The Pascal Visual Object Classes Challenge Development Kit

March 16, 2005

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 four object classes:

- motorbikes
- bicycles
- people
- cars

There are two tasks:

1.1 Classification task

For each of the four 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 four 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. 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:

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

MATLAB code for computing this overlap measure is present in the example code. Multiple detections of the *same* object in an image will be 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 user's system to filter multiple detections from its output.

1.3 Image sets

There are five sets of images provided. The image sets are to be used both for the classification and detection tasks.

`train`:	Training data
`val`:	Validation data (suggested). The validation data may be used as additional training data (see below).
`train+val`:	The union of `train` and `val`.
`test1`:	First test set. This test set is taken from the same distribution of images as the training and validation data, and is expected to provide an 'easier' challenge.
`test2`:	Second test set. This test set has been freshly collected for the challenge. It is not therefore expected to have the same distribution as the training data, and should provide a 'harder' challenge.

1.4 Competitions

Eight competitions are defined according to the task, the choice of training data: (i) taken from the VOC `train+val` data provided, or (ii) from any source excluding the VOC {`test1|test2`} data provided; and the choice of test data: (i) `test1` ('easier') or (ii) `test2` ('harder'):

No.	Task	Training data	Test data
1	Classification	`train+val`	`test1`
2	Classification	`train+val`	`test2`
3	Classification	**not** VOC `test1` or `test2`	`test1`
4	Classification	**not** VOC `test1` or `test2`	`test2`
5	Detection	`train+val`	`test1`
6	Detection	`train+val`	`test2`
7	Detection	**not** VOC `test1` or `test2`	`test1`
8	Detection	**not** VOC `test1` or `test2`	`test2`

To emphasize, in competitions 3–4 and 7–8, *any* source of training data may be used *except* the provided test data `test1` or `test2`. Competitions 1–2 and 5–6 must use *only* the provided test data `train` and `val`.

Note that any annotation provided in the VOC `train` and `val` sets may be used for training, for example bounding boxes, particular class labels e.g. `PAScarFrontal` or `PAScarSide`, polygonal outlines where provided, etc.

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

2 Development Kit Contents

The development kit is packaged in two gzipped tar files, one containing images and annotation in the standard PASCAL format, and one containing code, image lists specifying training/validation sets, and (this) documentation.

3 Installation and Configuration

The simplest installation is achieved by untarring the two development kit files to a single location, resulting in the following directory structure:

```
    VOCdevkit/                  % documentation and example code
    VOCdevkit/PASCAL/           % PASCAL/VOC utility code
    VOCdevkit/PASCAL/imgsets/   % image sets (internal use only)
    VOCdevkit/VOCdata/          % VOC images and annotation
    VOCdevkit/results/          % directory for your results
```

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 and results directories, edit the `VOCinit.m` file, e.g.

```
% change this path to point to your copy of the PASCAL images
PASopts.imgdir='/homes/group/VOCdata/';

% change this path to a writable directory for your results
PASopts.resultsdir='/homes/me/VOCresults/';
```

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

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

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. The code has been written in such a manner that you should be able to incorporate your own classifier/detector by replacing just a few lines of code.

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+val` set; the classifier is then applied to the 'easier' test set `test1` and an ROC curve plotted. The classifier output and ROC curve is saved in the format required for submitting results to PASCAL.

Incorporation of your own classifier code can be achieved by replacing the definitions of the `example_train` and `example_classify` functions. Of course, you are free to use whatever other methods may be appropriate, but results for submission to PASCAL must be produced by calling the `VOCroc` function.

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+val` set; the detector is then applied to the 'easier' test set `test1` and a precision/recall curve plotted. The precision/recall curve is saved in the format required for submitting results to PASCAL.

Incorporation of your own detector code can be achieved by replacing the definitions of the `example_train` and `example_detect` functions. Of course, you are free to use whatever other methods may be appropriate, but results for submission to PASCAL must be produced by calling the `VOCpr` function.

5 Using the Development Kit

The development kit provides functions for loading annotation data related to the VOC challenge and generating ROC and precision/recall curves for submission to PASCAL.

5.1 `VOCinit`

The `VOCinit` script initializes a single structure `PASopts` which contains options for the PASCAL functions including directories containing the PASCAL data, options for the evaluation functions (not to be modified), and the list of object classes for the VOC challenge.

The field `VOCclass` is a structure array listing the VOC object classes. Each element has a field `label` which is the VOC label string for the class, and a field `PASlabels` which is the list of PASCAL labels from which the VOC class is derived e.g.

```
>> PASopts.VOCclass(1)

ans =

        label: 'VOCmotorbikes'
    PASlabels: {'PASmotorbike'  'PASmotorbikeSide'}
```

The `label` field is used to identify results to the evaluation code and for submitting results to PASCAL, see below and the example code.

