Efficient Model Evaluation with Bilinear Separation Model

Fanyi Xiao Martial Hebert
The Robotics Institute, Carnegie Mellon University

Abstract

In this paper, we investigate the issue of evaluating efficiently a large set of models on an input image in detection and classification tasks. We show that by formulating the visual task as a large matrix multiplication problem, something that is possible for a broad set of modern detectors and classifiers, we are able to dramatically reduce the rate of growth of computation as the number of models increases. The approach, based on a bilinear separation model, combines standard matrix factorization with a task-dependent term which ensures that the resulting smaller size problem maintains performance on the original task. Experiments show that we are able to maintain, or even exceed, the level of performance compared to the default approach of using all the models directly, in both detection and classification tasks. This approach is complementary to other efforts in the literature on speeding up computation through GPU implementation, fast matrix operations, or quantization, in that any of these optimizations can be incorporated.

1. Introduction

As techniques for *object detection* and *image classification* tasks become increasingly powerful, one can contemplate using them in applications in which a large number of models are used. However, as the number of models increases, so does the computation effort involved, to the point that the computational cost may become prohibitive. In this paper, we investigate a general approach to efficiently run a broad class of detection and classification algorithms by estimating their responses on an input image using a *bilinear separation model* which is much more computationally economical than evaluating every model. The starting point of our approach is the well-known observation that, conceptually, many common detection and classification algorithms can be summarized in a rather general form [28, 18, 22, 27, 26]:

$$R = W^T X, W \in \mathbb{R}^{D \times M}, X \in \mathbb{R}^{D \times N}, \quad (1)$$

where M is the number of models (e.g., HOG templates in object detection or weights for bag of words feature in image classification problems), N is the number of data points to evaluate, e.g., the number of windows in a detection task, and D is the dimension of the feature. The way the models W are learned and the features used in X, of course, depends on the specific approach but the fundamental operation applied to an input image reduces to (1). Our approach relies on approximating this operation in order to reduce the computation spent on this matrix-matrix multiplication, which is, in many cases, the most computational intensive operation in vision algorithms.

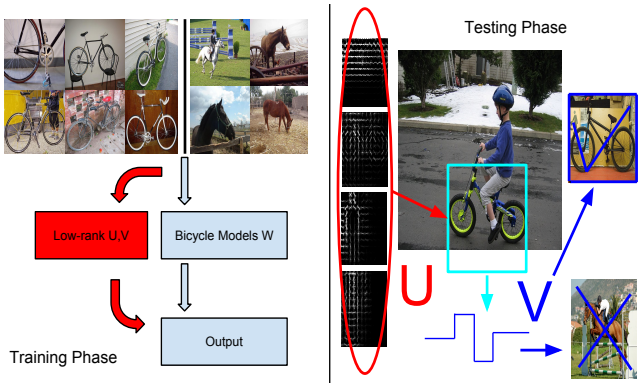

Figure 1: The illustration of the training and testing phase of the proposed bilinear separation model (BSM). The left picture illustrates the training phase of BSM: the low-rank matrices U and V are learned to mimic the behaviour of the original model W in terms of the prediction performance. Whereas the right picture illustrates the testing phase of BSM for efficient model evaluation: the HOG patch is first projected to the low-rank basis U (red) and then evaluated with V (blue), which contains the predictors for different models.

The general form of (1) applies to many standard approaches both in detection and classification. For example, the popular object detection paradigm, i.e., sliding window search with linear classifiers over HOG representation [3, 10, 18, 14], can be easily put in the form (1). In this setting, the matrix $W \in \mathbb{R}^{D \times M}$ consists of the stacked parameters $w_i, i = 1, 2, ..., M \in \mathbb{R}^D$ corresponding to each model, i.e., the linear classifier trained over the HOG feature. Similarly, each column of the matrix X is the HOG feature vector extracted from a specific sliding window in

the HOG pyramid of the image. Furthermore, as for the more complicated part-based models, the form (1) could also be applied to get the part responses (as well as the root response) followed by a distance transform to get the final scores. In the realm of image classification, the ImageNet dataset and the associated challenge has become a benchmark for large-scale image classification problem [5]. Among various state-of-the-art approaches on this challenge, many different features and encoding methods are employed [17, 15]. However, most share the same way of applying the classifiers to an input image, that is to evaluate all the models, represented by a matrix W in which the vectors of weights of all the models are stacked, on all the images X: $R = W^T X$ as in (1).

As implied by (1), the computational complexity of evaluating M models on N data points with D dimensional feature is $O(MDN)$, which would easily become prohibitive as all these three parameters scale up quickly [18, 25]. In order to apply (1) in a more efficient way, a natural approach would be to use a low-rank approximation for the parameter matrix $W = UV^T, U \in \mathbb{R}^{D \times L}, V \in \mathbb{R}^{M \times L}$, thus transforming the problem into the form $R = W^T X = VU^T X$, which is much more efficient if L, the constrained rank, is small enough compared with D. An intuitive way of accomplishing this low-rank approximation is to apply an SVD-based matrix decomposition to W, which implicitly minimizes the reconstruction error $||W - UV^T||^2$ subject to the rank constraint. However, this "obvious" approach is not appropriate. First, the objective minimized by this approach is not directly related to the *prediction performance* of model evaluation. Second, this factorization does not take into account any data for the specific task. We propose in this paper a novel approach that addresses these two problems by using a *bilinear separation model*, which has been successfully applied in many different computer vision tasks [28, 22, 12, 31]. We propose to learn a bilinear model by optimizing the criteria that is directly related to the prediction performance for specific tasks. As we will show in Section 2, the proposed factorization can be computed with a coordinate-descend method based on its bi-convexity property [13, 22].

