

Spotlight the Negatives: A Generalized Discriminative Latent Model

Hossein Azizpour¹
azizpour@kth.se

Mostafa Arefiyan²
mostafa@brown.edu

Sobhan Naderi Parizi²
sobhan@brown.edu

Stefan Carlsson¹
stefanc@csc.kth.se

¹ CVAP
Royal Institute of Technology (KTH)
Stockholm, Sweden

² LEMS
Brown University
Providence, USA

Abstract

Discriminative latent variable models (LVM) are frequently applied to various visual recognition tasks. In these systems the latent (hidden) variables provide a formalism for modeling structured variation of visual features. Conventionally, latent variables are defined on the variation of the foreground (positive) class. In this work we augment LVMs to include *negative* latent variables corresponding to the background class. We formalize the scoring function of such a generalized LVM (GLVM). Then we discuss a framework for learning a model based on the GLVM scoring function. We theoretically showcase how some of the current visual recognition methods can benefit from this generalization. Finally, we experiment on a generalized form of Deformable Part Models with negative latent variables and show significant improvements on two different detection tasks.

1 Introduction

Discriminative latent variable models (LVM) are frequently and successfully applied to various visual recognition tasks [6, 7, 10, 13, 12, 15, 16]. In these systems the latent (hidden) variables provide a formalism for modeling structured variation of visual features. These variations can be the result of different possible object locations, deformations, viewpoint, subcategories, etc. Conventionally, these have all been defined only based on the foreground (positive) class. We call such latent variables “positive”. In this work we introduce generalized discriminative LVM (GLVM) which use both “positive” and “negative” latent variables. Negative latent variables are defined on the background (negative) class and provide *counter evidence* for presence of the foreground class. They can, for instance, learn mutual exclusion constraints, model scene subcategories where the positive object class is unlikely to be found, or capture specific parts which potentially indicate the presence of an object of a similar but not the same class.

The objective of the proposed framework is to learn a model which maximizes the evidence (characterized by positive latent variables) for the foreground class and at the same time minimizes the counter evidence (characterized by negative latent variables) lying in background class. Thereby GLVM empowers the latent variable modeling by highlighting the role of

negative data in its own right. An interesting analogy is with game theory; when playing a game a simple strategy is to maximize your score. This resembles LVM's objective which is to maximize evidence for the presence of the foreground class. A better strategy, however, is to maximize your scores at the same time as minimizing the opponents' scores. This is analogous to GLVM's objective which takes into account counter evidence emerging from negative latent variables while looking for evidence from the foreground class. The score of the opponents is counter evidence in our hypothetical example.

The concept of *negative parts* was noted in [10]. However, [10] focuses on automatic discovery and optimization of a part based model with negative parts. In this paper we extend the notion of negative parts to negative latent variables and propose a framework for defining models that use both positive and negative latent variables.

Many different modeling techniques benefit from latent variables. Felzenszwalb *et al.* [6] introduced latent SVM to train deformable part models, a state of the art object detector [7]. Yu and Joachims [16] transferred the idea to structural SVMs. Yang *et al.* [15] proposed a kernelized version of latent SVM and latent structural SVM and applied it to various visual recognition tasks. Razavi *et al.* [13] introduced latent variables to the Hough transform and achieved significant improvements over a Hough transform object detection baseline. And-Or trees use latent variable modeling for object recognition and occlusion reasoning [14].

In this paper we introduce a novel family of models which generalizes discriminative latent variable models. We review LVMs in Section 2, formulate our generalization of LVMs in Section 3. Then, we discuss training of GLVMs and propose an algorithm for learning the parameters of a GLVM in Section 4. In Section 6 we discuss how various computer vision models can benefit from GLVM. Finally, in Section 7 we experiment on generalized DPMs.

