Experimental Evaluation of Multiplicative Kernel SVM Classifiers for Multi-Class Detection

Valentina Zadrija
Mireo d.d.
Zagreb, Croatia
Email: valentina.zadrija@mireo.hr

Siniša Šegvić
Faculty of Electrical Engineering and Computing
University of Zagreb
Zagreb, Croatia
Email: sinisa.segvic@fer.hr

***Abstract*—We consider the multi-class object detection approach based on a non-parametric multiplicative kernel, which provides both separation against backgrounds and feature sharing among foreground classes. The training is carried out through the SVM framework. According to the obtained support vectors, a set of linear detectors is constructed by plugging the foreground training samples into the multiplicative kernel. However, evaluating the complete set would be inefficient at runtime, which means that the number of detectors has to be reduced somehow. We propose to reduce that number in a novel way, by an appropriate detector selection procedure. The proposed detection approach has been evaluated on the Belgian traffic sign dataset. The experiments show that detector selection succeeds to reduce the number of detectors to the half of the number of object classes. We compare the obtained performance to the results of other detection approaches and discuss the properties of our approach.**

I. Introduction

A long-standing goal of computer vision has been to design a system capable of detecting various classes of objects in cluttered scenes. Traditionally, this task has been solved by building a dedicated detector for each class. This approach requires a significant number of examples per each class, which may not be available. In order to overcome the problem, additional partitioning into subclasses can be performed. However, the problem is that domain-based partitioning may not be optimal for the task. In case of multi-view object detection, it can also be time consuming and error prone. Therefore, it would be desirable to omit the manual partitioning stage and embed the process into the classifier itself.

Another interesting idea is to train a single classification function for all classes jointly. This approach may exploit feature sharing among classes in order to improve classification against backgrounds. The feature sharing offers great potential for: (i) improving the detection rate for classes with a low number of examples, and (ii) reducing the runtime computational complexity.

In this paper, we train a joint classification function with the multiplicative kernel as presented in [1]. However, in contrast to [1], where the authors aim to solve the detection and recognition problems at the same time, we focus on the task of detection. Once the object locations are known, object class can be determined for those locations only, thus alleviating the runtime complexity.

Multi-class detection and feature sharing is achieved by means of a non-parametric multiplicative kernel. The approach avoids the partitioning into subclasses by using the foreground training samples as class membership labels. More details are given in section III. After the training, we construct a set of linear detectors as described in section III-A. According to [1], each detector corresponds to a single foreground training sample, which makes a detector set extremely large and inefficient for the detection task. The contributions of our work are as follows: (i) we propose an efficient detector selection in order to identify a representative set of detectors out of the large initial pool as described in section III-B, (ii) we show the properties of the multiplicative kernel method and compare the results with other methods on a Belgian traffic sign dataset (BTSD) [2] as described in section IV.

II. Related Work

In recent years, a lot of work has been proposed in the area of multi-class object detection. However, the work presented in [3] achieved a significant breakthrough in the area. The authors consider multiple overlapping subsets of classes. The experiments have shown that a jointly trained classification function requires a significantly smaller number of features in order to achieve the same performance as independent detectors. More specifically, the number of features grows logarithmically with respect to the number of subclasses.

The approach presented in [4] employs a tree-based classifier structure called Cluster Boosted Tree (CBT) in order to solve the multiview detection problem. The tree structure is constructed automatically according to the weak learners selected by the boosting algorithm. The node splits are achieved by means of unsupervised clustering. Therefore, in contrast to [3], this approach does not require manual partitioning into classes, but it implies the hierarchical feature sharing.

The authors in [5] consider a classifier structure comprised out of several boosted classifiers. This approach also avoids manual partitioning into classes, but the classifiers do not share weak learners. Initially, the training data is partitioned randomly into subsets. At each round of training, the sample is (re)assigned to the subset corresponding to the classifier that has the highest probability of classifying that sample. The resulting classifiers are further transformed into decision trees, which reduce the average number of weak learner evaluations during classification.

