Regionlets for Generic Object Detection

Xiaoyu Wang, Ming Yang, Shenghuo Zhu, and Yuanqing Lin

Abstract—Generic object detection is confronted by dealing with different degrees of variations, caused by viewpoints or deformations in distinct object classes, with tractable computations. This demands for descriptive and flexible object representations which can be efficiently evaluated in many locations. We propose to model an object class with a cascaded boosting classifier which integrates various types of features from competing local regions, each of which may consist of a group of subregions, named as *regionlets*. A regionlet is a base feature extraction region defined proportionally to a detection window at an arbitrary resolution (i.e., size and aspect ratio). These regionlets are organized in small groups with stable relative positions to be descriptive to delineate fine-grained spatial layouts inside objects. Their features are aggregated into a one-dimensional feature within one group so as to be flexible to tolerate deformations. The most discriminative regionlets for each object class are selected through a boosting learning procedure. Our regionlet approach achieves very competitive performance on popular multi-class detection benchmark datasets with a single method, without any context. It achieves a detection mean average precision of 41.7 percent on the PASCAL VOC 2007 dataset, and 39.7 percent on the VOC 2010 for 20 object categories. We further develop *support pixel integral images* to efficiently augment regionlet features with the responses learned by deep convolutional neural networks. Our regionlet based method won second place in the ImageNet Large Scale Visual Object Recognition Challenge (ILSVRC 2013).

Index Terms—Object detection, regionlet, boosting, object proposals, selective search, deep convolutional neural network

1 Introduction

Despite the success of face detection, where the target objects are roughly rigid, generic object detection remains an open problem mainly due to the challenge of handling all possible variations with tractable computations. In particular, different object classes demonstrate variable degrees of deformation in images, either due to their nature, e.g., living creatures like cats are generally more deformable than man-made objects like vehicles, or due to viewing distances or angles, e.g., deformable objects may appear somewhat rigid at a distance and even rigid objects may show larger variations under different viewing angles. These pose a fundamental dilemma to object class representations: on the one hand, a delicate model describing rigid object appearances may hardly handle deformable objects; on the other hand, a high tolerance of deformation may result in imprecise localization or false positives for rigid objects.

Prior arts in object detection cope with object deformation with primarily four typical strategies. First, if spatial layouts of object appearances are roughly rigid such as faces or pedestrians at a distance, the classical Adaboost detector [1] models local variations with an ensemble classifier of fast features. This enables a sliding window search with cascaded classifiers, achieving precise and efficient localization. Second, the deformable part model (DPM) method [2] inherits HOG window template matching [3] but explicitly models deformations using latent variables. Using the DPM, an exhaustive search of possible locations, scales, and aspect ratios is critical to localize objects. In order to alleviate the computational cost, DPM approaches have been accelerated by coarse-to-fine search [4], branch and bound [5], and cross-talk methods [6]. Third, object recognition methods using spatial pyramid matching (SPM) of bag-of-words (BoW) models [7], which can inherently tolerate large deformations, are adopted for detection [8]. These detectors are applied to thousands of object-independent candidate detection windows [8], [9], [10], instead of millions of sliding windows. Recently, deep convolutional neural networks (DCNN) [11] have exhibited superior capacities in learning invariance in multiple object categories from large amounts of training data [12], [13], and they have been successfully adapted to object detection with a sliding window search [14] or object proposal based search [10], [15].

Objects may appear at different scales with different aspect ratios. Most existing approaches [1], [2], [3], [4], [7] train an object detector at a fixed scale and aspect ratio. In the training phase, object bounding boxes are normalized to exactly the same resolution. In the testing phase, a test image is resized to image pyramids in order to detect objects appearing at scales different from that of the testing model. Multiple models are usually learned to detect objects with different aspect ratios. On the one hand, normalizing training samples may yield more consistent visual appearance which simplifies the learning process. On the other hand, the normalization may alter the object appearance at the original scale, and evaluating the model at all possible scales involves expensive computation. An alternative strategy is to employ a large codebook to encode features extracted at the original object resolution to a fixed length vector [8], regardless of the scale or aspect ratio of an object.

- *X. Wang and Y. Lin are with the Department of Media Analytics, NEC Laboratories America, Cupertino, CA, 95014. E-mail: fanghuaxue@gmail.com, ylin@nec-labs.com.*
- *M. Yang is with the AI Research, Facebook Inc., Menlo Park, CA, 94025. E-mail: mingyang@fb.com.*
- *S. Zhu is with Alibaba Group. E-mail: shenghuo@gmail.com.*

Manuscript received 4 July 2014; revised 13 Oct. 2014; accepted 3 Dec. 2014. Date of publication 8 Jan. 2015; date of current version 4 Sept. 2015. Recommended for acceptance by V. Ferrari. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TPAMI.2015.2389830

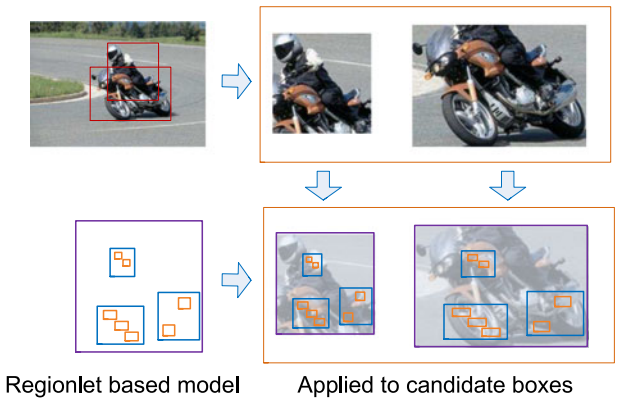


Fig. 1. Illustration of the regionlet representation. Regionlet representation can be applied to candidate bounding boxes that have different sizes and aspect ratios. A regionlet-based model is composed of a number of regions (denoted by blue rectangles), and then each region is represented by a group of competing regionlets (denoted by the small orange rectangles inside each region).

Therefore, it is feasible to train a single detector to detect objects at their original resolutions without resizing testing images. These detection approaches inspired us to investigate a descriptive and flexible object representation, which delivers the modeling capacity for describing both rigid and deformable objects in a unified framework and handles multiple scales and aspect ratios efficiently.

In this paper, we propose a new object representation strategy for generic object detection, which incorporates adaptive deformation handling into both object classifier learning and basic feature extraction. Each object bounding box is classified by a cascaded boosting classifier, where each weak classifier takes the feature response of a region inside the bounding box as its input. The region is in turn represented by a group of small subregions, named as *regionlets*. The sets of regionlets are selected from a huge pool of candidate regionlet groups by boosting. The relative spatial positions of both the regionlets within the region and the region within an object bounding box are stable. Therefore, the proposed regionlet representation can model fine-grained spatial appearance layouts. Moreover, the feature responses of regionlets within one group are aggregated into a one-dimensional feature, and the resulting feature is robust to local deformation. Also, our regionlet model is designed to be flexible enough to take bounding boxes with different sizes and aspect ratios. The flexibility enables our regionlet-based classifier to directly evaluate object proposals from selective search [8], which are often thousands of candidate bounding boxes in contrast to hundreds of thousands (if not millions) of sliding windows required for exhaustive search.

Fig. 1 illustrates the regionlet representation, where the regionlets shown as orange boxes are grouped within blue rectangular regions. The regionlets and their groups for one object class are learned in boosting with stable relative positions to each other. When they are applied to two candidate bounding boxes, the feature responses of regionlets are obtained at their respective scales and aspect ratios. The effective regionlets configurations are learned in training and fixed in testing with no spatial configuration inference (like DPM [2]), leading to a fast evaluation speed.

The features extracted from the regionlets in one group are aggregated into a 1D feature as one weak classifier, which allows the boosting classifier to incorporate multiple types of features flexibly. This requires efficient random access of features inside arbitrary regionlets during training and testing, which is straightforward for dense features extracted at a pixel grid but complicated for spatially sparse features (e.g., sparse SIFT features not available at each pixel). To leverage sparse features in the regionlet detector, we propose a *support pixel integral images* (SPII) technique, that utilizes a two-layer indexing scheme to retrieve integral vectors of spatially sparse features. As the first layer, we identify spatial locations where the integral computation is indispensable, referred to as the set of support pixels, then store and hash these integral vectors by their spatial locations for fast access. The second layer stores the hashing entries of the integral vector for each pixel location. SPII is much faster than conventional integral image and substantially reduces the memory usage. We demonstrate the effectiveness of SPII by incorporating the responses from a deep convolutional neural network into our regionlets framework and improving the detection performance significantly.

The major contributions of this paper are three-fold. 1) The novel regionlet-based representation that models relative spatial layouts inside an object. It accommodates variations, especially deformations, by the max-pooling of feature responses and data-driven regionlet group selection. 2) The regionlet-based detector efficiently applies to arbitrary bounding boxes at different scales and aspect ratios. 3) The support pixel integral image allows fast access to feature vectors in arbitrary regions given a set of spatially sparse features in an image. As validated in the experiment, the proposed regionlet detector adaptively handles a varying degree of deformation in diverse object classes in a data driven fashion, leading to the state-of-the-art performance on PASCAL VOC 2007 and 2010 datasets [16] *without* outside training data. Moreover, this approach achieved second place on the detection task in the ImageNet Large Scale Visual Object Recognition Challenge (ILSVRC 2013) [17].