2 Discriminative Latent Variable Models (LVM)

Consider a supervised classification problem provided with a set of n training examples $X \in \mathcal{R}^{d \times n}$ and their corresponding labels $Y \in \mathcal{Y}^n$. The goal of discriminative learning is then to learn a scoring function $S_w(x, y)$, parametrized by w , that ranks a data point x_i and its true label y_i higher than the rest of labels. For a linear scoring function this is written as follows:

$$S_w(x_i, y_i) > S_w(x_i, y) \quad \forall y \neq y_i \quad (1)$$

$$S_w(x_i, y) = w^T \phi(x_i, y) \quad (2)$$

Here $\phi(x, y)$ is the representation function (e.g. Histogram of Oriented Gradients, ConvNet representation, etc.). Since the scoring function of (2) is linear in w any invariance to non-linear object class deformation should be reflected in the representation function ϕ . This requires a very rich feature encoding. LVMs model known structured variations of the object class by using various latent variables a corresponding to different sources of variation. Each latent variable models a source of variation and accordingly alters the representation ϕ so that it best matches the model parameters w . Thus, all latent variables in an LVM are maximized over. For instance, in weakly supervised object detection location of the object in an image is modeled by a latent variable. The scoring function of an LVM takes this form:

$$S_w(x_i, y) = \max_a w^T \phi(x_i, y, a) \quad (3)$$

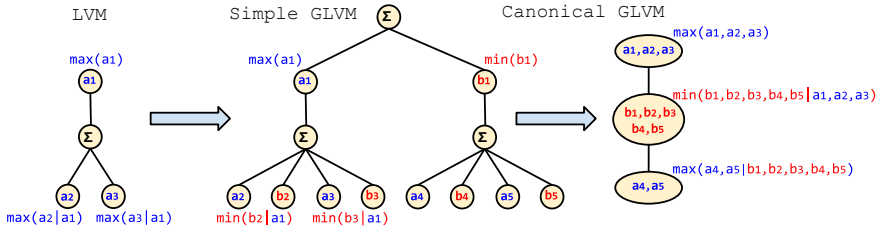


Figure 1: Example of an LVM scoring function on three latent variables (left), a Simple GLVM scoring function (middle), and a GLVM in its canonical form. The models become more and more complex from left to right.

In some cases the set of labels \mathcal{Y} comprises of only known classes. In open-set classification, however, one of the possible labels in \mathcal{Y} refers to an *unknown class*. The simplest form of open-set classification is binary classification where samples are classified into the class of interest where $y = +1$ (e.g. cat) or anything else where $y = -1$ (e.g. non-cat) which are so called *foreground* and *background* respectively. In this work we are interested in open-set classification problems which suit many real-world applications better. In the next section we show how the latent variables in the scoring function of (3) can be altered to “model” the negative data in its own right and thereby enrich the expressive power of discriminative latent variable models. In the rest of the paper we focus on the binary classification problem.

3 Generalized LVM with Negative Modeling (GLVM)

The scoring function in discriminative LVM is usually constructed in such a way that it looks for “evidence” for the foreground class, and the weakness (or lack) of such evidence indicates that the input is from the background class. For instance, in detecting *office* scenes, the location of a work desk (evidence for office) can be modeled by a latent variable.

We generalize LVMs by equipping them with latent variables that look for “counter evidence” for the foreground class. For example, detecting a stove (evidence for kitchen) or sofa (evidence for living room) provides counter evidence for office. Such latent variables can also be contextual, for example for a *cow* detector, presence of saddle/rider/aeroplane counts as counter evidence whereas presence of meadow/stable/milking parlor counts as evidence. For visual examples refer to Section 1 in supplementary material.

We call the latent variables that collect evidence for the foreground class *positive latent variables* and the latent variables that collect counter evidence for the foreground class *negative latent variables* and denote them by a and b respectively throughout the paper.

Representation function. Before we go further, let’s explain how our representation function ϕ is constructed. We assume the model parameters (associated with *all* latent variables) are concatenated into a single vector w . Moreover, w and its corresponding feature vector $\phi(x_i, a_1, b_1, a_2, b_2, \dots)$ have the same dimensionality. Every latent variable a_k or b_k encodes its own place in $\phi()$. The places for different latent variables do not overlap. When some latent variables are omitted from the inputs of $\phi()$, their place is filled with zero vectors. These assumption are made for the ease of notation in the upcoming formulations. For clarity we take three steps towards defining the generic form of GLVM scoring function.

