OS2D: One-Stage One-Shot Object Detection by Matching Anchor Features

Anton Osokin[1], Denis Sumin[2], and Vasily Lomakin[2]

[1] National Research University Higher School of Economics, Samsung-HSE lab, Moscow, Russia
[2] mirum.io, Moscow, Russia

Abstract. In this paper, we consider the task of one-shot object detection, which consists in detecting objects defined by a single demonstration. Differently from the standard object detection, the classes of objects used for training and testing do not overlap. We build the one-stage system that performs localization and recognition jointly. We use dense correlation matching of learned local features to find correspondences, a feed-forward geometric transformation model to align features and bilinear resampling of the correlation tensor to compute the detection score of the aligned features. All the components are differentiable, which allows end-to-end training. Experimental evaluation on several challenging domains (retail products, 3D objects, buildings and logos) shows that our method can detect unseen classes (e.g., toothpaste when trained on groceries) and outperforms several baselines by a significant margin.

Keywords: one-shot detection, object detection, few-shot learning

1 Introduction

The problem of detecting and classifying objects in images is often a necessary component of automatic image analysis. Currently, the state-of-the-art approach to this task consists in training convolutional neural networks (CNNs) on large annotated datasets. Collecting and annotating such datasets is often a major cost of deploying such systems and is the bottleneck when the list of classes of interest is large or changes over time. For example, in the domain of retail products on supermarket shelves, the assortment of available products and their appearance is gradually changing (e.g., 10% of products can change each month), which makes it hard to collect and maintain a dataset. Relaxing the requirement of a large annotated training set will make the technology easier to apply.

In this paper, we consider the task of detecting objects defined by a single demonstration (one-shot object detection). In this formulation, a system at the detection (testing) stage has to "understand" what it needs to detect by a single exemplar of a target class. After the demonstration, the system should detect objects of the demonstrated target class on new images. Differently from the standard object detection systems, the classes of objects used for training and the classes of objects used for testing do not overlap. The task of one-shot detection is a step towards decreasing the complexity of collecting and maintaining data for object detection systems.

(a) classes (b) baseline (c) ours: OS2D

Fig. 1. Qualitative comparison of our OS2D model with the baseline (state-of-the-art object detection and image retrieval systems) when detecting unseen object classes. In this example, the goal is to detect objects that consist of several parts (a), which were not available in the training set. The baseline system has to finalize the object bounding box before knowing what class it is detecting (b), so as the object detector fails to merge parts, the retrieval system cannot fix the boxes and recognize correctly (c). Our model knows that the target objects consist of two parts, so it detects correctly.

The computer vision community developed a few very successful methods for regular object detection, and currently, there are well-maintained implementations in all major deep learning libraries. One-shot detection is studied substantially less with one notable exception: human face detection combined with personality recognition. A significant simplification of this setting is that a human face is a well-defined object, and one can build a detector that works well on unseen faces. Defining a general object is harder [1], so such detector suffers from insufficient generalization. Several recent works [16, 26, 12] tackled one-shot detection for general object classes (ImageNet [39] and COCO [21]). These methods build on the ideas of deep metric learning [17] and have two distinct stages: a detector of all objects (e.g., a region proposal network) and an embedding block. Karlinsky et al. [16], Michaelis et al. [26] linked the two stages by joint finetuning of network weights but, at the test stage, their models relied on class-agnostic object proposals (no dependence on the target class image). Differently, the recent work of Hsieh et al. [12] used co-attention visual cues to construct non-local feature maps that incorporate information about the target class into object proposals.

Contributions. First, we build a one-stage one-shot detector, OS2D, by bringing together the dense grid of anchor locations of the Faster R-CNN [34] and SSD [22] object detectors, the transformation model of Rocco et al. [35, 37] designed for semantic alignment and the differentiable bilinear interpolation [15]. Our approach resembles the classical pipelines based on descriptor matching and subsequent geometric verification [23, 24], but utilizes dense instead of sparse matches, learned instead of hand-crafted local features and a feed-forward geometric transformation model instead of RANSAC. The key feature of our model is that detection and recognition are performed jointly without the need to define a general object (instead, we need a good feature descriptor and transformation model). Figure 1 shows how detection and recognition can work jointly compared to the baseline consisting of separate detection and retrieval systems (see Section 6.2 for the baseline details).

Second, we design a training objective for our model that combines the ranked list loss [46] for deep metric learning and the standard robust L_1 loss

for bounding box regression [8]. Our objective also includes remapping of the recognition targets based on the output of the forward pass, which allows us to use a relatively small number of anchors.

Finally, we apply our model to several domains (retail products based on the GroZi-3.2k dataset [7], everyday 3D objects, buildings and logos based on the INSTRE dataset [44]). For the retails products, we have created a new consistent annotation of all objects in GroZi-3.2k and collect extra test sets with new classes (e.g., toothpaste, when trained on groceries). Our method outperformed several baselines by a significant margin in all settings. Code of our method and the baselines together with all collected data is available online.[3]

We proceed by reviewing the background works of Rocco et al. [35, 36, 37] in Section 2. We present our model and its training procedure in Sections 3 and 4, respectively. Next, in Section 5, we review the related works. Finally, Section 6 contains the experimental evaluation, Section 7 – the conclusion.

2 Preliminaries: matching networks

We build on the works of Rocco et al. [35, 36, 37] that targeted the problem of semantic alignment where the goal was to align the source and target images. Their methods operate on dense feature maps $\{\boldsymbol{f}_{kl}\}$, $\{\boldsymbol{g}_{kl}\}$ of the spatial size $h^{\mathcal{T}} \times w^{\mathcal{T}}$ and feature dimensionality d. The feature maps are extracted from both images with a ConvNet, e.g., VGG or ResNet. To match the features, they compute a 4D tensor $c \in \mathbb{R}^{h^{\mathcal{T}} \times w^{\mathcal{T}} \times h^{\mathcal{T}} \times w^{\mathcal{T}}}$ containing correlations $c[k,l,p,q] = \frac{\langle \boldsymbol{f}_{kl}, \boldsymbol{g}_{pq} \rangle}{\|\boldsymbol{f}_{kl}\|_2 \|\boldsymbol{g}_{pq}\|_2}$ between all pairs of feature vectors from the two feature maps.

Next, they reshape the tensor c into a 3D tensor $\tilde{c} \in \mathbb{R}^{h^{\mathcal{T}} \times w^{\mathcal{T}} \times (h^{\mathcal{T}} w^{\mathcal{T}})}$ and feed it into a ConvNet (with regular 2D convolutions) that outputs the parameters of the transformation aiming to map coordinates from the target image to the source image. Importantly, the kernel of the first convolution of such ConvNet has $h^{\mathcal{T}} w^{\mathcal{T}}$ input channels, and the overall network is designed to reduce the spatial size $h^{\mathcal{T}} \times w^{\mathcal{T}}$ to 1×1, thus providing a single vector of transformation parameters. This ConvNet will be the central element of our model, and hereinafter we will refer to it as TransformNet. Importantly, TransformNet has 3 conv2d layers with ReLU and BatchNorm [13] in-between and has receptive field of 15×15, i.e., $h^{\mathcal{T}} = w^{\mathcal{T}} = 15$ (corresponds to the image size 240×240 if using the features after the fourth block of ResNet). The full architecture is given in Appendix B.1.

