The PASCAL Visual Object Classes Challenge 2008 (VOC2008) Development Kit

Mark Everingham John Winn

April 21, 2008

Contents

1 Challenge

The goal of this challenge is to recognize objects from a number of visual object classes in realistic scenes (i.e. not pre-segmented objects). There are twenty object classes:

- person
- bird, cat, cow, dog, horse, sheep
- aeroplane, bicycle, boat, bus, car, motorbike, train
- bottle, chair, dining table, potted plant, sofa, tv/monitor

There are two main tasks:

- Classification: For each of the classes predict the presence/absence of at least one object of that class in a test image.
- Detection: For each of the classes predict the bounding boxes of each object of that class in a test image (if any).

In addition, there are two "taster" tasks operating on a subset of the provided data:

- Segmentation: For each pixel in a test image, predict the class of the object containing that pixel or 'background' if the object does not belong to one of the twenty specified classes.
- Person Layout: For each 'person' object in a test image (indicated by a bounding box of the person), predict the presence/absence of parts (head/hands/feet), and the bounding boxes of those parts.

2 Data

The VOC2008 database contains a total of 10,057 annotated images. The data is released in two phases: (i) training and validation data with annotation is released with this development kit; (ii) test data *without* annotation is released at a later date. After completion of the challenge, annotation for the test data will be released.

2.1 Classification/Detection Image Sets

For the main tasks – classification and detection, there are four sets of images provided:

`train`:	Training data
`val`:	Validation data (suggested). The validation data may be used as additional training data (see below).
`trainval`:	The union of `train` and `val`.
`test`:	Test data. The test set is not provided in the development kit. It will be released in good time before the deadline for submission of results.

Table 1: Statistics of the main image sets. Object statistics list only the 'non-difficult' objects used in the evaluation.

	train		val		trainval		test	
	img	**obj**	**img**	**obj**	**img**	**obj**	**img**	**obj**
Aeroplane	119	159	117	157	236	316	–	–
Bicycle	92	133	100	136	192	269	–	–
Bird	166	239	139	237	305	476	–	–
Boat	111	170	96	166	207	336	–	–
Bottle	129	229	114	228	243	457	–	–
Bus	48	61	52	68	100	129	–	–
Car	243	426	223	414	466	840	–	–
Cat	159	186	169	192	328	378	–	–
Chair	177	313	174	310	351	623	–	–
Cow	37	61	37	69	74	130	–	–
Diningtable	53	55	52	55	105	110	–	–
Dog	186	238	202	239	388	477	–	–
Horse	96	139	102	146	198	285	–	–
Motorbike	102	137	102	135	204	272	–	–
Person	947	1996	1055	2172	2002	4168	–	–
Pottedplant	85	178	95	183	180	361	–	–
Sheep	32	67	32	78	64	145	–	–
Sofa	69	74	65	77	134	151	–	–
Train	78	83	73	83	151	166	–	–
Tvmonitor	107	138	108	136	215	274	–	–
Total	2113	5082	2227	5281	4340	10363	–	–

Table 1 summarizes the number of objects and images (containing at least one object of a given class) for each class and image set. The data has been split into 50% for training/validation and 50% for testing. The distributions of images and objects by class are approximately equal across the training/validation and test sets.

2.2 Segmentation Taster Image Sets

For the segmentation taster task, corresponding image sets are provided as in the classification/detection tasks. To increase the amount of data, the training and validation image sets include images from the 2007 segmentation taster, indicated by the '2007' prefix. The test set contains only new images, and is a subset of the test set for the main tasks for which pixel-wise segmentations have been prepared. Table 2 summarizes the number of objects and images (containing at least one object of a given class) for each class and image set, for the combined 2007 and 2008 data. In addition to the segmented images for training and validation, participants are free to use the un-segmented training/validation images supplied for the main classification/detection tasks.

Table 2: Statistics of the segmentation taster image sets.

	train		val		trainval		test	
	img	**obj**	**img**	**obj**	**img**	**obj**	**img**	**obj**
Aeroplane	32	35	26	32	58	67	–	–
Bicycle	27	35	23	33	50	68	–	–
Bird	35	44	24	33	59	77	–	–
Boat	33	56	27	31	60	87	–	–
Bottle	30	55	28	41	58	96	–	–
Bus	25	31	27	40	52	71	–	–
Car	46	76	34	77	80	153	–	–
Cat	30	34	35	39	65	73	–	–
Chair	49	109	40	79	89	188	–	–
Cow	19	48	25	41	44	89	–	–
Diningtable	35	36	32	35	67	71	–	–
Dog	26	31	39	51	65	82	–	–
Horse	30	38	37	40	67	78	–	–
Motorbike	33	37	26	38	59	75	–	–
Person	172	299	171	320	343	619	–	–
Pottedplant	26	46	34	82	60	128	–	–
Sheep	20	49	27	72	47	121	–	–
Sofa	27	31	33	41	60	72	–	–
Train	27	32	30	34	57	66	–	–
Tvmonitor	36	44	33	44	69	88	–	–
Total	511	1166	512	1203	1023	2369	–	–

Table 3: Statistics of the person layout taster image sets. Object statistics list only the 'person' objects for which layout information (parts) is present.

