SPADE: Scalar Product Accelerator by Integer Decomposition for Object Detection

Mitsuru Ambai and Ikuro Sato

Denso IT Laboratory, Inc.
{manbai,isato}@d-itlab.co.jp

Abstract. We propose a method for accelerating computation of an object detector based on a linear classifier when objects are expressed by binary feature vectors. Our key idea is to decompose a real-valued weight vector of the linear classifier into a weighted sum of a few ternary basis vectors so as to preserve the original classification scores. Our data-dependent decomposition algorithm can approximate the original classification scores by a small number of the ternary basis vectors with an allowable error. Instead of using the original real-valued weight vector, the approximated classification score can be obtained by evaluating the few inner products between the binary feature vector and the ternary basis vectors, which can be computed using extremely fast logical operations. We also show that each evaluation of the inner products can be cascaded for incorporating early termination. Our experiments revealed that the linear filtering used in a HOG-based object detector becomes 36.9× faster than the original implementation with 1.5% loss of accuracy for 0.1 false positives per image in pedestrian detection task.

Keywords: linear classifier, binary features, object detection.

1 Introduction

In spite of its simplicity, a linear classifier is widely acknowledged as a powerful tool for object detection. In many cases, both training and detection time of the linear classifier are greatly reduced compared to non-linear classifiers such as deep neural networks[11] and kernel methods[9,14]. Although the linear classifier simply defines a decision boundary by a hyper plane in a feature space, recent feature representations, e.g. Histograms of Oriented Gradients (HOG)[3], Fisher vector[16,15], explicit feature maps[21] and deformable part models[7], produce comparable classification performances to the non-linear object detectors.

However, even for the linear classifier, the detection time differs from real-time due to the fact that the detection task is done by sliding window approach, in which the linear classifier is exhaustively applied at all possible locations on an image. Coupled with the high-dimensionality of the recent feature representations, the computational cost of evaluating classification scores is enormous. This drawback becomes more serious when a part-based model is used[7] because such an object model must compute scores of multiple linear part filters. However,

D. Fleet et al. (Eds.): ECCV 2014, Part V, LNCS 8693, pp. 267–281, 2014.
© Springer International Publishing Switzerland 2014

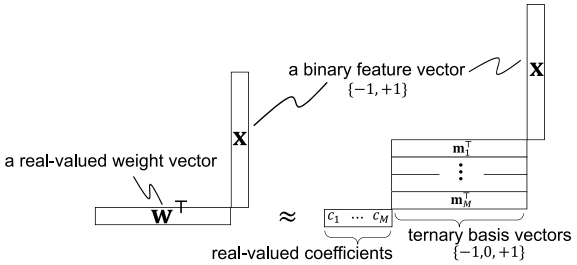


Fig. 1. A basic idea of our method. Classification score is approximated by weighted sum of inner products between binary feature vector and ternary basis vectors.

reducing the computational load is crucial issue for industrial applications, such as in-vehicle safety system.

Our aim with this study is to accelerate the score computation of the linear object detectors, such as HOG with SVM. Figure 1 illustrates the basic idea of our method. Our work was inspired by recent research on binary hashing[8,23] and binary descriptors[18,13,1], which represent a visual feature as a binary vector. In our framework, a feature vector is restricted to binary values -1 and +1. In this case, the classification score is formulated as an inner product between a real-valued weight vector (trained by a machine learning technique such as SVM) and the binary feature vector. Our key idea is to decompose the real-valued weight vector into a weighted sum of a few ternary basis vectors that only contain -1, 0, and +1. In this case, the classification score is approximated by the weighted sum of a few inner products between the ternary basis vectors and the binary feature vector. Instead of using time-consuming floating-point operations, each of the inner products can be computed extremely fast by simple logical operations such as XOR, AND, and bit counts. For this idea to work well, it is important to approximate the real-valued weight vector into a small number of ternary vectors with an allowable error. To address this issue, we introduce a data-dependent decomposition algorithm that minimizes the sum of squared errors between the original and approximated classification scores of training samples. In addition, cascading approach is introduced to reject a large number of object candidates without evaluating all the decomposed inner products.

1.1 Related Work

There have been extensive studies on accelerating linear object detectors. We review just a few representatives. Felzenszwalb *et al.*[6] introduced an idea of rejection cascade popularized by Viola and Johns[22] into deformable part models. Dubout *et al.*[5] processed the sliding-window filtering in a frequency domain. The filtering is accelerated by using a fast Fourier transform without any approximations. If a large number of multiple linear filters are needed, one solution is to approximate each of the given filters by a weighted sum of a smaller number of shared filters[20,19,17]. Song *et al.*[20,19] used this idea for multi-class object

detection based on deformable part models. In their approach, a large number of part filters are decomposed into shared filters and are approximated by sparse linear combinations of them. Rigamonti *et al.*[17] proposed a method for finding shared filters under the condition in which each are separable. Since the responses of the separable filters can be computed by applying one-dimensional filters twice in the x- and y- directions, the total computation time can be drastically reduced. Lampert *et al.*[12] revealed that a branch-and-bound approach can directly find peaks of a filter response without exhaustive sliding-window search if a good quality bounding function is given. The closest idea to ours was proposed by Hare *et al.*[10] within the context of matching binary local descriptors. In their work, classifier weights are decomposed into a few binary basis vectors in a similar manner to ours. However, as shown later, our approach approximates the classifier weights with significantly smaller errors than the proposed by Hare *et al.*[10]. They also did not discuss the rejection cascade.

1.2 Contributions

We call our framework consisted of the following three components as Scalar Product Accelerator by integer DEcomposition (SPADE).

1. **Ternary representation:** The classification score is approximated using a small number of ternary basis vectors, as illustrated in Figure 1. We show that the inner product between the ternary basis vector and the binary feature vector can be computed extremely fast by a combination of three logical operations: XOR, AND, and bit counts.
2. **Data-dependent decomposition algorithm:** The ternary basis vectors and their coefficients are optimized to minimize the sum of squared errors between the original and approximated classifier scores of training samples. Although this minimization is a hard combinatorial optimization problem, we propose an efficient method for finding an approximated solution.
3. **Rejection cascade:** When M ternary basis vectors are given, we show that M-stage cascade can be built for early termination. With the introduction of the rejection cascade, we also propose a method for determining safe thresholds that do not decrease classification accuracy.