The concept of feature sharing is also explored through shape-based hierarchical compositional models [6], [7]. Different object categories can share parts or an appearance. The

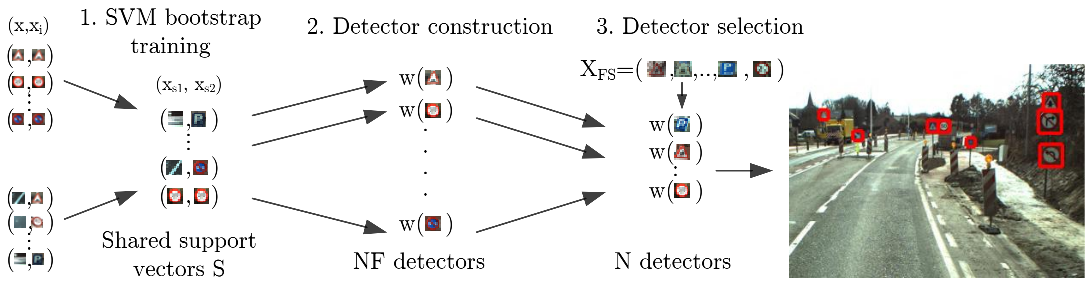


Fig. 1. An Overview of the multiplicative kernel detection pipeline. Note that we use HOG vectors instead of image patches.

parts on lower hierarchy levels are combined into larger parts on higher levels. In general, parts on lower levels are shared amongst various object categories, while those in higher levels are more specific in category.

In the more recent work [8], the authors employ Hough forest for multi-class detection. In contrast to the above methods, this approach uses the implicit shape model for detection rather than sliding window. The key feature of this approach is the separation between feature appearance and location. The appearance-based features are shared across classes providing a generalization against backgrounds. However, in order to become discriminative for individual classes, location needs to be fixed. Due to the feature sharing, the number of necessary votes grows sub-linearly with respect to the number of classes.

III. Methodology

We employ a non-parametric detection approach proposed in [1] as shown in steps one and two of Fig. 1. In the step three, we introduce a novel detector selection algorithm. The approach avoids the traditional partitioning of foreground object classes into subclasses. We treat the multi-class detection as a binary problem considering all foreground classes as a single class. Therefore, this method is also applicable in cases where the labels for foreground classes are not available. We obtain the required functionality by organizing the training samples into pairs (x, x_i) as shown in Fig. 1. The first element corresponds to the HOG vector of either foreground or background image patch. The second element x_i corresponds to the HOG vector of the foreground image patch, $i \in \{1..NF\}$, where *NF* denotes the number of foreground training samples. The purpose of x_i is to denote the membership to foreground classes. The idea behind this concept is that the descriptors belonging to the same foreground class or subclass share certain amount of similarity. However, rather than defining the partitioning by ourselves, we let the classification function to do that job. In this way, the foreground training pairs are actually duplicated foreground HOG descriptors. However, each background sample x can be associated with any foreground sample in order to form a valid negative pair. A total number of negative pairs is therefore huge and corresponds to $NF \cdot NB$, where *NB* denotes the number of background samples x.

The classification function $C(x, x_i)$ is therefore trained jointly for all classes and provides a following decision:

$$C(x, x_i) \begin{cases} > 0, \text{x is a sign from the same class as } x_i \\ \leq 0, \text{otherwise} \end{cases} \quad (1)$$

In order to provide the above functionality, the function $C(x, x_i)$ is defined as follows:

$$C(x, x_i) = \sum_{(x_{s1}, x_{s2}) \in S} \alpha_s \cdot k_i(x_{s2}, x_i) \cdot k_x(x_{s1}, x) \quad (2)$$

The pair $x_s = (x_{s1}, x_{s2})$ denotes a support vector, where x_{s1} corresponds to the HOG descriptor of either foreground or background sample, while x_{s2} denotes the assigned foreground training sample. Further, α_s denotes the Lagrange multiplier assigned to the support vector. The term $k_i(x_{s2}, x_i)$ denotes the *between-foreground* kernel. The purpose of this kernel is two-fold. Firstly, it is used to measure the similarity between foreground classes, thus enabling the feature sharing. Secondly, this kernel is also responsible for the foreground partitioning, i.e. it produces higher values for the similar pairs of foreground training samples. The $k_x(x_{s1}, x)$ term denotes the *foreground-background* kernel used for separation against backgrounds.