	`train`		`val`		`trainval`		`test`	
	img	**obj**	**img**	**obj**	**img**	**obj**	**img**	**obj**
Person	141	202	104	165	245	367	–	–

2.3 Person Layout Taster Image Sets

For the person layout taster task, corresponding image sets are provided as in the classification/detection tasks. A person is indicated by a bounding box, and each person has been annotated with part layout (head, hands, feet). As in the segmentation taster task, the training and validation image sets include images from the 2007 person layout taster, indicated by the '2007' prefix. The test set contains only new images, and is disjoint from the test set for the main tasks. Table 3 summarizes the number of 'person' objects annotated with layout for each image set.

2.4 Ground Truth Annotation

Objects of the twenty classes listed above are annotated in the ground truth. For each object, the following annotation is present:

- **class**: the object class e.g. 'car' or 'bicycle'
- **bounding box**: an axis-aligned rectangle specifying the extent of the object visible in the image.
- **view**: 'frontal', 'rear', 'left' or 'right'. The views are subjectively marked to indicate the view of the 'bulk' of the object. Some objects have no view specified.
- **'truncated'**: an object marked as 'truncated' indicates that the bounding box specified for the object does not correspond to the full extent of the object e.g. an image of a person from the waist up, or a view of a car extending outside the image.
- **'occluded'**: an object marked as 'occluded' indicates that a significant portion of the object within the bounding box is occluded by another object.
- **'difficult'**: an object marked as 'difficult' indicates that the object is considered difficult to recognize, for example an object which is clearly visible but unidentifiable without substantial use of context. Objects marked as difficult are currently *ignored* in the evaluation of the challenge.

In preparing the ground truth, annotators were given a detailed list of guidelines on how to complete the annotation. These are available on the main challenge web-site [1].

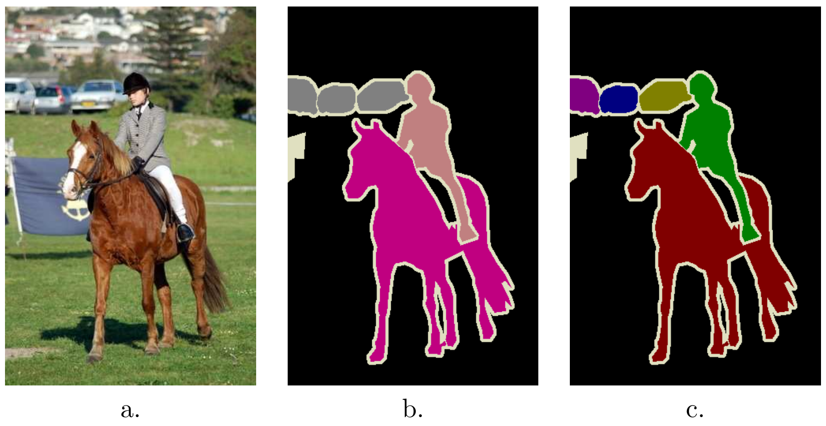

Figure 1: Example of segmentation taster ground truth. **a.** Training image **b.** Class segmentation showing background, car, horse and person labels. The cream-colored 'void' label is also used in border regions and to mask difficult objects. **c.** Object segmentation where individual object instances are separately labelled.

2.5 Segmentation Taster Ground Truth

For the segmentation image sets, each image has two corresponding types of ground truth segmentation provided:

- class segmentation: each pixel is labelled with the ground truth class or background.
- object segmentation: each pixel is labelled with an object number (from which the class can be obtained) or background.

Figure 2.5 gives an example of these two types of segmentation for one of the training set images. The ground truth segmentations are provided to a high degree of accuracy, but are not pixel accurate, as this would have greatly extended the time required to gather these segmentations. Instead, they were labelled so that a bordering region with a width of five pixels may contain either object or background. Bordering regions are marked with a 'void' label (index 255), indicating that the contained pixels can be any class including background. The void label is also used to mask out ambiguous, difficult or heavily occluded objects and also to label regions of the image containing objects too small to be marked, such as crowds of people. All void pixels are ignored when computing segmentation accuracies and should be treated as unlabelled pixels during training.

In addition to the ground truth segmentations given, participants are free to use any of the ground truth annotation for the classification/detection tasks.

2.6 Person Layout Taster Ground Truth

For the person layout taster task, 'person' objects are additionally annotated with three 'parts':

- head – zero or one per person
- hand – zero, one, or two per person
- foot – zero, one, or two per person

For each annotated person, the presence or absence of each part is listed, and for each part present, the bounding box is specified. The test images for the person layout taster are disjoint from the main image sets. There are no 'difficult' objects.

3 Classification Task

3.1 Task

For each of the twenty object classes predict the presence/absence of at least one object of that class in a test image. The output from your system should be a real-valued confidence of the object's presence so that a precision/recall curve can be drawn. Participants may choose to tackle all, or any subset of object classes, for example "cars only" or "motorbikes and cars".

