Using the Deformable Part Model with Autoencoded Feature Descriptors for Object Detection

Hyunghoon Cho and David Wu

December 10, 2010

1 Introduction

Given its performance in recent years' PASCAL Visual Object Classes VOC Challenge [1], the Deformable Part Model (DPM) is widely regarded to be one of the state-of-the-art object detection and localization algorithms. The DPM as described by Felzenszwalb, et al. [2] uses Histogram of Oriented Gradients (HOG) descriptors [3] as the underlying feature representation for an object. In this paper, we consider using features learned by a single-layered, sparse autoencoder as a substitute for HOG descriptors in the DPM. The rationale for this is that these learned feature descriptors may capture additional details present in the image that are not present in a human-engineered set of features such as HOG. Using this more descriptive set of features may in turn yield a better object detector. Additionally, it is noteworthy that while deep learning algorithms such as the sparse autoencoder alone generally do not perform as well compared to other vision algorithms, it may be possible to integrate learned features into an existing image classification system such as the DPM. To further evaluate the performance of such an integration, we consider the effect of factors such as block normalization and colored features on the performance of the autoencoder-backed DPM. Finally, we also examine the performance of the autoencoder-backed DPM across several different object classes from the PASCAL VOC Challenge.

2 Methodology

Our training images are sampled from the PASCAL 2010 training image sets and our test images are sampled from the PASCAL 2007 test image sets [1]. For this paper, we consider two different training set sizes: small (20 positives : 200 negatives) and medium (50:500).

2.1 Autoencoder Training

We train a sparse, single layered, autoencoder with a sigmoid activation function[1] to obtain a set of features. We iterate over each positive training image in the training set and crop the image to the ground truth bounding box for the image. We convert the image to grayscale and whiten it. From this set of whitened, positive examples, we sample 400,000 8x8 image patches to train an autoencoder with 64, 100, or 128 hidden nodes. We center and standardize each patch $x \in \mathbb{R}^{64}$ according to $\frac{x-\mu_x}{\sigma_x}$ prior to training the autoencoder. We also consider colored image patches in which each 8x8 image patch now corresponds to a vector $x \in \mathbb{R}^{192}$ (64 components per color channel) and train an autoencoder with 256 hidden nodes. In the DPM, Felzenszwalb, et al. [2] consider the mirror image for each training example to effectively double the size of the training set. With HOG descriptors, it is sufficient to mirror the orientation of each gradient to determine the HOG representation of the mirror image. To allow for similar functionality with the autoencoder, we compute the mirror image of each feature learned by the original autoencoder to obtain a mirrored autoencoder with the same number of hidden units. We concatenate the original and mirrored autoencoders to obtain a set of 128, 200, or 256 features. Finally, Felzenszwalb, et al. [2] also incorporate one additional occlusion feature that denotes whether the full object is visible in the image or not. Likewise, we introduce one additional occlusion feature.

2.2 Model Training and Visualization

We replace the HOG feature descriptors in the DPM. At the low level, the DPM divides the image into a rigid grid of 8x8 pixel non-overlapping cells and computes a histogram for each cell. In addition, the model captures finer details (part filters) in the object by using smaller 4x4 pixel patches. We adapt this approach to work with the autoencoder features.

[1] Autoencoder training toolkit provided by Adam Coates

2.2.1 Histograms of Autoencoder Features

To determine the features that comprise a particular image patch, we determine the distribution of features in a particular image patch x by computing $g(Wx+b)$ where $g(x) = \frac{1}{1+e^{-x}}$ is the sigmoid activation function for the hidden layer, and W and b are the corresponding weight and intercept terms for the inputs to the autoencoder. The resultant activation vector expresses the degree to which each hidden node is activated by the given image patch.