The classification function training can be achieved through the SVM framework. Due to the memory constraints, we cannot include all $NF \cdot NB$ negative pairs in the SVM training at once. Therefore, similar to [1] and [9], we also perform bootstrap training in order to identify the hard negatives. Initially, *NB* negative pairs are chosen randomly and SVM optimisation is performed. The obtained model is evaluated for all negative pairs and false positives are added to the negative sample set. The process converges when there are no more false positives to add.

A. Detector Construction

The individual detectors $w(x, x_i)$ are constructed from the support vectors S by plugging the specific foreground sample values x_i into (2). With the known value of a parameter x_i, the value of *between-foreground* kernel can be precomputed. We have chosen the non-linear Radial Basis Function (*RBF*) kernel for k_i. On the other hand, the *foreground-background* kernel k_x is evaluated at detection time, so it should be efficiently computed. Therefore, we have chosen the linear kernel $k_x = x^T \cdot x$ for that purpose. According to the chosen kernels, we obtain the following detector $w(x, x_i) = w(i) \cdot x$, where $w(i)$ denotes the vector of linear classifier weights:

$$w(i) = \sum_{(x_{s1}, x_{s2}) \in S} \alpha_s \cdot RBF(x_{s2}, x_i) \cdot x_{s1}^T \quad (3)$$

The feature sharing is achieved through support vectors *S*. The obtained number of the detectors corresponds to the number of

```
Given
 1: Foreground samples $X_{FS} = \{x_j\}$, $j \in \{1 \dots NS\}$
 2: Detectors $W = \{w(i)\}$, $i \in \{1 \dots NF\}$
Initialize
 3: Evaluate $W$ on $X_{FS}$: to each example $x_j$, assign a set of
    detectors $E(x_j)$ which detect that example
TRAVERSE($X_{FS}$, $W$):
 4: Find the example $x_m$ with the maximum nr. of detectors
 5: repeat
 6:     Find the detector $w(r) \in E(x_m)$ with the minimum
        nr. of detections
 7:     while
 8:         if removing $w(r)$ doesn't result with false negatives
 9:             Remove $w(r)$ from $W$, break
10:         else
11:             Find the next detector $w(r) \in E(x_m)$ with with
                minimum nr. of detections
12:     if $w(r)$ is not found
13:         Proceed to the next example $x_m$ with maximum
            nr. of detectors
14:     else
15:         Find the example $x_m$ with the maximum nr. of
            detectors
16: until all examples are traversed
17: return selected set of detectors $W$
```

Fig. 2. Detector selection algorithm.

foreground training samples *NF*. Evaluating all the detectors at runtime is related to the *k*-nearest neighbours (*k*-NN) method [10], with a parameter $k = 1$, i.e. the object is simply assigned to the class of the single nearest neighbour selected among all detectors. This approach would result in extremely slow detection. Therefore, the number of detectors needs to be reduced somehow. The authors in [1] propose clustering. On the other hand, we argue that the selection algorithm presented in the next section is a better approach.

B. Detector Selection

We apply detector selection according to the pseudo-code outlined in Fig. 2. The goal is to remove detectors which do not contribute to the overall detection score. The resulting set of detectors obtains the same recall as the original set of detectors on the selection samples. The set of samples X_{FS} is comprised out of foreground HOG vectors as shown in step three of Fig. 1. Note that selection and training datasets are disjoint. We traverse the selection examples according to the number of detectors which detect particular example. In each round, we consider the example with the maximum number of detectors (*line 4*). The algorithm is greedy, i.e. among the detectors that detect that example, we select the detector with the lowest number of detections (*line 6*) as a candidate for removal. The condition in *line 8* ensures that the recall remains unchanged by removing the detector. There is a possibility, that removing every detector from the $E(x_m)$ would result in a false negative (*line 12*). In that case we proceed to the next example with the lowest number of detectors (*line 13*). Due to the fact that the algorithm is greedy, it may get stuck in a local optimum. However, as the experiments in section IV show, it achieves a very high performance rate.

