Active Deformable Part Models Inference*

Menglong Zhu, Nikolay Atanasov, George J. Pappas, and Kostas Daniilidis

GRASP Laboratory, University of Pennsylvania
3330 Walnut Street, Philadelphia, PA 19104, USA**

Abstract. This paper presents an active approach for part-based object detection, which optimizes the order of part filter evaluations and the time at which to stop and make a prediction. Statistics, describing the part responses, are learned from training data and are used to formalize the part scheduling problem as an *offline* optimization. Dynamic programming is applied to obtain a policy, which balances the number of part evaluations with the classification accuracy. During inference, the policy is used as a look-up table to choose the *part order* and the *stopping time* based on the observed filter responses. The method is faster than cascade detection with deformable part models (which does not optimize the part order) with negligible loss in accuracy when evaluated on the PASCAL VOC 2007 and 2010 datasets.

1 Introduction

Part-based models such as deformable part models (DPM) [7] have become the state of the art in today's object detection methods. They offer powerful representations which can be learned from annotated datasets and capture both the appearance and the configuration of the parts. DPM-based detectors achieve unrivaled accuracy on standard datasets but their computational demand is high since it is proportional to the number of parts in the model and the number of locations at which to evaluate the part filters. Approaches for speeding-up the DPM inference such as cascades, branch-and-bound, and multi-resolution schemes, use the responses obtained from initial part-location evaluations to reduce the future computation. This paper introduces two novel ideas, which are missing in the state-of-the-art methods for speeding up DPM inference.

First, at each location in the image pyramid, a part-based detector has to make a decision: whether to evaluate more parts and in what order or to stop and predict a label. This decision can be treated as a *planning problem*, whose

* Electronic supplementary material -Supplementary material is available in the online version of this chapter at `http://dx.doi.org/10.1007/978-3-319-10584-0_19`. Videos can also be accessed at `http://www.springerimages.com/videos/978-3-319-10583-3`

** Financial support through the following grants: NSF-IIP-0742304, NSF-OIA-1028009, ARL MAST CTA W911NF-08-2-0004, ARL Robotics CTA W911NF-10-2-0016, NSF-DGE-0966142, NSF-IIS-1317788 and TerraSwarm, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA is gratefully acknowledged.

D. Fleet et al. (Eds.): ECCV 2014, Part VII, LNCS 8695, pp. 281–296, 2014.

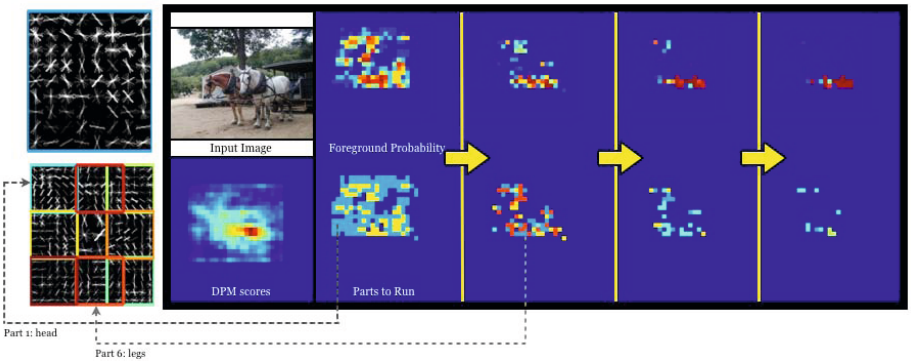


Fig. 1. Active DPM Inference: A deformable part model trained on the PASCAL VOC 2007 horse class is shown with colored root and parts in the first column. The second column contains an input image and the original DPM scores as a baseline. The rest of the columns illustrate the ADPM inference which proceeds in rounds. The foreground probability of a horse being present is maintained at each image location (top row) and is updated sequentially based on the responses of the part filters (high values are red; low values are blue). A policy (learned off-line) is used to select the best sequence of parts to apply at different locations. The bottom row shows the part filters applied at consecutive rounds with colors corresponding to the parts on the left. The policy decides to stop the inference at each location based on the confidence of foreground. As a result, the complete sequence of part filters is evaluated at very few locations, leading to a significant speed-up versus the traditional DPM inference. Our experiments show that the accuracy remains unaffected.

state space consists of the set of previously used parts and the confidence of whether an object is present or not. While existing approaches rely on a predetermined sequence of parts, our approach optimizes the order in which to apply the part filters so that a minimal number of part evaluations provides maximal classification accuracy at each location. Our second idea is to use a decision loss in the optimization, which quantifies the trade-off between false positive and false negative mistakes, instead of the threshold-based stopping criterion utilized by most other approaches. These ideas have enabled us to propose a novel object detector, Active Deformable Part Models (ADPM), named so because of the active part selection. The detection procedure consists of two phases: an off-line phase, which learns a part scheduling policy from training data and an online phase (inference), which uses the policy to optimize the detection task on test images. During inference, each image location starts with equal probabilities for object and background. The probabilities are updated sequentially based on the responses of the part filters suggested by the policy. At any time, depending on the probabilities, the policy might terminate and predict either a background label (which is what most cascaded methods take advantage of) or a positive label. Upon termination all unused part filters are evaluated in order to obtain the complete DPM score. Fig. 1 exemplifies the inference process.

We evaluated our approach on the PASCAL VOC 2007 and 2010 datasets [5] and achieved state of the art accuracy but with a 7 times reduction in the number

of part-location evaluations and an average speed-up of 3 times compared to the cascade DPM [6]. This paper makes the following **contributions** to the state of the art in part-based object detection:

1. We obtain an active part selection policy which optimizes the order of the filter evaluations and balances number of evaluations with the classification accuracy based on the scores obtained during inference.
2. The ADPM detector achieves a significant speed-up versus the cascade DPM without sacrificing accuracy.
3. The approach is independent of the representation. It can be generalized to any classification problem, which involves a linear additive score and uses several parts (stages).