3.2 Competitions

Two competitions are defined according to the choice of training data: (i) taken from the VOC `trainval` data provided, or (ii) from any source excluding the VOC `test` data provided:

No.	Task	Training data	Test data
1	Classification	`trainval`	`test`
2	Classification	**any but** VOC `test`	`test`

In competition 1, any annotation provided in the VOC `train` and `val` sets may be used for training, for example bounding boxes or particular views e.g. 'frontal' or 'left'. Participants are *not* permitted to perform additional manual annotation of either training or test data.

In competition 2, any source of training data may be used *except* the provided `test` images. Researchers who have pre-built systems trained on other data are particularly encouraged to participate. The test data includes images from "flickr" (`www.flickr.com`); this source of images may *not* be used for training. Participants who have acquired images from flickr for training must submit them to the organizers to check for overlap with the test set.

3.3 Submission of Results

A separate text file of results should be generated for each competition (1 or 2) and each class e.g. 'car'. Each line should contain a single identifier and the confidence output by the classifier, separated by a space, for example:

```
comp1_cls_test_car.txt:
    2008_000002 0.129824
    2008_000005 0.556163
    2008_000010 0.227097
```

```
2008_000014 0.764145
2008_000016 0.098249
```

Greater confidence values signify greater confidence that the image contains an object of the class of interest. The example classifier implementation (section 7.2.1) includes code for generating a results file in the required format.

3.4 Evaluation

The classification task will be judged by the precision/recall curve. The principal quantitative measure used will be the average precision (AP). Example code for computing the precision/recall and AP measure is provided in the development kit.

Images which contain only objects marked as 'difficult' (section 2.4) are currently *ignored* by the evaluation. The final evaluation may include separate results including such "difficult" images, depending on the submitted results.

Participants are expected to submit a *single* set of results per method employed. Participants who have investigated several algorithms may submit one result per method. Changes in algorithm parameters do *not* constitute a different method – all parameter tuning must be conducted using the training and validation data alone.

4 Detection Task

4.1 Task

For each of the twenty classes predict the bounding boxes of each object of that class in a test image (if any). Each bounding box should be output with an associated real-valued confidence of the detection so that a precision/recall curve can be drawn. Participants may choose to tackle all, or any subset of object classes, for example "cars only" or "motorbikes and cars".

4.2 Competitions

Two competitions are defined according to the choice of training data: (i) taken from the VOC `trainval` data provided, or (ii) from any source excluding the VOC `test` data provided:

No.	Task	Training data	Test data
3	Detection	`trainval`	`test`
4	Detection	**any but** VOC `test`	`test`

In competition 3, any annotation provided in the VOC `train` and `val` sets may be used for training, for example bounding boxes or particular views e.g. 'frontal' or 'left'. Participants are *not* permitted to perform additional manual annotation of either training or test data.

In competition 4, any source of training data may be used *except* the provided `test` images. Researchers who have pre-built systems trained on other data are particularly encouraged to participate. The test data includes images from "flickr" (`www.flickr.com`); this source of images may *not* be used for training. Participants who have acquired images from flickr for training must submit them to the organizers to check for overlap with the test set.

4.3 Submission of Results

A separate text file of results should be generated for each competition (3 or 4) and each class e.g. ‘car’. Each line should be a detection output by the detector in the following format:

```
    <image identifier> <confidence> <left> <top> <right> <bottom>
```

where `(left,top)-(right,bottom)` defines the bounding box of the detected object. The top-left pixel in the image has coordinates $(1, 1)$. Greater confidence values signify greater confidence that the detection is correct. An example file excerpt is shown below. Note that for the image 2008_000016, multiple objects are detected:

```
comp3_det_test_car.txt:
    2008_000014 0.764145 44.182900 49.462700 466.030200 235.963600
    2008_000016 0.098249 15.763800 81.605700 486.900500 220.593300
    2008_000016 0.098249 200.044400 58.959100 359.902300 99.811700
    2008_000016 0.098249 13.543500 47.413900 177.841900 96.703400
    2008_000016 0.098249 467.806400 69.616300 500.000000 103.364100
    2008_000020 0.112754 91.691600 35.703600 482.859300 305.613800
```

The example detector implementation (section 7.2.2) includes code for generating a results file in the required format.

4.4 Evaluation

The detection task will be judged by the precision/recall curve. The principal quantitative measure used will be the average precision (AP). Example code for computing the precision/recall and AP measure is provided in the development kit.

Detections are considered true or false positives based on the area of overlap with ground truth bounding boxes. To be considered a correct detection, the area of overlap a_o between the predicted bounding box B_p and ground truth bounding box B_{gt} must exceed 50% by the formula:

$$a_o = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})} \tag{1}$$

Example code for computing this overlap measure is provided in the development kit. Multiple detections of the *same* object in an image are considered *false* detections e.g. 5 detections of a single object is counted as 1 correct detection and 4 false detections – it is the responsibility of the participant’s system to filter multiple detections from its output.