TransformNet was first trained with full supervision on the level of points to be matched [35, 36] (on real images, but synthetic transformations). Later, Rocco et al. [37, Section 3.2] proposed a *soft-inlier count* to finetune TransformNet without annotated point matches, i.e., with weak supervision. The soft-inlier count is a differentiable function that evaluates how well the current transformation aligns the two images. It operates by multiplying the tensor of correlations c by the soft inlier mask, which is obtained by warping the identity mask with a spatial transformer layer [15] consisting of grid generation and bilinear resampling.

[3] https://github.com/aosokin/os2d

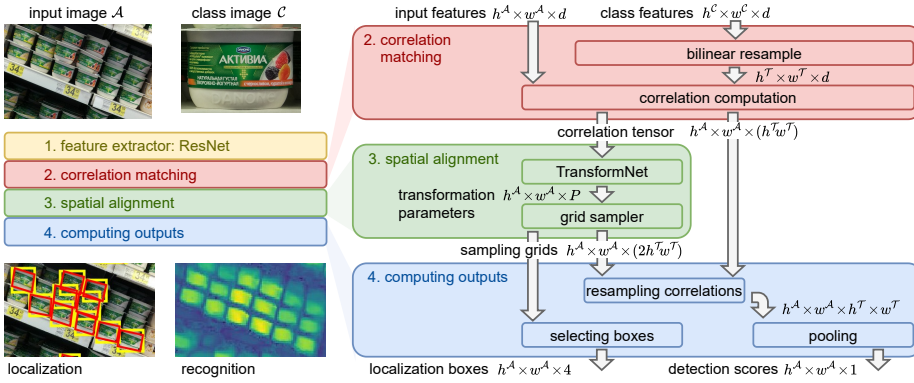

Fig. 2. (left) Inputs, outputs and main components of the OS2D model. Bottom left – the detection boxes (yellow) that correspond to the peaks in the score map, red parallelograms illustrate the corresponding affine transformations produced by TransformNet; and the detection score map (the lighter – the higher the score). **(right)** Details of the OS2D model. Boxes represent network layers, arrows – data flow (for convenience, we show the sizes of intermediate tensors). Best viewed in color.

3 The OS2D model

We now describe our OS2D model for one-shot detection. The main idea behind OS2D is to apply TransformNet to a large feature map in a fully-convolutional way and to modify the soft-inlier count to work as the detection score. We will explain how to modify all the components of the model for our setting.

The OS2D model consists of the following steps (see Figure 2): (1) extracting local features from both input and class images; (2) correlation matching of features; (3) spatially aligning features according to successful matches; (4) computing the localization bounding boxes and recognition score. We now describe these steps highlighting the technical differences to [35–37].

Feature extraction. On an input image $\mathcal{A}$ and a class image $\mathcal{C}$, a feature extractor (same as in works [35–38]) computes dense feature maps $\{\boldsymbol{f}^{\mathcal{A}}_{kl}\} \in \mathbb{R}^{h^{\mathcal{A}} \times w^{\mathcal{A}} \times d}$ and $\{\boldsymbol{f}^{\mathcal{C}}_{pq}\} \in \mathbb{R}^{h^{\mathcal{C}} \times w^{\mathcal{C}} \times d}$, respectively. The spatial sizes of the two feature maps differ and depend linearly on the sizes of the respective images.

The architecture of TransformNet requires at least one of the extracted feature maps to be of the fixed size $h^{\mathcal{T}} \times w^{\mathcal{T}}$ (this size defines the number of the input channels of the first TransformNet convolution). As the input image $\mathcal{A}$ can be of a large resolution and thus need a large feature map, we chose to convert the class feature map $\{\boldsymbol{f}^{\mathcal{C}}_{pq}\}$ to the fixed size to get $\{\boldsymbol{f}^{\mathcal{T}}_{pq}\}$. Our experiments showed that resizing class images (as in [35–37]) significantly distorts the aspect ratio of some classes, e.g., a bottle becomes a can, which degrades the quality of feature matching. We found that it worked better to resize extracted class feature maps directly by applying the differentiable bilinear resampling [15].

We also found that it is very important to extract features from the input and class images in the Siamese way, i.e., with the networks with identical parameters, which ruled out more efficient (than an image pyramid) ways of computing

multi-scale features maps, e.g., FPN [19]. We tried to untie the two branches and use FPN to extract features from the input image: such system worked well on the classes seen during training, but did not generalize to new classes.

Feature matching and alignment. Following Rocco et al. [35] we compute a 4D correlation tensor $c \in \mathbb{R}^{h^{\mathcal{A}} \times w^{\mathcal{A}} \times h^{\mathcal{T}} \times w^{\mathcal{T}}}$, $c[k,l,p,q] = \frac{\langle \boldsymbol{f}_{kl}^{\mathcal{A}}, \boldsymbol{f}_{pq}^{\mathcal{T}} \rangle}{\|\boldsymbol{f}_{kl}^{\mathcal{A}}\|_2 \|\boldsymbol{f}_{pq}^{\mathcal{T}}\|_2}$, between the input and resized class feature maps. After reshaping the tensor c to the 3D tensor $\tilde{c} \in \mathbb{R}^{h^{\mathcal{A}} \times w^{\mathcal{A}} \times (h^{\mathcal{T}} w^{\mathcal{T}})}$, we apply TransformNet in a fully-convolutional way to obtain a $h^{\mathcal{A}} \times w^{\mathcal{A}} \times P$ tensor with parameters of transformations at each location of the input feature map. All transformations are defined w.r.t. local coordinates (different for all locations), which we then convert to the global w.r.t. the input feature map coordinates (we denote the transformations by G_{kl}).

The direction of the output transformations G_{kl} is defined by training the model. The TransformNet weights released by Rocco et al. [35, 37] align the input image to the class image (according to their definition of the source and target), but we need the inverse of this to map everything to the coordinates w.r.t. the input image. Note that we cannot simply swap the two images because their feature maps are of different sizes. To use the released weights, we need to invert the direction, which, in the case of affine transformations, requires a batch inversion of 3×3 matrices. An alternative is to retrain TransformNet from scratch and to interpret its output transformations in the direction we need.

Finally, we feed all the transformations G_{kl} into a grid sampler to produce a grid of points aligning the class image at each location of the input image (we use the grids of the same size $h^{\mathcal{T}} \times w^{\mathcal{T}}$ as the TransformNet input). The output of the grid sampler is a tensor $g \in \mathbb{R}^{h^{\mathcal{A}} \times w^{\mathcal{A}} \times h^{\mathcal{T}} \times w^{\mathcal{T}} \times 2}$, $(g[k,l,p,q,0], g[k,l,p,q,1]) := G_{kl}(p,q)$ with coordinates w.r.t. the input feature map.