2 Related Work

Object detection is arguably an indispensable component for most vision tasks, and it has achieved prominent success for specific targets such as faces [1], [18] and pedestrians [3], [19], [20], [21], [22], [23], [24]. We briefly review related techniques in object detection.

Discriminative and efficient features are the cornerstones for object detection. Viola and Jones's face detector [1] employed Haar features in a cascaded boosting classifier to differentiate facial textures; Dalal and Triggs [3] proposed the histogram of oriented gradients (HOG) templates to model pedestrian silhouettes by a linear SVM. Later, Wang et al. [20] showed that the local binary patterns (LBP) [25] are complementary to HOG features and their combination enhances pedestrian detection performance. Tuzel et al. [19] showed that covariance features, which encode the intensity gradients as well as the second order derivative among pixel locations, can effectively delineate pedestrian appearances. In these methods, the fixed-size templates strictly

align features according to their spatial locations in the classifiers. They are capable of handling roughly rigid objects, but they have difficulty in detecting more deformable generic object classes.

The most well-known works on handling object deformation are the deformable part model [2] and its extensions [2], [4], [26], [27]. The DPM object detector consists of a root filter and several part filters. Deformations among parts are inferred with latent variables. Since the resolutions of the object templates are fixed, an exhaustive sliding window search [2] is required to find objects at different scales with different aspect ratios. The exhaustive search can be accelerated by more efficient search methods as in [4], [5], [6], [23], [26], [28], [29].

Recently, deep convolutional neural networks [12], [13] demonstrated superior capabilities in learning a general object representation from large amounts of training data. By amortizing the convolution computation among multiple locations [14] and fine tuning the general representation to a specific object category [10], DCNNs have been successfully applied to object detection [10], [14].

The proposed regionlet-based detection extends the line of boosting with efficient representations, which handles object deformation directly in feature extraction and incorporates HOG, LBP, covariance and DCNN features. Our approach differs from standard boosting methods [1] in two aspects: firstly, it defines a flexible representation which can be adapted to arbitrary detection windows, while conventional boosting only deals with fixed resolution windows. Thus our framework can easily deal with objects at multiple scales and aspect ratios. Secondly, it defines a max-pooling operation among different features within a group of regionlets to obtain more robust representation. The method in [30] shared a similar effort to improve object recognition and segmentation with multiple features. Our method is different in feature design as well as the object detector learning.

Performing object detection with object proposals has been attracting increasing attentions [8], [31], [32], [33], [34], [35]. Our regionlets detector is applied to object proposals obtained from low-level segmentation [8]. It is also applicable to other object proposal methods [32], [36].

3 Regionlets for Detection

Object detection is composed of two key components: determining where the candidate object locations are in an image and discerning whether they are the objects of interests. Beyond the straightforward exhaustive search of all locations, our regionlet detection approach screens the candidate windows derived from selective search [8]. Selective search over-segments an image into superpixels, and then groups the superpixels in a bottom-up manner to propose candidate bounding boxes. The work in [8] shows that such candidate bounding boxes, about 1,000 $\sim$ 2,000 each image, achieve a very high recall. With the proposed bounding boxes, the detection task boils down to extraction of an appropriate object representation from each proposed box and learning of a scoring function to rank the boxes. To this end, we introduce the regionlet-based representation for each candidate bounding box. In our proposed method, we construct a large regionlet feature pool and then design a cascaded boosting learning process to select the most discriminative regionlets for detection.

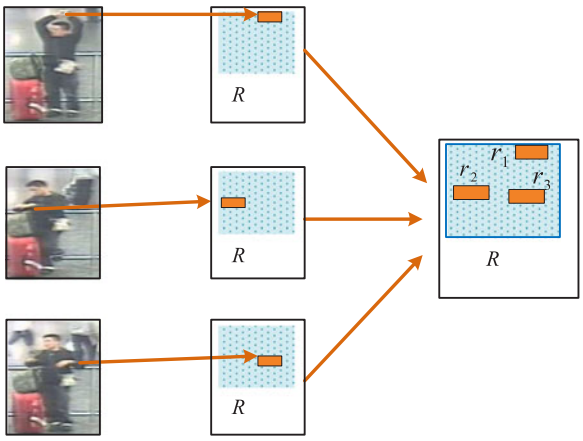


Fig. 2. Illustration of the relationship among a detection bounding box, a feature extraction region and regionlets. A feature extraction region R, shown as a light blue rectangle, is cropped from a fixed position from three samples of a person. Inside R, several small subregions denoted as r_1, r_2 and r_3 (in orange small rectangles) are the *regionlets* to capture the possible locations of the hand for person detection.

In this section we describe what the regionlets are and explain how they are designed to handle deformation. Section 4 introduces the support pixel integral image which allows fast extraction of spatially sparse features for regionlets. Section 5 presents how to construct a regionlet pool and learn a cascaded boosting classifier for an object category by selecting the most discriminative regionlets.

3.1 Regionlets Definition

In object detection, an object category is essentially defined by a classifier where both object appearance and the spatial layout inside an object should be taken into account. For simplicity, appearance features are mostly extracted from some *rectangular* regions within an object, which we refer as feature extraction regions in the paper. Features extracted from a small region often provide good localization ability, but are vulnerable to variations; a big region tends to tolerate more variations but may not be sensitive enough for accurate localization. When large variations, especially deformations, occur, a large rectangle region may not be appropriate for extracting descriptive features of an object because some parts of the region may not be informative or even be distractive. This motivates us to define the subregions of a region, *regionlets*, as the basic units to extract appearance features, and organize them into small groups, which are flexible to describe distinct object categories with different degrees of deformation.

3.1.1 Regionlets Defined Inside Regions

We would like to introduce the regionlets with an example illustrated in Fig. 2. The first column in Fig. 2 shows three samples of a person that are the target object to be detected and they are cropped by black bounding boxes in the second column. A rectangle feature extraction region inside the bounding box is denoted as R, and will contribute a weak classifier to the boosting classifier. Within this region

R, we further spot some small subregions (e.g., r_1,r_2 and r_3) and define them as a group of regionlets. We employ the term *regionlet*, because the features of these subregions will be aggregated into a single feature for R, and they are below the level of a stand-alone feature extraction region in an object classifier. In summary, in the proposed method, a detection bounding box is represented by a number of regions, each of which is composed of a small set of regionlets.

This example also illustrates how regionlets are designed to handle deformation. A hand, as a supposedly informative part for a person, may appear at different locations within the bounding box of a person. If we extract the feature for a hand from the whole region R which roughly covers the possible locations of the hand, the appearance of some non-hand regions on the torso or background are also included in the feature. An ideal deformation handling strategy is to extract features only from the hand region in all three cases. To that end, we introduce three regionlets inside R (In general, a region can contain many regionlets. Here "three" is mainly for illustrative purpose). Each regionlet r serves as a possible location of a hand. Then only features from the regionlets are extracted and aggregated to generate a compact representation for R. Irrelevant appearances from background are largely discarded. More regionlets in R will increase the capacity to model deformations, e.g., a hand may appear in more than three positions. On the other hand, rigid objects may only require one regionlet from a feature extraction region.

3.1.2 Regionlets Normalized by Detection Windows

In this work, the proposed regionlet representations are evaluated on the candidate bounding boxes derived from the selective search approach [8]. In principle, they are also applicable for sliding windows. The selective search approach groups over-segmented superpixels in a bottom-up manner to propose some candidate bounding boxes. This approach typically produces 1,000 to 2,000 candidate bounding boxes for each image, in contrast to millions of windows in an exhaustive sliding window search.

However, these proposed bounding boxes have arbitrary sizes and aspect ratios. As a result, it is not feasible to use template regions (or template regionlets) with fixed absolute sizes that are widely used in sliding window search. We address this difficulty by using the *relative* positions and sizes of the regionlets and their groups to an object bounding box. Fig. 3 shows our way of defining regionlets in contrast to fixed regions with absolute sizes. When using a sliding window search, a feature extraction region is often defined by the top-left (l, t) and the bottom-right corner (r, b) *w.r.t.* the anchor position of the candidate bounding box. In contrast, our approach normalizes the coordinates by the width w and height h of the box and records the relative position of a region $(l', t', r', b') = (\frac{l}{w}, \frac{t}{h}, \frac{r}{w}, \frac{b}{h}) = R'$. As shown in Fig. 3, if a detection window is scaled up by two times, traditional feature region definition covers a visually different patch. Here using normalized coordinates to specify the regionlets ensures that the visual appearances captured by a regionlet are the same even the sample is upscaled or downscaled. Furthermore, these relative region definitions allow us to directly evaluate the regionlet-based

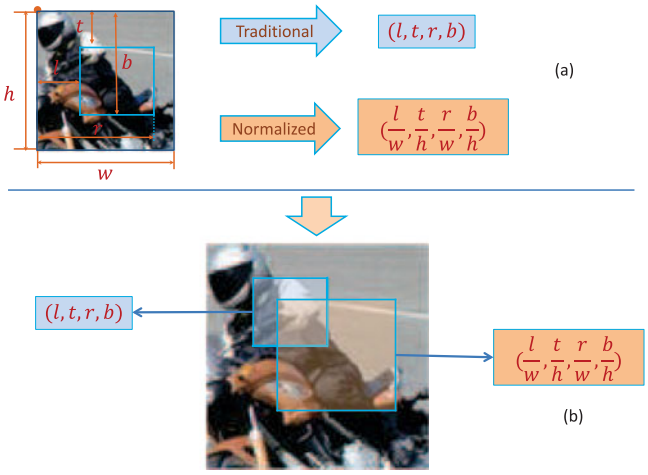


Fig. 3. Relative regions normalized by a candidate window adapt to scale and aspect ratio changes. Feature extraction region for a regionlet/region is jointly determined by the relative coordinates of the regionlet/region and the target detection window.

representation on candidate windows at different sizes and aspect ratios without scaling images into multiple resolutions or using multiple components for enumerating possible aspect ratios.

