The PASCAL Visual Object Classes Challenge 2006 Development Kit

Mark Everingham

February 14, 2006

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 ten object classes:

- bicycle, bus, car, motorbike
- cat, cow, dog, horse, sheep
- person

There are two tasks:

1.1 Classification Task

For each of the ten 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 an ROC curve can be drawn.

1.2 Detection Task

For each of the ten 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.

1.3 Image Sets

There are four sets of images provided, for use in both the classification and detection tasks.

`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`.

Table 1: Statistics of the image sets

	train		val		trainval		test	
	img	**obj**	**img**	**obj**	**img**	**obj**	**img**	**obj**
Bicycle	127	161	143	162	270	323	268	326
Bus	93	118	81	117	174	235	180	233
Car	271	427	282	427	553	854	544	854
Cat	192	214	194	215	386	429	388	429
Cow	102	156	104	157	206	313	197	315
Dog	189	211	176	211	365	422	370	423
Horse	129	164	118	162	247	326	254	324
Motorbike	118	138	117	137	235	275	234	274
Person	319	577	347	579	666	1156	675	1153
Sheep	119	211	132	210	251	421	238	422
Total	1277	2377	1341	2377	2618	4754	2686	4753

`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 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. In total there are 5,304 images, containing 9,507 annotated objects.

1.3.1 Ground Truth Annotation

Objects of the ten 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.
- **'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 dificult 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 reproduced in Appendix A.

1.4 Competitions

Four competitions are defined according to the task and 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`
3	Detection	`trainval`	`test`
4	Detection	**any but** VOC `test`	`test`

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 'side'. Participants are free to perform manual annotation on the training data if they wish. Manual annotation of the test data to optimize algorithm performance is *not* permitted.

In competitions 2 and 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 the Microsoft Research Cambridge object recognition database, and "flickr" (`www.flickr.com`); these sources 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.

For each competition, participants may choose to tackle all, or any subset of object classes, for example "cars only" or "motorbikes and cars".

1.5 Submission of Results

In contrast to the 2005 VOC challenge, ground truth for the test images will *not* be released. Participants are therefore required to submit the raw output of their classifier and detector implementations, rather than ROC and precision/recall curves as in the 2005 challenge.

Results for each competition are to be submitted as text files. Details on the submission procedure will be given at a later date. The file formats required for the classification and detection tasks are described here:

1.5.1 Classification Results

For the classification task, 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_val_car.txt:
    000004 0.702732
    000006 0.870849
    000008 0.532489
    000018 0.477167
    000019 0.112426
```

Greater confidence values signify greater confidence that the image contains an object of the class of interest. The example classifier implementation (section 4.1) includes code for generating a results file in the required format.

1.5.2 Detection Results

For the detection task, 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 `000006`, multiple objects are detected:

```
comp3_det_val_car.txt:
    000004 0.702732 89 112 516 466
    000006 0.870849 373 168 488 229
    000006 0.852346 407 157 500 213
    000006 0.914587 2 161 55 221
    000008 0.532489 175 184 232 201
```

The example detector implementation (section 4.2) includes code for generating a results file in the required format.

1.6 Evaluation

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.

1.6.1 Classification Task

The classification task will be judged by the Receiver Operating Characteristic (ROC) curve. The principal quantitative measure used will be the area under curve (AUC). Example code for computing the ROC and AUC measure is provided in the development kit.

Images which contain only objects marked as 'difficult' (section 1.3.1) are currently *ignored* by the evaluation. The final evaluation may include separate results including such "difficult" images, depending on the submitted results.

1.6.2 Detection Task

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 1.3.1) are currently *ignored* by the evaluation. The final evaluation may include separate results including such "difficult" images, depending on the submitted results.

2 Development Kit Contents

The development kit is packaged in a single gzipped tar file containing 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.

3 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/              % PASCAL/VOC utility code
VOCdevkit/results/              % directory for your results
VOCdevkit/local/                % directory for example code
VOCdevkit/VOC2006/ImageSets     % image sets
VOCdevkit/VOC2006/Annotations   % annotation files
VOCdevkit/VOC2006/PNGImages     % images
```

If you set the current directory in MATLAB to the `VOCdevkit` directory you should be able to run the example functions `example_classifier` and `example_detector`.