Recognition scores. The next step is to use the grids of matching points to extract scores $s \in \mathbb{R}^{h^{\mathcal{A}} \times w^{\mathcal{A}} \times 1}$ indicating how likely the location has a detection. The soft-inlier count [37] is a natural candidate for this score, however computing it requires enormous amount of the device memory. Specifically, one needs to create a mask of the size $h^{\mathcal{A}} \times w^{\mathcal{A}} \times h^{\mathcal{T}} \times w^{\mathcal{T}} \times h^{\mathcal{T}} \times w^{\mathcal{T}}$, which is $h^{\mathcal{T}} w^{\mathcal{T}} = 225$ times larger than any tensor we have created so far. To circumvent the problem, we use a related but different quantity, which is more efficient to compute in a fully-convolutional way. We directly resample the correlation tensor c w.r.t. the computed grids, i.e., $\hat{s}[k,l,p,q] := c[g[k,l,p,q,0], g[k,l,p,q,1], p, q]$, $\hat{s} \in \mathbb{R}^{h^{\mathcal{A}} \times w^{\mathcal{A}} \times h^{\mathcal{T}} \times w^{\mathcal{T}}}$. The values $c[y,x,p,q]$ at non-integer points (y,x) are computed by differentiable bilinear resampling [15] on the 2D array $c[:,:,p,q]$. Note that this operation is not directly supported by the standard bilinear interpolation (different channels need to be resampled at different points). One can either loop over all the channels and resample them sequentially (can be slow) or create a specialized layer. The last step to get s is to pool $\hat{s}$ w.r.t. its two last dimensions. We use the average pooling and omit the boundary of the grids to reduce effect of background matches.

Localization boxes. We extract the localization of the detections by taking max and min of grid tensor g w.r.t. its 3rd and 4th dimensions, i.e., output the tight bounding boxes around the transformed grid points.

4 Training the model

Training batches and data augmentation. In our datasets, input images are of high resolution and contain many objects (see Figure 3 for examples). We can't downsample them to a small fixed size (as typical in object detection) because of the strong distortion of the aspect ratio, and each object might simply get too few pixels. Instead, when constructing a batch we randomly choose a scale and location at which the current image will be processed and resample it to provide a training image of the target size (random crop/scale data augmentation). For each batch, we collect a set of class images (we cannot use all classes, because there are too many) by taking the annotated classes of the batch and adding some random classes as negatives to fill the class batch size.

Objective function. As in regular object detection we build the training objective from recognition and localization losses: the hinge-embedding loss with margins for recognition and the smoothed L_1 loss for localization:

$$\ell_{\text{rec}}^{\text{pos}}(s) = \max(m_{\text{pos}} - s, 0), \tag{1}$$

$$\ell_{\text{rec}}^{\text{neg}}(s) = \max(s - m_{\text{neg}}, 0), \tag{2}$$

$$\ell_{\text{loc}}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{c=1}^{4} \begin{cases} \frac{1}{2}(x_c - y_c)^2, & \text{if } |x_c - y_c| < 1, \\ |x_c - y_c| - \frac{1}{2}, & \text{otherwise.} \end{cases} \tag{3}$$

Here $s \in [-1, 1]$ is the recognition score (trained to be high for positives and low for negatives), m_{neg} and m_{pos} are the negative and positive margins, respectively, $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^4$ are the output and target encodings of bounding boxes (we use the standard encoding described, e.g., in [34, Eq. 2]).

As in detection and retrieval, our task has an inherent difficulty of non-balanced number of positives and negatives, so we need to balance summands in the objective. We started with the Faster R-CNN approach [34]: find a fixed number of hardest negatives per each positive within a batch (positive to negative ratio of 1:3). The localization loss is computed only for the positive objects. All the losses are normalized by the number of positives n_{pos} in the current batch. For recognition, we start with the contrastive loss from retrieval [27, 41, 31]: squared ℓ_{rec} and the positive margin m_{pos} set to 0.

$$\mathcal{L}_{\text{rec}}^{\text{CL}} = \frac{1}{n_{\text{pos}}} \sum_{i:t_i=1} \ell_{\text{rec}}^{\text{pos}}(s_i)^2 + \frac{1}{n_{\text{pos}}} \sum_{i:t_i=0} \ell_{\text{rec}}^{\text{neg}}(s_i)^2, \tag{4}$$

$$\mathcal{L}_{\text{loc}} = \frac{1}{n_{\text{pos}}} \sum_{i:t_i=1} \ell_{\text{loc}}(\boldsymbol{x}_i, \boldsymbol{y}_i). \tag{5}$$

Here, the index i loops over all anchor positions, $\{s_i, \boldsymbol{x}_i\}_i$ come from the network and $\{t_i, \boldsymbol{y}_i\}_i$ come from the annotation and assigned targets (see below).

We also tried the recently proposed ranked list loss (RLL) of Wang et al. [46], which builds on the ideas of Wu et al. [47] to better weight negatives. We have

$$\mathcal{L}_{\text{rec}}^{\text{RLL}} = \sum_{i:t_i=1} \frac{1}{\bar{n}_{\text{pos}}} \ell_{\text{rec}}^{\text{pos}}(s_i) + \sum_{i:t_i=0} w_i^{\text{neg}} \ell_{\text{rec}}^{\text{neg}}(s_i), \tag{6}$$

$$w_i^{\text{neg}} \propto \exp(T \ell_{\text{rec}}(s_i, 0))[\ell_{\text{rec}}^{\text{neg}}(s_i) > 0]. \tag{7}$$

Here $\tilde{n}_{\text{pos}}$ is the number of active positives, i.e., positives such that $\ell_{\text{rec}}^{\text{pos}}(s_i) > 0$. The weights w_i^{neg} are normalized in such a way that they sum to 1 over all the negatives for each image-class pair. The constant T controls how peaky the weights are. Wang et al. [46] fixed T in advance, but we found it hard to select this parameter. Instead, we chose it adaptively for each image-class pair in such a way that the weight for the negative with the highest loss is 10^3 times larger that the weights of the negatives at the margin boundary, i.e., $s_i = m_{\text{neg}}$. Finally, we did not back-propagate gradients through the weights w_i^{neg} keeping those analogous to probabilities used for sampling negatives.

Target assignment. For each position of the feature map extracted from the input image $\mathcal{A}$, we assign an anchor location w.r.t. which we decode from the output of the transformation net. At each location, as the anchor we use the rectangle corresponding to the receptive field of the transformation net. The next step is to assign targets (positive or negative) to all the anchor-class pairs and feed them into the loss functions as targets. First, we tried the standard object detection strategy, i.e., assign positive targets to the anchors with intersection over union (IoU) with a ground-truth object above 0.5 and negatives – to all anchors with IoU < 0.1. Note that we cannot force each ground-truth object to have at least one positive anchor, because we process an image in a training batch at only one scale, which might differ from the scale of the annotated objects. With the latter constraint, our set of positive anchors is sparser than typical in object detection. We tried to lower the IoU threshold for positives, but it never helped. To overcome this problem, we propose to use the standard IoU threshold only to determine localization targets (bounding boxes), and decide for classification targets *after* the output localization is computed. There, we set the high IoU thresholds of 0.8 for positives and 0.4 for negatives. We refer to this technique as *target remapping*. Target remapping is suitable for our model because we compute recognition scores at locations different from the anchors (due to the transformation model), which is not the case in regular object detection models.