Simple Form. The simplest form of a GLVM scoring function involving two independent sets of positive and negative latent variables is (\mathcal{Z} is the set of possible latent assignments):

$$S_w(x_i, y) = \max_{a \in \mathcal{Z}^+} w^T \phi(x_i, y, a) - \max_{b \in \mathcal{Z}^-} w^T \phi(x_i, y, b) \quad (4)$$

Note that the second max cannot be absorbed in the first max. Thus, the GLVM scoring function is more general than that of LVM. In particular, (4) reduces to (3) when $|\mathcal{Z}^-| \leq 1$. One can characterize positive and negative latent variables according to the way they impact the value of the scoring function. Each assignment of a latent variable has a corresponding value associated to it. For instance, in the deformable part models (DPMs) latent variables indicate the location of the object parts [6]. Each assignment of a latent variable in a DPM corresponds to a placement of a part which in turn is associated with a value in the score map of the part. A positive (negative) latent variable is one whose associated value is positively (negatively) correlated with the final value of the scoring function. This should not be confused with negative entries in the model vector w . In a linear model with the score function $S(x) = w^T \phi(x)$ the value of the d -th dimension of the feature vector (i.e. $\phi(x)_d$) is negatively correlated with $S(x)$ when $w_d < 0$. Yet, negative latent variables are fundamentally different from this. In particular, a negative latent variable cannot be obtained by flipping the sign in some dimensions of the parameter vector and using a positive latent variable instead. The value associated with a negative latent variable (i.e. $\max_b w^T \phi(x, b)$) is a convex function of w whereas the value associated to negative entries in w (i.e. $w_d \phi(x)_d$) is linear in w .

Dependent Form. The value of the GLVM score can be thought of as the value of the best strategy in a 2-player game. To show this we need to modify the notation as follows:

$$\begin{aligned} S_w(x_i) &= \max_{a \in \mathcal{Z}^+} w^T \phi(x_i, a) + \min_{b \in \mathcal{Z}^-} -w^T \phi(x_i, b) \\ &= \max_{a \in \mathcal{Z}^+} \min_{b \in \mathcal{Z}^-} w^T \phi(x_i, a) - w^T \phi(x_i, b) \\ &= \max_{a \in \mathcal{Z}^+} \min_{b \in \mathcal{Z}^-} w^T \phi(x_i, a, b) \end{aligned} \quad (5)$$

In the first step we dropped y from the notation since we will be focusing on binary classification. In the second step min is pulled back. For the last step we let $\phi(x_i, a, b) = \phi(x_i, a) - \phi(x_i, b)$ although any feature function that *depends on both* b and a would do. So, (5) is more general than (4). Note that the game theory analogy of minimax is now more apparent in the new formulation of (5).

Canonical Form. The scoring function in (5) is linear when the latent variables a and b are fixed. One can imagine augmenting this linear function with more positive and negative latent variables; the same way that we extended (2) to get to (5). This creates multiple *interleaved* negative and positive latent variables. This way, we can make a hierarchy of alternating latent variables recursively. This general formulation has interesting ties to compositional models, like object grammar [6], and high-level scene understanding. For example, we can model scenes according to the presence of some objects (positive latent variable) and the absence of some other objects (negative latent variable). Each of those objects themselves can be modeled according to presence of some parts (positive) and absence of others (negative). We can continue the recursion even further. The GLVM scoring function, after being expanded recursively, can be written in the following general form:

$$S_w(x_i) = \max_{a_1} \min_{b_1} \max_{a_2} \min_{b_2} \dots \max_{a_K} \min_{b_K} w^T \phi(x_i, (a_k, b_k)_{k=1}^K) \quad (6)$$

We call this the *canonical form* of a GLVM scoring function. Note that any other scoring function that is a linear combination of max's and min's of linear functions can be converted to this canonical form by simple algebraic operations. Moreover, (6) with $K > 1$ and non empty a_k and b_k cannot be simplified into (5), in general. Its proof follows the *max-min inequality* [11]. See Figure 1 for an example.

4 Training GLVMs for Classification

In this section we propose an algorithm for training GLVMs to do binary classification. We focus on SVM training objective and use hinge loss. That being said, the same idea can be used to optimize any training objective that involves a linear combination of convex and/or concave functions of GLVM scores; *e.g.* SVM with squared hinge loss. For the ease of notation we drop the subscript w in S_w in the rest of this section.

Let $D = \{(x_i, y_i)\}_{i=1}^n$ denote a set of n labeled training examples. Each training example is a pair (x_i, y_i) where x_i is the input data and $y_i \in \{+1, -1\}$ is the ground-truth label associated with it. SVM training objective can be written as follows:

$$O(w) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \max(0, 1 - y_i S(x_i)) \quad (7)$$

where $S(x_i)$ is a scoring function. Training a classifier requires finding the model w^* that minimizes the function *i.e.* $w^* = \arg \min_w O(w)$. In most practical applications, however, the objective function O is complex (in particular, non-convex). This makes it hard to find the global minimizer. Thus, we usually have to resort to local minima of the objective function.

