Fine-grained recognition of thousands of object categories with single-example training

Leonid Karlinsky[*], Joseph Shtok[*], Yochay Tzur, Asaf Tzadok
IBM Research
{leonidka,josephs,yochayt,asaf}@il.ibm.com

Abstract

We approach the problem of fast detection and recognition of a large number (thousands) of object categories while training on a very limited amount of examples, usually one per category. Examples of this task include: (i) detection of retail products, where we have only one studio image of each product available for training; (ii) detection of brand logos; and (iii) detection of 3D objects and their respective poses within a static 2D image, where only a sparse subset of (partial) object views is available for training, with a single example for each view. Building a detector based on so few examples presents a significant challenge for the current top-performing (deep) learning based techniques, which require large amounts of data to train. Our approach for this task is based on a non-parametric probabilistic model for initial detection, CNN-based refinement and temporal integration where applicable. We successfully demonstrate its usefulness in a variety of experiments on both existing and our own benchmarks achieving state-of-the-art performance.

1. Introduction

Most of the current top-performing learning-based approaches, especially deep learning methods, rely on large amounts of annotated data for training. This poses a significant challenge in situations when we need to recognize many thousands of visually similar (fine-grained) categories for which only a few examples are available. This situation frequently arises in recognition of retail product categories which are inherently fine-grained, and where we usually have just a single studio-image example of a product to train on. Another example is the detection of the camera pose of a query image with respect to a large scene modeled by a 3D point-cloud. Here we train on a sparsely sampled set of partial views of the modeled scene, and the task is to detect unseen views. Yet another example is the detection of brand logos in community photos.

[*]authors contributed equally to this work

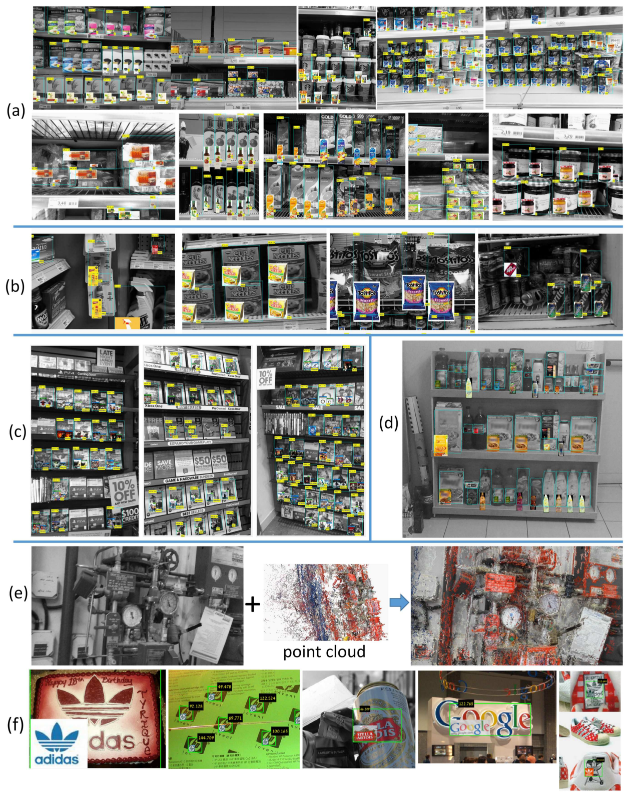

Figure 1. Qualitative examples of the performance of the proposed approach on various datasets (Section 4): (a)GroZi-3.2K, (b) GroZi-120, (c) GameStop, (d) Retail121, (e) PCPE, (f) FlickrLogos-32. The images are shown in greyscale, the cyan detection boxes are annotated with the detected category images in the bottom left corner. Best viewed in color.

In this paper we propose an approach for the scenario of fine-grained object detection and recognition with limited training data and large-scale datasets. Designed to work for both image and video inputs, the method consists of three main components: a fast initial detection and classification algorithm, Deep Neural Network (DNN) based fine-grained category refinement, and temporal integration for video inputs. In the following, we summarize our main contributions and discuss the related work.

1.1. Main Contributions

- A non-parametric probabilistic model for multi-scale, fine-grained multi-class detection and recognition, capable of working in the challenging one-training-example per class setup. Accompanied by a sequential three-step inference approach to cope with a very large space of possible unobserved variable assignments: task specific objectness $\rightarrow$ per hypothesis class short list prediction $\rightarrow$ per hypothesis structured prediction and refinement.
- A fast Nearest Neighbor (NN) search technique capable of searching for hundreds of thousands of image patch descriptors within a set of millions, in less than a second per mega-pixel.
- A DNN combining the detected object images and their associated classification results (obtained using the probabilistic model) in order to produce fine-grained classification refinement. The network is trained on synthetic data.
- Experimental results validating the usefulness of the proposed approach by showing promising results on six datasets of retail product detection, logo detection, and single-image camera pose estimation for 3D point clouds.

1.2. Related work

In recent studies on object recognition and classification, much attention has been given to natural object categories, with substantial intra-class variations. Typically, the number of categories is up to a thousand, and presence of rich training data is usually assumed. In this domain, currently dominated by deep learning based methods, notable recent works include [32, 3, 26, 25]. In [3] a CNN is used to classify region proposals provided by an external detector; later in [26], both the region proposal and the classification are produced by a single end-to-end network. This Region-CNN (RCNN) paradigm exhibits state-of-the-art performance on PASCAL VOC 2012 dataset.

While deep learning is a very powerful and useful tool, it requires a massive training set with labeled examples that are sufficiently similar to the test images. Training with a single example per category poses a challenge for these methods. The same claim also holds for the majority of more classical general purpose object detectors [5, 17]. A plausible solution is to synthesize training data from the few available examples. We use this method to train a network for the Phase 2 of our approach. However, training data synthesis alone does not seem to be sufficient for successful training, as illustrated in the Results section by the lower performance of the FRCNN detector [26] (kindly provided by the authors) trained on the same synthetic data we use for our Phase 2.

