Computer Vision and Image Understanding 150 (2016) 95–108

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu

Large scale Gaussian Process for overlap-based object proposal scoring

S.L. Pintea[a,*], S. Karaoğlu[a], J.C. van Gemert[a,b], A.W.M. Smeulders[a]

[a] Intelligent Systems Lab Amsterdam, University of Amsterdam, Science Park 904, 1098 XH, Amsterdam, Netherlands
[b] Computer Vision Lab, Delft University of Technology, Delft, Netherlands

ARTICLE INFO

Article history:
Received 1 September 2015
Revised 22 April 2016
Accepted 10 May 2016
Available online 11 May 2016

Keywords:
Gaussian Process
Metric learning
Kernel methods
Object proposals

ABSTRACT

This work considers the task of object proposal scoring by integrating the consistency between state-of-the-art object proposal algorithms. It represents a novel way of thinking about proposals, as it starts with the assumption that consistent proposals are most likely centered on objects in the image. We pose the box-consistency problem as a large-scale regression task. The approach starts from existing popular object proposal algorithms and assigns scores to these proposals based on the consistency within and between algorithms. Rather than generating new proposals, we focus on the consistency of state-of-the-art ones and score them on the assumption that mutually agreeing proposals usually indicate the location of objects. This work performs large-scale regression by starting from the strong Gaussian Process model, renowned for its power as a regressor. We extend the model in a natural manner to make effective use of the large number of training samples. We achieve this through metric learning for reshaping the kernel space, while maintaining the kernel-matrix size fixed. We validated the new Gaussian Process models on a standard regression dataset — Airfoil Self-Noise — to prove the generality of the method. Furthermore, we test the suitability of the proposed approach for the undertaken box scoring task on Pascal-VOC2007. We conclude that box scoring is possible by employing overlap statistics in a new Gaussian Process model, fine tuned to handle large amounts of data.

1. Introduction

In this work we focus on the task of object proposal scoring by integrating the consistency of proposals among multiple state-of-the-art algorithms. We formulate the box-consistency problem as a large-scale regression task, as there are over 2000 proposals per image within only one algorithm. Furthermore, assigning scores to boxes is inherently a regression task. Thus, we choose as a starting point for our approach the strong Gaussian Process model (Bottou, 2007; Hensman et al., 2013; Quiñonero-Candela and Rasmussen, 2005; Snelson and Ghahramani, 2005). In this work, we extend the standard Gaussian Process regression model to make effective use of the large number of training samples by employing metric learning. We achieve this by looking at the consistency between the proposals of different state-of-the-art algorithms.

Object proposal methods can be considered to have reached a satisfactory level when inspecting the recall of state-of-the-art methods (Alexe et al., 2010; Cheng et al., 2014; Krähenbühl and Koltun, 2014; Manen et al., 2013; Rahtu et al., 2011; Uijlings et al., 2013; Zitnick and Dollár, 2014). However, the high recall comes at the cost of a large number of boxes — between 1000 and 3000 boxes per image. This work aims at precisely this: re-scoring existing proposals of different algorithms such that we can more easily find the good ones.

Another gain following from the ability to assign goodness scores to boxes is self-assessment — providing a goodness score to each bounding box. This allows for the selection of a limited number of boxes to be used at a subsequent step for object detection. Well-known methods such as *selective search* (Uijlings et al., 2013) and *prim* (Manen et al., 2013), despite their good performance, lack the ability of self-assessment by design. This work provides a manner of assigning goodness scores to any proposal box.

There is a common denominator between well-known object detection methods (Alexe et al., 2010; Cheng et al., 2014; Krähenbühl and Koltun, 2014; Manen et al., 2013; Rahtu et al., 2011; Uijlings et al., 2013; Zitnick and Dollár, 2014) — they use as a starting point different assumptions yet they attain comparable performance. Therefore, there is gain in jointly employing them. In this work, we hypothesize that the consistency in proposals between different methods is revealing as to the true location of objects in an image. This idea is underlined in Fig. 1. The figure depicts a box proposed by the *edge-boxes* algorithm (Zitnick and Dollár, 2014), and its closest neighbors in a set of 6 other object

* Corresponding author.
E-mail address: s.l.pintea@uva.nl (S.L. Pintea).

http://dx.doi.org/10.1016/j.cviu.2016.05.002
1077-3142/© 2016 Elsevier Inc. All rights reserved.

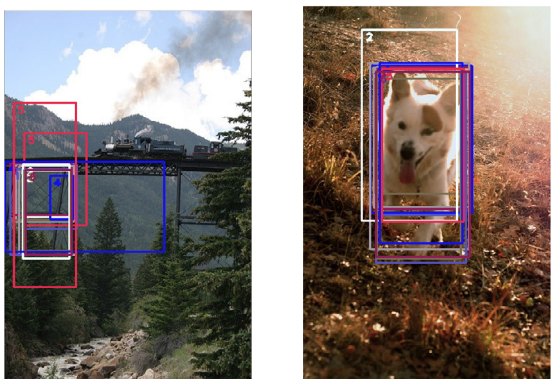


Fig. 1. Proposals characterized by consistency in overlap with other proposals, tend to be centered on objects.

proposal algorithms (Alexe et al., 2010; Cheng et al., 2014; Krähenbühl and Koltun, 2014; Manen et al., 2013; Rahtu et al., 2011; Uijlings et al., 2013). In this work, rather than extracting appearance features from the pixels enclosed in the bounding boxes, we build our features upon the consistency in overlap between the neighboring boxes within the same proposal algorithm as well as the other considered proposal algorithms. To our knowledge we are the first to consider box scoring from box-overlap information. Moreover, this method can be applied to any object proposal algorithm, as it is not restricted to only the seven algorithms discussed here.

We cast the box-consistency as a large-scale regression problem, given that the scores associated with boxes are continuous variables to be learned. The box scoring function is nonlinear in the feature space, since a satisfactory filtering of boxes is hard to achieve. Gaussian Processes are strong nonlinear — kernel-based — regressors characterized by a high descriptive power (Bottou, 2007; Hensman et al., 2013; Quiñonero-Candela and Rasmussen, 2005; Snelson and Ghahramani, 2005). The proper kernel characteristics, describing the similarities between samples, are estimated directly from the data. Moreover, they are non-parametric and have comparable computational costs with their discriminative counterpart, the SVR (Support Vector Regressors). Similar to SVR, Gaussian Processes are kernel methods, thus, require the estimation of a kernel matrix, squared in the size of training data. Given that we have numerous training samples — proposals in the training set, we introduce an adaption of the Gaussian Process for large-scale problems. This allows us to retain the descriptive power of the Gaussian Process, while limiting the kernel-matrix size to a fixed small set of centroids. We subsequently employ additional samples to learn the kernel distances in a metric learning formulation through loss optimization. Thus, we reshape the kernel space such that the model better describes the target space distribution. This theoretical extension is not restricted to box scoring, and it can be applied to regression problems with a prohibitively large number of samples. Therefore, we additionally test the new Gaussian Process models on a standard machine learning dataset. This demonstrates the generality of our theoretical contribution.

To summarize this work: (i) we theoretically extend the Gaussian Process model for large-scale regression. We do so by retaining a fixed kernel-matrix size. To compensate for the lost information we employ metric learning for reshaping the kernel space to better fit the training targets. (ii) We introduce a novel view of box proposal scoring by learning it from the consistency between the box proposals of different algorithms. (iii) We validate the new Gaussian Process models on a standard machine learning regression data-set — the Airfoil Self-Noise Data-set of NASA. This proves the generality of the model. Finally, we test the suitability of the proposed approach in the context of box regression, on the Pascal-VOC2007 data-set.

2. Related work

2.1. Object proposal methods

In the literature, generating object proposals has been a main focus. Methods such as *objectness* (Alexe et al., 2010) and *core* (Rahtu et al., 2011), rely on the fact that objects are salient. On the other hand, methods such as *prim* (Manen et al., 2013), and *selective search* (Uijlings et al., 2013), consider a hierarchical approach to object proposals, based on the assumption that object parts are internally coherent in terms of color, texture or location in the image. They generate proposals by starting from an over-segmented image and iteratively merging similar segments. Finally, the most recent methods — *bing* (Cheng et al., 2014), *geodesic* (Krähenbühl and Koltun, 2014) and *edge-boxes* (Zitnick and Dollár, 2014) — ascertain that objects are visible through strong boundaries. In this work, rather than generating object proposals by introducing a new paradigm, we start from existing popular proposal methods and learn the goodness of boxes. We relate the idea of box goodness to the consistency in proposals between different methods. Our underlying assumption is that none of the above paradigms wins exclusively in the end, but rather, all bare reliable truth about object locations. Thus, there is gain to be achieved by combining them.

2.2. Deep net proposals

Convolutional Neural Networks (CNN) have been recently used with success for object detection starting from already existing object proposals (Girshick, 2015), (Girshick et al., 2014), (Ren et al., 2015), or for proposing class-agnostic bounding boxes (Erhan et al., 2014), (Karianakis et al., 2015). (Girshick et al., 2014), brings forth the well known RCNN (Regions with CNN features) model which uses Selective Search (Uijlings et al., 2013), object proposals in a CNN for object detection. Girshick (2015), improves the RCNN method of Girshick et al. (2014), in terms of training and test speed, as well as detection accuracy and it coins the new method “Fast RCNN”. Ren et al. (2015), introduces a fully-convolutional network that predicts object bounding boxes. The network can be trained to share features with the “Fast RCNN” (Girshick, 2015), and thus, be used for object detection. Instead of considering the feasible locations and sizes of objects in the image (Zhao et al., 2014), in Erhan et al. (2014), a neural network based on saliency features is proposed for generating class independent bounding boxes together with an object likelihood score. Karianakis et al. (2015), advises the use of CNNs for generating object proposals by advancing a boosting approach based on the hierarchical CNN features which gives competitive performance on the object detection task. The recent work of Chavali et al. (2015), points out shortcomings in the current object proposal evaluation protocols and offers a fully annotated data-set for evaluation as well as performing diagnostics on existing proposal methods. It is of interest to take into account these findings, however we do not aim at diagnosing popular object proposals, but rather at testing if consistency in proposals discloses the true location of objects. In contrast to methods generating new object proposals — based on hand-crafted features, integrating prior knowledge about the task at hand or, learned in a CNN framework — we aim at combing the information given by a set of largely used object proposal algorithms such that we can estimate the goodness of a given proposal box.

2.3. Combining methods

To learn the goodness of bounding boxes, we start from a set of existing proposal methods. We consider the overlap between the boxes as the only required training information. This can be seen somewhat similar to the idea of combining existing methods which was studied in Karaoglu et al. (2014); Xu et al. (2014). Karaoglu et al. (2014), combines object detectors by using the detection scores together with the maximum overlap with other detections. Xu et al. (2014), merges pedestrian detectors by employing the scores associated with each detection and clustering the detections. Here we do not have scores associated with each box. Therefore, we need a method that allows us to both assign scores to boxes and also integrate the information of all proposal methods. Rather than merging existing proposals in a straight-forward fashion, we learn box goodness based on consistency. We correlate the consistency in the proposals of different methods to the goodness of a certain box. We do so in a Gaussian Process regression framework.