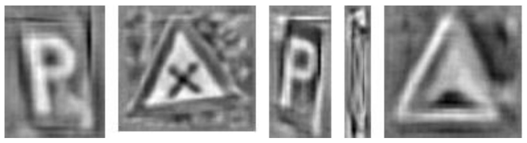


Fig. 3. Visualisation of the support vectors for the multiplicative kernel [12].

IV. CASE STUDY

As a case study, we apply the described methodology for traffic sign detection. Traffic signs have been chosen because they are characterized by a very large variation with respect to the sign shape and the type of the ideogram. We use a subset of the BTSD [2] containing 19 different classes shown in Fig. 4, Fig. 5 and Fig. 7. The selected classes capture a representative subset of traffic signs: triangular with a red rim, circular with a red rim and white or blue interior as well as the rectangular blue signs.

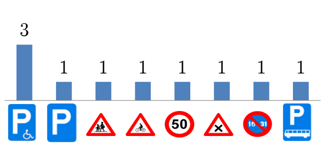
A. Implementation Details

There are 1024 sign images used for training and 393 sign images used for detector selection. There are also 11200 background training patches extracted from 16000 background images. The HOG vectors [11], computed from training images, are cropped to 24 × 24 pixels. We use 10-fold stratified cross-validation to determine the model parameters.

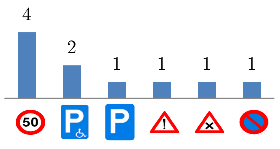
The test dataset contains 1027 images in 1628 × 1235 resolution. There are approximately 3 images per a single physical traffic sign. The training and test datasets are disjoint. We use the sliding window approach and scan the image using the 1.05 scale step. We also consider 5 aspect ratios, i.e. from ratio 1 to 3 with the step of 0.4 (height/width). We set the scale space search range from 24 × 24 to 330 × 435 pixels.

B. Results

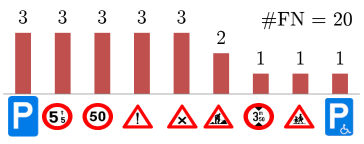
All experiments are performed on a 3.4 GHz Intel Core i7-3770 CPU. We perform the bounding box evaluation according to the scheme laid out in PASCAL object detection challenges [13]. In order to evaluate performance, we use the area under the precision-recall curve (*AuPR*) and area under the receiver operating characteristics curve (*AuROC*). As indicated in [14], the *AuPR* metric gives a more informative picture of a method's performance for highly skewed datasets. The sliding window approach employed in this work generates a large number of background candidate windows and only a small number of windows containing a foreground object. Therefore, the *AuPR* measure would seem to be the most relevant choice for the metric. However, as the results in the section IV-B1 show, an algorithm that optimizes the *AuPR* is not guaranteed to optimize the *AuROC* measure. More specifically, the method with the maximum *AuPR* may not be the optimal choice because of the very poor recall (detection rate). In addition, we also report the detection rate and false positive rate per image (*FP/img*). The rest of the section is organized as follows. We discuss the multiplicative kernel results in section IV-B1. Further, we compare these results with other approaches in section IV-B2.



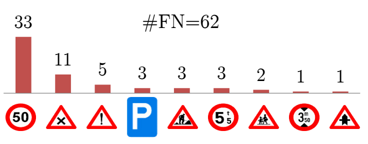
(a) Distribution of selected detectors for *Mult. Selected*



(b) Distribution of detectors for *Mult. Clustered*

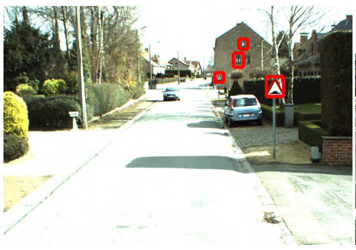


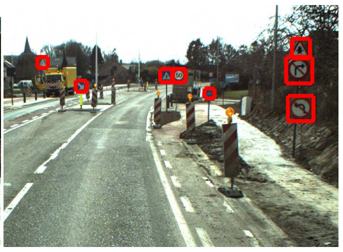
(c) Distribution of false negatives for *Mult. Selected*