Objects marked as ‘difficult’ (section 2.4) are currently *ignored* by the evaluation. The final evaluation may include separate results including such “difficult” images, depending on the submitted results.

Participants are expected to submit a *single* set of results per method employed. Participants who have investigated several algorithms may submit one result per method. Changes in algorithm parameters do *not* constitute a different method – all parameter tuning must be conducted using the training and validation data alone.

5 Segmentation Taster

5.1 Task

For each test image pixel, predict the class of the object containing that pixel or 'background' if the object does not belong to one of the twenty specified classes. The output from your system should be an indexed image with each pixel index indicating the number of the inferred class (1-20) or zero, indicating background.

5.2 Competitions

A single competition is defined:

No.	Task	Training data	Test data
5	Segmentation	`trainval`	`test`

Any annotation provided in the VOC `train` and `val` sets may be used for training, for example segmentation, bounding boxes or particular views e.g. 'frontal' or 'left'. Both the images with and without segmentation provided may be used if desired. Participants are *not* permitted to perform additional manual annotation of either training or test data.

5.3 Submission of Results

Submission of results should be as collections of PNG format indexed image files, one per test image, with pixel indices from 0 to 20. The example segmenter implementation (section 7.2.3) includes code for generating results in the required format.

Along with the submitted image files, participants should also state whether their method used segmentation training data only or both segmentation and bounding box training data.

5.4 Evaluation

The segmentation taster task will be judged by average segmentation accuracy across the twenty classes and the background class. For VOC2008 the segmentation accuracy for a class will be assessed using the intersection/union metric, defined as the number of correctly labelled pixels of that class, divided by the number of pixels labelled with that class in either the ground truth labelling or the inferred labelling. Equivalently, the accuracy is given by the equation,

$$\text{segmentation accuracy} = \frac{\text{true positives}}{\text{true positives} + \text{false positives} + \text{false negatives}}$$

Code is provided to compute segmentation accuracies for each class, and the overall average accuracy (see section 8.5.2).

Participants are expected to submit a *single* set of results per method employed. Participants who have investigated several algorithms may submit one result per method. Changes in algorithm parameters do *not* constitute a different method – all parameter tuning must be conducted using the training and validation data alone.

6 Person Layout Taster

6.1 Task

For each 'person' object in a test image (their bounding box is provided) predict the presence/absence of parts (head/hands/feet), and the bounding boxes of those parts. Each prediction for a person layout should be output with an associated real-valued confidence of the layout so that a precision/recall curve can be drawn. Multiple estimates of layout may be output for each person, but estimates other than the first correct are treated as false positives as in the detection task.

As noted, the bounding box of the person is provided. To be considered a correct estimate of the layout, two conditions must be satisfied: i) correct prediction of parts present/absent; ii) correct prediction of bounding boxes for all parts.

6.2 Competitions

Two competitions are defined according to the choice of training data: (i) taken from the VOC `trainval` data provided, or (ii) from any source excluding the VOC `test` data provided:

No.	Task	Training data	Test data
6	Layout	`trainval`	`test`
7	Layout	**any but** VOC `test`	`test`

In competition 6, any annotation provided in the VOC `train` and `val` sets may be used for training, for example bounding boxes or particular views e.g. 'frontal' or 'left'. Participants are *not* permitted to perform additional manual annotation of either training or test data.

In competition 7, any source of training data may be used *except* the provided `test` images. Researchers who have pre-built systems trained on other data are particularly encouraged to participate. The test data includes images from "flickr" (`www.flickr.com`); this source of images may *not* be used for training. Participants who have acquired images from flickr for training must submit them to the organizers to check for overlap with the test set.

6.3 Submission of Results

To support the hierarchical (person+parts) nature of this task, an XML format has been adopted for submission of results. A separate XML file of results should be generated for each competition (6 or 7). The overall format should follow:

```
<results>
    <layout>
        ... layout estimate 1 ...
    </layout>
    <layout>
        ... layout estimate 2 ...
    </layout>
</results>
```

Each detection is represented by a `<layout>` element. The order of detections is not important. An example detection is shown here:

```
    <layout>
        <image>2008_000200</image>
        <object>3</object>
        <confidence>-8</confidence>
        <part>
            <class>head</class>
            <bndbox>
                <xmin>78</xmin>
                <ymin>82</ymin>
                <xmax>120</xmax>
                <ymax>136</ymax>
            </bndbox>
        </part>
        <part>
            <class>hand</class>
            <bndbox>
                <xmin>41</xmin>
                <ymin>151</ymin>
                <xmax>74</xmax>
                <ymax>190</ymax>
            </bndbox>
        </part>
        <part>
            <class>hand</class>
            <bndbox>
                <xmin>119</xmin>
                <ymin>146</ymin>
                <xmax>147</xmax>
                <ymax>184</ymax>
            </bndbox>
        </part>
    </layout>
```

The `<image>` element specifies the image identifier. The `<object>` specifies the index of the object to which the layout relates (the first object in the image has index 1) and should match that provided in the image set files (section 8.1.4). The `<confidence>` element specifies the confidence of the layout estimate, used to generate a precision/recall curve as in the detection task.

