Joint calibration of Ensemble of Exemplar SVMs

Davide Modolo[1], Alexander Vezhnevets[1], Olga Russakovsky[2], Vittorio Ferrari[1]
[1]*University of Edinburgh* [2]*Stanford University*

Abstract

We present a method for calibrating the Ensemble of Exemplar SVMs model. Unlike the standard approach, which calibrates each SVM independently, our method optimizes their joint performance as an ensemble. We formulate joint calibration as a constrained optimization problem and devise an efficient optimization algorithm to find its global optimum. The algorithm dynamically discards parts of the solution space that cannot contain the optimum early on, making the optimization computationally feasible. We experiment with EE-SVM trained on state-of-the-art CNN descriptors. Results on the ILSVRC 2014 and PASCAL VOC 2007 datasets show that (i) our joint calibration procedure outperforms independent calibration on the task of classifying windows as belonging to an object class or not; and (ii) this improved window classifier leads to better performance on the object detection task.

1. Introduction

The Ensemble of Exemplar SVMs [1] (EE-SVM) is a powerful non-parametric approach to object detection. It is widely used [2–12] because it explicitly associates a training example to each object it detects in a test image. This enables transferring meta-data such as segmentation masks [1, 7], 3D models [1], viewpoints [12], GPS locations [10] and part-level regularization [4]. Furthermore, EE-SVM can also be used for discovering objects parts [3, 5], scene classification [3, 9], object classification [6], image parsing [7], image matching [2], automatic image annotation [11] and 3D object detection [8].

An EE-SVM is a large collection of linear SVM classifiers, each trained from one positive example and many negative ones (an E-SVM). At test time each window is scored by all E-SVMs, and the highest score is assigned to the window. Because of this max operation, it is necessary to calibrate the E-SVMs to make their scores comparable. A common procedure is to calibrate each SVM independently, by fitting a logistic sigmoid to its output on a validation set [1]. Such independent calibration, however, does not take into account that the final score is the max over many E-SVMs. Moreover, calibrating one E-SVM in isolation requires choosing which positive training samples it should score high and which ones it can afford to score low. Such a prior association of positive training samples to E-SVMs is arbitrary, as there is no predefined notion of how much and in which way a particular E-SVM should generalize. What truly matters is the interplay between all E-SVMs through the max operation.

In this paper we present a *joint* calibration procedure that takes into account the max operation. We calibrate all E-SVMs at the same time by optimizing their joint performance *after* the max. Our method finds a threshold for each E-SVM, so that (i) all positive windows are scored positively by at least one E-SVM, and (ii) the number of negative windows scored positively by any E-SVM is minimized. The first criterion ensures that there are no positive windows scored negatively after the max, while the second criterion minimizes the number of false positives.

We formalize these two criteria in a well-defined constrained optimization problem. The first requirement is formalised in its constraints, while the second comes in as a loss function to be minimized. Each threshold defines which training samples the respective E-SVM is scoring positively. By lowering a threshold we cover more positives and thereby satisfy more constraints, but we also include more negatives and therefore suffer a greater loss. Any positive sample can be potentially covered by any E-SVM, but at a different loss. This combinatorial nature of the problem makes it difficult to find the global optimum. We propose an efficient, globally optimal optimization technique. By exploiting the structure of the problem we are able to identify areas of the solution space that cannot contain the optimal solution and discard them early on. Our globally optimal algorithm is able to calibrate a few hundred E-SVMs quickly. In order to solve larger problems with thousands of E-SVMs, we present a simple modification of our exact algorithm to deliver high quality approximate solutions.

The rest of the paper is organized as follows. We start by reviewing related work in sec. 2. Sec. 3 introduces the formulation of our optimization problem, while sec. 4 presents our algorithm for efficiently finding the global optimal solution as well as its approximation. We train EE-SVM on state-of-the-art CNN descriptors [23] and present experiments on 10 classes of the ILSVRC 2014 dataset [13] and

on all 20 classes of PASCAL VOC 2007 [14] in sec. 5. These experiments show that (i) our joint calibration procedure outperforms standard independent sigmoid calibration [1] on the task of classifying windows as belonging to an object class or not; and (ii) this translates to better object detection performance. Finally, we conclude in sec. 6.

2. Related Work

In the machine learning literature, classifier calibration has been considered in the context of deriving probabilistic output for binary classifiers [15–18] or multi-class classification [10, 19]. Multi-class problems are often cast as a series of binary problems (e.g. 1-vs-all) and [1, 19, 20] showed that calibrating these binary classifier often leads to improved prediction.

The two most popular methods for calibrating binary classifiers are Platt scaling [15] and isotonic regression [16]. They both fit a monotonic function of the classifier score to the empirical label probability, obtaining an estimate of the conditional probability of a class label given the score. Platt scaling [15] uses a simple sigmoid function, while [16] employs a more flexible isotonic regression. In computer vision, Platt scaling is the most popular calibration tool [1, 5, 21]. We compare to both methods in sec. 5.3.

All these works [15–18] assume that the set of positive training samples for each classifier is fixed and given beforehand, even if small. In contrast, in the EE-SVM model, any positive sample can potentially be associated with any E-SVM. In the original EE-SVM paper [1] this was resolved in a greedy fashion, where each E-SVM was calibrated independently. The association of a positive sample to an E-SVM was resolved by comparing its uncalibrated E-SVM score to a fixed threshold. Instead, we calibrate E-SVMs *and* associate positive samples with them jointly over all positives and all E-SVMs. Our joint formulation (sec. 3) ensures that every positive is associated with at least one E-SVM, while the total number of false positives is minimized. As an additional benefit, this enables removing up to 25% of redundant E-SVMs that are not associated with any positives after the global optimum is found.

Two interesting exceptions to the classic EE-SVM calibration procedure [1] were presented recently [10, 12]. Gronat *et al*. [10] learns a per-location classifier for visual place recognition, while [12] learns exemplar-based 3D "chair" representations. Both works employ a calibration strategy based purely on negative samples, sidestepping the association of positive samples to E-SVMs. For completeness, we compare to [12] in sec. 5.3. All techniques reviewed above calibrate each E-SVM independently.