2.4. Box overlap as features

Vezhnevets and Ferrari (2015), defines three overlap statistics that describe the relative position of two object proposal bounding boxes. These overlap statistics indicate roughly the positioning of two boxes: their relative overlap, and whether the first box is included in the second or vice-versa. Unlike Vezhnevets and Ferrari (2015), where the statistics are used as targets for regression, here we employ these three statistics in the feature definition of our proposals. Furthermore, we do not make use of any additional appearance features based on the pixel values enclosed by the bounding boxes. We want to challenge the idea of box scoring from overlap information only.

In Vezhnevets and Ferrari (2015), the use of Gaussian Processes is also advanced, albeit with a different goal in mind — object detection. The authors of Vezhnevets and Ferrari (2015), learn from appearance features extracted from the pixels enclosed by the bounding boxes to predicted overlap statistics with other boxes. These overlap statistics are subsequently used together with appearance features in an Exemplar-SVM for detection. Dissimilar to Vezhnevets and Ferrari (2015), here we employ these three overlap statistics to describe the boxes, with the goal of learning the quality of box proposals in the large-scale Gaussian Process regression framework.

2.5. Gaussian process versus other regressors

The problem of box scoring is inherently a regression problem as the goodness scores are continuous variables. Given that the task of estimating box goodness is not straight-forwardly solved and we have numerous training samples, we use a non-linear regressor. We propose the use of Gaussian Processes (Rasmussen, 2006), as they are renowned for their strength as non-linear regressors. Neal (2012) shows that when the number of hidden units tends to infinity, the distribution of functions generated by a neural network converges to a Gaussian Process. Moreover, RVM (Relevance Vector Machines), which are another choice of non-linear regressors, can be seen as a special case of Gaussian Processes. In the RVM case the covariance function is degenerate (Bishop, 2006; Rasmussen, 2006). When comparing the SVR (Support Vector Regression)/SVM (Support Vector Machine) with the Gaussian Process regressor/the Gaussian Process classifier, they can be shown to optimize very similar quantities (Rasmussen, 2006). However, they are not equivalent as the former is a discriminative model while the Gaussian Process is generative. This also entails that the Gaussian Process can associate a certainty estimate with every prediction. Moreover, the Gaussian Process model can learn the kernel characteristics automatically from the data. This provides more flexibility to the model.

Similar to Vivarelli and Williams (1999), we also propose the use of a full covariance matrix in the kernel function of the Gaussian Process. But unlike this work, rather than using eigen analysis, we propose to learn this covariance through metric learning. This step helps to better model the target distribution by employing additional available training data.

2.6. Large scale Gaussian Processes

Gaussian Processes focusing on the local information in the data samples have been proposed in Bo and Sminchisescu (2012), 26), Snelson and Ghahramani (2007), U039">Urtasun and Darrell (2008), proposes a local mixture of Gaussian Processes where the hyperparameters of each component are learned in an online fashion. The Gaussian Process models proposed in this paper are based on a restricted set of training sample which represent cluster centers describing the data in a certain area of the feature space. Despite these samples locally describing the feature space, the Gaussian Process model we propose is a global one rather than a local one. The gain of our method, with respect to the local regression methods, is having one unified model rather than a set of models trained on different parts of the data.

Previous work has also focused on sparse methods for restraining the kernel matrix size in the Gaussian Process. Methods such as Cao et al. (2013), Csató and Opper (2002), Hensman et al. (2013), Lawrence et al. (2003), Quiñonero-Candela and Rasmussen (2005), Ranganathan et al. (2011), Snelson and Ghahramani (2005), Titsias (2009) propose global approximations in order to achieve efficiency. Csató and Opper (2002), proposes an online algorithm in which the relevant training samples are selected in a sequential manner. Quiñonero-Candela and Rasmussen (2005), presents a literature survey where existing sparse Gaussian Process methods are presented in a unified manner by changing the definition of the prior, thus emphasizing similarities between existing methods. Snelson and Ghahramani (2005), learns a small set of pseudo-inputs together with the model hyperparameters through gradient optimization. This method can be seen as a Bayesian regression model where the noise is input dependent. Lawrence et al. (2003), builds on active learning and forward selection to find a sparse set of training samples which is advantageous both in terms of speed and storage requirements. Titsias (2009) introduces a variational inference method that finds the inducing variables by minimizing the KL divergence between the variational distribution and the exact posterior distribution. Hensman et al. (2013), proposes a stochastic variational inference approach that relies on a set of global variables which factorize into observations and latent variables. Cao et al. (2013), jointly optimizes the selection of the inducing points — which provide the Gaussian Process regression with sparsity — and finds the optimal Gaussian Process hyperparameters. Ranganathan et al. (2011), proposes an online sparse Gaussian Process regression method that uses Cholesky updates for sparse kernel matrices. Similarly, the models proposed in the paper are also sparse in the sense that rather than using the complete training data, we rely on a fixed set of cluster centers in the data space. Dissimilar to existing methods, we follow an approach based on metric learning through loss minimization for hyperparameter optimization. This is more common for discriminative methods — i.e. SVM , SVR .

2.7. Metric learning

In order to efficiently employ the large amount of training data while keeping the kernel-matrix size fixed, we use metric learning.

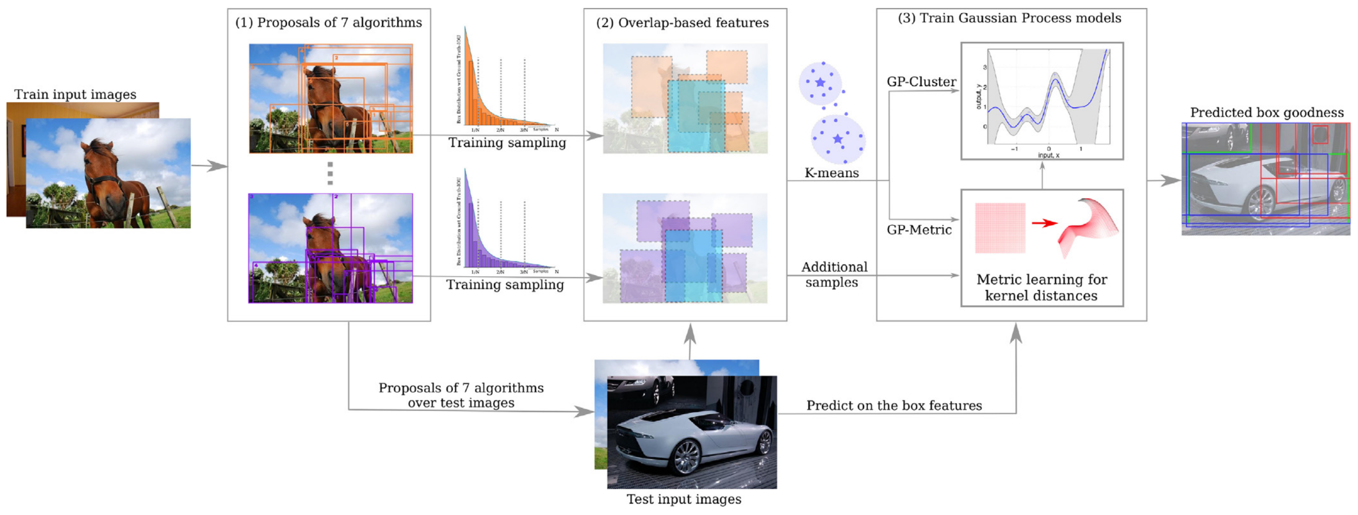

Train input images
(1) Proposals of 7 algorithms
Training sampling
Training sampling
(2) Overlap-based features
K-means
Additional samples
(3) Train Gaussian Process models
GP-Cluster
GP-Metric
Metric learning for kernel distances
Predicted box goodness
Proposals of 7 algorithms over test images
Predict on the box features
Test input images

Fig. 2. Method overview. In the first step we generate box proposals for each image using the seven considered proposal algorithms (Alexe et al., 2010; Cheng et al., 2014; Krähenbühl and Koltun, 2014; Manen et al., 2013; Rahtu et al., 2011; Uijlings et al., 2013; Zitnick and Dollár, 2014). In the training phase, we sample the proposals of each algorithm based on their distribution with respect to the overlap with the ground truth. Subsequently, in the second step, we compute overlap statistics for each retained box with boxes from all proposal algorithms. Both during training and test, these overlap statistics with neighboring boxes describe our features. During training, a subset of these box features are used to obtain cluster centers from K-means. In the third step we propose 2 models: *GP-Cluster* — a Gaussian Process model trained on the K-means cluster centers only, and *GP-Metric* — a Gaussian Process trained on the same K-means cluster centers, but for which additional training samples (boxes with associated features) are used to adjust the kernel distances through metric learning. The targets of the regressors are the overlap scores with the ground truth boxes.

Metric learning has been previously the focus of works such as Huang and Sun (2013); Kostinger et al. (2012); Titsias and Lázaro-Gredilla (2013); Weinberger and Saul (2009); Xing et al. (2002); Ying and Li (2012). Ying and Li (2012), proposes a metric learning approach based on eigenvalue optimization, connecting these two trends together. Xing et al. (2002), analyzes the use of metric learning for improving clustering and proposes learning similarity measures in a convex optimization formulation. Somewhat similar, we also combine clustering with metric learning, but clustering is not the end goal. We do so in order to effectively employ the training data in the Gaussian Process and to add more descriptiveness to the model. Titsias and Lázaro-Gredilla (2013), advances a variational Gaussian Process method which shifts the kernel into a space where the hyperparameters are neutralized to value one. Unlike in our work, the goal of Titsias and Lázaro-Gredilla (2013), is to achieve a model where the hyperparameters are integrated out. Here, we adapt the kernel shape such that it incorporates information from the discarded training samples. Huang and Sun (2013), proposes kernel regression with sparse metric learning that imposes a regularization over the projection matrix to be learned. Similar to Huang and Sun (2013), we also use metric learning to allow the model to better map the target distribution. In this work, we perform the metric learning through loss optimization in the SGD (Stochastic Gradient Descent). We, additionally, employ in the SGD the cone projection step described in Weinberger and Saul (2009), together with their update of the learning rate.

3. Box goodness through regression

3.1. Method overview

This work proposes learning box goodness in a Gaussian Process regression framework based on the consistency in proposals of seven different object proposal algorithms (Alexe et al., 2010; Cheng et al., 2014; Krähenbühl and Koltun, 2014; Manen et al., 2013; Rahtu et al., 2011; Uijlings et al., 2013; Zitnick and Dollár, 2014). Fig. 2 depicts the main steps entailed by our method.