The rest of this paper is structured as follows. In Section 2, we introduce the three components of SPADE. In Section 3, we explain experimental results and give a discussion. We give concluding remarks in Section 4. Throughout this paper, AND, XOR, and bit count operations are denoted by $L_{\mathrm{and}}(\mathbf{x}_1, \mathbf{x}_2)$, $L_{\mathrm{xor}}(\mathbf{x}_1, \mathbf{x}_2)$ and $L_{\mathrm{pop}}(\mathbf{x}_1)$, respectively, where $\mathbf{x}_1$and $\mathbf{x}_2$ are binary vectors that contains only -1 or $+1$. The positive values are regarded as true bits, and the negative values are regarded as false bits.

2 Scalar Product Accelerator by Integer Decomposition

In this section, we introduce three key ideas of SPADE: ternary representation, data-dependent decomposition algorithm and rejection cascade. First we begin with formulating the linear classifier as follows.

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b, \tag{1}$$

where $\mathbf{w} \in \mathbb{R}^D$ is a weight vector, $\mathbf{x} \in \{-1, +1\}^D$ is a binary feature vector extracted from an image, and $b \in \mathbb{R}$ is a bias term. Since the method for extracting a good binary feature is an application-specific issue, we skip this discussion until Section 3. In this section, we assume that the binary feature is properly designed for the object detection task.

Even though $\mathbf{x}$ is binary, Eq. (1) still requires $\mathcal{O}(D)$ floating-point operations. To reduce them, we introduce the following approximated classifier.

$$f_{\text{approx}}(\mathbf{x}) = \sum_{i=1}^{M} c_i \mathbf{m}_i^\top \mathbf{x} + b, \tag{2}$$

where $\mathbf{m}_i$ is an i-th integer basis vector, $c_i \in \mathbb{R}$ is its coefficient, and M is the number of the basis vectors used for this approximation. The difference between Eqs. (1) and (2) is that $\mathbf{w}$ is replaced with a linear combination of the integer vectors $\sum_{i=1}^{M} c_i \mathbf{m}_i$.

Hare *et al.*[10] investigated the case in which $\mathbf{m}_i \in \{-1, +1\}$. In this case, the inner product $\mathbf{m}_i^\top \mathbf{x}$ can be quickly computed using XOR followed by bit counts as follows[1]:

$$\mathbf{m}_i^\top \mathbf{x} = D - 2 \cdot L_{\text{pop}}(L_{\text{xor}}(\mathbf{m}_i, \mathbf{x})). \tag{3}$$

In this way, the order of floating-point operations to obtain $f_{\text{approx}}(\mathbf{x})$ is reduced from $\mathcal{O}(D)$ to $\mathcal{O}(M)$. For this method to work well, M must be small enough. Hare *et al.*[10] reported that only two binary basis vectors are sufficient for classifying binary local descriptors in the context of keypoint tracking. However, for accelerating a linear object detector, such as a HOG with an SVM, we found that much larger number of basis vectors is necessary to preserve the original classification accuracy. In the following three subsections, we introduce three key ideas of SPADE that greatly improve this drawback.

2.1 Ternary Representation

Even if M is set to a small value, good approximation can be obtained by permitting various values to be taken in $\mathbf{m}_i$. However, fast computation of $\mathbf{m}_i^\top \mathbf{x}$ may become more difficult. There is a trade-off between computation time and approximation quality depending on constraints on $\mathbf{m}_i$.

From the viewpoint of this trade-off, a good balanced approach is to use a ternary vector $\mathbf{m}_i = (m_{i1}, \cdots, m_{iD})^\top \in \{-1, 0, +1\}^D$. Even in this case, logical operations are available to compute $\mathbf{m}_i^\top \mathbf{x}$ similar to Eq. (3). By introducing $N_{\text{plus}}(\mathbf{m}_i, \mathbf{x})$ and $N_{\text{minus}}(\mathbf{m}_i, \mathbf{x})$, $\mathbf{m}_i^\top \mathbf{x}$ can be rewritten as follows.

$$\mathbf{m}_i^\top \mathbf{x} = N_{\text{plus}}(\mathbf{m}_i, \mathbf{x}) - N_{\text{minus}}(\mathbf{m}_i, \mathbf{x}), \tag{4}$$

[1] In the original work [10], this inner product is computed using AND operation instead of XOR. However, there are no essential difference between them.

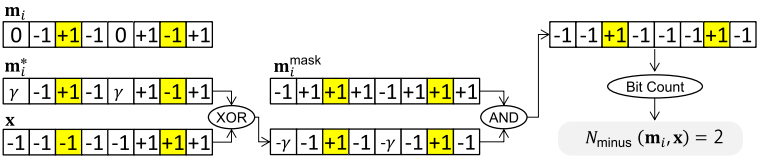


Fig. 2. N_{minus} is obtained by XOR, AND, and bit count operations by introducing two binary vectors $\mathbf{m}_i^*$ and $\mathbf{m}_i^{\text{mask}}$ derived from $\mathbf{m}_i$. Locations that satisfy $m_{ij}x_j = -1$ are indicated by yellow boxes.