5 Related works

Object detection. The standard formulation of object detection assumes a fixed list of target classes (usually 20-80) and an annotated dataset of images (preferably of a large size), where all the objects of the target classes are annotated with bounding boxes. Starting from [42, 40, 9], the state-of-the-art methods for object detection are based on neural networks. Modern detectors are of two types: two-stage and one-stage. The two-stage detectors follow Faster R-CNN [8, 34] and consist of two stages: region proposal (RPN) and detection networks. RPN generates bounding boxes that might contain objects, and the detection network classifies and refines them; a special pooling operation connects the two nets. In this approach, the RPN is, in fact, a detector of one joint *object* class. The one-stage methods like YOLO [32, 33], SSD [22], RetinaNet [20] use only RPN-like networks and are often faster but less accurate than two-stage methods. Our OS2D architecture is a one-stage method. It never computes the detections of a general object, which potentially implies better generalization to unseen classes.

One-shot object detection. Several recent works [5, 16, 26, 12] tackled the problems of one-shot and few-shot detection. Chen et al. [5] built a two-stage detector combining the elements of Faster R-CNN and SSD. To work in the low-shot (and one-shot) regime, they finetuned the model on the target classes with a particular regularization that suppressed background signal and used connections between train and test classes. Contrary to [5], our method does not need to perform any finetuning on the test classes. Michaelis et al. [26] computed the global representation of the image of the target class by spatially pooling a feature map extracted from the input image, then obtained the vector of absolute differences between this representation and each position of the feature map. These absolute differences were concatenated with the feature map and fed into the heads of both RPN and the final detector. Karlinsky et al. [16] also worked with the global representation of the target class image and built the model to take in several feature vectors coming from different exemplars of the target class. The final predictions were made by computing the L2 distances between the representations. Hsieh et al. [12] used co-attention (a non-local operation [45]) to influence the RPN object proposals (thus having non class-agnostic proposals) and co-excitation to build context dependent global representations. Our model differs from the works [5, 16, 26, 12] in the two key aspects: (1) we do not build one global representation of the image of the target classes, but work with lower-level features and match feature map to feature map instead of vector to vector; (2) our model is one-stage and never outputs non-class specific bounding boxes, which helps to generalize to unseen classes.

Image retrieval. Image retrieval aims to search a large dataset of images for the ones most relevant to a query image. A specific example of such a formulation is to search for photos of a building given one photo of the same building. Differently from one-shot detection, the output is a list of images without object bounding boxes. Most modern methods for image retrieval model relevance via distances between global image representations obtained by pooling neural network features. These representations can rely on pre-trained networks [3] or, as common recently, be learned directly for image retrieval [2, 11, 31, 46]. Differently, the work of Noh et al. [28] matches local image features similarly to OS2D. However, differently from OS2D, they do not match densely extracted features directly but subsample them with an attention mechanism, train the network not end-to-end (several steps), and the spatial verification of matches (RANSAC) is not a part of their network but a post-processing step.

Tracking in video. Visual object tracking resembles one-shot detection if we think about the marked object on the first frame as the class image and the remaining video frames as the input images. Starting from the fully-convolutional Siamese model of Bertinetto et al. [4], matching features extracted by Siamese branches with correlation has become a popular approach in tracking. Recent well-performing systems [43, 18] introduce asymmetry into the branches before computing the correlation. OS2D computes correlations with the features extracted by identical networks. We tried breaking this symmetry similarly to [18], but it severely degraded performance on the unseen classes.

Semantic alignment. The works of Rocco et al. [35, 36, 37] studied the problem of semantic alignment, where the goal is to compute correspondence between two images with objects of the same class, and the primary output of their methods (what they evaluate) is the sparse set of matching keypoints. In one-shot detection, we are interested in the score representing similarity and the bounding box around the object, which is related, but different. Our model reuses the TransformNet architecture of [35–37] (network that takes dense correlation maps as input and outputs the transformation parameters). Differently from these works, we apply this network in a fully-convolutional way, which allows us to process input images of arbitrarily large resolution. As a consequence, we need to resample correlation maps in a fully-convolutional way, which makes the existing network head inapplicable. Another line of comparison lies in the way of training the models. Our approach is similar to [37], where the transformation is trained under weak supervision, whereas Rocco et al. [35, 36] used strong supervision coming from synthetically generated transformations.

6 Experiments

Datasets and evaluation. Prior works [5, 16, 26, 12] constructed one-shot detection dataset by creating episodes from the standard object detection datasets (PASCAL [6] or COCO[21]) where the class image was randomly selected from one of the dataset images. However, we believe that this approach poses a task that requires either strong assumptions about what the target classes (inevitably limit generalization to new classes, which is our main goal) are or richer definitions of the target class (e.g., more than one shot). This happens because in the standard object detection the classes are very broad, e.g., contain both objects and their parts annotated as one class, or the objects look very different because of large viewpoint variation. In the standard setting, a large number of training images allows to define such classes, which is not possible in the one-shot setting (e.g., detect a full dog given only a head as the class image or detect the front of a car given its side view).

Having these points in mind, we chose to evaluate in domains where the one-shot setting is more natural. Specifically, we used the GroZi-3.2k dataset [7] for retail product detection as the development data. We create consistent bounding box annotations to train a detection system and collected extra test sets of retail products to evaluate on unseen classes. Additionally, we evaluate our methods on the INSTRE dataset [44] originally collected for large scale instance retrieval. The dataset is hard due to occlusions and large variations in scales and rotations [14]. The dataset has classes representing physical objects in the lab of the dataset creators and classes collected on-line (buildings, logos and common objects). We refer to the two parts of the datasets as INSTRE-S1 and INSTRE-S2, respectively. We provide the details about the dataset preparation in Appendix A.

In all experiments, we use the standard Pascal VOC metric [6], which is the mean average precision, mAP, at the intersection-over-union (IoU) threshold of 0.5. The metric uses the "difficult" flag to effectively ignore some detections.

	Config options	Training configs											
	TransformNet	V1				V2							
training	loss	CL	RLL	RLL	RLL	RLL	RLL	RLL	RLL	RLL	CL	CL	RLL
	remap targets			+	+	+	+	+	+	+		+	
	mine patches				+		+			+			
	init transform							+	+	+	+	+	+
	zero loc loss								+	+	+	+	+
	mAP	81.8	85.7	87.1¶	86.5	87.0	86.4	88.0	89.5§	**90.6**	87.4	88.1	87.1

Table 1. Validation mAP of different OS2D training configs. The V1-train and V2-train configs selected for experiments in Table 2 are marked with "¶" and "§", respectively.

6.1 Ablation study

Model architecture. The OS2D model contains two subnets with trainable parameters: TransformNet and feature extractor. For TransformNet, we use the network of the same architecture as Rocco et al. [35, 37] and plug it into our model in the two ways: (V1) simplified affine transformations with $P = 4$ (only translation and scaling) applied in the direction natural to OS2D; (V2) full affine transformations with $P = 6$ applied in another direction (used in [35, 37]). The V1 approach allows to have full supervision (the annotated bounding boxes define targets for the 4 parameters of the simplified transformations) and is more convenient to use (no need to invert transformations). The V2 approach as in [37] requires a form of weak supervision for training, but can be initialized from the weights released by Rocco et al. [35, 37], and, surprisingly is fully-functional without any training on our data at all. Comparison of V1 vs. V2 in different settings is shown in Tables 1, 2, 3, 4 (see full descriptions below). For the features extractor, we consider the third block of ResNet-50 and ResNet-101. See Table 2 for the comparison (the description is below).