The first step generates proposal boxes by applying all seven algorithms. Given the large number of proposals generated by each algorithm — $\approx$ 2000 per image — during training, we sample boxes from each proposal method. The sampling is based on the distribution of the overlap of the training boxes with the ground truth boxes. This is meant to retain a set of diverse boxes for training, ranging from bad to good.

In the second step, features are defined for the retained training boxes. These features measure the consistency in overlap with other boxes from the same proposal algorithm as well as other algorithms. The feature definition is used both for training and test boxes. Additionally, during training we cluster these features into a predefined set of clusters (in our experiments, we use 500 cluster centers) using *K*-means. This aims at both reducing the kernel-matrix size of the Gaussian Process and making the regressor more robust.

Finally, we introduce two models: the *GP-Cluster* — Gaussian Process trained on the *K*-means cluster centers only, and *GP-Metric* — Gaussian Process regressor trained on the same *K*-means cluster centers but employing metric learning on additional training samples. The targets of the Gaussian Process regressor are represented by the maximum overlap with a ground truth box, as this indicates the goodness of a box. During test time, we extract overlap features for all proposal boxes of all seven algorithms and assign a quality score to each proposal by performing inference in the trained Gaussian Process models. Metric learning is used to reshape the kernel such that the model better describes the target space. The proposed methods not only provide a box scoring solution based on consistency between different proposal algorithms, but also allow for self-assessment. This is specifically advantageous for methods that do not have a way in which to incorporate box scores, such as Manen et al. (2013); Uijlings et al. (2013) — this will be addressed in experiment **Exp 2.3**. Furthermore, the considered Gaussian Process extension by employing metric learning is a general addition to the model that is not restricted to only the box scoring problem. This model can be applied whenever dealing with a large number of training samples — as shown in experiment **Exp 1**.

3.2. Combining proposals

Our underlying assumption for the box scoring problem is that the consistency between object proposals is useful in deciding

Table 1
Evaluation of the seven considered object proposal algorithms, run by us on Pascal-VOC2007 — consistent with the literature (Alexe et al., 2010; Cheng et al., 2014; Hosang et al., 2014; Krähenbühl and Koltun, 2014; Manen et al., 2013; Rahtu et al., 2011; Uijlings et al., 2013; Zitnick and Dollár, 2014). Edge-boxes achieves the best recall, while at the same time, generating the largest number of proposals per image.

Method	# Proposals/Image	# True Boxes	Recall
Core Rahtu et al. (2011)	1000	9348	0.776
Objectness Alexe et al. (2010)	1000	10,660	0.886
Prim Manen et al. (2013)	2494	11,418	0.949
SSE Uijlings et al. (2013)	2008	11,516	0.957
Bing Cheng et al. (2014)	1924	11,470	0.953
Edge-Boxes Zitnick and Dollár (2014)	3479	11,860	0.985
Geodesic Krähenbühl and Koltun (2014)	653	11,059	0.919
Ground truth	–	12,032	–
Combined	**10,758**	**12,005**	**0.998**

the goodness of boxes. The recent paper of Hosang et al. (2014), considers twelve state-of-the-art object proposal methods. Out of these we have selected six methods based on their being relatively fast at prediction time — less than 3 s per image. Moreover, we have additionally considered a newer method, geodesic object proposals (Krähenbühl and Koltun, 2014), which provides good performance in practice.

Table 1 lists the average number of boxes generated by each one of the considered algorithms together with their recall, evaluated by us on the Pascal-VOC2007 data-set. We choose the Pascal-VOC2007 data-set as it is a popular data-set for testing object proposals. In addition, we are interested in the overlap between different proposals, so the actual choice of the data-set has limited influence on the experimental outcome. The numbers indicate that the boxes proposed by the *edge-boxes* are the most accurate — achieving the highest recall. However, the total number of proposed boxes is relatively high. When considering all the proposals of all algorithms, the recall is very close to one. Thus, there is gain in trying to re-rank boxes of different algorithms based on their goodness.

In this work, we consider the boxes of these seven algorithms, as just merging all proposals from all algorithms achieves 0.998 recall. This almost solves the problem, were it not for this recall being reached at the cost of obtaining an impractically large set of boxes. We aim to perform box regression for finding an ordering of these proposals such that we can attain a good performance at a smaller number of boxes.

3.3. Box description and selection

Each one of the seven discussed algorithms provides a set of approximately 2000 boxes per image. We first need to select a subset of these boxes, as it is unfeasible to use all boxes in the kernel computation. For the selected boxes we devise a set of features that describe them in terms of the overlap with other boxes. These features are subsequently clustered. The cluster centers represent the actual training samples to be used for computing the training/test kernel distances.

Training Box Sampling. The training set is represented by proposed boxes and there are on average 2000 boxes proposed per image, thus $\approx 7 \times 2000$ training samples per image. This generates a prohibitively large kernel matrix. A first step towards making effective use of the training data is to sample the bounding boxes based on their IOU (Intersection Over Union) score with the ground truth boxes. The scores are also used as targets during regression training. We retain only the training boxes that have an IOU score greater than zero — boxes that intersect at least one ground truth box. The QWS (Quasi-random Weighted Sampling) (Kalal et al., 2008) implies adding the IOU scores of all boxes in one algorithm and one image, on a unit line. The line is divided into N equally sized segments and we sample one unique box from each such segment. In the experimental part, we sample 100 boxes per box-proposal algorithm out of 500 random training images using QWS.

Features. Given the input boxes of all algorithms, we define their features in terms of the overlap with neighboring boxes. We aim to employ the consistency between proposals as features for learning box-goodness. We achieve this by making use of the three statistics proposed in Vezhnevets and Ferrari (2015), depicted in Eq. 1. For each considered box we estimate its closest five neighbors in all the seve algorithms and compute these three statistics with the corresponding neighbors.

$$\left(\frac{\text{box}_1 \cap \text{box}_2}{\text{box}_1 \cup \text{box}_2}, \frac{\text{box}_1 \cap \text{box}_2}{\text{box}_1}, \frac{\text{box}_1 \cap \text{box}_2}{\text{box}_2}\right), \tag{1}$$

where box_1 represents the current box to be described, and box_2 represents one of its closest five neighbors. These statistics are concatenated into a feature vector of 105 dimensions, describing each box — 5 neighbors $\times$ 7 algorithms $\times$ 3 statistics. For ensuring stability of the features, the neighbors are ordered in descending order of their proximity to the current box being described. The regression targets are the maximum over the IOU scores with the ground truth boxes.

Clustering. The number of used training samples determines the kernel-matrix size in the Gaussian Process. This restricts us to using a very small fraction out of the available samples. In order to both limit the kernel-matrix size as well as make the regressor more robust, we cluster the box features corresponding to the sampled boxes. The clustering is performed using K-means. For box scoring we use 500 cluster centers, yet we also show in the experimental section the performance with respect to varying number of clusters.

4. Large scale Gaussian Process regression

We aim to assign goodness scores to proposals based on the maximum over the IOU scores with the ground truth boxes. As this is a regression problem, non-linear in the feature space, we adopt the Gaussian Process model. This is renowned for its power as a non-linear regressor, while having similar computational costs with its discriminative counterpart, the SVR (Rasmussen, 2006). In the next subsections we briefly revisit the standard Gaussian Process model. We subsequently, indicate the changes, entailed by the large-scale nature of the data, that we introduce in model.

4.1. Standard Gaussian Process regression model

For the estimation of the Gaussian Process kernel matrix we use the squared exponential kernel, as this is the standard choice

(Rasmussen, 2006). The training procedure involves the computation of the inverse of the kernel matrix. The test-time prediction together with the standardly used squared exponential kernel are given by Eqs. (2)-(3):

$$y^* = k_l(\mathbf{x}^*, \mathbf{X})^T \left(k_l(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathbf{I}\right)^{-1} \mathbf{y}, \tag{2}$$

$$k_l(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\| \mathbf{x}_i - \mathbf{x}_j \|^2}{2l^2}\right), \tag{3}$$

where $\mathbf{X}$ is the matrix of training samples, $k_l(\cdot, \cdot)$ is the kernel function depending on l — the length-scale hyperparameter of the Gaussian Process, $\mathbf{y}$ is the vector of training targets and y^* is the prediction on the current test sample, $\mathbf{x}^*$ is the input test sample, and σ^2 is the noise hyperparameter, while $\mathbf{I}$ denotes the identity matrix.

Gaussian Processes are non-parametric models, as seen in Eq. (2) – there are no weights to be estimated from the data. The length-scale, l, and the noise, σ^2, represent the hyperparameters of the model. They do not directly describe the model, they only affect the kernel distances. The hyperparameters are estimated from the data during training and help shape the kernel. This is achieved by adjusting the kernel distances to more suitably describe the similarities between samples.

4.2. Augmenting Gaussian Processes with metric learning

The clustering of samples solves the problem of too large kernel-matrix sizes. However, this discards valuable information as it only retains the few cluster centers and disregards the rest of the samples. Therefore, we may ignore the variation in the target space given by the disregarded samples. In the kernel-based methods, the choice of the kernel defines the distance metric between the samples. By employing metric learning we make effective use of the additional samples present in the training data and use them to reshape the kernel space. This enables the Gaussian Process to better learn the target space variations. By doing so, we keep the kernel-matrix size fixed while adjusting the kernel distances on additional training samples.

Covariance-based Kernels. We aim to add back into the model the information lost by training on the cluster centers only. To do so, similarly to Vivarelli and Williams (1999), we add more descriptiveness into the representation by expanding the kernel definition to incorporate a covariance matrix. We change the length-scale parameter of the Gaussian Process – Eq. 3 – to be a covariance matrix as depicted in Eq. (4):

$$k_{\boldsymbol{\Sigma}}(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{1}{2}(\mathbf{x}_i - \mathbf{x}_j)\boldsymbol{\Sigma}(\mathbf{x}_i - \mathbf{x}_j)^T\right). \tag{4}$$

This makes the kernel more flexible. It allows us to learn from the data not only the correct scale — as in the case of the scalar lengthscale, l — but also the correct shape. We subsequently, perform metric learning to determine the covariance, $\boldsymbol{\Sigma}$, from new sets of training examples.

Metric Learning Optimization. We learn the added covariance matrix through metric learning, unlike Vivarelli and Williams (1999). This enables us to find a kernel function that facilitates the model to better describe the target distribution. In order to learn $\boldsymbol{\Sigma}$ from the data, we assume the squared loss: $\mathcal{L} = \sum_n (y_n - y_n^*)^2$. Consequently, we evaluate the gradient of the loss function over the Gaussian Process model with respect to $\boldsymbol{\Sigma}$. This gradient is used in an SGD (Stochastic Gradient Descent) optimization to iteratively update the covariance over batches of samples. We estimate the gradient formulation as in Eqs. (5)-(9) — detailed derivations provided in Appendix A.

