www.s2is.org
International Journal on Smart Sensing and Intelligent Systems
ISSN 1178-5608

A NOVEL KNOWLEDGE-COMPATIBILITY BENCHMARKER FOR SEMANTIC SEGMENTATION

Vektor Dewanto*‡ , Aprinaldi*, Zulfikar Ian*, Wisnu Jatmiko*

*Faculty of Computer Science, University of Indonesia

‡Department of Computer Science, Bogor Agricultural University

Jawa Barat - Indonesia

Emails: vektor.dewanto@gmail.com, wisnuj@cs.ui.ac.id

Submitted: Jan. 15, 2015 ***Accepted: Mar. 24, 2015*** ***Published: Jun. 1, 2015***

Abstract- The quality of a semantic annotation is typically measured with its averaged class-accuracy value, whose computation requires **scarce** ***ground-truth annotations. We observe that humans accumulate knowledge through their vision and believe that the quality of a semantic annotation is proportionally related to its compatibility with the vision-based knowledge. We propose a knowledge-compatibility benchmarker, whose backbone is a regression machine. It takes as input a semantic annotation and the vision-based knowledge, then outputs an estimate of the corresponding averaged class-accuracy value. The knowledge encodes three kinds of information, namely: co-occurrence statistics, scene properties and relative positions. We introduce three types of feature vectors for regression. Each specifies the characteristics of a probability vector that captures the compatibility between an annotation and each kind of the knowledge. Experiment results show that the Gradient Boosting regression outperforms the ν-Support Vector regression. It achieves best performance at an R^2-score of 0.737 and an MSE of 0.034. This indicates not only that the vision-based knowledge resembles humans' common sense but also that the feature vector for regression is justifiable.***

Index-terms: vision-based knowledge, knowledge-compatibility benchmarker, semantic segmentation, averaged class accuracy, regression

I. Introduction

Semantic segmentation has become a major challenge in computer vision. The goal is to label *every* pixel in an image with a semantic object-class annotation from some pre-defined set. Some works in this area include [1], [2], [3], [4], and [5]. Semantic segmentation is also suited for practical applications, e.g. airport security [6] and embryo segmentation in biomedical imaging [7]. As suggested by [8], the performance of semantic segmentation is typically measured with a performance metric called the class accuracy (CA). It is computed as: "true positives" / ("true positives"+"false positives"+"false negatives"). In particular, the performance of semantic segmentation on one or more images is calculated by averaging the CA over classes to obtain the averaged class-accuracy: averaged-CA. Hence, the computation of averaged-CA requires ground-truth pixel-wise annotations. However, it is so expensive to manually label images that there are relatively a few ground-truth annotation available. For example, two main resources, i.e. MSRC-21 and VOC-2010 datasets, have only 591 and 1928 standard ground-truth annotations, respectively. This obviously becomes a serious impediment to the development of accurate semantic segmentators.

On the other hand, we observe that humans accumulate knowledge through their vision. This happens since there exist patterns (structures) of real-world sceneries. For instance, from a set of 2D images, we can conclude that a keyboard mostly appears along with a mouse, a building is commonly seen above a road and an aeroplane is usually photographed at an airport. As a result, we aim to mimic the construction of such vision-based knowledge from ground-truth semantic annotations. The knowledge encodes three types of information, namely: co-occurrence statistics, relative locations and scene properties.

We believe that the quality of a semantic annotation is proportionally related to its compatibility with the vision-based knowledge. This key observation leads us to formulate a supervised regression problem. Concretely, we propose a knowledge-compatibility benchmarker, whose backbone is a regression machine. It takes as input a semantic annotation and the vision-based knowledge, then outputs a knowledge-compatibility score, which is essentially an estimate of the averaged class-accuracy value.

In addition to minimizing the need of ground-truth semantic annotations in computing the performance metric, a knowledge-compatibility score can be employed during a semantic segmentation process. In MRF-based semantic segmentation, one possible way is to calculate a knowledge-compatibility score of a temporary annotation for each inference iteration. The

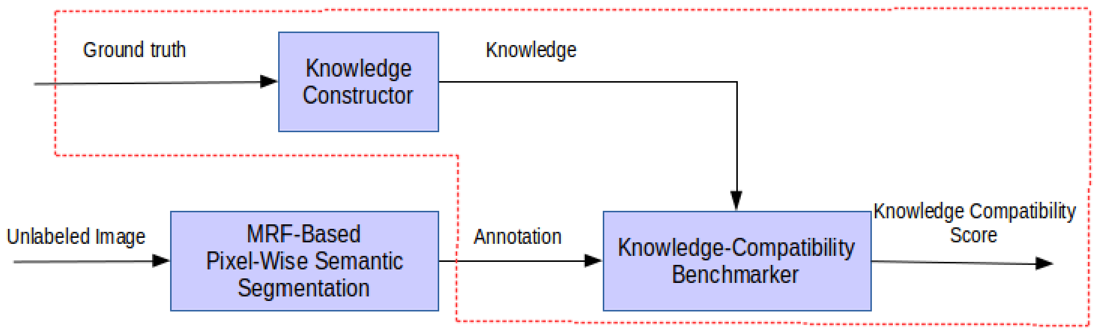

Ground truth
Knowledge Constructor
Knowledge
Unlabeled Image
MRF-Based Pixel-Wise Semantic Segmentation
Annotation
Knowledge-Compatibility Benchmarker
Knowledge Compatibility Score

Figure 1: The block-diagram of our proposed knowledge-compatibility benchmarker.

score (after some scaling and normalization) is then taken into account as feedback in the next energy potential calculation. This technique is in synergy with the fact that humans enhance their vision performance using the knowledge they acquired by seeing at all times.

We construct the vision-based knowledge from ground-truth semantic annotations of the Pascal VOC-2010 dataset [8]. Experiment results show that the constructed knowledge has a relatively high agreement on a set of human-centric queries about some extreme cases of the knowledge. This suggests that it generally resembles the humans' common-sense. Furthermore, we introduce three kinds of feature vectors for regression so as to capture the compatibility between a semantic annotation and each type of the knowledge. Each kind substantially specifies the characteristics of a probability vector that is derived from the given annotation and the knowledge. We found that as the core of our proposed knowledge-compatibility benchmarker, the Gradient Boosting regression outperforms the ν-Support Vector regression. The former achieves best performance at an R^2-score of 0.737 and an MSE of 0.034, while the latter's is at an R^2-score of 0.697 and an MSE of 0.038. This indicates not only that the vision-based knowledge is well constructed but also that the feature vector for regression is justifiable. Figure 1 shows the block diagram of our proposed knowledge-compatibility benchmarker. Our implementation code is publicly available at `https://github.com/tttor/lab1231-sun-prj`.

II. Related Work

We review several works that attempt to construct knowledge from images. In particular, we are insterested in certain aspects, such as: the kinds of knowledge, the construction methods, as well as the knowledge evaluation. We also emphasize on the utilization of the constructed knowledge, specifically, for MRF-based semantic segmentation.

Regarding co-occurrence statistics, the work in [9] argues that a co-occurrence relationship between classes can give a more semantic information. For instance, a cow is not likely to appear with dining table, resulting a more accurate labelling due to richer information. This work contributes an MRF-based semantic segmentation model that incorporates co-occurrence statistics as global information in order to impose semantic constraints among object categories. Moreover, the work in [10] establishes semantic context information that takes into account the co-occurrence frequency among object labels in the training set of the database. It uses a fully connected MRF over segments, in which the co-occurrence value is used as pairwise potential in the energy formula.

For spatial layout information, the work in [11] presents a region based model about scene geometry, where a scene is decomposed into vertical and horizontal categories. Its main contributions include an inference technique that utilize appearance and scene geometry jointly for optimizing energy function. A three dimensional spatial layout representation in [12] also utilizes geometrical information to enhance the accuracy in contrast of only labelling objects. It represents a subclass of geometric information in three dimensional sense.

Meanwhile, for the work related with relative position, the work in [13] uses a relation such as car on street on top of binary relation of co-occurrence between car and street. Besides, the work in [14] proposes a method for capturing global information from inter-class spatial relationships and encoding it as a local feature. It tries to identify the fact that, for example, relative to "tree" pixels, pixels above and to the sides are more likely to be "sky" whereas pixels below are more likely to be "grass". Pertaining to the scene-property knowledge, the work in [15] presents an empirical analysis of the role of context by estimating the likelihood of observing an object given a scene image. It analyzes the role of context by estimating the likelihood of observing an object given a scene image on a standardized dataset.

Furthermore, the work in [16] employs two kinds of knowledge, i.e. object co-occurrences and spatial relationships. It proposes an efficient model that captures the contextual information among more than a hundred of object categories. The work in [17] proposes a visual concept ontology composed of several types of concepts (spatial concepts and relations, color concepts and texture concepts). The ontology knowledge is used to perform complex object categorization. Moreover, work in [18] incorporates constraints including co-occurrence and spatial layout into an MRF inference framework for solving label set classification. The work in [19] introduces Hierarchy and Exclusion (HEX) graphs to overcome limitations from per-pixel classifier. It formalizes acquisition of semantic relations between any two labels applied to the same object: mutual exclusion, overlap and subsumption.

III. The vision-based knowledge construction

We aim to mimic how humans obtain some knowledge through their vision. In particular, the vision-based knowledge encodes three kinds of information of 2D images, namely: 1) co-occurrence statistics, 2) scene properties and 3) relative locations. We argue that those three kinds of knowledge are the most essential. They are summarized from the literature, including [10], [9], [15], [16] and [14]. We have tried to include another type, such as the spatial-layout knowledge. It encodes the fact that, for example, the sky usually presents in the top of an image, while roads are normally at the bottom. However, we found that it generally does not yield meaningful and significant clues.

1. Co-occurrence statistics

The co-occurrence statistics encodes the likelihood of two objects appear together in a 2D image. Take for example, based on their experience, humans can reason that a table is likely occur together with a chair. Likewise, given a car in an image having several objects, we are more sure that the others include a person than a horse. This means that the co-occurrence statistics knowledge discourages strange object combinations. Moreover, if one or more pairs of objects are already known to be together with high confidence, then the appearance of some object may be suppresses as they are unexpected.

This knowledge is represented by an n-by-n symmetrical matrix C, where n is the number of the pre-specified object classes in a dataset. Let i and j, where $i \neq j$, denote two different object classes. Then, the element $C[i, j] = C[j, i]$, where $i \neq j$, indicates the number of co-occurrence of i and j in the dataset. We do not take into account whether an occurrence of an object class comes from one or more separate objects. This implies that an element $C[i, i]$ indicates the number of occurrence of i in the dataset, *not* the co-occurrence between an object class with itself. In other words, there is no information about the co-occurrence of two or more objects with the same class. For instance, we do not know how likely an object Person appears together with another object Person in an image. Our co-occurrence knowledge constructor takes as input ground-truth pixel-wise semantic annotation and a list of pre-specified object clasess. Listing 1 shows our implementation in Python, where the matrix C is built using a dictionary data structure.