In addition to using the sigmoid activation function, we consider another method of computing a distribution of features for each image patch. In particular, we substitute a nonlinear activation function $H(x)$ that is 0 below some threshold c_1, and linear with slope a above the threshold. We cap the activation at a value c_2.

$$H(x) = \begin{cases} 0 & x < c_1 \\ \min\{a(x-c_1), c_2\} & x \geq c_1 \end{cases}$$

We determine our histogram of features by computing $H(Wx+b)$. In particular, we choose c_1,c_2, and a so that the range of values in the histogram of features is similar to that computed by HOG.

To train the DPM, we compute histograms of features for 4x4 image patches which are used by the part filters. We cannot train a completely separate autoencoder on 4x4 patches since the features learned from 4x4 image patches do not necessarily coincide with features learned from 8x8 image patches. We consider two different approaches. In the first approach, we construct a 4x4 autoencoder from the 8x8 autoencoder where for each feature in the 8x8 autoencoder, we remove every other row and column to produce a "scaled down" 4x4 feature. We use this autoencoder to construct the histogram of features for 4x4 image patches. In the second approach, we use only the 8x8 autoencoder. To construct the histogram for 4x4 image patches, we scale the 4x4 image patch up to obtain an 8x8 image patch, which we then run through the autoencoder to obtain the histogram.

We consider a simple method of incorporating color into the histograms using the existing autoencoder. When training the model, instead of working with grayscale images, we consider each color channel separately. Now, for each feature, we simply take the maximum activation for each feature over the three color channels and use that as the activation.

For the final test, we consider the result of combining HOG descriptors with autoencoder features. In this scheme, we first compute a histogram of autoencoder features using an autoencoder with 100 hidden nodes. We then compute the HOG descriptors and adjoin these two vectors to obtain a 232 element vector (100 autoencoder + 100 mirror + 32 HOG) that serves as the feature descriptor for the image patch.

2.2.2 Normalization

In Dalal and Triggs' HOG implementation [3], they consider block normalization where the gradients for a patch are averaged over neighboring blocks. Felzenszwalb, et al. has experimentally demonstrated noticeable improvements using a block normalization procedure [2]. We consider a similar normalization method as [3]. First we calculate the energy $E_{i,j}$ (the square of the norm) of the feature histogram for each image patch (i,j). Then we take the histogram at each patch (x,y) and multiply the elements by the normalization constant $C_{x,y}$ where

$$C_{x,y} = \frac{1}{4} \sum_{i,j \in \{1,-1\}} \left(\frac{1}{\sqrt{E_{x,y} + E_{x+i,y} + E_{x,y+j} + E_{x+i,y+j}} + \epsilon} \right)$$

where ϵ is a small constant so we do not divide by zero. This normalization has the effect of mitigating situations where there is a high local variation in the energy of a block.

2.2.3 Model Visualization

To visualize the model, we take the image patch that maximally activates each hidden node in the autoencoder. These image patches are given by the rows in the weight matrix for the input layer. For each patch in the model, we alpha-blend the feature image patches using $\alpha_i = \frac{a_i^2}{E}$ as the opacity of the i^{th} feature, where a_i is the activation value and E is the energy of the feature histogram.

2.2.4 Tuning the Latent SVM

Finally, we consider the effect of changing the regularization parameter used to train the latent SVM (LSVM) in the DPM. The value used by Felzenszwalb, et al. [2] works well in the case of HOG descriptors, but may not be well tuned for autoencoder features. Hence, we consider both higher and lower values of the regularization parameter C in the LSVM objective function. We evaluate by testing on both a subset of the training set as well as the test set.

2.3 Model Evaluation

To evaluate an object model, we test on a sample of test images from the PASCAL VOC 2007 [1] test set that are similar to the training images. We use the same scoring criterion as PASCAL: a detection is correct if and only if $\frac{area[\text{Predicted} \cap \text{True}]}{area[\text{Predicted} \cup \text{True}]} > 0.5$ where Predicted denotes the bounding box predicted by the model and True denotes the true bounding box for the image. If one object is detected multiple times, only one is taken to be a true positive and all the other ones are taken to be false positives. Using this information, we can compute a precision-recall curve for the detector, as well as compute a value describing the average precision for the detector. This is the metric used by PASCAL.