3.2 Region Feature Extraction

Feature extraction from R takes two steps: 1) extracting appearance features, e.g., the HOG [3] and LBP descriptors [25] from each regionlet respectively; and 2) generating the representation of R based on regionlets' features.

In the first step, we need to extract fixed-length features from individual groups of regionlets, so their features can be aggregated in the second step to generate the region feature. Efficient extraction is straightforward for dense features where the features are accessible at every pixel location, e.g., HOG, LBP, and covariance features, yet complicated for spatially sparse features, e.g., sparse SIFT features (which will be discussed in Section 4). Using the HOG feature as an example, we compute gradients for all pixels inside one regionlet. Then these gradient orientations are used to construct the orientation histogram (eight evenly spaced bins, without thresholding on the gradient magnitude) for this regionlet. Hence, we can extract the HOG feature for an arbitrary regionlet, regardless of its size and aspect ratio. Note that this is different from extracting HOG features in a fixed size cell (e.g., 8×8 cells) followed by pooling and coding. We apply L2 normalization for HOG features and L1 normalization for LBP features. Covariance features are normalized by the corresponding variance.

For the second step, we define a max-pooling operation over features extracted from individual regionlet. It is motivated by that a permutation invariant and exclusive operation over regionlet features allows for deformations inside these regionlets. Denote $T(R)$ as the feature representation for region R, and $T(r_j)$ as the feature extracted from the jth regionlet r_j in R. The operation is defined as following:

$$T(R) = \max_j T(r_j), \tag{1}$$

where $j \in [1, \ldots, N_R]$ and N_R is the total number of regionlets in region R. For one regionlet r_j, we first extract its

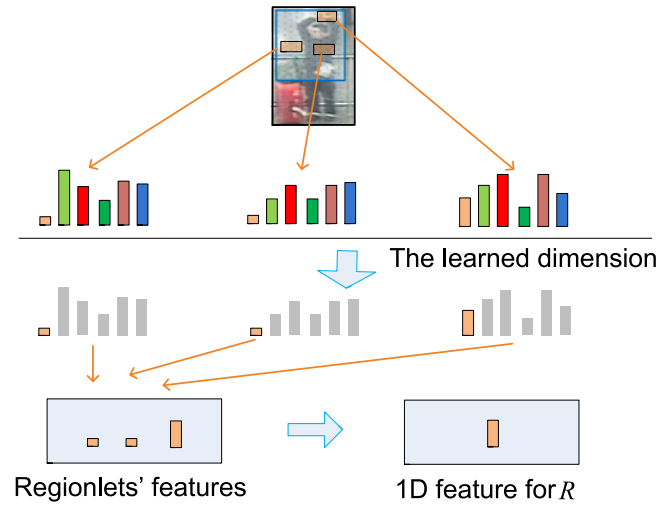


Fig. 4. Example of regionlet-based feature extraction.

feature vectors, such as HOG or LBP histograms. Then, we pick a 1D feature from the same dimension of these feature vectors in one group of regionlets and apply Eq. (1) to form the feature for region R. We have millions of such 1D features in a detection window and the most discriminative ones are determined through a boosting type learning process (to be described in Section 5.2).

Fig. 4 illustrates the process to extract $T(R)$, the 1D feature for a region R. Here we again use the example in Fig. 2, where the blue region R is the one covering the variation of hand locations. Assuming the first dimension of the concatenated low-level features is the most distinctive feature dimension learned for hand, we collect this dimension from all the three regionlets and represent $T(R)$ by the strongest feature response among the group of regionlets.

4 Support Pixel Integral Images for Regionlets

The regionlet-based detector requires efficient access to features from arbitrary regionlets described in Section 3.2, especially when selecting regionlets in the boosting classifier training. This is not an issue for dense features such as HOG and LBP histograms, since the computations like gradient calculation or LBP extraction are amortized among multiple regionlets. However, this is hard for spatially sparse features, where the features are only available on a small set of pixel locations. In this section, we introduce a new technique called *support pixel integral images*, which extends the integral image [1] and integral histogram [37] with a two-layer hashing, for fast access to spatially sparse features. This enables the proposed detector to exploit sparse SIFT or DCNN features.

4.1 Support Pixel Integral Images

Let us first recapitulate the notations in integral images [1] that are widely used to speed up feature computation in object detection. In an integral image, the value of each pixel $I(x, y)$ is the summation of all the top left pixel values $p(i, j)$ from the original image. As a result, the summation of the pixel values in an arbitrary rectangular region is calculated by only three operations in Eq. (2):

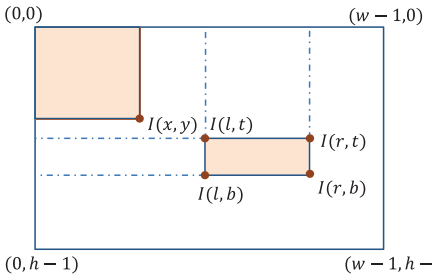


Fig. 5. Illustration of an integral image. $I(x, y)$ is the summation of feature values of all pixels which are located to the top-left of (x, y), including the pixel itself. The sum of features in a rectangular region is computed by three operations on the integral image.

$$sum_{R(l,t,r,b)} = I(r, b) + I(l, t) - I(r, t) - I(l, b), \quad (2)$$

where

$$I(x, y) = \sum_{i \le x, j \le y} p(i, j). \quad (3)$$

Here l, t, r, b represents the left, top, right and bottom coordinates of the rectangular region. The top row and left column pixel values are not included in the region. Fig. 5 illustrates how to compute the integral image and randomly access features in an arbitrary rectangular region.

Certain types of features are only available at a few spatial locations, unlike HOG and LBP features that are extracted from every pixel. For example, sparse SIFT descriptors [38] are extracted on interest points, and the DCNN may only be evaluated on a few locations. For these cases, conventional integral images need to accumulate the feature vectors for each pixel, leading to large memory and computation costs. For instance, given a 640 $\times$ 480 image, a SIFT integral image consumes $640 \times 480 \times 128 \times 4 = 150$ MB memory (if using 32-bit floating point numbers). The computation complexity is O(whd), where w and h are the width and height of the image respectively, and d is the feature dimension. An integral map is built for each feature dimension independently without taking advantage of the spatial sparsity of these features.

To optimize the memory usage and computation cost for integral images on spatially sparse features, we treat each feature vector as an entity and build a two-layer indexing structure to retrieve integral values for features at arbitrary locations. The first layer stores effective integral vectors on a set of support pixels. Here we define "support pixels" as the locations where the integral computation is unavoidable. The integral vectors of support pixels are further indexed by a hashing table. The second layer produces a dense map with the same size as the input image, indicating the hashing table entry of the corresponding integral vector.

Given spatially sparse features, we investigate how an integral value $I(x, y)$ can be represented by one of the previously computed integral values $\{I(x', y') | x' \le x, y' \le y, (x', y') \neq (x, y)\}$. The integral value at (x, y) is the summation of feature values from all the pixels inside the region $(0, 0, x, y)$. An integral image usually starts with the top-left pixel, and subsequent integral values are computed by

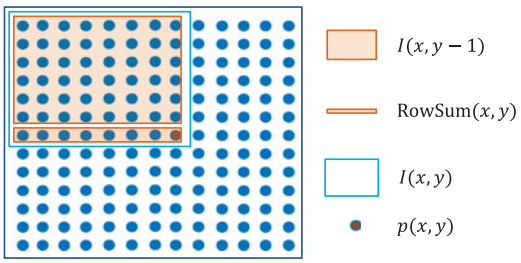


Fig. 6. Inductive integral image computation. The integral value at current point (x, y) equals to the summation of its top immediate integral value at $(x, y-1)$ and the sum of all the pixel values in the current row till (x, y).

$$\begin{aligned} I(x,y) &= \sum_{i=0,j=0}^{i\le x, j\le y} p(i,j) \\ &= \sum_{i=0,j=y}^{i\le x, j=y} p(i,j) + \sum_{i=0,j=0}^{i\le x, j\le y-1} p(i,j) \\ &= \sum_{i=0}^{x} p(i,y) + I(x,y-1) \\ &= RowSum(x,y) + I(x,y-1). \end{aligned} \tag{4}$$

It is clear that $I(x, y) = I(x, y-1)$ if $RowSum(x, y) = 0$, so the computation for integral at location (x, y) can be saved if there is no feature in row y before the point (x, y). On the other hand, Eq. (4) can be re-formed as

$$\begin{aligned} I(x,y) &= \sum_{j=0}^{y} p(x,y) + I(x-1,y) \\ &= ColSum(x,y) + I(x-1,y). \end{aligned} \tag{5}$$

Similarly, the integral value does not change at (x, y) compared to $I(x-1, y)$ if the column summation before (x, y) equals 0. Fig. 6 shows how Eqs. (4) and (5) work for constructing an integral image.