Listing 1: Our implementation of cooccurrence knowledge constructor in Python.

```python
def construct(ann_ids, obj_names, ann_dir):
    cooccurrence = dict.fromkeys(obj_names, None)
    for key in cooccurrence.iterkeys():
        cooccurrence[key] = dict.fromkeys(obj_names, 0)

    for i,ann_id in enumerate(ann_ids):
        print 'Processing',i+1,'of',len(ann_ids),'ann_id=',ann_id

        ann_filepath = ann_dir+'/'+ann_id+'.csv'
        ann = np.genfromtxt(ann_filepath, delimiter=',')

        obj_ids = list(set( ann.flatten().tolist() ))
        objs = voc.translate(obj_ids)
        objs = [i for i in objs if i not in voc.ignored_class_name_list]

        for i in objs:
            cooccurrence[i][i] = cooccurrence[i][i] + 1
            for j in objs:
                if i is not j:
                    cooccurrence[i][j] = cooccurrence[i][j] + 1
    return cooccurrence
```

2. Scene properties

This kind of knowledge defines a has-a property of a scene class of a 2D image. For instance, an image of highways typically contains (has-a) roads, cars and road signs. Meanwhile, an image of public parks contains people, benches, trees, etc. The scene-property knowledge essentially encodes the context of a scene in an image.

In this work, all scene classes are place-centric, such as airports, bedrooms and forests. We do not consider scenes like playing badminton, jogging, sunset, etc. Furthermore, we assume that an image is already tagged with its scene class. The work of Oliva et al in [20] demonstrates a scene-class classification, where GIST global features are extracted out of an image then fed to an SVM classifier.

Let S denote an m-by-n matrix that encodes the scene-property knowledge, where m and n are the number of valid scene-classes and the number of pre-specified object classes of a dataset, respectively. Then, an element $S[i, j]$ indicates the frequency of a scene class i contains an object class j. Listing 2 shows our implementation in Python, where the matrix S is built using a dictionary data structure.

Listing 2: Our implementation of scene-property knowledge constructor in Python.

```python
def construct(img_ids, scepe_prop_ann_dir):
    ann_filepaths = [scepe_prop_ann_dir+'/'+img_id+'.xml' for img_id in img_ids]

    scene_prop = {}
    for i, img_id in enumerate(img_ids):
        print 'Processing',i+1,'of',len(img_ids),'id=',img_id

        ann_filepath = scepe_prop_ann_dir+'/'+img_id+'.xml'
        root = etree.parse(ann_filepath).getroot()
        for place_sub in root.findall('place'):
            place = place_sub.get('name').lower()
            objs = [obj_sub.get('name').lower() for obj_sub \
                                                in place_sub.findall('object')]

            if place in scene_prop.keys():
                obj_count_dict = scene_prop[place]
                for obj in objs:
                    if obj in obj_count_dict.keys():
                        obj_count_dict[obj] = obj_count_dict[obj] + 1
                    else:
                        obj_count_dict[obj] = 1
            else:
                obj_count_dict = dict.fromkeys(objs, 1)
                scene_prop[place] = obj_count_dict

    return scene_prop
```

3. Relative locations

The relative-location knowledge encodes the likelihood of relativeness relationships between two objects in a 2D image. For example, the sky and road are is likely *above* and *below* buildings, respectively. Likewise, as a cow is commonly surrounded by grass, the knowledge should suggest that grass is *around* a cow.

To this end, we follow the work of Gould et al in [14]. Let R_{st} denote an n-by-n matrix that encodes the relative location probability of object s with respect to object t, where n is the number of pixels used to quantize an image. For a pixel p' whose label is t, an element $R_{st}[i,j]$ contains the probability that a pixel p at offset (i,j) from p' has class label s. In order to have a proper conditional probability distribution over labels s, we impose that $\sum_{s=1}^{K} R_{st}(i,j) = 1$, where K is the number of pre-specified object classes O. Eventually, we have $K \times K$ number of R_{st} matrices, i.e. K matrices for each $t \in O$. Listing 3 shows our implementation in Python for constructing one R_{st} matrix.

Listing 3: Our implementation of relative-location knowledge constructor in Python.

```python
def construct(chosen_cprime, img_list_filepath, gt_csv_dir, img_dir):
    #
    relative_location_matrix_shape = (200,200) #following [Gould, 2008]
    variance_factor = 0.10 #following [Gould, 2008]
    dirichlet_noise = False
    dirichlet_noise_alpha = (5.0,5.0) #following [Gould, 2008]

    #
    c_labels = [{'id':key, 'name':val} for key,val \
                in dataset.class_id2name_map.iteritems() \
                if val not in dataset.ignored_class_name_list]
    cprime_labels = c_labels
    if chosen_cprime is not 'all':
        cprime_labels = [{'id':key, 'name':val} for key,val \
                         in dataset.class_id2name_map.iteritems() \
                         if val==chosen_cprime]

    prob_map = init_prob_map([i['name'] for i in cprime_labels], [i['name'] \
                            for i in c_labels], relative_location_matrix_shape)
    with open(img_list_filepath) as f:
        img_ids = f.readlines()
    img_ids = [x.strip('\n') for x in img_ids]

    for i, img_id in enumerate(img_ids):
        img_filepath = img_dir + '/' + img_id + dataset.ori_img_format
        img = img_as_float(io.imread(img_filepath))
        img_height, img_width = img.shape

        segmentation = get_segmentation(img)
        segment_list = get_segment_list(segmentation)

        gt_ann_filepath = gt_csv_dir + '/' + img_id + '.csv'
        gt_annotation = np.genfromtxt(gt_ann_filepath, delimiter=',')

        for j, segment in enumerate(segment_list):
            centroid = get_centroid(segment)
            centroid_label = get_label(centroid, gt_annotation)
            centroid_weight = get_weight(segment)

            if centroid_label['name'] in dataset.ignored_class_name_list:
                continue

            if centroid_label['name'] not in prob_map.keys():
                continue

            for label in c_labels:
                pixels = get_pixel_of_label(label, gt_annotation)
                for pixel in pixels:
                    offset = get_offset(centroid,pixel)
                    if dirichlet_noise=='True':
                        offset = add_dirichlet_noise(offset,dirichlet_noise_alpha,\
                                                     relative_location_matrix_shape)
                    norm_offset = normalize_offset(offset, \
                                                   relative_location_matrix_shape,\
                                                   gt_annotation.shape)

                    idx=get_prob_map_idx(norm_offset,relative_location_matrix_shape)
                    count = prob_map[centroid_label['name']][label['name']] [idx]
                    count = count + centroid_weight

                    prob_map[centroid_label['name']][label['name']] [idx] = count

    norm_prob_map = normalize_prob_map(prob_map, relative_location_matrix_shape)
    sigma = (np.sqrt(variance_factor*img_height),\
             np.sqrt(variance_factor*img_width))
    filtered_norm_prob_map = apply_gaussian_filter(sigma, norm_prob_map, \
                                                   relative_location_matrix_shape)
    return filtered_norm_prob_map
```

IV. The knowledge-compatibility benchmarker

We aim to predict the quality of a (predicted) semantic annotation, which is equivalent to the performance of semantic segmentation. We observe that such quality is proportionally related to the compatibility between an annotation and the vision-based knowledge: the higher the compatibility, the better the quality. As suggested by [8], for a single object class, the quality of a semantic annotation is typically measured with a quality metric called the class accuracy (CA). It is computed as: "true positives" / ("true positives"+"false positives"+"false negatives"). In particular, for a semantic annotation with several object classes, its quality is calculated by averaging the CA over classes to obtain the averaged-CA.

The aforementioned observation leads us to formulate a supervised regression problem. Specifically, given a semantic annotation and the vision-based knowledge, a regression machine is utilized to predict the averaged-CA value, refer to section III for the knowledge construction. The benchmarker's backbone is a regression function f that maps a semantic annotation and the vision-based knowledge to the corresponding averaged-CA value. That is $f : \mathbf{x} \mapsto y$, where $\mathbf{x} = \phi(annotation, knowledge)$ is a feature vector extracted from a semantic annotation and knowledge, while $y \in \mathbb{R}$ is the averaged-CA value. In testing phase, the estimate of an averaged-CA value is considered as a knowledge-compatibility score.

There are, at least, two obvious benefits of a knowledge-compatibility benchmarker. First, it can be employed to approximately measure the quality of a semantic annotation (therefore, semantic segmentation) whenever its ground-truth annotation is not available. Secondly, a knowledge-compatibility score can be utilized during a semantic segmentation process. In CRF-based semantic segmentation, one possible way is to calculate a knowledge-compatibility score of an annotation for each inference iteration. The score (after some scaling and normalization) is then taken into account as feedback in the energy potential for the next iteration. This technique is in synergy with the fact that humans enhance their vision performance by the knowledge they acquire and accumulate by seeing the world at all times.

1. Feature extraction

The feature extractor function ϕ takes as input a semantic annotation and the vision-based knowledge. In order to capture the compatibility between a semantic annotation and each type of the knowledge, we introduce three kinds of feature vectors. Each of those specifies the characteristics of a probability vector that is derived from the given annotation and the knowledge. Hence, we have ϕ_c, ϕ_s, and ϕ_r that output co-occurrence statistics features $\mathbf{x}_c \in \mathbb{R}^4$, scene-property features $\mathbf{x_s} \in \mathbb{R}^4$ and relative location features $\mathbf{x_r} \in \mathbb{R}^4$, respectively. The final feature vector is a concatenation of those row feature vectors, i.e. $\mathbf{x} = [\mathbf{x}_c \; \mathbf{x_s} \; \mathbf{x_r}]^T, \mathbf{x} \in \mathbb{R}^{4 \times 3}$.

1.1. Co-occurrence statistics features: $\mathbf{x}_c$

Let C_n denote a normalized co-occurrence statistics matrix. It is calculated as $C_n[i,j]_{i \neq j} = C[i,j]_{i \neq j}/n_j$, where n_j is the number of occurrence of object j in a dataset, so that we have $C_n[i,j]_{i \neq j} \in [0,1]$. Given a semantic annotation and the vision-based knowledge, the co-occurrence feature extractor ϕ_c first constructs a co-occurrence probability vector $\mathbf{p}^c = [p_1^c \ p_2^c \ \cdots \ p_m^c]$, where $m = {}^2\mathbb{C}_n$ is the number of combinations of 2 from n objects contained in the given semantic annotation. A scalar member of $\mathbf{p}^c$, i.e. $p_k^c = C_n[i,j]$ is a probability that two objects i and j appear together in a semantic annotation according to the knowledge.

Afterward, the extractor does extract four characteristics of the probability vector $\mathbf{p}^c$, namely 1) the mean μ^c, 2) the first quartile q_1^c, 3) the second quartile q_2^c and 4) the third quartile q_3^c, refer to listing 4 for this. Finally, $\mathbf{x}_c = [\mu^c \ q_1^c \ q_2^c \ q_3^c]$, as shown in listing 5.