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}} = \left[\frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}}\right] + \left[\frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}}\right]^T - \text{diag}\left[\frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}}\right] \tag{5}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}} = 2\sum_n^{\text{train}} (y_n - y_n^*) \left[\sum_i^{\text{clusters}} \left[\sum_j^{\text{clusters}} -K_{ij}^{inv} \mathbf{M}_{ji} K_{ji}^{inv}\right.\right.$$
$$\left.\left. y_i k_{\boldsymbol{\Sigma}}(\mathbf{x}_i, \mathbf{x}_n) + \alpha_i \mathbf{M}_{in}\right] + 2\lambda \boldsymbol{\Sigma}, \tag{6}$$

$$\mathbf{M}_{ji} = -\frac{1}{2}(\mathbf{x}_j - \mathbf{x}_i)^T(\mathbf{x}_j - \mathbf{x}_i) k_{\boldsymbol{\Sigma}}(\mathbf{x}_j, \mathbf{x}_i), \tag{7}$$

$$\mathbf{K}^{inv} = (k_{\boldsymbol{\Sigma}}(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathbf{I})^{-1}, \tag{8}$$

$$\boldsymbol{\alpha} = (k_{\boldsymbol{\Sigma}}(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathbf{I})^{-1} \mathbf{y}. \tag{9}$$

Given that $\boldsymbol{\Sigma}$ is symmetric, we use the derivation for symmetric matrices — Eq. (5). After each gradient step we reinforce that $\boldsymbol{\Sigma}$ has to be a symmetric and semi-positive definite matrix. This is done by performing a cone projection step as described in Weinberger and Saul (2009). Algorithm 1 provides the steps for estimating $\boldsymbol{\Sigma}$. Given that the optimization in terms of $\boldsymbol{\Sigma}$ may have local optima (Rasmussen, 2006), we restart the SGD at different length-scale ranges. We do so by initializing $\boldsymbol{\Sigma}$ with a diagonal matrix where the elements on the diagonal are $\frac{1}{l^2}$. The same ranges are used in the standard Gaussian Process for estimating the optimal length-scale parameter — l^*. This parameter optimization is done in the standard model through cross-validation over a held-out training set. After each SGD step we evaluate the reached $\boldsymbol{\Sigma}$ on the held-out training set. As suggested in Sutskever et al. (2013), we use the momentum parameter to make the gradient updates more smooth between iterations. We start by initializing the learning rate as $\epsilon = 0.05$ of the ratio between the Frobenius norm of the initial $\boldsymbol{\Sigma}$ setting and the first gradient step. We subsequently update the learning rate as suggested in Weinberger and Saul (2009). Following Bottou (2012), we add to the loss optimization a regularization term based on the norm of $\boldsymbol{\Sigma}$. This helps us in dealing with overfitting. Moreover, also as a way of avoiding overfitting, in the experimental part we use the Huber loss rather than the squared

Algorithm 1: Metric learning SGD for kernel distances.

```
1:  Get training samples using QWS and cluster them.
2:  Initialize starting lengthscale, $l$, and noise, $\sigma$.
3:  Assume the kernel of eq. 4.
4:  Initialize $\boldsymbol{\Sigma}_t \leftarrow \frac{1}{l^2}\mathbf{I}$, $\mathcal{V}_t \leftarrow \varnothing$, $t \leftarrow 0$.
5:  while $| \mathcal{L}_t - \mathcal{L}_{t+1} | \geq \theta$ do
6:      Sample a new set of training samples using QWS.
7:      $\nabla\mathcal{L}_t \leftarrow \frac{\partial \mathcal{L}_t}{\partial \boldsymbol{\Sigma}_t}$ as in eq. (5)-(9) over new samples.
8:      if (t == 0) then
9:          Initialize $\eta \leftarrow \epsilon \frac{Frob(\boldsymbol{\Sigma}_t)}{Frob(\nabla\mathcal{L}_t)}$, where $Frob(\cdot)$ is the Frobenius
            norm.
10:     end if
11:     Update $\mathcal{V}_{t+1} \leftarrow \mu\mathcal{V}_t - \eta\nabla\mathcal{L}_t$
12:     Update $\boldsymbol{\Sigma}_{t+1} \leftarrow \boldsymbol{\Sigma}_t + \mathcal{V}_{t+1}$.
13:     $\boldsymbol{\Sigma}_{t+1} \leftarrow \Pr(\boldsymbol{\Sigma}_{t+1})$,
        where $\Pr(\cdot)$ is the cone projection of $\boldsymbol{\Sigma}$.
14:     Compute the loss over the newly sampled set:
        $\mathcal{L}_{t+1} \leftarrow \sum_n (y_n - y_n^*)^2 + \lambda \mid \boldsymbol{\Sigma} \mid_2$ .
15: end while
16: Output $\boldsymbol{\Sigma}$ estimated through metric learning.
```

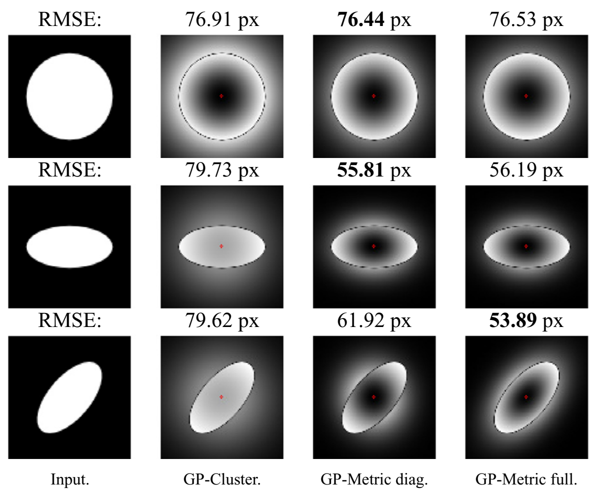


Fig. 3. Losses in predicting pixel intensities from pixel location on 3 images: a circle, an ellipse and a rotate ellipse. We use the center point – depicted in red – as the only cluster center to compute the kernel on. We randomly sample 100 pixels for evaluating the hyperparameter, $\mathbf{\Sigma}$. The second column depicts the loss of the standard Gaussian Process on one cluster. The third column shows the loss for the metric learning-based Gaussian Process with a diagonal covariance matrix, $\mathbf{\Sigma}$. The last column corresponds to the metric learning-based Gaussian Process with a full matrix, $\mathbf{\Sigma}$.

Table 2
Runtime estimates when using 500 cluster centers in the three proposed Gaussian Process models as well as the standard Gaussian Process where 1500 samples are randomly picked from the data. The times are estimated as average time for predicting goodness scores for one box as well as for all boxes (≈14,000 per image) in one image of Pascal VOC-2007 Dataset. The newly designed Gaussian Process models are more time-efficient when compared with the standard Gaussian Process model.

	GP Std.	GP models		
		Cluster	Metric-diag.	Metric-full
# Samples	≈ 1500	500	500	500
Millisec./Box	8.847 ms.	1.124 ms.	1.177 ms.	1.181 ms.
Sec./Image	132.76 s	16.87 s	17.66 s	17.73 s

loss. By doing so, we bound the contribution to the gradient for the samples that are far away from the corresponding target.

Metric Learning-based Gaussian Process Illustration. We have argued that by expanding the kernel definition to incorporate the covariance matrix, $\mathbf{\Sigma}$, we allow the Gaussian Process to be more descriptive. This helps in more effectively learning the shape of the target distribution. Fig. 3 depicts precisely this idea. Here, we use as input three gray-scale images, displayed on the first column in Fig. 3. We want to learn to predict the pixel intensity values from pixel locations in the image. Thus, our targets are represented by pixel intensities while our input features are the pixel locations. For all models we use only one training sample in the kernel computation, depicted in red. Hence, our training kernels have sizes 1 $\times$ 1. The two *GP-Metric* models use 100 additional pixel samples to learn the covariance, $\mathbf{\Sigma}$. However, they still use the same 1 sample for computing the training and test kernels. We consider two cases of the metric learning-based Gaussian Process: *GP-Metric diag* – the metric learning-based Gaussian Process in which the covariance matrix, $\mathbf{\Sigma}$, is assumed to be diagonal for time efficiency, and *GP-Metric full* – the metric learning-based Gaussian Process using a full covaraince matrix, $\mathbf{\Sigma}$. The first one is able to learn the correct shape of the target distribution. Nonetheless, due to the restriction imposed on the $\mathbf{\Sigma}$, to be diagonal, it cannot learn the appropriate rotation. The later learns both the shape and the appropriate orientation from additional samples. We depict the losses between the input image and the predictions for the three considered models. Low values (darker) correspond to small losses, while high values (brighter) represent larger prediction losses. We also plot the RMSE (Root Mean Squared Error) achieved by each method on each task. Fig. 3 indicates that the *GP-Cluster* can only learn the appropriate scale. However it cannot learn the shape and the orientation of the samples in the target space. The diagonal model learns the ellipse but fails to learn the rotated ellipse. Finally, the full model can predict the rotated ellipse from 1 cluster only, by reshaping the kernel space through metric learning. This illustration shows the gain of employing metric learning for reshaping the kernel space while restricting the kernel-matrix size.

5. Complexity analysis

The train-time computational complexity of a standard Gaussian Process is $\mathcal{O}(N^3 + N^2D)$. Here N represents the total number of training samples and D represents the data dimensions. The train-time complexity of the Gaussian Process trained on cluster centers is $\mathcal{O}(K^3 + K^2D)$. K represents the number of considered clusters and it is taken to be considerably smaller than the complete number of training samples, N. For the proposed metric learning based extension of the Gaussian Process method, the train-time complexity, as derived from Algorithm 1, is $\mathcal{O}(T(K^3 + K^2D + KMD(1 + KD)))$. Here T is the number of iterations and M is the mini-batch size in the SGD. For a reasonable parameter setting, we readily obtain train-time computational gains when compared to the standard Gaussian Process. If we set K to 500 clusters, T to 100 iterations, M to 100 samples in the mini-batch and assume 100-dimensional data, gains are achieved for training data sizes, N, larger than 7000 samples for the model based on diagonal covariance, and 30,000 samples using the full covariance.

At test time, the gain is even more notable, as for one test sample the standard Gaussian Process has an $\mathcal{O}(ND)$ complexity. While in our case, for either of the two Gaussian Process models proposed, the test-time complexity is $\mathcal{O}(KD)$, with K taken to be considerably smaller than N. This is specifically desirable as it provides substantially faster test-time predictions.

Table 2 displays real runtime estimated when predicting on a single image in the Pascal VOC-2007 dataset. These estimates are obtained when using 500 cluster centers. We also show time estimates when using 1500 samples rather than 500 cluster centers. The proposed method based on only clustering is able to perform inference considerably faster than the standard Gaussian Process model while being more accurate. The subsequent two models – *Meric-diag* and *Metric-full* – based on metric learning with diagonal and full covariances, respectively, are less than 1 s slower per image than the clustering based model while further boosting the accuracy.

