The NBNN kernel

T. Tuytelaars
K.U.Leuven, ESAT - PSI, IBBT
Leuven, Belgium
Tinne.Tuytelaars@esat.kuleuven.be

M. Fritz
MPI Informatics
Saarbrücken, Germany
mfritz@mpi-inf.mpg.de

K. Saenko, T. Darrell
UC Berkeley EECS & ICSI
Berkeley, US
saenko@eecs.berkeley.edu
trevor@eecs.berkeley.edu

Abstract

Naive Bayes Nearest Neighbor (NBNN) has recently been proposed as a powerful, non-parametric approach for object classification, that manages to achieve remarkably good results thanks to the avoidance of a vector quantization step and the use of image-to-class comparisons, yielding good generalization. In this paper, we introduce a kernelized version of NBNN. This way, we can learn the classifier in a discriminative setting. Moreover, it then becomes straightforward to combine it with other kernels. In particular, we show that our NBNN kernel is complementary to standard bag-of-features based kernels, focussing on local generalization as opposed to global image composition. By combining them, we achieve state-of-the-art results on Caltech101 and 15 Scenes datasets. As a side contribution, we also investigate how to speed up the NBNN computations.

1. Introduction

Recently, Boiman *et al.* [3] proposed a novel, non-parametric method for object classification, the *Naive Bayes Nearest Neighbor* classifier, or NBNN for short. NBNN is remarkably simple: given an image, one first computes a set of local features. Then, one searches for the class that minimizes the sum over all features of distances to the respective nearest neighbours belonging to that class. In spite of its simplicity and the complete absence of a training phase, NBNN achieves surprisingly good results on standard benchmarking data sets such as Caltech101, competitive with the state-of-the-art. The authors of [3] attribute this good performance to i) the lack of a vector quantization step and ii) the use of an 'image-to-class' distance instead of comparing 'image-to-image'. The former avoids discretization errors which they show are especially outspoken for the more informative features found in less dense areas of feature space. The latter enables a good generalization beyond the provided labelled images. Indeed, when evaluating a test image, NBNN combines bits and pieces of information from different example images. This is especially valuable when only a limited number of labelled images are available.

Figure 1. Limos and cars are an example of two classes that may be hard to distinguish using NBNN, since they are both composed of very similar local features. Also the picture on the right would probably be recognized as a car with high confidence, since most of its local features resemble car features. Bag-of-features based classifiers, on the other hand, look at the overall feature distribution and would have no problem classifying these images. This shows the complementarity between both methods.

However, the NBNN framework also has its limitations. The needed computation time during testing is high, especially when sampling very densely which often seems necessary to obtain good results. Moreover, the method assumes similar densities in feature space for all classes, such that the same kernel bandwidth can be used for all of them. In practice, this assumption is often violated, resulting in a strong bias towards one or a few object classes. These two points have been addressed by [1] and [21] respectively, who both introduce a learning phase in order to do so.

Additionally, the independence assumption underlying NBNN can also be criticized. Since each feature is treated separately, information concerning the overall image composition is ignored. As a result, distinguishing e.g. between a limo and a normal car is likely to be difficult for NBNN. This is illustrated in Figure 1. For every local feature found on a limo (resp. car), very similar features can be found both on other limos as well as on other cars. Likewise, a set of tires may get a good score for either of these two classes, since most of its local features resemble car or limo features, even though obviously important object parts are missing. This is in sharp contrast to bag-of-features based

approaches, which directly encode the overall distribution of features in an image and as a result would not encounter the same difficulties. This illustrates the complementarity between the two methods.

To combine both methods in a single framework, we propose to *kernelize* the NBNN classifier. This way, it can be integrated in a multiple kernel learning framework (e.g. [6, 11, 26]). Various authors have studied the use of kernel learning for object categorization [6, 10, 12, 16]. However, these works mostly focussed on combining different features (e.g. grayscale and color features, different levels of invariance) or different spatial binning schemes. Here we propose to exploit another sort of complementarity.

Building a kernel that exploits image-to-class instead of image-to-image comparisons may seem contradictory at first, since a kernel by definition works on a pair of (image) representations. However, the two image representations (sets of local features) need not be compared directly. Instead, we compute their distances to the different classes and compare these, as illustrated in Figure 2.