The concave-convex procedure (CCCP) [14] is a general framework for optimizing non-convex functions. For minimization, in each iteration the concave part is replaced by a hyperplane tangent to it at the current solution. The result is a convex function. Local minima of a convex function are global minima. Thus, one can use simple methods such as gradient descent to minimize a convex function.

We train GLVMs using an iterative algorithm that is very similar to CCCP [14]. But, unlike CCCP, we do not linearize the concave part of the objective function. Our training algorithm alternates between two steps: 1) constructing a convex upper bound to the training objective O and 2) minimizing the bound. Let $B_t(w)$ denote the bound constructed at iteration t . Also let $w_t = \arg \min_w B_t(w)$ be the minimizer of the bound. All bounds must touch the objective function at the solution of the previous bound; that is $B_t(w_{t-1}) = O(w_{t-1}), \forall t$. This process is guaranteed to converge to a local minimum or a saddle point of the training objective O .

To construct bounds B_t , as we shall see next, we only need to construct a concave lower bound $\hat{S}(x_i)$ and a convex upper bound $\check{S}(x_i)$ to the scoring function of (6). We define B_t as follows:

$$B_t(w) = \frac{1}{2} \|w\|^2 + C \sum_{\substack{i=1 \\ y_i=-1}}^n \max(0, 1 + \check{S}_t(x_i)) + C \sum_{\substack{i=1 \\ y_i=+1}}^n \max(0, 1 - \hat{S}_t(x_i)) \quad (8)$$

In the next section we will explain how $\check{S}(x_i)$ and $\hat{S}(x_i)$ are constructed for GLVMs. Algorithm 1 in the supplementary material summarizes all steps in our training procedure.

Convex Upper Bound and Concave Lower Bound. Here we explain how we obtain $\check{S}_t(x_i)$ and $\hat{S}_t(x_i)$ when GLVM scoring function (6) is used. We fix all the negative latent variables in $S(x_i)$ to get $\check{S}_t(x_i)$ and fix all the positive latent variables to get $\hat{S}_t(x_i)$.

Let w_t denote the model at iteration t and let $a = (a_1, \dots, a_K)$ be an arbitrary assignment of the positive latent variables. Given a , we denote the fixed values for the negative latent

variables on image x_i by $b^{(i)}(a) = (b_1^{(i)}, \dots, b_K^{(i)})$ and define it as in (9). Similarly, we define $a^{(i)}(b) = (a_1^{(i)}, \dots, a_K^{(i)})$ as in (10).

$$b^{(i)}(a) = \arg \min_{b_1, \dots, b_K} w_t^T \phi \left(x_i, (a_k, b_k)_{k=1}^K \right) \quad (9)$$

$$a^{(i)}(b) = \arg \max_{a_1, \dots, a_K} w_t^T \phi \left(x_i, (a_k, b_k)_{k=1}^K \right) \quad (10)$$

Note that (9) uses a min and (10) uses a max. Also, note that in both equations the fixed assignments are computed according to w_t . Finally, we define $\check{S}_t(x_i)$ and $\hat{S}_t(x_i)$ as follows:

$$\check{S}_t(x_i) = \max_{a_1} \max_{a_2} \dots \max_{a_K} w_t^T \phi \left(x_i, \left(a_k, b_k^{(i)}(a) \right)_{k=1}^K \right) \quad (11)$$

$$\hat{S}_t(x_i) = \min_{b_1} \min_{b_2} \dots \min_{b_K} w_t^T \phi \left(x_i, \left(a_k^{(i)}(b), b_k \right)_{k=1}^K \right) \quad (12)$$

(11) is convex because it is a max of linear functions and it is an upper bound to $S(x)$ because the min functions are replaced with a fixed value. Similarly for (12).

For discussions regarding the dependency structure of the latent variables and how it affects the scoring and inference functions refer to Section 2 in the supplementary material.

