and legs) of the 6 animal classes of PASCAL-Part, using Percentage of Correctly estimated Parts as measure (PCP). They also need an extra measure called Percentage of Objects with Part estimated (POP), as they compute PCP only over object instances for which their system outputs a detection for that part class. Additionally, they use ground-truth object bounding-boxes at test time. More precisely, for each ground-truth box, they retain the best overlapping object detection, and evaluate part detection only within it. As table 2 shows, we outperform [11] on PCP, and our POP is


(a) airplane - wing (b) bird - tail (c) cat - ear

(d) person - arm (e) dog - leg (f) car - wheel

Figure 7. **Qualitative results.** *Example part detections for the baseline model (yellow) and our model combining all cues (green). We also show part ground-truth bounding-boxes (red), and object detections output by our method (blue).*

substantially better, demonstrating the higher recall reached by our method. We note how [11] only report results in this easier setting, whereas we report results in a fully automatic setting without using any ground-truth at test time (sec. 5.1).

Fine-grained [6–8]. These fine-grained recognition works report part detection results on the CUB200-2011 [19] bird dataset for the head and body. They all evaluate using PCP and including object ground-truth bounding-boxes at test time. Our model outperforms [6, 7] by a large margin and is comparable to [8]. Only [7] report results without using object ground-truth at test time. In this setting, our method performs almost as well as with object ground-truth at test time, achieving a very remarkable improvement (+25.8 PCP) compared to [7]. Furthermore, we note that CUB200-2011 is an easier dataset than PASCAL-Part, with typically just one, large, fully visible bird instance per image.

Gkioxari et al. [14]. This action and attribute recognition work reports detection results on three person parts (head, torso, legs) on PASCAL VOC 2009 images (tab. 1 in [14]). As these do not have part ground-truth bounding-boxes, they construct them by grouping the keypoint annotations of [59] (sec. 3.2.2 of [14]). For an exact comparison, we train and test our full model using their keypoint-derived bounding-boxes and use their evaluation measure (AP at various IoU thresholds). We also report (in parenthesis) results using the standard part ground-truth bounding-boxes of PASCAL-Part during both training and testing (as PASCAL VOC 2009 is a subset of PASCAL-Part). We outperform [14] using their bounding-boxes, and obtain even better results using the standard bounding-boxes of PASCAL-Part. Moreover, we note how their part detectors have been trained with more expensive annotations (on average 4 keypoints per part, instead of one bounding-box).

5.3. Connection to part segmentation

Above we have considered directly detecting bounding-boxes on semantic parts. Here we explore using a part segmentation technique as an intermediate step to obtain bounding-box detections, motivated by the apparently good performance of segmentation techniques [13, 22, 24, 27, 60]. We adapt a state-of-the-art part segmentation approach [60] by fitting bounding-boxes to its output segmentations.

We train DeepLab V2 [60] with ResNet-101 [61] on all 105 parts of PASCAL-Part [11] `train` set, using the original pixel-wise annotations and following the training protocol of [60]. At test time, we run the trained model on the `val` set of PASCAL-Part and place a bounding-box around each connected component of the output segmentation. This method achieves 11.6 mAP, averaged over all 105 part classes, which is much lower than what obtained by our full model, e.g. 40.1 mAP with VGG16 (despite ResNet-101 performing generally better than VGG16 [61]).

This exploratory experiment shows that it is not obvious how to go from segmentations to bounding-boxes. Powerful models such as DeepLab V2 have shown good results on a pixel-level part segmentation measure for a restricted set of classes (humans and quadrupeds [60]). However, these do not necessarily translate to good performance on an instance-level bounding-box measure (mAP), and when considering a more comprehensive set of classes.

5.4. Conclusions

We presented a semantic part detection model that detects parts in the context of their objects. Our model includes several types of object information: object class and appearance as indicators of what parts lie inside, and also part relative location conditioned on the object appearance. Our model leads to a considerably better performance than detecting parts based only on their local appearance, improving by +5 mAP on the PASCAL-Part dataset. Moreover, our model outperforms several other part detection methods [6, 7, 11, 14] on PASCAL-Part and CUB200-2011.

Acknowledgements. This work was supported by the ERC Starting Grant VisCul. A. Gonzalez-Garcia also acknowledges the Spanish project TIN2016-79717-R.