The main contributions of this paper can be summarized as follows. We introduce a kernelized version of NBNN. The NBNN kernel incorporates the main ideas underlying NBNN – nearest neighbor search in feature space and image-to-class comparison. Yet it is also a Mercer kernel, so it can be used with a support vector machine to discriminatively train a classifier. Moreover, the NBNN kernel can then be combined with the standard bag-of-features kernels, building on the strengths of both approaches. With NBNN focussing on appearance details without discretization errors and good generalization beyond the provided labelled images, and bag-of-features encoding the overall feature distribution in an image, the combined scheme outperforms each of them individually and yields state-of-the-art results.

As a side contribution, we also investigate how to speed up NBNN as well as the NBNN kernel. To this end, we introduce an asymmetric scheme, where one samples more densely for the training images (to generate the database for the nearest neighbor search) while sampling less densely during testing. This reduces the computation time with only a limited effect on the accuracy.

The remainder of this paper is organized as follows. We first discuss related work. Section 2 explains our NBNN kernel. Section 3 starts with a complexity analysis and then proposes a way to speed up the algorithm. Section 4 describes our experimental results. Section 5 concludes the paper.

Related work A few papers have looked into variations of the original NBNN algorithm, mostly aiming at reducing the bias in case of unbalanced datasets, by adapting the (isotropic) kernel bandwidth [1, 23] or using metric learning [21]. These propose dedicated solutions for the particular NBNN setting. Our work, on the other hand, reformulates the core ideas of NBNN in the standard kernel framework.

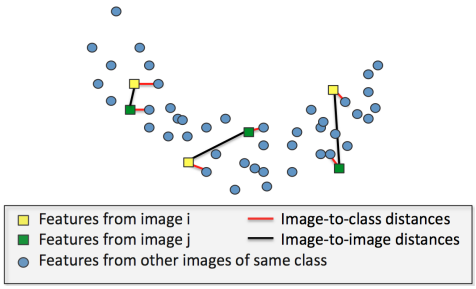

Figure 2. The image-to-class concept: even though the features of the two images are not very similar (close), their distances to the class distribution are similar, and that is what counts for NBNN.

Other researchers have investigated alternative ways to avoid the vector quantization step, using sparse coding [22] or locality-constrained linear coding [21]. However, neither of these integrates the 'image to class' ideas of NBNN.

Before the introduction of the bag-of-features paradigm [4] and its wide adoption in the computer vision community, other kernels for directly comparing sets of features have been proposed as well, with varying success. Wallraven *et al.* [19] proposed the 'matching kernel', which involved explicit matching of features between images. However, later it was found by Liu *et al.* [13] this is not a Mercer Kernel, although for many practical cases this hardly matters. Instead, they propose the sum match kernel, which is at the basis of our NBNN kernel. Bo and Sminchisescu [2] have proposed a method to make the computation of such kernels more efficient by projecting to a lower dimensional space.

Also worth mentioning is the Kullback Leibler kernel proposed by [17], which directly compares two feature distributions. Finally, based on its name, one might think the SVM-kNN framework proposed by Zhang *et al.* [24] is related as well. However, they are in fact solving a very different problem.

2. Kernelizing NBNN

2.1. NBNN

The Naive Bayes Nearest Neighbor algorithm [3] assumes all features $\{\mathbf{x}\}$ are independently sampled from a class-specific feature distribution $p(\mathbf{x}|c)$. Classification of a query image Q then boils down to a Maximum Likelihood classifier:

$$\hat{c} \;=\; argmax_c \;\; p(c|Q) = argmax_c \prod_{\mathbf{x}} p(\mathbf{x}|c) \quad (1)$$

Algorithm 1: NBNN

1. Compute a set of features $X = \{\mathbf{x}\}$.
2. $\forall \mathbf{x} \forall c$ Compute the NN of $\mathbf{x}$ in c, $NN^c(\mathbf{x})$, and its distance-to-class $d^c_{\mathbf{x}} = ||\mathbf{x} - NN^c(\mathbf{x})||^2$.
3. $\hat{c} = argmin_{c \in C} \sum_{\mathbf{x} \in X} d^c_{\mathbf{x}}$.

assuming a uniform prior $p(c)$. This is shown to be equivalent to minimizing an image-to-class Kulback Leibler distance $KL(p(\mathbf{x}|Q)||p(\mathbf{x}|c))$. The class specific feature distribution $p(\mathbf{x}|c)$ is approximated using Parzen density estimation. When only the Parzen kernel around the nearest neighbour is retained and the same kernel bandwidth is used for all classes, this results in a very simple algorithm: Given a set of local features $X = \{\mathbf{x}\}$, one searches for the class $\hat{c}$ which minimizes the sum $\sum_{\mathbf{x} \in X} ||\mathbf{x} - NN^c(\mathbf{x})||^2$, where NN^c is the nearest neighbor of $\mathbf{x}$ belonging to class c (see Algorithm 1).