Addressing the limitation of a small training set, there are a number of works focusing on one-shot learning for object recognition. In the seminal paper [18], each object category is represented using a probabilistic Constellation model. Model parameters are first learned for a generic category on sufficiently large training data; then models for individual categories are obtained by tuning the parameters using the few available examples. The method was tested on the Caltech101 dataset with 101 roughly cropped (one object per image), visually distinct categories. Additional works [21, 16, 37] explore the problem of one-shot learning for character recognition.

An interesting work in [8] deals with recognition of 3D retail products from arbitrary angles, training on one studio image per product. The method of [8] is to pre-train a DNN on an auxiliary dataset with a richer data of 3D views, and then fine-tune it on the single-image-per-class for the targeted products. Their method is tested on 300 objects in the RGB-D dataset, displaying its usefulness in teaching the network to generalize to novel views.

In the first phase of our approach, we use a non-parametric probabilistic model for the initial detection and classification. We compute the various probabilities using a variant of efficient sub-linear nearest neighbor (NN) search [22, 35], in the spirit of the seminal work [1]. The model itself is similar to the non-parametric star model used as a single scale hand detector in [12], but in our case it is modified by adding unobserved variables to support efficient multi-scale and multi-class fine-grained detection and recognition. In addition, we propose a sequential inference procedure that accommodates for the extremely large search space of possible assignments to the unobserved variables of the model.

A number of works deal with fine-grained recognition, both using classical [40] and deep [41, 34] methods. To disambiguate similar categories [40, 41] detect object parts and decide on the category labels based on rectified images of these parts. [34] trains a deep model using triplet loss. We use a DNN classifier for fine-grained refinement in the second Phase. During training, its learning capacity is focused on discriminating between the strongest candidates from the first Phase detector, which often represent similar object categories.

We consider data with little intra-class variation (e.g. a class is represented by a specific retail product). This is related to instance recognition methods, based on template matching techniques, such as [31]. Leading such methods [13, 24, 11] were tested and outperformed in [7], which was used as a baseline in this work. Image retrieval works [23, 29] typically assume that the object is either the entire image or appears roughly cropped, whereas in our case multiple potentially small objects may appear in a large image.

As mentioned above, we use data synthesis to train our DNN for Phase 2. Common methods for augmentation of training data produce new images using translations, mirror reflections, and adding Gaussian noise [15]. We go a step further, using random homographies to generate new views and learned linear filters to degrade the high-quality training studio images for a more realistic appearance.

The literature for detecting and recognizing retail products in images [33, 7, 20, 36] is rather limited compared to general purpose techniques. The closest work to our setting is [7], which demonstrated promising results on the challenging task of detecting 3235 grocery products in still images. A short-list of possible categories is first estimated using random forests, applied on a regular grid of subwindows. The training images of the found categories are then matched, one by one, to localize and recognize the products. In contrast, in the proposed approach we first localize the objects regardless of their class, and only then perform classification by re-using (for efficiency) the computations from the first step. An earlier work comparing the performance of different descriptors on the dataset containing 120 products and 28 in-store test videos is presented in [20]. Despite the listed efforts and industrial advances, the problem of robust and scalable retail products recognition in unconstrained realistic conditions still remains largely open.

2. Method

We organized our approach in three main phases: (i) fast detection & recognition; (ii) deep (fine grained) classification refinement; (iii) temporal integration & tracking based refinement. In the following we describe each of those phases in detail.

2.1. Phase 1: Fast detection and recognition of thousands of categories with very limited training

[1] In this section we describe the fast detection and recognition method that we use to generate object hypotheses (bounding boxes) and classify them according to the category. As mentioned above, the method is designed to work even with just a single training example per-category. Facing the training data restriction, we adopt a classical approach to the initial detection: reasoning over patches extracted from both the query and the training images. We sample the patches densely and represent them using standard descriptors. The training patches are indexed using efficient sub-linear search data structures (e.g. LSH or kd-trees) and the query patches are searched within those structures. The Nearest Neighbors (NNs) found for the query patches are used to infer the desired output using the non-parametric probabilistic model described next. If accepted, the relatively short Matlab code for the proposed approach will be publicly released. The details and the intuition behind the inference process are given below, and are graphically illustrated in Figure 2 to underline its simplicity.

2.1.1 Training: indexing patches from all the training images

Given a set of training images $\{I_t\}$, each depicting a single object of interest (could also be Bounding Box (BB) crops from images with multiple objects), we first convert each I_t into a scale (Gaussian) pyramid and compute a set of dense descriptors for each pyramid level. We used dense greyscale SIFT [19, 6] in our experiments, but are in no way limited to it. Obviously, color information is important for many classes, especially so for retail products, and we in fact use it in the Phase 2 (fine-grained refinement) of our approach. However, initial detection and recognition in greyscale has its benefits in terms of higher resilience to color variation due to lighting and camera changes. A significant portion of the final performance of our approach is achieved in Phase 1 without using color.

The grid step for the dense descriptors is reduced according to scale (we use the base step of 4 pixels) and the patch size is fixed (we use 24x24 patches). We denote by $\{F_t^i\}$ the set of all descriptors collected from I_t. The pyramid scales are taken in the range $(0.5, 1]$ (we used $\{0.6, 0.8, 1\}$). The support for the rest of the scales is achieved at query time by down-scaling the query image I_q by a factor of 0.5 in a loop until minimal vertical size is reached. This way the running time is $\leq 1.33\times$ the running time of I_q processed in its original scale alone. The training descriptors are indexed using efficient sub-linear search data structures as described in section 2.1.4, and the query patches are searched within those structures. Due to the sub-linear nature of the search in these data structures, the running time is very little affected by the increase in the number of indexed training descriptors, making it beneficial to sample them in multiple scales as described (this is basically a memory vs speed trade-off). For each sampled training descriptor F_t^i we retain the following metadata: (i) $l(F_t^i) = l(I_t)$ - category label of the source image I_t; (ii) $s(F_t^i)$ scale of the pyramid level from which F_t^i was sampled; (iii) $o(F_t^i)$ relative position of F_t^i within the object. In our implementation we use the relative offset to the object‘s center of mass.