3 Results and Discussion

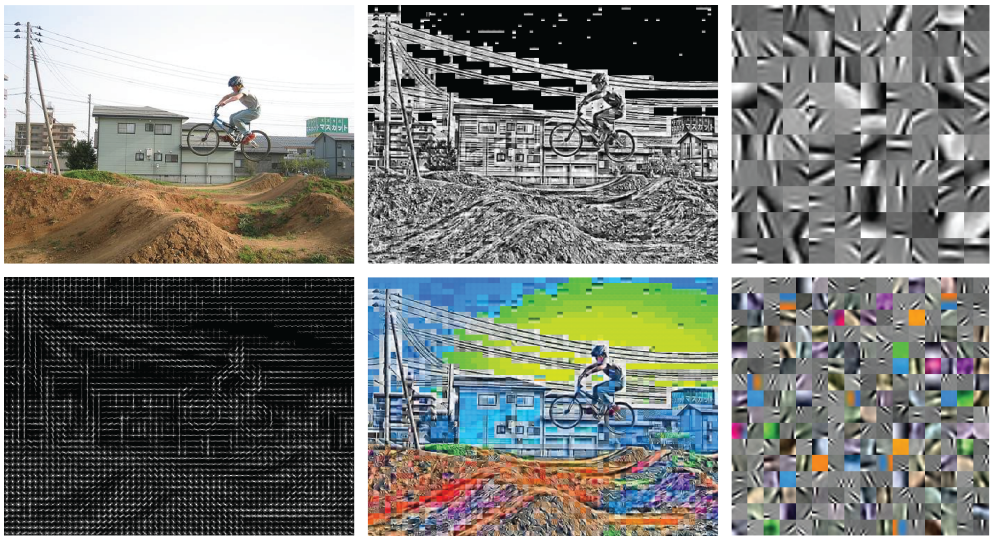

Figure 1: Top left: sample training image of bicycle. Bottom left, middle: The same picture represented using HOG descriptors, grayscale autoencoder features, and colored autoencoder features. Right: Grayscale (top) and colored (bottom) autoencoder features learned from 8x8 grayscale and colored image patches of bicycles.

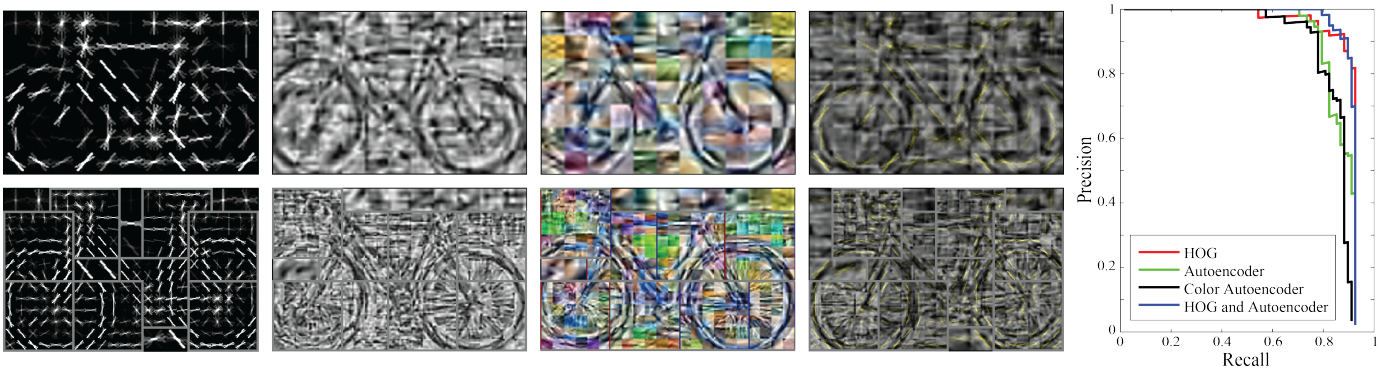

Figure 2: From left to right: model of bicycle using HOG descriptors, autoencoder with 100 hidden nodes, colored autoencoder with 256 hidden nodes, and both HOG and autoencoder with 100 hidden nodes (HOG features in yellow). Top row shows root filter and bottom row shows finer part filters. To the right, we show the precision-recall curve for the four models.