5 Connection to existing Models

LSSVM: An interesting connection is to latent structural SVMs (LSSVM) [16] where the scoring function is given by,

$$S(x_i, y) = \max_h w^T \phi(x_i, y, h) \quad (13)$$

In this form, the formulation of scoring function does not make use of negative latent variables as defined in GLVM. In LSSVM objective function the individual loss induced by a sample (x_i, y_i) is calculated as below,

$$L(x_i, y_i) = \max_{(y, h)} [w^T \phi(x_i, y, h) + \Delta(y_i, y, h)] - \max_h w^T \phi(x_i, y_i, h) \quad (14)$$

Note that the representation function takes as input both class y and latent variable h . Now, assume a binary classification ($y \in \{-1, 1\}$) framework with a symmetric loss function Δ independent of h . Then, the scoring and loss function can be reformulated as,

$$S'(x_i) = \max_h w^T \phi(x_i, +1, h) - \max_h w^T \phi(x_i, -1, h) \quad (15)$$

$$L(x_i, y_i) = \max(0, \Delta(+1, -1) - y_i S'(x_i)) \quad (16)$$

Note that (15) is similar to the shallow structure of GLVMs in (4). This shows how negative latent variables can potentially emerge from LSSVMs.

Mid-level features based recognition: Many recent scene image classification techniques pre-train multiple mid level features (parts) [8, 9, 9] to represent an image. They construct the representation of an image by concatenating the maximum response of each part in the image and then train a linear classifier on top of those representations. Naderi *et al.* [10] have shown that training a classifier on those representations effectively corresponds to the definition of *negative parts*. In these works, the score of a part at an image is *maximized* over

different locations and thus its position and scale are, in essence, treated as a latent variables. In that respect, a negative weight in the linear classifier for such a latent variable can be translated to a *negative latent variable* (i.e. its maximum score contributes negatively to the score of an image). The training procedure for the scene classification models of [9, 9, 9, 10] is similar to the shallow version of the GLVM framework proposed in this paper.

ConvNet: [9] argues that the kernels in Convolutional Networks (ConvNet) can be interpreted as parts in part based models and shows that DPM is effectively a ConvNet. However, since ConvNets have the extra ability of learning weights (potentially negative) over these multiple kernels it is modeling counter evidences in a similar form as GLVM.

6 Showcases

To elicit the implications of the proposed generalization of LVMs, in this section, we review a few different state of the art computer vision models and demonstrate how they can benefit from the proposed framework. We consider adding negative latent variables to deformable part models, And-Or trees, and latent Hough transform and discuss the implications of it.

Show Case 1: Deformable Part Models. Deformable Part Model (DPM) [6] is an object detector that models an object as a collection of deformable parts. DPM uses multiple mixture components to model viewpoint variations of the object. The location of the parts (denoted by l_j) and the choice of the mixture component (indexed by c) are unknown and thus treated as latent variables. The scoring function of a DPM with n parts can be written as follows:

$$S(x) = \max_{c=1}^k \sum_{j=1}^n \max_{l_j} w_{c,j}^T \phi(x, l_j, c) \quad (17)$$

We extend DPMs to include negative parts. We refer to this as generalized DPM (GDPM). For instance, when detecting cow, positive parts in DPM include head, back, legs of a cow. In GDPM one can imagine several candidates for negative parts. For example, since horses appear frequently in the high scoring false positives of cow detectors, saddle or horse-head may be good negative parts for a cow detector. We define GDPM scoring function as:

$$S(x) = \max_{c=1}^k \left[\sum_{j^+=1}^n \max_{l_{j^+}} w_{c,j^+}^T \phi(x, l_{j^+}, c) - \sum_{j^-=1}^m \max_{l_{j^-}} w_{c,j^-}^T \phi(x, l_{j^-}, c) \right] \quad (18)$$

Following the recursion of GLVM, the outer max can also be cloned with a negative counterpart. In that respect, the negative *components* would look, for instance, for different viewpoints of a horse with its own positive and negative parts. We implemented GDPM and evaluated it on two different datasets. We discuss the experimental results in Section 7.