2.2. The NBNN kernel

Here, we propose to transfer these ideas to a discriminative scheme. With the *NBNN kernel*, the core ideas underlying the NBNN algorithm are preserved and can be combined with the mature technology of kernel-based learning.

Here, we build on the *normalized sum match kernel* proposed by Lyu *et al.* to compare sets of features $X = \{\mathbf{x}\}$ and $Y = \{\mathbf{y}\}$(see [8, 13, 2]):

$$K(X,Y) = \Phi(X)^T \Phi(Y) = \frac{1}{|X||Y|} \sum_{\mathbf{x} \in X} \sum_{\mathbf{y} \in Y} k(\mathbf{x}, \mathbf{y}) \quad (2)$$

Since we work in a multiclass setting, we sum together various class-specific kernels:

$$\begin{aligned} K(X,Y) &= \sum_{c \in C} K^c(X,Y) & (3) \\ &= \frac{1}{|X||Y|} \sum_{c \in C} \sum_{\mathbf{x} \in X} \sum_{\mathbf{y} \in Y} k^c(\mathbf{x}, \mathbf{y}) & (4) \end{aligned}$$

with $C = \{c\}$ the set of all classes.

It can be shown [13] that (unlike the match kernel proposed by [19]) $K(X,Y)$ is a Mercer kernel if the *local kernel* operating at the feature-level, $k^c(\mathbf{x}, \mathbf{y})$, is a Mercer kernel. Choosing $k^c(\mathbf{x}, \mathbf{y}) = \delta(\mathbf{x}, \mathbf{y})$, with $\delta(\mathbf{x}, \mathbf{y}) = 1$ if $\mathbf{x}$ and $\mathbf{y}$ have been assigned to the same visual word and 0 otherwise, this degenerates into a linear kernel computed on bag-of-features [2]. Instead, for our NBNN kernel, we choose the local kernel as a combination of functions of distances to the different classes:

$$\begin{aligned} k^c(\mathbf{x}, \mathbf{y}) &= \phi^c(\mathbf{x})^T \phi^c(\mathbf{y}) & (5) \\ &= f^c(d^1_{\mathbf{x}}, \ldots, d^{|C|}_{\mathbf{x}})^T f^c(d^1_{\mathbf{y}}, \ldots, d^{|C|}_{\mathbf{y}}) & (6) \end{aligned}$$

Algorithm 2: the NBNN kernel

1. Compute a set of features $X = \{\mathbf{x}\}$.
2. $\forall \mathbf{x} \forall c$ Compute the NN of $\mathbf{x}$ in c: $NN^c(\mathbf{x})$, and its distance-to-class $d^c_{\mathbf{x}} = ||\mathbf{x} - NN^c(\mathbf{x})||^2$.
3. $\forall c \;\; \Phi^c(X) = \sum_{\mathbf{x} \in X} f(d^1_{\mathbf{x}}, \ldots, d^{|C|}_{\mathbf{x}})$.
4. $\Phi(X) = [\Phi^1(X) \ldots \Phi^{|C|}(X)]^T$.
5. Repeat steps 1-4 for a second set of features $Y = \{\mathbf{y}\}$.
6. $K(X,Y) = \Phi(X)^T \Phi(Y)$.

with $d^c_{\mathbf{x}}$ again the distance from $\mathbf{x}$ to its nearest neighbour belonging to class c. In other words, we do not compare features $\mathbf{x}$ and $\mathbf{y}$ directly, but instead compare the distances to their respective nearest neigbours extracted from reference images of various object classes. $\mathbf{x}$ and $\mathbf{y}$ may be far apart in feature space; if they have similar distances to the various classes (i.e., both close to or both far away from the same classes), they are considered similar. This is in line with the ideas underlying NBNN: features are not vector quantized, and information from multiple reference images is combined, using a distance-to-class instead of a distance-to-image.