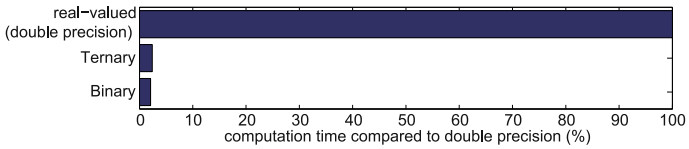


Fig. 3. Comparison of computation times of $\mathbf{m}_i^\top \mathbf{x}$. Three cases of $\mathbf{m}_i$: binary, ternary and real-valued, are shown in this figure. A dimension D is set to 4896 in this test.

where $N_{\text{plus}}(\mathbf{m}_i, \mathbf{x})$ and $N_{\text{minus}}(\mathbf{m}_i, \mathbf{x})$ are numbers of elements that satisfy $m_{ij}x_j = +1$ and $m_{ij}x_j = -1$, respectively. A formal definition of $N_{\text{minus}}(\mathbf{m}_i, \mathbf{x})$ is

$$N_{\text{minus}}(\mathbf{m}_i, \mathbf{x}) = \sum_{j=1}^{D} I(m_{ij}, x_j), \quad I(m_{ij}, x_j) = \begin{cases} 1 \text{ if } m_{ij}x_j = -1 \\ 0 \quad \text{otherwise} \end{cases}, \tag{5}$$

where $I(m_{ij}, x_j)$ is an indicator function. If $N_{\text{minus}}(\mathbf{m}_i, \mathbf{x})$ are given, $N_{\text{plus}}(\mathbf{m}_i, \mathbf{x})$ can be easily obtained as follows.

$$N_{\text{plus}}(\mathbf{m}_i, \mathbf{x}) = D - z_i - N_{\text{minus}}(\mathbf{m}_i, \mathbf{x}), \tag{6}$$

where z_i is a number of zero values in $\mathbf{m}_i$ and is pre-computable because $\mathbf{m}_i$ is determined at the training phase and is fixed when detecting objects. By substituting Eq. (6) into Eq. (4), we obtain

$$\mathbf{m}_i^\top \mathbf{x} = D - z_i - 2 \cdot N_{\text{minus}}(\mathbf{m}_i, \mathbf{x}). \tag{7}$$

In practice, $N_{\text{minus}}(\mathbf{m}_i, \mathbf{x})$ can be quickly computed by logical operations without the element-wise summation shown in Eq. (5). This is illustrated in Figure 2. Instead of directly using $\mathbf{m}_i$, we prepared two binary vectors $\mathbf{m}_i^{\text{mask}} = (m_{i1}^{\text{mask}}, \cdots, m_{iD}^{\text{mask}})^\top$ and $\mathbf{m}_i^* = (m_{i1}^*, \cdots, m_{iD}^*)^\top$, which are defined as follows.

$$m_{ij}^{\text{mask}} = \begin{cases} -1 & \text{if } m_{ij} = 0 \\ +1 & \text{otherwise} \end{cases}, \quad m_{ij}^* = \begin{cases} \gamma & \text{if } m_{ij} = 0 \\ m_{ij} & \text{otherwise} \end{cases}, \tag{8}$$

where γ is an arbitrary value that takes -1 or $+1$, $\mathbf{m}_i^{\text{mask}}$ and $\mathbf{m}_i^*$ are binary vectors, and $N_{\text{minus}}(\mathbf{m}_i, \mathbf{x})$ is given by the following equation.

$$N_{\text{minus}}(\mathbf{m}_i, \mathbf{x}) = L_{\text{pop}}(L_{\text{and}}(L_{\text{xor}}(\mathbf{m}_i^*, \mathbf{x}), \mathbf{m}_i^{\text{mask}})). \tag{9}$$

The first XOR operation finds positions that satisfy $m_{ij}x_j = -1$. Although this operation may also find positions that satisfy $m_{ij}x_j = 0$, they are filtered out by the following AND operation. Finally, by counting the number of true bits, we obtain $N_{\text{minus}}(\mathbf{m}_i, \mathbf{x})$. It should be noted that $N_{\text{minus}}(\mathbf{m}_i, \mathbf{x})$ does not depend on the choice of γ because the corresponding locations are filtered out by the AND operation. By substituting Eq. (9) into Eq. (7), $\mathbf{m}_i^\top \mathbf{x}$ is obtained very fast.

Computation times of $\mathbf{m}_i^\top \mathbf{x}$ are compared in Figure 3. Three types of constraints: binary, ternary, and real-valued (no constraints), are imposed on $\mathbf{m}_i$. A *popcnt* instruction of Intel Core i7 processor was used for computing $L_{\text{pop}}(\cdot)$. Interestingly, the computation time of the ternary representation was comparable to the binary representation, although the ternary representation potentially has a capability to reduce the approximation error better than the binary representation.

2.2 Decomposition Algorithms

In this subsection, we explain two different algorithms for decomposing $\mathbf{w}$ into $\mathbf{m}_i$ and c_i. One is a *data-dependent* algorithm and the other is a *data-independent* algorithm. These two algorithms minimize different cost function. The former uses training datasets, but the latter does not. First, we introduce each cost function and discuss their advantages and disadvantages. Next, two optimization algorithms that minimize them are proposed. Finally, a connection between our study and a related study[10] is discussed.

To preserve a decision boundary, it is natural to minimize the sum of the squared differences between the original and approximated classifier scores. This requires training datasets $\mathbf{X} = (\mathbf{x}_1, \cdots, \mathbf{x}_N) \in \{-1, +1\}^{D \times N}$, where $\mathbf{X}$ contains positive and negative samples used to train $\mathbf{w}$ and b. The cost function is defined as follows.

$$E_1 = \sum_{k=1}^{N} (f(\mathbf{x}_k) - f_{\text{approx}}(\mathbf{x}_k))^2 = ||\mathbf{w}^\top \mathbf{X} - (\sum_{i=1}^{M} c_i \mathbf{m}_i)^\top \mathbf{X}||_2^2. \tag{10}$$

We call this optimization *data-dependent* decomposition. While this decomposition well preserves the decision boundary, the above optimization is difficult to solve. An alternative solution is to minimize an L_2 norm of a residual vector between $\mathbf{w}$ and $\sum_{i=1}^{M} c_i \mathbf{m}_i$. The cost function is defined as follows.

$$E_2 = ||\mathbf{w} - \sum_{i=1}^{M} c_i \mathbf{m}_i||_2^2. \tag{11}$$

We call this optimization *data-independent* decomposition because the training dataset $\mathbf{X}$ does not appear in this cost function. This optimization is relatively easier than the case of the data-dependent decomposition.

It should be noted that data-independent decomposition can be regarded as a special case of data-dependent decomposition. The two cost functions can be rewritten as follows.