The same property holds for integral vector $\boldsymbol{I}(x, y)$. Let $S = \{(x, y)|\mathbb{1}(x, y) \neq 0\}$ be the set of pixel locations where feature vectors are present, and $\mathbb{1}(\cdot)$ is an indicator function whose value is 1 if the feature vector is available at that location, otherwise 0. Assume all the integral vectors before the point (x, y), i.e., $\{(x', y')|x' \le x, y' \le y, (x', y') \neq (x, y)\}$, have been constructed (in a row-first order), a new integral vector $\boldsymbol{I}(x, y)$ is needed for the position (x, y) only when the following two conditions are satisfied:

$$\begin{aligned} S_y(x) &= \{(x', y')|x' = x, y' \le y, (x', y') \in S\} \neq \emptyset, \\ S_x(y) &= \{(x', y')|y' = y, x' \le x, (x', y') \in S\} \neq \emptyset, \end{aligned} \tag{6}$$

where S_y is the set of feature vector locations along the top vertical direction of (x, y) and S_x is a set of feature vector locations along the left horizontal direction of (x, y).

Eq. (6) reveals that the integral vector at (x, y) shall be computed if there are at least two feature vectors that are located to the left and top of (x, y). In other words, given a pair of feature points (x_1, y_1) and (x_2, y_2), the integral vector computation at $(x_3, y_3) = (\max(x_1, x_2), \max(y_1, y_2))$ is a must because these three locations satisfy Eq. (6), as shown in Fig. 7c. We call this set of pixels *support pixels* if their integral vectors cannot be represented by previously computed integral vectors. The integral vectors of support pixels are computed as $\boldsymbol{I}(x, y)^{(x,y)\in S_I}$, where

$$\begin{aligned} S_I = \{(x,y)|(x,y) = \max((x_1,y_1),(x_2,y_2))\} \\ \text{subject to } (x_1,y_1) \in S, (x_2,y_2) \in S, \end{aligned} \tag{7}$$

and $(x, y) \in S_I$ means pixel (x, y) belongs to the support pixel set. The max operation is independent for horizontal and vertical coordinates. These support integral vectors are still sparsely distributed in the image. To retrieve integral vectors rapidly at any pixel location, we build a dense hashing map to index the entries of support integral vectors:

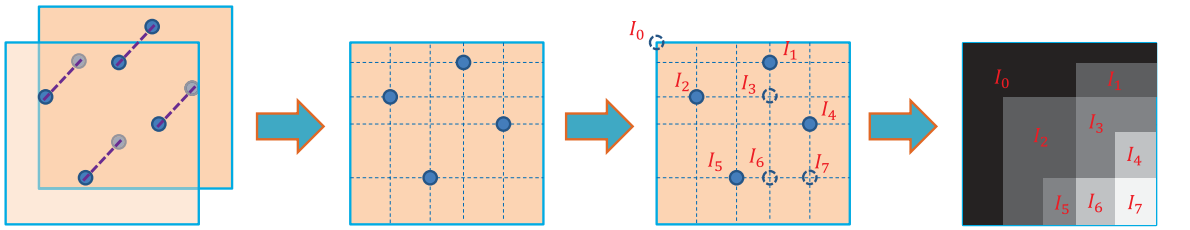
$$\mathcal{I}(x,y) = \begin{cases} \text{Hash}(\boldsymbol{I}(x,y)) & \text{if } S_x(y) \neq \emptyset,\ S_y(x) \neq \emptyset, \\ \mathcal{I}(x-1,y) & \text{if } S_x(y) \neq \emptyset,\ S_y(x) = \emptyset, \\ \mathcal{I}(x,y-1) & \text{if } S_x(y) = \emptyset,\ S_y(x) \neq \emptyset, \end{cases} \tag{8}$$

where $\text{Hash}(\boldsymbol{I}(x, y))$ is the entry of $\boldsymbol{I}(x, y)$ in the hashing table. Given an arbitrary rectangular region $R = (l, t, r, b)$, the summation of the feature vectors inside R is computed as:

$$\boldsymbol{F}(R) = \mathcal{I}^{-1}(r,b) - \mathcal{I}^{-1}(l,b) - \mathcal{I}^{-1}(r,t) + \mathcal{I}^{-1}(l,t), \tag{9}$$

where $\mathcal{I}^{-1}(\cdot)$ is the integral vector retrieved from the hashing table entry $\mathcal{I}(\cdot)$.

The algorithm to build a support pixel integral image is summarized in Algorithm 1. Fig. 7 presents an example for the case of four feature vectors in (a), the grid given by these four pixels is shown in (b). We compute integral vectors only on the support pixels as in (c). Then we construct a dense hashing map for fast retrieval as in (d). For example, all the pixels of the map in the black area stores the same value, which is the hashing table entry of I_0.



(a) Sparse feature vector points (b) Sparse grids on feature points (c) Effective integral vector entries (d) Dense virtual integral vector Map

Fig. 7. Illustration of a support pixel integral image: (a) Two-dimensional features at four pixel locations, (b) The grid specified by these four locations, (c) The integral vectors on the support pixels, and (d) The dense hashing map.

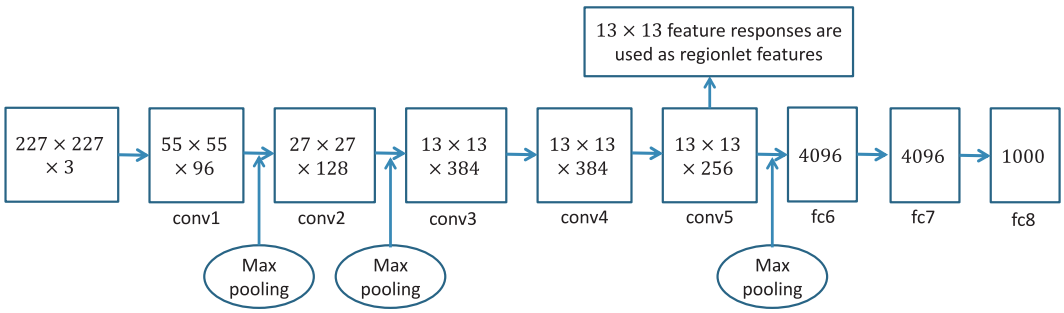


Fig. 8. The Deep CNN architecture used to extract features.

Algorithm 1. Support Pixel Integral Image by a Two-Layer Hashing

Input: Integral vectors $F(x,y)^{(x,y)\in S}$, image width w, image height h, $\boldsymbol{I}(0,0)$.

begin
  $i \leftarrow 0, j \leftarrow 0$
  for $j < h$ **do**
    $i \leftarrow 0$
    for $i < w$ **do**
      if $|S_x(y)| > 0,\ |S_y(x)| > 0$ **then**
        $\boldsymbol{I}(x,y) =$ new integral vector at (x,y)
        $\mathcal{I}(x,y) = Hash(\boldsymbol{I}(x,y))$
      else
        if $|S_x(y)| = 0$ **then**
          $\mathcal{I}(x,y) = \mathcal{I}(x,y-1)$
        else
          $\mathcal{I}(x,y) = \mathcal{I}(x-1,y)$
      $i \leftarrow i+1$
    $j \leftarrow j+1$
  end

Output: Integral vectors on the support pixels $\boldsymbol{I}(x,y)^{(x,y)\in S_I}$. The dense hashing map $\mathcal{I}(x,y)^{x\in[0,w),\ y\in[0,h)}$.

4.2 SPII on DCNN Responses

In this section, we explain how to incorporate a deep convolutional neural network as a feature generator in the regionlet framework using support pixel integral images. Our detector requires access to regionlet features at arbitrary locations. Nonetheless, it is computationally inefficient to produce high-order DCNN responses at each pixel location, considering each DCNN evaluation involves millions of operations (though the low-level convolutions can be amortized). In contrast, we only utilize the responses from the convolutional layer of a DCNN as regionlet features. It allow us to produce features for hundreds of locations in one DCNN evaluation. These features are spatially sparse and can be efficiently indexed by the support pixel integral image.

A DCNN consists of several convolutional layers, max-pooling layers and fully-connected layers. Responses at higher layers such as the last fully-connected layer have been demonstrated to be very effective for object recognition.