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
```

4 Example Code

Example implementations for both the classification and detection tasks are provided. The aim of these implementations is solely to demonstrate use of the code in the development kit.

4.1 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 Receiver Operating Characteristic (ROC) curve is plotted and the 'area under curve' (AUC) measure displayed.

4.2 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.

5 Using the Development Kit

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

5.1 Image Set Files

The `VOC2006/ImageLists/` directory contains text files specifying lists of images for the `train` and `val` sets. Corresponding files for the test set will be provided at a later date.

5.1.1 Image Sets

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

```
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.

5.1.2 Classification Task Image Sets

To simplify matters for participants tackling only the *classification* competition, class-specific image sets with per-image ground truth are also provided. The file `<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:

```
000601 -1
000604  0
000610  1
```

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 output of the classifier for this image does not affect its evaluation.

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 ROC curve or AUC measure. The final evaluation may include separate results including such "difficult" images, depending on the submitted results. Participants are free to include these images in training as either positive or negative examples.

5.2 Development Kit Functions

5.2.1 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={'bicycle','bus','car','cat','cow','dog',...
                 'horse','motorbike','person','sheep'};
```

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 classifier and detector implementations.

5.3 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,'000008'))

rec =

     imgname: 'VOC2006/PNGImages/000008.png'
     imgsize: [500 375 3]
    database: 'The VOC2006 Database'
     objects: [1x2 struct]
```

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

```
>> rec.objects(1)

ans =

        label: 'PAShorseTrunc'
     orglabel: 'PAShorseTrunc'
         bbox: [267 72 498 326]
      polygon: []
         mask: ''
        class: 'horse'
         view: ''
    truncated: 1
    difficult: 0
```

The label field specifies the PASCAL label for the object. For convenience, the class field contains the corresponding VOC challenge 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 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 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 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.

5.4 VOCroc(VOCopts,id,cls,draw)

The VOCroc function computes an ROC curve for the classification task. The arguments id and cls specify the results file to be loaded, for example:

```
>> [fp,tp,auc]=VOCroc(VOCopts,'comp1','car',true);
```

See example_classifier for further examples. If the argument draw is true, the ROC curve is drawn in a figure window. The function returns vectors of false and true positive rates in fp and tp, and the area under curve (AUC) measure in auc.

5.5 VOCpr(VOCopts,id,cls,draw)

The VOCpr function computes a precision/recall curve for the detection task. The arguments id and cls specify the results file to be loaded, for example:

```
>> [rec,prec,ap]=VOCpr(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.

5.6 viewanno(imgset)

The viewanno function displays the annotation for images in the image set specified by imgset. The image set may be one of 'train', 'trainval' or 'val', or include a class, for example 'car_train':

```
>> viewanno('train');
>> viewanno('car_train');
```

5.7 `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.

A Annotation Guidelines

This appendix lists the guidelines on annotation which were given to annotators.

A.1 Guidelines on what and how to label

What to label. All objects of the defined categories, unless:

- you are unsure what the object is.
- the object is very small (at your discretion).
- less than 10-20% of the object is visible.

If this is not possible because of too many objects, mark the image as bad.

Viewpoint. Record the viewpoint of the 'bulk' of the object e.g. the body rather than the head. Allow viewpoints within 10-20 degrees. If ambiguous, leave as 'Unspecified'.

Bounding box. Mark the bounding box of the visible area of the object (not the estimated total extent of the object). The bounding box should contain all visible pixels, except where the bounding box would have to be made excessively large to include a few additional pixels ($< 5\%$) e.g. a car aerial.

Occlusion/truncation. If more than 15-20% of the object is occluded and lies outside the bounding box, mark as 'Truncated'. Do not mark as truncated if the occluded area lies within the bounding box.

Image quality/illumination. Images which are poor quality (e.g. excessive motion blur) should be marked bad. However, poor illumination (e.g. objects in silhouette) should not count as poor quality unless objects cannot be recognized.

Clothing/mud/snow etc. If an object is 'occluded' by a close-fitting occluder e.g. clothing, mud, snow etc., then the occluder should be treated as part of the object.

Transparency. Do label objects visible through glass, but treat reflections on the glass as occlusion.

Mirrors. Do label objects in mirrors.

Pictures. Label objects in pictures/posters/signs only if they are photorealistic but not if cartoons, symbols etc.

A.2 Guidelines on categorization

Car. Includes cars, vans, people carriers etc. Do not label where only the vehicle interior is shown.

A.3 "Difficult" flag

Objects were marked as "difficult" by a single annotator. Only the image area corresponding to the bounding box of each object was displayed and a subjective judgement of the difficulty of recognizing the object was made. Reasons for marking an object as difficult included small image area, blur, clutter, high level of occlusion, occlusion of a very characteristic part of the object, etc.