Many previous works have been published to address the computation issues in vision tasks. The work most related to ours are from Ramanan et al. [21] and Song et al. [27, 26], which are both based on the idea of decomposing classifiers to efficiently evaluate (1). However, there are significant differences with our work. In [21], they decompose the weight parameters for each single classifier which corresponds to one column of W in equation (1). Whereas our idea is to find out the factors shared among all classifiers. [27, 26] use sparse coding to learn an *overcomplete* basis whereas we directly learn a low-rank compact basis which preserves the predicting capability. Also, previous works *directly* use their approximation as the detection responses, instead we show that we can use a *re-evaluation* step to recover the detection performance, which relaxes the need for the approximation accuracy while maintaining the high performance on the specific tasks. Furthermore, our work differs from the prior works in that it has the capability of exploiting pre-trained models: instead of training from scratch, our method is able to treat pre-trained models as black boxes and mimic their behaviour while being much more efficient. Finally, we show that our method is able to achieve performance almost identical to directly evaluating (1) which is hard for both [21] and [27, 26].

Other works attempt to reduce either the number of windows evaluated, or the number of models evaluated. The cascaded methods are applied to select testing windows that are with high probability containing objects through cheaper operations [29, 9, 20]. More recent approaches propose that, instead of modelling the probability of containing *certain* objects (meaning with supervision), they directly model the *interestingness* of image windows without any supervision [19, 1, 7]. A complementary line of research aims at reducing the number of models M. For example, Pirsiavash et al. propose to use the shared steerable filters as the basis for constructing a large number of part templates [21]. Also aiming at reducing the number of models, Dean et al. take a different approach using locality-sensitive hashing for model selection at every window [4] in order to scale to 100000 categories[1]. In addition to these two complementary lines of research, [24] accelerates the model evaluation directly by vector quantization. Instead of only selecting models or windows, we propose to estimate the responses of every model on every window by a learned bilinear separation model, then re-evaluate a small fraction of the top-ranked *"model-window"* pairs, according to the estimated response values, to recover the detection results as if we have evaluated all the *"model-window"* pairs. Note that even though we also select *"model-window"* pairs for re-evaluation as [24] did, we take a very different approach that follows the model evaluation procedure of many publicly available object detector packages so that it is fairly easy to integrate our approach.

Other approaches attempt to reduce computation by optimizing the computations involved in (1) directly, rather than reducing the complexity of the model representation. They include GPU implementations [2, 27], fast matrix operations [6], or quantization [24]. It is important to note that our approach is complementary to these efforts in that any of these optimization can be incorporated in our approach to achieve further speed-up.

Our contributions are threefold: (1) We present a method

[1] Girshick et al. [11] demonstrated that large matrix multiplication, which is what we study in this paper, is much more efficient than hashing technique.

that treats the pre-trained models as black boxes and mimic their behaviours which enables us to exploit the large amount of pre-trained models available nowadays. (2) We present both theoretical and empirical results on the computation to show that our method scales gracefully as the number of models increases. (3) Our method is able to achieve almost identical performance to the exhaustive model evaluation as we demonstrated on both object detection and image classification tasks.

2. Learning a Bilinear Model for Efficient Model Evaluation

In this section, our goal is to learn a low-rank approximation $W \approx UV^T$ such that $\tilde{R} = VU^T X$ gives us a *good* estimation of $R = W^T X$ in terms of the *task-specific prediction performance*. The general idea of the training and testing phase of efficient model evaluation through low-rank approximation is shown in Figure 1.

2.1. SVD on Model Parameters W

Before describing our approach, let us first look at an intuitive way of accomplishing the low-rank approximation and point out its shortcomings to motivate our approach.

The correlation among different models has been exploited by many approaches [27, 26]. The key observation is that since models $w_i, i = 1, 2, ..., M$ are, in most cases, highly related to each other, e.g., detectors of visually similar objects, it makes sense to model this correlation to produce more compact representations for these models, and to evaluate them more efficiently. One of the most intuitive approach to model the correlations is to apply *Singular Value Decomposition* (SVD) over the stacked model parameters, and obtain a low-rank approximation $W \approx UDV^T$, in which U serves as a low-rank projection basis whereas each row of V as the combination coefficients on this basis for each model. Thus, the model evaluation could be approximated as:

$$\tilde{R} = (VD)U^T X = \bar{V}U^T X, U \in \mathbb{R}^{D \times L}, \bar{V} \in \mathbb{R}^{M \times L}, \quad (2)$$

where $\bar{V} = VD$. This computation could therefore be done by first computing $X_p = U^T X$, which is a low-rank projection of the data, and then $\tilde{R} = \bar{V} X_p$.