To utilize the DCNN responses as regionlet features, we map the response back to its spatial support region in the original image. Convolutions and max-pooling in a DCNN preserve their spatial support information, while neurons in fully-connected layers do not. Since spatial configurations of visual appearances are crucial to localize objects, we only take the responses from the convolutional layers of a DCNN to augment the feature pool for regionlets.

We choose the publicly available neural network model from [39] for feature extraction. As illustrated in Fig. 8, the architecture is similar to [13] except that the input image is 227×227. This network includes five convolutional layers and three fully-connected layers. We map the responses from the fifth convolutional layer before max-pooling back to the original image domain. This convolutional layer outputs $13 \times 13 \times 256$ features [39], i.e., 256 response maps spanned over 13×13 regions with a stride 16×16 in a 227×227 image. Thus one evaluation of this DCNN produces 169 256-dimensional feature vectors. We did not normalize the feature vectors. To produce such feature vectors for images larger than 227×227, we shift the evaluation window of the DCNN to generate features for the entire image. The resulting features for an image are 256-dimensional feature vectors with a 16×16 pixel stride.

These features induced from a DCNN responses are evenly distributed in an image both vertically and horizontally. For two locations $(x_1, y_1) \in S$ and $(x_2, y_2) \in S$, we have $(x_1, y_2) \in S$ and $(x_2, y_1) \in S$, so the support pixel set of integral vectors in Eq. (7) is exactly the set of DCNN response features:

$$S_I = \{(x,y) | (x,y) \in S\}. \tag{10}$$

The total number of integral vectors to compute is $|S_I| \approx \frac{w}{16} \times \frac{h}{16} = \frac{wh}{256}$.

Conventional integral images accumulate the integral vector for each pixel location, leading to a computation complexity of $O(whd)$. The SPII method only requires a computation complexity of $O(|S_I|d)$, which is about 256 times faster if the feature map stride is 16×16. Using the two-layer hashing of integral vectors on the support pixels, the memory cost is also reduced by approximately 256 times.[1]

While DCNN response features are extracted with a fixed stride, regionlets can have arbitrary sizes. Thus a regionlet may cover multiple DCNN responses. We simply apply average pooling over these features to produce the feature for the regionlets.

1. The storage used for hashing entries is less than 1 percent of storage for saving integral vectors at each pixel.

5 Learning the Regionlet Model

We follow the boosting framework to learn the discriminative regionlet groups and their configurations from a huge pool of candidate regions and regionlets.

5.1 Regions/Regionlets Pool Construction

Deformation may occur at different scales. For instance, in person detection, deformation can be caused by a moving finger or a waving hand. A set of small regionlets that is effective to capture finger-level deformation may hardly handle deformation caused by hand movements. In order to account for diverse variations, we build an over-complete pool for regions and regionlets with various positions, aspect ratios, and sizes. Before regionlet learning, a region R' or a regionlet r' is not applied to a detection window yet, so we call R' a feature region prototype and r' a regionlet prototype.

We first explain how the pool of region feature prototypes is constructed. Using the definition in Section 3.1.2, we denote the 1D feature of a region relative to a bounding box as $R' = (l', t', r', b', k)$ where k denotes the kth element of the low-level feature vector of the region. R' represents a feature prototype. The region pool is spanned by $\mathcal{X} \times \mathcal{Y} \times \mathcal{W} \times \mathcal{H} \times \mathcal{F}$, where $\mathcal{X}$ and $\mathcal{Y}$ are respectively the space of horizontal and vertical anchor position of R in the detection window, $\mathcal{W}$ and $\mathcal{H}$ are the width and height of the feature extraction region R', and $\mathcal{F}$ is the space of low-level feature vectors (e.g., the concatenation of HOG and LBP, or the DCNN response features). Enumerating all possible regions is impractical and not necessary. We employ a sampling process to reduce the pool size. Algorithm 2 describes how we sample multiple region feature prototypes.

Algorithm 2. Generation of Region Feature Prototypes

Input: Region width step s_w and height step s_h; maximum width W and height H of region prototypes; horizontal step p_x and vertical step p_y for the region anchor position; minimum width w_{min} and height h_{min} of region prototypes; the number of features N to extract from one region

begin
  $w \leftarrow w_{min}, h \leftarrow h_{min}, i \leftarrow 0$
  for $w < W$ **do**
    $h \leftarrow h_{min}$
    for $h < H$ **do**
      $h \leftarrow h + s_h$
      $l \leftarrow 0, t \leftarrow 0$
      for $l < W - w$ **do**
        $t \leftarrow 0$
        for $t < H - h$ **do**
          for $k = 1, \ldots N$ **do**
            $r \leftarrow l + w, b \leftarrow t + h$
            $R' = (l/w, t/h, r/w, b/h, k)$
            $\mathcal{R} \leftarrow \mathcal{R} \cup \{R'\}$
          $t \leftarrow t + p_y, i \leftarrow i + 1$
        $l \leftarrow l + p_x$
      $h \leftarrow h + s_h$
  $w \leftarrow w + s_w$

Output: Region feature prototype pool $\mathcal{R}$

Afterwards, we propose a set of regionlets with random positions inside each region. Although the sizes of regionlets in a region could be arbitrary, we restrict regionlets in a group to have the identical size because our regionlets are designed to capture the same appearance in different possible locations due to deformation. The sizes of regionlets in different groups could be different. A region may contain up to five regionlets in our implementation.

So the final feature space used as the feature pool for boosting is spanned by $\mathcal{R} \times \mathcal{C}$, where $\mathcal{R}$ is the region feature prototype space, and $\mathcal{C}$ is the configuration space of regionlets. Therefore, we augment a region feature prototype $R' = (l', t', r', b', k, c)$ with a regionlet configuration c.

5.2 Training with Boosting Regionlet Features

We use RealBoost [40] to train cascaded classifiers for our object detector. One boosting classifier consists of a set of selected weak classifiers. Similar to [18], we define the weak classifier using a lookup table:

$$h(x) = \sum_{o=1}^{n-1} v^o \mathbb{1}(B(x) = o), \tag{11}$$

where $h(x)$ is a piece-wise constant function defined by a lookup table, v^o is the table value for the oth entry, $B(x)$ quantizes the feature value x into a table entry, and $\mathbb{1}(\cdot)$ is an indicator function. n is the number of bins which we set to be 8. In each round of the training, v^o is computed based on the sample weight distribution as $v^o = \frac{1}{2}\ln(\frac{U_+^o}{U_-^o})$, where U_+^o is the summation of the weights of the positive examples whose feature values fall into the oth entry of the table. The U_-^o is defined in a similar manner for the weights of negative examples. Let's denote Q as a candidate object bounding box, $R'(Q)$ as a rectangular region in Q, and $T(R'(Q))$ as the one-dimensional feature computed on $R'(Q)$. Substituting x in Eq. 11 with the extracted feature, we can get the weak classifier in the tth round of training for the bounding box Q:

$$h_t(T(R'(Q))) = \sum_{o=1}^{n-1} v_t^o \mathbb{1}(B_t(T(R'(Q))) = o), \tag{12}$$

where v_t^o is the table value of the oth entry at the tth round of training. Then, for each boosting classifier, the learning process obtains a set of weak classifiers H for separating the positive samples from negative ones:

$$H(Q) = \sum_{t=1}^{T} \beta_t h_t(T(R_t'(Q))), \tag{13}$$

where R_t' is the selected region and its regionlets configuration in the tth round of boosting training, and β_t is the weight of the selected weak classifier. The classification result of the bounding box Q is determined by the final round of cascade if it passes all previous ones, and it is expressed as $f(Q) = sign(H^*(Q))$ where H^* denotes the last stage of cascade.

In each cascade training, we generate 100 millions of candidate weak classifiers. A reduced set of 20 K weak classifiers are sampled uniformly to fit into memory. Each round of cascade training except the last cascade terminates once the error rates (37.5 percent for negative samples and 1 percent for positive samples) are achieved. The cascade

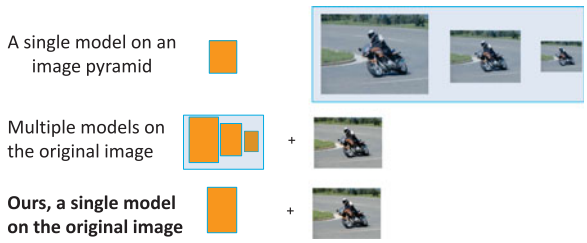


Fig. 9. The regionlet-based detector learns a single classification model but automatically adapts to objects with arbitrary sizes and aspect ratios, without building image pyramids.

rejection threshold is chosen to reject 37.5 percent of the negative training samples. The last round stops when it collects 5 K weak classifiers. The training usually finishes in six-seven rounds.

5.3 Testing with Regionlets

Given a test image, we first propose a number of candidate object bounding boxes [8]. Then, each bounding box is passed along the cascaded boosting classifiers. Because of early rejections, only a small number of bounding boxes reaches the last stage of the cascade. Therefore, our method yields a very fast testing speed. Unlike most methods which learn multiple models for different object aspect ratios or build image pyramids to detect objects at different scales, our approach utilizes a single model to handle objects at different scales or with different aspect ratios, which is possible due to the definition of regionlets by normalized coordinates. This advantage is illustrated in Fig. 9.