2 Related Work

We refer to work on object detection that optimizes the inference stage rather than the representations since our approach is representation independent. We show that the approach can use the traditional DPM representation [7] as well as lower-dimansional projections of its filters. Our method is inspired by an acceleration of the DPM object detector, the cascade DPM [6]. While the sequence of parts evaluated in the cascade DPM is predefined and a set of thresholds is determined empirically, our approach selects the part order and the stopping time at each location based on an optimization criterion. We find the closest approaches to be [21,24,9,12]. Sznitman et al. [21] maintain a foreground probability at each stage of a multi-stage ensemble classifier and determine a stopping time based on the corresponding entropy. Wu et al. [24] learn a sequence of thresholds by minimizing an empirical loss function. The order of applying ensemble classifiers is optimized in Gao et al. [9] by myopically choosing the next classifier which minimizes the entropy. Karayev at el. [12] propose anytime recognition via Q-learning given a computational cost budget. In contrast, our approach optimizes the stage order and the stopping criterion jointly.

Kokkinos [13] used Branch-and-Bound (BB) to prioritize the search over image locations driven by an upper bound on the classification score. It is related to our approach in that object-less locations are easily detected and the search is guided in location space but with the difference that our policy proposes the next part to be tested in cases when no label can yet be given to a particular location. Earlier approaches [15,17,14] relied on BB to constrain the search space of object detectors based on a sliding window or a Hough transform but without deformable parts. Another related group of approaches focuses on learning a sequence of object template tests in position, scale, and orientation space that minimizes the total computation time through a coarse-to-fine evaluation [8,18].

The classic work of Viola and Jones [22] introduced a cascade of classifiers whose order was determined by importance weights, learned by AdaBoost. The approach was studied extensively in [2,3,10,16,25]. Recently, Dollar et al. [4] introduced cross-talk cascades which allow detector responses to trigger or suppress the evaluation of weak classifiers in the neighboring image locations. Weiss

et al. [23] used structured prediction cascades to optimize a function with two objectives: pose refinement and filter evaluation cost. Sapp et al. [20] learn a cascade of pictorial structures with increasing pose resolution by progressively filtering the pose-state space. Its emphasis is on pre-filtering structures rather than part locations through max-margin scoring so that human poses with weak individual part appearances can still be recovered. Rahtu et al. [19] used general "objectness" filters in a cascade to maximize the quality of the locations that advance to the next stage. Our approach is also related to and can be combined with active learning via Gaussian processes for classification [11].

Similarly to the closest approaches above [6,13,21,24], our method aims to balance the number of part filter evaluations with the classification accuracy in part-based object detection. The novelty and the main advantage of our approach is that in addition it optimizes the part filter ordering. Since our "cascades" still run only on parts, we do not expect the approach to show higher accuracy than structured prediction cascades [20] which consider more sophisticated representations that the pictorial structures in the DPM.

3 Technical Approach

The state-of-the-art performance in object detection is obtained by star-structured models such as DPM [7]. A star-structured model of an object with n parts is formally defined by a $(n+2)$-tuple $(F_0, P_1, \ldots, P_n, b)$, where F_0 is a root filter, b is a real-valued bias term, and P_k are the part models. Each part model $P_k = (F_k, v_k, d_k)$ consists of a filter F_k, a position v_k of the part relative to the root, and the coefficients d_k of a quadratic function specifying a deformation cost of placing the part away from v_k. The object detector is applied in a sliding-window fashion to each location x in an image pyramid, where $x = (r, c, l)$ specifies a position (r, c) in the l-th level (scale) of the pyramid. The space of all locations (position-scale tuples) in the image pyramid is denoted by $\mathcal{X}$. The response of the detector at a given root location $x = (r, c, l) \in \mathcal{X}$ is:

$$score(x) = F_0' \cdot \phi(H, x) + \sum_{k=1}^{n} \max_{x_k} \Big(F_k' \cdot \phi(H, x_k) - d_k \cdot \phi_d(\delta_k) \Big) + b,$$

where $\phi(H, x)$ is the histogram of oriented gradients (HOG) feature vector at location x and $\delta_k := (r_k, c_k) - (2(r, c) + v_k)$ is the displacement of the k-th part from its anchor position v_k relative to the root location x. Each term in the sum above implicitly depends on the root location x since the part locations x_k are chosen relative to it. The score can be written as:

$$score(x) = \sum_{k=0}^{n} m_k(x) + b, \tag{1}$$

where $m_0(x) := F_0' \cdot \phi(H, x)$ and for $k > 0$, $m_k(x) := \max_{x_k} \big(F_k' \cdot \phi(H, x_k) - d_k \cdot \phi_d(\delta_k) \big)$. From this perspective, there is no difference between the root and the parts and we can think of the model as one consiting of $n + 1$ parts.

3.1 Score Likelihoods for the Parts

The object detection task requires labeling every $x \in \mathcal{X}$ with a label $y(x) \in \{\ominus, \oplus\}$. The traditional approach is to compute the complete score in (1) at every position-scale tuple $x \in \mathcal{X}$. In this paper, we argue that it is not necessary to obtain all $n+1$ part responses in order to label a location x correctly. Treating the part scores as noisy observations of the true label $y(x)$, we choose an effective order in which to receive observations and an optimal time to stop. The stopping criterion is based on a trade-off between the cost of obtaining more observations and the cost of labeling the location x incorrectly.

Formally, the part scores $m_0, \ldots, m_n$ at a fixed location x are random variables, which depend on the input image, i.e. the true label $y(x)$. To emphasize this we denote them with upper-case letters M_k and their realizations with lower-case letters m_k. In order to predict an effective part order and stopping time, we need statistics which describe the part responses. Let $h^{\oplus}(m_0, m_1, \ldots, m_n)$ and $h^{\ominus}(m_0, m_1, \ldots, m_n)$ denote the joint probability density functions (pdf) of the part scores conditioned on the true label being positive $y = \oplus$ and negative $y = \ominus$, respectively. We make the following assumption.

Assumption. *The responses of the parts of a star-structured model with a given root location $x \in \mathcal{X}$ are independent conditioned on the the true label $y(x)$, i.e.*

$$\begin{aligned} h^{\oplus}(m_0, m_1, \ldots, m_n) &= \textstyle\prod_{k=0}^{n} h_k^{\oplus}(m_k), \\ h^{\ominus}(m_0, m_1, \ldots, m_n) &= \textstyle\prod_{k=0}^{n} h_k^{\ominus}(m_k), \end{aligned} \tag{2}$$

where $h_k^{\oplus}(m_k)$ is the pdf of $M_k \mid y = \oplus$ and $h_k^{\ominus}(m_k)$ is the pdf of $M_k \mid y = \ominus$.