2.1.2 Probabilistic model

We denote by $\{F_q^j\}$ the set of descriptors sampled from the query image I_q, in a dense grid of every 4 pixels. We use

[1]The code will be available at https://github.com/leokarlin/msmo_star_model

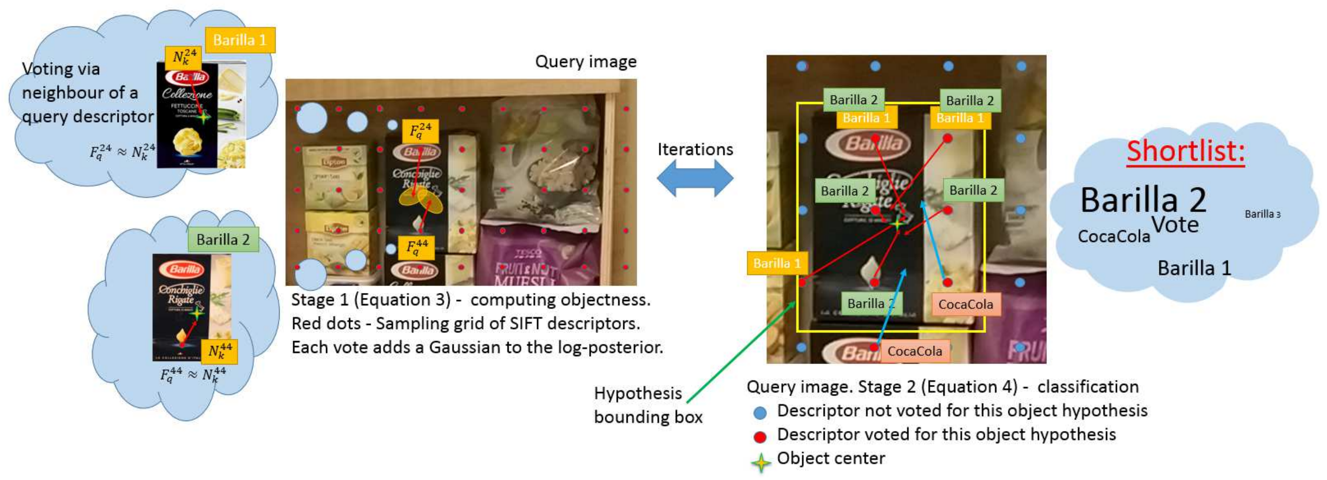
Voting via neighbour of a query descriptor
Barilla 1
N_k^{24}
$F_q^{24} \approx N_k^{24}$
Barilla 2
N_k^{44}
$F_q^{44} \approx N_k^{44}$
Query image
F_q^{24}
F_q^{44}
Stage 1 (Equation 3) - computing objectness.
Red dots - Sampling grid of SIFT descriptors.
Each vote adds a Gaussian to the log-posterior.
Iterations
Barilla 2
Barilla 1
CocaCola
Hypothesis bounding box
Query image. Stage 2 (Equation 4) - classification
● Descriptor not voted for this object hypothesis
● Descriptor voted for this object hypothesis
Object center
Shortlist:
Barilla 2
CocaCola
Vote
Barilla 3
Barilla 1

Figure 2. Graphical diagram illustrating the inference process

24 $\times$ 24 patches. Denote by U a triplet of random variables $U =< X, S, C >$ representing (a random) event of appearance of an object from category C at image location X (object center coordinate) and scale S (scale relative to the nominal size of the object, X and S jointly define a bounding box). Our goal is to compute (or approximate) the posterior probability $P(U|\{F_q^j\})$ for all the possible assignments to U. After doing that, we extract the top scoring hypotheses U out of this posterior by a straightforward Non-Maximal Suppression (NMS) process by looking at overlaps between the hypotheses. We assume a uniform prior over U, and that the posterior decomposes via Naïve-Bayes:

$$P\left(U|\left\{F_q^j\right\}\right) \propto P\left(\left\{F_q^j\right\}|U\right) = \prod_j P\left(F_q^j|U\right). \quad (1)$$

Since objects usually occupy a small portion of an image, we assume most of the descriptors $\{F_q^j\}$ are not generated by any specific assignment to U, that represents a hypothesis that a certain object is present at a certain image region - they belong either to background or to other objects. This intuition is expressed via the decomposition $P\left(F_q^j|U\right) = \sum_{R^j} P(R^j) \cdot P(F_q^j|U, R^j)$, where R^j is an unobserved binary random variable with $R^j = 1$ denoting the event that F_q^j is indeed related to the hypothesis U. Formally, $P(F_q^j|U, R^j) = Q(F_q^j|U)$ if $R^j = 1$ and $P(F_q^j|U, R^j) = Q(F_q^j)$ if $R^j = 0$. According to the assumption above: $P(R^j = 0) \gg P(R^j = 1)$. We explain the distributions Q, and the way to compute them in the next subsection. Figure 3 shows a graphical illustration of the model. In the supplementary material we prove that:

$$\begin{aligned} &LogP\left(U|\{F_q^j\}\right) = \text{const} + \textstyle\sum_j LogP(F_q^j|U) \approx \\ &\textstyle\sum_j \frac{Q(F_q^j|U)}{Q(F_q^j)} \propto \sum_j \frac{Q(F_q^j,U)}{Q(F_q^j)}. \end{aligned} \quad (2)$$