There are two problems with this seemingly reasonable approach. First, the SVD-decomposition is implicitly optimizing the objective $||U\bar{V}^T - W||_2$ subject to the low-rank constraint. However, there is no guarantee on the performance of the *specific tasks* by optimizing the above objective. For example, it is possible to have a pair of U_1 and $\bar{V}_1$ which has lower value according to the above objective function, but has poorer *prediction performance* on the desired task than another pair U_2 and $\bar{V}_2$. Second, the form $||W - U\bar{V}^T||^2$ does not take into account any data, whereas in the form (1), W^T is applied to the data matrix X. Directly optimizing $||W - U\bar{V}^T||^2$ is equivalent to making the assumption that each dimension of our data (visual features extracted from images) has the same magnitude in expectation, which is clearly not the case. To overcome these two problems, we propose to learn a low-rank *Bilinear Separation Model* (BSM) which we can still apply as shown in (2), by optimizing an objective that is *data-dependent* and directly related to the *prediction performance*. It is called a *bilinear* form because it is linear with regard to each of the two variables over which we aim to optimize given another is fixed.

2.2. Bilinear Separation Model

To justify the objective we propose to optimize, we take the object detection task as an example for illustrative purpose. The similar reasoning holds for the image classification task. We first introduce some notations to help us formalize the derivation of our model: X_i and X^j denote the i^{th} column and j^{th} row of matrix X, respectively. Consequently, X_i^j refers to the element on the i^{th} column and j^{th} row of X.

The evaluation of (1) provides scores for objects and their locations. Usually a *threshold* is then applied on this response matrix R to obtain the candidate detections, i.e., the "model-window" pairs. Therefore, what we really care about for an estimate of R, or the *quality* of an estimate of R, is that for those "model-window" pairs that are above the threshold $\{R_{ij} | (i,j) \in \mathcal{S}_c(t)\}$[2], they remain above the threshold in the approximation $\tilde{R}$ whereas for those pairs below the threshold, it is desirable they remain below the threshold as well. More formally, we want an approximation $\tilde{R}$ such that the following conditions are satisfied:

$$\begin{aligned} &\tilde{R}_{ij} \geq t, \forall (i,j) \in \mathcal{S}_c(t) \\ &\tilde{R}_{ij} < t, \forall (i,j) \notin \mathcal{S}_c(t). \end{aligned} \quad (3)$$

In order to enforce the *separation* constraint in (3), one would like the estimation $\tilde{R}$ to be penalized if it estimates a point to be above the threshold that is in fact below the threshold and vice versa. We show that this could be achieved using the following loss function:

$$L_1(\tilde{R}_{ij}, y_{ij}) = \max(0, -y_{ij} \cdot \tilde{R}_{ij}), \quad (4)$$

where $y_{ij} \in \{-1, 1\}$ is a variable indicating whether the corresponding value in the ground-truth response matrix R_{ij} is above the threshold:

$$y_{ij} = \mathbb{1}\{R_{ij} \geq t\}, \quad (5)$$

where the function $\mathbb{1}\{\cdot\}$ returns 1 while the statement is true, -1 otherwise. Although it represents the constraint

[2] $\mathcal{S}_c(t)$ is used to denote the set of "model-window" pairs (model index,window index) that are above the threshold t.

shown in (3), the loss function in (4) is a non-smooth function. To make use of many solvers designed for smooth functions, we choose to optimize the squared version of (4):

$$L_2(\tilde{R}_{ij}, y_{ij}) = \max(0, -y_{ij} \cdot \tilde{R}_{ij})^2. \quad (6)$$

Note that this squaring operation is similar to the L_2-loss for SVM [16]. Having the loss function representing the constraint (3), which is directly related to the prediction performance, we can then apply this loss function (6) to the low-rank approximation $\tilde{R} = VU^T X$, which leads to the following objective function[3]:

$$J = \sum_{i=1}^{M} \sum_{j=1}^{N_i} L_2(V^i U^T X_j^{(i)}, y_j^{(i)}) + \gamma ||W - UV^T||^2, \quad U \in \mathbb{R}^{D \times L}, V \in \mathbb{R}^{M \times L}. \quad (7)$$

Recall that M is the number of models, $X^{(i)} \in \mathbb{R}^{D \times N_i}$ is the data matrix for model i, N_i is the number of data points for the i^{th} model and $X_j^{(i)}$ is the j^{th} column of $X^{(i)}$. Similarly, $y^{(i)}$ is a vector of variables for model i, $y_j^{(i)}$ is the indicating variable for the j^{th} data point $X_j^{(i)}$. V^i is the i^{th} row of matrix V.

The first term in (7) enforces the squared separation loss function over the low-rank approximation $\tilde{R} = VU^T X$. Thus, the learned bilinear model U and V are well-tuned to predict whether or not a "model-window" pair would be a candidate detection. Also, the data $X_j^{(i)}$ in the first term makes it a data-dependent term compared with direct SVD over W. The *data-dependency* enforces adaptation to the distribution of the training data. The second term $\gamma ||W - UV^T||^2$ is exactly the objective of the SVD approach in Section 2.1 multiplied by a scalar weight γ, which serves to regularize the optimization.