Table 1 shows the effect of various parameters on average precision of the model. With the exception of the specified parameter we are evaluating, each model described in the table is for a bicycle model trained without block normalization and using only the 8x8 autoencoder with 100 hidden nodes on a medium training set. Noting that the autoencoder with the nonlinear $H(x)$ activation function slightly outperformed models trained with the sigmoid $g(x)$ activation function, for the majority of tests, we only train and evaluate models using that particular activation function. First, we see that while there is no sizable benefit from using a larger training set in the case of HOG, there is an improvement in the autoencoder models by using a larger training set. Therefore, one possible way of achieving better performance with the autoencoder may be

	HOG Descriptors	Activation Function Sigmoid $g(x)$	Activation Function Nonlinear $H(x)$
Small Training Set	0.9054	-	0.8154 (64 Hidden Nodes) 0.8228 (100 Hidden Nodes)
Medium Training Set	0.8915	0.8563	0.8596 (64 Hidden Nodes) 0.8607 (100 Hidden Nodes)
Block Normalization	-	0.8406	0.7850
Without Patch Standardization	-	-	0.7499
Scaled Down 4x4 Autoencoder	-	-	0.7139
Features over Max Color Channel	-	-	0.8466
Higher Threshold for $H(x)$ ($c_1 = -4$)	-	-	0.8489
Highest Threshold for $H(x)$ ($c_1 = -3$)	-	-	0.8235

Table 1: Average precision for different training set sizes and methods of computing histogram of autoencoder features

to further increase the size of the training set. Interestingly, increasing the number of features did not appear to have as substantial an impact as increasing the training set size.

Next, we consider four variations (described above) on the way we compute our feature histogram: adding block normalization, removing patch standardization when training the DPM, using a scaled-down 4x4 autoencoder for computing 4x4 part features, and evaluating on each color channel separately. Contrary to expectations, the block normalization scheme does not increase the effectiveness of the model, and in the case where we use the $H(x)$ activation function, results in significantly worse performance. Similarly, removing patch standardization also decreases the performance of the model. However, removing this additional preprocessing step significantly increases the speed of the training by almost a factor of two. Hence, in cases where run-time performance may be more important than the accuracy of the model, this may be a viable trade-off. Finally, we consider the approach of using a separate 4x4 autoencoder (effectively a "scaled" down version of the 8x8 autoencoder). This leads to a significant drop in detector precision compared to the model trained using only the 8x8 autoencoder (where 4x4 image patches are scaled up to 8x8 and represented using features from the 8x8 autoencoder). This may have been due to the fact that simply removing every other pixel from each feature in the 8x8 autoencoder is not a reasonable way of constructing a 4x4 autoencoder and may have ended up distorting the features. Finally, we consider a way of incorporating color similar to what was done by Felzenszwalb, et al. [2]. We use the maximum hidden node activation value over the three color channels as the activation for a particular image patch. This appears to have a fairly insignificant, even detrimental, effect on the performance of the model. A potentially better method of incorporating color may be to use a separate autoencoder that incorporates colored image patches. We discuss this result later.

The final set of tests we consider is to change the threshold used in computing the $H(x)$ activations. Higher thresholds would imply fewer hidden node activations for each image patch, possibly resulting in less noise in the histogram of features, but at the same time, reduce the number of features that are expressed in the histogram. The data indicate that performance generally deteriorated for higher values of the threshold (admitting fewer node activations).

	Bicycle	Car	Bottle	Horse	Sofa	Airplane
HOG Descriptors	0.8915	0.5688	0.2467	0.6546	0.5250	0.6263
Autoencoder Features	0.8607	0.4928	0.0870	0.4674	0.2421	0.5478

Table 2: Average precision values for different object classes

Table 2 shows the average precision values for different object models from the VOC challenge [1]. For each object class, we train the autoencoder-backed DPM with a medium training set and using $H(x)$ as the activation function. We test both the autoencoder-backed model and a HOG-backed model on a sample test set taken from the VOC challenge. Though the autoencoder-backed DPM did not achieve superior performance, these results indicate that our system has the potential to work across many different object categories.