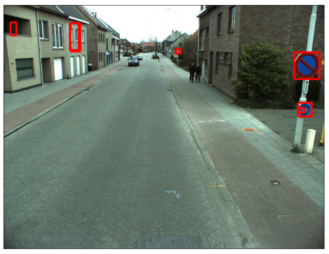


(d) Distribution of false negatives for *Mult. Clustered*

Fig. 4. A comparison of detector selection (*Mult. Selected*) and clustering (*Mult. Clustered*) for multiplicative kernel.


(a) Speed bump


(b) Right of the way at the next crossroads


(c) No parking allowed

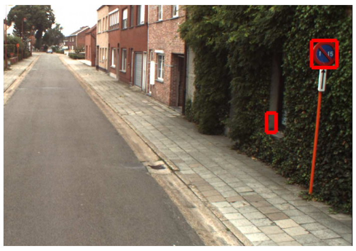

(d) No parking from the 1st to the 15th of the month

Fig. 5. Example of detections for traffic signs without an assigned detector for *Mult. Selected* method. All instances of these signs are detected in test images.

1) Multiplicative Kernel Results: The SVM training described in section III yields a shared set of support vectors containing 653 vectors out of which 23% corresponds to the positive training pairs. Fig. 3 shows the examples of support vectors obtained from the original image patches according to the procedure described in [12]. The support vectors correspond to the image patches containing the traffic signs captured under difficult conditions, i.e. rotated signs and hard perspective views. According to the obtained support vector set, 1024 linear detectors are constructed. The number of detectors is then reduced using the following methods:

a) Multiplicative Selected: This method employs the detector selection algorithm as described in section III-B.

b) Multiplicative Clustered: This method employs the clustering as proposed in [1]. As a clustering method, k-medoids with Euclidean distance is used. The purpose is to determine if the detector selection is really necessary or the problem can be solved with clustering.

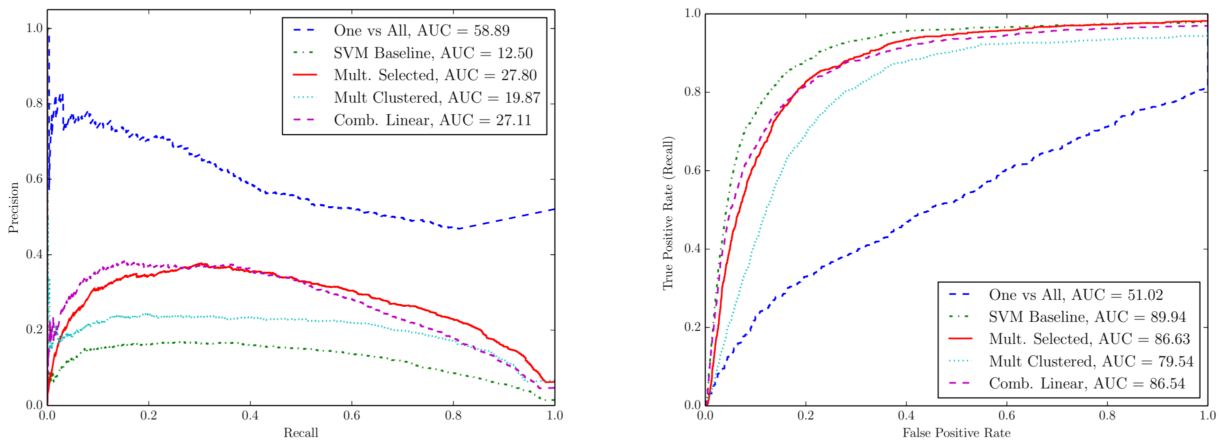
The results shown in Fig. 4 and Fig. 5 point out three important consequences which indicate that the *Multiplicative Selected* outperforms the *Multiplicative Clustered* approach.