6 Experiments

Following most of recent work on generic object detection, we evaluate our object detection performance on the challenging PASCAL VOC 2007 and VOC 2010 datasets. The PASCAL VOC datasets contain 20 object categories. They are popular benchmark datasets for generic multi-class object detection. Detection performance of each category is measured by the average precision. The overall performance is reported by the mean of average precisions (mAP) over all classes. The PASCAL VOC datasets have two evaluation tracks: with or without external training data, named comp3 and comp4 respectively. Our approach competes in comp3 when we do not use DCNN features and comp4 when we use them. Our method is further validated on the much larger ImageNet object detection dataset (ILSVRC2013) [17] which has 200 object categories.

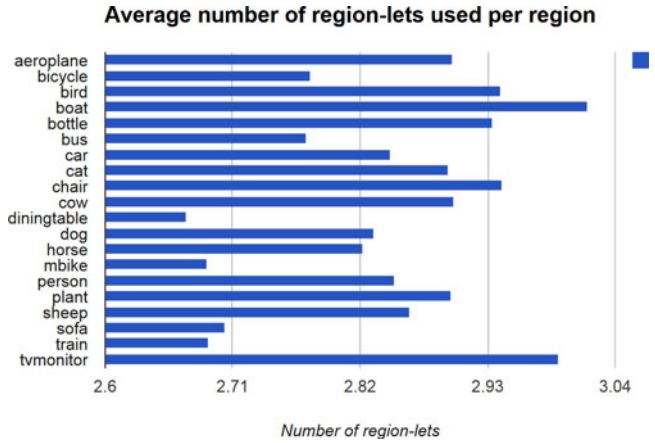


Fig. 10. Statistics of number of regionlets used for each class. Deformable objects generally prefer more regionlets.

6.1 Experiment on PASCAL VOC Datasets

We implement the selective search method [8] to generate candidate detection bounding boxes. HOG [3], LBP [25] and covariance features [19] are adopted as candidate features for the regionlets. To validate the advantages of the proposed approach, we compare it with three baselines: deformable part-based model [2] which is one of the most effective sliding window based detectors, and two recent approaches based on object proposals [8], [32].

Accuracy. Table 1 presents the performance on the PASCAL VOC 2007 dataset, from which we gain some interesting insights from the comparison between the first two baselines. In [2] features from each small 8×8 patches are

TABLE 1
Performance Comparison with the Baselines on the PASCAL VOC 2007 Dataset (Average Precision Percentage)

	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
DPM [2][1]	33.2	**60.3**	10.2	16.1	**27.3**	54.3	58.2	23.0	**20.0**	24.1	26.7	12.7	**58.1**	48.2	43.2	12.0	21.1	36.1	46.0	43.5	33.7
SS_SPM [8][2]	43.5	46.5	10.4	12.0	9.3	49.4	53.7	39.4	12.5	36.9	42.2	26.4	47.0	52.4	23.5	12.1	29.9	36.3	42.2	48.8	33.8
Regionlets-S	50.8	44.6	17.0	23.5	16.7	48.9	67.6	39.1	16.5	32.4	44.0	18.9	52.1	46.6	36.6	13.8	33.8	27.6	55.5	50.4	36.8
Regionlets-M	**54.2**	52.0	**20.3**	**24.0**	20.1	**55.5**	**68.7**	**42.6**	19.2	**44.2**	**49.1**	**26.6**	57.0	**54.5**	**43.4**	**16.4**	**36.6**	**37.7**	**59.4**	**52.3**	**41.7**

DPM: *deformable part based model.* ***SS_SPM:*** *selective search with spatial pyramid features.* ***Regionlets-S:*** *our regionlets approach with a single regionlet per region.* ***Regionlets-M:*** *our regionlets approach allowing for multiple regionlets per region.* ***mAP*** *is the mean average precision over all the 20 categories.*
[1] *We obtained the latest results from the author's website http://www.cs.berkeley.edu/rbg/latent/index.html.*
[2] *We read the performance from the figure in [8].*

TABLE 2
Performance Comparison with the Baselines on the PASCAL VOC 2010 Dataset (Average Precision Percentage)

	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
DPM [2]	45.6	**49.0**	11.0	11.6	**27.2**	50.5	43.1	23.6	**17.2**	23.2	10.7	20.5	42.5	44.5	41.3	8.7	29.0	18.7	40.0	34.5	29.6
SS_SPM [8]	58.2	41.3	19.2	14.0	14.3	44.8	36.7	48.8	12.9	28.1	28.7	**39.4**	**44.1**	52.5	25.8	14.1	38.8	**34.2**	43.1	42.6	34.1
Regionlets-M	**65.0**	48.9	**25.9**	**24.6**	24.5	**56.1**	**54.5**	**51.2**	17.0	**28.9**	**30.2**	35.8	40.2	**55.7**	**43.5**	**14.3**	**43.9**	32.6	**54.0**	**45.9**	**39.7**

TABLE 3
Performance of Different Features on the PASCAL VOC 2007 Dataset

Feature	HOG	LBP	COV	HOG+LBP	HOG+COV	LBP+COV
mAP	35.1	33.5	33.7	38.47	37.3	37.7

aligned inside the detection window for the root filter, allowing local deformation for parts with misplacement costs, while [8] builds spatial pyramids over detection windows. Felzenszwalb et al. [2] enforces a strict spatial constraint in the root filter but allows for small local deformations, while [8] ignores spatial configuration to a large extent. From Table 1, we observe that [2] is excellent at detecting rigid objects (such as buses or cars) or objects with well defined contours, but without abundant global deformations (such as horses or people). In contrast, [8] performs better for objects with significant global deformations such as cats, cows, and sheep, as expected. It is very interesting that [8] significantly outperforms [2] in the aeroplane and tvmonitor categories, both of which seem to be rigid objects. After taking a closer look at the data we found that it is because these categories have very diverse viewpoints, or rich sub-categories.

Our method significantly outperforms the baselines, as we won 16 out of 20 categories. Our approach performs decently for both rigid objects and those objects with local or global deformations. Compared to [8], our edge comes from our regionlet representation encoding of an object's spatial configuration. Compared to [2], our improvement on accuracy is led by the joint deformation and misalignment handling powered by the regionlets' representation with multiple resolution features. If we limit the number of regionlets in a region to one, our method obtains a mean average precision of 36.8 percent. Allowing multiple regionlets consistently improves the object detection accuracy for each class and pushes the number to 41.7 percent, which to our knowledge is the best performance reported on the VOC 2007 dataset without using outside training data.

The DPM+Objectness approach [32] is one of the original approaches applying detectors to object proposals. It achieves 0.6 percent improvement in mAP over the original DPM which reported 26.8 percent mAP. This attempt suggests that the objectness score does not add much to the discriminative power of the DPM detector. Table 3 compares the performance of using different regionlet feature configurations. Fig. 10 shows the average number of regionlets used per region for each class. Deformable object categories such as bird, cat, sheep, person and dog, etc. prefer more regionlets than rigid object classes like bicycle, bus, diningtable, motorbike, sofa and train. An interesting yet consistent phenomenon has been observed for rigid objects like aeroplane and tvmonitor, as in the comparison of [2] and [8]. Our algorithm selects even more regionlets. We speculate the regionlets in these two cases may help to handle misalignment due to multiple viewpoints and sub-categories.

Table 2 shows our detection performance compared with the baselines on the VOC 2010 dataset. Our regionlet approach again achieves the best mean average precision. Table 4 compares our approach with other state-of-the-art methods on both VOC 2007 and VOC 2010 datasets in terms of mean average precision over the 20 categories. While the accuracies of [34] and [41] are close to ours, both of them explore context information. In contrast, our approach does

TABLE 4
Comparison with State of the Arts Using mAP Over 20 Classes

	VOC 2007	VOC 2010	Results year
DPM(WC) [2]	35.4	33.4	2008
UCI_2009 [42]	27.1	N/A	2009
INRIA_2009 [43]	28.9	N/A	2009
NLPR(WC) [16]	N/A	36.8	2010
MITUCLA(WC) [16]	N/A	36.0	2010
UVA [16]	N/A	32.9	2010
MIT_2010 [27]	29.6	N/A	2010
Song et al. (WC) [41]	37.7	36.8	2011
Li et al. (WC) [44]	35.2	N/A	2011
SS_SPM [8]	33.8	34.1	2011
Cinbis et al. (WC) [45]	35.0	N/A	2012
Cinbis et al. (WC) [34]	40.4	38.4	2013
Ours (Regionlets)	**41.7**	**39.7**	2013

"WC" means the method utilizes context cues. We do not use any context information in our method.