3. Joint calibration formulation

In many object detection pipelines [22–24] a single linear classifier $w \in R^d$ is applied to all K candidate windows

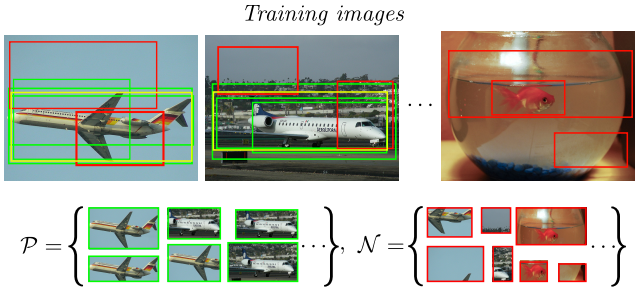


Figure 1: *Example of window proposals used in our calibration technique. $\mathcal{P}$ is the set of positive windows (□) and $\mathcal{N}$ is the set of negative windows (□) in the training set. Finally, (□) indicates E-SVMs ground-truth bounding-box. A window is positive if it has an intersection-over-union ≥ 0.5 with a ground-truth box [14].*

$\{x\}_{i=1}^{K}$ in an image, where $x \in R^d$ is the window descriptor. The windows are then ranked according to the classifier score $w \cdot x$. An EE-SVM, instead, contains E classifiers: $\{w_j\}_{j=1}^{E}$. The score of a window x is defined as the highest score among all classifiers applied to it:

$$S(w) = \max_j (w_j \cdot x) \tag{1}$$

Our goal is to find a threshold θ_j for each E-SVM e_j such that (i) all positive windows are scored positively by at least one E-SVM, and (ii) the number of negative windows scored positively by any E-SVM is minimized. A window x is scored positively by E-SVM e_j if $w_j \cdot x - \theta_j > 0$. We formalize these criteria in an optimization problem:

$$\min_{\boldsymbol{\Theta}=\{\theta_j\}_{j=1}^{E}} \overbrace{\sum_{x \in \mathcal{N}} \mathbb{1}[\max_j (w_j \cdot x - \theta_j)]}^{\mathcal{L}(\boldsymbol{\Theta})} \quad s.t.\ \mathbb{1}[\max_j (w_j \cdot x - \theta_j)] > 0,\ \forall x \in \mathcal{P} \tag{2}$$

where $\mathbb{1}$ is the indicator function and $\mathcal{P}$ and $\mathcal{N}$ are the sets of positive and negative windows in the training set (fig. 1). We refer to the top term as the *loss function* $\mathcal{L}(\boldsymbol{\Theta})$ and the bottom terms as the *constraints*.

Calibration is performed by adjusting the thresholds $\boldsymbol{\Theta}$. Given a configuration of thresholds $\boldsymbol{\Theta} = [\theta_1, \theta_2, \dots \theta_E]$, the loss $\mathcal{L}(\boldsymbol{\Theta})$ counts the number of negative windows scored positively after the max operation. Each constraint i ensures that a positive window x_i is scored positively by at least one E-SVM. We refer to a configuration $\boldsymbol{\Theta}$ satisfying all the constraints as a *feasible solution*.

4. Globally optimal and efficient solution

In this section we develop a computationally efficient algorithm to find the global optimal solution of (2). We start in sec. 4.1 by analysing the space of all possible solutions of (2). In sec. 4.2 we then introduce a data structure to represent this space, and finally in sec. 4.3 we present an efficient algorithm to search this data structure for the globally optimal solution.

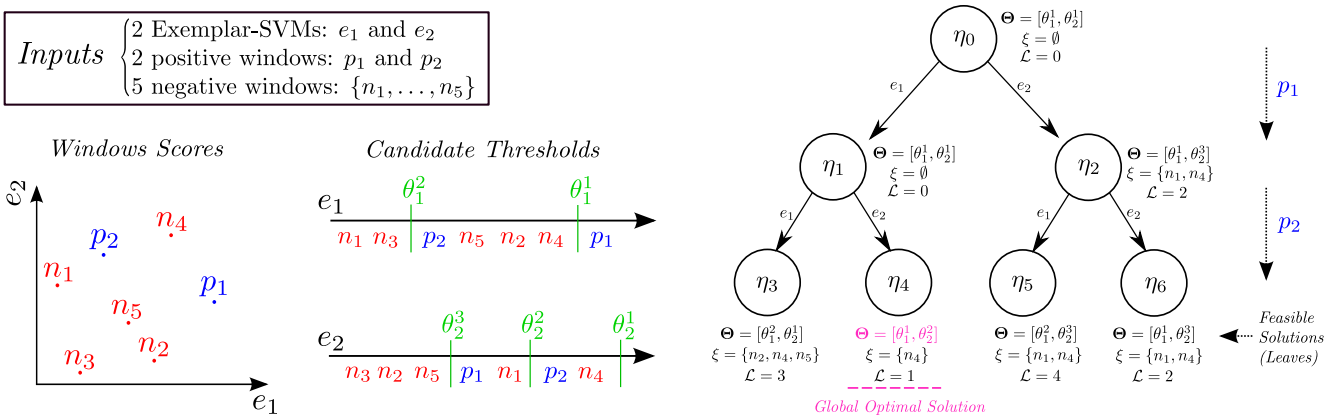


Figure 2: *Illustration of our joint calibration algorithm.* ***(left)*** *shows the window scores of the two E-SVMs e_1 and e_2.* ***(middle)*** *shows the candidate thresholds for these two E-SVMs. These are $[\theta_1^1, \theta_1^2]$ and $[\theta_2^1, \theta_2^2, \theta_3^3]$, respectively. Finally,* ***(right)*** *shows the tree representing the space of all possible solutions. Note how the only feasible threshold configurations are those in the leaves.*

4.1. Space of candidate thresholds

At first sight, (2) appears to be a continuous optimization problem where each threshold can take any value in $\mathbb{R}$. However, since E-SVMs are evaluated only on a finite set of training windows, there exist an infinite set of *equivalent* thresholds leading to the same loss. For this reason, (2) is in practice a discrete optimization problem.