Training configurations. We now evaluate multiple options of our OS2D training config (see Table 1 for the results). We experiment with the ResNet-50 feature extractor initialized from the weights trained on ImageNet [39] for image classification. For each run, we chose the best model by looking at the mAP on the validation set `val-new-cl` over the training iterations, and report this quantity as mAP.

We start with the V1 approach and training with the standard contrastive loss for recognition (4) and smoothed L1 loss for localization (5) and gradually add more options. We observed that RLL (6) outperformed the contrastive loss (4), recognition target remapping (see Section 4) helped. In addition to the performance difference, these feature regularized the training process (the initial models were quickly starting to overfit). Finally, we implemented a computationally expensive feature of mining hard patches to feed into batches (like hard negative mining of images for image retrieval [2, 31]). This step significantly slowed down training but did not improve V1 models.

Next, we switch to the V2 approach. The main benefit of V2 is to initialize TransformNet from the weights of Rocco et al. [35, 37], but training runs without

	Src task	Src data	V1-train	V2-init	V2-train
ResNet-50	—	—	59.6	2.0	67.9
	classification	ImageNet	84.8	79.6	89.0
	classification†	ImageNet	87.1	**86.1**	**89.5**
	classification†‡	ImageNet	83.2	68.2	87.1
	detection	COCO	81.9	77.5	88.1
ResNet-101	classification	ImageNet	85.6	81.1	89.4
	classification†	ImageNet	87.5	81.0	88.8
	retrieval	landmarks	**88.7**	83.3	89.0
	detection	COCO	85.6	73.1	86.8
	alignment	PASCAL	85.7	77.2	88.7

Table 2. Initializations for the OS2D feature extractor (mAP on the validation set). Symbol "†" marks the ImageNet models converted from Caffe, symbol "‡" marks the model with group norm instead of batch norm.

target remapping worsen the mAP at initialization. We also found that training signals coming to TransformNet from the localization and recognition losses were somewhat contradicting, which destabilized training. We were not able to balance the two, but found that simply zeroing out the localization loss helped (see Figure 2 bottom-left for a visualization of the learned transformations). Finally, we re-evaluated the options, which helped V1, in the context of V2 and confirmed that they were important for successful training. Mining hard patches now improved results (see Table 1).

Finally, we selected the best config for V1 as V1-train and the best config overall as V2-train (we use V2-init to refer to its initialization). The additional implementation details are presented in Appendix B.

We also separately evaluated the effect of using the inverse and full affine transformations – they did not hurt mAP, but when visualizing the results we did not see the model learning transformations richer than translation and scaling, so we omitted the architectures in-between V1 and V2.

Initialization of the feature extractor. We observed that the size of the GroZi-3.2k dataset was not sufficient to train networks from scratch, so choosing the initialization for the feature extractor was important. In Table 2, we compared initializations from different pre-trained nets. The best performance was achieved when starting from networks trained on ImageNet [39] for image classification and the Google landmarks dataset [28] for image retrieval. Surprisingly, detection initializations did not work well, possibly due to the models largely ignoring color. Another surprising finding is that the V2-init models worked reasonably well even without any training for our task. The pre-trained weights of the affine transformation model [37] appeared to be compatible not only with the feature extractor trained with them, likely due to the correlation tensors that abstract away from the specific features.

Running time. The running time of the feature extractor on the input image depends on the network size and is proportional to the input image size.

The running time of the feature extractor on the class images can be shared across the whole dataset and is negligible. The running time of the network heads is in proportional to both the input image size and the number of classes to detect, thus in the case of a large number of classes dominates the process. On GTX 1080Ti in our evaluation regime (with image pyramid) of the `val-new-cl` subset, computing input features takes 0.46s per image and the heads take 0.052s and 0.064s per image per class for V1 and V2, respectively, out of which the transformation net itself takes 0.020s. At training, we choose the number of classes such that the time on the heads matches the time of feature extraction. Training on a batch of 4 patches of size 600x600 and 15 classes takes 0.7s.

6.2 Evaluation of OS2D against baselines

Class detector. Naturally, we started with training regular detectors on the GroZi-3.2k dataset (we refer to these as *class detectors*). We used the maskrnn-benchmark system [25] to train the Faster R-CNN model with Resnet-50 and Resnet-101 backbones and the feature pyramid network to deal with multiple scales [19]. However, these systems can detect only the training classes, which is the `val-old-cl` subset.

Main baseline: object detector + retrieval. The natural baseline for one-shot detection consists in combining a regular detector of all objects merged into one class with the image retrieval system, which uses class objects as queries and the detections of the object detector as the database to search for relevant images. If both the detector and retrieval are perfect then this system solves the problem of one-shot detection. We trained object detectors with exactly same architectures as the class detectors above. The ResNet-50 and ResNet-101 versions (single class detection) delivered on the validation images (combined `val-old-cl` and `val-new-cl`) 96.42 and 96.36 mAP, respectively. For the retrieval, we used the software[4] of Radenović et al. [30, 31]. We trained the models on the training subset of GroZi-3.2k and chose hyperparameters to maximize mAP on the `val-new-cl` subset. Specifically, GeM pooling (both on top of ResNet-50 and ResNet-101) and end-to-end trainable whitening worked best (see Appendix C for details).

Sliding window baselines. To evaluate the impact of our transformation model, we use the same feature extractor as in OS2D (paired with the same image pyramid to detect at multiple scales), but omit the transformation model and match the feature map of the class image directly with the feature map of the input image in the convolutional way. In the first version (denoted as "square"), we used the feature maps resized to squares identical to the input of the transformation model (equivalent to fixing the output of the transformation model to identity). In the second version (denoted as "target AR"), we did not resize the features and used them directly from the feature extractor, which kept the aspect ratio of the class image correct.

Extra baselines. We have compared OS2D against the recently released code[5] of Hsieh et al. [12] (see C for details). We also compared against other

[4] https://github.com/filipradenovic/cnnimageretrieval-pytorch
[5] https://github.com/timy90022/One-Shot-Object-Detection

Method	Trained	val-old-cl	val-new-cl	dairy	paste-v	paste-f
Class detector	yes	87.1	—	—	—	—
Main baseline	no*	72.0	69.1	46.6	34.8	31.2
Main baseline	yes	87.6	86.8	70.0	44.3	40.0
Sliding window, square	no	57.6	58.8	33.9	8.0	7.0
Sliding window, target AR	no	72.5	71.3	63.0	65.1	45.9
Hsieh et al. [12]	yes	**91.1**	88.2	57.2	32.6	27.6
ours: OS2D V1-train	yes	89.1	88.7	70.5	61.9	48.8
ours: OS2D V2-init	no	79.7	86.1	65.4	68.2	48.4
ours: OS2D V2-train	yes	85.1	**90.7**	**71.8**	**73.3**	**54.5**

Table 3. Comparison to baselines, mAP, on the task of retail product detection. *In this version of the main baseline, the detector is still trained on the training set of GroZi-3.2k, but the retrieval system uses the weights trained on ImageNet for classification.