In order to find the most likely hypotheses U by computing $arg\max_U LogP(U|\{F_q^j\})$ we need to overcome the enormous size (assuming large number of categories) of the proposal set for $U =< X, S, C >$. We therefore resort to a sequential approximate inference, each time inferring part of the variables and conditioning on them to infer the rest:

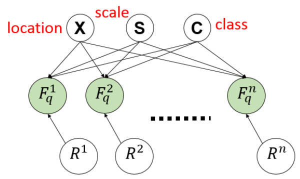

Figure 3. Probabilistic model used for detection and recognition in Phase 1

$$\hat{X}, \hat{S} \leftarrow \arg\max_{X,S} \textstyle\sum_j \frac{Q(F_q^j,X,S)}{Q(F_q^j)} \quad (3)$$

here $Q(F_q^j, X, S) = \sum_C Q(F_q^j, X, S, C)$,

$$\begin{aligned} &\hat{C} \leftarrow \arg\max_C P(C|\{F_q^j\}, \hat{X}, \hat{S}) = \\ &\arg\max_C \textstyle\sum_j \frac{Q(F_q^j,\hat{X},\hat{S},C)}{Q(F_q^j)}, \end{aligned} \quad (4)$$

$$\begin{aligned} &\hat{X}, \hat{S} \leftarrow \arg\max_{X,S} P(X, S|\{F_q^j\}, \hat{C}) = \\ &= \arg\max_{X,S} \textstyle\sum_j \frac{Q(F_q^j,X,S,\hat{C})}{Q(F_q^j)}. \end{aligned} \quad (5)$$

Each of the inference steps (3-5) above returns multiple hypotheses to be processed by the subsequent step, where the final output of (3) is the input to the NMS. We call Equation (3) the objectness step where we infer the potential locations and scales of all the objects present in I_q regardless of their category. We call Equation (4) the short-list step where we generate the short list of potential candidate categories for each object hypothesis returned by the objectness step. Equation (5) is the final detection refinement step, where we refine the location and the scale for each candidate returned by (4) and compute its final score. The following sections explain how to apply the proposed sequential MAP inference process in practice.

2.1.3 Non-parametric estimation of the probabilities and efficient inference

To perform the inference we need to compute the distributions $Q(F_q^j), Q(F_q^j, X, S)$, and $Q(F_q^j, X, S, C)$ for every descriptor F_q^j sampled from the query image I_q. We first find a set of K approximate Euclidean nearest neighbors $\{N_k^j | 1 \leq k \leq K\}$ of F_q^j in the set of all the training image descriptors $\{F_t^i | \forall i, t\}$ obtained as explained in Subsection 2.1.1. From the set of nearest neighbors we compute $Q(F_q^j, X, S)$ using approximate Kernel Density Estimate:

$$Q(F_q^j, X, S) \approx Q(S) \cdot \frac{1}{K} \sum_{k=1}^{K} \phi(F_q^j, N_k^j, X, S), \qquad (6)$$

where

$$\begin{aligned} &\phi(F_q^j, N_k^j, X, S) = exp(-\tfrac{1}{2}\|F_q^j - N_k^j\|^2/\sigma^2) \cdot ... \\ &\cdot exp\left(-\tfrac{1}{2}S^2\|X - \mid z(F_q^j) + o(N_k^j) \mid \|^2/\rho^2\right) \cdot ... \\ &\cdot exp\left(-\tfrac{1}{2}\left(S - s(N_k^j)\right)^2/\gamma^2\right) \end{aligned} \qquad (7)$$

The value for $Q(F_q^j, X, S)$ is thus an averaged contribution of the K nearest neighbors. The probability of observing F_q^j at location X, scale S, conditioned on the specific neighbor descriptor N_k^j, is modeled as a product of three components in the equation above. The first one $exp(-\frac{1}{2}\|F_q^j - N_k^j\|^2/\sigma^2)$ is the data fidelity term, penalizing distance between the descriptors F_q^j and N_k^j. Here σ is our tolerance to variation of the expected descriptor appearance. The second term $exp\left(-\frac{1}{2}S^2\|X - \mid z(F_q^j) + o(N_k^j) \mid \|^2/\rho^2\right)$ is the penalty for deviation in expected spatial location, where $z(F_q^j)$ is the image location of F_q^j in I_q and $o(N_k^j)$ is the relative offset between N_k^j and the center of the object in its respective scale. Here ρ is our tolerance to the expected local object deformation. The final term $exp(-\frac{1}{2}\left(S - s(N_k^j)\right)^2/\gamma^2)$ measures the discrepancy in scale, so γ is our tolerance to local scale variation. The $\phi(F_q^j, N_k^j, X, S)$ is the 'belief' of F_q^j that some object (from any category of interest) appears at location X with scale S, based on matching training descriptor N_k^j. We use values $\sigma = 0.2$, $\rho = 15$ and $\gamma = 0.1$ throughout all our experiments. The parameters were set on a small validation set not used in the experiments. To balance the different object scales inherently being represented by a different number of descriptors, we set $Q(S) \propto \frac{1}{S^2}$. Marginalizing over X and S, we are left with the fidelity term alone:

$$Q(F_q^j) \approx \frac{1}{K} \sum_{k=1}^{K} exp\left(-\frac{1}{2}\|F_q^j - N_k^j\|^2/\sigma^2\right) \qquad (8)$$