Listing 4: Our implementation for obtaining the characteristics of a probability vector **p**

```python
def get_prob_list_representation(prob_list):
    assert len(prob_list)>0, 'len(prob_list)==0'

    prob_series = np.asarray(prob_list)
    rep = []

    # mean
    rep.append( np.mean(prob_series) )

    # quartile
    for i in [25,50,75]:
        rep.append( np.percentile(prob_series,i) )
    return rep
```

Listing 5: Our implementation of co-occurrence feature extractor in Python.

```python
def extract_cooccurrence_fea(ann, knowledge):
    numeric_classes = list( set(ann['ann'].flatten()) )
    classes = [i for i in voc.translate(numeric_classes) \
               if i not in voc.ignored_class_name_list]

    prob_list = [0.0]
    if len(classes) > 1:
        prob_list = [knowledge[i][j] for i,j \
                     in itertools.combinations(classes,2)]

    fea = fxu.get_prob_list_representation(prob_list)
    return fea
```

1.2. Scene-property features: $\mathbf{x}_s$

Given a semantic annotation and the vision-based knowledge, a scene-property feature extractor ϕ_s begins with classifying the scene class of the given annotation. A scene-class classifier, typically, takes as input an original image, of which some global features can be extracted. For work on scene-class classification, we refer the reader to [20].

Having the estimated scene class, the extractor ϕ_s proceeds to form a scene-property probability vector $\mathbf{p}^s = [p^s_1\ p^s_2\ \dots\ p^s_n]$, where n is the number of pre-specified object classes in a dataset. Afterward, similiar to ϕ_c, it obtains four characteristics of $\mathbf{p}^s$, see also listing 4. Thus, $\mathbf{x}_s = [\mu^s\ q^s_1\ q^s_2\ q^s_3]$, as shown in listing 6.

Listing 6: Our implementation of scene-property feature extractor in Python.

```python
def extract_sceneprop_fea(ann, knowledge):
    scene_class = fxu.get_scene_class(ann['filename']).lower()

    numeric_classes = list( set(ann['ann'].flatten()) )
    classes = [i for i in voc.translate(numeric_classes) \
               if i not in voc.ignored_class_name_list]

    prob_list = [0.0]
    if len(classes) > 1:
        prob_list = [knowledge[scene_class][obj] \
                     if (obj in knowledge[scene_class].keys()) \
                     else 0.0 for obj in classes]

    fea = fxu.get_prob_list_representation(prob_list)
    return fea
```

1.3. Relative-location features: $\mathbf{x_r}$

Similiar to ϕ_c and ϕ_s, the extractor of relative-location features ϕ_r takes as input an semantic annotation and the vision-based knowledge. First, it performs standard (non-semantic) segmentation on the original image in order to obtain a set segments or superpixels $S = \{S_i\}_{i=1}^N$. The second step essentially follows [14]. For each superpixel S_i, it creates two types of votes for each class of pre-defined object classes in a dataset. Based on the relative-location knowledge, each superpixel casts a vote for where it would expect to find pixels of every class (including its own class) given its location (superpixel centroid) and predicted label. This means that each superpixel S_i receives $N-1$ votes from all the other superpixels S_j. The two votes of S_i, namely: "self" votes $v^{self}(S_i)$ and "other" votes $v^{other}(S_i)$, are used to aggregate the votes from class $c_j = c_i$ and $c_j \neq c_i$ separately. Here, c_i is the semantic segmentation at the centroid pixel of a superpixel S_i.

In the third step, this feature extractor ϕ_r builds a pseudo probability vector $\mathbf{p}^r = [p^r_1\ p^r_2\ \dots\ p^r_n]$, where $p^r_i = log\ v^{other}_{c_j}(S_j) + log\ v^{self}_{c_j}(S_j)$, following [14], and $n = N \times K$ with K is the number of pre-defined object classes. It is a pseudo probability vector in that its element p^r_i does not need to be in the range of $[0,1]$. Finally, a representation of $\mathbf{p}^r$ is extracted so that we have $\mathbf{x_r} = [\mu^r\ q^r_1\ q^r_2\ q^r_3]$. Listing 7 shows our implementation of ϕ_r in Python.

Listing 7: Our implementation of relative-location feature extractor in Python.

```python
def extract_relloc_fea(ann, knowledge):
    # do (standard) segmentation
    ori_img_filepath = ann['ori_img_dir']+'/'+ann['filename']+ann['ori_img_ext']
    img = img_as_float(io.imread(ori_img_filepath))

    segmentation = rlk.get_segmentation(img)
    segment_list = rlk.get_segment_list(segmentation)
    class_list = knowledge.keys()

    # compute relative location votes
    init_vote = dict.fromkeys(range(len(segment_list)), None)
    for key in init_vote.iterkeys():
        init_vote[key] = dict.fromkeys(class_list, 0.0)

    vote_other = vote_self = init_vote
    for i, si in enumerate(segment_list):
        xi, yi = rlk.get_centroid(si)
        ci_hat = voc.class_id2name_map[ ann['ann'][xi,yi] ]
        if ci_hat in voc.ignored_class_name_list:
            continue

        for c in class_list:
            if c in voc.ignored_class_name_list:
                continue

            vote_other[i][c] = vote_self[i][c] = 0.0
            for j in range(len(segment_list)):
                if j == i:
                    continue
                sj = segment_list[j]
                P_cjhat_sj = 1.0 # the probability of a superpixel sj has a class label cjhat
                alpha_j = P_cjhat_sj * len(sj)

                xj, yj = rlk.get_centroid(sj)
                cj_hat = voc.class_id2name_map[ ann['ann'][xj,yj] ]
                if cj_hat in voc.ignored_class_name_list:
                    continue

                offset = rlk.get_offset((xi,yi),(xj,yj))
                norm_offset = rlk.normalize_offset(offset, (200,200), \
                                                   ann['ann'].shape)

                prob_map = knowledge[c][cj_hat]
                idx_x, idx_y = rlk.get_prob_map_idx(norm_offset, prob_map.shape)

                if ci_hat==cj_hat:
                    vote_self[i][c] = vote_self[i][c] \
                                      + (alpha_j*prob_map[idx_x][idx_y])
                else:
                    vote_other[i][c] = vote_other[i][c] \
                                       + (alpha_j*prob_map[idx_x][idx_y])

    # Compute the relloc features
    w_other = 1.0; w_self = 1.0; epsilon = 0.00001

    prob_list = []
    for s in range(len(segment_list)):
        for c in class_list:
            f_relloc = (w_other*math.log(vote_other[s][c]+epsilon)) \
                     + (w_self*math.log(vote_self[s][c]+epsilon))
            prob_list.append(f_relloc)

    # Get the representation of pseudo prob map
    fea = fxu.get_prob_list_representation(prob_list)
    return fea
```

2. Regression methods

Given a training set $D_{tr} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n_{tr}}$ with inputs $\mathbf{x}_i \in \mathbb{R}^{12}$ and outputs $y_i \in \mathbb{R}$ of n_{tr} independent samples from some underlying joint distribution, the goal of regression is to find a function $f^*(\mathbf{x})$ that maps $\mathbf{x}$ to y, such that the expected value of a loss function $E_{\mathbf{x},y}[\Psi(y, f(\mathbf{x}))]$ is minimized. Since the expected loss is commonly approximated by its empirical estimate, the regression problem becomes: $f^*(\mathbf{x}) = \text{argmin}_{f(\mathbf{x})} \sum_{i=1}^{n_{tr}} \Psi(y_i, f(\mathbf{x}_i))$. To this end, we apply two regression methods for the knowledge-compatibility benchmarker, namely: the ν-SVR (Support Vector Regression) [21] and the Gradient Boosting Regression (GBR) [22].

2.1. ν-Support Vector Regression

The nu-SVR [21] is an extension of Support Vector Machines (SVMs). It attempts to minimize the generalization error bound so as to achieve generalized performance. The idea of SVR is based on the computation of a linear regression function in a high dimensional feature space where the input data are mapped via a nonlinear function. In general, the superiority of SVMs includes 1) effectiveness in high dimensional feature vectors, even when the number of feature dimensions is greater than the number of samples, 2) the quadratic optimisation problem is convex, therefore it gives the globally optimal value, while Neural Networks yield multiple solutions associated with local minima, 3) being memory-efficient since it uses a subset of training points in the decision function (called support vectors), 4) robustness against training samples that have some bias, due to a good out-of-sample generalization by choosing the right parameters C in the case of a Gaussian kernel and 5) versatility in the sense that different kernel functions can be specified for the decision function.

Specifically, several advantages of SVR over artificial neural networks (ANNs) and Partial Least Squares (PLS) according to [23] are as follows:

- ε-insensitive loss function: In order to prevent the noise of the training set from influencing the mathematical model, any residue of regression less than some small value ε is considered to be meaningless. This means that SVR can suppress the overfitting due to noisy training samples.
- the principle of flatness: the norm of weight vector is minimized in order to prevent the magnification of the error of the training sample.
- application of kernel functions: this makes nonlinear data sets can be treated by linear

algorithms.

- the principle of data dependent structure risk minimization: hence, the relation between the target function and the data set is utilized. Moreover, this principle makes SVRs do not depend on the dimension of data set. For comparison, ANNs merely apply the principle of empirical risk minimization. Consequently, they are much dependent to the dimension of data sets and tend to overfit whenever the dimension is high.

2.2. Gradient Boosting Regression

In addition to SVR, we also utilize the Gradient Boosting Regression (GBR) [22, 24]. It is based on the idea of creating a highly accurate prediction rule by combining many relatively weak and inaccurate rules. As stated in [25], it has shown excellent accuracy and generalization in various real-world practical practical applications. Its benefits include: 1) highly customizable to the particular needs of the application, i.e. supports different loss functions, 2) automatically detects (non-linear) feature interactions, 3) accomodate features measured on different scale, and 4) able to effectively capture complex non-linear function dependencies. As a result, we believe that the GBR is competitive in comparison with ν-SVR.

V. Experiment Results

There are two main experiments, namely: the vision-based knowledge construction and the knowledge-compatibility benchmarker. For both, we use the Pascal VOC 2010 object-class segmentation dataset [8]. It has 1928 ground-truth semantic annotations of 20 pre-specified object classes, whose ground-truth color-map is depicted in figure 2. Some original images and ground-truth semantic annotations are shown in figure 9, 10 and 11.

1. The vision-based knowledge construction

This experiment aims to construct vision-based knowledge by encoding information of ground-truth 2D semantic annotations. We run our knowledge constructor using all 1928 ground-truth annotations from the Pascal VOC 2010 dataset. We remark that for building scene-property knowledge, we manually label each original image with its common-sense scene class. In total, there are 27 place-centric scene-classes.

The evaluation of the vision-based knowledge is carried out with respect to two aspects.