Fig. 3 shows an example. Since each constraint in (2) evaluates one positive windows, an E-SVM needs at most $P+1$ thresholds to satisfy each of them (fig. 2), where $P = |\mathcal{P}|$. Furthermore, considering a threshold between two positive samples is not necessary, because the loss only changes when new negative samples are scored positively. The only thresholds worth considering are those between the score of a negative sample and a positive (not the reverse, fig. 3). We denote the set of candidate thresholds for an E-SVM e_j as $[\theta_j^1, \theta_j^2, \dots \theta_j^{M_j}]$, where $1 \leq M_j \leq P+1$ and $\theta_j^a < \theta_j^b$, for $\forall a, b : 1 \leq a < b \leq M_j$. By construction, the lowest threshold $\theta_j^{M_j}$ satisfy all the constraints in (2).

To conclude, the number of candidate thresholds for an individual E-SVM is relatively small (at most $P+1$), but the joint space of E-SVMs thresholds is nonetheless huge. In the worse case scenario (all E-SVMs have P candidate thresholds), there are P^E threshold configurations, many of which are not a feasible solution to (2). In the next section we present a data structure to enumerate all these configurations and highlight the feasible solutions.

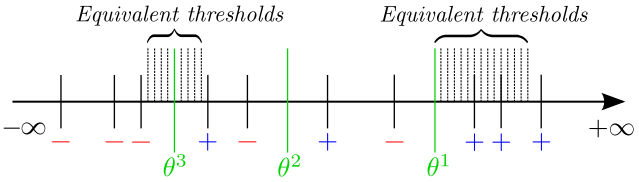


Figure 3: *Candidate thresholds, given the scores on positive (+) and negative (-) windows. The only thresholds worth considering according to (2) are the ones between a negative and positive window. Of all equivalent thresholds between two window scores, we consider only the mean of the two scores.*

4.2. Exhaustive search tree

We represent the space of all possible solutions as a search tree (fig. 2).

Definition 1. (Search Tree) *Our search tree T is a perfect k-ary tree: a rooted tree where every internal node has exactly k children and all leaves are at the same depth h. Each node η contains a configuration $\boldsymbol{\Theta} = [\theta_1, \theta_2, \dots \theta_E]$ of thresholds.*

A configuration $\boldsymbol{\Theta}$ at node η is used to compute the loss $\mathcal{L}(\boldsymbol{\Theta})$ by counting how many negative windows are scored positively according to (2). The root node has the configuration $\boldsymbol{\Theta} = [\theta_1^1, \theta_2^1, \dots \theta_E^1]$ of all tightest threshold for each E-SVM. We denote the set of false positives at a node η as ξ_η and $\mathcal{L}(\boldsymbol{\Theta}_\eta) = |\xi_\eta|$. Note how the root has $\xi = \emptyset$.

In our representation we have $h = P$ and $k = E$. Each level l of the tree corresponds to a positive window p_l, and each edge corresponds to an E-SVM e_j. An edge e_j indicates that e_j is responsible for p_l and it should score it positively, hence satisfying one constraint of (2). Given an E-SVM e_j and its current threshold θ_j, the edge lowers the threshold so that $w_j \cdot x_l - \theta_j > 0$ (if this condition is already satisfied then the threshold does not change). Lowering the threshold *might* increase the loss, but not necessarily. Lowering the threshold will make E-SVM e_j score positively some negative windows, but these affect the loss only if they were not already scored positively by another E-SVM.

The deeper the level, the more constraints of (2) are satisfied. By construction, the configuration of thresholds at a leaf satisfy all constraints, and the set of all leaves represent the set of all feasible solutions. Also note that the number of false positives always increases (or remains the same) along a path from the root to a leaf: given a node η and any child η', we have that $\mathcal{L}(\boldsymbol{\Theta}_\eta) \leq \mathcal{L}(\boldsymbol{\Theta}_{\eta'})$.

4.3. Efficient search

In this section we find the global optimal solution to (2) by searching the tree. Exhaustive search is computationally prohibitive even for small problems with few E-SVMs and positive windows, as the total number of nodes in our tree is $(E^{P+1}-1)/(E-1)$ (with E the number of E-SVMs and P the number of positive windows).

The key to our efficient algorithm is to prune the tree by iteratively removing subtrees which cannot contain the global optimal solution. In the following paragraphs we present several observations that enable to drastically reduce the space of solutions to consider. The last paragraph presents the actual search algorithm.

Observation 1. (Pruning by bound). *If η is a leaf and η' is a node not on the path from the root to η, and $\mathcal{L}(\mathbf{\Theta}_\eta) \leq \mathcal{L}(\mathbf{\Theta}_{\eta'})$, then the subtree rooted at η' cannot contain a better solution than η and can be discarded.*

The key intuition is that the loss can only increase with depth. The observation leads to two cases. First, if $\mathcal{L}(\mathbf{\Theta}_\eta) < \mathcal{L}(\mathbf{\Theta}_{\eta'})$, then η' cannot lead to an optimal solution since its loss is already higher than an already found feasible solution. Second, if $\mathcal{L}(\mathbf{\Theta}_\eta) = \mathcal{L}(\mathbf{\Theta}_{\eta'})$, η' could lead to an optimal solution, but it would be equivalent to the one in η. In both cases we can discard the subtree rooted at η', as we are interested in finding only one optimal solution.

Consider the example in fig. 4. Since $\mathcal{L}(\mathbf{\Theta}_{\eta_3}) = 5 < \mathcal{L}(\mathbf{\Theta}_{\eta_2}) = 6$, the subtree rooted at η_2 cannot contain an optimal solution: any solution in it has a loss ≥ 6, which is higher than the feasible solution in η_3.

Observation 2. (Pruning by equivalence). *If two nodes η and η' have the same parent and $\xi_\eta = \xi_{\eta'}$, then they are equivalent and only one subtree needs be searched.*

The key intuition is that the loss in (2) only increases when *new* negative windows are scored positively. Consider the example in fig. 5, where $\xi_{\eta_1} = \xi_{\eta_2}$ and η_1 and η_2 have the same parent (i.e., they satisfy the same constraints of (2)). η_1 has $\mathbf{\Theta} = [\theta_1^2, \theta_2^1]$ because the edge from η_0 to η_1 adjusted the threshold of e_1. Note, however, that this configuration can be changed into $[\theta_1^2, \theta_2^2]$ without increasing the loss, as they are equivalent solutions. Because of this equivalence, both subtrees lead to equivalent feasible solutions and only one of them needs be searched.