Each `<part>` element specifies the detection of a particular part of the person i.e. head/hand/foot. If the part is predicted to be absent/invisible, the corresponding element should be omitted. For each part, the `<class>` element specifies the type of part: `head`, `hand` or `foot`. The `<bndbox>` element specifies the predicted bounding box for that part; bounding boxes are specified in image co-ordinates and need not be contained in the provided person bounding box.

To ease creation of the required XML results file for MATLAB users, a function is included in the development kit to convert MATLAB structures to XML. See the `VOCwritexml` function (section 8.6.1). The example person layout implementation (section 7.2.4) includes code for generating a results file in the required format.

6.4 Evaluation

The person layout task will be judged by the precision/recall curve. The principal quantitative measure used will be the average precision (AP). Example code for computing the precision/recall and AP measure is provided in the development kit.

To be considered a true positive, each layout estimate must satisfy two criteria:

- set and number of predicted parts matches ground truth exactly e.g. {head, hand, hand} or {head, hand, foot}
- predicted bounding box of each part overlaps ground truth by at least 50%

The overlap between bounding boxes is computed as in the detection task. Note that in the case of multiple parts of the same type e.g. two hands, it is *not* necessary to predict which part is which.

7 Development Kit

The development kit is packaged in a single gzipped tar file containing MATLAB code and (this) documentation. The images, annotation, and lists specifying training/validation sets for the challenge are provided in a separate archive which can be obtained via the VOC web pages [1].

7.1 Installation and Configuration

The simplest installation is achieved by placing the development kit and challenge databases in a single location. After untarring the development kit, download the challenge image database and untar into the same directory, resulting in the following directory structure:

```
VOCdevkit/                           % development kit
VOCdevkit/VOCcode/                   % VOC utility code
VOCdevkit/results/VOC2008/           % your results on VOC2008
VOCdevkit/results/VOC2007/           % your results on VOC2007
VOCdevkit/local/                     % example code temp dirs
VOCdevkit/VOC2008/ImageSets          % image sets
VOCdevkit/VOC2008/Annotations        % annotation files
VOCdevkit/VOC2008/JPEGImages         % images
VOCdevkit/VOC2008/SegmentationObject % segmentations by object
VOCdevkit/VOC2008/SegmentationClass  % segmentations by class
```

If you set the current directory in MATLAB to the `VOCdevkit` directory you should be able to run the example functions:

- `example_classifier`

- `example_detector`
- `example_segmenter`
- `example_layout`

If desired, you can store the code, images/annotation, and results in separate directories, for example you might want to store the image data in a common group location. To specify the locations of the image/annotation, results, and working directories, edit the `VOCinit.m` file, e.g.

```
% change this path to point to your copy of the PASCAL VOC data
VOCopts.datadir='/homes/group/VOCdata/';

% change this path to a writable directory for your results
VOCopts.resdir='/homes/me/VOCresults/';

% change this path to a writable local directory for the example code
VOCopts.localdir='/tmp/';
```

Note that in developing your own code you need to include the `VOCdevkit/VOCcode` directory in your MATLAB path, e.g.

```
>> addpath /homes/me/code/VOCdevkit/VOCcode
```

7.2 Example Code

Example implementations are provided for all tasks. The aim of these (minimal) implementations is solely to demonstrate use of the code in the development kit.

7.2.1 Example Classifier Implementation

The file `example_classifier.m` contains a complete implementation of the classification task. For each VOC object class a simple classifier is trained on the `train` set; the classifier is then applied to the `val` set and the output saved to a results file in the format required by the challenge; a precision/recall curve is plotted and the 'average precision' (AP) measure displayed.

7.2.2 Example Detector Implementation

The file `example_detector.m` contains a complete implementation of the detection task. For each VOC object class a simple (and not very successful!) detector is trained on the `train` set; the detector is then applied to the `val` set and the output saved to a results file in the format required by the challenge; a precision/recall curve is plotted and the 'average precision' (AP) measure displayed.

7.2.3 Example Segmenter Implementation

An example segmenter is provided which converts detection results into segmentation results, using `create_segmentations_from_detections` (described below). For example:

```
>> example_detector;
>> example_segmenter;
```

This runs the example detector, converts the detections into segmentations and displays a table of per-class segmentation accuracies, along with an overall average accuracy.

7.2.4 Example Layout Implementation

The file `example_layout.m` contains a complete implementation of the person layout task. For each specified person a simple (and not very successful!) layout predictor is trained on the `train` set; the layout predictor is then applied to the `val` set and the output saved to a results file in the format required by the challenge; a precision/recall curve is plotted and the 'average precision' (AP) measure displayed.

7.3 Non-MATLAB Users

For non-MATLAB users, the file formats used for the VOC2008 data should be straightforward to use in other environments. Image sets (see below) are vanilla text files. Annotation files are XML format and should be readable by any standard XML parser. Images are stored in JPEG format, and segmentation ground truth in PNG format.