6. Experimental evaluation

The proposed Gaussian Process models are not restricted to the problem of box regression and can be applied to any task with numerous training samples. To demonstrate the generality of the proposed extensions, we validate the Gaussian Process model choices and the model formulation in **Exp 1**. This is done on an independent machine learning regression dataset – the Airfoil Self-Noise Dataset of NASA Lichman (2013). In **Exp 2** we evaluate the performance of the advanced Gaussian Process models on the box scoring problem. **Exp 2.1** analyzes the features used. The choice of using the Gaussian Process regressors versus linear and non-linear SVR (Support Vector Regression), as well as the large-scale Gaussian Process model of Bo and Sminchisescu (2012), is validated in **Exp 2.2. Exp 2.3** supports the ability of performing self-assessment for individual object proposal algorithms. **Exp 2.4** evaluates the

Table 3
RMSE on the Self-Noise Dataset of NASA Lichman (2013), for a baseline Least Squares (LS) regressor on all training samples, standard Gaussian Process on all samples, *GP-Cluster* — Gaussian Process trained on 300 clusters centers, *GP-Metric diag/full* — the metric-learning kernel version of Eq. 4 on 300 clusters. (We show in bold where the methods are better than the baselines and underline the best method.)

	GP	Least	GP models		
	Std.	Squares	Cluster	M. diag	M. full
#Samples	1503	1503	300	300	300
RMSE	3.29 dB.	4.88 dB.	3.63 dB.	**3.18** dB.	<u>**2.86**</u> dB.

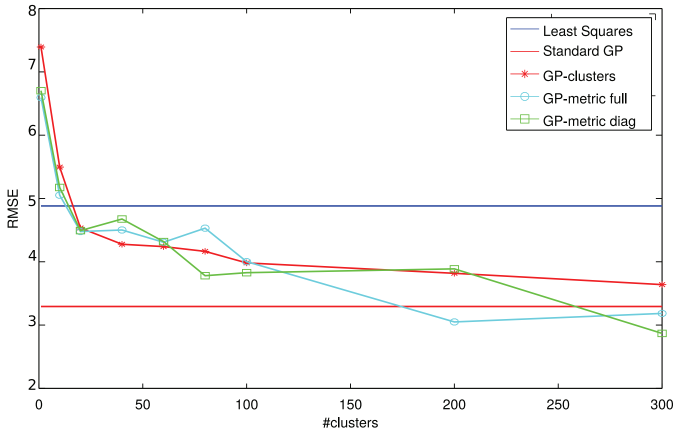


Fig. 4. Plots of the changes in RMSE with respect to varying numbers of clusters for the 2 baselines trained on all training samples — Least Squares (LS) and standard Gaussian Process, and the three Gaussian Process variants we have proposed: *GP-Cluster, GP-Metric diag* and *GP-Metric full*.

Gaussian Process models on the task of scoring object proposals based on box consistency. The consistency between proposals is integrated in the definition of the features as a manner of combining the multiple object proposal algorithms considered. We compare the results of our method of combining proposals of all seven algorithms with the interleaved baseline — where at each position the best box is picked out of the seven algorithms based on the provided ranking, if any. And we additionally compare with the best performing method in terms of the goodness of proposed boxes (Table 1) — *edge-boxes* (Zitnick and Dollár, 2014).

6.1. ***Exp 1***: *Analysis of model choices*

This experiment tests our model choices on an independent machine learning dataset — the Airfoil Self-Noise Dataset of NASA Lichman (2013). The dataset comprises 1503 data samples. The features represent five different statistics of airfoils such as size, frequency and speed and outputs are sound pressure levels, in decibels. We shuffle the data keeping half for training and the other half for testing. For this experiment we use all training samples, thus no QWS sampling is applied. We report the performance with respect to varying numbers of clusters. This experiment is designed to support the generality of the proposed Gaussian Process adaption. The model variations advanced in this paper are not restricted only to the problem of box regression. They can be applied to numerous problems where the number of training samples is prohibitively large.

Table 3 depicts the results on this dataset obtained by the Least Squares (LS) regressor as a baseline. We show as well the Gaussian Process trained on the full training data as the upper bound. We compare the clustering Gaussian Process, with the two variants of metric-learning Gaussian Process — with diagonal $\mathbf{\Sigma}$ and full $\mathbf{\Sigma}$. Fig. 4 shows how the performance varies with respect to the number of clusters. The results indicate that, indeed, incorporating the target variance by reshaping the kernel space is beneficial. This is true, as the metric-learning models improve over the Gaussian Process model using the same number of clusters. Moreover, they attain similar performance to the standard Gaussian Process model using the complete training set, yet the proposed models use only 300 samples — cluster centers.

6.2. ***Exp 2***: *Gaussian Process models for box scoring*

In this experiment we evaluate the performance of the developed Gaussian Process model variations on Pascal-VOC2007 for the task of box scoring. The purpose of this experiment is to verify the suitability of the approaches brought forth by this work, in the context of estimating box goodness. In the introduced models we use the QWS box-sampling to retain a number of 100 boxes per box-proposal algorithm from 500 randomly selected training images. We subsequently cluster the statistics used as features — Eq. 1 — into 500 clusters. This setting represents our starting model — *GP-Cluster*. The *GP-Metric* uses the same cluster centers as training data, yet it learns the appropriate kernel distances from 100 additional boxes per iteration in the SGD mini-batches, sampled using QWS. For the box-scoring task we only use the metric-learning Gaussian Process model with an associated diagonal $\mathbf{\Sigma}$ as this is more efficient. Given that we rank the boxes of all seven algorithms, we perform an additional NMS (non maximum suppression) at 0.7 overlap threshold over the scores to remove near duplicates generated by different algorithms. We also apply this step for all methods we compare against.

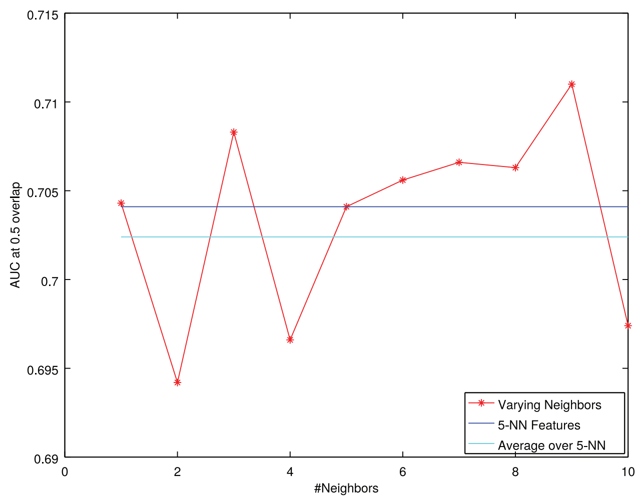
6.2.1. ***Exp 2.1***: *Box feature analysis*

In order to describe the consistency in box prediction, we estimate the overlap between boxes of all 7 considered algorithms. For each box we retain the closest neighbors in all seven algorithms and use the three overlap scores of Eq. 1 to define the features. In this experiment we test the effect of the number of neighbors considered in the feature computation on the overall performance.

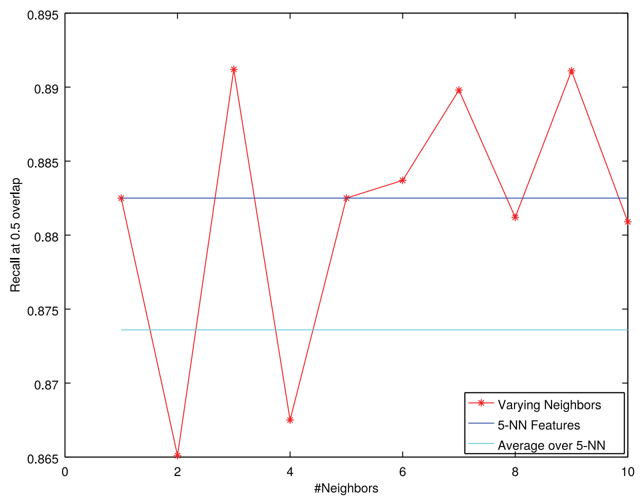
We plot the change in recall as well as the change in the AUC scores when we vary the number of neighbors from 1 to 10. Fig. 5 depicts these scores at a 0.5 overlap threshold with the ground truth. We additionally plot the performance when the average over five neighbors is considered per algorithm — giving rise to a 7 *D* feature vector. The scores when considering five neighbors is depicted separately, as this is taken to be our default setting in the subsequent experiments. It can be observed the AUC scores vary 2% when considering different numbers of neighbors in the feature computation, while the recall varies 3%. The setting considering five neighbors attains an average performance and at the same time it retains the feature dimensions within reasonable bounds — using ten neighbors rather than five does not bring substantial gain, yet it increases the feature dimensions twofold, fact which affects the kernel matrix computation, and thus the runtimes.

6.2.2. ***Exp 2.2***: *Gaussian Process models vs. other regressors*

The goal of this experiment is to test the performance of the proposed Gaussian Process models — based on sample selection, clustering and metric learning — when compared to other regressors trained either on the same cluster centers or on randomly selected samples. We inspect, therefore, a linear regressor, the linear-SVR , as well as the non-linear counterpart of it. In the non-linear SVR we choose the RBF kernel as this is closely related to the squared exponential kernel used in the Gaussian Process formulation. We train the SVR models on $\approx$ 1500 random samples. We additionally compare our models with two Gaussian process



(a) AUC scores w.r.t. the number of neighbors.



(b) Recall w.r.t. the number of neighbors.

Fig. 5. Recall and AUC scores at 0.5 overlap threshold, with respect to the number of neighbors considered in the feature computation. We plot separately the case when 5 neighbors are used in the feature computation, as this is the standard setting in our experiments. Overall, the considered number of neighbors seems to have limited influence on the performance of the method, as the ranges have limited variance both in terms of AUC scores as well as recall.

Table 4
Box regression results on the Pascal-VOC2007 dataset for different regression baselines and different training selection methods. We report recall and AUC at top 500 and 1K boxes with 0.5 overlap. (We show in bold where the proposed Gaussian Process model extensions outperform the baselines and underline the best method.). The new Gaussian Process models proposed in this paper are more suitable for performing the box scoring task while using a smaller set of samples to define the kernel matrix.

	Linear SVR		RBF-SVR		GP baselines		Proposed GP models	
	Standard	Cluster	Standard	Cluster	Standard	Large scale Bo and Sminchisescu (2012)	Cluster	Metric-diag.
#Samples	≈ 1500	500	≈ 1500	500	≈ 1500	≈ 1500	500	500
Recall @ 1K	87.09%	88.86%	90.30%	92.51%	89.48%	93.62%	**94.12**%	**94.73**%
Recall @ 500	78.71%	81.80%	83.60%	86.15%	81.03%	88.62%	**89.25**%	**89.52**%
AUC @ 1K	69.97%	72.45%	71.91%	74.38%	71.99%	73.76%	**74.47**%	**74.80**%
AUC @ 500	63.81%	66.80%	66.80%	69.52%	65.00%	69.66%	**70.41**%	**71.03**%

baseline: the standard Gaussian Process on ≈ 1500 randomly selected samples — *GP-Standard* — instead of on the 500 cluster centers, as well as the large scale Gaussian Process method of Bo and Sminchisescu (2012), also trained on ≈ 1500 randomly selected samples. The strength of Bo and Sminchisescu (2012), is in the ability to retain all training samples, and still perform the optimization, therefore for this method we use 3 × more data than for our proposed methods. The first Gaussian Process baseline validates the proposed way of defining training features based on sampling and clustering, while the second Gaussian Process baseline evaluates the performance of our method as a large scale Gaussian Process regression method. We additionally evaluate the performance of the SVR regressors when trained on samples selected as proposed in this paper: QWS sampling and K-means clustering. We do so in order to test the choice of the non-linear regressor, independent of the sample-selection.