Observation 3. (Reducing tree depth). *Given the root configuration $\mathbf{\Theta} = [\theta_1^1, \theta_2^1, \dots \theta_E^1]$, there might exist some $x \in \mathcal{P} : \max_j(w_j \cdot x - \theta_j^1) > 0$. Since $\mathbf{\Theta}$ already satisfies the constraint for these positives at zero cost, these can be eliminated right away, reducing the depth of the tree.*

Consider the example in fig. 2. Initially, $\mathbf{\Theta} = [\theta_1^1, \theta_2^1]$. This configuration already scores p_1 positively. Whatever optimal solution $\mathbf{\Theta}$ the tree retrieves, p_1 will always be scored positively by at least E-SVM e_1. Hence, we can eliminate it from the tree to reduce its depth.

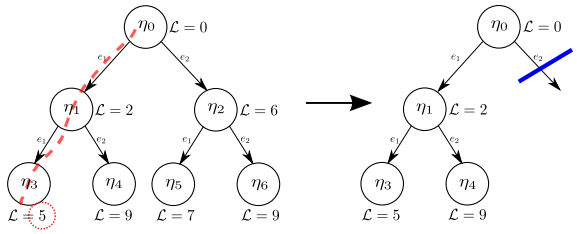


Figure 4: *Example of pruning by bound. Since $\mathcal{L}(\mathbf{\Theta}_{\eta_3}) = 5 < \mathcal{L}(\mathbf{\Theta}_{\eta_2}) = 6$, the subtree rooted at η_2 cannot contain an optimal solution.*

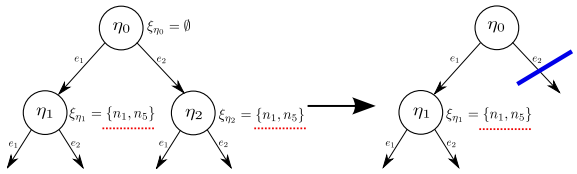
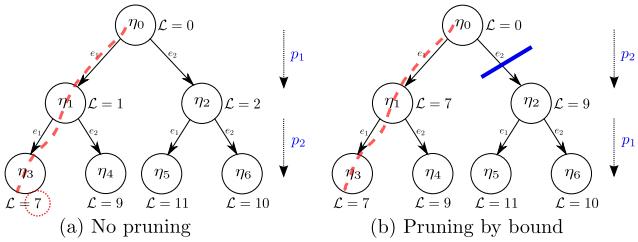


Figure 5: *Example of pruning by equivalence. Since the two nodes η_1 and η_2 have the same parent and $\xi_{\eta_1} = \xi_{\eta_2}$, they are equivalent and only one subtree needs be searched.*



(a) No pruning (b) Pruning by bound

Figure 6: *Efficient pruning by bound can be achieved by sorting the positive windows by decreasing difficulty.*

Observation 4. (Order of positive windows). *By sorting the positive windows by decreasing difficulty, pruning by bound can discard larger subtrees. The difficulty of a positive window x is measured as* $\min_j \delta(e_j, x)$.

where $\delta(e_j, x)$ counts how many false positives e_j produces by scoring the positive sample $x \in \mathcal{P}$ positively.

The key intuition is that it is better to prune subtrees rooted at the higher levels of the tree, as they contain more nodes. This can be achieved by placing difficult positive windows up in the higher levels. Some positive windows lead to constraints intrinsically more difficult to satisfy than others, as any E-SVM asked to satisfy it would score positively many negatives as well, and hence incur a large loss. By sorting the tree levels according to the difficulty of positive windows, it is likely that the loss of many high-level configurations is higher than a previously found feasible solution, and therefore can be pruned (observation 1).

Consider the example in fig. 6. Tree (a) evaluates first p_1

and then p_2, while (b) does the opposite. Tree (a) cannot be pruned by observation 1, but tree (b) can.

In sec. 5.5 we evaluate experimentally how effective the above pruning techniques are on various real EE-SVM calibration problems.

Search algorithm. We present here a depth-first search algorithm to efficiently find the global optimal solution, based on the above observations (Algo. 1).

The algorithm works as follows. The initial configuration of thresholds $\mathbf{\Theta}$ is the one from the root node *(line 1)*. As preprocessing, the algorithm starts by reducing the tree depth using observation 3 *(line 2)* and re-ordering the positive windows using observation 4 *(line 3)*. In the first step, it does a depth-first search until it reaches a leaf η and finds a first feasible solution $\mathbf{\Theta}$ *(line 4)*. During this traversal, when going down a level, the algorithm always chooses the edge leading to the smallest loss. Next, the algorithm continues by going up *(line 6)* and down *(line 10)* the tree. When visiting a node, the algorithm tries to prune as many children subtrees as it can using observation 1 *(line 8)* and observation 2 *(line 9)*. When the algorithm reaches a leaf then this must contain a better solution than the current one $\mathbf{\Theta}$ (in term of the loss (2)). Hence, it updates $\mathbf{\Theta}$ *(line 13)*. The algorithm continues until all nodes have been visited or pruned. The final $\mathbf{\Theta}$ is the global optimum of (2).