We learn non-parametric representations for the $2(n+1)$ pdfs $\{h_k^{\oplus}, h_k^{\ominus}\}$ from an annotated set D of training images. We emphasize that the above assumption does not always hold in practice but simplifies the representation of the score likelihoods significantly[1] and avoids overfitting. Our algorithm for choosing a part order and a stopping time can be used without the independence assumption. However, we expect the performance to be similar while an unreasonable amount of training data would be required to learn a good representation of the joint pdfs. To evaluate the fidelity of the decoupled representation in (2) we computed correlation coefficients between all pairs of part responses (Table 1) for the classes in the PASCAL VOC 2007 dataset. The mean over all classes, 0.23, indicates a weak correlation. We observed that the few highly correlated parts have identical appearances (e.g. car wheels) or a spatial overlap.

To learn representations for the score likelihoods, $\{h_k^{\oplus}, h_k^{\ominus}\}$, we collected a set of scores for each part from the the training set D. Given a positive example $I_i^{\oplus} \in D$ of a particular DPM component, the root was placed at the scale and position x^* of the top score within the ground-truth bounding box. The

[1] Removing the independence assumption would require learning the 2 joint $(n+1)$ dimensional pdfs of the part scores in (2) and extracting the $2(n+1)$ marginals and the $2(n+1)(2^n-1)$ conditionals of the form $h(m_k \mid m_I)$, where $I \subseteq \{0, \ldots, n\} \setminus \{k\}$.

Table 1. Average correlation coefficients among pairs of part responses for all 20 classes in the VOC 2007 dataset

aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	**mean**
0.36	0.37	0.14	0.18	0.24	0.29	0.40	0.16	0.13	0.17	0.44	0.11	0.23	0.21	0.14	0.21	0.26	0.22	0.24	0.20	0.23

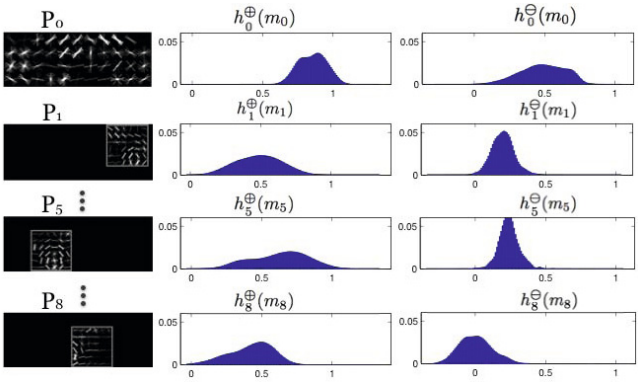


Fig. 2. Score likelihoods for several parts from a car DPM model. The root (P_0) and three parts of the model are shown on the left. The corresponding positive and negative score likelihoods are shown on the right.

response m_0^i of the root filter was recorded. The parts were placed at their optimal locations relative to the root location x^* and their scores $m_k^i,\ k > 0$ were recorded as well. This procedure was repeated for all positive examples in D to obtain a set of scores $\{m_k^i \mid \oplus\}$ for each part k. For negative examples, x^* was selected randomly over all locations in the image pyramid and the same procedure was used to obtain the set $\{m_k^i \mid \ominus\}$. Kernel density estimation was applied to the score collections in order to obtain smooth approximations to $h_k^{\oplus}$ and $h_k^{\ominus}$. Fig. 2 shows several examples of the score likelihoods obtained from the part responses of a car model.

3.2 Active Part Selection

This section discusses how to select an ordered subset of the $n+1$ parts, which when applied at a given location $x \in \mathcal{X}$ has a small probability of mislabeling x. The detection at x proceeds in rounds $t = 0, \ldots, n{+}1$. The DPM inference applies the root and parts in a predefined topological ordering of the model structure. Here, we do not fix the order of the parts a priori. Instead, we select which part to run next *sequentially*, depending on the part responses obtained in the past. The part chosen at round t is denoted by $k(t)$ and can be any of the parts that have not been applied yet. We take a Bayesian approach and maintain a probability $p_t := \mathbb{P}(y = \oplus \mid m_{k(0)}, \ldots, m_{k(t-1)})$ of a positive label at location x conditioned on the part scores from the previous rounds. The state at time t consists of a

binary vector $s_t \in \{0,1\}^{n+1}$ indicating which parts have already been used and the information state $p_t \in [0,1]$. Let $S_t := \{s \in \{0,1\}^{n+1} \mid \mathbf{1}^T s = t\}$ be the set[2] of possible values for s_t. At the start of a detection, $s_0 = \mathbf{0}$ and $p_0 = 1/2$, since no parts have been used and we have an uninformative prior for the true label.

Suppose that part $k(t)$ is applied at time t and its score is $m_{k(t)}$. The indicator vector s_t of used parts is updated as:

$$s_{t+1} = s_t + e_{k(t)}. \tag{3}$$

Due to the independence of the score likelihoods (2), the posterior label distribution is computed using Bayes rule:

$$p_{t+1} = \frac{h^{\oplus}_{k(t)}(m_{k(t)})}{h^{\oplus}_{k(t)}(m_{k(t)}) + h^{\ominus}_{k(t)}(m_{k(t)})} p_t. \tag{4}$$

In this setting, we seek a conditional plan π, which chooses which part to run next or stops and decides on a label for x. Formally, such a plan is called a *policy* and is a function $\pi(s,p) : \{0,1\}^{n+1} \times [0,1] \to \{\ominus, \oplus, 0, \ldots, n\}$, which depends on the previously used parts s and the label distribution p. An admissible policy does not allow part repetitions and satisfies $\pi(\mathbf{1}, p) \in \{\ominus, \oplus\}$ for all $p \in [0,1]$, i.e. has to choose a label after all parts have been used. The set of admissible policies is denoted by Π.