8 Using the Development Kit

The development kit provides functions for loading annotation data. Example code for computing precision/recall curves and segmentation accuracy, and for viewing annotation is also provided.

8.1 Image Sets

8.1.1 Classification/Detection Task Image Sets

The `VOC2008/ImageSets/Main/` directory contains text files specifying lists of images for the main classification/detection tasks.

The files `train.txt`, `val.txt`, `trainval.txt` and `test.txt` list the image identifiers for the corresponding image sets (training, validation, training+validation and testing). Each line of the file contains a single image identifier. The following MATLAB code reads the image list into a cell array of strings:

```
imgset='train';
ids=textread(sprintf(VOCopts.imgsetpath,imgset),'%s');
```

For a given image identifier `ids{i}`, the corresponding image and annotation file paths can be produced thus:

```
imgpath=sprintf(VOCopts.imgpath,ids{i});
annopath=sprintf(VOCopts.annopath,ids{i});
```

Note that the image sets used are the same for all classes. For each competition, participants are expected to provide output for all images in the `test` set.

8.1.2 Classification Task Image Sets

To simplify matters for participants tackling only the *classification* task, class-specific image sets with per-image ground truth are also provided. The file `VOC2008/ImageSets/Main/<class>_<imgset>.txt` contains image identifiers and ground truth for a particular class and image set, for example the file `car_train.txt` applies to the 'car' class and `train` image set.

Each line of the file contains a single image identifier and ground truth label, separated by a space, for example:

```
    2008_000002 -1
    2008_000005  0
    2008_000010 -1
```

The following MATLAB code reads the image list into a cell array of strings and the ground truth label into a corresponding vector:

```
imgset='train';
cls='car';
[ids,gt]=textread(sprintf(VOCopts.clsimgsetpath, ...
                    cls,imgset),'%s %d');
```

There are *three* ground truth labels:

- `-1`: Negative: The image contains no objects of the class of interest. A classifier should give a 'negative' output.
- `1`: Positive: The image contains at least one object of the class of interest. A classifier should give a 'positive' output.
- `0`: "Difficult": The image contains only objects of the class of interest marked as 'difficult'.

The "difficult" label indicates that all objects of the class of interest have been annotated as "difficult", for example an object which is clearly visible but difficult to recognize without substantial use of context. Currently the evaluation ignores such images, contributing nothing to the precision/recall curve or AP measure. The final evaluation may include separate results including such "difficult" images, depending on the submitted results. Participants are free to omit these images from training or include as either positive or negative examples.

8.1.3 Segmentation Taster Image Sets

The `VOC2008/ImageSets/Segmentation/` directory contains text files specifying lists of images for the segmentation taster task.

The files `train.txt`, `val.txt`, `trainval.txt` and `test.txt` list the image identifiers for the corresponding image sets (training, validation, training+validation and testing). Each line of the file contains a single image identifier. The following MATLAB code reads the image list into a cell array of strings:

```
imgset='train';
ids=textread(sprintf(VOCopts.seg.imgsetpath,imgset),'%s');
```

For a given image identifier `ids{i}`, file paths for the corresponding image, annotation, segmentation by object instance and segmentation by class can be produced thus:

```
imgpath=sprintf(VOCopts.imgpath,ids{i});
annopath=sprintf(VOCopts.annopath,ids{i});
clssegpath=sprintf(VOCopts.seg.clsimgpath,ids{i});
objsegpath=sprintf(VOCopts.seg.instimgpath,ids{i});
```

Participants are expected to provide output for all images in the `test` set.

8.1.4 Person Layout Taster Image Sets

The `VOC2008/ImageSets/Layout/` directory contains text files specifying lists of image for the person layout taster task.

The files `train.txt`, `val.txt`, `trainval.txt` and `test.txt` list the image identifiers for the corresponding image sets (training, validation, training+validation and testing). Each line of the file contains a single image identifier, and a single object index. Together these specify a 'person' object for which layout is provided or to be estimated, for example:

```
2008_000026 1
2008_000034 4
```

The following MATLAB code reads the image list into a cell array of strings and the object indices into a corresponding vector:

```
imgset='train';
[imgids,objids]=textread(sprintf(VOCopts.layout.imgsetpath, ...
                            VOCopts.trainset),'%s %d');
```

The annotation for the object (bounding box only in the test data) can then be obtained using the image identifier and object index:

```
    rec=PASreadrecord(sprintf(VOCopts.annopath,imgids{i}));
    obj=rec.objects(objids{i});
```

8.2 Development Kit Functions

8.2.1 `VOCinit`

The `VOCinit` script initializes a single structure `VOCopts` which contains options for the PASCAL functions including directories containing the VOC data and options for the evaluation functions (not to be modified).