	ResNet-50		ResNet-101	
Method	INSTRE-S1	INSTRE-S2	INSTRE-S1	INSTRE-S2
Main baseline-train	72.2	64.4	79.0	53.9
Sliding window, AR	64.9	57.6	60.0	51.3
Hsieh et al. [12]	73.2	66.7	68.1	74.8
ours: OS2D V1-train	83.9	73.8	87.1	76.0
ours: OS2D V2-init	71.9	64.5	69.7	63.2
ours: OS2D V2-train	**88.7**	**77.7**	**88.7**	**79.5**

Table 4. Results on the INSTRE dataset, mAP.

available codes [26, 16] both with and without retraining the systems on our data but were not able to obtain more than 40 mAP on `val-new-cl`, which is very low, so we did not include these methods in the tables.

Comparison with baselines. Tables 3 and 4 show quantitative comparison of our OS2D models to the baselines on the datasets of retail products and the INSTRE datasets, respectively. Figure 3 shows qualitative comparison to the main baseline (see Suppl. Mat. D for more qualitative results). Note, that the datasets `paste-f`, INSTRE-S1 and INSTRE-S2 contain objects of all orientations. Features extracted by CNNs are not rotation invariant, so as is they do not match. To give matching-based methods a chance to work, we augment the class images with their 3 rotations (90, 180 and 270 degrees) for all the methods on these datasets.

First, note that when evaluated on the classes used for training some one-shot method outperformed the corresponding class detectors, which might be explained by having too little data to train regular detectors well enough. Second, in all the cases of classes unseen at the training time, the trained OS2D versions outperform the baselines. Qualitatively, there are at least two reasons for such

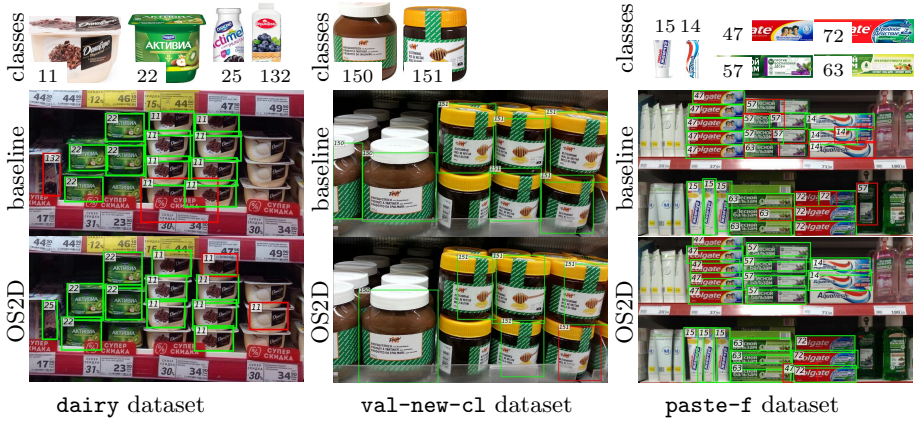

Fig. 3. Qualitative comparison of the best OS2D model vs. the best baseline. Correct detections – green boxes, incorrect – red boxes.

significant difference w.r.t. our main baseline: the object detector of the baseline has to detect objects without knowing what class is has to detect, which implies that it is very easy to confuse object with its part (see examples in Figure 1 and Figure 3-right) or merge multiple objects together; the current retrieval system is not explicitly taking the geometry of matches into account, thus it is hard to distinguish different objects that consist of similar parts. Finally, note that the sliding window baseline with the correct aspect ratio performed surprisingly well and sometimes outperformed the main baseline, but the learned transformation model always brought improvements.

One particular difficult case for the matching-based OS2D models appears to be rotating in 3D objects (see wrong detection in Figure 3-mid). The model matches the class image to only half of the object, which results in producing incorrect detection localization. Richer and maybe learnable way of producing bounding boxes from matched features is a promising direction for future work.

7 Conclusion

In this paper, we proposed the OS2D model for one-shot object detection. The model combines a deep feature extractor, correlation matching, feed-forward alignment network and bilinear interpolation in a differentiable way that allows end-to-end training. We trained our model with an objective function combining the recognition and localization losses. We applied our model to the challenging task of retail product recognition and construct a large dataset with consistent annotation for a large number of classes available (the recent SKU110k dataset [10] is of larger scale, but contains only object bounding boxes without the class labels). The OS2D model outperformed several strong baselines, which indicates the potential of the approach for practical usage.

Acknowledgments

We would like to personally thank Ignacio Rocco, Relja Arandjelović, Andrei Bursuc, Irina Saparina and Ekaterina Glazkova for amazing discussions and insightful comments, without which this project would not be possible.

This research was partly supported by Samsung Research, Samsung Electronics, by the Russian Science Foundation grant 19-71-00082 and through computational resources of HPC facilities at NRU HSE.

References

[1] Alexe, B., Deselaers, T., Ferrari, V.: What is an object? In: CVPR (2010)
[2] Arandjelović, R., Gronat, P., Torii, A., Pajdla, T., Sivic, J.: NetVLAD: CNN architecture for weakly supervised place recognition. TPAMI **40**(6), 1437–1451 (2018)
[3] Babenko, A., Slesarev, A., Chigorin, A., Lempitsky, V.: Neural codes for image retrieval. In: ECCV (2014)
[4] Bertinetto, L., Valmadre, J., Henriques, J.F., Vedaldi, A., Torr, P.H.: Fully-convolutional Siamese networks for object tracking. arXiv:1606.09549v2 (2016)
[5] Chen, H., Wang, Y., Wang, G., Qiao, Y.: LSTD: A low-shot transfer detector for object detection. In: AAAI (2018)
[6] Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The Pascal Visual Object Classes (VOC) challenge. International Journal of Computer Vision **88**(2), 303–338 (2010)
[7] George, M., Floerkemeier, C.: Recognizing products: a per-exemplar multi-label image classification approach. In: ECCV (2014)
[8] Girshick, R.: Fast R-CNN. In: ICCV (2015)
[9] Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: CVPR (2014)
[10] Goldman, E., Herzig, R., Eisenschtat, A., Ratzon, O., Levi, I., Goldberger, J., Hassner, T.: Precise detection in densely packed scenes. In: CVPR (2019)
[11] Gordo, A., Almazán, J., Revaud, J., Larlus, D.: End-to-end learning of deep visual representations for image retrieval. IJCV **12**, 237–254 (2017)
[12] Hsieh, T.I., Lo, Y.C., Chen, H.T., Liu, T.L.: One-shot object detection with co-attention and co-excitation. In: NeurIPS (2019)
[13] Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: ICML (2015)
[14] Iscen, A., Tolias, G., Avrithis, Y., Furon, T., Chum, O.: Efficient diffusion on region manifolds: Recovering small objects with compact CNN representations. In: CVPR (2017)
[15] Jaderberg, M., Simonyan, K., Zisserman, A., Kavukcuoglu, K.: Spatial transformer networks. In: NIPS (2015)
[16] Karlinsky, L., Shtok, J., Harary, S., Schwartz, E., Aides, A., Feris, R., Giryes, R., Bronstein, A.: RepMet: Representative-based metric learning for classification and one-shot object detection. In: CVPR (2019)