5.2 `VOCreadimgset(PASopts,label,subset)`

The `VOCreadimgset` function loads the annotation data associated with a particular image set i.e. the training/validation/test set for a particular VOC object class. The argument `label` specifies the VOC class label e.g. `'VOCmotorbikes'` and the argument `subset` specifies the subset of data which should be one of `'train|val|train+val|test1|test2'`. For example, to load the training set for the `VOCmotorbikes` class, use:

```
>> imgset=VOCreadimgset(PASopts,'VOCmotorbikes','train')

imgset =
      label: 'VOCmotorbikes'
     subset: 'train'
       recs: [1x342 struct]
    posinds: [1x107 double]
    neginds: [1x235 double]
```

The returned value is a structure with the label and subset strings preserved. The `recs` field is an array of PASCAL image annotation records. For convenience, the `posinds` field lists the indices of 'positive' records which contain at least one instance of the object class of interest (see below), and the `neginds` field lists the corresponding 'negative' records in which no objects of interest are present.

Each element of the `recs` field is a standard PASCAL image annotation record (see the PASCAL annotation files for details) e.g.

```
>> imgset.recs(1)

ans =

     imgname: 'Caltech/PNGImages/motorbikes_side/0002.png'
     imgsize: [440 280 3]
    database: 'The Caltech Database'
     objects: [1x1 struct]
     present: 1
```

For convenience, the additional `present` field indicates if an instance of the object class of interest (e.g. `VOCmotorbikes` here) is present in the image. The `objects` field contains a structure array containing annotation for each object of interest present in the image (if any) e.g.

```
>> imgset.recs(1).objects

ans =

       label: 'PASmotorbikeSide'
    orglabel: 'motorbikes'
        bbox: [11 13 431 270]
     polygon: []
        mask: ''
```

Note that the only objects listed are instances of the object class of interest (e.g. `VOCmotorbikes` here) i.e. `recs(i).present=length(recs(i).objects)>0`.

5.3 `VOCroc(PASopts,imgset,confidence,draw)`

The `VOCroc` function computes an ROC curve for the object *classification* task in which the aim is to determine solely the presence of absence of an object class in an image.

The `imgset` argument specifies the image set loaded using `VOCreadimgset`. The `draw` argument can be set nonzero to draw the ROC curve on the current figure.

The `confidence` argument specifies the output of the user's classifier for the image set `imgset`. It should contain one element per image with greater values indicating greater confidence that an instance of the object class of interest is present e.g. something like $P(object\_present|image)$.

The return value from the function is a structure containing the vector of confidence values input to the function and a vector of false positives `fp` and true positives `tp`, with the identity of the class label and image set preserved e.g.

```
>> roc=VOCroc(PASopts,valset,confidence,true)

roc =

         label: 'VOCmotorbikes'
        subset: 'val'
    confidence: [342x1 double]
            tp: [1x342 double]
            fp: [1x342 double]
```

5.4 VOCsaveroc(PASopts,roc,expt)

The `VOCsaveroc` saves the ROC curve computed by `VOCroc` to a file in the format required for submission to PASCAL e.g.

```
>> VOCsaveroc(PASopts,roc,'pascal_develtest')
```

The `expt` argument specifies a string identifier for the experiment which the user is free to choose. The output file is stored in the user's specified results directory (see `VOCinit`) and named automatically according to the experiment, VOC class label and experiment.

5.5 VOCpr(PASopts,imgset,dets,draw)

The `VOCpr` function computes a precision/recall curve for the object *detection* task in which the aim is to determine both the presence and bounding boxes of instances of a particular object class in an image.

The `imgset` argument specifies the image set loaded using `VOCreadimgset`. The `draw` argument can be set nonzero to draw the precision/recall curve on the current figure.

The `dets` argument specifies the output of the user's detector for the image set `imgset`. It should be an array of structures with fields `imgnum`, `confidence`, and `bbox` e.g.

```
>> dets

dets =

1x342 struct array with fields:
    imgnum
    confidence
    bbox
```

The `imgnum` field specifies the index of the image in the image set `imgset` to which the detection applies. The `confidence` field indicates confidence in the detection, with greater values indicating greater confidence that the object is present at the specified position e.g. something like $P(object\_present|bbox)$. The `bbox` field specifies the bounding box of the detection in the image (in MATLAB image coordinates) as a vector of the form `[xmin ymin xmax ymax]` (coordinates need not be integer valued). An example:

```
>> dets(1)

ans =

        imgnum: 1
    confidence: 3.9953
          bbox: [32.6162 -4.2666 413.6433 229.2844]
```

The return value from the function is a structure containing a vector of recall and corresponding precision values, with the identity of the class label and image set preserved e.g.

```
>> pr=VOCpr(PASopts,valset,dets,true)

pr =

        label: 'VOCmotorbikes'
       subset: 'val'
       recall: [342x1 double]
    precision: [342x1 double]
```

5.6 VOCsaveroc(PASopts,pr,expt)

The `VOCsavepr` saves the precision/recall curve computed by `VOCpr` to a file in the format required for submission to PASCAL e.g.

```
>> VOCsavepr(PASopts,pr,'pascal_develtest')
```

The `expt` argument specifies a string identifier for the experiment which the user is free to choose. The output file is stored in the user's specified results directory (see `VOCinit`) and named automatically according to the experiment, VOC class label and experiment.