The PASCAL Visual Object Classes Challenge 2010 (VOC2010) Development Kit

Mark Everingham John Winn

May 8, 2010

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 three 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).
- Segmentation: For each pixel in a test image, predict the class of the object containing that pixel or 'background' if the pixel does not belong to one of the twenty specified classes.

In addition, there are two "taster" tasks:

- 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.
- Action Classification: For each of the action classes predict if a specified person (indicated by their bounding box) in a test image is performing the corresponding action. There are nine action classes:
  - phoning; playing a musical instrument; reading; riding a bicycle or motorcycle; riding a horse; running; taking a photograph; using a computer; walking

Finally a taster on large scale visual recognition is being run by the ImageNet organizers. Further details can be found at their website: `http://www.image-net.org/challenges/LSVRC/2010/index`.

2 Data

The VOC2010 database contains a total of 21,738 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.

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	283	369	296	369	579	738	–	–
Bicycle	228	305	243	309	471	614	–	–
Bird	340	486	326	485	666	971	–	–
Boat	222	345	210	342	432	687	–	–
Bottle	300	507	283	507	583	1014	–	–
Bus	180	245	173	253	353	498	–	–
Car	523	892	507	882	1030	1774	–	–
Cat	502	563	503	569	1005	1132	–	–
Chair	469	946	456	944	925	1890	–	–
Cow	125	228	123	236	248	464	–	–
Diningtable	209	234	206	234	415	468	–	–
Dog	591	707	608	709	1199	1416	–	–
Horse	209	306	216	315	425	621	–	–
Motorbike	225	306	228	305	453	611	–	–
Person	1717	3559	1831	3737	3548	7296	–	–
Pottedplant	225	408	225	413	450	821	–	–
Sheep	152	344	138	357	290	701	–	–
Sofa	205	224	201	227	406	451	–	–
Train	226	261	227	263	453	524	–	–
Tvmonitor	247	342	243	341	490	683	–	–
Total	4998	11577	5105	11797	10103	23374	–	–

2.1 Classification/Detection Image Sets

For the classification and detection tasks 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 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. To increase the amount of data, the dataset includes images from the 2008/2009 datasets. The assignment of images to training/test sets follows the 2008/2009 assignments i.e. the 2008/2009 training/test sets are a subset of the corresponding 2010 sets. Note that no annotation for the 2008/2009 test sets has been released.

Table 2: Statistics of the segmentation image sets.

	train		val		trainval		test	
	img	**obj**	**img**	**obj**	**img**	**obj**	**img**	**obj**
Aeroplane	59	70	57	66	116	136	–	–
Bicycle	49	62	48	62	97	124	–	–
Bird	64	86	72	90	136	176	–	–
Boat	56	89	50	64	106	153	–	–
Bottle	55	94	62	82	117	176	–	–
Bus	48	64	51	83	99	147	–	–
Car	83	130	78	146	161	276	–	–
Cat	85	102	84	93	169	195	–	–
Chair	94	194	76	154	170	348	–	–
Cow	38	85	44	76	82	161	–	–
Diningtable	54	55	50	53	104	108	–	–
Dog	78	97	78	97	156	194	–	–
Horse	49	67	56	69	105	136	–	–
Motorbike	55	61	48	65	103	126	–	–
Person	260	451	266	474	526	925	–	–
Pottedplant	54	83	61	126	115	209	–	–
Sheep	41	92	42	117	83	209	–	–
Sofa	54	62	63	76	117	138	–	–
Train	50	57	56	61	106	118	–	–
Tvmonitor	59	74	54	74	113	148	–	–
Total	964	2075	964	2128	1928	4203	–	–

2.2 Segmentation Image Sets

For the segmentation 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 2007–2009 segmentation tasters. The test set contains only 2008–2010 images (i.e. those for which no annotation has been released), 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 2007–2010 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, and any annotation provided for the main challenge e.g. bounding boxes.