In practice, we have experimented with two different functions $f^c(d^1_{\mathbf{x}}, d^2_{\mathbf{x}}, \ldots, d^{|C|}_{\mathbf{x}})$, namely

$$\begin{aligned} f^c_1(d^1_{\mathbf{x}}, \ldots, d^{|C|}_{\mathbf{x}}) &= d^c_{\mathbf{x}} & (7) \\ f^c_2(d^1_{\mathbf{x}}, \ldots, d^{|C|}_{\mathbf{x}}) &= d^c_{\mathbf{x}} - d^{\bar{c}}_{\mathbf{x}} & (8) \end{aligned}$$

where $d^{\bar{c}}_{\mathbf{x}}$ represents the closest distance to all classes except c. When using f^c_1, $\Phi^c(X)$ corresponds to the average of the distances to the nearest neighbours for all features extracted from the query image, which is very similar to the sum of distances used in the NBNN algorithm. With f^c_2, we subtract the distance to the nearest neighbor not belonging to class c. This corresponds to using the likelihood ratio instead of the likelihood in equation 1, and in our experiments gave superior results. This is somewhat similar to the Mutual Information measure used by [23]. Of course, other functions can be tried as well, e.g. adding a kernel parameter p as exponential weight to introduce a bias as in [13].

Algorithm 2 summarizes the computation of the NBNN kernel. Once we have computed the kernel, we solve the object categorization problem using a support vector machine, but other kernel methods could be used as well.

In fact, when using f^c_1, one can also explain the resulting method in another, simpler yet more ad hoc manner, as follows: we compute the sums of distances to nearest neighbours of each class, exactly as in the standard NBNN procedure. However, instead of then selecting the class for which this sum is minimal, we concatenate all the sums into a large vector. This constitutes our new image representation based on which a support vector machine then learns to distinguish the different object classes.

	features/image	spatial coord.	accuracy
NBNN	2000	no	44.1 ± 1.5
NBNN	2000	yes	$\mathbf{62.7 \pm 0.5}$
NBNN	500	yes	56.5 ± 1.7
NBNN, asymmetric	2000/500	yes	$\mathbf{60.0 \pm 0.9}$

Table 1. Caltech-101 classification accuracy with our implementation of the NBNN algorithm of [3] using 15 training images. 'Slow' methods on top, 'faster' methods below. (mean and standard deviation)

3. Optimizing NBNN

Complexity analysis In spite of its simplicity, the NBNN algorithm is not particularly fast at test time. This is due to the fact that, for each feature extracted in a test image, one needs to search for the nearest neighbor among all features extracted from the labeled images and this for each class. Using a naive implementation, this is quadratic in the number of features, i.e. $O(N^2)$ with N the average number of features extracted from an image. Using approximate nearest neighbors, this is reduced to $O(Nlog(N))$. Boiman *et al.* [3] report a computation time of 1.6 seconds per image per class. This means that, for instance, for the Caltech101 dataset with 101 classes it takes 163 seconds per image or almost three days to process one split of 15 training images. The same nearest neighbor computations are also needed for the kernelized NBNN, so the time complexity at test time is roughly the same. However, now we also have a training phase for which we need to compute the distances for all the training images.

[21] have suggested to sample less densely, and try to counter the reduction in accuracy by learning a good metric in feature space. Here, we suggest an alternative method to reduce the computational complexity, namely the use of an asymmetric scheme.

An asymmetric approach When comparing two images, a good repeatability is ensured as long as we sample very densely in at least one of the images. If the training images are sampled very densely, the test images can be sampled more coarsely. Indeed, computing the average distance to the nearest neighbor of a particular class probably does not need tens of thousands of distances, but can be approximated from just a few hundreds of distances. Since the computation time is linear in the number of features in the test image, reducing the number of features by an order of magnitude will also speed up the classification procedure by an order of magnitude.

4. Experimental results

4.1. Caltech-101

Implementation details In our experiments with the Caltech-101 data set, we follow the experimental setup of Vedaldi *et al.* [18]. In particular, we use the same train-test splits as they do, rescale the large images the same way they do, and add jittered images the same way they do. Also, we use their 'phowGray' kernels with three different spatial binnings as our 'bag-of-features kernels'. We experimented with the multiple kernel learning of [16] included in their framework, but found that the mclp-boost method proposed by Gehler and Nowozin [6] yields superior results.