Algorithm 1. *Data-independent decomposition* that minimizes E_2

function *data_independent_decomposition*($\mathbf{w}, M$)
$\mathbf{r} = \mathbf{w}$
for $i = 1$ to M **do**
  initialize $\mathbf{m}_i$ by random three integer values $\{-1, 0, +1\}$
  repeat
    $c_i = \mathbf{m}_i^\top \mathbf{r} / \mathbf{m}_i^\top \mathbf{m}_i$
    $m_{ij} = \underset{\alpha \in \{-1,0,+1\}}{\arg\min} (r_j - c_i \alpha)^2 \ , \quad \text{for } j = 1, \cdots, D$
  until c_i and $\mathbf{m}_i$ have not been updated.
  $\mathbf{r} \leftarrow \mathbf{r} - c_i \mathbf{m}_i$
end for
return $\{\mathbf{m}_i\}_{i=1}^M$, $\{c_i\}_{i=1}^M$

$$E_1 = (\mathbf{w} - \tilde{\mathbf{w}})^\top \mathbf{A} (\mathbf{w} - \tilde{\mathbf{w}}) \tag{12}$$

$$E_2 = (\mathbf{w} - \tilde{\mathbf{w}})^\top (\mathbf{w} - \tilde{\mathbf{w}}), \tag{13}$$

where $\tilde{\mathbf{w}} = \sum_{i=1}^M c_i \mathbf{m}_i$ and $\mathbf{A} = \mathbf{X}\mathbf{X}^\top$. If $\mathbf{A}$ is proportional to an identity matrix, E_1 becomes equivalent to E_2. Even when $\mathbf{A}$ does not satisfy this assumption, E_1 tends to decrease by minimizing E_2 in practice. Based on this observation, we propose a two-step algorithm, in which minimization of E_1 is started from a good initial solution obtained by minimizing E_2.

Data-independent decomposition is shown in Algorithm 1. Our strategy is to sequentially reduce the residual error in Eq. (11). At the i-th iteration, c_i and $\mathbf{m}_i$ are determined to minimize $||\mathbf{r} - c_i \mathbf{m}_i||_2^2$, where $\mathbf{r} = (r_1, \cdots, r_D)^\top$ is a residual vector initialized by $\mathbf{w}$ before the first iteration. This optimization is done using an alternative approach. When $\mathbf{m}_i$ is fixed, c_i is updated using a least square method. When c_i is fixed, the j-th element m_{ij} in $\mathbf{m}_i$ is separately updated by testing only three candidates $\{-1, 0, +1\}$. After the convergence, the residual vector $\mathbf{r}$ is updated by subtracting $c_i \mathbf{m}_i$ before going to the next $(i+1)$-th iteration.

Data-dependent decomposition shown in Algorithm 2 uses Algorithm 1 to produce a good initial solution. At the i-th iteration, c_i and $\mathbf{m}_i$ are determined to minimize the following cost value.

$$\epsilon_i = ||(\mathbf{r} - c_i \mathbf{m}_i)^\top \mathbf{X}||_2^2 = (\mathbf{r} - c_i \mathbf{m}_i)^\top \mathbf{A} (\mathbf{r} - c_i \mathbf{m}_i). \tag{14}$$

If $\mathbf{m}_i$ is fixed, c_i is updated using a least square method as well as Algorithm 1.

$$c_i = \mathbf{m}_i^\top \mathbf{A}\mathbf{r} / \mathbf{m}_i^\top \mathbf{A}\mathbf{m}_i. \tag{15}$$

Even when c_i is fixed, optimizing $\mathbf{m}_i$ is still difficult due to the fact that the j-th element m_{ij} cannot be separately optimized to minimize ϵ_i unlike in the case of data-independent decomposition. To address this issue, our algorithm permits replacing only n_c randomly-chosen elements at the same time. In this

Algorithm 2. *Data-dependent decomposition* that minimizes E_1

function *data_dependent_decomposition*$(\mathbf{w}, M, \mathcal{T}_{\min}, \rho)$
$\mathbf{r} = \mathbf{w}$
for $i = 1$ to M **do**
  initialize c_i and $\mathbf{m}_i$ by calling *data_independent_decomposition*$(\mathbf{r}, 1)$
  $N_{\text{iter}} = 0; N_{\text{update}} = 0$
  loop
    $N_{\text{iter}} \leftarrow N_{\text{iter}} + 1$
    if $(N_{\text{iter}} > \mathcal{T}_{\min})$ and $(N_{\text{update}}/N_{\text{iter}} < \rho)$ **then**
      break loop
    end if
    create Ω by randomly choosing n_c indices from $\{1, \cdots, D\}$
    $\mathbf{m}_{i,\Omega} = \underset{\boldsymbol{\alpha} \in \{-1,0,+1\}^{n_c}}{\arg\min} \epsilon_i(\boldsymbol{\alpha})$ {*see Eq.(17)*}
    if cost value ϵ_i is not improved, **then**
      continue loop
    end if
    $N_{\text{update}} \leftarrow N_{\text{update}} + 1$
    $c_i = \mathbf{m}_i^\top \mathbf{A}\mathbf{r} / \mathbf{m}_i^\top \mathbf{A}\mathbf{m}_i$
  end loop
  $\mathbf{r} \leftarrow \mathbf{r} - c_i \mathbf{m}_i$
end for
return $\{\mathbf{m}_i\}_{i=1}^M, \{c_i\}_{i=1}^M$

case, only 3^{n_c} candidates are necessary to be tested to update $\mathbf{m}_i$. Although such greedy-like optimization is seemingly difficult to sufficiently minimize the cost function, our algorithm gives good results because this optimization starts from the good initial solution produced by Algorithm 1.