To compute (3) efficiently with $Q(F_q^j, X, S)$ as defined by (6) for each scale S (in our case $S \in \{0.6, 0.8, 1\}$) we first compute a weighted histogram of the size of I_q, where each pixel $z(F_q^j) + o(N_k^j)$ accumulates weight $\dfrac{exp\left(-\frac{1}{2}\|F_q^j - N_k^j\|^2/\sigma^2\right)}{S^2 Q(F_q^j)}$ and then convolve it with a 2D symmetric Gaussian kernel with STD ρ/S. Finally, we average across nearby scales using a 1D Gaussian kernel with STD γ. Using the same notation as above we set:

$$Q(F_q^j, X, S, C) \propto \sum_k id\left(l(N_k^j) = C\right) \cdot \phi(F_q^j, N_k^j, X, S)$$

where $id(\cdot)$ is the indicator function. Now, Equation 4 can be simply computed by maximizing over a weighted histogram accumulating to cells indexed by $l(N_k^j)$ with weights given by $\phi(F_q^j, N_k^j, \hat{X}, \hat{S})/Q(F_q^j)$. Finally, Equation 5 is computed exactly as Equation 3, but filtering out all the weights for which $l(N_k^j) \neq \hat{C}$.

2.1.4 Efficient nearest neighbor computation

To perform inference efficiently, we need to quickly find K approximate nearest neighbors of multiple query descriptors F_q^j in a large set of indexed training descriptors. For example, in the GroZi-3.2K dataset with 3.2K categories we have 7M training descriptors and 55K query descriptors in an average query image. A possible approach is to use a kd-tree [22]. However, for the numbers mentioned, it takes over 20 seconds for the NN search alone, on a modern PC. We therefore resort to a different technique building on and extending the seminal ideas proposed in CSH [14] and in [28]. We first use the spectral hashing [35] based LSH technique to find good matches (≤ 0.5 distance in L2-normalized descriptor space) for a subset (20%) of all the F_q^j descriptors. Here we assume that on each of the objects in the query there is at least one descriptor that has found a good match. We then extend these matches as follows: assume a patch F_q^j in location $z(F_q^j)$ has a match N_k^j. For every unmatched nearby patch F_q^m we consider a match N_k^m, sampled from the same training image as N_k^j, such that $o(N_k^j) - o(N_k^m) = z(F_k^j) - z(F_k^m)$. To save the computations, we pair an unmatched patch F_q^m only with a single already matched patch F_q^j that is closest in query image pixel coordinates (according to the distance transform). Only good matches (as above) are kept. We iteratively repeat the match extension process. The process is randomized by independently ignoring each previously matched patch with probability p (we used $p = 0.5$). This way, for any unmatched patch, the probability of extending the match from its n^{th} closest already matched patch is equal to $(1-p)p^{n-1}$. This implicitly allows to extend the matching from a number of nearby patches and not just from the spatially closest one.

2.2. Phase 2: Deep fine-grained refinement

So far we have presented an approach for fast and reliable detection of objects of interest and their initial classification using a very limited training data (one example per class). In this section we describe a deep Convolutional Neural Network (CNN) based method for improving fine-grained classification performance of the system. We design a CNN that admits the detected object bounding boxes together with their associated scored short-lists of possible classifications produced by Phase 1, and produces improved classifications for these bounding boxes. The CNN is trained to infer the distribution over the categories, using deep visual features combined with Phase 1 predictions. We synthesize large amounts of training images using geometric and photometric transformations of the few available training examples. For each object detection returned by Phase 1 in a test image, we multiply the scores of the top 5 classification hypotheses of Phase 1 by corresponding CNN confidences thus obtaining the final result.

Architecture of the CNN: Our DNN is based on fine tuning a variant of the VGG-f network [3]. Specifically, we use the first 2-15 layers VGG-f trained on ImageNet [4] unchanged. The reason we do not perform full training is the essentially limited data available per class. We reduce the stride of the first conv1 layer from 4 to 2 to increase the spatial sampling rate of the input images. This decreases the minimal image size admissible by the network. To remedy this, we add one additional set of layers (conv, ReLU, max-pooling, and LRN) after the 15th pool5 layer. The size of the conv. filters is 3×3, and max-pool is the same as other layers of this type in VGG-f. We change the size of the fc6 and fc7 fully-connected layers from 4096 to 2048. For canonical image size (which we set to 325 by appropriate selection of the spatial dimensions of the additional filter), the output of fc7 is $1 \times 1 \times 2048$, and we concatenate it with a vector of $1 \times 1 \times N_{cats}$ per-category confidence values produced by Phase 1 for the currently examined hypothesis (N_{cats} is the number of categories), before feeding it to fc8.

CNN training data synthesis: We use random geometric and photometric image transformations to generate $O(10^4)$ training examples from a single (studio) image per category. Learning the parameters for photometric transformations requires a few dozen annotated objects in real world images. The synthesis process (Figure 3 in supplementary material) starts by generating an array of training object examples over a random natural image background. Second, the generated image undergoes a random homography followed by a random photometric transformation(obtained as explained below).

Geometric variations: The homography is generated by drawing five random values for the yaw, pitch and roll angles, translation and scale. This extends the mirror reflections and translations commonly used for geometric augmentation in other methods [15].

Photometric variations: We use a small auxiliary dataset of annotated objects consisting of 80 retail in-situ product images (the crops) paired with their studio catalog images (the templates). Each such crop-template pair is registered by a homography. For each pair we compute a local linear filter that, when applied to the template, produces an image as close as possible to the corresponding crop, in the least squares sense. We use the IRLS algorithm [9] to compute the filters. Intuitively, the learned filter represents the combined effect of the camera Point Spread Function, illumination, and contrast loss, observed in the crop. In our experiments, we produced filters of sizes $m \times m$ where $m \in \{3, 5, 7, 9\}$ from each crop-template pair, resulting in a collection of 320 photometric filters. Photometric distortions are generated as random convex combinations of three randomly chosen filters from the collection.