For the NBNN-kernel, we use the code of [14] to compute approximate nearest neighbors, and liblinear [5] for learning the support vector machine. Since NBNN works best with balanced data and cannot cope with 40.000 features per image, we do not use the same dense sampling (every other pixel) used for the bag-of-features kernels, but only a subset thereof, making sure the number of features per image is more or less constant.

During training of the NBNN kernel, the distances-to-class must be computed for the training images. Here, one must be careful to exclude all features originating from the image itself (as well as jittered versions thereof). We start with experiments using 15 training images per class.

Standard NBNN First, we evaluate the effect of different settings for the standard NBNN algorithm as proposed by [3] – see Table 1. In the original NBNN publication, the image coordinates were added as additional elements in the local feature descriptors. This steers the NN search to features in nearby locations. Indirectly, this enforces some kind of global distribution, as it makes extreme multi-to-one matchings as in the right part of Figure 1 impossible. The impact of this 'trick' of adding the spatial coordinates on the results cannot be underestimated: not including the spatial coordinates causes a drop in performance of almost 20%. We set the weight α to balance the spatial coordinates with the rest of the feature vector empirically and keep it fixed for all experiments.

Also, good results can only be obtained when sampling sufficiently densely (as also pointed out by [21]). With 2000 features per image, we get an accuracy of almost 63%, which is still below the 65% reported by [3] [1]. However, as discussed in section 3, sampling more densely makes

[1] Note that they did not include the background class as we did - if we leave out that class, our score goes up a bit to $63.3 \pm 0.4\%$.

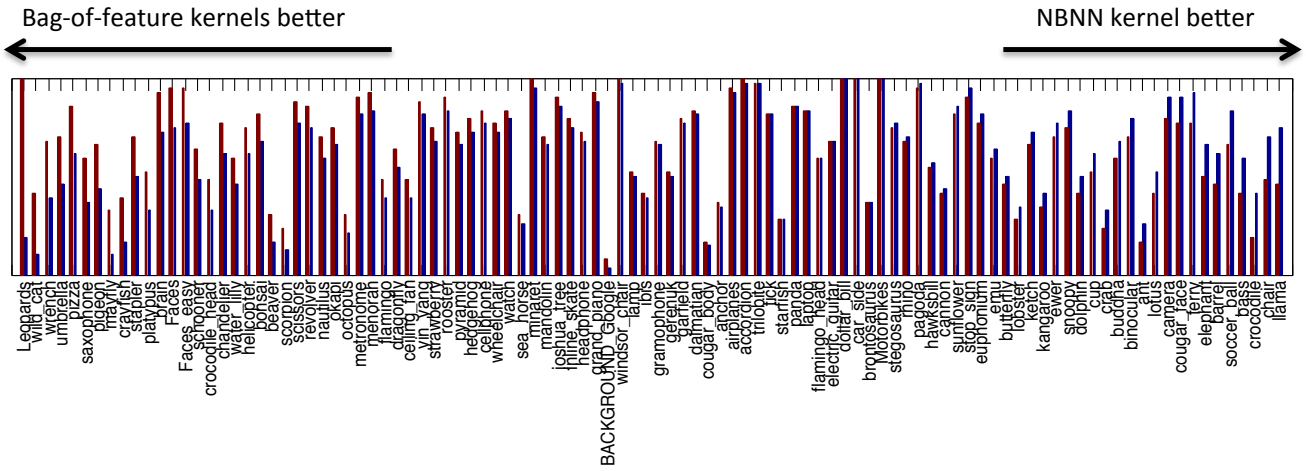

Figure 3. Classification accuracy per class (i.e. the diagonal of the matrices shown in figure 4) for NBNN (blue) and the bag-of-features based approach (red).