When c_i is fixed, $\mathbf{m}_i$ is updated as follows. Let us define a set Ω containing indices of n_c randomly-chosen elements. Its complement is $\bar{\Omega} = \{1, \cdots, D\} \setminus \Omega$. The cost value ϵ_i can be rewritten as follows.

$$\begin{aligned} \epsilon_i = &(\mathbf{r}_\Omega - c_i \mathbf{m}_{i,\Omega})^\top \mathbf{A}_{\Omega\Omega} (\mathbf{r}_\Omega - c_i \mathbf{m}_{i,\Omega}) + \\ &(\mathbf{r}_{\bar{\Omega}} - c_i \mathbf{m}_{i,\bar{\Omega}})^\top \mathbf{A}_{\bar{\Omega}\bar{\Omega}} (\mathbf{r}_{\bar{\Omega}} - c_i \mathbf{m}_{i,\bar{\Omega}}) + \\ &2(\mathbf{r}_\Omega - c_i \mathbf{m}_{i,\Omega})^\top \mathbf{A}_{\Omega\bar{\Omega}} (\mathbf{r}_{\bar{\Omega}} - c_i \mathbf{m}_{i,\bar{\Omega}}), \end{aligned} \tag{16}$$

where $\mathbf{m}_{i,\Omega}, \mathbf{m}_{i,\bar{\Omega}}, \mathbf{r}_\Omega$, and $\mathbf{r}_{\bar{\Omega}}$ are sub-vectors of $\mathbf{m}_i$ and $\mathbf{r}$ and $\mathbf{A}_{\Omega\Omega}$, $\mathbf{A}_{\bar{\Omega}\bar{\Omega}}$, and $\mathbf{A}_{\Omega\bar{\Omega}}$ are sub-matrices of $\mathbf{A}$. These sub-vectors and sub-matrices contain certain rows and columns indicated by their subscripts Ω and $\bar{\Omega}$. In the case of a matrix, the first subscript means row indices and the second subscript means column indices, e.g. $\mathbf{A}_{\Omega\bar{\Omega}}$ is an n_c-by-$(D - n_c)$ matrix. In Algorithm 2, not only c_i but also $\mathbf{m}_{i,\bar{\Omega}}$ are fixed. In this case, the cost value ϵ_i is simplified as follows.

$$\begin{aligned} \epsilon_i(\mathbf{m}_{i,\Omega}) = &c_i^2 \mathbf{m}_{i,\Omega}^\top \mathbf{A}_{\Omega\Omega} \mathbf{m}_{i,\Omega} \\ &+2c_i \mathbf{m}_{i,\Omega}^\top (c_i \mathbf{A}_{\Omega\bar{\Omega}} \mathbf{m}_{i,\bar{\Omega}} - \mathbf{A}_{\Omega\Omega} \mathbf{r}_\Omega - \mathbf{A}_{\Omega\bar{\Omega}} \mathbf{r}_{\bar{\Omega}}) + \text{constant}. \end{aligned} \tag{17}$$

Algorithm 3. Fast classification by rejection cascade

function *classify* $(\mathbf{x}, b, \mathbf{m}_1 \cdots, \mathbf{m}_M, R_1, \cdots, R_M, M)$
$y = b$; $f = positive$
for $i = 1$ to M **do**
  $y \leftarrow y + c_i \mathbf{m}_i^\top \mathbf{x}$
  if $y < R_i$ **then**
    $f = negative$; exit for loop
  end if
end for
return f

This sub-problem can be easily solved by testing only 3^{n_c} candidates $\{-1, 0, +1\}^{n_c}$ because $\mathbf{m}_{i,\Omega}$ is an n_c-dimensional ternary vector. Depending on the choice of Ω, the initial value of $\mathbf{m}_{i,\Omega}$ is already optimal for $\epsilon_i(\mathbf{m}_{i,\Omega})$. Therefore, we repeat the random choice and the sub-problem optimization until the cost value is improved. If the cost value is not improved for a long time, we go to the next $(i+1)$-th iteration. This is controlled by two pre-defined thresholds $\mathcal{T}_{\min}$ and ρ. In practice, the parameters $\{n_c, \mathcal{T}_{\min}, \rho\}$ are set to $\{4, 100, 0.1\}$, respectively.

Obviously, with a slight modification, both Algorithms 1 and 2 are also available when $\mathbf{m}_i$ is constrained to binary values. In this case, Algorithm 1 produces the same results as a previous work[10]. From this perspective, the Algorithms 1 and 2 can be regarded as a generalization of their work.

2.3 Rejection Cascade

Since Algorithms 1 and 2 sequentially compute $\mathbf{m}_i$ by reducing the residual error one by one, $\mathbf{m}_i$ with the smaller index i more contributes to $\mathbf{w}$ in a similar way to a principal component analysis. In other words, the original decision boundary is almost reconstructed by the first basis vector $\mathbf{m}_1$ and its coefficient c_1. The rest of basis vectors $\mathbf{m}_2, \cdots, \mathbf{m}_M$ and coefficients $c_2, \cdots, c_M$ acts as correction terms for further fine reconstruction. This characteristic is very convenient to construct rejection cascade. In most cases, we can estimate the sign of the approximated classification score $f_{\text{approx}}(\mathbf{x})$ by using only the first several inner products $\mathbf{m}_i^\top \mathbf{x}$.

Algorithm 3 shows our strategy of the early termination. Our rejection cascade has M stages. A cumulative score y is initialized by the bias term b at the beginning. At i-th stage, $c_i \mathbf{m}_i^\top \mathbf{x}$ is added to y. The score y is thresholded by R_i to examine whether $\mathbf{x}$ belongs to a negative class or not.

The thresholds $R_1, \cdots, R_M$ must not cause a loss of accuracy. We discuss how the j-th threshold R_j should be determined. At the j-th stage, j weighted inner products $c_1 \mathbf{m}_1^\top \mathbf{x}, \cdots, c_j \mathbf{m}_j^\top \mathbf{x}$ have already been given. The rest of the weighted inner products are not given at this time. We split them as follows.

$$f_{\text{approx}}(\mathbf{x}) = \sum_{i=1}^{j} c_i \mathbf{m}_i^\top \mathbf{x} + b + \sum_{i=j+1}^{M} c_i \mathbf{m}_i^\top \mathbf{x}. \tag{18}$$

While the first and second terms of Eq. (18) are given at the j-th stage, the third term is not given yet. The problem is how we estimate whether $f(\mathbf{x})$ is negative without knowing the third term.