Note that our objective function is closely related to other forms of matrix factorization in machine learning, e.g., *max-margin matrix factorization* [23]. However, there are also important differences between those approaches and ours. Recall that M and D are the number of models and the dimension of features. First, suppose that there is no rank constraint on U and V, which means that the sizes of U and V are $[D \times \min(M, D)]$ and $[M \times \min(M, D)]$, respectively. Thus, the optimum of the objective (7) is exactly the original model parameter W, since both the first and the second term of (7) would be zero. That is, our formulation will get closer and closer to the original parameter matrix W as the rank constraint is relaxed. On the contrary, the *max-margin matrix factorization* enforces a large-margin loss function, i.e., an SVM-like hinge loss $\max(0, 1 - y \cdot x)$, which prevents it from recovering the original model parameters W even when we relax the rank constraint.

Although we used the object detection task to explain our approach, the proposed optimality objective also applies to the image classification problem: instead of being image patches as in detection, X in (1) becomes feature from the global image in the image classification task.

2.3. Optimization

The objective shown in (7) is quadratic and therefore convex, with respect to the optimizing variable V when U is fixed, and vice versa. This property is called bi-convexity and has been extensively studied [13]. Because of the bi-convexity, (7) could be optimized using a *coordinate descend* method alternating between optimizing V and U.

Initialization: Since the problem is not jointly convex in U and V, a careful initialization is required. With a given rank L, we choose to initialize the low-rank projection basis U and the coefficients V with the SVD over W, which is equivalent to the second term in the objective (7).

Updating V: When U is fixed, the updating of V is straightforward. The matrix V can be updated by sequentially updating each row of V, which correspond to the coefficients on the basis U for different models. The subproblem of optimizing each row of V can be formulated as the following convex program and thus be solved using the Quasi-Newton method.

$$\min_{V^i} \sum_{j=1}^{N_i} L_2(V^i U^T X_j^{(i)}, y_j^{(i)}) + \gamma ||W_i - UV^{iT}||^2. \quad (8)$$

Updating U: The updating step of the low-rank projection basis U is not as obvious as updating V. However, it is still a convex problem with respect to U when V is fixed. The optimization over U can be done by updating one column U_k once. Let us first look at the first term of (7), which is the loss function over the approximation. For a data point $X_j^{(i)}$, the loss function over it, as shown in (7), is in the form $L_2(V^i U^T X_j^{(i)}, y_j^{(i)})$, which could be re-written as $L_2(tr(V^i U^T X_j^{(i)}), y_j^{(i)}) = L_2(tr(U^T X_j^{(i)} V^i), y_j^{(i)})$. By fixing all columns of U except for the k^{th} one, the form could be further re-written as: $L_2(tr(U^T X_j^{(i)} V^i), y_j^{(i)}) = L_2(U_k^T X_j^{(i)} V_k^i + b_{kij}, y_j^{(i)})$. Where $b_{kij} = \sum_{l \neq k} U_l^T X_j^{(i)} V_l^i$, which is a constant when all columns of U are fixed except for the k^{th} one. The gradient $\frac{\partial L_2}{\partial U_k}$ can be easily computed as $2(U_k^T X_j^{(i)} V_k^i + b_{kij}) \cdot X_j^{(i)} V_k^i$ when $-y_j^{(i)}(U_k^T X_j^{(i)} V_k^i + b_{kij}) > 0$, whereas being 0 otherwise. Similarly, the regularization term in (7) could also be decomposed as $||W - (U_k V_k^T + \sum_{l \neq k} U_l V_l^T)||^2 = ||W^{(k)} - U_k V_k^T||^2$, where $W^{(k)} = W - \sum_{l \neq k} U_l V_l^T$. Therefore, we get the following subproblem, which we also optimize with the Quasi-Newton

[3] In practice, both the original model w_i and our learned bilinear model incorporate a bias term b_i, here for the simplicity of the formulation, we assume that data is pre-processed to zero-mean, which does not affect any conclusion made here.

method:

$$\min_{U_k} \sum_{i=1}^{M} \sum_{j=1}^{N_i} L_2(U_k^T X_j^{(i)} V_k^i + b_{kij}, y_j^{(i)}) + \gamma ||W^{(k)} - U_k V_k^T||^2, \tag{9}$$

Note that all the computations described in this section are performed once and for all during training BSM model.

2.4. Computational Analysis

We analyze the computation cost of applying (2) compared with directly applying the original form (1). The computational complexity of evaluating (1) is $O(MDN)$. However, for (2), the low-rank projection $X_p = U^T X$ is first computed, which has complexity $O(LDN)$, where L is the constrained rank. Then, the coefficient matrix V is multiplied with X_p which has complexity of $O(MLN)$. Theoretically, the ratio of computation complexity of evaluating (2) and (1) is $\frac{LDN+MLN}{MDN} = \frac{L(D+M)}{MD}$. As M, which is the number of models, becomes very large, the above term becomes L/D. This means that, as we have more and more models, the maximum speed-up we can achieve is D/L, which is the ratio between the original feature dimension and the low-rank projected feature dimension. As we will show in Section 3, this value could be very large without sacrificing prediction performance, in both object detection and image classification. One thing to mention is that feature computation, together with model evaluation, constitute the actual computation time in our evaluation, therefore there is discrepancy between the empirical speed-up in experiments and the theoretical analysis. We also note that all the sophisticated acceleration from the hardware side (e.g., GPU, MKL) could also be used combining with our approach, which could further accelerates the operation.