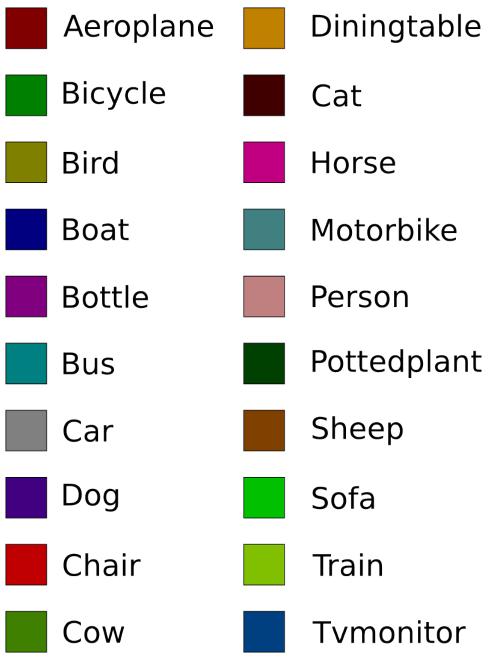
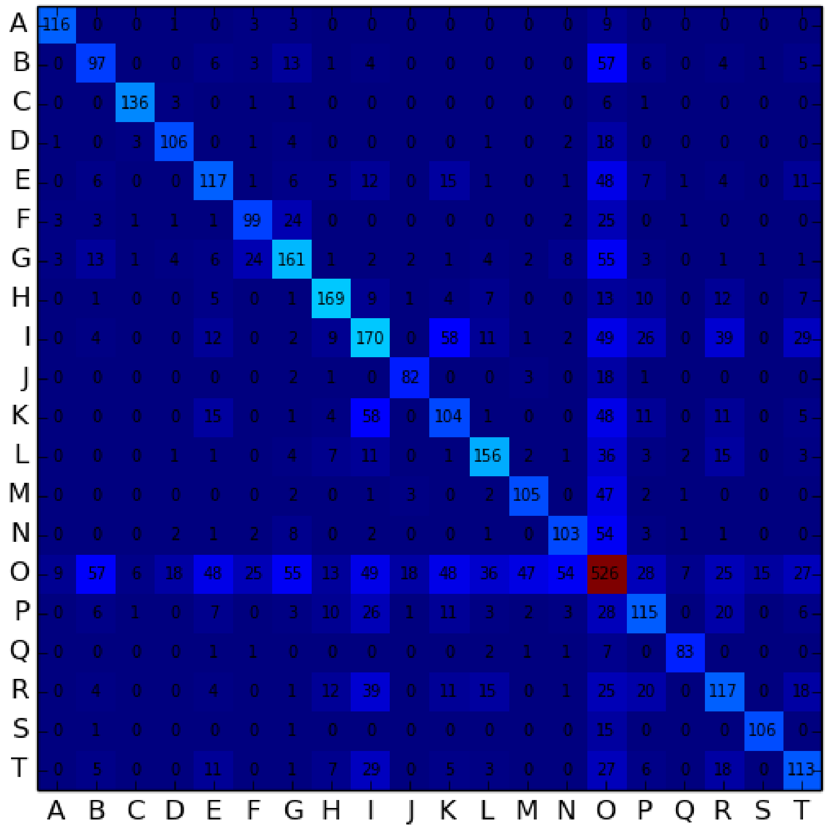


Figure 2: The color-map of 20 pre-specified object classes of Pascal VOC-2010 [8].

First, we investigate the agreement between the vision-based knowledge and the typical human knowledge on some extreme points. For example, “what is the most/least frequent object class that occurs together with bicycles?”, “what is the most/least probable object class on top of a chair?” and “what is the most/least probable object class that appears in the scene-class of airports?”. This, however, requires an assumption that the given ground-truth annotations represent, to some extent, what humans perceive daily. Secondly, the vision-based knowledge utility can be measured by the performance of the knowledge-compatibility benchmarker. The benchmarker heavily relies on the knowledge for extracting input features for regression, see section IV. As a result, if the benchmarker can produce knowledge-compatibility scores that accurately estimate averaged-CA values, then we can conclude that the vision-based knowledge is also of high quality. We postpone the evaluation based on the second aspect until subsection 2 of section V.

1.1. Co-occurrence statistics knowledge

Figure 3 visualizes the co-occurrence statistics knowledge, i.e. the symmetrical matrix C. Notice that $C[i][j]$ indicates the co-occurrence frequency of two objects i and j if $i \neq j$ and the occurrence of object i in the dataset if $i = j$. As can be seen, the object-class Person appears most frequently in the dataset; in 526 out of 1928 images. Its highest co-occurrence is with object-class Bicycle, while its lowest co-occurrence is with object-class Bird. Meanwhile, the



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
A	116	0	0	1	0	3	3	0	0	0	0	0	0	0	9	0	0	0	0	0
B	0	97	0	0	6	3	13	1	4	0	0	0	0	0	57	6	0	4	1	5
C	0	0	136	3	0	1	1	0	0	0	0	0	0	0	6	1	0	0	0	0
D	1	0	3	106	0	1	4	0	0	0	0	1	0	2	18	0	0	0	0	0
E	0	6	0	0	117	1	6	5	12	0	15	1	0	1	48	7	1	4	0	11
F	3	3	1	1	1	99	24	0	0	0	0	0	0	2	25	0	1	0	0	0
G	3	13	1	4	6	24	161	1	2	2	1	4	2	8	55	3	0	1	1	1
H	0	1	0	0	5	0	1	169	9	1	4	7	0	0	13	10	0	12	0	7
I	0	4	0	0	12	0	2	9	170	0	58	11	1	2	49	26	0	39	0	29
J	0	0	0	0	0	0	2	1	0	82	0	0	3	0	18	1	0	0	0	0
K	0	0	0	0	15	0	1	4	58	0	104	1	0	0	48	11	0	11	0	5
L	0	0	0	1	1	0	4	7	11	0	1	156	2	1	36	3	2	15	0	3
M	0	0	0	0	0	0	2	0	1	3	0	2	105	0	47	2	1	0	0	0
N	0	0	0	2	1	2	8	0	2	0	0	1	0	103	54	3	1	1	0	0
O	9	57	6	18	48	25	55	13	49	18	48	36	47	54	526	28	7	25	15	27
P	0	6	1	0	7	0	3	10	26	1	11	3	2	3	28	115	0	20	0	6
Q	0	0	0	0	1	1	0	0	0	0	0	2	1	1	7	0	83	0	0	0
R	0	4	0	0	4	0	1	12	39	0	11	15	0	1	25	20	0	117	0	18
S	0	1	0	0	0	0	1	0	0	0	0	0	0	0	15	0	0	0	106	0
T	0	5	0	0	11	0	1	7	29	0	5	3	0	0	27	6	0	18	0	113

Figure 3: The co-occurrence statistics knowledge, constructed from 1928 ground-truth semantic annotations. This is the visualization of a matrix C, where the number in each box is the value of $C[i][j]$. LEGEND: A=aeroplane, B=bicycle, C=bird, D=boat, E=bottle, F=bus, G=car, H=cat, I=chair, J=cow, K=diningtable, L=dog, M=horse, N=motorbike, O=person, P=pottedplant, Q=sheep, R=sofa, S=train, T=tv/monitor

object-class Chair mostly co-locates with object-class Diningtable. In general, observation on the resulted co-occurrance matrix suggests that it largely agrees with humans' common sense.

1.2. Scene-property knowledge

Figure 4 visualizes the scene-property knowledge, where edge costs are taken from a normalized scene-property matrix S_n. For example, in the scene-class of airports, the knowledge obviously advises that Aeroplane is more likely to present than Person. Meanwhile, Person dominates in the scene-class of parks, restaurants, even roads. Another interesting finding is that Boat has the highest probability in the scene of lakes, while Diningtable and Chair are two most likely objects in the scene-class of dining-room. Intuitively, the likelihood of Car, Bus

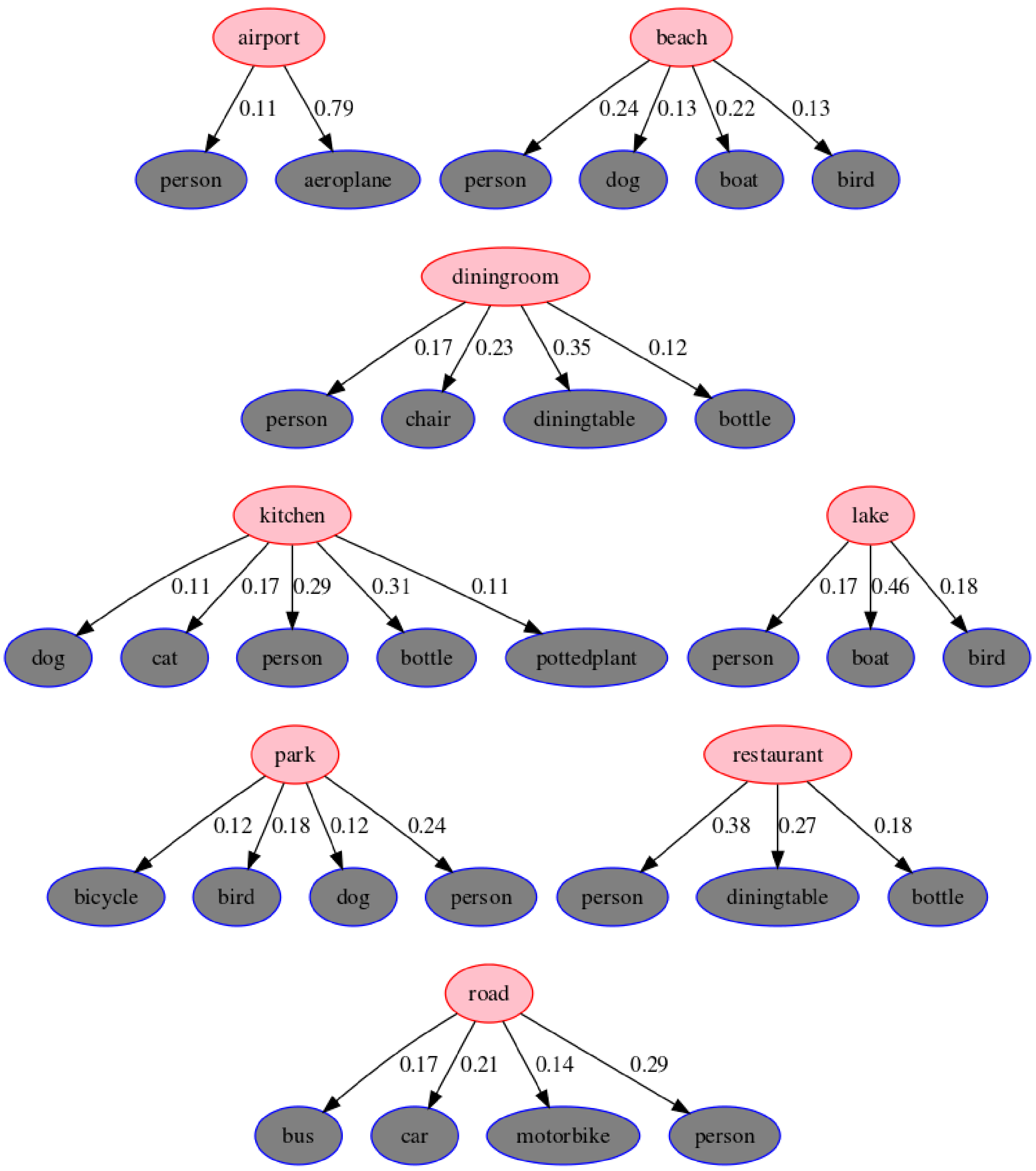


Figure 4: The scene-property knowledge based on 1928 ground-truth semantic annotations. An edge costs, $c = S_n[i][j]$, indicates the probability that an object-class j (gray nodes) belongs to a scene-class i (red nodes). Only object-class nodes whose edge cost is at least 0.10 are drawn. Due to space issues, we show merely 8 out of 27 scene-classes.

and Bottle is low in the scene class of lake, beach, as well as park.