[17] Koch, G., Zemel, R., Salakhutdinov, R.: Siamese neural networks for one-shot image recognition. In: ICML (2015)
[18] Li, B., Yan, J., Zhu, W.W.Z., Hu, X.: High performance visual tracking with Siamese region proposal network. In: CVPR (2018)
[19] Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: CVPR (2017)
[20] Lin, T.Y., Goyal, P., Girshick, R., Kaiming He, P.D.: Focal loss for dense object detection. In: ICCV (2017)
[21] Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: common objects in context. In: ECCV (2014)
[22] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: SSD: Single shot multibox detector. In: ECCV (2016)
[23] Lowe, D.G.: Object recognition from local scale-invariant features. In: ICCV (1999)
[24] Lowe, D.G.: Distinctive image features from scale-invariant keypoints. IJCV **60**(2), 91–110 (2004)
[25] Massa, F., Girshick, R.: maskrcnn-benchmark: fast, modular reference implementation of instance segmentation and object detection algorithms in PyTorch. https://github.com/facebookresearch/maskrcnn-benchmark (2018), accessed: 01 March 2020
[26] Michaelis, C., Ustyuzhaninov, I., Bethge, M., Ecker, A.S.: One-shot instance segmentation. arXiv:1811.11507v1 (2018)
[27] Mobahi, H., Collobert, R., Weston, J.: Deep learning from temporal coherence in video. In: ICML (2009)
[28] Noh, H., Araujo, A., Sim, J., Weyand, T., Han, B.: Large-scale image retrieval with attentive deep local features. In: ICCV (2017)
[29] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: PyTorch: An imperative style, high-performance deep learning library. In: NeuIPS (2019)
[30] Radenović, F., Tolias, G., Chum, O.: CNN image retrieval learns from BoW: Unsupervised fine-tuning with hard examples. In: ECCV (2016)
[31] Radenović, F., Tolias, G., Chum, O.: Fine-tuning CNN image retrieval with no human annotation. TPAMI (2018)
[32] Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You Only Look Once: Unified, real-time object detection. In: CVPR (2016)
[33] Redmon, J., Farhadi, A.: YOLO9000: Better, faster, stronger. In: CVPR (2017)
[34] Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks. TPAMI **39**(6), 1137–1149 (2017)
[35] Rocco, I., Arandjelović, R., Sivic, J.: Convolutional neural network architecture for geometric matching. In: CVPR (2017)

[36] Rocco, I., Arandjelović, R., Sivic, J.: Convolutional neural network architecture for geometric matching. TPAMI (2018)
[37] Rocco, I., Arandjelović, R., Sivic, J.: End-to-end weakly-supervised semantic alignment. In: CVPR (2018)
[38] Rocco, I., Cimpoi, M., Arandjelović, R., Torii, A., Pajdla, T., Sivic, J.: Neighbourhood consensus networks. In: NeurIPS (2018)
[39] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet large scale visual recognition challenge. IJCV **115**(3), 211–252 (2015)
[40] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., LeCun, Y.: Overfeat: Integrated recognition, localization and detection using convolutional networks. In: ICLR (2014)
[41] Simo-Serra, E., Trulls, E., Ferraz, L., Kokkinos, I., Fua, P., Moreno-Noguer, F.: Discriminative learning of deep convolutional feature point descriptors. In: ICCV (2015)
[42] Szegedy, C., Toshev, A., Erhan, D.: Deep neural networks for object detection. In: NIPS (2013)
[43] Valmadre, J., Bertinetto, L., Henriques, J.F., Vedaldi, A., Torr, P.H.: End-to-end representation learning for correlation filter based tracking. In: CVPR (2017)
[44] Wang, S., Jiang, S.: INSTRE: a new benchmark for instance-level object retrieval and recognition. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) **11**(3), 37 (2015)
[45] Wang, X., Girshick, R., Gupta, A., He, K.: Non-local neural networks. In: CVPR (2018)
[46] Wang, X., Hua, Y., Kodirov, E., Hu, G., Garnier, R., Robertson, N.M.: Ranked list loss for deep metric learning. In: CVPR (2019)
[47] Wu, C.Y., Manmatha, R., Smola, A.J., Krähenbühl, P.: Sampling matters in deep embedding learning. In: ICCV (2017)

A Data and evaluation

A.1 Retail products

Data. We used the GroZi-3.2k dataset [7] for retail product detection (680 images collected from 5 different stores) as development data. We could not use the original annotation of this dataset because it was often grouping multiple similar objects into a single bounding box and had no strict policy about what objects are of the same class and what are of different classes. Thus, we created new annotation where each object has an individual bounding box, and only identical objects belong to the same class. For each class, we selected a template class image either from the original annotation (if available) or from the internet by querying the product names. Importantly, each object was assigned the "difficult" flag when the human annotator could not assign a class without guessing. We ended up having 8921 objects of 1063 classes annotated.

Data splits. We created the splits of the dataset by selecting 185 classes and keeping the images with those classes away from training: 622 objects in 84 images. We will refer to this set as `val-new-cl`. The selected images also contained 518 objects of classes that appeared at training. We will refer to this set as `val-old-cl`. Throughout our experiments we used `val-new-cl` as our main validation set, and `val-old-cl` as a secondary validation showing performance on the training classes.

Extra test sets. To assess the generalization ability of our method, we collected extra test sets containing objects of new classes in the images taken in different conditions. The `dairy` set consists of 12 images with 786 objects of 166 classes of dairy products. The `paste-f` set consists of 91 images with 4861 objects of 259 classes of toothpaste and accompanying products. However, the `paste-f` set contains objects of all orientations, which is different from the training conditions. We also selected a subset `paste-v`, where all the objects with incorrect orientation are masked out with the "difficult" flag.

Table 3 of the main paper contains results of the OS2D methods and baselines on the test and validation subsets.

A.2 Additional dataset: INSTRE

In addition to the domain of retail products, we applied our methods to the INSTRE dataset [44], which was originally collected for large scale instance retrieval and has bounding box annotation for all objects. The dataset is considered hard due to occlusions and large variations in scales and rotations [14]. The dataset has 28,543 images and 200 object classes: 100 classes representing physical objects in the lab of the dataset creators, 100 classes collected on-line representing buildings, logos and common objects. We refer to the classes of the first and second types as INSTRE-S1 and INSTRE-S2, respectively.

The INSTRE dataset is used for evaluating retrieval systems, so does not have splits into train and test. We modify the evaluation protocol of Iscen et al. [14] who selected 5 images of each class as queries (correspond to class images in

our terminology) by splitting the classes (with the corresponding images) of both INSTRE-S1 and INSTRE-S2 in the following proportion: 75% for training, 5% for validation and 20% for testing. In this setting, we can train and evaluate our method and the baselines. Table 4 of the main paper provides results on the two versions on the dataset. We report the results on the test sets of INSTRE-S1 and INSTRE-S2 after training of the training subsets of INSTRE-S1 and INSTRE-S2, respectively.

A.3 Evaluation metric.

In our experiments, we've used the standard Pascal VOC metric [6], which is the mean average precision at the intersection-over-union (IoU) threshold of 0.5 (we will refer to it as mAP). Importantly, this metric also supports the "difficult" flag of the annotation: it is used to exclude ground-truth objects and their detections when computing both recall and precision, which means that the method is neither penalized nor rewarded for detecting objects with this flag.