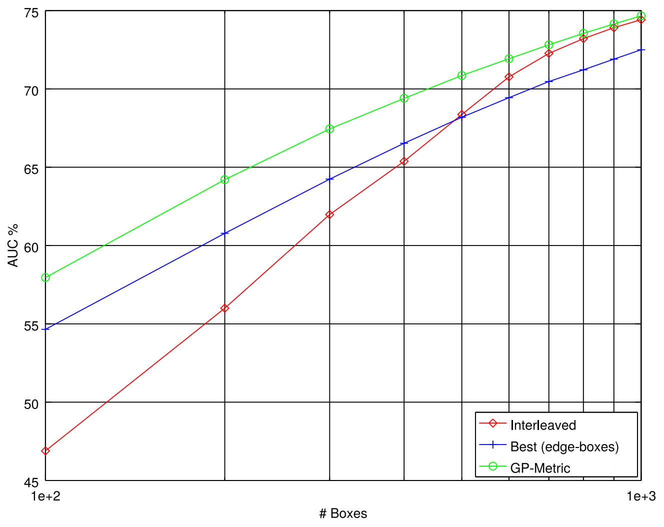
Table 4 depicts the achieved recall and AUC at top 500 boxes and 1K boxes. As seen from the results, the proposed sampling and clustering is highly effective. Regardless of the choice of the regressor, this achieves an improvement in the recall of 2% to 3%. Moreover, for the Gaussian Process case, the improvement achieved by clustering is more substantial: 5% and 8%. What is even more advantageous in our box selection method is the fact that these gains are achieved while training on a third of the data. When compared to the standard case, we use only 500 cluster centers instead of 1500 random samples. This is an important gain as, at test time, the Gaussian Process prediction has a complexity $\mathcal{O}(ND)$ (where N is training data and D are data dimensions). So with the proposed training data selection in the Gaussian Process model, we gain a 3 × computational speedup at test time and an additional 5% to 8% recall improvement. Furthermore, when performing the data selection as proposed in this paper by applying QWS and clustering, the Gaussian Process regressor proves to be the most appropriate for the box-scoring task. Table 4 shows that both recommended Gaussian Process models — the *GP-Cluster* and the *GP-Metric* — outperform the linear and non-linear SVR regressors.

As we argued that the task of box-scoring is highly non-linear, it is to be expected that the linear SVR is outperformed by the other regressors. The RBF-SVR and the Gaussian Process rely on the same non-linear kernel. The only difference is that in the Gaussian Process, the kernel distances are reshaped during training to better fit the data. This provides more descriptive power to the Gaussian Process model and explains the obtained performance gain, for the box-scoring task, when compared with its discriminative counterpart — the RBF-SVR.

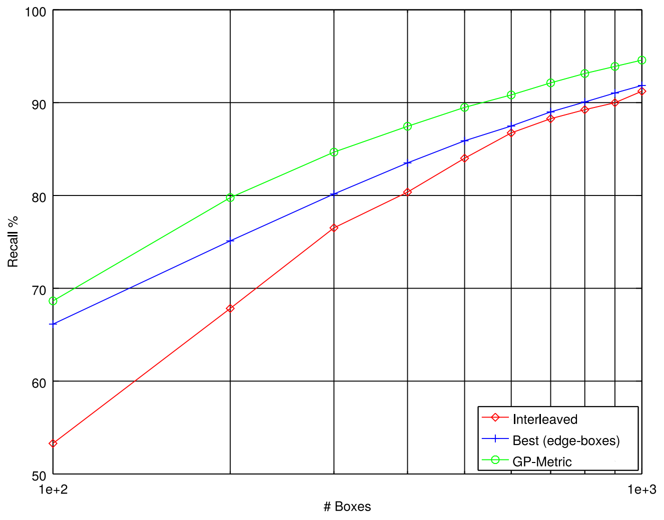
When comparing the proposed models with the large scale baseline of Bo and Sminchisescu (2012), we observe that the proposed methods are slightly more accurate while using a smaller number of training samples and, thus, a smaller training/test kernel matrix. The fact that our proposed Gaussian Process methods outperform the results of Bo and Sminchisescu (2012), on this data, validates our models as large scale regression models. We, further, conclude that both developed Gaussian Process methods are able to discover the underlying structure present in the data. Our metric-learning based Gaussian Process model performs on par with our proposed clustering-based method. This drives us to the conclusion that the target variance is not substantially present in the data. Therefore, the more simple model, employing a scalar

Table 5
Evaluation of the ability to perform self-assessment within the proposed GP regression method. We depict the scores at top 1K boxes when using the provided scores (if any) versus the ranking obtained my applying the box-goodness learning based on GP regression with metric learning. We show in bold where our ranking method exceeds the original scores.

	Recall @ 500		AUC @ 500	
Method	Provided	GP-Metric	Provided	GP-Metric
Core Rahtu et al. (2011)	70.76%	**76.37**%	59.06%	**62.40**%
Objectness Alexe et al. (2010)	84.14%	**86.90**%	59.78%	**61.32**%
Prim Manen et al. (2013)	82.38%	**87.14**%	66.41%	**69.06**%
SSE Uijlings et al. (2013)	85.53%	**88.09**%	66.37%	**70.36**%
Bing Cheng et al. (2014)	87.49%	**90.07**%	59.19%	**61.48**%
Edge-Boxes Zitnick and Dollár (2014)	85.89%	**90.82**%	68.19%	**70.82**%
Geodesic Krähenbühl and Koltun (2014)	88.84%	**89.61**%	70.28%	**71.34**%



(a) AUC scores w.r.t. the number of boxes.



(b) Recall w.r.t. the number of boxes.

Fig. 6. Recall and AUC scores with respect to the number of boxes, at 0.5 overlap threshold with the ground truth. We evaluate on Pascal-VOC2007 the developed Gaussian Process model based on metric learning, compared with the interleaved baseline — selecting the best box at each position, out of each of the seven algorithms, based on their provided ordering and, the best method in terms of proposed good boxes (according to Table 1) — *edge-boxes* (Zitnick and Dollár, 2014).

lengthscale rather than a diagonal covariance, is sufficient for tackling the box-scoring problem.

6.2.3. ***Exp 2.3***: *Object proposals self-assessment*

One of the claimed gains of the proposed box-scoring method is the ability of performing self-assessment for any object proposal algorithms. In this experiment we test precisely this claim. Thus we train on only boxes of a fixed reference algorithm. We still sampled the boxes using the QWS method, followed by the K-mean clustering. The only difference with the previous experiment is that the boxes used for training come from the proposal algorithm to be evaluated. The features are, however, defined as before by looking at the consistency with the five closest neighbors in the other proposal algorithms. Subsequently, at test time we only score the boxes of the evaluated proposal method by looking at the consistency with the other algorithms.

Table 5 displays the recall and AUC scores at 0.5 overlap threshold on the top 500 boxes on Pascal-VOC2007. We compare the *GP-Metric* — Gaussian Process trained through metric learning for estimating the diagonal covariance matrix — with the scores obtained when using the ranking provided by the proposal algorithms. If no scores are provided, as in the case of *prim* and *selective search*, we randomly shuffle the boxes and then evaluate the performance. For all methods there is a substantial gain in performance — up to 6% in recall and up to 4% in AUC — when employing the proposed method. A considerable gain is obtained by performing self-assessment on the *edge-boxes* method. This is due to this method having more precise boxes present in the list of predicted boxes (see Table 1). The introduced box-scoring method aims at giving higher scores to those good boxes. For the *geodesic* object proposal method the gain is not substantial when compared to the provided ranking. The *geodesic* method produces a small number of proposals to start with, 653 on average per image, while in Table 5 we evaluate the AUC and Recall at 500 boxes. Overall, we can observe that the proposed method of box-scoring is effective in practice and it is useful when scoring boxes of individual algorithms.

6.2.4. ***Exp 2.4***: *Combining object proposals*

The ambition of this paper is in developing a method for scoring object proposals based solely on the overlap information with other boxes within the same algorithm as well as other algorithms. For this we use two baselines to compare against: *Interleaved* — selecting the best box from each algorithm greedily at each position, *Best* — the best performing algorithm in terms of goodness of proposed boxes, which as seen in Table 1 is the *edge-boxes* (Zitnick and Dollár, 2014).

Fig. 6(a) displays the change in the AUC with the number of retained boxes for the metric-learning based Gaussian Process variant proposed in this paper — *GP-Metric* — and the two baselines: *Interleaved* and *Best*. We can notice that the Gaussian Process method outperforms the other two methods in terms of AUC at already only 100 boxes retained. The tendency remains stable as the number of boxes increases. The Gaussian Process model is on average 2% more precise in the AUC scores than the *Best*. The

Interleaved method is less precise than the *Best* for a smaller number of considered boxes. However, it gains in performance as the number of boxes increases. This is to be expected as the best box per algorithm is not necessarily the best box over all algorithms. The *Interleaved* method is characterized by more diversity, yet the proposed box-scoring model outperforms both these methods.

For the recall, we can observe a similar tendency — Fig. 6(b) — the proposed Gaussian Process regression model being on average more precise than the *Best*, regardless of the number of considered boxes. Finally, we notice that at 1000 boxes the *Interleaved* method slightly outperforms *Best*. We argue that this is due to the *Interleaved* baseline being characterized by more diversity in the proposals. And this brings a gain in the performance as the number of boxes increases. Moreover, the proposed box-scoring regression method outperforms both baselines in terms of recall. This is due to the boxes being more precise — more likely to be centered on true objects — as it is the case with the *Best* baseline, and more diverse boxes, as it is the case with the *Interleaved* baseline.