2.5. BSM for Object Detection

In order to apply our method to any object detection algorithm, we replace the model evaluation in (1) with the low-rank form (2) to get $\tilde{R}$, an estimate of R, which contains the actual responses of all models on all windows. Then, a small fraction of the "model-window" pairs (i, j) with the highest estimated responses in $\tilde{R}$ are re-evaluated using the actual model w_i and feature X_j (high-dimensional HOG template which is typically in the order of thousands of dimensions). Ideally, we would like all the elements in R that are above the threshold to remain above the threshold in $\tilde{R}$. After re-evaluating the "model-window" pairs, standard post-processing steps like *Non-maximum Suppression* (NMS) are applied to generate the final detections. In terms of the parameters, we then have 3 important parameters in this pipeline: 1) the constrained rank L in (2), and 2) the ratio of the re-evaluated "model-window" pairs $0 < p < 1$, as well as 3) the BSM specific parameter γ which is the weight for the regularizer in (7).

2.6. BSM for Image Classification

For image classification, each column of X is the feature vector of *one* image, whereas each column of W is the classifier for one image category. And UV^T is the low-rank approximation of matrix W. For testing, we simply replace the form (1) with (2) while keeping all the other components of the image classification system unchanged. The parameters in the image classification setting are: 1) L, the constrained rank, and 2) γ, the regularization weight. Note that p is not involved in the image classification task since we do not apply any re-evaluation in this setting.

3. Experiments

3.1. Object Detection

Setup: For the object detection experiments, we compare the detection performance using the learned BSM to the performance achieved by directly applying (1), as well as the performance of the baseline method which also uses the approximation form (2). For quantitative measure of the performance, we use the standard *mean average precision* as well as the *retrieval rate*, which is defined as $N_{retrieved}/N_{total}$. $N_{retrieved}$ is the number of the *candidate*[4] "model-window" pairs occurring in the top p of the estimated responses in matrix $\tilde{R}$, whereas N_{total} is the total number of the candidate "model-window" pairs in the full matrix R. Ideally we want the retrieval rate to be 1 so that the BSM-approximated model $\tilde{R}$ mimics the full matrix R. This is an important performance indicator for BSM since the closer to 1 the retrieval rate is, the closer the performance of the BSM-approximated detection system is to the performance of the full system.

We use the PASCAL VOC 2007 as the dataset for object detection experiments [8]. As discussed in Section 2.4, a large model collection (i.e., large M) is necessary to demonstrate the benefits of our approach. Therefore we choose to use Malisiewicz et al.'s Exemplar SVM (ESVM)[18] method since it has a large number of pretrained models immediately available, which enables to use a published baseline rather than retraining our own. Specifically, there are 12608 ESVM models for exemplars across 20 categories. The ESVM models are trained using a *single* positive data point with a large number of negative data points (both are HOG feature vectors) to serve as the representation of a particular exemplar. We use 12×12 templates throughout our experiments so that D, the dimension of the feature, is $12\times12\times31 = 4464$ (the templates of size smaller than 12×12 are padded to be 12×12 by placing them to the upper-left corner of the enlarged templates). For a typical image in the PASCAL VOC 2007 testing set, the number of windows N, is within 15000-25000.

[4] By candidate "model-window" pairs, we refer to those pairs that have responses above the threshold.

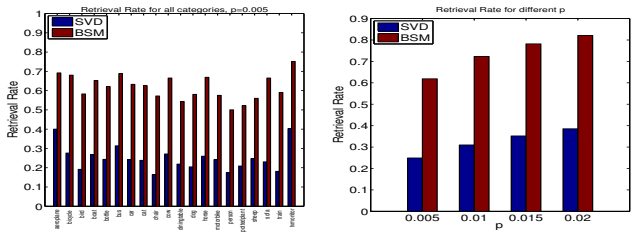

Figure 3: The left graph shows the retrieval rate of both SVD and BSM on all 20 categories of PASCAL VOC 2007 when p is fixed to 0.005. Whereas the right graph shows the retrieval rate of both methods with varying p, the results are averaged over 20 categories. It is clear that our method is much better than the baseline in terms of the retrieval rate measure over different categories and different p.

We note that even though we use ESVM to test our approach, we can use any detector that can be formulated as (1). In particular, any window scanning detector would fit our approach. We choose ESVM as a convenient way of using a large number of models to demonstrate the benefits of our approach. In particular, we do not advocate that ESVM is the best choice of detection algorithm on this dataset.

Data for Training the Bilinear Separation Model: In order to obtain data for optimizing the objective (7), we sample 500 positive images and 500 negative images for each category from the *trainval* set of PASCAL VOC 2007. With these 1000 images per category, the detectors are run to gather the windows with scores above the threshold. To gather the windows which have scores below the threshold, we directly sample 10 random windows per image to form a pool.