B Implementation details

B.1 TransformNet architecture

We follow Rocco et al. [35, 36, 37] and use the same architecture of TransformNet: ReLU, channelwise L2-normalization, conv2d with the kernel $7 \times 7 \times 225 \times 128$, batch norm, ReLU, conv2d with the kernel $5 \times 5 \times 128 \times 64$, batch norm, ReLU, conv2d with the kernel $5 \times 5 \times 64 \times P$. Here P is the number of parameters of the transformation, which equals 6 for the affine transformation. This network was designed for aligning two feature maps of size 15×15, i.e., $h^{\mathcal{T}} = w^{\mathcal{T}} = 15$ (corresponds to the image size 240×240 if using the features after the fourth block of ResNet).

Note that the network starts with ReLU, which corresponds to taking only positive correlations when building transformations (Rocco et al. [35] did not include this layer into TransformNet but applied it right after computing the correlations).

B.2 OS2D details

Implementation and hardware. We implemented the OS2D model based on the PyTorch library [29]. The models were both trained and tested on GPUs. The hyperparameters for training were selected to fit the process on Nvidia GTX 1080 Ti. However evaluation of retail test sets required more device memory because of higher resolution, small objects and a large quantity of classes. We used Nvidia V100 devices for such runs.

Training. The OS2D models were trained with the SGD optimizer for 200k steps with the learning rate of 10^{-4}, weight decay of 10^{-4} and momentum of 0.9. We decreased the learning rate by a factor of 10 after 100k and 150k training

iterations. We used the input image batch size of 4, cropped patches of size 600×600 and used at most 15 different labels per batch. Note that cropping patches of the correct size is effectively a version of random crop/scale data augmentation. We tried using more types of data augmentation, but none of them was effective.

When training all the models, we converted the switched layers of the feature extractor to the evaluation mode, i.e., did not estimate batch mean and variance. Keeping batchnorm in the training mode significantly degraded the performance. When training the V1 and V2 models we kept batchnorm of the transformation network in the training and evaluation modes, respectively.

We followed Rocco et al. [35, 36, 37] and trained TransformNet on only positive pairs. Technically we achieved this by computing two versions of the transformations at training – one with the full computational graph, another with the TransformNet parameters detached from the graph. The first version was used to train on positives, the second one – to train on negatives. We used this approach because when training transformations on negatives the networks started to ruin the transformation model by moving the transformation in random directions. On top of that, we often have very similar classes, and we still want them to be aligned properly to better compare the matched features.

When training all V1 models we initialized TransformNet to always output identity transformation by setting the weights of the last convolutional layer of TransformNet to 0 and biases to 0 or 1.

For the objective function, we use the margins $m_{\text{pos}} = 0.3$ and $m_{\text{neg}} = 0.25$ when the recognition scores were normalized to the segment $[0, 1]$. To train the V1 models, we used the weight of the localization loss of 0.2. To fine-tune the V2 models, we turned the localization loss completely, i.e., set its weight to zero.

Detection. Before computing the final results, for all the methods we used the standard non-maximum suppression (NMS) with the IoU threshold of 0.3. Differently from the maskrcnn-benchmark [25], we did not do joint NMS for all the classes – it always degraded the performance.

At evaluation, we resized the class images with preserving their aspect ratio to have their product of dimensions equal to 240^2. For the input images, we used the image pyramid to deal with objects of different scales. We always use the pyramid of 7 levels: 0.5, 0.625, 0.8, 1, 1.2, 1.4, 1.6 times the dataset scale. For each dataset, we estimated its scale by computing and rounding the average object size. The GroZi-3.2k dataset was of scale 1280, the `dairy` dataset was of scale 3500, the `paste-v` dataset were also of scale 3500. However, the `paste-v` dataset had too many labels, so the largest image size did not fit into the GPU memory, thus we reduced its scale to 2000 for all experiments with OS2D (the baselines were still run on the initial scale). Evaluation of an image at a particular scale, e.g., 0.5 * 3500 = 1750, means that we resize the input image such that its largest size equals the scale, e.g., 1750, before feeding it into the feature extractor (or objects detector for the baselines).

C Details of the baselines

In this section, we describe the details of the baselines that were important to improve their performance. Note that implementations of both baselines use open-source code, and we provide all the changes and launching scripts together with the OS2D code.[6]

C.1 Main baseline: detector + retrieval

For the detector, we used the maskrnn-benchmark system [25]. We used the Faster R-CNN detectors [34] with the feature pyramid backbone [19] based on ResNet-50 and ResNet-101. We used the standard hyperparameters, but added multi-scale training and testing (supported by the library), which were improving results. The scales of images for both training and testing were set the same to the OS2D training regime.

For the retrieval system used on top of the detections, we used the open-source library by Radenović et al. [30, 31]. We used the trainable Generalized-Mean (GeM) pooling and end-to-end trainable whitening layers. For the training dataset, we used the class images as queries, annotated detections of the correct/incorrect classes as positives and negatives, respectively. We also randomly sampled 10 bounding boxes per training image and automatically labeled them as positive/negatives based on their IoU with annotated objects. In the training process, we used the standard setting with the contrastive loss, hard negative mining, Adam optimizer and learning rate schedule with an exponential decay. We resized all images (queries, positives and negatives) to have the maximal side equal to 240 (with preserving the aspect ratio).

At the testing stage, we used the same image pyramid as in OS2D for the detector and the multi-scale descriptor (3 scales) for retrieval.

C.2 CoAE one-shot detector

We compared our methods with the official implementation of the recent CoAE method of Hsieh et al. [12]. Their released models (trained on ImageNet) did not generalize well to our settings, so we reported only the results of the retrained models. For fair comparison with OS2D, we added multi-scale training and testing to the original code. Multi-scale training helped significantly, while multi-scale testing did not help at all. In training, we used the same number of iterations as for OS2D (the process converged well) and the same learning rate schedule (but different initial value).

D Additional qualitative results

In Figures 4, 5, 6, we present extra detection results, provided by an OS2D model. For the purposes of visualization, we've run these results through NMS over all classes. The detection threshold was set a bit lower, so one can also see highest scoring wrong detections.

[6] https://github.com/aosokin/os2d

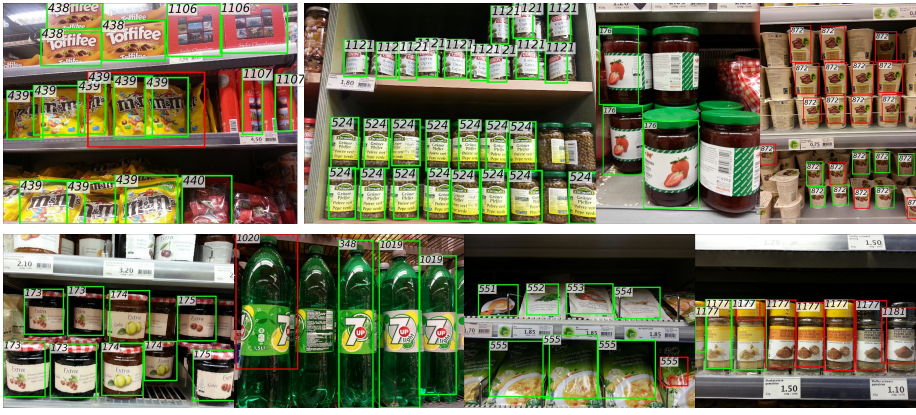

Fig. 4. Detection results on the `val-new-cl` subset of the GroZi-3.2k dataset.

Fig. 5. Detection results on the `dairy` test set.

Fig. 6. Detection results on the `paste-f` test set.