The field `classes` lists the object classes for the challenge in a cell array:

```
VOCopts.classes={'aeroplane','bicycle','bird','boat',...
                 'bottle','bus','car','cat',...
                 'chair','cow','diningtable','dog',...
                 'horse','motorbike','person','pottedplant',...
                 'sheep','sofa','train','tvmonitor'};
```

The field `trainset` specifies the image set used by the example evaluation functions for training:

```
VOCopts.trainset='train'; % use train for development
```

Note that participants are free to use both training and validation data in any manner they choose for the final challenge.

The field `testset` specifies the image set used by the example evaluation functions for testing:

```
VOCopts.testset='val'; % use validation data for development
```

Other fields provide, for convenience, paths for the image and annotation data and results files. The use of these paths is illustrated in the example implementations.

Running on VOC2007 test set. The flag `VOC2007` defined at the start of the `VOCinit.m` script specifies whether the VOC2007 or VOC2008 data should be used. This changes the directories used for image sets and images and the results directory. To run on the VOC2007 test set, set the flag to "true" as indicated in the script. Note that the intention is to *train* on VOC2008 data and *test* on VOC2007 data.

8.2.2 `PASreadrecord(filename)`

The `PASreadrecord` function reads the annotation data for a particular image from the annotation file specified by `filename`, for example:

```
>> rec=PASreadrecord(sprintf(VOCopts.annopath,'2008_000026'))

rec =

       folder: 'VOC2008'
     filename: '2008_000026.jpg'
       source: [1x1 struct]
        owner: [1x1 struct]
         size: [1x1 struct]
    segmented: 0
      imgname: 'VOC2008/JPEGImages/2008_000026.jpg'
      imgsize: [500 375 3]
     database: 'The VOC2008 Database'
      objects: [1x2 struc
```

The `imgname` field specifies the path (relative to the main VOC data path) of the corresponding image. The `imgsize` field specifies the image dimensions as `(width,height,depth)`. The `database` field specifies the data source (VOC2008). The `segmented` field specifies if a segmentation is available for this image. The `folder` and `filename` fields provide an alternative specification of the image path, and `size` an alternative specification of the image size:

```
>> rec.size

ans =

     width: 500
```

```
    height: 375
     depth: 3
```

The `source` field contains additional information about the source of the image e.g. web-site and owner. This information is obscured until completion of the challenge.

Objects annotated in the image are stored in the struct array `objects`, for example:

```
>> rec.objects(1)

ans =

        class: 'person'
         view: 'Frontal'
    truncated: 1
     occluded: 1
    difficult: 0
        label: 'PASpersonFrontalTruncOcc'
     orglabel: 'PASpersonFrontalTruncOcc'
         bbox: [122.1875 7.8125 371.5625 375]
       bndbox: [1x1 struct]
      polygon: []
         mask: []
     hasparts: 1
         part: [1x3 struct]
```

The `class` field contains the object class. The `view` field contains the view: `Frontal`, `Rear`, `Left` (side view, facing left of image), `Right` (side view, facing right of image), or an empty string indicating another, or un-annotated view.

The `truncated` field being set to 1 indicates that the object is "truncated" in the image. The definition of truncated is that the bounding box of the object specified does not correspond to the full extent of the object e.g. an image of a person from the waist up, or a view of a car extending outside the image. Participants are free to use or ignore this field as they see fit.

The `occluded` field being set to 1 indicates that the object is significantly occluded by another object. Participants are free to use or ignore this field as they see fit.

The `difficult` field being set to 1 indicates that the object has been annotated as "difficult", for example an object which is clearly visible but difficult to recognize without substantial use of context. Currently the evaluation ignores such objects, contributing nothing to the precision/recall curve. The final evaluation may include separate results including such "difficult" objects, depending on the submitted results. Participants may include or exclude these objects from training as they see fit.

The `bbox` field specifies the bounding box of the object in the image, as `[left,top,right,bottom]`. The top-left pixel in the image has coordinates $(1, 1)$. The `bndbox` field specifies the bounding box in an alternate form:

```
>> rec.objects(1).bndbox
```

```
ans =

    xmin: 122.1875
    ymin: 7.8125
    xmax: 371.5625
    ymax: 375
```

For backward compatibility, the `label` and `orglabel` fields specify the PASCAL label for the object, comprised of class, view and truncated/difficult flags. The `polygon` and `mask` specify polygon/per-object segmentations, and are not provided for the VOC2008 data.

The `hasparts` field specifies if the object has sub-object "parts" annotated. For the VOC2008 data, such annotation is available for a subset of the 'person' objects, used in the layout taster task. Object parts are stored in the struct array `part`, for example:

```
>> rec.objects(1).part(1)

ans =

        class: 'head'
         view: ''
    truncated: 0
     occluded: 0
    difficult: 0
        label: 'PAShead'
     orglabel: 'PAShead'
         bbox: [193.7966 9.5616 285.6810 130.8302]
       bndbox: [1x1 struct]
      polygon: []
         mask: []
     hasparts: 0
         part: []
```

The format of object parts is identical to that for top-level objects. For the 'person' parts in the VOC2008 data, parts are not annotated with view, or truncated/difficult flags. The bounding box of a part is specified in image coordinates in the same way as for top-level objects. Note that the object parts may legitimately extend outside the bounding box of the parent object.

