Efficient Semantic Segmentation with Gaussian Processes and Histogram Intersection Kernels

Alexander Freytag Björn Fröhlich Erik Rodner
Joachim Denzler
Computer Vision Group, Friedrich Schiller University Jena
{*alexander.freytag,bjoern.froehlich,erik.rodner,joachim.denzler*}@*uni-jena.de*

Abstract

Semantic interpretation and understanding of images is an important goal of visual recognition research and offers a large variety of possible applications. One step towards this goal is semantic segmentation, which aims for automatic labeling of image regions and pixels with category names. Since usual images contain several millions of pixel, the use of kernel-based methods for the task of semantic segmentation is limited due to the involved computation times. In this paper, we overcome this drawback by exploiting efficient kernel calculations using the histogram intersection kernel for fast and exact Gaussian process classification. Our results show that non-parametric Bayesian methods can be utilized for semantic segmentation without sparse approximation techniques. Furthermore, in experiments, we show a significant benefit in terms of classification accuracy compared to state-of-the-art methods.

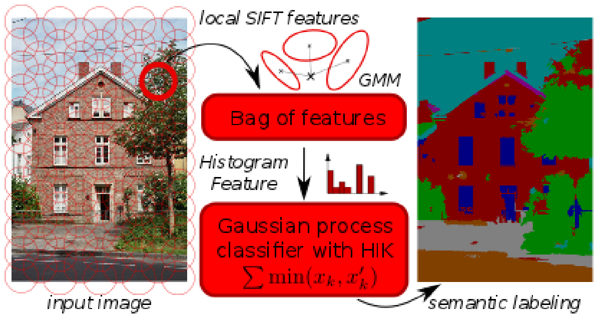

Figure 1. Our approach: histogram features allow for using histogram intersection kernels (HIK) and Bayesian inference with Gaussian processes.

1. Introduction

The objective of semantic segmentation is pixel-wise labeling of a given image, *i.e.*, each pixel is assigned to one of the learned classes. Semantic segmentation is an essential tool when accurate locations of objects are required, as needed in robotics, automotive applications, and street scene analysis. The large variability of the visual appearance (see Figure 1) requires a high model complexity. However, a common approach is to classify local features with only linear methods [2]. In contrast, we propose a semantic segmentation method using a Bayesian kernel method, which is able to learn complex non-linear models from thousands of local patches.

Kernel methods, *e.g.*, SVMs or Gaussian processes [7], come along with the drawback of time-consuming evaluations of the kernel functions. Especially for semantic segmentation with several thousands up to millions of examples for training and testing, these methods are not directly applicable. A key idea to allow for non-linear classification is to exploit the properties of the histogram intersection kernel (HIK) as done by Maji *et al.* [6] for fast SVM classification. Furthermore, Wu [11] showed how to speed up learning with SVMs and histogram intersection kernels. We adapt both ideas for Gaussian processes (GP) and show that the special properties of histogram intersection kernels are highly beneficial for fast and exact semantic segmentation with Gaussian process classification [8]. An outline of our approach is shown in Figure 1.

The remainder of this paper is organized as follows. We present our framework for semantic segmentation in Sect. 2. Short introductions to GP and fast kernel evaluations using HIK are given in Sect. 3 and 4. How to enable GP inference exploiting HIK properties is figured out in Sect. 5. Experimental results are shown in Sect. 6 highlighting the suitability of fast exact GP classification for the challenging task of semantic segmentation.

2. An outline for semantic segmentation

Semantic segmentation is a challenging task gaining more and more interest over the last years. Different ways for solving this problem exist, *e.g.*, incorporating texton features in a random decision forest framework [9] or applying conditional random fields to model global dependencies [5]. We focus on a more generic semantic segmentation method, which does not incorporate any additional model-based knowledge like: "a typical window is rectangular". Previous approaches [2, 3] propose a powerful framework for tackling this task, consisting of four steps: (1) *local feature extraction and clustering*, (2) *pixel-wise classification*, (3) *unsupervised segmentation*, and finally (4) the *combination of the classification results and unsupervised segmentation* (see Figure 1).