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 task, the training and validation image sets include images from the 2007–2009 person layout tasters. The test set contains only 2008–2010 images (i.e. those for which no annotation has been released), and is disjoint

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	207	296	169	260	376	556	–	–

Table 4: Statistics of the action classification taster image sets.

	train		val		trainval		test	
	img	**obj**	**img**	**obj**	**img**	**obj**	**img**	**obj**
Phoning	25	25	25	26	50	51	–	–
Playinginstrument	27	38	27	38	54	76	–	–
Reading	25	26	26	27	51	53	–	–
Ridingbike	25	33	25	33	50	66	–	–
Ridinghorse	27	35	26	36	53	71	–	–
Running	26	47	25	47	51	94	–	–
Takingphoto	25	27	26	28	51	55	–	–
Usingcomputer	26	29	26	30	52	59	–	–
Walking	25	41	26	42	51	83	–	–
Total	226	301	228	307	454	608	–	–

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 Action Classification Taster Image Sets

For the action classification 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 the set of actions they are performing from the set {phoning, playing a musical instrument, reading, riding a bicycle or motorcycle, riding a horse, running, taking a photograph, using a computer, walking}. The image sets are disjoint from those of the main task and person layout taster task. Note that they are *not* fully annotated – only 'person' objects forming part of the training and test sets are annotated. Table 4 summarizes the action statistics for each image set.

2.5 Ground Truth Annotation

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

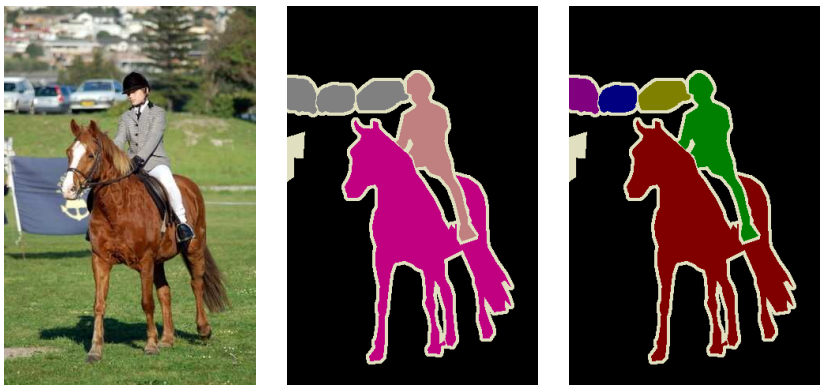
- **class**: the object class e.g. 'car' or 'bicycle'
- **bounding box**: an axis-aligned rectangle specifying the extent of the object visible in the image.

- **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].

Note that for the action classification taster images, only people have been annotated, and only the bounding box is available. Note also that for these images the annotation is not necessarily complete i.e. there may be unannotated people.

2.6 Segmentation Ground Truth



a. b. c.

Figure 1: Example of segmentation 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.

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 1 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 e.g. bounding boxes.

2.7 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.

2.8 Action Classification Taster Ground Truth

For the action classification taster task, 'person' objects are annotated with bounding box and a set of flags, one per action class e.g. 'phoning' or 'walking'. For each action the flag indicates if the person is performing the corresponding action. Note that actions are not mutually exclusive, for example a person may simultaneously be walking and phoning. The image sets are disjoint from the main tasks and layout taster tasks. 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:
    ...
    2009_000001 0.056313
    2009_000002 0.127031
    2009_000009 0.287153
    ...
```

Greater confidence values signify greater confidence that the image contains an object of the class of interest. The example classifier implementation (section 8.2.1) 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. See also section 3.4.1.

Images which contain only objects marked as 'difficult' (section 2.5) 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.

3.4.1 Average Precision (AP)

The computation of the average precision (AP) measure has been changed in VOC2010 to improve precision and ability to measure differences between methods with low AP. It is computed as follows:

1. Compute a version of the measured precision/recall curve with precision monotonically decreasing, by setting the precision for recall r to the maximum precision obtained for any recall $r' \geq r$.

2. Compute the AP as the area under this curve by numerical integration. No approximation is involved since the curve is piecewise constant.

Note that in previous years the AP is computed by sampling the monotonically decreasing curve at a fixed set of uniformly-spaced recall values $0, 0.1, 0.2, \ldots, 1$. By contrast, VOC2010 effectively samples the curve at *all* unique recall values.

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 `2009_000032`, multiple objects are detected:

```
comp3_det_test_car.txt:
    ...
    2009_000026 0.949297 172.000000 233.000000 191.000000 248.000000
    2009_000032 0.013737 1.000000 147.000000 114.000000 242.000000
    2009_000032 0.013737 1.000000 134.000000 94.000000 168.000000
    2009_000035 0.063948 455.000000 229.000000 491.000000 243.000000
    ...
```

The example detector implementation (section 8.2.2) 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) (see section 3.4.1). 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 2.5) 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 Task

5.1 Task

For each test image pixel, predict the class of the object containing that pixel or 'background' if the pixel 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

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
5	Segmentation	`trainval`	`test`
6	Segmentation	**any but** VOC `test`	`test`

For competition 5, 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'. However, if training uses annotation of any images other than the segmented training images, this must be reported as part of the submission (see below) since this allows a considerably larger training set. Participants are *not* permitted to perform additional manual annotation of either training or test data.

For competition 6, any source of training data may be used *except* the provided `test` images.

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 PNG color map should be the same as the color map used in the provided training and validation annotation (MATLAB users can use `VOClabelcolormap` – see section 9.5.3). The example segmenter implementation (section 8.2.3) includes code for generating results in the required format. Participants may choose to include segmentations for only a subset of the 20 classes in which case they will be evaluated on only the included classes.

For competition 5, along with the submitted image files, participants **must** also state whether their method used segmentation training data only or both segmentation and bounding box training data. This information will be used when analysing and presenting the competition results.

5.4 Evaluation

Each segmentation competition will be judged by average segmentation accuracy across the twenty classes and the background class. 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 9.5.2).

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. The 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. Only a single estimate of layout should be output for each person.

The success of the layout prediction depends both on: (i) a correct prediction of parts present/absent (e.g. is the hand visible or occluded); (ii) a correct prediction of bounding boxes for the visible 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
7	Layout	`trainval`	`test`
8	Layout	**any but** VOC `test`	`test`

In competition 7, 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 8, 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>2009_000183</image>
        <object>1</object>
        <confidence>-1189</confidence>
        <part>
            <class>head</class>
            <bndbox>
                <xmin>191</xmin>
                <ymin>25</ymin>
                <xmax>323</xmax>
                <ymax>209</ymax>
            </bndbox>
        </part>
        <part>
            <class>hand</class>
            <bndbox>
                <xmin>393</xmin>
                <ymin>206</ymin>
                <xmax>488</xmax>
                <ymax>300</ymax>
            </bndbox>
        </part>
        <part>
            <class>hand</class>
            <bndbox>
                <xmin>1</xmin>
                <ymin>148</ymin>
                <xmax>132</xmax>
                <ymax>329</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 9.1.4).

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 9.6.1). The example person layout implementation (section 8.2.4) includes code for generating a results file in the required format.

6.4 Evaluation

The person layout task will principally be judged by how well each part *individually* can be predicted. For each of the part types (head/hands/feet) a precision/recall curve will be computed, using the confidence supplied with the person layout to determine the ranking. A prediction of a part is considered true or false according to the overlap test, as used in the detection challenge, i.e. for a true prediction the bounding box of the part overlaps the ground truth by at least 50%. For each part type, the principal quantitative measure used will be the average precision (AP) (see section 3.4.1). Example code for computing the precision/recall curves and AP measure is provided in the development kit.

Invitation to propose evaluation schemes. We invite participants to propose additional evaluation schemes for the person layout task. In particular, we are interested in schemes which (i) evaluate accuracy of *complete* layout predictions; (ii) incorporate the notion of ranking of results by confidence. If you have a successful layout prediction method and insight please propose promising evaluation techniques in the form of (i) motivation and explanation; (ii) MATLAB implementation compatible with the VOC results format.

7 Action Classification Taster

7.1 Task

For each of the nine action classes predict if a specified person (indicated by their bounding box) in a test image is performing the corresponding action. The output from your system should be a real-valued confidence that the action is being performed so that a precision/recall curve can be drawn. Participants may choose to tackle all, or any subset of action classes, for example "walking only" or "walking and running".

7.2 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
9	Classification	`trainval`	`test`
10	Classification	**any but** VOC `test`	`test`

In competition 9, any annotation provided in the VOC `train` and `val` sets may be used for training. Participants may use images and annotation for any of the competitions for training e.g. horse bounding boxes/segmentation to learn 'ridinghorse'. Participants are *not* permitted to perform additional manual annotation of either training or test data.

In competition 10, 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.

7.3 Submission of Results

A separate text file of results should be generated for each competition (9 or 10) and each action class e.g. 'phoning'. Each line should contain a single image identifier, object index, and the confidence output by the classifier, separated by a space, for example:

```
comp9_action_test_phoning.txt:
    ...
    2010_006107  1  0.241236
    2010_006107  2  0.758739
    2010_006108  1  0.125374
    ...
```

The image identifier and object index specify the 'person' object to which the output corresponds; these are provided in the corresponding image sets. Greater confidence values signify greater confidence that the person is performing the action of interest. The example implementation (section 8.2.5) includes code for generating a results file in the required format.

7.4 Evaluation

The action classification task will be judged by the precision/recall curve. The principal quantitative measure used will be the average precision (AP) (see section 3.4.1). Example code for computing the precision/recall and AP measure is provided in the development kit.

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.

8 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].

8.1 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/VOC2010/            % your results on VOC2010
VOCdevkit/local/                      % example code temp dirs
VOCdevkit/VOC2010/ImageSets           % image sets
VOCdevkit/VOC2010/Annotations         % annotation files
VOCdevkit/VOC2010/JPEGImages          % images
VOCdevkit/VOC2010/SegmentationObject % segmentations by object
VOCdevkit/VOC2010/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`
- `example_action`

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
```

8.2 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.

8.2.1 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.

8.2.2 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.

8.2.3 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.

8.2.4 Example Layout Implementation

The file `example_layout.m` contains a complete implementation of the person layout task. 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.

8.2.5 Example Action Implementation

The file `example_action.m` contains a complete implementation of the action classification task. For each VOC action class a simple classifier is trained on the `train` set; the classifier is then applied to all specified 'person' objects in 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.

8.3 Non-MATLAB Users

For non-MATLAB users, the file formats used for the VOC2010 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.

9 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.

9.1 Image Sets

9.1.1 Classification/Detection Task Image Sets

The `VOC2010/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.

9.1.2 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 `VOC2010/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:

```
...
2009_000040 -1
2009_000042 -1
2009_000052  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.

9.1.3 Segmentation Image Sets

The `VOC2010/ImageSets/Segmentation/` directory contains text files specifying lists of images for the segmentation 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.

9.1.4 Person Layout Taster Image Sets

The `VOC2010/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:

```
...
2009_000595  1
2009_000595  2
2009_000606  1
...
```

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});
```

9.1.5 Action Classification Taster Image Sets

The `VOC2010/ImageSets/Action/` directory contains text files specifying lists of images and 'person' objects for the action classification 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.action.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 action classes. For each competition, participants are expected to provide output for all 'person' objects in each image of the `test` set.

9.1.6 Action Class Image Sets

To simplify matters for participants tackling the action classification task, action class-specific image sets with per-object ground truth are also provided. The file `VOC2010/ImageSets/Action/<class>_<imgset>.txt` contains image identifiers, object indices and ground truth for a particular action class and image set, for example the file `phoning_train.txt` applies to the 'phoning' action class and `train` image set.

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

```
...
2010_006215  1  1
2010_006217  1 -1
2010_006217  2 -1
...
```

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

```
imgset='train'; cls='phoning';
[imgids,objinds,gt]=textread(sprintf(VOCopts.action.clsimgsetpath,
                                     cls,imgset),'%s %d %d');
```

The annotation for the object (bounding box and actions) can then be obtained using the image identifier and object index:

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

There are two ground truth labels:

`-1`: Negative: The person is not performing the action of interest. A classifier should give a 'negative' output.

`1`: Positive: The person is performing the action of interest. A classifier should give a 'positive' output.

9.2 Development Kit Functions

9.2.1 `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 `actions` lists the action classes for the action classification task in a cell array:

```
VOCopts.actions={'phoning','playinginstrument','reading',...
                 'ridingbike','ridinghorse','running',...
                 'takingphoto','usingcomputer','walking'};
```

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.