Firstly, the selection approach converges with the number of detectors (10) which is almost 50% percent smaller than the number of classes (19). This suggests sub-linear detection complexity. The target number of cluster centres was then configured with the same value in order to provide a fair comparison. The distribution in Fig. 4a points out that the *Parking reserved for disabled people* is the most difficult sign for detection using the *Multiplicative Selected* approach. Hence, there are three detectors assigned to detect that sign. In general, rectangular signs are assigned five detectors, triangular three and circular only two signs. This would suggest that circular signs benefit the most from feature sharing. This is quite different from the detectors obtained by clustering, where the circular signs exhibit the maximum number of detectors (5) as shown in Fig. 4b.

Secondly, the selection approach yields a better set of detectors than the clustering. The false negative distribution shown in Fig. 4c and Fig. 4d supports that fact. The number of false negatives obtained by the *Multiplicative Clustered* method (62) is 3.1 times larger than the one obtained using the *Multiplicative Selected* (20). In addition, despite the fact that the *Speed limit* is assigned the maximum number of cluster centres in *Multiplicative Clustered* method, this particular sign exhibits the maximum number of false negatives (33). On the other hand, selection approach yields only one detector for the *Speed limit* class. However, almost all *Speed limit* signs are detected as shown on Fig. 4c. This suggests that the detectors obtained by clustering are not optimal for a specific class and that the selection is a better option. Further analysis of the false negatives for the *Multiplicative Selected* method shows interesting results. One would expect that most of false negatives belong to the classes which do not have a corresponding detector. However, this is not entirely true, i.e. 9 out of 20 false negatives belong to such classes. The signs with the assigned detector, *Parking allowed for all vehicles*, *Speed limit* and *Intersection with priority to the right* have the maximum number of false negatives (3). However, a detailed analysis shows that these false negatives are a consequence of difficult conditions, rather than the fact that they are “hard to detect”. Typical examples include perspective views, worn-out, occluded or difficult annotations as shown on Fig. 7.

Thirdly, there are 4 classes without an assigned detector that do not have any false negatives shown in Fig. 5. Note that Fig. 5b and Fig. 5c also contain detections of other signs. This shows that the feature sharing improves performance with respect to the dedicated detector approach.

2) Comparison with Other Approaches: We compare the performance of the multiplicative kernel with the following methods. All methods are implemented in C++ using the SVM-Light library [15] for the SVM training.



(a) PR curve (b) ROC curve

Fig. 6. Comparison with other approaches.

a) One vs All: This configuration includes 19 class-specific linear SVM detectors. Traffic signs are not used as negatives.

b) SVM Baseline: This is a baseline jointly-trained approach. The classification function employs the linear SVM which treats all traffic sign classes as a single foreground class:

$$C(x) = \sum_{x_s \in S} \alpha_s \cdot x_s^T \cdot x \tag{4}$$

The parameter x_s denotes a support vector, which is a HOG vector of either foreground or background image patch.

c) Combined Linear: This approach is similar to the multiplicative kernel because it also uses the notion of foreground training samples as class memberships. The classification function employs a single linear kernel with a concatenated vector $[x, x_i]$ as a parameter, where x denotes a feature vector which we want to classify and x_i a foreground training sample.

$$C(x, x_i) = \sum_{(x_{s1}, x_{s2}) \in S} \alpha_s \cdot [x_{s1}, x_{s2}]^T [x, x_i] \tag{5}$$

The parameter $x_s = (x_{s1}, x_{s2})$ denotes a support vector. This method also produces a set of detectors $w(x, x_i)$, each corresponding to the specific foreground example x_i. Similar as in section III-A, individual detectors are constructed by plugging the x_i value into (5). We obtain the following expression for the detector $w(x, x_i) = w \cdot x + b(i)$, where:

$$w = \sum_{(x_{s1}, x_{s2}) \in S} \alpha_s \cdot x_{s1}^T \tag{6}$$

$$b(i) = \sum_{(x_{s1}, x_{s2}) \in S} \alpha_s \cdot x_{s2}^T \cdot x_i \tag{7}$$

In contrast to (3), where the weight vector depends of the foreground sample x_i, the equation (6) shows that this value is the constant for all detectors. As a consequence, the detector with the maximum value $b(i)$ exhibits the same recall value as the entire set of detectors. Therefore, in order to make the detection process practical, we use only that detector at runtime. The purpose of this method is to assess if the multiplicative kernel is really necessary to solve the problem.