Efficient Model Evaluation Using Bilinear Separation Model: We follow the standard evaluation protocol of PASCAL VOC 2007 for ESVM, the SVD baseline (SVD) as well as the proposed bilinear separation model (BSM). For the SVD baseline and BSM, the exemplar models W are replaced by the rank-L matrices U and V. Then, after obtaining an approximation $\tilde{R}$ as shown in (2), a small fraction $0 < p < 1$ of the "model-window" pairs with the highest estimated responses are re-evaluated. Then, NMS and the co-occurrence matrix are applied to get the final detection results as in [18].

The detection performance of 6 out 20 categories on the PASCAL VOC 2007, measured by the *mean average precision*, is shown in Figure 2[5]. As we can see, BSM outperforms the baseline method by a large margin and has almost the same performance as the exhaustive model evaluation used in ESVM (measured by mean AP 22.3% vs 22.6%). Also, the simple SVD approach (Section 2.1) does not work well (mean AP 20.2% vs 22.6%) due to the shortcomings discussed in Section 2.1.

[5]For the full results on all 20 categories, please see our supplementary material.

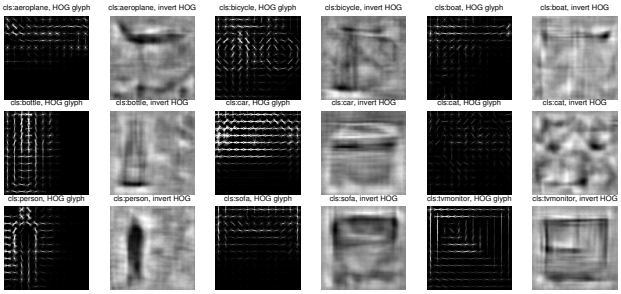

Figure 4: The visualization of the column of U, which is the low-rank projection basis, learned from optimizing (7). We show 6 entries from basis of 6 different categories. The visualization is in both HOG glyph[3] and inverted HOG [30]. The spatial invariance is clearly encoded in the *tvmonitor* category, in which one can see the contours of TV monitors with different scales.

Another important performance measure that affects the final mean AP is the *retrieval rate*, as defined above. The comparison of the retrieval rate of SVD baseline and BSM are shown in Figure 3. As we can see, the retrieval rate of the proposed BSM method largely outperforms the baseline method on all 20 categories, which is consistent with the superior performance in terms of the mean AP measure. Also, the retrieval rate with varying p (while L is fixed), the ratio of the re-evaluated models, is shown in Figure 3. The results in Figure 3 are averaged over all 20 categories.

To better understand what has been learned by optimizing the BSM objective (7), we visualize some entries of the low-rank projection basis, i.e., the columns of matrix U, for various categories in Figure 4 using both the HOG glyph [3] and iHOG [30]. There are several interesting points we would like to note in Figure 4: 1) For categories that have more coherent appearances across large number of exemplars, the basis learned through BSM is visually meaningful, for example, the categories *aeroplane, bicycle, bottle, car, person*, etc. 2) For categories that have large intra-category appearance variations, for example the *cat* category, the learned basis does not seem to be visually meaningful. This is also consistent with the poor detection performance on that category. 3) By looking at the basis of the *tvmonitor* category, we can clearly see the scale invariance encoded in the basis by the contours of the TV monitors with different scales.

For all the results in Figure 2, L, which is the rank of U and V, is set to $0.01 \times D = 45$, whereas p, the ratio of the re-evaluated "model-window" pairs, is set to 0.005. Another parameter specifically for BSM, γ in (7), is set to 0.1.

Parameter Sensitivity: In order to understand the effects of different parameters, L, p and γ. We select 5 out of 20 categories, namely *aeroplane, bicycle, boat, horse and train*, to perform a parameter search over the grid $L = D \times \{0.004, 0.006, 0.008, 0.01\}$, $p = \{0.005, 0.01, 0.015, 0.02\}$

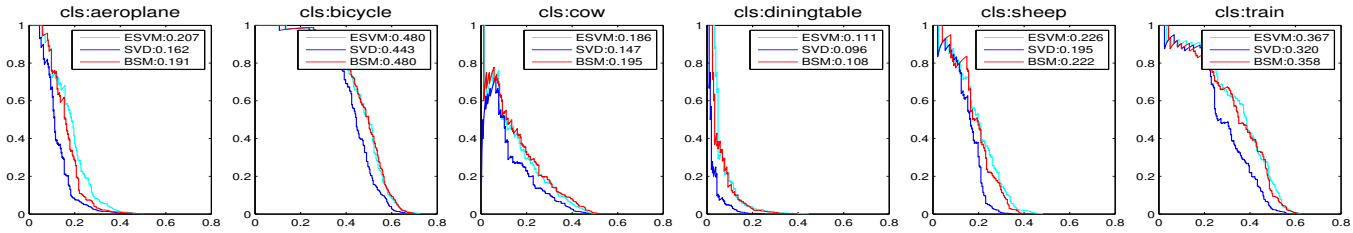

Figure 2: The Precision-Recall curve of 6 out 20 categories in PASCAL VOC 2007 with three methods: the standard ESVM (cyan), the SVD baseline (blue) and the proposed BSM (red). As we can see, on most categories, the performance of BSM outperforms the SVD baseline with a large margin, and is almost the same with ESVM, which evaluates all "model-window" pairs exhaustively.