Let $\tau(\pi) := \inf\{t \geq 0 \mid \pi(s_t, p_t) \in \{\ominus, \oplus\}\} \leq n+1$ denote the stopping time of policy $\pi \in \Pi$. Let $\hat{y}_\pi \in \{\ominus, \oplus\}$ denote the label guessed by policy π after its termination. We would like to choose a policy, which decides *quickly* and *correctly*. To formalize this, define the probability of making an error as $Pe(\pi) := \mathbb{P}(\hat{y}_\pi \neq y)$, where y is the hidden correct label of x.

Problem (Active Part Selection). *Given $\epsilon > 0$, choose an admissible part policy π with minimum expected stopping time and probability of error bounded by ϵ:*

$$\begin{aligned} \min_{\pi \in \Pi} \quad & \mathbb{E}[\tau(\pi)] \\ s.t. \quad & Pe(\pi) \leq \epsilon, \end{aligned} \tag{5}$$

where the expectation is over the label y and the part scores $M_{k(0)}, \ldots, M_{k(\tau-1)}$.

Note that if ϵ is chosen too small, (5) might be infeasible. In other words, even the best sequencing of the parts might not reduce the probability of error sufficiently. To avoid this issue, we relax the constraint in (5) by introducing a Lagrange multiplier $\lambda > 0$ as follows:

$$\min_{\pi \in \Pi} \quad \mathbb{E}[\tau(\pi)] + \lambda Pe(\pi). \tag{6}$$

[2] *Notation*: $\mathbf{1}$ denotes a vector with all elements equal to one, $\mathbf{0}$ denotes a vector with all elements equal to zero, and e_i denotes a vector with one in the i-th component and zero everywhere else.

The Lagrange multiplier λ can be interpreted as a cost paid for choosing an incorrect label. To elaborate on this, we rewrite the cost function as follows:

$$\begin{aligned}
&\mathbb{E}\Big[\tau + \lambda \mathbb{E}_y\big[\mathbb{1}_{\{\hat{y}\neq y\}} \mid M_{k(0)}, \ldots, M_{k(\tau-1)}\big]\Big] \\
&\quad = \mathbb{E}\Big[\tau + \lambda \mathbb{1}_{\{\hat{y}\neq \oplus\}} \mathbb{P}\big(y = \oplus \mid M_{k(0)}, \ldots, M_{k(\tau-1)}\big) \\
&\qquad\qquad + \lambda \mathbb{1}_{\{\hat{y}\neq \ominus\}} \mathbb{P}\big(y = \ominus \mid M_{k(0)}, \ldots, M_{k(\tau-1)}\big)\Big] \\
&\quad = \mathbb{E}\Big[\tau + \lambda p_\tau \mathbb{1}_{\{\hat{y}=\ominus\}} + \lambda(1-p_\tau)\mathbb{1}_{\{\hat{y}=\oplus\}}\Big].
\end{aligned}$$

The term λp_τ above is the cost paid if label $\hat{y} = \ominus$ is chosen incorrectly. Similarly, $\lambda(1-p_\tau)$ is the cost paid if label $\hat{y} = \oplus$ is chosen incorrectly. To allow flexibility, we introduce separate costs λ_{fp} and λ_{fn} for false positive and false negative mistakes. The final form of the **Active Part Selection** problem is:

$$\min_{\pi \in \Pi} \quad \mathbb{E}\Big[\tau + \lambda_{fn} p_\tau \mathbb{1}_{\{\hat{y}=\ominus\}} + \lambda_{fp}(1-p_\tau)\mathbb{1}_{\{\hat{y}=\oplus\}}\Big]. \tag{7}$$

Computing the Part Selection Policy. Problem (7) can be solved using Dynamic Programming [1]. For a fixed policy $\pi \in \Pi$ and a given initial state $s_0 \in \{0,1\}^{n+1}$ and $p_0 \in [0,1]$, the value function:

$$V_\pi(s_0, p_0) := \mathbb{E}\Big[\tau + \lambda_{fn} p_\tau \mathbb{1}_{\{\hat{y}=\ominus\}} + \lambda_{fp}(1-p_\tau)\mathbb{1}_{\{\hat{y}=\oplus\}}\Big],$$

is a well-defined quantity. The *optimal* policy π^* and the corresponding *optimal* value function are obtained as:

$$\begin{aligned}
V^*(s_0, p_0) &= \min_{\pi \in \Pi} V_\pi(s_0, p_0), \\
\pi^*(s_0, p_0) &= \arg\max_{\pi \in \Pi} V_\pi(s_0, p_0).
\end{aligned}$$

To compute π^* we proceed backwards in time. Suppose that the policy has not terminated by time $t = n+1$. Since there are no parts left to apply the policy is forced to terminate. Thus, $\tau = n+1$ and $s_{n+1} = \mathbf{1}$ and for all $p \in [0,1]$ the optimal value function becomes:

$$\begin{aligned}
V^*(\mathbf{1}, p) &= \min_{\hat{y} \in \{\ominus, \oplus\}} \Big\{\lambda_{fn} p \mathbb{1}_{\{\hat{y}=\ominus\}} + \lambda_{fp}(1-p)\mathbb{1}_{\{\hat{y}=\oplus\}}\Big\} \\
&= \min\{\lambda_{fn} p, \lambda_{fp}(1-p)\}.
\end{aligned} \tag{8}$$

The intermediate stage values for $t = n, \ldots, 0$, $s_t \in S_t$, and $p_t \in [0,1]$ are:

$$\begin{aligned}
V^*(s_t, p_t) = \min\Big\{&\lambda_{fn} p_t, \lambda_{fp}(1-p_t), \\
&1 + \min_{k \in \mathcal{A}(s_t)} \mathbb{E}_{M_k} V^*\Big(s_t + e_k, \frac{h_k^{\oplus}(M_k) p_t}{h_k^{\oplus}(M_k) + h_k^{\ominus}(M_k)}\Big)\Big\},
\end{aligned} \tag{9}$$

where $\mathcal{A}(s) := \{i \in \{0, \ldots, n\} \mid s_i = 0\}$ is the set of available (unused) parts[3]. The optimal policy is readily obtained from the optimal value function. At stage t, if the first term in (9) is smallest, the policy stops and chooses $\hat{y} = \ominus$; if the second term is smallest, the policy stops and chooses $\hat{y} = \oplus$; otherwise, the policy chooses to run the part k, which minimizes the expectation.