To address this issue, we consider replacing the unknown third term with a possible maximum value α_i defined as follows:

$$\alpha_i = \max_{\mathbf{x} \in \{\mathbf{x}_1, \cdots, \mathbf{x}_N\}} (c_i \mathbf{m}_i^\top \mathbf{x}), \tag{19}$$

where $\mathbf{x}_1, \cdots, \mathbf{x}_N$ are training samples. By replacing $c_i \mathbf{m}_i^\top \mathbf{x}$ in the third term of Eq. (18) with α_i, we obtain the upper bound of $f_{\text{approx}}(\mathbf{x})$ as follows.

$$f_{\text{approx}}(\mathbf{x}) \leq \sum_{i=1}^{j} c_i \mathbf{m}_i^\top \mathbf{x} + b + \sum_{i=j+1}^{M} \alpha_i. \tag{20}$$

If the upper bound is less than zero, $f_{\text{approx}}(\mathbf{x})$ also takes a negative value. In this case, the feature vector $\mathbf{x}$ can be regarded as a negative sample. Therefore, the j-th threshold R_j can be determined as follows.

$$\sum_{i=1}^{j} c_i \mathbf{m}_i^\top \mathbf{x} + b + \sum_{i=j+1}^{M} \alpha_i < 0 \tag{21}$$

$$\sum_{i=1}^{j} c_i \mathbf{m}_i^\top \mathbf{x} + b < - \sum_{i=j+1}^{M} \alpha_i = R_j. \tag{22}$$

In addition, obviously $R_M = 0$.

3 Experiments and Discussion

We evaluated SPADE by taking pedestrian detection task as an example. We used INRIA pedestrian dataset[3] and a software supplied by [4] to evaluate miss rate against false positives per image (FPPI).

To avoid confusion, four different decompositions are abbreviated as follows.

- Data-Independent, binary basis vectors (DI2, DI2M1, $\cdots$,DI2M5)
- Data-Independent, ternary basis vectors (DI3, DI3M1, $\cdots$,DI3M5)
- Data-Dependent, binary basis vectors (DD2, DD2M1, $\cdots$,DD2M5)
- Data-Dependent, ternary basis vectors (DD3, DD3M1, $\cdots$,DD3M5)

The first 'DI' or 'DD' specifies the decomposition algorithm. The following digit, '2' or '3', defines the constraint on basis vectors. Optionally, the number of basis vectors M may be added to the end of the abbreviation. For example, 'DD3M5' uses data-dependent decomposition algorithm, ternary basis vectors and $M = 5$. It should be noted that DI2 is completely equivalent to the previous work[10], when the rejection cascade is not used.

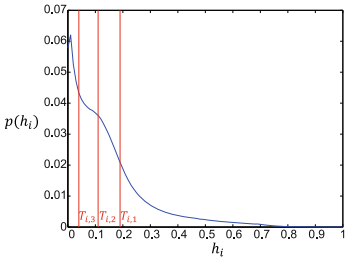


Fig. 4. probability distribution of h_i

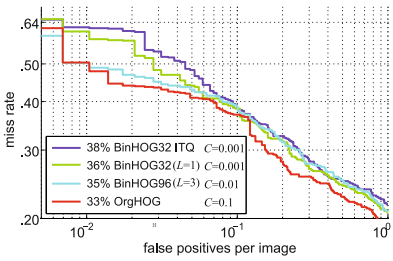


Fig. 5. Performance comparison

3.1 Implementation Details: Pedestrian Detector Using Binary HOG

While real-valued HOG feature is frequently used in object detection task[3,7,6], SPADE requires binary features. We show an interesting observation that the HOG can be converted to binary features by simple multi-level thresholding with little loss of detection accuracy. In our implementation[2], eight contrast-insensitive gradient orientations are used to form a 32-dimensional HOG feature $\mathbf{h} = (h_1, \cdots, h_{32})^\top$ per block. L-level thresholding is applied to h_i as follows.

$$h_{i,l} = \begin{cases} +1 & h_i \geq T_{i,l} \\ -1 & h_i < T_{i,l} \end{cases}. \tag{23}$$

The thresholds $T_{i,1}, \cdots, T_{i,L}$ for h_i are determined as follows. Each element h_i extracted from training samples are sorted in descending order, where $T_{i,l}$ is chosen to make the top $100l/(L+1)\%$ values take $+1$ and the rest take -1 in the sorted list. For example, when $L = 3$, three thresholds are chosen from values at the top 25, 50, and 75% positions in the sorted list. Figure 4 shows an example of a probability density distribution of h_i and the chosen three thresholds $T_{i,1}$, $T_{i,2}$, and $T_{i,3}$. In this way, a $32L$-dimensional binary feature per block is obtained.

Figure 5 compares the miss rates of the real-valued and the binarized HOG. Two parameters, $L = 1$ and 3, are tested. In addition, iterative quantization (ITQ) [8] is also tested to generate a 32-bit binary feature from $\mathbf{h}$. At the beginning of the captions, log-average miss rate[4] is denoted. The pedestrian models are trained using a linear SVM with appropriate soft margin parameters C, which are shown at the end of the captions. Interestingly, the binarized HOG produced comparable results to the real-valued HOG when L was set to 3. Even when $L = 1$, the log-average miss rate dropped only 3%. It should be also noted that our 32-bit HOG exhibited almost the same accuracy as that of ITQ. Since extracting binary features must also be fast, we used the simple multi-level thresholding instead of using the conventional binary hashing. In the following subsections, we use three abbreviations: 'BinHOG32', 'BinHOG96', and 'OrgHOG' that correspond to 32-bit($L = 1$), 96-bit($L = 3$) and 32-dimensional real-valued HOG, respectively. Unless otherwise specified, the BinHOG32 and rejection cascade were used in the following experiments.

[2] Unless otherwise stated, we followed the same parameters described in [3].