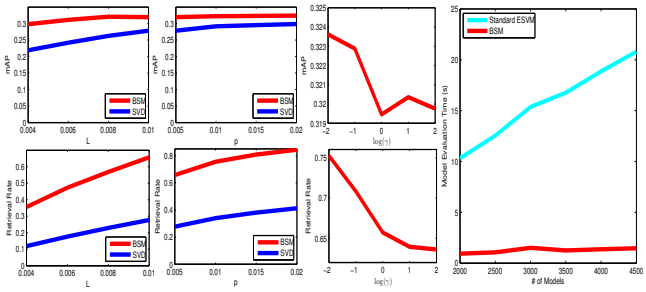

Figure 5: The left 3 columns show the effects of varying the parameters L, p and γ. Two out of three parameters $L = D\times 0.01$, $p = 0.005$ and $\gamma = 0.1$ are fixed to see the effect of another parameter. The first row shows the effects of different parameters on the mAP measure whereas the second row shows the effects on the retrieval rate. In all plots except for those of γ, which is not a parameter in the SVD baseline, both the SVD baseline and the proposed BSM are plotted. The graph on the right-end is the comparison of the computation of model evaluation between directly applying (1) (ESVM) and applying low-rank approximation (BSM).

and $\gamma = \{0.01, 0.1, 1, 10, 100\}$. To show the effects of different parameters, we plot both the mAP measure and retrieval rate in Figure 5. Two of the three values $L = D\times 0.01$, $p = 0.005$ and $\gamma = 0.1$ are fixed to show the effect of another parameter. As we can see from Figure 5, 1) The performance increases as we increase L, which is the rank-constraint. 2) BSM outperforms SVD with a large margin on all sets of parameters. 3) Also, the performance measured by mAP is not sensitive to p, this means that we do not need a large p, which incurs more computation, to get a good detection performance. 4) BSM prefers a small γ which emphasizes the importance of the data-dependent term.

Computation Efficiency: Since the main reason for using low-rank approximation to the parameter matrix is computation efficiency, we perform experiments to study the computational benefits in object detection with BSM compared with the standard ESVM detection. Since we are interested in the effect of using low-rank approximation for model evaluation, we do not consider the time for all pre- and post-processing like computing the HOG and Non Maximum Suppression. We conduct experiments on the *person* category of PASCAL VOC 2007 since it is the category with the largest number of ESVM models (4690 models). We fix the rank L to $0.01\times D = 45$ where we can achieve almost the identical performance to standard ESVM evaluation and show in Figure 5, the computation as the function of M ranging from 2000 to 4500 models, for both the standard ESVM evaluation and the low-rank approximation. As we can see, as we increase M, the computation of the standard ESVM grows much faster than that of BSM, which is consistent with our theoretical analysis in Section 2.4. Note that the speed-up of BSM over the standard ESVM is not as large as the theoretical analysis because in practice, the actual computation time consists of both feature computation and model evaluation. We can achieve around 3 folds speed-up with considering all the pre- and post- processing with un-optimized MATLAB code.

Finally, as we noted earlier, our aim is to reduce the complexity/number of the "model-window" pairs prior to computing the scores, not to optimize the matrix product computation itself. As a result, other approaches that attempt to reduce computation by optimizing the computations involved such as GPU implementations, fast matrix operations, or quantization can be readily incorporated in the BSM implementation to achieve further speed-up.

3.2. Image Classification

For the image classification task, we use the *ImageNet ILSVRC 2010* dataset which is the only version of ILSVRC that has released both its testing images and the ground truth labels for them[6]. To best use the existing pipeline, we adopt the state-of-the-art feature generated by Krizhevsky et al.'s network [15] and trained our own 1000-way softmax regressor W on ILSVRC 2010 training set. We use U and V learned from both SVD and BSM, as we described in Sec-

[6]Note that both the number of categories and images are almost identical to the more recent ILSVRC 2012.

tion 2.6, to obtain an approximation of R as $\tilde{R}$[7]. The performance of the standard softmax classifier W, and the approximations with SVD and BSM is shown in Table 1. First, as we increase the rank L, we get better and better classification performance as we expected. Also, our method could achieve performance that is not only much better than that of the SVD baseline, but also better than the original softmax regression classifier when using only around 4% rank relative to D. We suspect this is because we reduce the overfitting by projecting down the feature dimension.

L/D	0.005	0.01	0.015	0.02	0.04	0.06	0.08	0.1	SOFTMAX
SVD	0.191	0.366	0.470	0.534	0.635	0.672	0.690	0.703	0.728
BSM	**0.444**	**0.634**	**0.690**	**0.709**	**0.735**	**0.745**	**0.750**	**0.750**	

Table 1: Comparison of the classification accuracy between SVD and BSM, as well as the standard softmax regression classifier. The results are shown with different L/D, which is the ratio between L, the constrained rank, and D, the dimension of the feature (In this case 4096).