Alg. 1 summarizes the steps necessary to compute the optimal policy π^* using the score likelihoods $\{h_k^{\oplus}, h_k^{\ominus}\}$ from Sec. 3.1. The one dimensional space $[0, 1]$ of label probabilities p can be discretized into d bins in order to store the function π returned by Alg. 1. The memory required is $O(d2^{n+1})$ since the space $\{0, 1\}^{n+1}$ of used-part indicator vectors grows exponentially with the number of parts. Nevertheless, in practice the number of parts in a DPM is rarely more than 20 and Alg. 1 can be executed.

Algorithm 1. Active Part Selection

Input: Score likelihoods $\{h_k^{\ominus}, h_k^{\oplus}\}_{k=0}^{n}$ for all parts, false positive cost λ_{fp}, false negative cost λ_{fn}
Output: Policy $\pi : \{0, 1\}^{n+1} \times [0, 1] \to \{\ominus, \oplus, 0, \ldots, n\}$

$S_t := \{s \in \{0, 1\}^{n+1} \mid \mathbf{1}^T s = t\}$
$\mathcal{A}(s) := \{i \in \{0, \ldots, n\} \mid s_i = 0\}$ for $s \in \{0, 1\}^{n+1}$

$V(\mathbf{1}, p) := \min\{\lambda_{fn} p, \lambda_{fp}(1 - p)\}, \quad \forall p \in [0, 1]$

$$\pi(\mathbf{1}, p) := \begin{cases} \ominus, & \lambda_{fn} p \leq \lambda_{fp}(1 - p) \\ \oplus, & \text{otherwise} \end{cases}$$

for $t = n, n - 1, \ldots, 0$ **do**
  for $s \in S_t$ **do**
    for $k \in \mathcal{A}(s)$ **do**

$$Q(s, p, k) := \mathbb{E}_{M_k} V\left(s + e_k, \frac{h_k^{\oplus}(M_k) p}{h_k^{\oplus}(M_k) + h_k^{\ominus}(M_k)}\right)$$

    end for

$$V(s, p) := \min\left\{\lambda_{fn} p, \lambda_{fp}(1 - p), 1 + \min_{k \in \mathcal{A}(s)} Q(s, p, k)\right\}$$

$$\pi(s, p) := \begin{cases} \ominus, & V(s, p) = \lambda_{fn} p, \\ \oplus, & V(s, p) = \lambda_{fp}(1 - p), \\ \arg\min_{k \in \mathcal{A}(s)} Q(s, p, k), & \text{otherwise} \end{cases}$$

  end for
end for
return π

3.3 Active DPM Inference

A policy π is obtained *offline* using Alg. 1. During inference, π is used to select a sequence of parts to apply at each location $x \in \mathcal{X}$ in the image pyramid. Note that the labeling of each location is treated as an independent problem. Alg. 2 summarizes the ADPM inference process.

[3] Each score likelihood was discretized using 201 bins to obtain a histogram. Then, the expectation in (9) was computed as a sum over the bins. Alternatively, Monte Carlo integration can be performed by sampling from the Gaussian mixtures directly.

Algorithm 2. Active DPM Inference

1: **Input**: Image pyramid, model $(F_0, P_1, \ldots, P_n, b)$, score likelihoods $\{h_k^{\ominus}, h_k^{\oplus}\}_{k=0}^{n}$ for all parts, policy π
2: **Output**: $score(x)$ at all locations $x \in \mathcal{X}$ in the image pyramid
3:
4: **for** $x \in 1 \ldots |\mathcal{X}|$ **do** ▷ All image pyramid locations
5: $s_0 := \mathbf{0}$; $p_0 = 0.5$; $score(x) := 0$
6: **for** $t = 0, 1, \ldots, n$ **do**
7: $k := \pi(s_t, p_t)$ ▷ Lookup next best part
8: **if** $k = \oplus$ **then** ▷ Labeled as foreground
9: **for** $i \in \{0, 1, \ldots, n\}$ **do**
10: **if** $s_t(i) = 0$ **then**
11: Compute score $m_i(x)$ for part i ▷ $O(|\Delta|)$
12: $score(x) := score(x) + m_i(x)$
13: **end if**
14: **end for**
15: $score(x) := score(x) + b$ ▷ Add bias to final score
16: break;
17: **else if** $k = \ominus$ **then** ▷ Labeled as background
18: $score(x) := -\infty$
19: break;
20: **else** ▷ Update probability and score
21: Compute score $m_k(x)$ for part k ▷ $O(|\Delta|)$
22: $score(x) := score(x) + m_k(x)$
23: $p_{t+1} := \frac{h_k^{\oplus}(m_k(x))p_t}{h_k^{\oplus}(m_k(x))+h_k^{\ominus}(m_k(x))}$
24: $s_{t+1} = s_t + e_k$
25: **end if**
26: **end for**
27: **end for**

At the start of a detection at location x, $s_0 = \mathbf{0}$ since no parts have been used and $p_0 = 1/2$ assuming an uninformative label prior (LN. 5). At each round t, the policy is queried to obtain either the next part to run or a predicted label for x (LN. 7). Note that querying the policy is an $O(1)$ operation since it is stored as a lookup table. If the policy terminates and labels $y(x)$ as foreground (LN. 8), all unused part filters are applied in order to obtain the final discriminative score in (1). On the other hand, if the policy terminates and labels $y(x)$ as background, no additional part filters are evaluated and the final score is set to $-\infty$ (LN. 18). In this case, our algorithm makes computational savings compared to the DPM. The potential speed-up and the effect on accuracy are discussed in the Sec. 4. Finally, if the policy returns a part index k, the corresponding score $m_k(x)$ is computed by applying the part filter (LN. 21). This operation is $O(|\Delta|)$, where Δ is the space of possible displacements for part k with respect to the root location x. Following the analysis in [6], searching over the possible locations for part k is usually no more expensive than evaluating its linear filter F_k once. This is the case because once F_k is applied at some location x_k, the resulting response $\Phi_k(x_k) = F_k' \cdot \phi(H, x_k)$ is cached to avoid recomputing it later. The score m_k of part k is used to update the total score at x (LN. 22). Then, (3) and (4) are used to update the state (s_t, p_t) (LN. 23 - 24). Since the policy lookups and the state updates are all of $O(1)$ complexity, the worst-case complexity of Alg. 2 is $O(n|\mathcal{X}||\Delta|)$. The average running time of our algorithm depends on the

total number of score m_k evaluations, which in turn depends on the choice of the parameters λ_{fn} and λ_{fp} and is the subject of the next section.