TABLE 5
Performance of Regionlets Using DCNN Responses on the PASCAL VOC 2007 Dataset (Average Precision Percentage)

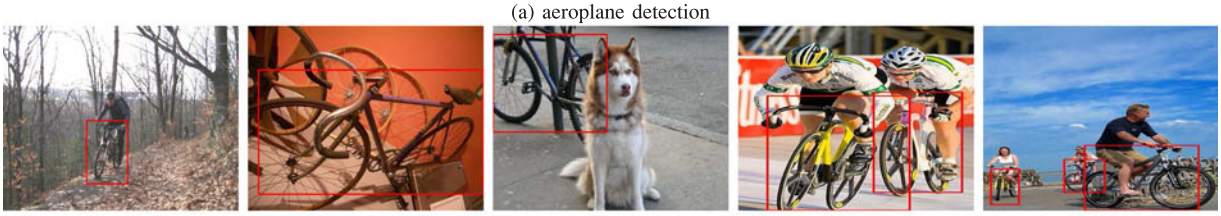
	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
Regionlets-M	54.2	52.0	20.3	24.0	20.1	55.5	68.7	42.6	19.2	44.2	49.1	26.6	57.0	54.5	43.4	16.4	36.6	37.7	59.4	52.3	41.7
Regionlets-CNN $pool_5$	59.3	63.1	35.4	31.8	25.0	56.2	70.4	64.3	21.9	45.6	56.7	47.1	61.6	60.4	51.4	21.4	44.0	49.8	64.7	56.0	49.3
R-CNN $pool_5$ [10]	51.8	60.2	36.4	27.8	23.2	52.8	60.6	49.2	18.3	47.8	44.3	40.8	56.6	58.7	42.4	23.4	46.1	36.7	51.3	55.7	44.2
R-CNN FT fc_7 BB [10]	68.1	72.8	56.8	43.0	36.8	66.3	74.2	67.6	34.4	63.5	54.5	61.2	69.1	68.6	58.7	33.4	62.9	51.1	62.5	64.8	58.5

Our result is directly comparable with the R-CNN approach using pooling five layer features.

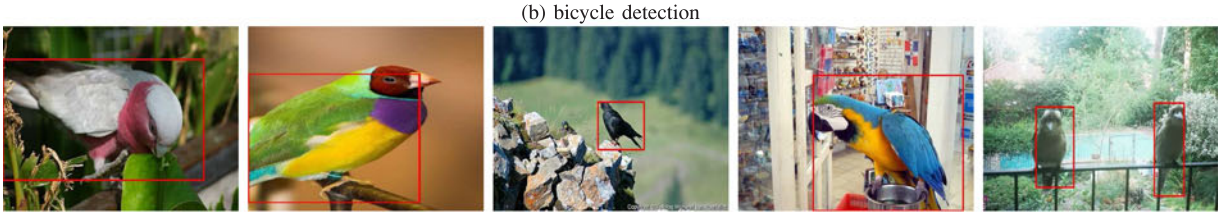
TABLE 6
Performance of Regionlets Using DCNN Responses on the PASCAL VOC 2010 Dataset (Average Precision Percentage)

	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
Regionlets-M	65.0	48.9	25.9	24.6	24.5	56.1	54.5	51.2	17.0	28.9	30.2	35.8	40.2	55.7	43.5	14.3	43.9	32.6	54.0	45.9	39.7
Regionlets-CNN $pool_5$	66.8	55.4	38.1	27.7	24.9	58.3	53.7	66.8	18.9	35.1	35.9	55.8	50.7	63.8	50.9	19.8	46.1	37.5	56.2	46.2	45.5
R-CNN FT fc_7 BB [10]	71.8	65.8	53.0	36.8	35.9	59.7	60.0	69.9	27.9	50.6	41.4	70.0	62.0	69.0	58.1	29.5	59.4	39.3	61.2	52.4	53.7

(a) aeroplane detection



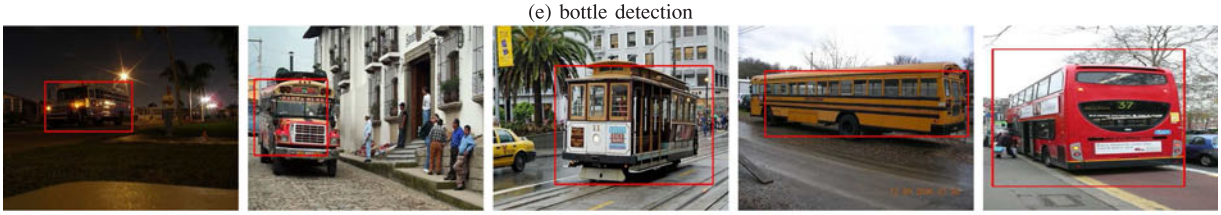
(b) bicycle detection



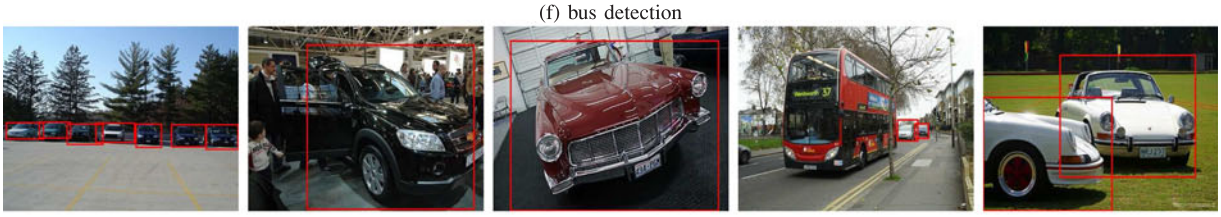
(c) bird detection

(d) boat detection

(e) bottle detection



(f) bus detection



(g) car detection

(h) person detection

Fig. 11. Example detections on the PASCAL VOC 2007 dataset.

TABLE 7
Comparison with the DPM on the ImageNet dataset, Trained on the Training Data and Tested on the Validation Data

	ImageNet validation 2013 (mAP)	Results year
DPM v5 [2]	10.0	2013
Ours (Regionlets)	**16.3**	2013

not utilize any context cues and the context information will likely further benefit our detection approach.

Tables 5 and 6 shows the performance of regionlets using HOG, LBP, Covariance, and DCNN response features. It is reasonable to think that adding DCNN responses to the regionlet feature pool significantly improves the detection performance (49.3 percent mAP on VOC 2007 and 45.5 percent mAP on VOC 2010). This shows that the regionlets are effective at combining diverse types of features. Note that the DCNN is trained on a much larger outside dataset [39] (ImageNet 2010 classification dataset). Our method outperforms the directly comparable R-CNN approach using the pooling-5 layer feature from a DCNN. The R-CNN achieves better performance with the fully-connected layer feature and object location regression. Our approach runs at 0.5 second per image while the R-CNN approach runs at 2 minutes per image. Sample detection results are shown in Fig. 11.

Speed. We conducted our experiments on 12-core Xeon 2.1 GHz blade servers. Multi-threading is utilized to speed up the training and testing procedures. Training for one category on a single machine takes 4 to 8 hours, depending on how difficult the category is. The detection runs at 50 frames per second on a server or five frames per second using a single core if the object region proposals are given. It runs at two frames per second using a single core if computing object proposals from scratch with our implementation. Recently, Cheng et al. [46] show that real-time object proposal generation can be achieved.

When augmenting features from a DCNN with millions of parameters, our detection system still consumes affordable computation. It runs at two frames per second with a multiple thread implementation. This is mainly because 1) our system only needs to run around 10 neural network forward passes to extract their responses for an image, and 2) the support pixel integral image efficiently computes integral images for these neural network responses.

6.2 Experiment on ImageNet Object Detection

Using the training set for training and the validation set for testing, we demonstrate the scalability of the regionlet-based detection on the ImageNet object detection task. With 20 machines (each machine has 8-12 cores), our training finishes in 2.5 days. The testing finishes in roughly 1 hour. The DPM performance is obtained by applying the DPM v5 (version 5) code on ImageNet.[2] The mAP over the 200 object categories is reported in Table 7. Our approach outperforms the latest DPM by 6.3 percent (mAP) across the 200 categories; this is a significant improvement. The objects in the ImageNet dataset have large variations in terms of deformation and sub-categories. The proposed approach tackles both issues by learning the regionlet-based cascaded boosting classifiers, which are capable of depicting multiple modes of spatial layouts and appearance variations for one object category.

2. We thank our intern Miao Sun from University of Missouri for evaluating the DPM performance on the ImageNet.

TABLE 8
Object Detection Performance in the ImageNet 2013 Challenge, Trained on the Training + Validation Data and Tested on the Test Data

	ImageNet test 2013 (mAP)	Results year
UvA-Euvision	22.6	2013
Regionlets+DCNN	20.9	2013
OverFeat-NYU	19.4	2013
Toronto A	11.5	2013
SYSU_Vision	10.4	2013
GPU_UCLA	9.8	2013
Delta	6.1	2013

We augmented the HOG, LBP and Covariance features with the DCNN features to participate in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC 2013). Our system produced a mean average precision of 20.9 percent on 200 object categories. It was ranked the second in all submissions. Results are shown in Table 8. The competitive performance of the regionlet-based detectors validates the effectiveness and scalability of boosting with descriptive competing local regions for generic object detection.