Algorithm 1 Our Efficient Search Algorithm

Input: search tree T
Output: global optimal $\mathbf{\Theta}$

1: $\mathbf{\Theta} \leftarrow [\theta_1^1, \theta_2^1, \dots \theta_E^1]$
2: $T \leftarrow$ REDUCE TREE DEPTH(T, $\mathbf{\Theta}$)
3: $T \leftarrow$ REORDER POSITIVE WINDOWS(T, $\mathbf{\Theta}$)
4: η, $\mathbf{\Theta}$, $\mathcal{L} \leftarrow$ DEPTH FIRST SEARCH(T, $\mathbf{\Theta}$)
5: **while** T not fully searched **do**
6: $\quad\eta \leftarrow$ GO UP ONE LEVEL (T, η)
7: $\quad$**while** $\neg$ ISLEAF(η) $\wedge$ $\neg$ HASCHILD(η) **do**
8: $\quad\quad T \leftarrow$ PRUNE BY BOUND $(T, \eta, \mathcal{L})$
9: $\quad\quad T \leftarrow$ PRUNE BY EQUIVALENCE $(T, \eta, \mathbf{\Theta})$
10: $\quad\quad \eta \leftarrow$ GO DOWN ONE LEVEL (T, η)
11: $\quad$**end while**
12: $\quad$**if** ISLEAF(η) **then**
13: $\quad\quad \mathbf{\Theta}, \mathcal{L} \leftarrow$ GET SOLUTION(η)
14: $\quad$**end if**
15: **end while**
16: return $\mathbf{\Theta}$

4.4. Approximate search

Above we presented an efficient algorithm that guarantees global optimality. If we relax the global optimality requirement, we can improve efficiency even further. Our method follows a depth-first search and as soon as it reaches a leaf, then it finds a feasible solution. This happens periodically during the execution of the algorithm, as better and better leaves are found while the tree is searched. This behaviour makes our method an *any-time* algorithm [25, 26]. After a short period required to reach the first leaf, we can terminate it at any time and it will return the best feasible solution it has found so far (although not necessarily the globally optimal one). This simple observation enables to employ our method, essentially unchanged, to find approximate solutions as well.

5. Experiments

5.1. Datasets

We present experiments on ILSVRC2014 [13] (sec. 5.3, 5.4) and PASCAL VOC 2007 [14] (sec. 5.4). ILSVRC2014 [13] contains 200 classes annotated by bounding-boxes. In our experiments we randomly sampled 10 classes: *airplane, bagel, baseball, bear, butterfly, koala, ladle, printer, sheep* and *violin*. Following [23] we consider three disjoint subsets of the data: `train`, $\texttt{val}_1$ and $\texttt{val}_2$. Since annotations for the test set are not released, we measure performance on $\texttt{val}_2$. We use $\texttt{val}_1$ and `train` for training. In total, these sets contain >80k images.

PASCAL VOC 2007 [14] contains 20 classes annotated by bounding-boxes. In our experiments we evaluate on all 20 classes. We use the subset `trainval` for training and we measure performance on `test`. In total, these sets contain about 10k images.

5.2. Settings

Object proposals and features. We generate class-independent object proposals using [27]. Given an image, this produces a small set of a few thousand windows likely to cover all objects. We then extract state-of-the-art CNN descriptors of 4096 dimensions for these proposals, as in [23]. These descriptors are the output of a convolutional neural network (CNN) initially trained for image classification [28, 29] and then fine-tuned for object detection [23] (on $\texttt{val}_1$ of ILSVRC2014, or on `trainval` of PASCAL VOC 2007).

EE-SVM. We learn a separate window classifier e for each instance of an object in the training set. We set $C = 10^{-4}$ and we mine hard negatives from 2000 random training images. In our experiments we observed that mining more images did not bring a significant improvement.

Calibration data. For each class we define $\mathcal{P}$ as the set of all positive training windows. A window is considered positive if it has intersection-over-union (IoU) [14] ≥ 0.5 with a ground-truth bounding-box of that class. Moreover, $\mathcal{N}$ contains negative windows that overlap ≤ 0.2. All calibration methods below are trained from this data.

ILSVRC 2014 - trained on $\texttt{val}_1$	Airplane	Bagel	Baseball	Bear	Butterfly	Koala	Ladle	Printer	Sheep	Violin	mean
Recall	*94.1*	*90.1*	*97.9*	*93.1*	*97.5*	*96.6*	*79.5*	*88.5*	*98.9*	*85.3*	*92.2*
EE-SVM no calibration	180124	171966	499056	33664	163727	80145	250602	221117	308458	37519	195k
EE-SVM indep. sigmoid calibration [1]	119552	42165	234734	77099	53986	13017	56616	88390	86507	33266	80k
EE-SVM joint calibration	65182	28460	180658	22694	53927	10746	87513	36573	55923	32570	57k
EE-SVM joint calibration w/ sigmoid	64996	28140	168129	22876	53867	10722	87329	35803	50064	33899	**55k**
EE-SVM indep. isotonic regression [19]	54173	23625	268302	16424	43580	13507	60285	55129	76997	13814	63k
EE-SVM indep. affine calibration [12]	51905	24507	224003	18978	37483	9947	67288	86757	110909	18218	65k
Single Linear-SVM (R-CNN) [23]	102676	335122	711185	109480	63849	305931	322332	469777	979131	121050	341k

Table 1: ***Window classification - False positives at test recall.*** *Results on a subset of ILSVRC 2014 $\texttt{val}_2$ (all positive windows and one million randomly sampled negative ones). We use the optimal thresholds found by our algorithm (sec. 4) to compute recall on $\texttt{val}_2$. This is the percentage of positive windows scored positively by our jointly calibrated EE-SVM. The table entries show the number of false positives produced in order to reach that recall level. Each row corresponds to a different method (ours are marked 'joint calibration').*

ILSVRC 2014 - trained on $\texttt{val}_1$	Airplane	Bagel	Baseball	Bear	Butterfly	Koala	Ladle	Printer	Sheep	Violin	mAP
EE-SVM indep. sigmoid calibration [1]	42.8	39.7	63.3	58.7	60.8	58.2	4.5	29.0	49.5	20.3	42.7
EE-SVM joint calibration w/ sigmoid	43.3	40.1	66.5	60.6	63.9	61.1	5.0	31.6	55.1	22.9	**45.0**
EE-SVM indep. isotonic regression [19]	44.6	42.4	61.4	59.3	63.8	63.2	5.7	22.5	50.9	23.2	43.7
EE-SVM Indep. affine calibration [12]	45.6	42.0	64.1	59.2	62.6	62.2	6.3	22.9	50.4	25.9	44.1
Single Linear-SVM (R-CNN) [23]	47.9	36.9	65.0	60.9	66.7	63.4	5.4	24.4	50.6	19.1	44.0

Table 2: ***Window classification - Average precision.*** *Results on a subset of ILSVRC 2014 $\texttt{val}_2$ (same data as table 1).*

Independent sigmoid calibration. As a baseline we compare against the standard technique of Malisiewicz *et al*. [1]. It operates in two steps. In step (1) it runs each E-SVM detector separately on a validation set, applies non-maximum suppression, and then eliminates all detections scoring below the -1 margin. All remaining detections are considered positives if they belong to $\mathcal{P}$, or negatives if they belong to $\mathcal{N}$. Note how this arbitrarily defines which positive training samples to associate with a certain E-SVM. In step (2), it then fits a logistic sigmoid to these data samples.