In step (1), local color and texture information are extracted in a local neighborhood on an equally spaced grid and in various scales [10]. Based on that, histogram features are computed using Gaussian mixture models. These histogram features are classified in step (2) using methods such as sparse logistic regression (SLR) [2]. In our work, we show how to use Gaussian Processes equipped with a histogram intersection kernel instead. An unsupervised segmentation is used in step (3) to get an over-segmented image. In step (4), the pixel-wise classification results on the dense grid are smoothed as introduced in [2]. For every region, the category with maximum average probability determines its class label and each pixel in a region is labeled identically. This leads to object boundaries aligned to image edges and a homogeneous object annotation.

3. GP regression and classification

We briefly review the GP framework before describing our modifications necessary to allow for efficient semantic segmentation with thousands of training examples. The goal of regression is to estimate a mapping f between input data, *e.g.*, D-dimensional feature vectors, and target values. Based on n training examples $\boldsymbol{x}^{(i)} \in \mathbf{X} \subset \mathcal{X}$ and their targets $y_i \in \mathbb{R}$, we are interested in predicting the unknown target value y_* of an unseen example $\boldsymbol{x}^* \in \mathcal{X}$. The Gaussian process framework belongs to the class of kernel methods and in contrast to SVM it can be derived completely by probabilistic instead of geometric assumptions. We assume f to be drawn from a stochastic process, *e.g.*, a Gaussian Process $f \sim \mathcal{GP}(0, K)$ with zero mean and covariance function K. If we expect the training data to be disturbed by additive Gaussian noise with variance σ^2:

$$p(y_i \mid f_i) = \mathcal{N}(y_i \mid f_i, \sigma^2) \ , \qquad (1)$$

the expected posterior value of y_* can be derived in a Bayesian manner [7] and requires $\mathcal{O}(n)$ operations:

$$\mu_* = \boldsymbol{k}_*^T (\mathbf{K} + \sigma^2 \cdot \mathbf{I})^{-1} \boldsymbol{y} = \boldsymbol{k}_*^T \boldsymbol{\alpha} \ . \qquad (2)$$

We use the notations $\boldsymbol{k}_*$ as vector containing the kernel values $(\boldsymbol{k}_*)_i = K(\boldsymbol{x}^{(i)}, \boldsymbol{x}^*)$ between the test example $\boldsymbol{x}^*$ and all training examples $\boldsymbol{x}^{(i)}$, $\mathbf{K}$ is the kernel matrix of the training data, and $\boldsymbol{y}$ is the vector containing all training labels.

For classification, we follow the suggestion of [4] and assume the discrete labels y_i to be generated by a real-valued function. Although neglecting the discrete nature of labels, this strategy leads to analytically tractable closed-form solutions. To perform multi-class classification, we use the standard one-vs-all approach [4] resulting in M binary classifiers. A final decision is achieved by maximizing over all class scores:

$$\mu_*^{mc} = \max_{c=1\ldots M} \boldsymbol{k}_*^T (\mathbf{K} + \sigma^2 \cdot \mathbf{I})^{-1} \boldsymbol{y}_c \ , \qquad (3)$$

where $\boldsymbol{y}_c$ is the binary label vector for class c.

4. Fast kernel evaluations using HIK

In computer vision applications, the histogram intersection kernel (HIK):

$$K^{\text{HIK}}(\boldsymbol{x}, \boldsymbol{x}') = \sum_{d=1}^{D} \min(x_d, x'_d) \ , \qquad (4)$$

is often used to compare histogram feature vectors $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^D$. It has been shown, that this kernel allows for efficient classification and learning with support vector machines (SVM) [6, 11]. We review the important details in the following and show how to perform efficient Gaussian process classification based on these ideas in Sect. 5 as recently done in [8].