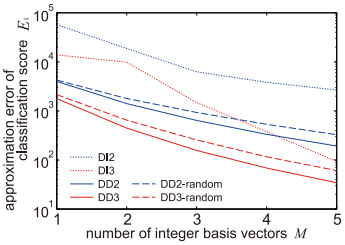


Fig. 6. Comparison of decompositions

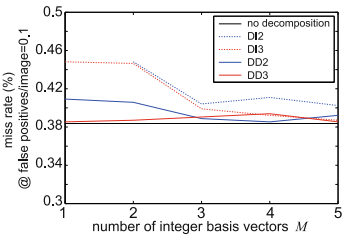


Fig. 7. Miss rate v.s. M

3.2 Comparing Approximation Qualities of Classification Scores

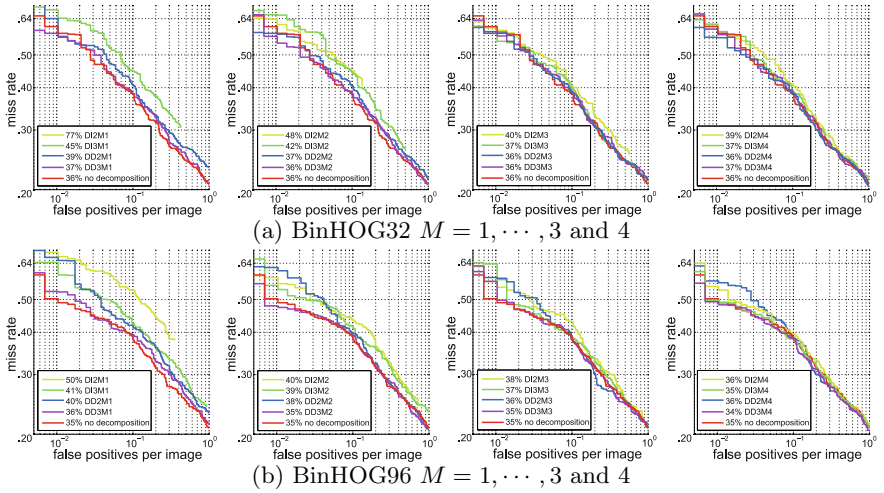
Figure 6 shows total approximation errors of classification scores E_1 defined in Eq.(10) versus M. The y-axis is represented in a logarithmic scale. In addition to the above mentioned four methods: DI2, DI3, DD2, and DD3, the data-dependent decomposition algorithm initialized by random values (instead of using the optimized initial solution generated by the Algorithm 1) is also tested. This is denoted by 'DD2-random' and 'DD3-random'.

Although the approximation errors E_1 were decreased by increasing M in all cases, the approximation qualities were quite different for each case. The cases of the ternary representation: DI3, DD3, and DD3-random, produced better results than the cases of the binary representation: DI2, DD2, and DD2-random. The binary representation required the larger number of basis vectors M than the ternary representation. Interestingly, DI2 and DI3 decreased E_1 in spite of the fact that they do not directly minimize E_1 but E_2. This fact suggests that DI2 and DI3 can produce a good initial solution. In fact, DD2-random and DD3-random produced poor results compared to DD2 and DD3. The best approximation quality was obtained by using DD3.

3.3 Classification Accuracy and M

Figure 7 shows miss rates at 10^{-1} false positives per image versus M. We also tested a baseline method that does not decompose the weight vector $\mathbf{w}$ into integer basis vectors. This baseline detector runs very slow, but its classification scores do not include any errors. This is denoted by 'no decomposition' in this figure. In this experiment, consistent results with section 3.2 were observed. In all the four cases, the miss rates were approached to the same level of the baseline detector by increasing M. However convergence speed were obviously different among them. Even when M was set to a small value, DD3 achieved almost the same miss rate as the baseline detector.

Figure 8 shows performance curves obtained by testing different M, different binary features and different decomposition algorithms. The results of BinHOG32 and BinHOG96 are separately shown in (a) and (b). In addition to the four methods: DI2, DI3, DD2, and DD3, the baseline detector (without decomposing $\mathbf{w}$) is also tested as well as Figure 7. By increasing M, the results of all the



(a) BinHOG32 $M = 1, \cdots, 3$ and 4

(b) BinHOG96 $M = 1, \cdots, 3$ and 4

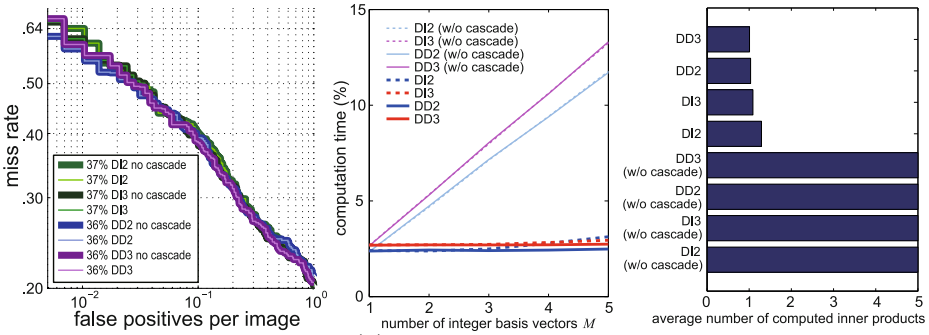
Fig. 8. In these figures, two kinds of binary features (BinHOG32 and BinHOG96), five approximation models(DI2, DI3, DD2, DD3, and no decomposition) and different $M = 1, \cdots, 4$ were tested to draw performance curves.

four methods (DI2, DI3, DD2, and DD3) approached to the performance curve of the baseline detector (without decomposition). The same trend was observed for BinHOG32 and BinHOG96. In both cases, DD3 reached at almost the same level of the baseline detector even when $M = 1$.

In summary, a conclusion resulting from Figure 6, 7, and 8 is that we should minimize the data-dependent cost function E_1 rather than the data-independent cost function E_2 addressed in the previous work[10]. Introducing the ternary decomposition helps to sufficiently minimize E_1, which brings significant performance gain over the (previously proposed) binary decomposition.