4 Experiments

4.1 Speed-Accuracy Trade-Off

The accuracy and the speed of the ADPM inference depend on the penalty, λ_{fp}, for incorrectly predicting background as foreground and the penalty, λ_{fn}, for incorrectly predicting foreground as background. To get an intuition, consider making both λ_{fp} and λ_{fn} very small. The cost of an incorrect prediction will be negligible, thus encouraging the policy to sacrifice accuracy and stop immediately. In the other extreme, when both parameters are very large, the policy will delay the prediction as much as possible in order to obtain more information.

To evaluate the effect of the parameter choice, we compared the average precision (AP) and the number of part evaluations of Alg. 2 to those of the traditional DPM as a baseline. Let R_M be the total number of score $m_k(x)$ evaluations for $k > 0$ (excluding the root) over all locations $x \in \mathcal{X}$ performed by method M. For example, $R_{DPM} = n|\mathcal{X}|$ since the DPM evaluates all parts at all locations in $\mathcal{X}$. We define the **relative number of part evaluations** (RNPE) of ADPM versus method M as the ratio of R_M to R_{ADPM}. The AP and the RNPE versus DPM of ADPM were evaluated on several classes from the PASCAL VOC 2007 training set (see Fig. 3) for different values of the parameter $\lambda = \lambda_{fn} = \lambda_{fp}$. As expected, the AP increases while the RNPE decreases, as the penalty of an incorrect declaration λ grows, because ADPM evaluates more parts. The dip in RNPE for very low λ is due to fact that ADPM starts reporting many false-positives. In the case of a positive declaration all $n + 1$ part responses need to be computed which reduces the speed-up versus DPM.

To limit the number of false positive mistakes made by the policy we set $\lambda_{fp} > \lambda_{fn}$. While this might hurt the accuracy, it will certainly result in less positive declarations and in turn significantly less part evaluations. To verify this intuition we performed experiments with $\lambda_{fp} > \lambda_{fn}$ on the VOC 2007 training set. Table 2 reports the AP and the RNPE versus DPM from a grid search over the parameter space. Generally, as the ratio between λ_{fp} and λ_{fn} increases, the RNPE increases while the AP decreases. Notice, however, that the increase in RNPE is significant, while the hit in accuracy is negligible.

In sum, λ_{fp} and λ_{fn} were selected with a grid search in parameter space with $\lambda_{fp} > \lambda_{fn}$ using the training set. Choosing different values for different classes should improve the performance even more.

4.2 Results

In this section we compare ADPM[4] versus two baselines, the DPM and the cascade DPM (Cascade) in terms of average precision (AP), relative number of

[4] ADPM source code is available at: `http://cis.upenn.edu/~menglong/adpm.html`

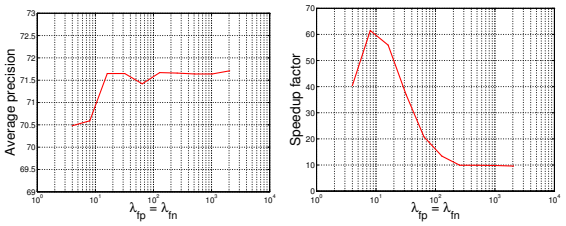


Fig. 3. Average precision and relative number of part evaluations versus DPM as a function of the parameter $\lambda = \lambda_{fn} = \lambda_{fp}$ on a log scale. The curves are reported on the bus class from the VOC 2007 **training** set.

Table 2. Average precision and relative number of part evaluations versus DPM obtained on the bus class from VOC 2007 **training** set. A grid search over $(\lambda_{fp}, \lambda_{fn}) \in \{4, 8, \ldots, 64\} \times \{4, 8, \ldots, 64\}$ with $\lambda_{fp} \geq \lambda_{fn}$ is shown.

Average Precision						RNPE vs DPM					
$\lambda_{fp}/\lambda_{fn}$	4	8	16	32	64	$\lambda_{fp}/\lambda_{fn}$	4	8	16	32	64
4	70.3					4	40.4				
8	70.0	71.0				8	80.7	61.5			
16	69.6	71.1	71.5			16	118.6	74.5	55.9		
32	70.5	70.7	71.6	71.6		32	178.3	82.1	59.8	37.0	
64	67.3	69.6	71.5	71.6	71.4	64	186.9	96.4	56.2	34.5	20.8

Table 3. Average precision (AP) and relative number of part evaluations (RNPE) of DPM versus ADPM on all 20 classes in VOC 2007 and 2010

VOC2007	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	**mean**
DPM RNPE	102.8	106.7	63.7	79.7	58.1	155.2	44.5	40.0	58.9	71.8	69.9	49.2	51.0	59.6	45.3	49.0	62.6	68.6	79.0	100.6	**70.8**
DPM AP	33.2	60.3	10.2	16.1	27.3	54.3	58.2	23.0	20.0	24.1	26.7	12.7	58.1	48.2	43.2	12.0	21.1	36.1	46.0	43.5	**33.7**
ADPM AP	33.5	59.8	9.8	15.3	27.6	52.5	57.6	22.1	20.1	24.6	24.9	12.3	57.6	48.4	42.8	12.0	20.4	35.7	46.3	43.2	**33.3**
VOC2010	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	**mean**
DPM RNPE	110.0	100.8	47.9	98.8	111.8	214.4	75.6	202.5	150.8	147.2	62.4	126.2	133.7	187.1	114.4	59.3	24.3	131.2	143.8	106.0	**117.4**
DPM AP	45.6	49.0	11.0	11.6	27.2	50.5	43.1	23.6	17.2	23.2	10.7	20.5	42.5	44.5	41.3	8.7	29.0	18.7	40.0	34.5	**29.6**
ADPM AP	45.3	49.1	10.2	12.2	26.9	50.6	41.9	22.7	16.5	22.8	10.6	19.7	40.8	44.5	36.8	8.3	29.1	18.6	39.7	34.5	**29.1**