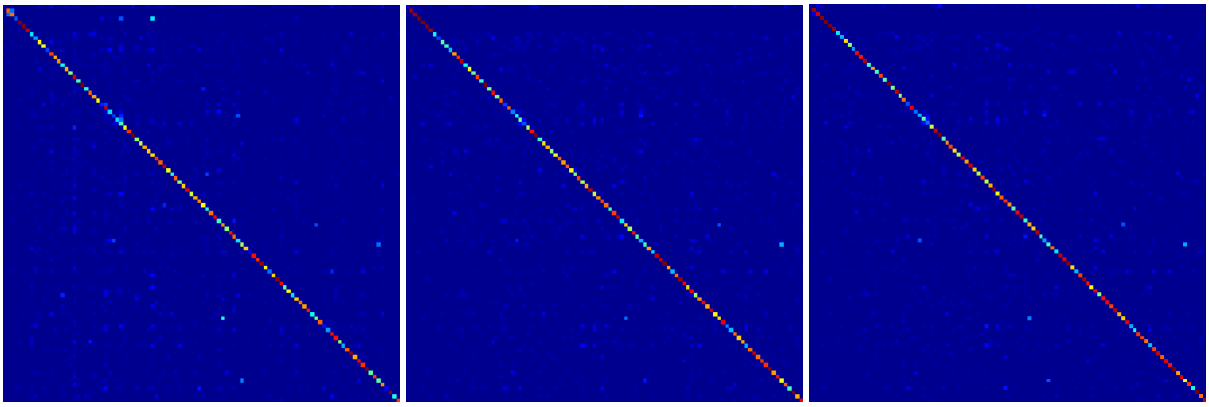

Figure 4. Confusion tables for NBNN (left), for the bag-of-features based kernels used in our experiments (middle), and for the combination of both (right).

the procedure really slow. With only 500 features per image the method becomes much faster, but the performance drops significantly as well. Using the proposed asymmetric scheme brings us back to a 60% accuracy while the computation time is still about 4 times lower than in the original setting.

Complementarity with bag-of-features Next, we compare the classification results of the standard NBNN algorithm with an SVM classifier trained on bag-of-features. For the latter, we use the 'phowGray' kernels L0, L1 and L2 of [18], corresponding to different spatial binnings. This is the best 'single feature' kernel combination reported by [18]. Figure 4 shows the respective confusion matrices and Figure 3 plots the classification accuracies for each class, sorted by the difference in performance between both methods.

Both methods show some complementarity. The largest difference can be found for the class 'Leopards'. This is due to the fact that the bag-of-features based method (using a varying number of features per image) can exploit the small size of images of this category, while NBNN ignores this information. This also explains why 'wild cats' score lower, since they get confused with leopards. NBNN scores better on those classes for which there aren't 'typical image compositions' (e.g. 'soccer ball' or 'chair') and less good for classes that lack distinctive features (e.g. 'wrench' or 'umbrella'). More important than the absolute classification accuracies are the confusion tables of Figure 4. Here, we observe that mistakes made by NBNN are often due to related classes such as 'Faces' vs. 'Faces-easy', 'crocodile' vs. 'crocodile-head', 'cougar-body' vs. 'cougar-head', etc.

	features	kernel	accuracy
NBNN kernel + SVM	2000	f_1	57.7 ± 1.5
NBNN kernel + SVM	2000	f_2	$\mathbf{61.3 \pm 0.2}$
NBNN kernel, asymmetric + SVM	2000/500	f_1	56.6 ± 0.7
NBNN kernel, asymmetric + SVM	2000/500	f_2	$\mathbf{60.4 \pm 1.1}$

Table 2. Caltech101 classification accuracy with the NBNN kernel, using 15 training images. (mean and standard deviation)

	accuracy
NBNN kernel	61.3 ± 0.2
phow kernels	66.9 ± 1.3
NBNN & phow kernels	$\mathbf{69.2 \pm 0.9}$

Table 3. Caltech101 classification accuracy with 15 training images per class. Comparison of different kernel combinations, all starting from the same features (dense sampling + SIFT). (mean and standard deviation)

	accuracy
NBNN	65.5 ± 1.0
NBNN kernel	69.6 ± 0.9
NBNN & phow kernels	$\mathbf{75.2 \pm 1.2}$

Table 4. Caltech101 classification accuracy with 30 training images per class. Comparison of different kernel combinations, all starting from the same features (dense sampling + SIFT). (mean and standard deviation)

Kernelized NBNN Next, we evaluate the performance of the kernelized NBNN. The results are summarized in Table 2. Contrary to our expectations and in spite of the discriminative learning, the kernelized version of NBNN did not manage to outperform the Naive Bayes scheme based on maximum likelihood. This may be due to overfitting to such small amounts of training data. The kernel based on the log likelihood ratio (i.e. using f_2) gives much better results than the one using the log likelihood directly (using f_1). Hence we will be using this kernel from now on.