4. Conclusion

In this paper, we have investigated the problem of efficient model evaluation with a bilinear separation model. By optimizing an objective function which is data-dependent and directly related to the prediction performance for the target task, we are able to achieve the same or even better performance for both object detection (PASCAL VOC 2007) and image classification (ImageNet ILSVRC 2010) tasks with much less computation, i.e., more precisely, with growth in computation much slower than the growth in the number of models, which is consistent with our theoretical analysis on computation efficiency.

References

[1] J. Carreira and C. Sminchisescu. Constrained parametric min-cuts for automatic object segmentation. In *CVPR 2010*, pages 3241–3248. IEEE. 2

[2] A. Coates, P. Baumstarck, Q. Le, and A. Y. Ng. Scalable learning for object detection with gpu hardware. In *IROS 2009*, pages 4287–4293. IEEE. 2

[3] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR 2005*, pages 886–893. IEEE. 1, 6

[4] T. Dean, M. A. Ruzon, M. Segal, J. Shlens, S. Vijayanarasimhan, and J. Yagnik. Fast, accurate detection of 100,000 object classes on a single machine. In *CVPR 2013*, pages 1814–1821. IEEE. 2

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR 2009*. 2

[6] C. Dubout and F. Fleuret. Exact acceleration of linear object detectors. In *ECCV 2012*, pages 301–311. Springer. 2

[7] I. Endres and D. Hoiem. Category independent object proposals. In *ECCV 2010*, pages 575–588. Springer. 2

[8] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 5

[7] We use the same data from ILSVRC 2010 training set to train the BSM.

[9] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester. Cascade object detection with deformable part models. In *CVPR 2010*, pages 2241–2248. IEEE. 2

[10] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *TPAMI 2010*, 32(9):1627–1645. 1

[11] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR 2014*. 2

[12] Y. Gong, S. Kumar, H. A. Rowley, and S. Lazebnik. Learning binary codes for high-dimensional data using bilinear projections. In *CVPR 2013*, pages 484–491. IEEE. 2

[13] J. Gorski, F. Pfeuffer, and K. Klamroth. Biconvex sets and optimization with biconvex functions: a survey and extensions. *Mathematical Methods of Operations Research*, 66(3):373–407, 2007. 2, 4

[14] B. Hariharan, J. Malik, and D. Ramanan. Discriminative decorrelation for clustering and classification. In *ECCV 2012*, pages 459–472. Springer. 1

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS 2012*, volume 1, page 4. 2, 7

[16] C.-P. Lee and C.-J. Lin. A study on l2-loss (squared hinge-loss) multiclass svm. *Neural computation*, 25(5):1302–1323, 2013. 4

[17] Y. Lin, F. Lv, S. Zhu, M. Yang, T. Cour, K. Yu, L. Cao, and T. Huang. Large-scale image classification: fast feature extraction and svm training. In *CVPR 2011*, pages 1689–1696. IEEE. 2

[18] T. Malisiewicz, A. Gupta, and A. A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *ICCV 2011*, pages 89–96. IEEE. 1, 2, 5, 6

[19] S. Manen, M. Guillaumin, L. Van Gool, and K. Leuven. Prime object proposals with randomized prims algorithm. In *ICCV 2013*. 2

[20] M. Pedersoli, A. Vedaldi, and J. Gonzalez. A coarse-to-fine approach for fast deformable object detection. In *CVPR 2011*, pages 1353–1360. IEEE. 2

[21] H. Pirsiavash and D. Ramanan. Steerable part models. In *CVPR 2012*, pages 3226–3233. IEEE. 2

[22] H. Pirsiavash, D. Ramanan, and C. Fowlkes. Bilinear classifiers for visual recognition. In *NIPS 2009*, pages 1482–1490. 1, 2

[23] J. D. Rennie and N. Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *ICML 2005*, pages 713–719. ACM. 4

[24] M. A. Sadeghi and D. Forsyth. Fast template evaluation with vector quantization. In *NIPS 2013*, pages 2949–2957. 2

[25] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek. Image classification with the fisher vector: Theory and practice. *International journal of computer vision*, 105(3):222–245, 2013. 2

[26] H. O. Song, T. Darrell, and R. B. Girshick. Discriminatively activated sparselets. In *ICML 2013*, pages 196–204. 1, 2, 3

[27] H. O. Song, S. Zickler, T. Althoff, R. Girshick, M. Fritz, C. Geyer, P. Felzenszwalb, and T. Darrell. Sparselet models for efficient multiclass object detection. In *ECCV 2012*, pages 802–815. Springer. 1, 2, 3

[28] J. B. Tenenbaum and W. T. Freeman. Separating style and content with bilinear models. *Neural computation*, 12(6):1247–1283, 2000. 1, 2

[29] P. Viola and M. Jones. Robust real-time object detection. *International Journal of Computer Vision*, 4:34–47, 2001. 2

[30] C. Vondrick, A. Khosla, T. Malisiewicz, and A. Torralba. Inverting and visualizing features for object detection. *arXiv preprint arXiv:1212.2278*, 2012. 6

[31] J. Ye, R. Janardan, Q. Li, et al. Two-dimensional linear discriminant analysis. In *NIPS 2004*, volume 4, page 4. 2