7. Discussion and illustrative results

This work proposes a manner of assigning goodness scores to boxes. We start with the idea that consistency in box proposals — high overlap between proposals of different algorithms — is a sufficient indication of box goodness. In Fig. 7 we show a few examples of top-10 ranked boxes for the two considered box-scoring baselines, *Interleaved* and *Best*, and our *GP-Metric* method. In green we display the ground truth boxes. In blue we indicate the boxes out of the top-ten ranked ones that overlap more than 0.5 with a ground truth box. In red we show the non-overlapping boxes. The first row indicates a failure case for our method: no ground truth box is present in the top-ten ranked boxes. This is due to the fact that, for this case, the top-ten boxes tend to focus on object parts rather than complete objects. On the second row we can observe that our method manages to find, within the first ten proposals, one out of the two people present. Also noteworthy is the fact that it correctly finds the human faces as good proposals. Similarly, in the example with the car, we notice that the highly ranked boxes for our method contain parts of the car such as the license plate and the wheels. The *Interleaved* baseline seems to have a preference for large boxes in the top-ten ranked boxes, yet we can also observe a few small ones. The *Interleaved* baseline takes greedily the best box per algorithm at each position. Therefore, intuitively, it is characterized by more diversity in the proposals. The *Best* (*edge-boxes*) baseline opts for medium-sized boxes that focus more on the textured part of the image. Thus, it is more precise as it tends to be more object-focused. The highly ranked boxes in our case are more diverse, as both small boxes as well as large boxes are present in the top ten. Additionally, they are also characterized by more precision, since they rely on the consistency in the proposals. Therefore, if more algorithms select a certain area of the image as likely to contain an object, the box corresponding to that area will be assigned a high score in our method. This explains why for the proposed *GP-Metric* method we observe also the object parts being highly ranked.

From the seven considered proposal algorithms, two of them do not provide scores — prim (Manen et al., 2013), selective search (Uijlings et al., 2013). As seen from ***Exp 2.3:****Object Proposals Self-Assessment*, the proposed regression models can be used as a manner of performing self-assessment for methods that do not provide a way to do so. For the algorithms that provide associated scores to boxes, however, a possible approach to integrating existing box scores in the learning of box goodness is transforming these scores into probabilities and using them as priors. Alternatively, as in Karaoglu et al. (2014), the scores can be included as part of the feature in the feature vector or as in Xu et al. (2014), where the individual scores are combined using theory of belief functions. Noteworthy is that in Karaoglu et al. (2014) and Xu et al. (2014), the goal is combining detection scores, thus these scores are class specific. In our case the scores do not correspond to class confidences but rather, presumed box-goodness scores. This fact makes the scores more unreliable in our case. Therefore, given that not all the algorithms provide scores and moreover that these scores are unreliable, in this work we choose not to make use of this information.

The selection of the seven box-proposal algorithms is a design choice based on their popularity as well as their characteristics — speed, performance. Ideally, the considered box proposal algorithms should be orthogonal to each other, employing complementary information. The question that arises is: how many different definitions of what makes a good object proposal we know? The literature offers three main ones: (i) an object is enclosed by a strong edge (Cheng et al., 2014; Krähenbühl and Koltun, 2014; Zitnick and Dollár, 2014); (ii) an object is salient (Alexe et al., 2010; Rahtu et al., 2011); (iii) an object is composed of similar parts (Manen et al., 2013; Uijlings et al., 2013). However, using correlated proposal methods is also beneficial as it adds to the robustness of the system. Although certain methods start from the same principle, they implement it differently and their combination gives more stable predictions.

Noteworthy, in our approach the precise choice of the object proposal algorithms is not essential. The use of the seven selected methods is a design choice and any of them can be removed, replaced with another, or box-proposal methods can be added without the need to change the mathematical definition of the regression model or its applicability.

8. Conclusions

This paper starts with the assumption that the consistency between the proposed boxes of different state-of-the art algorithms, is revealing as to how good a certain box is. The considered object proposal algorithms rely on different cues, which makes their naive combination able to achieve a recall close to one. We develop an addition over the standard Gaussian Process model by learning the kernel shape in a metric learning framework through loss optimization. The optimization enables us to keep the kernel-matrix size fixed, while using as much as possible of the information provided by the additional samples. We find that on the problem of box regression, both the simpler *GP-Cluster* approach and the metric-learning Gaussian Process models, capture the correlations in the data. Experiment **Exp 2.1** evaluates the influence of the number of neighbors considered in the feature definition. Additionally, we experimentally prove the suitability of the clustering and metric-learning Gaussian Process models, when compared with other regressors on the box scoring problem — **Exp 2.2**. We show the ability of performing self-assessment for individual object proposal methods in **Exp 2.3**. We prove experimentally — **Exp 2.4** — that features capturing the overlap are sufficiently descriptive for evaluating box goodness. **Exp 1** shows the effectiveness of the metric-learning Gaussian Process models on an independent regression problem as well as the suitability of the methods as general purpose large scale models.

The idea of considering only the overlap between boxes for scoring existing proposals can be extended to other similar problems such as: pedestrian detection (Dollár et al., 2012), where multiple detector predictions are available, or in the context of object tracking (Smeulders et al., 2014). Furthermore, the three proposed variations of the Gaussian Process model are suitable for a multitude of problems where the number of training samples is prohibitively large.

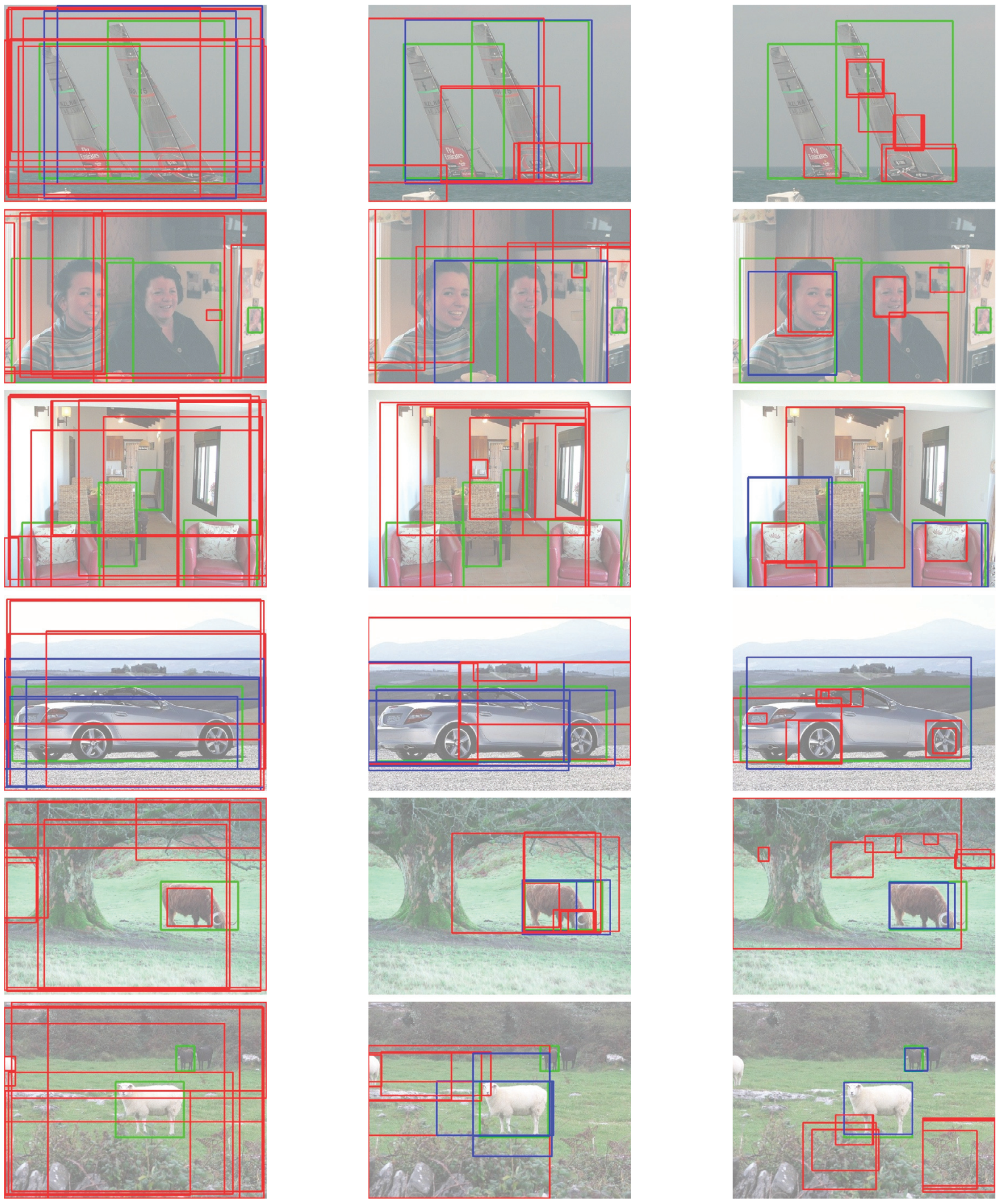


Fig. 7. Illustrative results of top ten boxes per image for the three considered methods: *Interleaved* — where the boxes of all seven algorithms are interleaved based on the provided ranking, *Best* — the method acquiring the best recall, *edge-boxes* (Table 1), and *GP-Metric* — assigning scores to boxes based on the consistency in the overlap. We show in green the ground truth boxes, in blue the boxes, out of the 10 ones retained, that have over 0.5 overlap with the ground truth and in red the ones that are not meeting the 0.5 overlap criterion. The first row displays a failure case where the boats are missed, yet parts of them are selected. The *Interleaved* method has a preference for large boxes being ranked higher in the list. Compared to the two baselines, the proposed approach gives rise to more diverse boxes in the top-ranked ones, as both small and large boxes are present in the top 10. Moreover, our method focuses on both object parts — i.e., faces of people, wheels of the cars, as well as entire objects.

Acknowledgments

Many thanks to Pascal Mettes for the countless useful comments on the paper. Thanks to Morris Franken for the help with visualizations and Dennis Koelma for technical support.

Research supported by the Dutch national program COMMIT.

Appendix A. Metric Learning Derivations

The standard Gaussian Process model definition estimates at test-time the kernel similarities between the input test sample, $\mathbf{x}^*$, and training samples, $\mathbf{X}$. In our case, these training samples represent cluster centers. The test-time kernel distances are weighed by the vector $\boldsymbol{\alpha}$, learned during training.

$$y^* = k_{\boldsymbol{\Sigma}}(\mathbf{x}^*, \mathbf{X})^T \boldsymbol{\alpha}. \tag{A.1}$$

$$\boldsymbol{\alpha} = (k_{\boldsymbol{\Sigma}}(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathbf{I})^{-1} \mathbf{y}. \tag{A.2}$$

$$k_{\boldsymbol{\Sigma}}(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{1}{2}(\mathbf{x}_i - \mathbf{x}_j)\boldsymbol{\Sigma}(\mathbf{x}_i - \mathbf{x}_j)^T\right). \tag{A.3}$$

As presented in Section 3, rather than a scalar length-scale, we use the covariance matrix, $\boldsymbol{\Sigma}$, in the kernel function definition.

In order to determine the covariance matrix, $\boldsymbol{\Sigma}$, we consider the squared loss and we optimize this by computing the gradient with respect to it and iteratively updating the parameters in an SGD optimization. Given that $\boldsymbol{\Sigma}$ is symmetric, we use the derivative formulation for symmetric matrices – Eq. (A.4).