Our joint calibration. Our joint calibration also operates in two steps. In step (1), instead of arbitrarily defining the positive training samples, our technique use the thresholds found by our algorithm (sec. 4) to associate positive samples to E-SVMs, which is the core underlying problem at the heart of such calibration. More specifically, for an E-SVM e_j, we consider as positives all windows $x \in \mathcal{P} : w_j \cdot x > \theta_j$, and as negatives all windows in $\mathcal{N}$. Step (2) of our procedure is then the same as in [1], but thanks to these optimal assignments, we fit better sigmoids.

We experimentally evaluate performance after each step. We refer to the output of step (1) as *joint calibration*, and to the output of step (2) as *joint calibration with sigmoid*. As step (1) fits thresholds, it results in binary classification of test windows, while step (2) produces a continuous score which can be used for later processing stages (e.g. non-maximum suppression for object detection).

Other independent calibration techniques. For completeness, we also compare against two independent calibration techniques not commonly used for EE-SVM: isotonic regression [19] and the recent affine calibration [12]. *Isotonic regression* fits a piecewise-constant non-decreasing function to the output of each E-SVM. We used the code of [30] to train the function parameters on $\mathcal{P}$ and $\mathcal{N}$. *Affine calibration* fits an affine transformation to the output of each E-SVM. As in [12], we train the affine parameters on 200k randomly sampled negative windows from $\mathcal{N}$.

Single Linear-SVM (R-CNN). Finally, we provide results for the state-of-the-art R-CNN object detection model [23]. The sole purpose is to compare performance to EE-SVM on CNN features, as previous EE-SVM works typically use weaker HOG features [1–7, 9, 11]. However, as it consists of a single linear SVM per class, R-CNN cannot associate training exemples to objects detected in test images. Hence, it is not suitable for annotation transfer. We trained the model using the code and parameters of [23]. Note how this uses the same object proposals and features as our EE-SVM models.

5.3. Globally optimal joint calibration

We evaluate here our globally optimal joint calibration technique on ILSVRC2014 [13]. We train E-SVMs on $\texttt{val}_1$ for the 10 classes listed in sec. 5.1. Each class has between 30 and 140 E-SVMs and between 500 and 3600 positive windows $\mathcal{P}$. We evaluate two tasks: window classification and object detection.

Window classification

For this experiment, we use all positive windows ($IoU \geq 0.5$) in the test set $\texttt{val}_2$ and 1 million randomly sampled negative ones ($IoU < 0.5$). We evaluate window classification in terms of two measures: false positives at test recall, and average precision.

False positives at test recall. This measure counts how many false positives are produced on the test set, at the recall point produced by the ensemble of E-SVMs calibrated by our method. Note this is exactly what our calibration procedure optimizes for. Given the thresholds $\mathbf{\Theta}$ output by

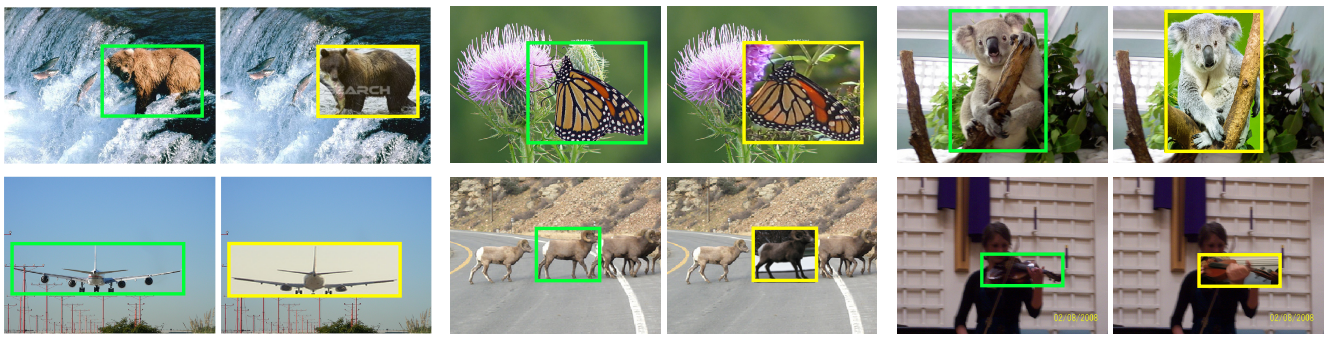


Figure 7: ***Association between detected objects and training exemplars.*** *Our globally optimal joint calibration is good at transferring annotations from exemplars onto test windows. In these figure we show detections (green) and their associated training exemplar superimposed on them (yellow).*

ILSVRC 2014 - trained on $\texttt{Val}_1$	Airplane	Bagel	Baseball	Bear	Butterfly	Koala	Ladle	Printer	Sheep	Violin	mAP
EE-SVM indep. sigmoid calibration [1]	43.3	11.9	27.2	45.2	51.1	46.3	0.6	8.4	31.4	7.4	27.3
EE-SVM joint calibration w/ sigmoid	46.2	10.1	41.3	44.7	66.8	41.4	1.0	10.8	34.3	9.5	30.6
EE-SVM indep. isotonic regression [19]	34.9	13.0	31.4	36.1	59.2	48.6	0.6	13.0	31.0	3.8	27.2
EE-SVM indep. affine calibration [12]	45.8	10.3	41.0	44.1	66.1	39.5	0.8	11.4	35.9	9.6	30.5
Single Linear-SVM (R-CNN) [23]	49.0	17.4	45.4	53.3	69.6	61.4	2.8	18.9	41.5	11.0	37.0

Table 3: ***Object detection - Average precision.*** *Results on ILSVRC 2014* $\texttt{Val}_2$.

our algorithm (sec. 4), we compute recall as the percentage of positive windows scored positively by the ensemble on the test set (top row of table 1). Interestingly, the thresholds generalize well to test data and lead to high recall on almost all classes.