3.4 Effect of Rejection Cascade and Computation Time

Figure 9 summarizes the effect of our rejection cascade. The cases in which early termination is not used are indicated by 'w/o cascade'. Figure 9(a) compares miss rates. Figure 9(b) shows computation time ratios compared to the case in which $\mathbf{w}$ is not decomposed. Figure 9(c) shows frequency of early termination. x-axis means an average number of evaluated inner products in the rejection cascade. If the early termination works well, this value approaches to one. Throughout this experiment, a *popcnt* instruction of Intel Core i7 processor was used. It was observed from Figure 9(a) that the rejection cascade did not cause loss of classification accuracy. The detection results with and without the rejection cascade were completely consistent. Figure 9(b)(c) revealed that the rejection cascade obviously improved the runtime computation even when M was set to a large value. As shown in Figure 9(c), DD3 could reject the largest number of



(a) Miss rate (b) Computation time (c) Early termination

Fig. 9. DI2, DI3, DD2, and DD3 with and without rejection cascade were compared

Table 1. Summary of speed and miss rate

method	acceleration	miss rate@FPPI=0.025	miss rate@FPPI=0.1
OrgHOG (baseline)	1×	0.436	0.370
BinHOG32 DD3M5	36.9×	0.503	0.385
BinHOG96 DD3M5	12.9×	0.494	0.385
BinHOG32 DI2M5 w/o cascade (Hare *et.al.*[10])	8.6×	0.504	0.402

candidates than DI2, DI3 and DD2. These results suggest that the first ternary basis vector $\mathbf{m}_1$ obtained by DD3 sufficiently preserved the decision boundary.

Finally, we compared our method with the previous work proposed by Hare *et.al.*[10] as shown in Table 1. As pointed out in Section 2.2, Hare's study is regarded as 'DI2 w/o cascade'. Our method outperformed both computation time and accuracy compared to the previous work.

4 Conclusion

In this paper, we proposed SPADE that accelerated runtime computation of the linear classifier by introducing three ideas: ternary representation, data-dependent decomposition and rejection cascade. Since state-of-the-art object detection[2] is beginning to focus on the Hare's study[10] to boost the detection task, it is widely expected that SPADE is able to improve such detection methods based on binary features. Finally, we discuss possible extensions for future works. While we focused on a single class case throughout this paper, we believe that it is potentially feasible to extend our method to a multi-class recognition. In this case, ternary basis vectors should be shared within multiple real-valued weight vectors in a similar spirit of related works[20,19,17], because visual features are often shared among different classes. From the other perspective, it is also expected that SPADE is widely available not only for computer vision but also for the other field of researches based on linear classifiers and binary features.

References

1. Ambai, M., Yoshida, Y.: CARD: Compact and real-time descriptors. In: ICCV, pp. 97–104 (2011)
2. Cheng, M.-M., Zhang, Z., Lin, W.-Y., Torr, P.: BING: Binarized normed gradients for objectness estimation at 300fps. In: CVPR (2014)
3. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR, pp. 886–893 (2005)
4. Dollár, P., Wojek, C., Schiele, B., Perona, P.: Pedestrian detection: An evaluation of the state of the art. In: PAMI, pp. 743–761 (2012)
5. Dubout, C., Fleuret, F.: Exact acceleration of linear object detectors. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part III. LNCS, vol. 7574, pp. 301–311. Springer, Heidelberg (2012)
6. Felzenszwalb, P.F., Girshick, R.B., McAllester, D.: Cascade object detection with deformable part models. In: CVPR, pp. 2241–2248 (2010)
7. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part based models. In: PAMI, pp. 1627–1645 (2010)
8. Gong, S.Y., Lazebnik: Iterative quantization: A procrustean approach to learning binary codes. In: CVPR, pp. 817–824 (2011)
9. Grauman, K., Darrell, T.: The pyramid match kernel: Discriminative classification with sets of image features. In: ICCV, pp. 1458–1465 (2005)
10. Hare, S., Saffari, A., Torr, P.H.S.: Efficient online structured output learning for keypoint-based object tracking. In: CVPR, pp. 1894–1901 (2012)
11. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: NIPS, pp. 1106–1114 (2012)
12. Lampert, C.H., Blaschko, M.B., Hofmann, T.: Beyond sliding windows: Object localization by efficient subwindow search. In: CVPR, pp. 1–8 (2008)
13. Leutenegger, S., Chli, M., Siegwart, R.Y.: BRISK: Binary robust invariant scalable keypoints. In: ICCV, pp. 2548–2555 (2011)
14. Maji, S., Berg, A.C., Malik, J.: Classification using intersection kernel support vector machines is efficient. In: CVPR, pp. 1–8 (2008)
15. F., Perronnin, Y.L., Sanchez, J., Poirier, H.: Large-scale image retrieval with compressed fisher vectors. In: CVPR, pp. 3384–3391 (2010)
16. Perronnin, F., Sánchez, J., Mensink, T.: Improving the fisher kernel for large-scale image classification. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part IV. LNCS, vol. 6314, pp. 143–156. Springer, Heidelberg (2010)
17. Rigamonti, R., Sironi, A., Lepetit, V., Fua, P.: Learning separable filters. In: CVPR, pp. 2754–2761 (2013)
18. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: ORB: An efficient alternative to SIFT or SURF. In: ICCV, pp. 2564–2571 (2011)
19. Song, H.O., Girshick, R., Darrell, T.: Discriminatively activated sparselets. In: ECCV, pp. 196–204 (2013)
20. Song, H.O., Zickler, S., Althoff, T., Girshick, R., Fritz, M., Geyer, C., Felzenszwalb, P., Darrell, T.: Sparselet models for efficient multiclass object detection. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part II. LNCS, vol. 7573, pp. 802–815. Springer, Heidelberg (2012)
21. Vedaldi, A., Zisserman, A.: Efficient additive kernels via explicit feature maps. In: PAMI, pp. 480–492 (2012)
22. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: CVPR, pp. 511–518 (2001)
23. Weiss, Y., Torralba, A., Fergus, R.: Spectral hashing. In: NIPS, pp. 1753–1760 (2008)