LSVM-MDPM Release 4 Notes

Ross Girshick
University of Chicago
rbg@cs.uchicago.edu

Pedro Felzenszwalb
University of Chicago
pff@cs.uchicago.edu

David McAllester
TTI at Chicago
mcallester@ttic.edu

April 21, 2010

1 Introduction

This note describes some recent advances that improve the performance of the object detection system described in [2]. Some of the improvements included here comprised the UoC-TTI LSVM-MDPM entry in the PASCAL VOC 2009 `comp3` challenge [1] and others were developed subsequently. Complete source code for the latest version of the object detection system can be found at `http://people.cs.uchicago.edu/~pff/latent/`.

2 Models

Figure 1 shows some of the models trained by the current version of the system.

In [2] each object class was represented by a two component mixture of deformable part models, where each component is bilaterally symmetric. We now use a richer class of models, where each object class is represented by a three component mixture of asymmetric models. Bilateral asymmetry allows each component to specialize at the task of detecting left or right object poses. During detection each component is matched to the image in both left and right orientations. This means that in effect we have a mixture model with six components, with the extra constraint that components are grouped into left-right symmetric pairs. The left-right pose distinction is automatically learned by our system in an unsupervised fashion without the use of additional pose labels (we ignore the incomplete pose labels given by the PASCAL annotations).

2.1 Left-Right Pose Clustering

The input data is a set of images containing instances of an object class. The location and extent of each instance is specified by a bounding box. As in [2] we use the aspect ratio of the bounding boxes to separate the instances into different clusters. But now we further break down these clusters to separate left and right facing examples.

We crop the image region under each bounding box and resize it to a fixed width and height. Each cropped and resized region, along with its vertically flipped counterpart, is mapped into a feature space (we use a variant of HOG features described in [2]) where clustering takes place.

The clustering algorithm is a variant of online k-means with the following constraint: no example and its flipped counterpart may be placed into the same cluster. The algorithm begins by selecting (uniformly at random) an example and its flipped counterpart. These feature vectors seed the two

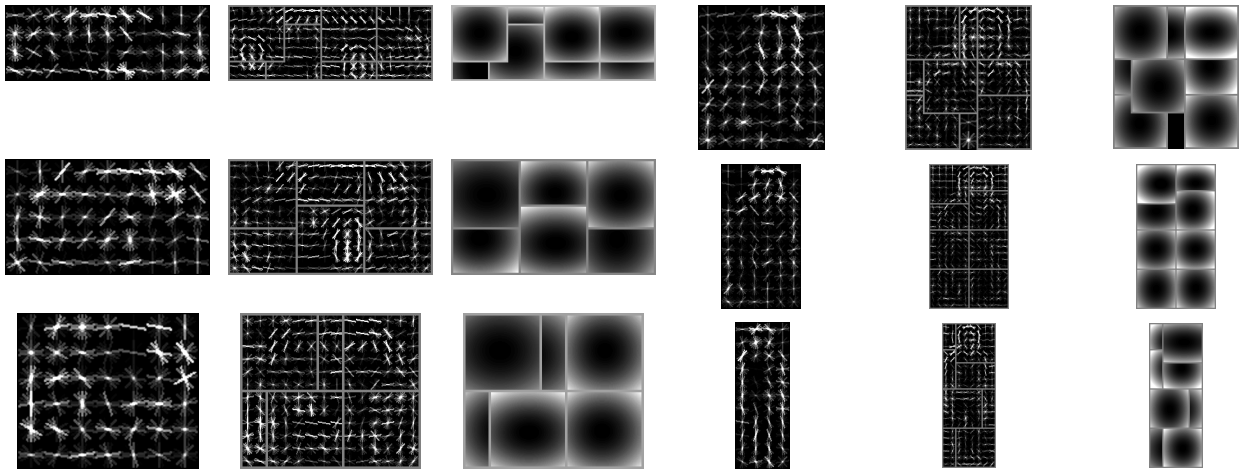
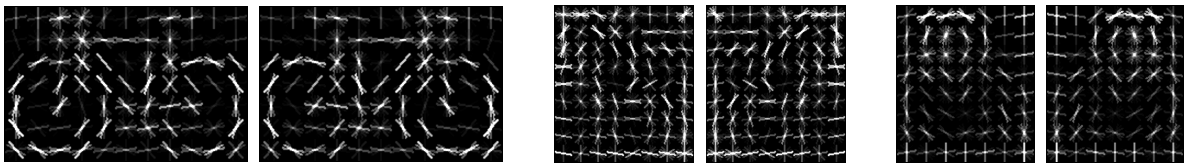


Figure 1: Fully trained car and person models using the PASCAL VOC 2007 train+val data. Here we display 3 components for each class. In practice each component also has a left-right flipped counterpart for a total of 6 components per class.



(a) (b) (c)

Figure 2: Examples of root filters trained on left-right pose clustered data from the classes (a) bicycle, (b) horse, and (c) person.

clusters. In each iteration a new example is drawn from the pool of remaining examples. The chosen example is assigned to the cluster with minimum Euclidean distance between the cluster center and the example. The example's flipped counterpart is then assigned to the other cluster.

After all examples have been assigned we use a local search method to improve the clustering. We repeatedly pick an example and its counterpart, and check if swapping their cluster labels can reduce the total sum of squared distances (SSD) from examples to their assigned cluster's center.