NBNN-kernel in combination with other kernels Finally, we use the mclp-boost code provided by Gehler and Nowozin [6] to combine the NBNN kernel with the bag-of-features kernels. As shown in Table 4, the boosting manages to exploit the complementarity of the different kernels, and outperforms each of them individually. The final confusion table is shown in Figure 4, right.

30 training images We repeat the same experiments with 30 training images. To keep the computational load under control, we only use the *asymmetric* scheme. Under this setting, our initial results (65.5%) are significantly below the 70.4% reported by Boiman *et al.* [3]. However, given the higher number of training images, the problem with overfitting is reduced, so when switching to the kernelized version, the accuracy increases to 69.6%. Combining the NBNN kernel with the bag-of-features kernels, we again outperform the individual kernels, leading to a final classification accuracy of 75.2%.

Comparison to state-of-the-art In Table 5, we compare these results with the state of the art, focussing on methods that use the same image representation (dense sampling + SIFT). In this category, we obtain the best results reported so far (to the best of our knowledge).

	15 images	30 images
Results of methods using dense sampling + SIFT		
Gehler *et al.* [6]	54.5 ± 0.9	63.8 ± 1.0
Griffin *et al.* [7]	59.4	67.6 ± 1.4
NBNN [3]*	65.0 ± 1.1	70.4
LLE [20]	65.43	73.4
Vedaldi *et al.* [18]	66.3 ± 1.1	?
ScSPM [22]	67.0 ± 0.5	73.2 ± 0.5
ours	$\mathbf{69.2 \pm 0.9}$	$\mathbf{75.2 \pm 1.2}$
Results using a single feature (other than SIFT)		
Gehler *et al.* [6] (1 descr.)	61.0 ± 0.2	69.4 ± 0.4
Pinto *et al.* [15]	61.4	67.4
NIMBLE [9]*	$\mathbf{70.8 \pm 0.7}$	$\mathbf{78.5 \pm 0.4}$

Table 5. Comparison with state-of-the-art, focussing on single-descriptor methods. Methods with an asterix did not include the background class in their experiments, so are probably slightly overestimated.

For reference, we also include the best performing methods using a single feature (other than SIFT). Here, our result is only outperformed by Kanan *et al.* [9] [2]. Moreover, note that their good result is due to the special features they introduce. This seems complementary to our contribution, so it seems likely that applying our scheme on top of their features could increase the score even further.

We have left out results combining different features. Usually the results get better the more features or kernels are taken into account, with currently best results for Gehler and Nowozin's combination of 48 different kernels, achieving 74.6% with 15 training images.

[2] But note their result again did not consider the background class.

	accuracy
NBNN	75 ± 3
NBNN kernel	79 ± 2
bag-of-features kernels	75 ± 3
NBNN & bag-of-features kernels	$\mathbf{85 \pm 4}$

Table 6. Classification accuracy for 15 Scenes dataset. Comparison of different kernel combinations, all starting from the same features (dense sampling + SIFT). (mean and standard deviation)

4.2. 15 scenes

We also tested our method in the context of scene categorization, using the 15 scenes dataset. Here, we use the standard scheme with different splits of 100 images for training and 100 images for testing. Table 6 summarizes the main results. Again, we start from densely sampled image patches described using SIFT. We extract around 1500 features per image. Since the number of training images is larger, we do not add jittered images. In this case, the discriminative learning brought by the kernelized version of NBNN does pay off, with a 4% increase in performance relative to the original NBNN.

Next, we combine the NBNN kernel with spatially binned bag-of-features kernels (again L0, L1 and L2, i.e. flat, 2×2 and 4×4 subdivisions), using a visual vocabulary of size 500. These results can probably be improved further when combined with denser features or more advanced kernels, e.g. based on sparse coding or soft quantization. Nevertheless, we already obtain results competitive with the current state-of-the-art (best score to date is also around 85% [25]).