7 Conclusion

In this paper, we propose a regionlet-based approach for generic object detection. Regionlets provide a radically different way to model object deformation compared to existing BoW approaches and DPM approaches. Our regionlet model can adapt itself for detecting rigid objects, objects with small local deformations or large long-range deformations. The regionlet-based framework is able to incorporate various types of features, e.g., dense HOG, LBP, Covariance features and sparse DCNN response features by support pixel integral images. This flexibility enables the potential of optimizing for a specific object class using certain types of features. Validated on the challenging PASCAL VOC datasets and the ImageNet object detection dataset, the proposed regionlet approach demonstrates very competitive performance compared to the existing approaches.

Acknowledgments

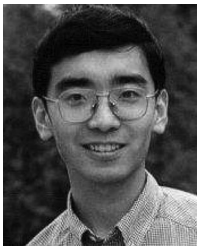
Xiaoyu Wang is the corresponding author.

References

[1] P. Viola and M. Jones, "Robust real-time face detection," *Int. J. Comput. Vis.*, vol. 57, no. 2, pp. 137–154, 2004.
[2] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2008, pp. 1–8.
[3] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2005, vol. 1, pp. 886–893.
[4] M. Pedersoli, A. Vedaldi, and J. Gonzalez, "A coarse-to-fine approach for fast deformable object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2011, pp. 1353–1360.

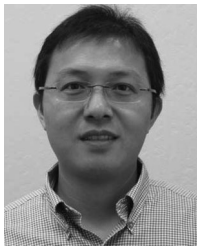
[5] I. Kokkinos, "Rapid deformable object detection using dual-tree branch-and-bound," in *Proc. Adv. Neural Inf. Process. Syst.*, 2011, pp. 2681–2689.
[6] P. Dollár, R. Appel, and W. Kienzle, "Crosstalk cascades for frame-rate pedestrian detection," in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 645–659.
[7] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2006, pp. 2169–2178.
[8] K. E. A. Van de Sande, J. R. R. Uijlings, T. Gevers, and A. W. M. Smeulders, "Segmentation as selective search for object recognition," in *Proc. Int. Conf. Comput. Vis.*, 2011, pp. 1879–1886.
[9] B. Alexe, T. Deselaers, and V. Ferrari, "What is an object?" in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2010, pp. 73–80.
[10] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 580–587.
[11] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[12] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, Q. Le, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Ng, "Large scale distributed deep networks," presented at the Neural Information Processing Systems, Lake Tahoe, NV, USA, 2012.
[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[14] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "OverFeat: Integrated recognition, localization and detection using convolutional networks," in *Proc. Int. Conf. Learn. Representation*, 2014.
[15] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov, "Scalable object detection using deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 2155–2162.
[16] M. Everingham, L. van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. (2010). The PASCAL visual object classes challenge 2010 results [Online]. Available: http://www.pascal-network.org/challenges/VOC/voc2010/workshop/index.html
[17] O. Russakovsky, J. Deng, J. Krause, A. Berg, and L. Fei-Fei. (2013). Large scale visual recognition challenge 2013 (ILSVRC2013) [Online]. Available: http://www.image-net.org/challenges/LSVRC/2013/
[18] C. Huang, H. Ai, B. Wu, and S. Lao, "Boosting nested cascade detector for multi-view face detection," in *Proc. Int. Conf. Patter Recognit.*, 2004, pp. 415–418.
[19] O. Tuzel, F. Porikli, and P. Meer, "Human detection via classification on Riemannian manifolds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2007, pp. 1–8.
[20] X. Wang, T. X. Han, and S. Yan, "An HOG-LBP human detector with partial occlusion handling," in *Proc. Int. Conf. Comput. Vis.*, Sep. 2009, pp. 32–39.
[21] P. Dollár, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: A benchmark," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 304–311.
[22] L. Bourdev, S. Maji, T. Brox, and J. Malik, "Detecting people using mutually consistent poselet activations," in *Proc. Eur. Conf. Comput. Vis.*, 2010, pp. 168–181.
[23] R. Benenson, M. Mathias, R. Timofte, and L. Van Gool, "Pedestrian detection at 100 frames per second," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 2903–2910.
[24] Y. Ding and J. Xiao, "Contextual boost for pedestrian detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 2895–2902.
[25] T. Ahonen, A. Hadid, and M. Pietikinen, "Face recognition with local binary patterns," in *Proc. Eur. Conf. Comput. Vis.*, 2004, pp. 469–481.
[26] P. Felzenszwalb, R. Girshick, and D. McAllester, "Cascade object detection with deformable part models," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2010, pp. 2241–2248.
[27] L. Zhu, Y. Chen, A. Yuille, and W. Freeman, "Latent hierarchical structural learning for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2010, pp. 1062–1069.
[28] C. H. Lampert, "An efficient divide-and-conquer cascade for nonlinear object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2010, pp. 1022–1029.
[29] Z. Zhang, J. Warrell, and P. H. S. Torr, "Proposal generation for object detection using cascaded ranking SVMs," in *pROC. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2011, pp. 1497–1504.
[30] J. Shotton, J. Winn, C. Rother, and A. Criminisi, "TextonBoost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation," in *Proc. Eur. Conf. Comput. Vis.*, 2006, pp. 1–15.
[31] F. Li, J. Carreira, and C. Sminchisescu, "Object recognition as ranking holistic figure-ground hypotheses," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2010, pp. 1712–1719.
[32] B. Alexe, T. Deselaers, and V. Ferrari, "Measuring the objectness of image windows," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 11, pp. 2189–2202, Nov. 2012.
[33] S. Fidler, R. Mottaghi, A. Yuille, and R. Urtasun, "Bottom-up segmentation for top-down detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2013, pp. 3294–3301.
[34] R. Cinbis, J. Verbeek, and C. Schmid, "Segmentation driven object detection with Fisher vectors," in *Proc. Int. Conf. Comput. Vis.*, 2013, pp. 2968–2975.
[35] E. Rahtu, J. Kannala, and M. Blaschko, "Learning a category independent object detection cascade," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2011, pp. 1052–1059.
[36] J. Carreira and C. Sminchisescu, "Constrained parametric min-cuts for automatic object segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2010, pp. 3241–3248.
[37] F. Porikli, "Integral histogram: A fast way to extract histograms in Cartesian spaces," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2005, vol. 1, pp. 829–836.
[38] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, pp. 91–110, 2004.
[39] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "DeCAF: A deep convolutional activation feature for generic visual recognition," *arXiv preprint arXiv:1310.1531*, 2013.
[40] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Mach. Learn.*, vol. 37, pp. 297–336, 1999.
[41] Z. Song, Q. Chen, Z. Huang, Y. Hua, and S. Yan, "Contextualizing object detection and classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2011, pp. 1585–1592.
[42] C. Desai, D. Ramanan, and C. Fowlkes, "Discriminative models for multi-class object layout," in *Proc. IEEE 12th Int. Conf. Comput. Vis.*, 2009, pp. 229–236.
[43] H. Harzallah, F. Jurie, and C. Schmid, "Combining efficient object localization and image classification," in *Proc. IEEE 12th Int. Conf. Comput. Vis.*, 2009, pp. 237–244.
[44] C. Li, D. Parikh, and T. Chen, "Extracting adaptive contextual cues from unlabeled regions," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2011, pp. 511–518.
[45] R. G. Cinbis and S. Sclaroff, "Contextual object detection using set-based classification," in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 43–57.
[46] M.-M. Cheng, Z. Zhang, W.-Y. Lin, and P. H. S. Torr, "BING: Binarized normed gradients for objectness estimation at 300fps," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 3286–3293.

Xiaoyu Wang received the BS degree from the University of Science and Technology of China in 2006. He received the PhD degree in electrical and computer engineering, and the MA degree in statistics from the University of Missouri in 2012. He is a research scientist at NEC Laboratories America. He was a research assistant in electrical engineering and information science in the University of Science and Technology of China. His research interests include computer vision, machine learning, object detection/recognition, large-scale image retrieval, deep learning, and boosting.



Ming Yang received the BE and ME degrees in electronic engineering from Tsinghua University, Beijing, China, in 2001 and 2004, respectively, and the PhD degree in electrical and computer engineering from Northwestern University, Evanston, IL, in June 2008. After graduation, he joined NEC Laboratories America, Cupertino, as a research staff member. He is currently a research scientist in Facebook AI Research since 2013. His research interests include computer vision, machine learning, large-scale image retrieval, face recognition, and intelligent multimedia content analysis.

Shenghuo Zhu received the PhD degree in computer science from the University of Rochester in 2003. He is at Alibaba Group. Prior to Alibaba, he spent 10 years at NEC Laboratories America, and one and half year at Amazon.com, Inc. His primary research interests include machine learning, computer vision, data mining, and information retrieval.



Yuanqing Lin received the PhD degree in electrical engineering from the University of Pennsylvania in 2008. After that, he joined NEC Labs America as a research staff member, working on feature learning and large-scale classification. In 2010, he led the NEC-UIUC team and received the No.1 place in ImageNet Large Scale Visual Recognition Challenge. In April 2012, he became the head of the Media Analytics Department at NEC Labs America. His research interests include machine learning and computer vision.