Fast kernel calculation Similar to SVM, the GP posterior mean is a weighted sum of kernel values (see Eq. (2)). The HIK allows for decomposing the summation in two parts [6]:

$$\begin{aligned}\boldsymbol{k}_*^T \boldsymbol{\alpha} &= \sum_{i=1}^{n} \alpha_i \left(\sum_{d=1}^{D} \min(x_d^{(i)}, x_d^*) \right) \\ &= \sum_{d=1}^{D} \Bigg(\sum_{\{i: x_d^{(i)} < x_d^*\}} \alpha_i \, x_d^{(i)} + x_d^* \sum_{\{j: x_d^{(j)} \geq x_d^*\}} \alpha_j \Bigg) \ . \qquad (5)\end{aligned}$$

To enable fast calculations, we compute permutations π_d for each dimension d which rearrange the training examples of that dimension in an increasing order:

$$\boldsymbol{k}_*^T \boldsymbol{\alpha} = \sum_{d=1}^{D} \Bigg(\underbrace{\sum_{i=1}^{r_d} \alpha_{\pi_d^{-1}(i)} x_k^{(\pi_d^{-1}(i))}}_{\doteq A(d, r_d)} + x_d^* \underbrace{\sum_{i=r_d+1}^{n} \alpha_{\pi_d^{-1}(i)}}_{\doteq B(d, r_d)} \Bigg) \quad (6)$$

with r_d being the index of the first example in the ordered sequence of dimension d, which is larger than x^*_d. Obviously, a new test example has only n possible locations in each resulting sequence. If we precompute A and B in the training step, we just need to find the index r_d for every dimension to compute $\boldsymbol{k}_*^T\boldsymbol{\alpha}$.

Summarizing, for training we need $\mathcal{O}(Dn\log n)$ operations for a given vector $\boldsymbol{\alpha}$, which is dominated by the effort for sorting the examples in each dimension. Evaluating the score of a new example can be done in $\mathcal{O}(D\log n)$. In addition, we can speed up the computations exploiting sparsity of the data.

Quantization of the feature space To further speed up the previously presented techniques, we assume feature values of dimension d to be bounded within $x^*_d \in [l_d, u_d]$. By quantizing the feature space [6] using q bins, only q different outputs can occur in the inner sums of Eq. (6). Having A and B on hand, we can precompute a lookup table T of size $D \times q$ with $\mathcal{O}\left(D\max(n,q)\right)$ operations. As a result of that, we can evaluate $\boldsymbol{k}_*^T\boldsymbol{\alpha}$ by quantizing $\boldsymbol{x}^*$ in every dimension and adding up the corresponding entries of T. Therefore, classification can be done in $\mathcal{O}\left(D\right)$ operations, which is independent of the number of examples used for training. This strategy is especially appealing for the task of semantic segmentation, where the number of training examples is very large in general.

5. Efficient GP multi-class classification with fast HIK multiplications

In this section, we show how to utilize the previously presented techniques for fast and exact Gaussian process inference [8]. As a result of that, we obtain a full Bayesian model without the necessity of storing the kernel matrix of size $\mathcal{O}(n^2)$ since our memory requirement is $\mathcal{O}(Dn)$ and thus linear in the number of examples.

Efficient computation of weight vector $\boldsymbol{\alpha}$ As reviewed in Sect. 3, the weight vector $\boldsymbol{\alpha}$ for GP regression can be obtained by solving the linear equation system:

$$(\mathbf{K} + \sigma^2 \cdot \mathbf{I}) \cdot \boldsymbol{\alpha} = \boldsymbol{y} \ . \tag{7}$$

Similar to efficient calculations of $\boldsymbol{k}_*^T\boldsymbol{\alpha}$, we can also perform fast multiplications with the kernel matrix with the same ideas as presented in Sect. 4. As a result of that, we can apply an iterative linear solver, *e.g.*, the linear conjugate gradient (CG) method, to Eq. (7). Doing so, the asymptotic runtime is $\mathcal{O}(nD(T_1M + \log n))$ including the effort for sorting of training examples as mentioned before. We denoted with T_1 the number of iterations used for the CG method, which depends on the condition of the kernel matrix $\mathbf{K}$. If the maximum norm of the residual drops below 10^{-2}, we also stop the CG method, even if T_1 was not reached yet. Note that the runtime is linear in the number of classes known during training, which is a direct result of Eq. (3). After solving the linear system in Eq. (7), we can quantize the feature space and build the final lookup table T.