5. Conclusion

We have shown that the NBNN classifier is complementary to the widely used bag-of-features based approaches, focussing on individual details as opposed to overall image composition respectively. To exploit this complementarity, we have introduced the NBNN kernel. The NBNN kernel keeps the basic ideas underlying NBNN intact (no vector quantization and image-to-class comparisons), yet allows to cast them in a discriminative setting. In some cases (when the number of training images is sufficiently large to avoid overfitting), this already outperforms standard NBNN. However, the main benefit comes from combining the new kernel with bag-of-features kernels in a multikernel framework, exploiting their complementarity. This outperforms each of them individually and systematically yields state-of-the-art results (best results to date starting from dense sampling + SIFT).

As future work, we plan to explore alternative (non-linear) kernels as well as feature selection. Indeed, a shortcoming of the current approach is that it does not scale well in the number of classes. It would be interesting if we could impose sparseness, such that one does not always have to compute the distance to all classes. And of course, we are curious to see what performance we can obtain on benchmark data sets if we combine our new NBNN-kernels with other kernels using a variety of different features.

Acknowledgments

The research leading to these results has received funding under the European Community's Seventh Framework Programme(FP7/2007-2013) from the European Research Council / ERC grant agreement nr. 240530 Cognimund and from the European Commission / Integrating Project AXES, and from a Feodor Lynen Fellowship granted by the Alexander von Humboldt Foundation.

References

[1] R. Behmo, P. Marcombes, A. Dalalyan, and V. Prinet. Towards optimal naive bayes nearest neighbors. In *European Conference on Computer Vision*, 2010. 1, 2

[2] L. Bo and C. Sminchisescu. Efficient match kernel between sets of features for visual recognition. In *Advances in Neural Information Processing Systems*, 2009. 2, 3

[3] O. Boiman, E. Schechtman, and M. Irani. In defense of nearest neighbor based image classification. In *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2008. 1, 2, 4, 6

[4] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Workshop on Statistical Learning in Computer Vision*, 2004. 2

[5] R. Fan, K. Chang, C. J. Hsieh, X. R. Wang, C. J. Lin, and S. Sonnenburg. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2009. 4

[6] P. Gehler and S. Nowozin. On feature combination for multiclass object detection. In *International Conference on Computer Vision*, 2009. 2, 4, 6

[7] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset, 2007. 6

[8] D. Haussler. Convolution kernels on discrete structures, 1999. Technical report. 3

[9] C. Kanan and G. Cottrell. Robust classification of objects, faces, and flowers using natural image statistics. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010. 6

[10] A. Kumar and C. Schminchisescu. Support kernel machines for object recognition. In *International Conference on Computer Vision*, 2007. 2

[11] G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004. 2

[12] Y. Lin, T. L. Liu, and C. S. Fuh. Local ensemble kernel learning for object category recognition. In *International Conference on Computer Vision*, 2007. 2

[13] S. Lyu. Mercer kernels for object recognition with local features. In *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2005. 2, 3

[14] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007. 4

[15] N. Pinto, D. Cox, and J. DiCarlo. Why is real-world visual object recognition hard? *PLoS Computational Biology*, 4, 2008. 6

[16] M. Varma and D. Ray. Learning the discriminative power-invariance trade-off. In *International Conference on Computer Vision*, 2007. 2, 4

[17] N. Vasconcelos, P. Ho, and P. Moreno. The kullback-leibler kernel as a framework for discriminant and localized representations for visual recognition. In *European Conference on Computer Vision*, 2004. 2

[18] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *International Conference on Computer Vision*, 2009. 4, 5, 6

[19] C. Wallraven, B. Caputo, and A. Graf. Recognition with local features: the kernel recipe. In *International Conference on Computer Vision*, 2003. 2, 3

[20] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2010. 6

[21] Z. Wang, Y. Hu, and L.-T. Chia. Image-to-class distance metric learning for image classification. In *European Conference on Computer Vision*, 2010. 1, 2, 4

[22] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, pages 1794–1801, 2009. 2, 6

[23] J. Yuan, Z. Liu, and Y. Wu. Discriminative subvolume search for efficient action detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009. 2, 3

[24] H. Zhang, A. Berg, M. Maire, and J. Malik. Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, 2006. 2

[25] X. Zhou, N. Cui, Z. Li, F. Liang, and T. S. Huang. Hierarchical gaussianization for image classification. In *International Conference on Computer Vision*, 2009. 7

[26] A. Zien and C. S. Ong. Multiclass multiple kernel learning. In *International Conference on Machine Learning*, pages 1191–1198, 2007. 2