References

[1] M. Simon and E. Rodner. Neural activation constellations: Unsupervised part model discovery with convolutional networks. In *ICCV*, 2015. 1, 2

[2] J. Liu, A. Kanazawa, D. Jacobs, and P. Belhumeur. Dog breed classification using part localization. In *ECCV*, 2012. 1

[3] O. M. Parkhi, A. Vedaldi, A. Zisserman, and CV Jawahar. Cats and dogs. In *CVPR*, 2012. 1, 2

[4] Shaoli Huang, Zhe Xu, Dacheng Tao, and Ya Zhang. Part-stacked cnn for fine-grained visual categorization. In *CVPR*, 2016. 1, 2

[5] T. Xiao, Y. Xu, K. Yang, J. Zhang, Y. Peng, and Z. Zhang. The application of two-level attention models in deep convolutional neural network for fine-grained image classification. In *CVPR*, 2015. 1, 2

[6] Di Lin, Xiaoyong Shen, Cewu Lu, and Jiaya Jia. Deep lac: Deep localization, alignment and classification for fine-grained recognition. In *CVPR*, 2015. 1, 2, 7, 8

[7] N. Zhang, J. Donahue, R. Girshick, and T. Darrell. Part-based r-cnns for fine-grained category detection. In *ECCV*, 2014. 1, 2, 5, 7, 8

[8] H. Zhang, T. Xu, M. Elhoseiny, X. Huang, S. Zhang, A. Elgammal, and D. Metaxas. Spda-cnn: Unifying semantic part detection and abstraction for fine-grained recognition. In *CVPR*, 2016. 1, 2, 7, 8

[9] Efstratios Gavves, Basura Fernando, C.G.M. Snoek, Arnold WM Smeulders, and Tinne Tuytelaars. Fine-grained categorization by alignments. In *ICCV*, 2013. 1, 2

[10] Christoph Goering, Erid Rodner, Alexander Freytag, and Joachim Denzler. Nonparametric part transfer for fine-grained recognition. In *CVPR*, 2014. 1, 2

[11] Xianjie Chen, Roozbeh Mottaghi, Xiaobai Liu, Sanja Fidler, Raquel Urtasun, and Alan Yuille. Detect what you can: Detecting and representing objects using holistic models and body parts. In *CVPR*, 2014. 1, 2, 3, 6, 7, 8

[12] H. Azizpour and I. Laptev. Object detection using strongly-supervised deformable part models. In *ECCV*, 2012. 1, 2

[13] Peng Wang, Xiaohui Shen, Zhe Lin, Scott Cohen, Brian Price, and Alan L Yuille. Joint object and part segmentation using deep learned potentials. In *ICCV*, pages 1573–1581, 2015. 1, 2, 5, 8

[14] G. Gkioxari, R. Girshick, and J. Malik. Actions and attributes from wholes and parts. In *ICCV*, 2015. 1, 2, 7, 8

[15] Ning Zhang, Ronan Farrell, Forrest Iandola, and Trevor Darrell. Deformable part descriptors for fine-grained recognition and attribute prediction. In *ICCV*, 2013. 1, 2

[16] Andrea Vedaldi, Siddarth Mahendran, Stavros Tsogkas, Subhrajyoti Maji, Ross Girshick, Juho Kannala, Esa Rahtu, Iasonas Kokkinos, Matthew B Blaschko, Daniel Weiss, et al. Understanding objects in detail with fine-grained attributes. In *CVPR*, 2014. 1

[17] Shuo Yang, Ping Luo, Chen-Change Loy, and Xiaoou Tang. From facial parts responses to face detection: A deep learning approach. In *ICCV*, pages 3676–3684, 2015. 1, 2

[18] R. Girshick. Fast R-CNN. In *ICCV*, 2015. 1, 3, 5, 6, 7

[19] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011. 2, 8

[20] Ning Zhang, Manohar Paluri, Marc'Aurelio Ranzato, Trevor Darrell, and Lubomir Bourdev. Panda: Pose aligned networks for deep attribute modeling. In *CVPR*, 2014. 2

[21] Jiongxin Liu, Yinxiao Li, and Peter N Belhumeur. Part-pair representation for part localization. In *ECCV*, 2014. 2

[22] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, 2015. 2, 8

[23] M. Sun and S. Savarese. Articulated part-based model for joint object detection and pose estimation. In *ICCV*, 2011. 2

[24] Xiaodan Liang, Xiaohui Shen, Jiashi Feng, Liang Lin, and Shuicheng Yan. Semantic object parsing with graph lstm. In *ECCV*, pages 125–143, 2016. 2, 8