Table 4. Average precision (AP), relative number of part evaluations (RNPE), and relative wall-clock time speedup (Speedup) of ADPM versus Cascade on all 20 classes in VOC 2007 and 2010

VOC2007	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	**mean**
Cascade RNPE	5.93	5.35	9.17	6.09	8.14	3.06	5.61	4.51	6.30	4.03	4.83	7.77	3.61	6.67	17.8	9.84	3.82	2.43	2.89	6.97	**6.24**
ADPM Speedup	3.14	1.60	8.21	4.57	3.36	1.67	2.11	1.54	3.12	1.63	1.28	2.72	1.07	1.50	3.59	6.15	2.92	1.10	1.11	3.26	**2.78**
Cascade AP	33.2	60.8	10.2	16.1	27.3	54.1	58.1	23.0	20.0	24.2	26.8	12.7	58.1	48.2	43.2	12.0	20.1	35.8	46.0	43.4	**33.7**
ADPM AP	31.7	59.0	9.70	14.9	27.5	51.4	56.7	22.1	20.4	24.0	24.7	12.4	57.7	48.5	41.7	11.6	20.4	35.9	45.8	42.8	**33.0**
VOC2010	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	**mean**
Cascade RNPE	7.28	2.66	14.80	7.83	12.22	5.47	6.29	6.33	9.72	4.16	3.74	10.77	3.21	9.68	21.43	12.21	3.23	4.58	3.98	8.17	**7.89**
ADPM Speedup	2.15	1.28	7.58	5.93	4.68	2.79	2.28	2.44	3.72	2.42	1.52	2.76	1.57	2.93	4.72	8.24	1.42	1.81	1.47	3.41	**3.26**
Cascade AP	45.5	48.9	11.0	11.6	27.2	50.5	43.1	23.6	17.2	23.1	10.7	20.5	42.4	44.5	41.3	8.7	29.0	18.7	40.1	34.4	**29.6**
ADPM AP	44.5	49.2	9.5	11.6	25.9	50.6	41.7	22.5	16.9	22.0	9.8	19.8	41.1	45.1	40.2	7.4	28.5	18.3	38.0	34.5	**28.8**

Table 5. An example demonstrating the computational time breakdown during inference of ADPM and Cascade on a single image. The number of part evaluations (PE) and the inference time (in sec) is recorded for the PCA and the full-dimensional stages. The results are reported once without and once with cache use. The number of part evaluations is independent of caching.

	PCA no cache	PCA cache	PE	Full no cache	Full cache	PE	Total no cache	Total cache	Total PE
CASCADE	4.34s	0.67s	208K	0.13s	0.08s	1.1K	4.50s	0.79s	209K
ADPM	0.62s	0.06s	36K	0.06s	0.04s	0.6K	0.79s	0.19s	37K

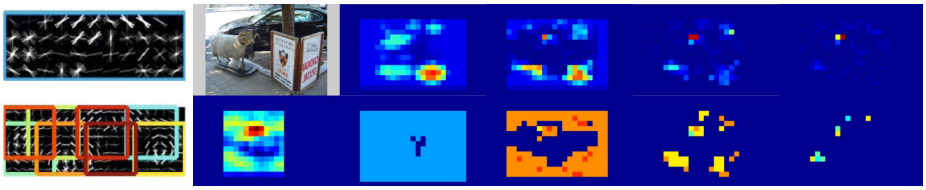
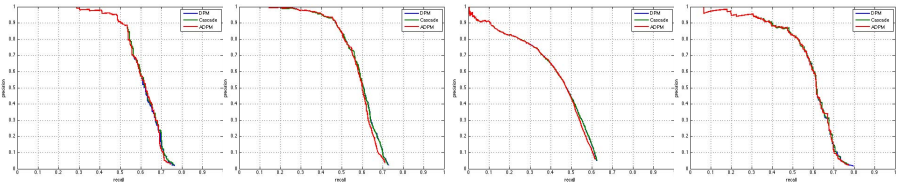


Fig. 4. Illustration of the ADPM inference process on a car example. The DPM model with colored root and parts is shown on the left. The top row on the right consists of the input image and the evolution of the positive label probability (p_t) for $t \in \{1, 2, 3, 4\}$ (high values are red; low values are blue). The bottom row consists of the full DPM $score(x)$ and a visualization of the parts applied at different locations at time t. The pixel colors correspond to the part colors on the left. In this example, despite the car being heavily occluded, ADPM converges to the correct location after four iterations.



(a) class: bicycle (b) class: car (c) class: person (d) class: horse

Fig. 5. Precision recall curves for bicycle, car, person, and horse classes from VOC 2007. Our method's accuracy ties with the baselines.

part evaluations (RNPE), and relative wall-clock time speedup (Speedup). Experiments were carried out on all 20 classes in the PASCAL VOC 2007 and 2010 datasets. Publicly available PASCAL VOC 2007 and 2010 DPM and Cascade models were used for all three methods. For a fair comparison, ADPM changes only the part order and the stopping criterion of the original implementations.

ADPM vs DPM: The inference of ADPM on two input images is shown in detail in Fig. 1 and Fig. 4. The probability of a positive label p_t (top row) becomes more contrasted as additional parts are evaluated. The locations at which the algorithm has not terminated decrease rapidly as time progresses. Visually, the locations with a maximal posterior are identical to the top scores obtained by the DPM. The order of parts chosen by the policy is indicative of

their informativeness. For example, in Fig. 4 the wheel filters are applied first which agrees with intuition. In this example, the probability p_t remains low at the correct location for several iterations due to the occlusions. Nevertheless, the policy recognizes that it should not terminate and as it evaluates more parts, the correct location of the highest DPM score is reflected in the posterior.