2.3. Phase 3: Temporal integration and tracking

When applying the Phases 1 and 2 to video input we can obtain additional performance boost using temporal integration of per-frame detection and recognition results. To verify this, we use the KLT tracker [30] (implemented in OpenCV [2]) to compute homographies between consecutive video frames, and track the detection bounding boxes to subsequent frames until they become covered (IoU≥ 0.5) by other detection boxes or exit the frame. To limit the effect of False Alarms we use a score decay factor of 0.8 for all the tracked boxes that are still uncovered. The benefit of this simple pipeline is illustrated by the quantitative results on all the tested video datasets (see Table 1).

3. Application to point clouds

The proposed approach can be used in order to detect the pose of a camera within a 3D Point Cloud Model (PCM) coordinate system using just a single (greyscale) image as an input. We assume the point cloud was generated via some form of SFM (we used VisualSFM [38, 39]). The PCM is accompanied by a set of *training* images depicting different (partial) views of the scene. For each image we have a set of correspondences between some of the image pixels and 3D points of the PCM, and a known camera pose (namely, rotation matrix R, translation vector T and calibration matrix K) w.r.t. the PCM coordinate system.

Our goal here is to find the camera pose (R and T) out of a single query image of the scene modeled by the PCM captured at a different time by a different camera (with known K). The main idea is as follows: even though the query image may be taken from a new pose not seen during the training, for many sufficiently local regions of the query there exist corresponding training image regions related to them only by 2D translation and scaling. Defining each training image as an object we would like to de-

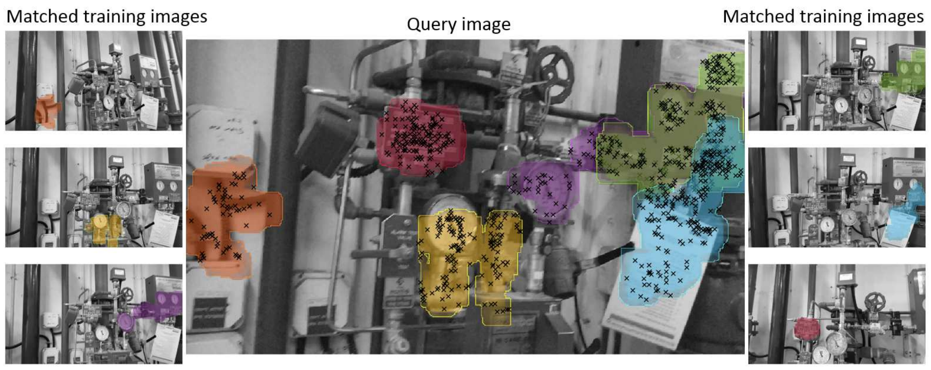

Figure 4. Finding region matches between a query image and the training images in order to infer the camera pose w.r.t. the PCM.

tect, we can use the algorithm of Phase 1 (as described in section 2.1) to find matches to those objects in the query image whenever possible. The found matches are regions of arbitrary location, scale and shape determined by back-projection (see Figure 3): for each detection hypothesis of a training image $\hat{C}$ detected at query image location $\hat{X}$ with scale $\hat{S}$, we build a 'back-projection' matrix BP of the same size as the query image, by assigning weights $\max_k[id\left(l(N_k^j) = \hat{C}\right) \cdot \phi(F_q^j, N_k^j, \hat{X}, \hat{S})]$ to locations $z(F_q^j)$ (from which the query descriptor F_q^j was sampled). This weight represents the maximal belief that the nearest neighbors $\{N_k^j\}$ of F_q^j have contributed to the detection hypothesis $\hat{C}$, $\hat{X}$, $\hat{S}$. Using the detected center location $\hat{X}$ and scale $\hat{S}$ we can map each cloud point P visible in the training image $\hat{C}$ onto the corresponding query image pixel p. Each matched pair $\{P, p\}$ is weighted by the $BP(p)$ - the BP value at pixel p. Collecting all the weighted point correspondences $\{P, p, BP(p)\}$ from all the detected hypotheses gives us a set of weighted matches, which we feed into the standard IRLS algorithm [9] in order to obtain the camera pose for the query image.

4. Results

To test the proposed approach we applied it to six different datasets. Three of the datasets were collected by us: PCPE - Point Clouds Pose Estimation dataset, video games dataset (3.7K categories), and retail products dataset (121 categories). Two additional existing retail benchmarks are Grozi-120 [20] and Grozi-3.2K [7] (called "Grocery Products" in original publication), containing 120 and 3.2K retail product categories respectively. Another benchmark is the FlickrLogos-32 dataset of logos in the wild [27]. The retail and logo datasets were captured in-situ with very limited training (primarily one studio image per retail product, 40 images per logo). In our experiments we demonstrate high quality performance on our own datasets, as well as improvement of the state-of-the-art on all the aforementioned benchmarks. In addition, to demonstrate the inherent difficulty of the very limited training setup to the top-performing deep methods such as the popular R-CNN methodology, we used the public implementation of the seminal Faster RCNN (FRCNN) [26] method (which exhibits strong results on PASCAL and other datasets) and applied it to some of our tested datasets. Having just one example per class, we had to train(fine-tune) FRCNN using the same simulated training data used to train our DNN in Phase 2 (section 2.2). Table 1 summarizes the quantitative performance of the various Phases of our method, as well as comparisons to FRCNN and state-of-the-art. Full precision-recall graphs are given in the supplementary.

Runtime: An un-optimized Matlab implementation of the proposed algorithm runs in less than 1 second for a 1 mega-pixel image on a regular PC. For the point cloud pose estimation application, somewhat more optimized C++ code runs at 0.2 seconds per frame.