1.3. Relative-location knowledge

For the construction of relative-location knowledge, all images are quantized to 200×200 pixels. A relative-location matrix for an object s with respect to an object t, i.e. matrix R_{st}, is

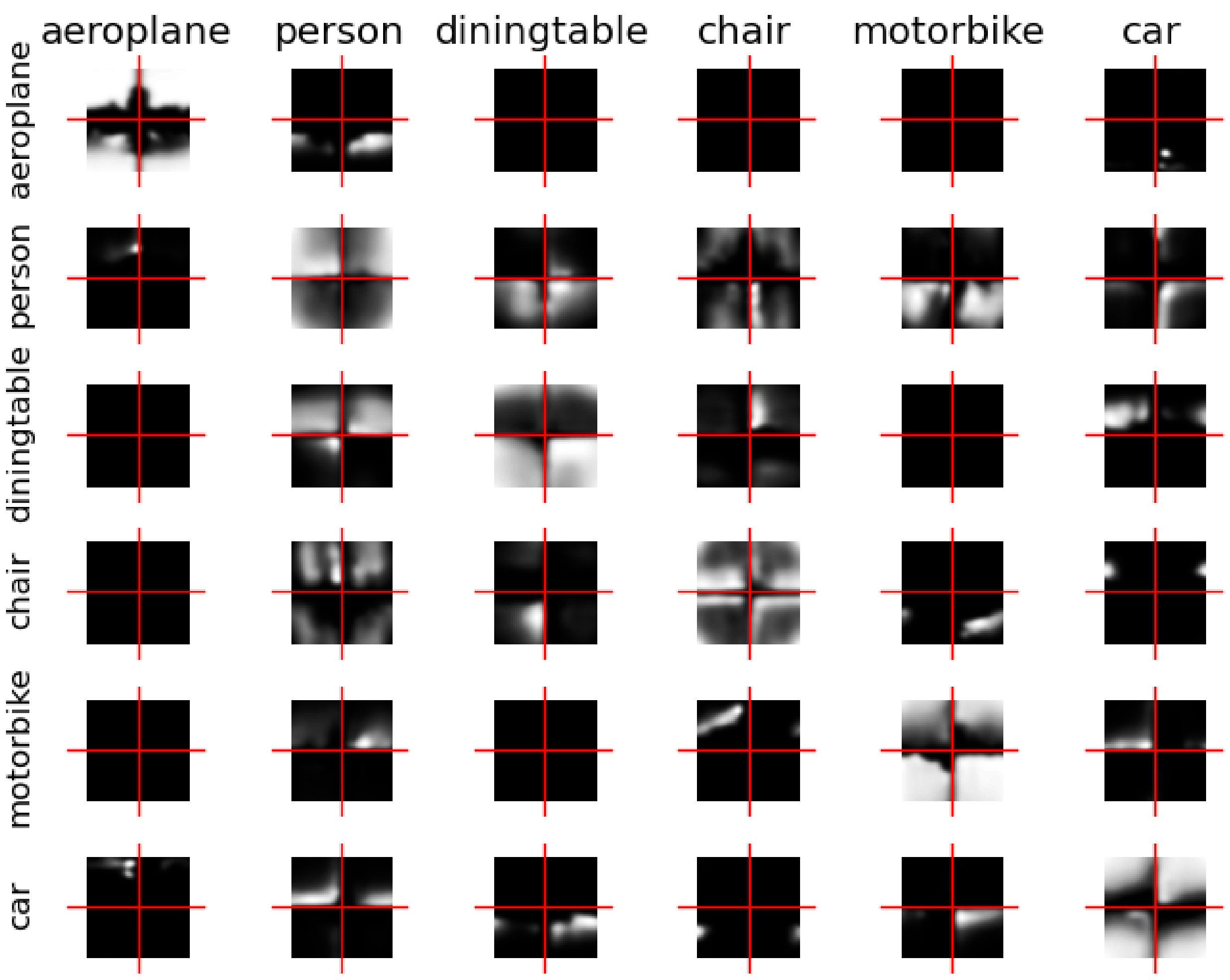


Figure 5: The relative-location knowledge based on 1928 ground-truth semantic annotations. Each square cell is a 200-by-200 R_{st} matrix that encodes the probability that, for a pixel p' whose label is t, a pixel p at offset (i, j) from p' has class label s. Darker areas indicate lower probability. Red solid horizontal and vertical lines help determining the center area of a cell. For example, from the first row and the second column, we get $s = person$ and $t = aeroplane$. Then it reads that a person is likely below an aeroplane since a lighter area is below the horizontal line. Due to space issues, we show merely 7×7 out of 20×20 object-classes.

calculated by counting the offset of pixels for each class from the centroid of each superpixel of a given class. In order to reduce the bias from small pixel shifts in ground-truth semantic annotations, as suggested in [14], values of R_{st} are by a Gaussian filter with variance equal to 10% of the width and height of the image.

Figure 5 visualizes the relative-location knowledge. It can be inferred that the probability that aeroplanes are above Person and Car is high. While, Diningtable likely presents below Person. The Car is suggested to be surrounded by another Car.

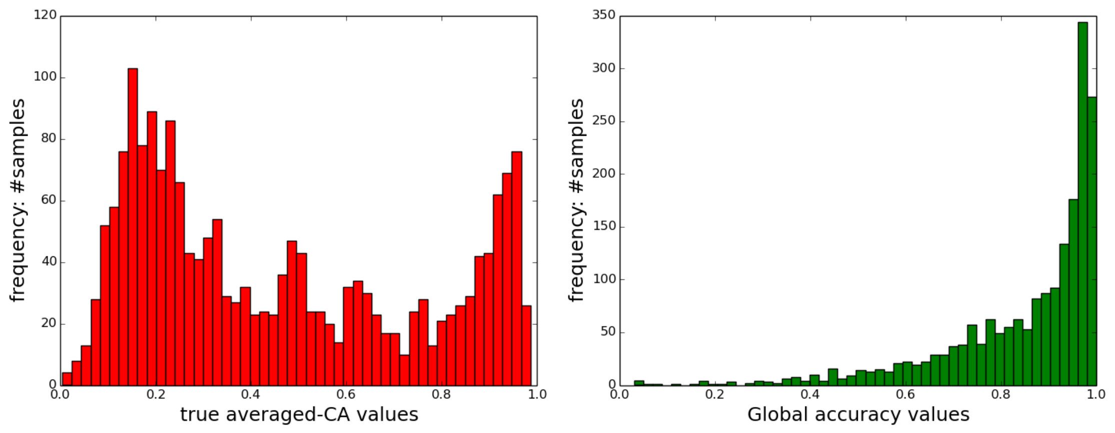


Figure 6: The histograms of true averaged-CA and global-accuracy values of 1928 samples in Pascal VOC-2010 dataset.

2. The knowledge-compatibility benchmarker

The knowledge-compatibility benchmarker is essentially a supervised regression machine. It maps an input feature vector $\mathbf{x} \in \mathbb{R}^{12}$ to an averaged-CA value $y \in \mathbb{R}$. In particular, $\mathbf{x} = [\mathbf{x}_c\ \mathbf{x_s}\ \mathbf{x_r}]^T$ is extracted from a semantic annotation and the vision-based knowledge. The regressor's estimate of averaged-CA is considered as the knowledge-compatibility score of a semantic annotation, from which $\mathbf{x}$ is extracted, see section III.

2.1. Setups

For developing the regression machine, we prepared a set of samples $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$, with $n = 1928$ is the number of images in Pascal VOC 2010 dataset that have ground-truth semantic annotations. Therefore, the true averaged-CA, i.e. $y_i \in D$. of each (predicted) semantic annotation can be calculated. To obtain (predicted) semantic annotations, we employ an MRF-based semantic segmentation method of [1] with unary and pairwise energy functions; our implementation is based on the OpenGM2 library [26]. Figure 6:left shows the histogram of true averaged-CA values. It indicates that those values are well distributed across the entire range: $[0, 1]$. In contrast, another performance metric, i.e global accuracy, has values that do not cover its range. This is the primary rationale of selecting true averaged-CA as the target value y for the regression.

We split the sample set D randomly into training data D_{tr} and testing data D_{te} with the ratio of 7 : 3, respectively. In order to obtain solid results so as to minimize the bias due to random splits, we performed 100 splits of D yielding $\{(D_{tr}^i, D_{te}^i)\}_{i=1}^{100}$. Consequently, we perform 100 independent end-to-end regressions (tuning, training and testing), one for each (D^i_{tr}, D^i_{te}), then report some statistics of regression performance over all splits.

We use two regression performance metrics, i.e. the mean squared error (MSE) and the R^2-score. The MSE is a risk metric corresponding to the expected value of the squared (quadratic) error loss. Whereas, the R^2-score is the coefficient of determination that provides a measure of how well future samples are likely to be predicted by the model. Let y_i and $\hat{y}_i$ denote the true output value and the corresponding predicted value of the i-th sample, respectively. Then, the MSE and R^2-score over n samples are defined as $MSE(y,\hat{y}) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \hat{y}_i)^2$ and $R^2(y,\hat{y}) = 1 - \frac{\sum_{i=0}^{n-1}(y_i-\hat{y}_i)^2}{\sum_{i=0}^{n-1}(y_i-\bar{y})^2}$, where $\bar{y} = \frac{1}{n}\sum_{i=0}^{n-1}$. For MSE, the best value is 0.0, higher values are worse. While, for R^2-score, the best possible score is 1.0, lower values are worse.

Furthermore, since Support Vector Machine algorithms are not scale-invariant, we standardize the data of D^i_{tr} to have a zero mean and a unit variance. The resulted standardization parameter then is also applied to the testing data D^i_{te}. We tune both the Support Vector Regressor and the Gradient Boosting Regressor using grid search with 10-fold cross-validation. For the Gradient Boosting Regression, we obtained the following hyperparameters: huber loss, 300 estimators, learning rate of 0.1 and maximum depth of 3. For the ν-SVR, we obtained: $C = 1.7$, $\nu = 0.4$, $\gamma = 0.0$ and RBF (Gaussian) kernel with the degree of 1. We also ensure that the two regression models do not suffer from either overfitting and/or high-bias by observing the learning curves. We remark that our regression implementation is based on the scikit-learn library [27].