Table 1. Recognition rates of our experiments in comparison to previous work.

approach	ARR	ORR
	eTRIMS	
RDF [3]	58.67% (±3.14)	64.54% (±1.37)
SLR [3]	65.57% (±2.47)	**71.18**% (±2.69)
CRF [13]	49.75%	65.80%
HCRF [12]	61.63%	69.00%
GP-HIK	**68.68**% (±2.07)	68.02% (±2.10)
	LabelMeFacade	
RDF [3]	44.08% (±0.45)	49.06% (±0.52)
SLR [3]	42.81% (±0.89)	48.46% (±1.58)
GP-HIK	**51.18**% (±0.08)	**54.01**% (±0.21)

Generalized HIK The histogram intersection kernel already offers non-linear classification models. However, it can be generalized by applying a polynomial transformation to each of the feature values, *i.e.*, instead of x_d we can use x_d^η with a parameter $\eta > 0$ [1, 8]. This does not violate the computed permutations in Eq. (6). In our experiments we optimize the hyperparameter η by cross-validation on a distinct validation split.

Handling unbalanced data Model regularization with GP is achieved by assuming noise on the training data (Eq. (1)). To handle unbalanced datasets, we use class-specific noise levels $\sigma^2_{\text{neg}} = 2\sigma^2\left(\frac{n_{\text{neg}}}{n}\right)$ and $\sigma^2_{\text{pos}} = 2\sigma^2\left(\frac{n_{\text{pos}}}{n}\right)$. For the multi-class case this strategy is performed for every binary classifier. Since the underlying data is the same for every binary classifier, the undisturbed kernel matrix $\boldsymbol{K}$ is shared among every class and the computations still remain efficient.

6. Experiments

For evaluation, we follow [3, 12, 13] and use the eTRIMS and LabelMeFacade dataset. The splits for eTRIMS are based on the random splits of [12, 13] and the split of the LabelMeFacade dataset is the same as in [3]. All other settings are identical with [3] but the classifier used, which in our case is non-linear and kernel-based. We compare our approach (GP-HIK) to random decision forests (RDF) [3], sparse logistic regres-

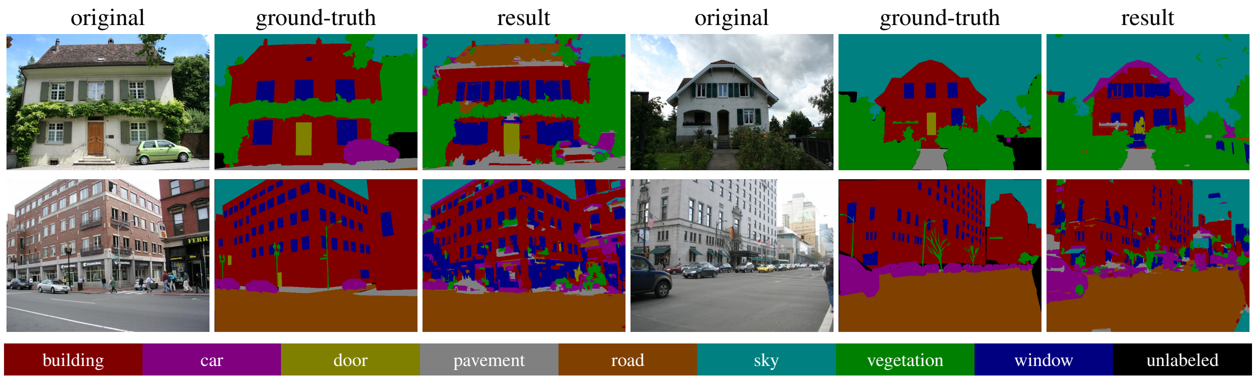

Figure 2. Example images from eTRIMS (first row) and LabelMeFacade database (second row).