We compare several methods at this recall point. The main four are EE-SVM with no calibration, EE-SVM with independent sigmoid calibration [1], our joint calibration fitting thresholds and our joint calibration with sigmoid (table 1, rows 2-5). As expected, EE-SVM with no calibration performs very poorly and some form of calibration is necessary. Our joint calibration method considerably outperforms independent calibration, and the version with sigmoid brings another small boost in performance. These results demonstrate the benefit of our joint calibration, that takes into account the max operation of the EE-SVM. Given these results, we omit EE-SVM with no calibration from further analysis. Table 1 also presents results for isotonic regression, affine calibration and R-CNN. On average, our joint calibration outperforms all these methods, albeit by a smaller margin.

Average precision. In table 2 we compare techniques in terms of average precision. As this measure requires a continuous score of test windows, we only consider our joint calibration with sigmoid. Joint calibration outperforms independent sigmoid calibration on all classes, and improves mAP by 2.3%. This further highlights the benefits of joint calibration, in a scenario that is not exactly what it was optimized for. Table 2 also presents results for isotonic regression, affine calibration and R-CNN. On average, joint calibration performs better than all these methods, abeit by a modest margin (about +1% mAP).

Object detection

We evaluate our joint calibration method against independent sigmoid calibration on the task of object detection. Note how this task adds a layer of non-maximum suppression (NMS) to the pipeline. As our calibration procedure does not take into account NMS, it is not obvious that the benefit seen so far on window classification will carry over to object detection. As table 3 shows, joint calibration outperforms independent sigmoid calibration on this task as well (+3.3% mAP). Joint calibration performs equally or better on all classes but *koala*. For some classes the improvement is substantial: +14% AP on *baseball* and +14.7% AP on *butterfly*.

Furthermore, table 3 also presents results for isotonic regression, affine calibration and R-CNN. Isotonic regression performs comparably to independent sigmoid calibration, whereas affine calibration delivers about the same mAP as our joint calibration. Interestingly, R-CNN does considerably better than all other methods, including our EE-SVM with joint calibration. This is somewhat surprising, as EE-SVM was shown much better than a single linear SVM on HOG features [1]. We attribute this phenomenon to the CNN features, which are more easily linearly separable [23, 31–33]. Besides, note that despite the high performance, R-CNN lacks the crucial ability of EE-SVM to associate training exemplars to objects detected in test images, and it is therefore not suitable for annotation transfer.

5.4. Approximate joint calibration

In this section we evaluate our approximate joint calibration technique (sec. 4.4). By relaxing global optimality, we can find a feasible solution even for large problems.

ILSVRC 2014 - trained on $\texttt{val}_1$+train	mean
Recall	85.2
EE-SVM no calibration	137k
EE-SVM indep. sigmoid calibration [1]	107k
EE-SVM joint calibration	82k
EE-SVM joint calibration w/ sigmoid	**54k**

Table 4: ***Window classification - False positives at test recall.*** *Results on a subset of ILSVRC 2014* $\texttt{val}_2$ *(mean over 10 classes). We used all positive windows and 2 million randomly sampled negative ones.*

ILSVRC 2014 - trained on $\texttt{val}_1$+train	mAP
EE-SVM indep. sigmoid calibration [1]	39.7
EE-SVM joint calibration w/ sigmoid	**42.9**

Table 5: ***Window classification - Average precision.*** *Results on a subset of ILSVRC 2014* $\texttt{val}_2$ *(mean over 10 classes, same data as table 4).*

ILSVRC2014. We experiment here by training and calibrating EE-SVM on the union of $\texttt{val}_1$ and `train`. This results in a large number of E-SVMs. Each class has between 640 and 2000 E-SVMs, and between 3000 and 13000 positive windows $\mathcal{P}$. We report results on the task of window classification averaged over the 10 classes. Note that these results cannot be compared to the ones in tables 1 and 2 because here we have larger training and test sets.

False positives at test recall. As table 4 shows, our approximate joint calibration procedure still achieves high recall while returning much fewer false positives than no calibration and independent sigmoid calibration. When adding a sigmoid our calibration improves even further by a good margin. This shows that our method provides an excellent association between positive windows and E-SVMs.

Average precision. Results are presented in table 5. Joint calibration improves over independent sigmoid calibration by +3.2% mAP.

PASCAL VOC 2007. In order to compare against the original EE-SVM of [1], we experiment here on the PASCAL VOC 2007 dataset. We train and calibrate EE-SVM on the `trainval` subset and evaluate them on `test`. We report results on object detection in terms of mAP over the 20 classes (table 6). We compare traditional EE-SVM on HOG features (19.8 mAP, as reported by [1]), independently calibrated EE-SVM on CNNs (40.8 mAP), and our joint calibration on the same features (42.7 mAP). These results highlight two points: (1) joint calibration improves over independent calibration by +2% mAP, confirming what observed on ILSVRC2014; (2) CNN features bring a huge improvement over HOG to EE-SVM models (doubling mAP in this case). This confirms recent findings [23] about the benefits of CNN features for object detection.

PASCAL VOC 2007 test	Feature	Calibration	mAP
EE-SVM	HOG	independent	19.8
	CNN	independent	40.8
	CNN	joint	42.7

Table 6: ***Object detection - Average precision.*** *Results on PASCAL VOC 2007* `test` *(mean over 20 classes). EE-SVM HOG results are from [1].*

5.5. Pruning statistics and runtimes

Pruning statistics. Here we experimentally evaluate how effective our pruning techniques of sec. 4.3 are. Observation 3 (*line 2*, Algo. 1) reduces the depth of the tree by 20%, on average. Observation 4 (*line 3*) improves pruning by bound immensely. In a small problem with 100 E-SVMs we tried ordering the positive windows $\mathcal{P}$ randomly. The algorithm took two hours to find the global solution. On the other hand, when sorting $\mathcal{P}$ according to observation 4, the algorithm found the same solution in about 2 minutes. Observations 1 and 2 also bring a substantial speed-up. After finding a first feasible solution (*line 1*), the algorithm (*lines 8,9*) prunes 40% of the nodes it visits on problems with 100 E-SVMs, and 70% on problems with 1000 E-SVMs.