2.2. Results

Table 1 shows the testing performance of both regression methods over 100 data-set splits. In addition, figure 7 plots the best Gradient Boosting regression (GBR) performance over 100 sets of testing data, i.e. $\{D^i_{te}\}_{i=1}^{100}$, in terms of estimated $\hat{y}$ vs the true averaged-CA y values. Whereas, figure 8 plots the one of ν-SVR. Notice that a predicted averaged-CA value is equivalent to a knowledge-compatibility score. In testing phase, the GBR achieves an averaged R^2-score of 0.671 and an averaged MSE of 0.030. It outperforms the ν-SVR that attains an averaged R^2-score of 0.639 and an averaged MSE of 0.033. We also observe that both regression machines tend to "underestimate" true averaged-CA values. This is beneficial whenever predicted averaged-CA values as the knowledge-compatibility scores are used as feedback during a segmentation process.

Table 1: The regression testing performance of both GBR and ν-SVR over all 100 sets of testing data. For MSE, the best value is 0.0, higher values are worse. While, for R^2-score, the best possible score is 1.0, lower values are worse.

	R^2-score		**MSE**	
	$mean \pm std$	(min, max)	$mean \pm std$	(min, max)
GBR	0.671 ± 0.026	$(0.604, 0.737)$	0.030 ± 0.002	$(0.024, 0.034)$
ν-SVR	0.639 ± 0.025	$(0.560, 0.697)$	0.033 ± 0.002	$(0.028, 0.038)$

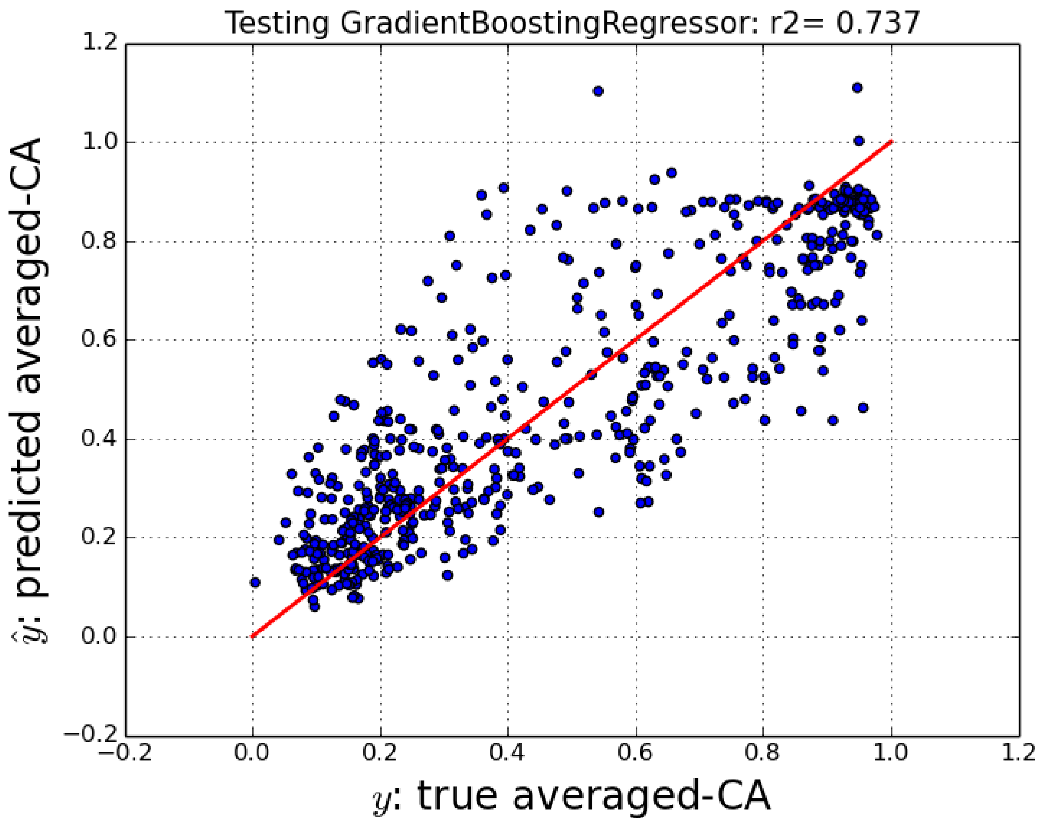


Figure 7: The best Gradient Boosting regression (GBR) performance over 100 sets of testing data: $\{D^i_{te}\}^{100}_{i=1}$, in terms of estimated $\hat{y}$ vs the true averaged-CA y values. Here, the GBR achieves an R^2-score of 0.737 and an MSE of 0.034. The red solid line indicates where $\hat{y} = y$ lies.

Furthermore, figure 9, 10 and 11 depict several qualitative results of semantic segmentation along with the corresponding true averaged-CA values as well as the knowledge-compatibility score. Using those 3 figures, we deliberately separate the presentation of images that have one, two, and three or more object classes in their ground-truth annotations. Recall that the co-occurrence and relative-location knowledge work merely with annotations with two or more different object classes. Besides, the scene-property knowledge provides best clues when the

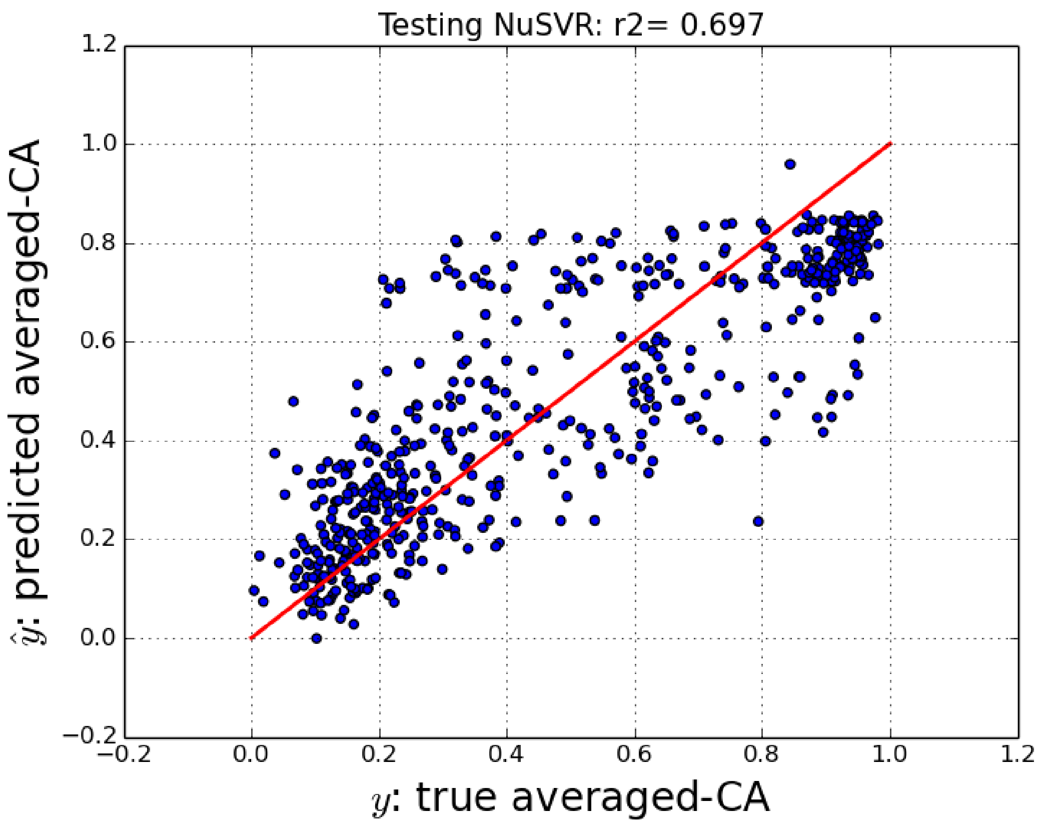


Figure 8: The best ν-Support Vector regression (SVR) performance over 100 sets of testing data: $\{D^i_{te}\}^{100}_{i=1}$, in terms of estimated $\hat{y}$ vs the true averaged-CA y values. Here, the ν-SVR achieves an R^2-score of 0.697 and an MSE of 0.038. The red solid line indicates where $\hat{y} = y$ lies.

scene class of an image is definitive and apparent.

Largely, semantic annotations that contain two or more object classes obtain relatively good knowledge-compatibility scores in the sense that they are close to the true averaged-CA values. This can be seen, for instance, in subfigure 10(a) to 10(k) and subfigure 11(a) to 11(f). The absolute difference between their scores and true averaged-CA is less than 0.100.

In addition, most scores are pessimistic predictions of true averaged-CA values. For instance, in subfigure 10(l) and 10(n), the benchmarker outputs lower scores than true average-CA values partly because the scene-class of those images are not obvious. Contrarily, in subfigure 9(j), we obtained a score of 0.256 that is of 0.066 bigger than the true average-CA, which is of 0.190. This happens because based on scene-property knowledge, the wrong object-class Cat share the same probability to exist with the true object-class DOG in the scene-class of living-room. Another optimistic score appears in subfigure 11(l), where the score is of 0.221 higher. It turns out the that original image is of the Diningroom scene-class. Unfortunately, the semantic annotation has all three object-classes, namely: Person, Table and Chair. Those object classes have high values in all three kinds of knowledge: they are likely to co-occur, their probabilities to present in the scene-class of Diningroom are almost equal and their relative location in the semantic annotation agrees with the relative-location knowledge.

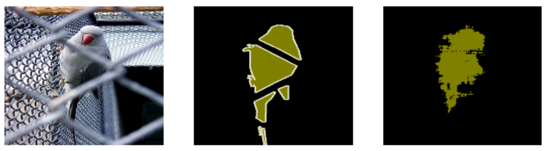
VI. Conclusions

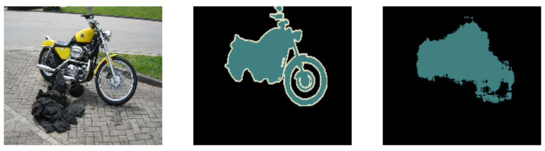
We believe that the quality of a semantic annotation is proportionally related to its compatibility with the vision-based knowledge. As a result, we formulate a supervised regression problem that maps a feature vector extracted from semantic annotations and the vision-based knowledge *to* averaged class-accuracy values. Concretely, we propose a knowledge-compatibility benchmarker, whose backbone is a regression machine. It outputs knowledge compatibility scores, which are essentially the estimates of true averaged class-accuracy values. The vision-based knowledge is constructed from ground-truth semantic annotations of 2D images. It encodes three kinds of information, namely: a) co-occurrence statistics, b) scene properties and c) relative positions. In order to capture the compatibility between a semantic annotation and each type of the knowledge, we introduce three kinds of feature vectors for regression. Each specifies the characteristics of a probability vector that is derived from the given annotation and the vision-based knowledge.