Show Case 2: And-Or trees. And-Or trees [12] are hierarchical models that have been applied to various visual recognition tasks. And-Or trees are represented by a tree-structured graph. Edges in the graph represent dependencies and nodes represent three different operations. *Terminal* nodes (T_i) are directly applied to the input image. A typical example of a terminal node is a template applied at a specific location of the image. *Or* nodes (O_i) take the maximum value among their children. An *Or* node can model, for example, the placement of terminal nodes or the choice of mixture components. *And* nodes (A_i) aggregate information

by summing over their children. The scoring functions of these nodes are defined as follows:

$$S(T_i|I; w_i) = w_i^T I(T_i), \quad S(A_i|C) = \sum_{c \in C(A_i)} S(c), \quad S(O_i|C) = \max_{c \in C(O_i)} S(c) \quad (19)$$

C encodes the hierarchy of the nodes and $C(N)$ denotes children of node N . In order to compute the score of an input image the scores are propagated from the terminal nodes to the root of hierarchy using dynamic programming. Or nodes find the most plausible interpretation, And nodes aggregate information, and terminal nodes collect positive evidence for the foreground class. Generalizing an And-Or tree with negative latent variables introduces a new type of node which we call a *Nor* node (N_i). Nor nodes collect counter evidence for the class of interest. We define the scoring function of a Nor node as follows:

$$S(N_i|C) = \min_{c \in C(N_i)} S(c) \quad (20)$$

Show Case 3: Latent Hough Transform. Hough transform (HT) has been used as a non-parametric voting scheme to localize objects. In Implicit Shape Models [10] the score of a hypothesis h (e.g. object location and scale) in an image x is calculated by aggregating its compliance to each *positive* training images. Let D_p denote the set of positive training images. Also let $S(x, h|x_i)$ denote the vote of a positive training image $x_i \in D_p$ in favor of hypothesis h . The scoring function of HT is defined as follows:

$$S_{HT}(x, h) = \sum_{i \in D_p} S(x, h|x_i) \quad (21)$$

One of the main issues with HT is the aggregation of votes from inconsistent images. For instance, training images of side-view cars contribute to the voting in a test image of a frontal-view car. Razavi *et al.* [13] alleviate this issue by introducing latent Hough transform (LHT). LHT groups consistent images (e.g. exhibiting same viewpoints) together using a latent variable $z \in Z$. Let $\phi(x, h)$ denote the vector of votes for hypothesis h obtained from all positive training images *i.e.* $\phi(x, h)_i = S(x, h|x_i)$. LHT assigns weight $w_{z,i}$ to the positive training image $x_i \in D_p$ indicating its relevance to the selected mixture component z . Scoring function of an LHT parameterized by w is defined as follows:

$$S_{LHT}(x, h) = \max_{z \in Z} w_z^T \phi(x, h) \quad (22)$$

In order to reformulate the problem as a GLVM we need to introduce *negative votes* [14] cast by negative training images in the scoring function of HT.

$$S_{HT}(x, h) = \sum_{i \in D_p} S(h|x_i) - \sum_{i \in D_n} S(h|x_i) \quad (23)$$

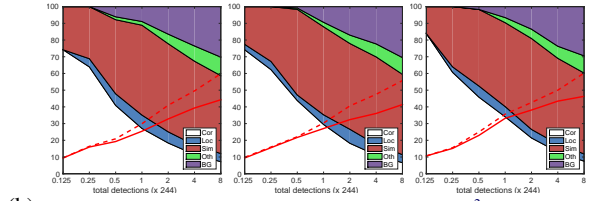
Then the scoring function of a GLVM generalization of LHT (called GLHT) is given by:

$$S_{GLHT}(x, h) = \max_{z^+} w_{z^+}^T \phi(x, h) - \max_{z^-} w_{z^-}^T \phi(x, h) \quad (24)$$

It should be emphasized that negative latent variables might be crucial when using negative votes in HT due the high multimodality of negative data distribution. That is negative images are probably more susceptible to the aggregation of inconsistent votes. This might explain the reason why negative voting is not popular for HT based object recognition. Note that GLHT can make use of the negative data in a more effective way.

	bird	cat	cow	dog	horse	sheep
DPM₈	10.9	16.6	21.9	12.5	56.0	18.0
GDPM₁⁷	10.9	17.0	22.0	12.5	57.2	19.2
GDPM₆²	9.5	18.6	23.2	12.2	56.0	19.8
GDPM₅³	10.5	13.9	21.7	11.9	54.8	19.1

(a) AP for animal classes of VOC 2007, comparing DPM with GDPM with different number of parts. Sub (super) indices indicate the number of positive (negative) parts.