Runtime. We measure runtimes on a 4-cores Intel Core i5 2.0GHz. Exhaustive search is extremely inefficient and takes 15h to find the globally optimal solution for a tiny problem with 15 E-SVMs and 50 positive windows. Our efficient and exact algorithm (sec. 4.3) finds the same solution in just a few seconds. This algorithm scales up to problems with about 200 E-SVMs and 4k positive windows in reasonable time (a few hours). For larger problems we rely on our approximate search algorithm (sec. 4.4). While we let it run for several hours, in most cases the loss stops decreasing significantly already after a few minutes.

6. Conclusion

We presented a method for calibrating the Ensemble of Exemplar SVMs model. While the standard approach calibrates each SVM independently, our method optimizes their joint performance as an ensemble. We formulated joint calibration as a constrained optimization problem and devised an efficient optimization algorithm to find its global optimum. In order to make the optimization computationally feasible, the algorithm dynamically discards parts of the solution space that cannot contain the optimum, by exploiting four observations about the structure of the problem.

We presented experiments on 10 classes from the ILSVRC 2014 dataset and 20 from PASCAL VOC 2007. Our joint calibration procedure outperforms the classic independent sigmoid calibration by a considerable margin on the task of classifying windows as belonging to an object class or not. On object detection, this better window classifier leads to an improvement of about 3% mAP.

Acknowledgement. We gratefully acknowledges support by SNSF fellowship PBEZP-2142889 and ERC Starting Grant VisCul.

References

[1] T. Malisiewicz, A. Gupta, and A. A. Efros, "Ensemble of exemplar-svms for object detection and beyond," in *ICCV*, 2011.

[2] A. Shrivastava, T. Malisiewicz, A. Gupta, and A. A. Efros, "Data-driven visual similarity for cross-domain image matching," in *SIGGRAPH Asia Conference*, 2011.

[3] S. Singh, A. Gupta, and A. A. Efros, "Unsupervised discovery of mid-level discriminative patches," in *ECCV*, 2012.

[4] Y. Aytar and A. Zisserman, "Enhancing exemplar svms using part level transfer regularization," in *BMVC*, 2012.

[5] I. Endres, K. J. Shih, J. Jiaa, and D. Hoiem, "Learning collections of part models for object recognition," in *CVPR*, 2013.

[6] J. Dong, W. Xia, Q. Chen, J. Feng, Z. Huang, and S. Yan, "Subcategory-aware object classification," in *CVPR*, 2013.

[7] J. Tighe and S. Lazebnik, "Finding things: Image parsing with regions and per-exemplar detectors," in *CVPR*, 2013.

[8] S. Song and J. Xiao, "Sliding shapes for 3d object detection in depth images," in *ECCV*, 2014.

[9] M. Juneja, A. Vedaldi, C. Jawahar, and A. Zisserman, "Blocks that shout: Distinctive parts for scene classification," in *CVPR*, 2013.

[10] P. Gronat, G. Obozinski, J. Sivic, and T. Pajdla, "Learning and calibrating per-location classifiers for visual place recognition," in *CVPR*, 2013.

[11] A. Vezhnevets and V. Ferrari, "Associative embeddings for large-scale knowledge transfer with self-assessment," in *CVPR*, 2014.

[12] M. Aubry, D. Maturana, A. A. Efros, B. C. Russell, and J. Sivic, "Seeing 3d chairs: exemplar part-based 2d-3d alignment using a large dataset of cad models," in *CVPR*, 2014.

[13] "Imagenet large scale visual recognition challenge (ILSVRC)." http://www.image-net.org/challenges/LSVRC/2014/index, 2014.

[14] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *IJCV*, 2010.

[15] J. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," *Advances in large margin classifiers*, 1999.

[16] B. Zadrozny and C. Elkan, "Learning and making decisions when costs and probabilities are both unknown," in *ACM SIGKDD*, 2001.

[17] B. Zadrozny and C. Elkan, "Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers," in *ICML*, 2001.

[18] A. Niculescu-Mizil and R. Caruana, "Predicting good probabilities with supervised learning," in *ICML*, 2005.

[19] B. Zadrozny and C. Elkan, "Transforming classifier scores into accurate multiclass probability estimates," in *ACM SIGKDD*, 2002.

[20] M. Aubry, B. C. Russell, and J. Sivic, "Painting-to-3d model alignment via discriminative visual elements," *ACM TOG*, vol. 33, no. 2, p. 14, 2014.

[21] D. Hoiem, A. A. Efros, and M. Hebert, "Putting objects in perspective," *IJCV*, vol. 80, no. 1, pp. 3–15, 2008.

[22] N. Dalal and B. Triggs, "Histogram of Oriented Gradients for human detection," in *CVPR*, 2005.

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

[24] R. Cinbis, J. Verbeek, and C. Schmid, "Segmentation driven object detection with fisher vectors," in *ICCV*, 2013.

[25] S. Zilberstein, "Using anytime algorithms in intelligent systems," *AI magazine*, vol. 17, no. 3, p. 73, 1996.

[26] V. Gogate and R. Dechter, "A complete anytime algorithm for treewidth," in *AUAI*, 2004.

[27] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *IJCV*, vol. 104, no. 2, pp. 154–171, 2013.

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012.

[29] Y. Jia, "Caffe: An open source convolutional architecture for fast feature embedding." http://caffe.berkeleyvision.org/, 2013.

[30] L. Duembge, "Isotonic and concave regression." http://www.imsv.unibe.ch/content/staff/personalhomepages/duembgen/software/isotonicregression/index_eng.html, 2000.

[31] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition," *arXiv preprint arXiv:1310.1531*, 2013.

[32] A. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: An astounding baseline for recognition," in *DeepVision workshop at CVPR*, 2014.

[33] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional networks," in *BMVC*, 2014.