8.2.3 `viewanno(imgset)`

The `viewanno` function displays the annotation for images in the image set specified by `imgset`. Some examples:

```
>> viewanno('Main/train');
>> viewanno('Main/car_val');
>> viewanno('Layout/train');
>> viewanno('Segmentation/val');
```

8.3 Classification Functions

8.3.1 `VOCevalcls(VOCopts,id,cls,draw)`

The `VOCevalcls` function performs evaluation of the classification task, computing a precision/recall curve and the average precision (AP) measure. The arguments `id` and `cls` specify the results file to be loaded, for example:

```
>> [rec,prec,ap]=VOCevalcls(VOCopts,'comp1','car',true);
```

See `example_classifier` for further examples. If the argument `draw` is true, the precision/recall curve is drawn in a figure window. The function returns vectors of recall and precision rates in `rec` and `prec`, and the average precision measure in `ap`.

8.4 Detection Functions

8.4.1 `VOCevaldet(VOCopts,id,cls,draw)`

The `VOCevaldet` function performs evaluation of the detection task, computing a precision/recall curve and the average precision (AP) measure. The arguments `id` and `cls` specify the results file to be loaded, for example:

```
>> [rec,prec,ap]=VOCevaldet(VOCopts,'comp3','car',true);
```

See `example_detector` for further examples. If the argument `draw` is true, the precision/recall curve is drawn in a figure window. The function returns vectors of recall and precision rates in `rec` and `prec`, and the average precision measure in `ap`.

8.4.2 `viewdet(id,cls,onlytp)`

The `viewdet` function displays the detections stored in a results file for the detection task. The arguments `id` and `cls` specify the results file to be loaded, for example:

```
>> viewdet('comp3','car',true)
```

If the `onlytp` argument is true, only the detections considered true positives by the VOC evaluation measure are displayed.

8.5 Segmentation Functions

8.5.1 `create_segmentations_from_detections(id,confidence)`

This function creates segmentation results from detection results.

`create_segmentations_from_detections(id)` creates segmentations from the detection results with specified identifier e.g. comp3. This is achieved by rendering the bounding box for each detection in class order, so that later classes overwrite earlier classes (e.g. a person bounding box will overwrite an overlapping an aeroplane bounding box). All detections will be used, no matter what their confidence level.

`create_segmentations_from_detections(id,confidence)` does the same, but only detections above the specified confidence will be used.

See `example_segmenter` for an example.

8.5.2 `VOCevalseg(VOCopts,id)`

The `VOCevalseg` function performs evaluation of the segmentation task, computing a confusion matrix and segmentation accuracies for the segmentation task. It returns per-class percentage accuracies, the average overall percentage accuracy, and a confusion matrix, for example:

```
>> [accuracies,avacc,conf,rawcounts] = VOCevalseg(VOCopts,'comp3')
```

Accuracies are defined by the intersection/union measure. The optional fourth output 'rawcounts' returns an un-normalized confusion matrix containing raw pixel counts. See `example_segmenter` for another example. This function will also display a table of overall and per-class accuracies.

8.5.3 `VOClabelcolormap`(N)

The `VOClabelcolormap` function creates the color map which has been used for all provided indexed images. You should use this color map for writing your own indexed images, for consistency. The size of the color map is given by N, which should generally be set to 256 to include a color for the 'void' label.

8.6 Layout Functions

8.6.1 `VOCwritexml`(rec,path)

The `VOCwritexml` function writes a MATLAB structure array to a corresponding XML file. It is provided to support the creation of XML results files for the person layout taster. An example of usage can be found in `example_layout`.

8.6.2 `VOCevallayout(VOCopts,id,draw)`

The `VOCevallayout` function performs evaluation of the person layout task, computing a precision/recall curve and the average precision (AP) measure. The arguments `id` and `cls` specify the results file to be loaded, for example:

```
>> [rec,prec,ap]=VOCevallayout(VOCopts,'comp6',true);
```

See `example_layout` for further examples. If the argument `draw` is true, the precision/recall curve is drawn in a figure window. The function returns vectors of recall and precision rates in `rec` and `prec`, and the average precision measure in `ap`.

Acknowledgements

We gratefully acknowledge the following, who spent many long hours providing annotation for the VOC2008 database: Jan-Hendrik Becker, Patrick Buehler, Kian Ming Chai, Miha Drenik, Chris Engels, Jan Van Gemert, Hedi Harzallah, Nicolas Heess, Zdenek Kalal, Lubor Ladicky, Marcin Marszalek, Alastair Moore, Maria-Elena Nilsback, Paul Sturgess, David Tingdahl, Hirofumi Uemura, Martin Vogt.

The preparation and running of this challenge is supported by the EU-funded PASCAL Network of Excellence on Pattern Analysis, Statistical Modelling and Computational Learning.

References

[1] The PASCAL Visual Object Classes Challenge (VOC2008). `http://www.pascal-network.org/challenges/VOC/voc2008/index.html`.