TABLE I. PERFORMANCE OF DIFFERENT DETECTION APPROACHES

Method	N	AuPR	AuROC	Det. rate	FP/img
One vs All	19	58.8	51	81.2	0.9
SVM Baseline	1	12.5	89.9	97.8	69.8
Mult. Selected	10	27.8	86.6	98.2	15.6
Mult. Clustered	10	19.8	79.5	94.4	14.5
Comb. Linear	**1**	27	**86.5**	96.9	21.2

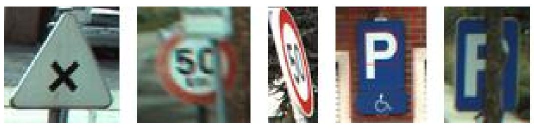


Fig. 7. Examples of false negatives for the *Multiplicative Selected* method.

Table I and Fig. 6 show the comparison results. For each method, we report the number of employed SVM linear detectors *N*. Lower number of detectors enables faster detection. The results point out four important facts.

Firstly, the individually trained *One vs All* method produces the best results in the PR space with respect to the other jointly trained methods. However, this is not a relevant measure because this method exhibits the lowest detection rate (81.2%) and the lowest *AuROC* value (51). Fig. 6b shows that this method performs worse than chance for higher FPR values. This is a result of the fact that certain true positives have lower detection score than false positives. This proves that feature sharing increases the performance in the ROC space.

Secondly, among the jointly trained methods, the *SVM Baseline* yields the best result in ROC space at the cost of the worst performance in the PR space (12.5). This method exhibits almost 70 FP per image. Therefore, it does not produce a good separation between foregrounds and backgrounds and it is not applicable for the detection task. Further, all jointly trained methods exhibit a low *AuPR* value. One possible way to improve the performance of these methods would be to apply the cascading approach. For example, training the *Multiplicative Selected* on hard negatives obtained from the first stage *SVM Baseline* classifier would decrease the number of FP and increase precision.

Thirdly, the selection algorithm employed in the *Multiplicative Selected* outperforms clustering in the *Multiplicative Clustered* method [1] in both ROC and PR space for 7% and 8%, respectively.

Fourthly, the multiplicative kernel employed in the *Multiplicative Selected* produces marginally better results with respect to the linear kernel of the *Combined Linear* in both PR and ROC space. In addition, the *Combined Linear* method is 10 times faster than the *Multiplicative Selected*. However, the recall of the single detector of the *Combined Linear* method is equivalent to the recall of the complete detector set produced by the equations (6) and (7). On the other hand, the recall of the *Multiplicative Selected* could be improved by increasing the number of detectors produced by the detector selection procedure. The target number of detectors N could be used as a stop condition in pseudo-code given in Fig. 2. To conclude, the *Combined Linear* yields the acceptable performance in ROC space at the low detection complexity. This result is somewhat different than the original hypothesis that multiplicative kernel is necessary in order to provide both feature sharing between foregrounds as well as the separation against backgrounds.

V. Conclusion

We explore the properties of the multiplicative kernel [1] for multi-class detection. In order to evaluate performance, we use the Belgian traffic sign dataset [2].

There are several benefits of the multiplicative kernel approach. Firstly, the experiments show that feature sharing increases the *AuROC* up to 35.6% with respect to the individual detector approach. This is because the individual detectors require a substantial number of samples per class to achieve good performance. Secondly, this approach avoids the partitioning into subclasses by using the foreground training samples as class membership labels and the multiplicative kernel. As a consequence, the number of obtained detectors equals to the number of foreground samples. Thirdly, we use a selection procedure in order to determine a representative set of detectors from the large initial detector pool. The experiments show that our approach converges with the number of detectors which is half the size of the number of classes. This suggests sub-linear detection complexity. In addition, detector selection yields better results than the original clustering approach [1].