9.2.2 `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,'2009_000067'))

rec =

       folder: 'VOC2009'
     filename: '2009_000067.jpg'
       source: [1x1 struct]
         size: [1x1 struct]
    segmented: 0
      imgname: 'VOC2009/JPEGImages/2009_000067.jpg'
      imgsize: [500 334 3]
     database: 'The VOC2009 Database'
      objects: [1x6 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)`. The `database` field specifies the data source (VOC2009 or VOC2010). 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: 334
     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(2)

ans =

        class: 'person'
         view: 'Right'
    truncated: 0
     occluded: 0
    difficult: 0
        label: 'PASpersonRight'
     orglabel: 'PASpersonRight'
         bbox: [225 140 270 308]
       bndbox: [1x1 struct]
      polygon: []
         mask: []
     hasparts: 1
         part: [1x4 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(2).bndbox

ans =

    xmin: 225
    ymin: 140
    xmax: 270
    ymax: 308
```

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` fields specify polygon/per-object segmentations, and are not provided for the VOC2010 data.

The `hasparts` field specifies if the object has sub-object "parts" annotated. For the VOC2010 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(2).part(1)

ans =

        class: 'head'
         view: ''
    truncated: 0
     occluded: 0
    difficult: 0
        label: 'PAShead'
     orglabel: 'PAShead'
         bbox: [234 138 257 164]
       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 VOC2010 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.

For 'person' objects in the action classification taster image sets, objects are additionally annotated with the set of actions being performed. The `hasactions` field specifies if the object has actions annotated. Action flags are stored in the struct `actions`, for example:

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

ans =

              phoning: 1
```

```
playinginstrument: 0
          reading: 0
       ridingbike: 0
      ridinghorse: 0
          running: 0
      takingphoto: 0
    usingcomputer: 0
          walking: 0
```

There is one flag for each of the nine action classes, with the flag set to true (1) if the person is performing the corresponding action. Note that actions are not mutually-exclusive.

9.2.3 `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');
```

9.3 Classification Functions

9.3.1 `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`.

9.4 Detection Functions

9.4.1 `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`.

9.4.2 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.

9.5 Segmentation Functions

9.5.1 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.

9.5.2 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.

9.5.3 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.

9.6 Layout Functions

9.6.1 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.

9.6.2 VOCevallayout_pr(VOCopts,id,draw)

The VOCevallayout_pr function performs evaluation of the person layout task, computing a precision/recall curve and the average precision (AP) measure for each part type (head/hands/feet). The arguments id and cls specify the results file to be loaded, for example:

```
>> [rec,prec,ap]=VOCevallayout_pr(VOCopts,'comp7',true);
```

See example_layout for further examples. If the argument draw is true, the precision/recall curves are drawn in a figure window. The function returns vectors of recall and precision rates in reci and prec{i}, and the average precision measure in ap{i}, where the index i indexes the part type in VOCopts.parts.

9.6.3 VOCevallayout_f(VOCopts,id,draw)

The VOCevallayout_f function performs secondary evaluation of the person layout task, computing a curve of f-measure versus recall. The arguments id and cls specify the results file to be loaded, for example:

```
>> [f,rec]=VOCevallayout_f(VOCopts,'comp7',true);
```

**** This information is preliminary and subject to change ****

9.7 Action Functions

9.7.1 VOCevalaction(VOCopts,id,cls,draw)

The VOCevalaction function performs evaluation of the action 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,'comp9','phoning',true);
```

See example_action 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 VOC2010 database: Yusuf Aytar, Jan Hendrik Becker, Patrick Buehler, Miha Drenik, Chris Engels, Ali Eslami, Adrien Gaidon, Sam Johnson, Jyri Kivinen, Lubor Ladicky, Markus Mathias, Alastair Moore, Glenn Sheasby, Paul Sturgess, David Tingdahl, Josiah Wang.

We also thank Alexander Sorokin for production and support of the annotation systems for Mechanical Turk, Ivan Laptev for development of the action classification task, and Marcin Eichner, Marcin Marszalek and Andrea Vedaldi for testing on the development kit.

The preparation and running of this challenge is supported by the EU-funded PASCAL2 Network of Excellence on Pattern Analysis, Statistical Modelling and Computational Learning. We are grateful to Alyosha Efros for providing additional funding for annotation on Mechanical Turk.

References

[1] The PASCAL Visual Object Classes Challenge (VOC2010). http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2010/index.html.