ADPM was compared to DPM in terms of AP and RNPE to demonstrate the ability of ADPM to reduce the number of part evaluations with minimal loss in accuracy irrespective of the features used. The parameters were set to $\lambda_{fp} = 20$ and $\lambda_{fn} = 5$ for all classes based on the analysis in Sec. 4.1. Table 3 shows that ADPM achieves a significant decrease (90 times on average) in the number of evaluated parts with negligible loss in accuracy. The precision-recall curves of the two methods are shown in Fig. 5 for several classes.

ADPM vs Cascade: The improvement in detection speed achieved by ADPM is demonstrated via a comparison to Cascade in terms of AP, RNPE, and wall-clock time (in sec). During inference, Cascade prunes the image locations in two passes. In the first pass, the locations are filtered using the PCA filters and the low-scoring ones are discarded. In the second pass, the remaining locations are filtered using the full-dimensional filters. To make a fair comparison, we adopted a similar **two-stage** approach. An **additional** policy was learned using PCA score likelihoods and was used to schedule PCA filters during the first pass. The locations, which were selected as foreground in the first stage, were filtered again, using the original policy to schedule the full-dimensional filters. The parameters λ_{fp} and λ_{fn} were set to 20 and 5 for the PCA policy and to 50 and 5 for the full-dimensional policy. A higher λ_{fp} was chosen to make the prediction more precise (albeit slower) during the second stage. Deformation pruning was not used for either method. Table 4 summarizes the results. A discrepancy in the speedup of ADPM versus Cascade is observed in Table 4. On average, ADPM is 7 times faster than Cascade in RNPE but only 3 times faster in seconds. A breakdown of the computational time during inference on a single image is shown in Table 5. We observe that the ratios of part evaluations and of seconds are consistent within individual stages (PCA and full). However, a single filter evaluation during the full-filter stage is significantly slower than one during the PCA stage. This does not affect the cumulative RNPE but lowers the combined seconds ratio. While ADPM is significantly faster than Cascade during the PCA stage, the speedup (in sec) is reduced during the slower full-dimensional stage.

5 Conclusion

This paper presents an active part selection approach which substantially speeds up inference with pictorial structures without sacrificing accuracy. Statistics learned from training data are used to pose an optimization problem, which balances the number of part filter convolution with the classification accuracy. Unlike existing approaches, which use a pre-specified part order and hard stopping thresholds, the resulting part scheduling policy selects the part order and the stopping criterion adaptively based on the filter responses obtained during

inference. Potential future extensions include optimizing the part selection across scales and image positions and detecting multiple classes simultaneously.

References

1. Bertsekas, D.P.: Dynamic Programming and Optimal Control, vol. 1. Athena Scientific (1995)
2. Bourdev, L., Brandt, J.: Robust object detection via soft cascade. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005, vol. 2, pp. 236–243. IEEE (2005)
3. Brubaker, S.C., Wu, J., Sun, J., Mullin, M.D., Rehg, J.M.: On the design of cascades of boosted ensembles for face detection. IJCV 77(1-3), 65–86 (2008)
4. Dollár, P., Appel, R., Kienzle, W.: Crosstalk cascades for frame-rate pedestrian detection. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part II. LNCS, vol. 7573, pp. 645–659. Springer, Heidelberg (2012)
5. Everingham, M., Van Gool, L., Williams, C., Winn, J., Zisserman, A.: The Pascal Visual Object Classes (VOC) Challenge. IJCV 88(2), 303–338 (2010)
6. Felzenszwalb, P.F., Girshick, R.B., McAllester, D.: Cascade object detection with deformable part models. In: CVPR, pp. 2241–2248. IEEE (2010)
7. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part-based models. PAMI 32(9), 1627–1645 (2010)
8. Fleuret, F., Geman, D.: Coarse-to-fine face detection. IJCV (2001)
9. Gao, T., Koller, D.: Active classification based on value of classifier. In: NIPS (2011)
10. Gualdi, G., Prati, A., Cucchiara, R.: Multistage particle windows for fast and accurate object detection. PAMI 34(8), 1589–1604 (2012)
11. Kapoor, A., Grauman, K., Urtasun, R., Darrell, T.: Gaussian processes for object categorization. IJCV (2010)
12. Karayev, S., Fritz, M., Darrell, T.: Anytime recognition of objects and scenes. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (oral, to appear, 2014)
13. Kokkinos, I.: Rapid deformable object detection using dual-tree branch-and-bound. In: NIPS (2011)
14. Lampert, C.H.: An efficient divide-and-conquer cascade for nonlinear object detection. In: CVPR. IEEE (2010)
15. Lampert, C.H., Blaschko, M.B., Hofmann, T.: Beyond sliding windows: Object localization by efficient subwindow search. In: CVPR, pp. 1–8. IEEE (2008)
16. Lehmann, A., Gehler, P.V., Van Gool, L.J.: Branch&rank: Non-linear object detection. In: BMVC (2011)
17. Lehmann, A., Leibe, B., Van Gool, L.: Fast prism: Branch and bound hough transform for object class detection. IJCV 94(2), 175–197 (2011)
18. Pedersoli, M., Vedaldi, A., Gonzalez, J.: A coarse-to-fine approach for fast deformable object detection. In: CVPR, pp. 1353–1360. IEEE (2011)
19. Rahtu, E., Kannala, J., Blaschko, M.: Learning a category independent object detection cascade. In: ICCV (2011)
20. Sapp, B., Toshev, A., Taskar, B.: Cascaded models for articulated pose estimation. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part II. LNCS, vol. 6312, pp. 406–420. Springer, Heidelberg (2010)

21. Sznitman, R., Becker, C., Fleuret, F., Fua, P.: Fast object detection with entropy-driven evaluation. In: CVPR (June 2013)
22. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: CVPR. IEEE (2001)
23. Weiss, D., Sapp, B., Taskar, B.: Structured prediction cascades. arXiv preprint arXiv:1208.3279 (2012)
24. Wu, T., Zhu, S.-C.: Learning near-optimal cost-sensitive decision policy for object detection. In: 2013 IEEE International Conference on Computer Vision (ICCV), pp. 753–760. IEEE (2013)
25. Zhang, Z., Warrell, J., Torr, P.H.: Proposal generation for object detection using cascaded ranking svms. In: CVPR, pp. 1497–1504. IEEE (2011)