We also discuss the following issues. Firstly, whether the multiplicative kernel is necessary in order to provide feature sharing and separation against backgrounds. In contrast to our initial hypothesis, the multiplicative kernel obtains marginally better results with respect to the linear kernel based upon the same concept of foreground samples as class memberships. This suggests that this task could also be efficiently solved with the linear kernel approach which is 10 times faster. Secondly, we are also interested in the method precision. The results in the PR space show that the multiplicative kernel yields maximal *AuPR* of 27.8%, which is half the precision obtained by the individual detector approach. Therefore, for future work, we propose the cascading approach in order to increase the precision.

Acknowledgment

This work has been supported by research projects Vista (EuropeAid/131920/M/ACT/HR) and Research Centre for Advanced Cooperative Systems (EU FP7 #285939). The authors wish to thank Thomas Mensink for useful suggestions regarding the choice of *Combined Linear* approach for comparison.

References

[1] Q. Yuan, A. Thangali, V. Ablavsky, and S. Sclaroff, "Learning a family of detectors via multiplicative kernels," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, no. 3, pp. 514–530, 2011.

[2] M. Mathias, R. Timofte, R. Benenson, and L. J. V. Gool, "Traffic sign recognition - how far are we from the solution?" in *IJCNN*. IEEE, 2013, pp. 1–8.

[3] A. Torralba, K. Murphy, and W. Freeman, "Sharing visual features for multiclass and multiview object detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, no. 5, pp. 854–869, 2007.

[4] B. Wu, H. Ai, C. Huang, and S. Lao, "Fast rotation invariant multi-view face detection based on real adaboost," in *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*, 2004, pp. 79–84.

[5] T.-K. Kim and R. Cipolla, *Multiple classifier boosting and tree-structured classifiers.* Berlin: Springer, 2013, pp. 163–196.

[6] S. Fidler and A. Leonardis, "Towards scalable representations of object categories: Learning a hierarchy of parts," in *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, June 2007, pp. 1–8.

[7] L. Zhu, Y. Chen, A. Torralba, W. Freeman, and A. Yuille, "Part and appearance sharing: Recursive compositional models for multi-view," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, June 2010, pp. 1919–1926.

[8] N. Razavi, J. Gall, and L. Van Gool, "Scalable multi-class object detection," in *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, ser. CVPR '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 1505–1512. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2011.5995441

[9] V. Zadrija and S. Segvic, "Multiclass road sign detection using multiplicative kernel," *CoRR*, vol. abs/1310.0311, 2013.

[10] D. Bremner, E. Demaine, J. Erickson, J. Iacono, S. Langerman, P. Morin, and G. Toussaint, "Output-sensitive algorithms for computing nearest-neighbour decision boundaries," in *Algorithms and Data Structures*, ser. Lecture Notes in Computer Science, F. Dehne, J.-R. Sack, and M. Smid, Eds. Springer Berlin Heidelberg, 2003, vol. 2748, pp. 451–461. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-45078-8_39

[11] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, 2005, pp. 886–893 vol. 1.

[12] C. Vondrick, A. Khosla, T. Malisiewicz, and A. Torralba, "Hoggles: Visualizing object detection features," in *ICCV*. IEEE, 2013, pp. 1–8.

[13] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun. 2010.

[14] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in *Proceedings of the 23rd International Conference on Machine Learning*, ser. ICML '06. New York, NY, USA: ACM, 2006, pp. 233–240. [Online]. Available: http://doi.acm.org/10.1145/1143844.1143874

[15] T. Joachims, "Making large-scale support vector machine learning practical," in *Advances in Kernel Methods – Support Vector Learning*, B. Schölkopf, C. J. C. Burges, and A. J. Smola, Eds. Cambridge, MA, USA: MIT Press, 1999, pp. 169–184. [Online]. Available: http://dl.acm.org/citation.cfm?id=299094.299104