Table 3 above shows the average precision values for models trained with a variable number of features. Note that for an autoencoder with n nodes, the number of features is $2n + 1$ (original autoencoder features + mirrored features + occlusion feature). In the case where we integrate the autoencoder features with the HOG descriptors, the occlusion feature is already incorporated in the HOG features. Each of the above models are trained on the bicycle medium training set using activation function $H(x)$. It is interesting to note that additional features (hidden nodes) did not necessarily correlate to better performance; there was a slight gain going from 64 hidden nodes to 100 hidden nodes, but not much thereafter. Interestingly, incorporating color also did not yield an observable benefit. It is possible that we need a larger training set in order to further improve performance. Finally, it is noteworthy that incorporating both HOG descriptors as well as autoencoder features lead to an observable improvement in the detector's performance. In particular, its performance is

	Average Precision
32 Features (32 HOG)	0.8915
129 Features (64 Hidden Nodes)	0.8596
201 Features (100 Hidden Nodes)	0.8607
257 Features (128 Hidden Nodes)	0.8388
232 Features (100 Hidden Nodes, 32 HOG)	0.8993
513 Features (256 Hidden Nodes with Color)	0.8402

Table 3: Average precision values using different number of features (bicycle, medium training set)

slightly higher than the HOG model trained on the same training set. This indicates that there might be features that are captured by HOG, but are not completely captured by the autoencoder.

Regularization Parameter	Average Precision (Training)	Average Precision (Test)
0.002 (HOG)	0.9145	0.8915
0.001	0.8908	0.8440
0.002	0.9076	0.8607
0.004	0.9083	0.8270
0.006	0.9107	0.8374
0.02	0.9111	0.8390

Table 4: Average precision values from different regularization parameters (bicycle, medium training set)

In Table 4, we consider using different values of the regularization parameter C used to train the latent SVM desribed in [2] to try and make the average precision on the training set converge to the average precision achieved on the test set. In general, while increasing the value of C increases performance on the training set, performance on the test set appear relatively unaffected. The value Felzenszwalb, et. al [2] used to train the HOG-backed DPM (0.002) appears to work well in practice for the autoencoder models. Overall, changing the regularization parameter did not appear to be very significant as far as performance on the test set is concerned.

4 Conclusion and Future Work

Overall, the results indicate that the autencoder-backed DPM is a viable object detector. While using only autoencoder features do not outperform the model trained using HOG features, the average precision values tend to be comparable. Furthermore, integrating autoencoder features with HOG descriptors produces a detector whose results are actually slightly better than those obtained by HOG alone. Results also indicate that variations such as using more hidden nodes, adding block normalization, and varying the threshold for $H(x)$ do not lead to noticeable gains in detector performance. Rather, the single more important factor appears to be training set size. Current results show that we may be able to achieve substantial improvements if we further increase the size of the training set and that is one avenue of future consideration. Another interesting possibility for further research would be to use multiple, stacked autoencoders to learn more complex low-level feature representations for a given object. Finally, we are currently looking into optimizing the run-time performance of the training algorithm in order to support substantially larger training sets, which should in turn, lead to better overall performance.

5 Acknowledgments

We would like to thank Professor Andrew Ng and Adam Coates for the help and advice they provided for this project.

6 References

1 M. Everingham and L. van Gool. The PASCAL Visual Objects Classes Challenge 2010. http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2010/index.html.

2 P. Felzenszwalb, D. McAllester, D. Ramanan. A Discriminatively Trained, Multiscale, Deformable Part Model. Proceedings of the IEEE CVPR 2008.

3 N. Dalal and B.Triggs. Histograms of oriented gradients for human detection. In *CVPR*, pages I: 886-893, 2005.