sion (SLR) [3, 2], RDF with conditional random field (CRF) [13] and its hierarchical extension (HCRF) [12]. As measures of quality we use averaged class-wise recognition rates (ARR) as well as pixel-wise overall accuracies (ORR). For a detailed evaluation of necessary runtimes of the GP-HIK we refer the reader to [8]. Note that applying GP in its plain formulation to the task of semantic segmentation with thousands of training examples is not possible due to memory limitations.

The results of our method compared to those of state-of-the-art are shown in Table 1. With respect to overall accuracies our approach leads to results comparable to those of state-of-the-art on the eTRIMS dataset. Considering the average accuracy, our approach clearly outperforms previous methods. This result is especially appealing since it shows that we are able to reliably recognize multiple classes known during training. In contrast to that, previous approaches suffer from the fact to favor the classes most prominent in the training data.

For the highly challenging LabelMeFacade dataset the results of our method are significantly better than the state-of-the-art. We achieve performance gains of 7% and 5% respectively. Therefore, it is obvious that the GP-HIK can handle with the intense label noise in the LabelMeFacade dataset in a suitable way. Some sample results of our method are shown in Figure 2.

7. Conclusions and future work

In this paper, we presented how to efficiently apply Bayesian classification using Gaussian processes to the challenging task of semantic segmentation. Our approach is built on strategies for exploiting fast computations of the histogram intersection kernel. In experiments, we presented significant performance gains compared to state-of-the-art approaches. For future work, we plan to extend our ideas for fast computations of the predicted variance, which could be beneficial for post-processing steps in semantic segmentation tasks.

References

[1] S. Boughorbel, J.-P. Tarel, and N. Boujemaa. Generalized histogram intersection kernel for image recognition. In *ICIP*, pages III–161–4, 2005.

[2] G. Csurka and F. Perronnin. An efficient approach to semantic segmentation. *IJCV*, 95(2):198–212, 2011.

[3] B. Fröhlich, E. Rodner, and J. Denzler. A fast approach for pixelwise labeling of facade images. In *ICPR*, pages 3029–3032, 2010.

[4] A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell. Gaussian processes for object categorization. *IJCV*, 88(2):169–188, 2010.

[5] V. S. Lempitsky, A. Vedaldi, and A. Zisserman. Pylon model for semantic segmentation. In *NIPS*, pages 1485–1493, 2011.

[6] S. Maji, A. Berg, and J. Malik. Classification using intersection kernel support vector machines is efficient. In *CVPR*, pages 1–8, 2008.

[7] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. The MIT Press, 2006.

[8] E. Rodner, A. Freytag, P. Bodesheim, and J. Denzler. Large-scale gaussian process classification with flexible adaptive histogram kernels. In *ECCV*, 2012.

[9] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *ECCV*, pages 1–15, 2006.

[10] K. E. A. van de Sande, T. Gevers, and C. G. Snoek. Evaluating color descriptors for object and scene recognition. *TPAMI*, 32(9):1582–1596, 2010.

[11] J. Wu. A fast dual method for hik svm learning. In *ECCV*, pages 552–565, 2010.

[12] M. Y. Yang and W. Förstner. A hierarchical conditional random field model for labeling and classifying images of man-made scenes. In *ICCV Workshops*, pages 196–203, 2011.

[13] M. Y. Yang and W. Förstner. Regionwise classification of building facade images. In *Photogrammetric Image Analysis*, pages 209–220. Springer, 2011.