The **Grozi-3.2K dataset** consists of 680 test images collected from 5 stores. Each image contains multiple instances of products out of total 3235 food product categories, also organized in 27 larger classes, such as: 'bakery', 'candy', 'cereal', etc. The training data consists of 1 studio image per product. Many of the products appear deformed on the shelves (e.g. rotated, piled, etc.) and with similar but different packaging. Some example test images accompanied by our detection results are shown in Figure 1-a. We conducted the experiments on this dataset using the protocol of [7]. For each category out of total 3235 only one studio example was available. Consequently, in order to test FRCNN on this dataset, we reduced the recognition task to only the 27 'larger classes' the products were mapped to, with hundreds of examples for each class. As a result FRCNN produced mAP of 81.1% on this simplified task, while our method produced mAP of 86.47%.

The **Grozi-120 dataset** from [20], is comprised of 29 test videos of 120 products on various shelves in retail stores. The test set has 4973 frames annotated with ground-truth (GT). The training consists of approximately 6 studio images per product category. We used the same protocol as [7] to evaluate the performance. Figure1-b shows some example test images and our method results. In Table 1 we

Table 1. Quantitative performance evaluation of our method and comparison to the state-of-the-art methods [7], [10], and FRCNN [26] on the relevant datasets (results of [7] and [10] are taken from respective papers). All the numbers are mean Average Precision (mAP) computed as in [7]. Due to poor image quality or the too large number of (1-example) diverse categories in the Grozi-120/3.2K datasets respectively, DL based methods (namely FRCNN and our Phase 2) were not tested there (please see text for a reduced experiment of applying FRCNN on Grozi-3.2K). The Grozi-3.2K and FlickrLogos-32 datasets contain only still images, making Phase 3 irrelevant. The top-5 mAP is mAP computed where hypothesis is considered correct if one of its 5 highest-scored classes is correct.

Dataset \ Algorithm	[7]	[10]	FRCNN [26]	ours Phase 1	ours Phases 1+2	ours full	ours full -top 5
Grozi-3.2K	23.49%	-	-	42.97%	**44.72%**	-	52.16%
Grozi-120	-	-	-	43.22%	-	**49.7%**	49.8%
Grozi-120 subset from [7]	13.21%	-	-	54.22%	-	**62.64%**	62.77%
GameStop	-	-	27.17%	81.3%	87.5%	**89.1%**	93.4%
Retail 121	-	-	53.67%	84.6%	84.7%	**91.3%**	91.9%
Flickr32	-	74.4%	-	78.5%	**79.3%**	-	-

show both the performance for full set, as well as on the subset of 885 frames used in [7] to evaluate their method. We also show the performance of our full system, which exploits the video-based tracking.

The **GameStop dataset** was collected by us in GameStop retail stores (with kind permission of the GameStop®), and contains 5 videos in each frame containing multiple instances (up to several dozens) of 3.7K video game categories captured in their natural store environment. We have manually annotated 1039 frames of those videos with bounding boxes of all the products in each frame. Figure 1-c shows examples of the test images with detection results by the proposed approach. Complete test videos with the detection results are given in supplementary material.

The **Retail-121 dataset** collected by us, contains two videos with multiple instances of 121 fine-grained retail product categories in each frame. The training consisted of one image per product category. The main reason we collected this dataset is to showcase our system performance in a more controlled conditions when the training images represent exactly the appearance of products on the shelves (i.e. same product packaging in both training and test). Figure 1-d shows examples from the 567 test images and the corresponding detection results by the proposed approach. Complete test videos with all the detection results are provided in supplementary material.

The **PCPE dataset** consists of five videos of large scenes and objects which were reconstructed into point clouds using VisualSFM [38, 39]. The point cloud was constructed using 100 frames sampled from a single training movie per scene. For each point cloud an additional test movie was captured at a different time and using a different device. In addition, we moved the objects and tested on them in a different lighting and environment. The task was to detect the presence of the point cloud in each individual frame of the test videos and to accurately detect its pose with respect to the point cloud. Figure 1-e shows example results, and we provide full results in the supplementary material. The pose is considered to be detected correctly if the average point cloud re-projection error is below 5% of image height. The proposed approach successfully detected the pose in 93.5% of the video frames (averaged over all the test videos).

The **FlickrLogos-32 dataset** consists of 32 brand logos, with 40 training images and 30 test images per brand (some examples are given at Figure 1-f). The train and test sets are as defined by FlickrLogos-32 authors. We have trained our system by using 12 plane rotations of each of the 40 training logo examples per brand (the examples are cropped from training images using provided masks). The mean AP of our method is 79.3%, exceeding by 4.9% the state-of-the-art result of [10] that is based on deep networks.

5. Conclusions & future work

We propose a method for fast detection and recognition of thousands of fine-grained object categories in still images and videos, using very restricted ($\sim$ one image per category) training data. The detection and recognition pipeline of the Phase 1, based on a non-parametric probabilistic model, displays superior performance on benchmarks available in the literature as well as on our own datasets. We have shown that the fine grained classification performance of Phase 1 is further improved by a deep network, trained using a specially designed training data synthesis and integrating Phase 1 output. Finally, we have shown that, as expected, an additional performance boost can be obtained on video data using simple tracking. Some natural extensions of the proposed approach that we are considering for future work include: (i) usage of task specific deep descriptors instead of hand crafted SIFT in Phase 1; (ii) combining the proposed objectness step of Phase 1 with the Region Proposal Network (RPN) module of FRCNN [26] for improving the RPN results; (iii) the complication presented by limited training data can be gradually lifted by automatic harvesting of additional examples in unlabeled data using the proposed algorithm, enabling effective training of more data-demanding (e.g., deep) methods.