The clustering process is repeated several times using different initial seed examples to avoid bad local minima. The clustering with the best SSD objective function value is selected. Empirically we have found that this process is very effective at clustering most object classes into two clusters corresponding to left and right facing examples. See Figure 2 for some examples. Naturally there are some exceptions (e.g., bottle) where the left-right distinction does not make sense.

2.2 Part Initialization

Given a root filter, we initialize k $d \times d$ part filters at twice the spatial resolution of the root by selecting part locations and cropping out sub-arrays from a higher resolution version of the root filter. The default configuration of the system uses $k = 8$ and $d = 6$. That is, we have 8 6×6 parts in each deformable model.

Part locations are selected using a two phase procedure. The first phase uses a greedy method and the second phase refines the initialization.

To initialize part locations, the root filter is interpolated to twice its original resolution and an "energy" map of the interpolated filter is computed. The energy map records the squared norm of the positive filter weights in each filter cell. The k parts are sequentially placed in the position that covers the largest amount of remaining energy. Cells that have already been covered have their energy set to zero.

After greedily selecting initial part locations, we use local search to move the parts, one at a time in random order, to maximize the amount of energy that is covered. When a better covering cannot be found, this phase is restarted from the initial part configuration to see if more energy can be covered by repositioning the parts in a different order. After several restarts the part configuration that covers the most energy is selected.

2.3 Image Boundary Occlusion

The PASCAL dataset contains many examples of objects that are truncated by the image boundary. To detect partially visible objects we pad each image's feature map with a boundary region.

In [2] feature vectors in the boundary region are set to the zero vector. This leads to a fixed score of zero for any filter placed entirely outside the image, which may be inappropriately calibrated with respect to filter responses inside the image. To compensate for this effect, we now augment feature vectors with an additional feature that takes the value 0 if the feature is inside the image and 1 if the feature is in the boundary region. This boundary occlusion feature enables the learning of a bias parameter for each filter cell that is added to the filter response if that filter cell is placed in the boundary region.

This 0/1 occlusion feature is the same as the one proposed in [3], however there are two differences in our implementation. The implementation described in [3] uses a single occlusion feature per filter (not per filter cell) that counts the number of filter cells that are placed in the boundary region. The second difference is in our training data requirements. The training procedure in [3] requires manually extending the PASCAL bounding boxes to indicate how far each bounding box, that is truncated by the image boundary, ought to extend into the boundary region. Our approach does not require any change to the PASCAL annotations. Instead, during the latent variable completion stage of our training procedure, we measure overlap of the putative detection window with the ground-truth bounding box after first clipping the detection window to the image boundary.

3 Regularization

In [2] we trained model parameters β by optimizing the latent SVM objective function

$$\frac{1}{2}||\beta||^2 + C\sum_{i=1}^{N}\max(0, 1 - y_i f_\beta(x_i)), \tag{1}$$

where $f_\beta(x) = \max_{z \in Z(x)} \beta \cdot \Phi(x, z)$.

For a mixture model with k components, the vector of model parameters β can be written as the concatenation of k vectors, one per component, $\beta = (\beta_1, \ldots, \beta_k)$. Empirically we have found that a regularizer that only penalizes the component vector with the largest norm leads to better results (as measured by average precision on test data).

The max-regularized objective function is given by

$$\frac{1}{2}\max_{i=1,\ldots,k}||\beta_i||^2 + C\sum_{i=1}^{N}\max(0, 1 - y_i f_\beta(x_i)). \quad (2)$$

For non-separable data this objective function will typically lead to component vectors with equal norm. This happens because in practice there is no regularization on a component vector β_i whose norm is below the maximum one. This seems to make the margin requirements for different examples more compatible.

4 Results

The tables below summarize the current results in the PASCAL 2006, 2007, and 2009 datasets following the comp3 protocol.

	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	*mean*
without context	39.5	48.2	11.4	12.3	28.6	42.3	40.4	25.0	17.4	20.5	15.3	14.5	42.1	44.4	41.9	12.7	24.3	16.5	43.3	32.2	28.6
with context	43.6	50.8	15.1	14.1	30.2	45.6	41.8	27.3	18.9	22.1	15.8	18.2	45.7	47.3	43.8	14.3	26.4	18.2	46.8	33.7	31.0

Table 1: PASCAL VOC 2009 comp3

	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	*mean*
without context	28.9	59.5	10.0	15.2	25.5	49.6	57.9	19.3	22.4	25.2	23.3	11.1	56.8	48.7	41.9	12.2	17.8	33.6	45.1	41.6	32.3
with context	31.2	61.5	11.9	17.4	27.0	49.1	59.6	23.1	23.0	26.3	24.9	12.9	60.1	51.0	43.2	13.4	18.8	36.2	49.1	43.0	34.1

Table 2: PASCAL VOC 2007 comp3

	bike	bus	car	cat	cow	dog	horse	mbike	person	sheep	*mean*
without context	67.1	65.8	70.7	26.8	47.7	15.8	48.3	66.0	41.0	45.6	49.5
with context	69.2	67.6	71.5	29.0	51.4	19.4	54.0	70.0	44.3	47.4	52.4

Table 3: PASCAL VOC 2006 comp3

We also trained and tested a model on the INRIA Person dataset. We scored the model using the PASCAL evaluation methodology in the complete test dataset, including images without people.

INRIA Person average precision: 88.2

References

[1] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2009 (VOC2009) Results.

[2] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2009.

[3] A. Vedaldi and A. Zisserman. Structured output regression for detection with partial occulsion. In *Advances in Neural Information Processing Systems*, 2009.