(b) Analysis of false positives of *cow* using [1]. DPM (left), GDPM₆² (middle) and GDPM₁⁷ (right) with similar part initialization. The volumes show the commulative number of top detections within 5 different categories: true detection (white), localization FP (blue), background FP (purple), similar objects (red), other objects (green). Red (dashed red) line is the percentage of recall at each number of detections using 50% (10%) overlap threshold. GDPM with careful initialization (right figure) slightly reduces confusion with similar objects.

7 Experiments


Figure 2: PASCAL VOC 2007

To demonstrate the efficacy of a GLVM we evaluate it on deformable part models (DPM) by augmenting it with negative parts (GDPM). As discussed in Section 4 adding negative parts changes the optimization framework of DPMs. This includes many noteworthy details about inference, relabeling, data mining, features cache, etc. For details of these modifications refer to Section 4 in supplementary material. We conducted our experiments of GDPM on two different datasets. Similar to [1] HOG is used as feature descriptor but the observed trends should be independent of this choice. The results are measured using Average Precision of the PR-curve. We use the PASCAL VOC criterion of 50% intersection over union for *recall*.

PASCAL VOC 2007 Animals. We first evaluate GDPM on animal subset of VOC 2007. The training set of VOC 2007 contains about 4500 negative images and a few hundreds (~ 400) of positive instances per animal class. The same goes for the test set. We use the same hyper-parameters as used for original DPM [1], namely 6 components (3 + 3 mirror components) and 8 parts per component. Our results slightly differ from those reported in [1] due to absence of post-processing (bounding box prediction and context re-scoring) and slight differences in implementation. Figure 2a summarizes the results for substituting some positive parts in DPM with negative parts. 4 out of the 6 tested classes benefit from replacing positive parts with negative parts. Depending on the class the optimal number of negative parts is different. For PR curves refer to Section 6 in supplementary material.

Initialization. Similar to DPM [1], we initialized the location and appearance of positive (negative) parts by finding the sub-patches of the root filter which contained highest positive (negative) weights. However, we observed that negative parts in GDPM are quite sensitive to initialization. This is due to the non-convexity of the optimization. Furthermore, while the positive label corresponds to a single visual class the negative label encompasses more classes and thus exhibits a multi-modal distribution. But, we are interested in *discrimination* of the positives, thus we only need to model the boundary cases of the negative distribution. Studying these boundary cases for DPM model of *cow* revealed that most of negative samples come from "sheep" and "horse". Initializing GDPM negative parts of *cow* with a part from similar objects increased the performance from **21.9** to **26.7**.

Negative Parts. Although, adding negative parts helps in general, it seems that, unless the parts are carefully initialized, the discovered negative parts with default initialization are not visually meaningful. We analyzed the false positives (FP) of DPM and GDPM detectors sim-



	Abyssinian	Bengal	Birman	Bombay	British Shorthair	Egyptian Mau	Maine Coon	Persian	Ragdoll	Russian Blue	Siamese	Sphynx
DPM_4	21.3	12.8	34.5	23.3	32.2	15.8	21.6	28.0	19.4	24.0	29.4	22.7
DPM_6	22.2	12.8	31.4	21.5	31.3	16.5	26.0	29.0	20.6	25.0	30.9	22.0
$GDPM_2^2$	24.3	11.9	38.4	23.9	31.0	19.7	27.7	29.7	24.5	29.9	35.9	27.4
$GDPM_4^2$	25.5	13.7	34.0	23.0	33.5	20.9	24.5	30.0	21.9	25.3	30.6	23.2

Table 1: Cat Head Detection: Results comparing DPM and GDPM with different number of positive and negative parts. Sub (super) indices show the number of positive (negative) parts. GDPM consistently outperforms DPM variants. The average cat head images are taken from [10].

ilar to [8] and observed that some of the FP reduction in GDPM is from non similar object classes or background. However, when we initialized the negative parts using other classes, the ratio of FP due to similar objects decreased more significantly (See Figure 2b).