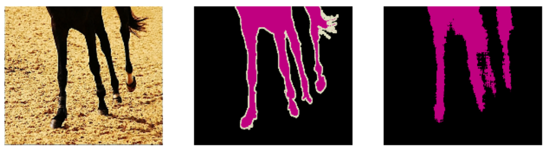
Experiment results show that the constructed vision-based knowledge has a relatively high agreement with a set of human-centric queries about some extreme cases of the knowledge. This suggests that it generally resembles the humans' common-sense. Furthermore, we found that as the core of our proposed knowledge-compatibility benchmarker, the Gradient Boosting regression outperforms the ν-Support Vector regression. The former achieves best performance at an R^2-score of 0.737 and an MSE of 0.034, while the latter's is at an R^2-score of 0.697 and an MSE of 0.038. This indicates not only that the vision-based knowledge is well constructed but also that the feature vector is justifiable. Future work includes the utilization of knowledge-compatibility scores on-the-fly during a semantic segmentation process.

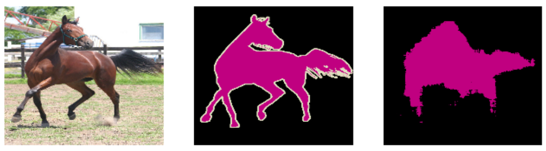
Acknowledgement

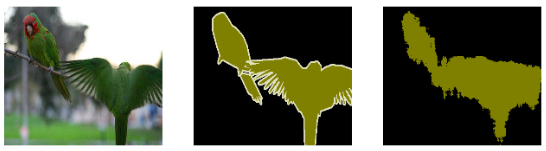
This research was supported by Universitas Indonesia and Directorate General of Higher Education, under Grant International Research Collaboration and Scientific Publication No: 0403 / UN2.R12 / HKP.05.00 / 2015.

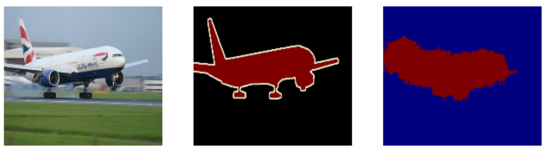

(a) score = 0.832, averaged-CA = 0.837, $\Delta = 0.005$

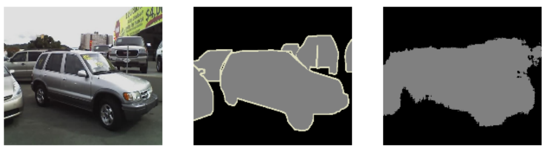

(b) score = 0.862, averaged-CA = 0.87, $\Delta = 0.007$

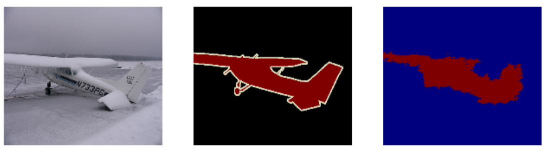

(c) score = 0.866, averaged-CA = 0.872, $\Delta = 0.006$

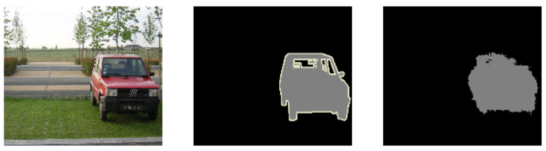

(d) score = 0.885, averaged-CA = 0.882, $\Delta = 0.002$

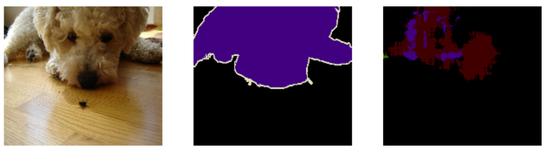

(e) score = 0.876, averaged-CA = 0.922, $\Delta = 0.046$

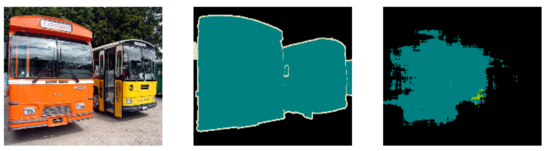

(f) score = 0.88, averaged-CA = 0.792, $\Delta = 0.088$

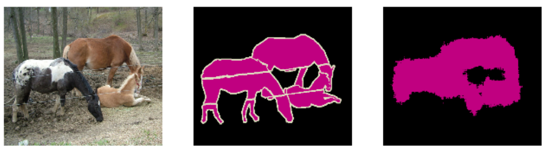

(g) score = 0.894, averaged-CA = 0.912, $\Delta = 0.018$

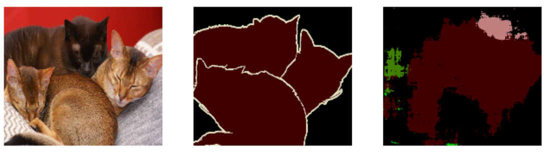

(h) score = 0.79, averaged-CA = 0.888, $\Delta = 0.098$


(i) score = 0.878, averaged-CA = 0.92, $\Delta = 0.042$


(j) score = 0.256, averaged-CA = 0.19, $\Delta = 0.066$


(k) score = 0.279, averaged-CA = 0.199, $\Delta = 0.079$


(l) score = 0.878, averaged-CA = 0.892, $\Delta = 0.014$


(m) score = 0.129, averaged-CA = 0.21, $\Delta = 0.081$

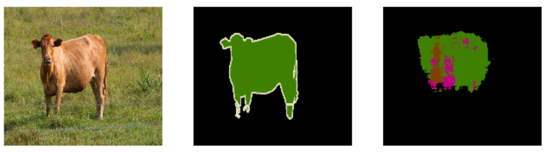
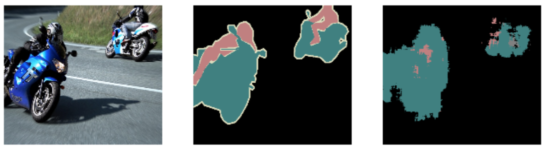
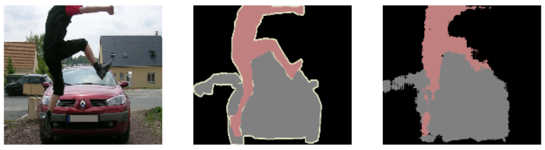
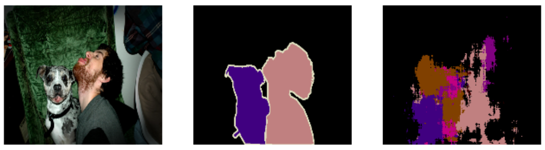
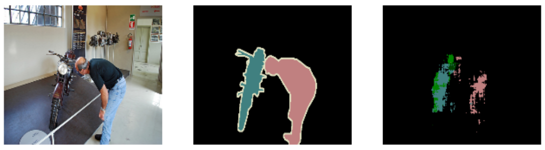

(n) score = 0.247, averaged-CA = 0.383, $\Delta = 0.135$

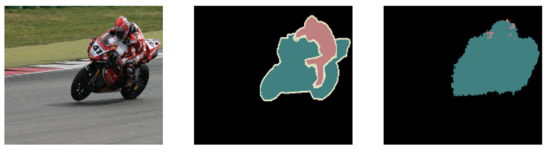
Figure 9: Qualitative results of semantic segmentation with one true object-classes. The Δ indicates an absolute difference between the knowledge-compatibility score and the true averaged-CA. In each subfigure, the left, middle, right images are original, ground-truth annotation, and (estimated) annotation, respectively.

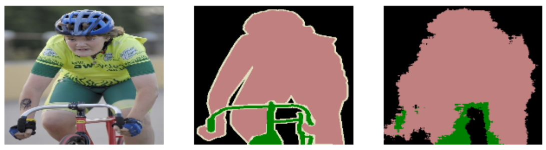

(a) score = 0.388, averaged-CA = 0.388, $\Delta = 0.0$

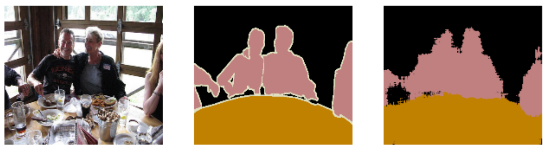

(b) score = 0.853, averaged-CA = 0.851, $\Delta = 0.003$

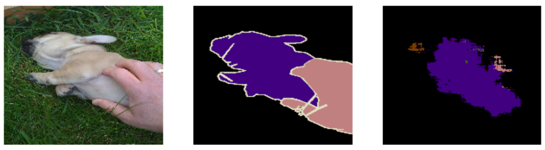

(c) score = 0.173, averaged-CA = 0.19, $\Delta = 0.017$

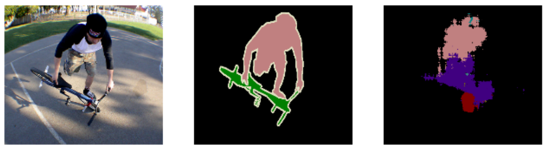

(d) score = 0.168, averaged-CA = 0.264, $\Delta = 0.096$

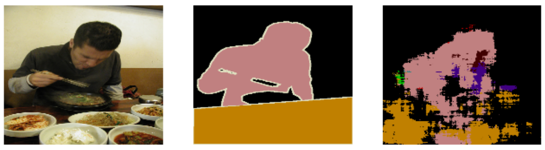

(e) score = 0.574, averaged-CA = 0.556, $\Delta = 0.018$

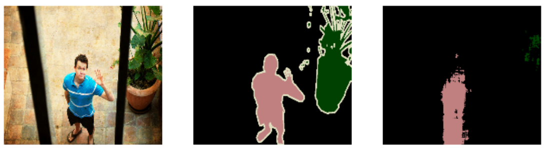

(f) score = 0.872, averaged-CA = 0.777, $\Delta = 0.094$

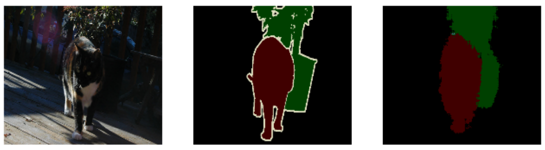

(g) score = 0.818, averaged-CA = 0.908, $\Delta = 0.091$

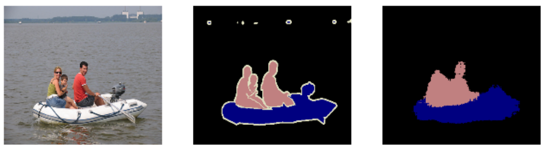

(h) score = 0.14, averaged-CA = 0.227, $\Delta = 0.087$


(i) score = 0.109, averaged-CA = 0.156, $\Delta = 0.047$


(j) score = 0.077, averaged-CA = 0.166, $\Delta = 0.09$


(k) score = 0.504, averaged-CA = 0.423, $\Delta = 0.081$


(l) score = 0.399, averaged-CA = 0.664, $\Delta = 0.265$


(m) score = 0.683, averaged-CA = 0.855, $\Delta = 0.172$

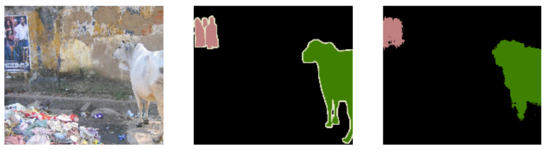
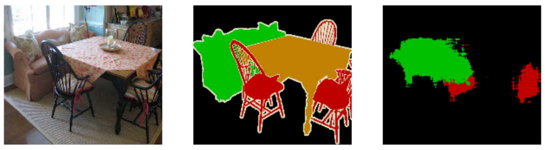
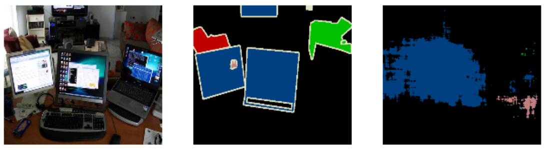
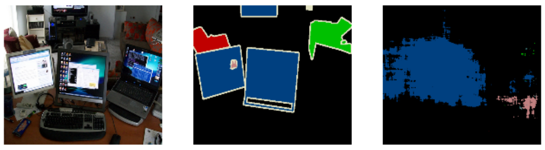
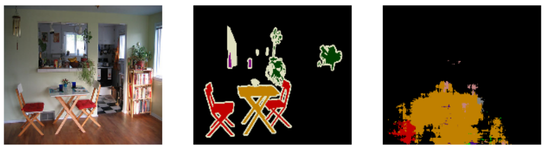

(n) score = 0.602, averaged-CA = 0.847, $\Delta = 0.245$

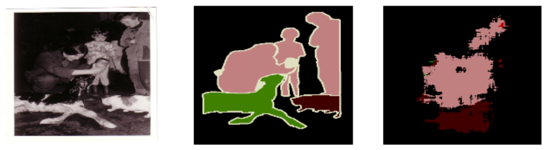
Figure 10: Qualitative results of semantic segmentation with two true object-classes. The Δ indicates an absolute difference between the knowledge-compatibility score and the true averaged-CA. In each subfigure, the left, middle, right images are original, ground-truth annotation, and (estimated) annotation, respectively.