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}} = \left[\frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}}\right] + \left[\frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}}\right]^T - \text{diag}\left[\frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}}\right]. \tag{A.4}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}} = 2\sum_n^{\text{train}}(y_n - y_n^*)\frac{\partial y_n^*}{\partial \boldsymbol{\Sigma}} + \lambda \mid \boldsymbol{\Sigma} \mid_2 . \tag{A.5}$$

$$\frac{\partial y_n^*}{\partial \boldsymbol{\Sigma}} = \sum_i^{\text{clusters}} \frac{\partial \alpha_i}{\partial \boldsymbol{\Sigma}} k_{\boldsymbol{\Sigma}}(\mathbf{x}_n, \mathbf{x}_i) + \alpha_i \frac{\partial k_{\boldsymbol{\Sigma}}(\mathbf{x}_n, \mathbf{x}_i)}{\partial \boldsymbol{\Sigma}}. \tag{A.6}$$

The predictive distribution of the Gaussian Process, as see in Eq. (A.1) has two components: the test-time kernel and the vector $\boldsymbol{\alpha}$. We use the product rule and estimate the derivative for each term separately.

The first term involves computing the gradient of α_i with respect to the covariance matrix. This is a function that depends on the train-time targets, y_i, and train-time kernel values, $\mathbf{K}_{ij} = k_{\boldsymbol{\Sigma}}(\mathbf{x}_i, \mathbf{x}_j)$, where $\mathbf{x}_i$ and $\mathbf{x}_j$ are cluster centers.

$$\begin{aligned}\frac{\partial \alpha_i}{\partial \boldsymbol{\Sigma}} &= \sum_j^{\text{clusters}} \frac{\partial (\mathbf{K}_{ij} + \sigma^2 \mathbf{I}_{ij})^{-1}}{\partial \boldsymbol{\Sigma}} y_i \\ &= \sum_j^{\text{clusters}} \left[-(\mathbf{K}_{ij} + \sigma^2 \mathbf{I}_{ij})^{-1} \frac{\partial \mathbf{K}_{ji}}{\partial \boldsymbol{\Sigma}} (\mathbf{K}_{ji} + \sigma^2 \mathbf{I}_{ji})^{-1}\right] y_i. \end{aligned} \tag{A.7}$$

The gradient of the train kernel depends on the covariance matrix $\boldsymbol{\Sigma}$, as derived from Eq. (A.3).

$$\frac{\partial \mathbf{K}_{ji}}{\partial \boldsymbol{\Sigma}} = \frac{\partial k_{\boldsymbol{\Sigma}}(\mathbf{x}_j, \mathbf{x}_i)}{\partial \boldsymbol{\Sigma}}. \tag{A.8}$$

$$\frac{\partial \mathbf{K}_{ji}}{\partial \boldsymbol{\Sigma}} = \frac{\partial \exp\left(-\frac{1}{2}(\mathbf{x}_j - \mathbf{x}_i)\boldsymbol{\Sigma}(\mathbf{x}_j - \mathbf{x}_i)^T\right)}{\partial \boldsymbol{\Sigma}}. \tag{A.9}$$

$$= -\frac{1}{2}(\mathbf{x}_j - \mathbf{x}_i)^T(\mathbf{x}_j - \mathbf{x}_i) k_{\boldsymbol{\Sigma}}(\mathbf{x}_j, \mathbf{x}_i). \tag{A.10}$$

The derivative of the kernel has the same formulation for test and train. At test time the only difference is that rather than having both samples represent cluster centers, one of them is the cluster center while the second represents the input test sample, $\mathbf{x}_*$.

$$\frac{\partial k_{\boldsymbol{\Sigma}}(\mathbf{x}_*, \mathbf{x}_i)}{\partial \boldsymbol{\Sigma}} = -\frac{1}{2}(\mathbf{x}_* - \mathbf{x}_i)^T(\mathbf{x}_* - \mathbf{x}_i) k_{\boldsymbol{\Sigma}}(\mathbf{x}_*, \mathbf{x}_i). \tag{A.11}$$

Putting all partial derivatives back into the formulation of the gradient of the loss function with respect to the covariance, $\boldsymbol{\Sigma}$, we obtain the complete derivation:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}} &= 2\sum_n^{\text{train}}(y_n - y_n^*)\left[\sum_i^{\text{clusters}}\left[\sum_j^{\text{clusters}} -K_{ij}^{inv}\mathbf{M}_{ji}K_{ji}^{inv}\right.\right. \\ &\quad y_i k_{\boldsymbol{\Sigma}}(\mathbf{x}_n, \mathbf{x}_i) + \alpha_i \mathbf{M}_{ni}\Big] + 2\lambda\boldsymbol{\Sigma}, \end{aligned} \tag{A.12}$$

$$\mathbf{M}_{ji} = -\frac{1}{2}(\mathbf{x}_j - \mathbf{x}_i)^T(\mathbf{x}_j - \mathbf{x}_i) k_{\boldsymbol{\Sigma}}(\mathbf{x}_j, \mathbf{x}_i), \tag{A.13}$$

$$\mathbf{K}^{inv} = (k_{\boldsymbol{\Sigma}}(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathbf{I})^{-1}. \tag{A.14}$$

References

Alexe, B., Deselaers, T., Ferrari, V., 2010. What is an object? CVPR.
Bishop, C.M., 2006. Pattern recognition and machine learning. Springer, Amsterdam, Netherlands.
Bo, L., Sminchisescu, C., 2012. Greedy block coordinate descent for large scale gaussian process regression. CoRR arXiv:1206.3238.
Bottou, L., 2007. Large-scale kernel machines. MIT Press, Amsterdam, Netherlands.
Bottou, L., 2012. Stochastic gradient descent tricks. Neural Networks: Tricks of the Trade.
Cao, Y., Brubaker, M.A., Fleet, D., Hertzmann, A., 2013. Efficient optimization for sparse gaussian process regression. NIPS.
Chavali, N., Agrawal, H., Mahendru, A., Batra, D., 2015. Object-proposal evaluation protocol is' gameable'. CoRR arXiv:1505.05836.
Cheng, M.-M., Zhang, Z., Lin, W.-Y., Torr, P., 2014. Bing: Binarized normed gradients for objectness estimation at 300fps. CVPR.
Csató, L., Opper, M., 2002. Sparse on-line gaussian processes. Neural Comput 14, 641–668.
Dollár, P., Wojek, C., Schiele, B., Perona, P., 2012. Pedestrian detection: An evaluation of the state of the art. PAMI 34, 743–761.
Erhan, D., Szegedy, C., Toshev, A., Anguelov, D., 2014. Scalable object detection using deep neural networks. CVPR.
Girshick, R., 2015. Fast r-cnn. CoRR.
Girshick, R., Donahue, J., Darrell, T., Malik, J., 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. CVPR.
Hensman, J., Fusi, N., Lawrence, N.D., 2013. Gaussian processes for big data. CoRR.
Hosang, J., Benenson, R., Schiele, B., 2014. How good are detection proposals, really? BMVC.
Huang, R., Sun, S., 2013. Kernel regression with sparse metric learning. JIFS 24, 775–787.
Kalal, Z., Matas, J., Mikolajczyk, K., 2008. Weighted sampling for large-scale boosting. BMVC.
Karaoglu, S., Liu, Y., Gevers, T., 2014. Detect2rank : Combining object detectors using learning to rank. CoRR 25, 233–248.
Karianakis, N., Fuchs, T.J., Soatto, S., 2015. Boosting convolutional features for robust object proposals. CoRR.
Kostinger, M., Hirzer, M., Wohlhart, P., Roth, P.M., Bischof, H., 2012. Large scale metric learning from equivalence constraints. CVPR.
Krähenbühl, P., Koltun, V., 2014. Geodesic object proposals. ECCV.
Lawrence, N., Seeger, M., Herbrich, R., 2003. Fast sparse gaussian process methods: The informative vector machine. NIPS.
Lichman, M., 2013. UCI machine learning repository.
Manen, S., Guillaumin, M., Gool, L.V., 2013. Prime object proposals with randomized prim's algorithm. ICCV.
Neal, R.M., 2012. Bayesian learning for neural networks, Vol. 118. Springer, Amsterdam, Netherlands.
Nguyen-Tuong, D., Peters, J.R., Seeger, M., 2009. Local gaussian process regression for real time online model learning. NIPS.
Quiñonero-Candela, J., Rasmussen, C.E., 2005. A unifying view of sparse approximate gaussian process regression. JMLR 6, 1939–1959.
Rahtu, E., Kannala, J., Blaschko, M., 2011. Learning a category independent object detection cascade. ICCV.
Ranganathan, A., Yang, M.-H., Ho, J., 2011. Online sparse gaussian process regression and its applications. TIP 20, 391–404.
Rasmussen, C.E., 2006. Gaussian processes for machine learning. Citeseer, Amsterdam, Netherlands.
Ren, S., He, K., Girshick, R., Sun, J., 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. CoRR.
Smeulders, A.W., Chu, D.M., Cucchiara, R., Calderara, S., Dehghan, A., Shah, M., 2014. Visual tracking: an experimental survey. PAMI 36, 1442–1468.
Snelson, E., Ghahramani, Z., 2005. Sparse gaussian processes using pseudo-inputs. NIPS.
Snelson, E., Ghahramani, Z., 2007. Local and global sparse gaussian process approximations. AISTATS.
Sutskever, I., Martens, J., Dahl, G., Hinton, G., 2013. On the importance of initialization and momentum in deep learning. ICML.
Titsias, M., Lázaro-Gredilla, M., 2013. Variational inference for mahalanobis distance metrics in gaussian process regression. NIPS.
Titsias, M.K., 2009. Variational learning of inducing variables in sparse gaussian processes. AISTATS.
Uijlings, J.R., van de Sande, K.E., Gevers, T., Smeulders, A.W., 2013. Selective search for object recognition. IJCV 104, 154–171.
Urtasun, R., Darrell, T., 2008. Sparse probabilistic regression for activity-independent human pose inference. CVPR.
Vezhnevets, A., Ferrari, V., 2015. Looking out of the window: object localization by joint analysis of all windows in the image. CoRR 4.

Vivarelli, F., Williams, C.K., 1999. Discovering hidden features with gaussian processes regression. NIPS.
Weinberger, K.Q., Saul, L.K., 2009. Distance metric learning for large margin nearest neighbor classification. JMLR 10, 207–244.
Xing, E.P., Jordan, M.I., Russell, S., Ng, A.Y., 2002. Distance metric learning with application to clustering with side-information. NIPS.
Xu, P., Davoine, F., Denœux, T., Compiègne, F., 2014. Evidential combination of pedestrian detectors. BMVC.
Ying, Y., Li, P., 2012. Distance metric learning with eigenvalue optimization. JMLR 13, 1–26.
Zhao, Q., Liu, Z., Yin, B., 2014. Cracking bing and beyond. BMVC.
Zitnick, C.L., Dollár, P., 2014. Edge boxes: Locating object proposals from edges. ECCV.