Cat Head. The second task is detection of heads for 12 breeds of cats. The cat head images are taken from Oxford Pets dataset [10]. For each cat breed the distractor set contains full images (including background clutter) of the other 11 breeds and 25 breeds of dogs. Each cat breed has around 65 positive images and 3500 negative images for training. Test set has the same number of images. This task is particularly hard due to the low number of positive training images and high level of similarity between cats heads which is sometimes hard for humans to distinguish. For sample images refer to Section 5 in supplementary material. Table 1 reports the results by comparing DPM and GDPM using various number of positive/negative parts. The aligned average images in [10] is used as header in Table 1. Since the number of positive images in the training set is low, we used one component per detector. It can be seen that GDPM consistently outperform the DPM using different number of parts. For a few classes (*e.g.* Bengal) DPM with 4 positive parts outperform GDPM with 2 positive and 2 negative parts, but as we increase the number of parts to 6, replacing 2 positive parts with 2 negative parts proves beneficial. We observed that adding more parts (more than 6) degrades the results due to over-fitting to the small number of positive images. Negative parts seems to be more robust for the classes where going from total of 4 parts to 6 parts show signs of over-fitting (*e.g.* Birman, Sphynx).

8 Conclusion

In this work we proposed a new framework for discriminative latent variable models by *explicitly* modeling *counter evidences* associated with negative latent variables along with the positive evidences and by that emphasized the role of negative data. The proposed framework enriches the expressive power of latent variable models and extends their applicability to various vision tasks. We also described a general learning framework and showcased how common computer vision latent models can benefit from the new perspective. Finally, we experimented on two tasks of object detection revealing the benefits of negative latent variables. We further observed that expressive power of GLVM crucially depends on a careful initialization.

We think that various latent variable models can benefit from the proposed GLVM framework. We hope that our work yields to fruitful discussions in the field regarding latent variable models.

Acknowledgement. We’d like to thank Pedro Felzenszwalb for the fruitful discussions. This work has been funded by the Swedish Foundation for Strategic Research (SSF) within the project VINST.

References

- [1] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004. ISBN 0521833787.
- [2] Christopher M. Brown. Inherent bias and noise in the Hough transform. *PAMI*, 5(5):493–505, 1983.
- [3] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Mid-level visual element discovery as discriminative mode seeking. In *NIPS*, pages 494–502, 2013.
- [4] Ian Endres, Kevin J. Shih, Johnston Jiaa, and Derek Hoiem. Learning collections of part models for object recognition. In *CVPR*, pages 939–946, 2013.
- [5] Pedro Felzenszwalb and David McAllester. Object detection grammars. In *Technical Report*, 2010.
- [6] Pedro F. Felzenszwalb, David A. McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *CVPR*, 2008.
- [7] Ross B. Girshick, Forrest N. Iandola, Trevor Darrell, and Jitendra Malik. Deformable part models are convolutional neural networks. In *CVPR*, 2015.
- [8] Derek Hoiem, Yodsawalai Chodpathumwan, and Qieyun Dai. Diagnosing error in object detectors. In *ECCV*, pages 340–353, 2012.
- [9] Mayank Juneja, Andrea Vedaldi, C. V. Jawahar, and Andrew Zisserman. Blocks that shout: Distinctive parts for scene classification. In *CVPR*, pages 923–930, 2013.
- [10] Bastian Leibe, Ales Leonardis, and Bernt Schiele. Robust object detection with interleaved categorization and segmentation. *IJCV*, 77(1-3):259–289, 2008.
- [11] Sobhan Naderi Parizi, Andrea Vedaldi, Andrew Zisserman, and Pedro F. Felzenszwalb. Automatic discovery and optimization of parts for image classification. In *ICLR*, 2015.
- [12] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar. Cats and dogs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.
- [13] Nima Razavi, Juergen Gall, Pushmeet Kohli, and Luc J. Van Gool. Latent Hough transform for object detection. In *ECCV*, pages 312–325, 2012.
- [14] Xi Song, Tianfu Wu, Yunde Jia, and Song-Chun Zhu. Discriminatively trained and-or tree models for object detection. In *CVPR*, pages 3278–3285, 2013.
- [15] Weilong Yang, Yang Wang, Arash Vahdat, and Greg Mori. Kernel latent SVM for visual recognition. In *NIPS*, pages 818–826, 2012.
- [16] Chun-Nam John Yu and Thorsten Joachims. Learning structural svms with latent variables. In *ICML*, pages 1169–1176, 2009.
- [17] Alan Yuille and Anand Rangarajan. The concave-convex procedure. *Neural Computation*, 15(4): 915–936, 2003.
- [18] Jun-Yan Zhu, Yong Jae Lee, and Alexei A Efros. Averageexplorer: Interactive exploration and alignment of visual data collections. *SIGGRAPH*, 33(4), 2014.