References

[1] O. Boiman, E. Shechtman, and M. Irani. In defense of nearest-neighbor based image classification. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, June 2008. 2

[2] G. Bradski. citeulike = 2236121. *Dr. Dobb's Journal of Software Tools*. 6

[3] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *British Machine Vision Conference*, 2014. 2, 6

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009. 6

[5] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, Sept 2010. 2

[6] B. Fulkerson, A. Vedaldi, and S. Soatto. Localizing objects with smart dictionaries. In *Proceedings of the 10th European Conference on Computer Vision: Part I*, ECCV '08, pages 179–192, Berlin, Heidelberg, 2008. Springer-Verlag. 3

[7] M. George and C. Floerkemeier. Recognizing products: A per-exemplar multi-label image classification approach. *Computer Vision ECCV 2014*. 2, 3, 7, 8

[8] D. Held, S. Thrun, and S. Savarese. Deep learning for single-view instance recognition. *CoRR*, abs/1507.08286, 2015. 2

[9] P. W. Holland and R. E. Welsch. Robust regression using iteratively reweighted least-squares. *Communications in Statistics - Theory and Methods*, 6(9):813–827, 1977. 6, 7

[10] F. N. Iandola, A. Shen, P. Gao, and K. Keutzer. Deeplogo: Hitting logo recognition with the deep neural network hammer. *CoRR*, abs/1510.02131, 2015. 8

[11] H. Jegou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *Proceedings of the 10th European Conference on Computer Vision: Part I*, ECCV '08, pages 304–317, Berlin, Heidelberg, 2008. Springer-Verlag. 2

[12] L. Karlinsky, M. Dinerstein, D. Harari, and S. Ullman. The chains model for detecting parts by their context. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 25–32. IEEE, 2010. 2

[13] J. Kim, C. Liu, F. Sha, and K. Grauman. Deformable spatial pyramid matching for fast dense correspondences. In *CVPR*, pages 2307–2314. IEEE Computer Society, 2013. 2

[14] S. Korman and S. Avidan. Coherency sensitive hashing. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, pages 1607–1614, Washington, DC, USA, 2011. IEEE Computer Society. 5

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems 25*, 2012. 3, 6

[16] B. M. Lake, R. R. Salakhutdinov, and J. Tenenbaum. One-shot learning by inverting a compositional causal process. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2526–2534. Curran Associates, Inc., 2013. 2

[17] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, CVPR '06, pages 2169–2178, Washington, DC, USA, 2006. IEEE Computer Society. 2

[18] F.-F. Li, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611, April 2006. 2

[19] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, Nov. 2004. 3

[20] M. Merler, C. Galleguillos, and S. Belongie. Recognizing groceries in situ using in vitro training data. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, June 2007. 3, 7

[21] E. G. Miller, N. E. Matsakis, and P. A. Viola. Learning from one example through shared densities on transforms. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 1, pages 464–471 vol.1, 2000. 2

[22] M. Muja and D. G. Lowe. Scalable nearest neighbor algorithms for high dimensional data. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36, 2014. 2, 5

[23] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, CVPR '06, pages 2161–2168, Washington, DC, USA, 2006. IEEE Computer Society. 3

[24] F. Perronnin, Y. Liu, J. Snchez, and H. Poirier. H (2010a) large-scale image retrieval with compressed fisher vectors. In *In: CVPR Perronnin F, Snchez J, Liu Y (2010b) Large-scale*. 2

[25] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015. 2

[26] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 91–99. Curran Associates, Inc., 2015. 2, 7, 8

[27] S. Romberg, L. G. Pueyo, R. Lienhart, and R. van Zwol. Scalable logo recognition in real-world images. In *Proceedings of the 1st ACM International Conference on Multimedia Retrieval*, ICMR '11, pages 25:1–25:8, New York, NY, USA, 2011. ACM. 7

[28] T. Sattler, B. Leibe, and L. Kobbelt. *Improving Image-Based Localization by Active Correspondence Search*, pages 752–765. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. 5

[29] X. Shen, Z. Lin, J. Brandt, and Y. Wu. *Mobile Product Image Search by Automatic Query Object Extraction*, pages 114–127. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. 3

[30] J. Shi and C. Tomasi. Good features to track. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1993. 6

[31] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proceedings of the International Conference on Computer Vision*, volume 2, pages 1470–1477, Oct. 2003. 2

[32] C. Szegedy, A. Toshev, and D. Erhan. Deep neural networks for object detection. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2553–2561. Curran Associates, Inc., 2013. 2

[33] S. S. Tsai, D. Chen, V. Chandrasekhar, G. Takacs, N.-M. Cheung, R. Vedantham, R. Grzeszczuk, and B. Girod. Mobile product recognition. In *Proceedings of the 18th ACM International Conference on Multimedia*, MM '10, pages 1587–1590, New York, NY, USA, 2010. ACM. 3

[34] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu. Learning fine-grained image similarity with deep ranking. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, pages 1386–1393, Washington, DC, USA, 2014. IEEE Computer Society. 2

[35] Y. Weiss, R. Fergus, and A. Torralba. *Computer Vision – ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part V*, chapter Multidimensional Spectral Hashing, pages 340–353. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. 2, 5

[36] T. Winlock, E. Christiansen, and S. Belongie. Toward real-time grocery detection for the visually impaired. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, pages 49 –56, June 2010. 3

[37] A. Wong and A. L. Yuille. One shot learning via compositions of meaningful patches. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015. 2

[38] C. Wu. Towards linear-time incremental structure from motion. In *3D Vision - 3DV 2013, 2013 International Conference on*, pages 127–134, June 2013. 6, 8

[39] C. Wu, S. Agarwal, B. Curless, and S. M. Seitz. Multicore bundle adjustment. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3057–3064, June 2011. 6, 8

[40] S. Yang, L. Bo, J. Wang, and L. G. Shapiro. Unsupervised template learning for fine-grained object recognition. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 3122–3130. Curran Associates, Inc., 2012. 2

[41] N. Zhang, J. Donahue, R. Girshick, and T. Darrell. *Computer Vision – ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I*, chapter Part-Based R-CNNs for Fine-Grained Category Detection. 2014. 2