[25] Norimichi Ukita. Articulated pose estimation with parts connectivity using discriminative local oriented contours. In *CVPR*, 2012. 2

[26] Wei Yang, Wanli Ouyang, Hongsheng Li, and Xiaogang Wang. End-to-end learning of deformable mixture of parts and deep convolutional neural networks for human pose estimation. In *CVPR*, 2016. 2

[27] Fangting Xia, Peng Wang, Liang-Chieh Chen, and Alan L Yuille. Zoom better to see clearer: Human and object parsing with hierarchical auto-zoom net. In *ECCV*, pages 648–663, 2016. 2, 8

[28] F. Xia, P. Wang, X. Chen, and A. Yuille. Joint multi-person pose estimation and semantic part segmentation. In *CVPR*, 2017. 2

[29] Parts-Obect-Context code. `http://github.com/agonzgarc/parts-object-context`, 2017. 2

[30] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Trans. on PAMI*, 32(9), 2010. 2, 5, 6

[31] N. Dalal and B. Triggs. Histogram of Oriented Gradients for human detection. In *CVPR*, 2005. 2

[32] M. Pandey and S. Lazebnik. Scene recognition and weakly supervised object localization with deformable part-based models. In *ICCV*, 2011. 2

[33] I. Endres, K. Shih, J. Jiaa, and D. Hoiem. Learning collections of part models for object recognition. In *CVPR*, 2013. 2

[34] P. Ott and M. Everingham. Shared parts for deformable part-based models. In *CVPR*, 2011. 2

[35] D.-L. Nguyen, V.-T. Nguyen, M.-T. Tran, and A. Yoshitaka. Boosting speed and accuracy in deformable part models for face image in the wild. In *ACOMP*. IEEE, 2015. 2

[36] M.A. Sadeghi and D. Forsyth. 30hz object detection with dpm v5. In *ECCV*, 2014. 2

[37] S. Divvala, A. Efros, and M. Hebert. How important are 'deformable parts' in the deformable parts model? In *ECCV*, 2012. 2

[38] B. Drayer and T. Brox. Training deformable object models for human detection based on alignment and clustering. In *ECCV*, 2014. 2

[39] M. Pedersoli, A. Vedaldi, and J. Gonzales. A coarse-to-fine approach for fast deformable object detection. In *CVPR*, 2011. 2

[40] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004. 2

[41] Adrian Bulat and Georgios Tzimiropoulos. Human pose estimation via convolutional part heatmap regression. In *ECCV*, pages 717–732. Springer, 2016. 2

[42] Xianjie Chen and Alan L Yuille. Articulated pose estimation by a graphical model with image dependent pairwise relations. In *NIPS*, pages 1736–1744, 2014. 2

[43] J. Wang and A. Yuille. Semantic part segmentation using compositional model combining shape and appearance. In *CVPR*, 2015. 2

[44] D. Modolo and V. Ferrari. Learning semantic part-based models from google images. *IEEE Trans. on PAMI*, 2017. 2

[45] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 2

[46] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. Selective search for object recognition. *IJCV*, 2013. 3, 6

[47] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 3, 6

[48] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*, 2014. 4

[49] ZongYuan Ge, Christopher McCool, Conrad Sanderson, and Peter Corke. Subset feature learning for fine-grained category classification. In *CVPR*, pages 46–52, 2015. 4

[50] Marcel Simon, Erik Rodner, and Joachim Denzler. Part detector discovery in deep convolutional neural networks. In *ACCV*, 2014. 4

[51] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014. 4

[52] A. Gonzalez-Garcia, D. Modolo, and V. Ferrari. Do semantic parts emerge in convolutional neural networks? *IJCV*, 2017. 4

[53] B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. *IEEE Trans. on PAMI*, 2012. 6

[54] P. Dollar and C. Zitnick. Edge boxes: Locating object proposals from edges. In *ECCV*, 2014. 6

[55] Imagenet large scale visual recognition challenge (ILSVRC). http://www.image-net.org/challenges/LSVRC/2012/index, 2012. 6

[56] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes (VOC) Challenge. *IJCV*, 2010. 6

[57] A. Vedaldi and K. Lenc. Matconvnet – convolutional neural networks for MATLAB. In *ACM Multimedia*, 2015. 6

[58] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 7

[59] L. Bourdev, S. Maji, T. Brox, and J. Malik. Detecting people using mutually consistent poselet activations. In *ECCV*, 2010. 8

[60] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Trans. on PAMI*, 2017. 8

[61] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 8