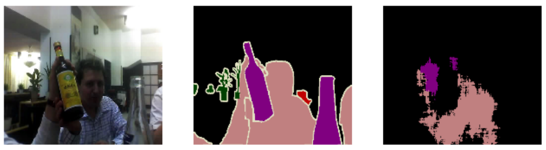

(a) score = 0.338, averaged-CA = 0.294, $\Delta = 0.045$

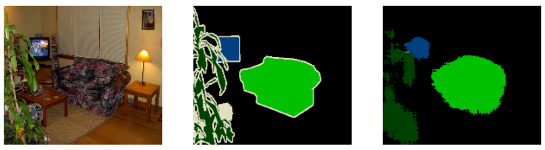

(b) score = 0.214, averaged-CA = 0.236, $\Delta = 0.022$

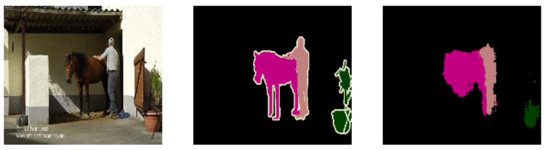

(c) score = 0.214, averaged-CA = 0.236, $\Delta = 0.022$

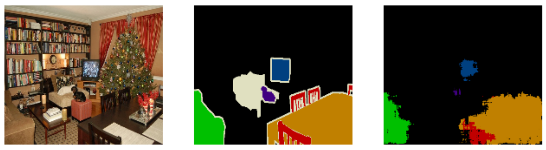

(d) score = 0.119, averaged-CA = 0.096, $\Delta = 0.023$

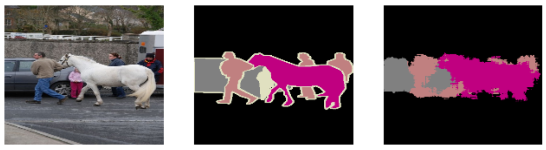

(e) score = 0.132, averaged-CA = 0.122, $\Delta = 0.01$

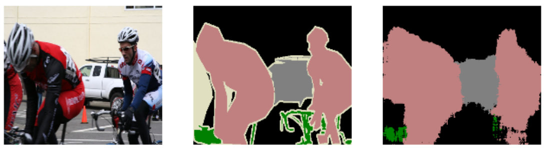

(f) score = 0.184, averaged-CA = 0.237, $\Delta = 0.054$

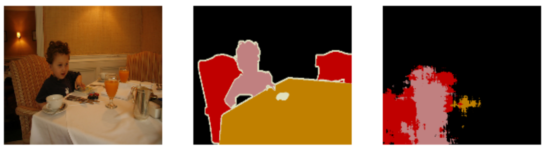

(g) score = 0.252, averaged-CA = 0.543, $\Delta = 0.291$

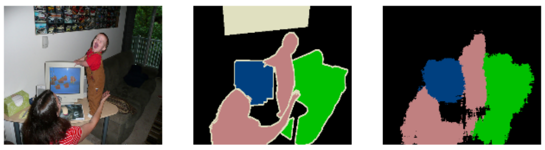

(h) score = 0.55, averaged-CA = 0.675, $\Delta = 0.125$


(i) score = 0.269, averaged-CA = 0.608, $\Delta = 0.339$


(j) score = 0.48, averaged-CA = 0.772, $\Delta = 0.292$


(k) score = 0.524, averaged-CA = 0.739, $\Delta = 0.215$


(l) score = 0.434, averaged-CA = 0.213, $\Delta = 0.221$


(m) score = 0.52, averaged-CA = 0.712, $\Delta = 0.192$

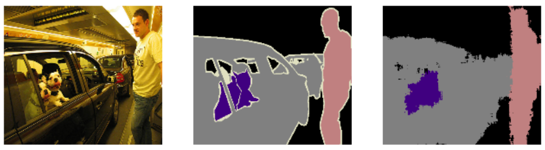

(n) score = 0.456, averaged-CA = 0.859, $\Delta = 0.403$

Figure 11: Qualitative results of semantic segmentation with three or more true object-classes. The Δ indicates an absolute difference between the knowledge-compatibility score and the true averaged-CA. In each subfigure, the left, middle, right images are original, ground-truth annotation, and (estimated) annotation, respectively.

References

[1] J. Shotton, J. Winn, C. Rother, and A. Criminisi, "Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context," *Int. J. Comput. Vision*, vol. 81, no. 1, pp. 2–23, Jan. 2009.

[2] L. Ladicky, C. Russell, P. Kohli, and P. H. S. Torr, "Associative hierarchical crfs for object class image segmentation," in *Computer Vision, 2009 IEEE 12th International Conference on*, Sept 2009, pp. 739–746.

[3] P. Krähenbühl and V. Koltun, "Efficient inference in fully connected crfs with gaussian edge potentials," in *Advances in Neural Information Processing Systems 24*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, Eds. Curran Associates, Inc., 2011, pp. 109–117.

[4] X. Boix, J. M. Gonfaus, J. van de Weijer, A. D. Bagdanov, J. S. Gual, and J. Gonzàlez, "Harmony potentials - fusing global and local scale for semantic image segmentation." *International Journal of Computer Vision*, vol. 96, no. 1, pp. 83–102, 2012.

[5] J. Alvarez, M. Salzmann, and N. Barnes, "Large-scale semantic co-labeling of image sets," in *Applications of Computer Vision (WACV), 2014 IEEE Winter Conference on*, March 2014, pp. 501–508.

[6] J. Z. Ning Zhang, "A study of x-ray machine image local semantic features extraction model based on bag-ofwords for airport security," *Internatioanal Journal on Smart Sensing and Intelligent Systems*, vol. 8, no. 1, p. 45, 2015.

[7] Aprinaldi, I. Habibie, R. Rahmatullah, A. Kurniawan, A. Bowolaksono, W. Jatmiko, and B. Wiweko, "Arcpso: Ellipse detection method using particle swarm optimization and arc combination," in *Advanced Computer Science and Information Systems (ICACSIS)*, ser. ICACSIS 2014. IEEE, 2014, pp. 408 – 413.

[8] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results," http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[9] L. Ladicky, C. Russell, P. Kohli, and P. H. S. Torr, "Inference methods for crfs with co-occurrence statistics," *International Journal of Computer Vision*, vol. 103, no. 2, pp. 213–225, 2013.

[10] A. Rabinovich, A. Vedaldi, C. Galleguillos, E. Wiewiora, and S. Belongie, "Objects in context," in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, Oct 2007, pp. 1–8.

[11] S. Gould, R. Fulton, and D. Koller, “Decomposing a scene into geometric and semantically consistent regions,” in *Computer Vision, 2009 IEEE 12th International Conference on*, Sept 2009, pp. 1–8.

[12] A. Gupta, A. A. Efros, and M. Hebert, “Blocks world revisited: Image understanding using qualitative geometry and mechanics,” in *European Conference on Computer Vision(ECCV)*, 2010.

[13] A. Gupta and L. S. Davis, “Beyond nouns: Exploiting prepositions and comparative adjectives for learning visual classifiers,” in *Proceedings of the 10th European Conference on Computer Vision: Part I*, ser. ECCV ’08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 16–29.

[14] S. Gould, J. Rodgers, D. Cohen, G. Elidan, and D. Koller, “Multi-class segmentation with relative location prior.” *International Journal of Computer Vision*, vol. 80, no. 3, pp. 300–316, 2008.

[15] S. Divvala, D. Hoiem, J. Hays, A. Efros, and M. Hebert, “An empirical study of context in object detection,” in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, June 2009, pp. 1271–1278.

[16] M. J. Choi, J. Lim, A. Torralba, and A. Willsky, “Exploiting hierarchical context on a large database of object categories,” in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, June 2010, pp. 129–136.

[17] N. E. Maillot and M. Thonnat, “Ontology based complex object recognition,” *Image and Vision Computing*, vol. 26, no. 1, pp. 102 – 113, 2008, cognitive Vision-Special Issue.

[18] J. Tighe and S. Lazebnik, “Understanding scenes on many levels,” in *Proceedings of the 2011 International Conference on Computer Vision*, ser. ICCV ’11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 335–342.

[19] S. Gould, J. Zhao, X. He, and Y. Zhang, “Superpixel graph label transfer with learned distance metric,” in *ECCV*, 2014.

[20] A. Oliva and A. Torralba, “Modeling the shape of the scene: A holistic representation of the spatial envelope,” *Int. J. Comput. Vision*, vol. 42, no. 3, pp. 145–175, May 2001.

[21] A. J. Smola and B. Schölkopf, “A tutorial on support vector regression,” *Statistics and Computing*, vol. 14, no. 3, pp. 199–222, Aug. 2004.

[22] J. H. Friedman, “Greedy function approximation: A gradient boosting machine,” *Annals of Statistics*, vol. 29, pp. 1189–1232, 2000.

[23] G. Li, H. Meng, M. Q. Yang, and J. Y. Yang, "Combining support vector regression with feature selection for multivariate calibration," *Neural Computing and Applications*, vol. 18, no. 7, pp. 813–820, 2009.

[24] J. H. Friedman, "Stochastic gradient boosting," *Comput. Stat. Data Anal.*, vol. 38, no. 4, pp. 367–378, Feb. 2002.

[25] A. Natekin and A. Knoll, "Gradient boosting machines, a tutorial," *Frontiers in Neurorobotics*, vol. 7, 2013.

[26] B. Andres, B. T., and J. H. Kappes, "OpenGM: A C++ library for discrete graphical models," *ArXiv e-prints*, 2012.

[27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.