The Pooled NBNN Kernel: Beyond Image-to-Class and Image-to-Image

Konstantinos Rematas[1], Mario Fritz[2], and Tinne Tuytelaars[1]

[1] KU Leuven, ESAT-IBBT
[2] Max Planck Institute for Informatics

Abstract. While most image classification methods to date are based on image-to-image comparisons, Boiman *et al.* have shown that better generalization can be obtained by performing image-to-class comparisons. Here, we show that these are just two special cases of a more general formulation, where the feature space is partitioned into subsets of different granularity. This way, a series of representations can be derived that trade-off generalization against specificity.
Thereby we show a connection between NBNN classification and different pooling strategies, where, in contrast to traditional pooling schemes that perform spatial pooling of the features, pooling is performed in feature space. Moreover, rather than picking a single partitioning, we propose to combine them in a multi kernel framework. We refer to our method as the *Pooled NBNN kernel*. This new scheme leads to significant improvement over the standard image-to-image and image-to-class baselines, with only a small increase in computational cost.

1 Introduction

When Boiman *et al.* [1] presented their naive Bayes nearest neighbors (NBNN) classifier for multi-class image classification, it was surprising (and confronting) to see how well such a simple algorithm performed as compared to the machine-learning heavy methods used normally. As explained in [1], the success of this method can be attributed to two deviations from the standard approaches: i) the avoidance of a vector quantization step (*i.e.* no visual vocabularies), and ii) the use of image-to-class rather than image-to-image comparisons. While the former is definitely interesting, our focus in this paper goes to the latter aspect.

The NBNN approach combines a naive Bayes assumption together with an approximate Parzen estimate, that results in a very intuitive classification algorithm. The score of each test image with respect to a particular class is determined by how well the set of all features in the class can "reconstruct" the features in the test image. This is measured by summing over the distances to the nearest neighbors. By searching for nearest neighbors over all training images belonging to a class, a new test image is effectively classified by combining bits and pieces of information extracted from different training images. This obviously allows better generalization beyond the original set of training images.

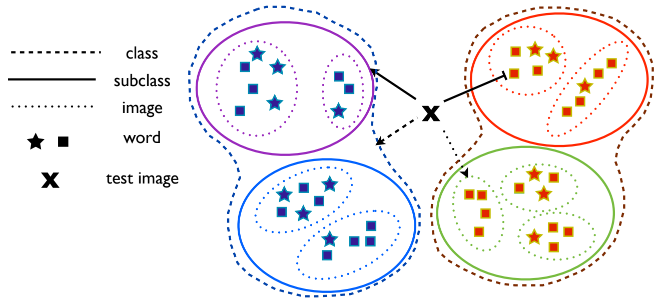

Fig. 1. The concept of our framework: For each feature from a given test image we can compute the distances to classes, distances to subclasses, distances to class specific feature clusters, or distances to individual images (= different ways of pooling the reference features in feature space). Summing these distances over all features in the test image (= traditional spatial pooling), we can build various alternative classification schemes in between image-to-image and image-to-class, or combine them for even better performance.

Here we propose to cluster the training images for each class in a number of smaller partitions and/or cluster the features in the reference images of a particular class into class-specific visual words (but without applying any vector quantization). Then a similar scheme as in the work of Boiman *et al.* can be applied (Figure 1), now sharing features within a cluster. This approach makes sense since an object class is defined mostly on semantic grounds, which do not necessarily match with coherence in visual appearance. For instance, different viewpoints typically correspond to different aspects, with completely different feature distributions. As a result, a class can contain different 'modes' or subclasses, where the visual appearance within a subclass is very similar yet can differ a lot between subclasses.

Recently, Tuytelaars *et al.* [2] have proposed a kernelized version of the NBNN framework of [1]. This new scheme keeps the image-to-class comparisons, while at the same time fitting it in the kernel-based line of work popular for image classification. The main advantage is that it allows easy integration with other kernels. Here, we build on the work of [2] and combine different kernels focusing on classes, sub classes and other partitionings at different levels of granularity.

In a sense, this is also reminiscent of the work on the pyramidal match kernel [3] or spatial pyramid [4]. Where they combine different levels of discretization in feature space and in the spatial domain respectively, our work investigates the combination of different levels of discretization along a different dimension, varying class granularity, from specific images over subclasses up to classes. In all these cases, instead of trying to select the one optimal operating point, various modes of operation can be combined into a single scheme for best performance.

NBNN as pooling along a different dimension. Most standard algorithms for image classification based on local features consist of the same three consecutive steps: i) feature coding, ii) feature pooling, and iii) classification. Typical examples of the first step (feature coding) include hard or soft assignment to visual words (vector quantization), sparse coding, or Locality-constrained Linear Coding. The second step is usually performed by spatial pooling, *e.g.* resulting in a spatial pyramid, and both sum pooling as well as max pooling have been proposed in this context. For the last step, a support vector machine remains the most popular choice. At first, NBNN or the NBNN kernel do not seem to fit in this scheme. Yet, they also perform various forms of pooling: the nearest neighbor selection can be seen as a max pooling step, while the Naive Bayes classification can be considered as sum pooling. This pooling is a bit extra-ordinary, in that it also pools information along a different dimension (over features extracted from the reference images, rather than features from the test image). We argue that, in other schemes, this type of pooling is also present, yet subsumed in the use of visual words, and therefore not made explicit.

In general, any image classification problem based on local features can be defined as follows: given the features from the training images, predict the correct label for a test image, based on a comparison between training and testing features. Their similarities can be represented by a very large matrix C of size $M \times N$ that contains the pairwise distances from M testing features to the N training features (with $N >> M$). Directly using the matrix C as input for classification doesn't work too well, since i) it's too large (making it impractical in terms of memory usage and sensitive to overfitting), and ii) it depends on the number of features extracted from the test image. So instead, the matrix has to be sparsified (typically achieved by coding) and to be reduced in dimensionality (typically done by pooling). Several methods perform these steps implicitly. For instance, vector quantization pools information from different features observed during training, and hence reduces N to W (the size of the vocabulary). Likewise, the bag-of-words model reduces the number of rows M in C to 1 by pooling information over the entire test image.

In this light, NBNN can be interpreted as follows: a) the NN part reduces the columns of C from N (the number of training features) to the number of classes, by performing max pooling over all the training features belonging to a class, and b) the NB part reduces the rows of C from M (the number of test features) to 1, by sum pooling over all test features. In other words, the Naive Bayes scheme pools information both horizontally as well as vertically. The extension proposed in this paper further considers alternative ways of performing the 'horizontal' max pooling, by partitioning the training data in different ways (into classes, class-specific subclasses and visual words).

Contributions: Our main contribution is an exploration of the spectrum between image-to-image and image-to-class distances, that we form in a data-driven manner by building on the recent success of NBNN techniques. We hereby propose a novel series of intermediate object class representations. Combining them, we obtain strong performance on three computer vision benchmark datasets. Moreover, we show a connection between NBNN classification and different pooling strategies, where pooling is performed in feature space.

The remainder of this paper is organized as follows. First, we describe related work (section 2). Then we provide some background information on the NBNN framework of [1] and the kernelized version thereof proposed by [2] (section 3). In section 4, we describe how we extend this framework to explore the full spectrum between image-to-image and image-to-class. Section 5 describes our experimental results and section 6 concludes our paper.

2 Related Work

The application of kernel methods to the problem of visual classification has been very successful. From early bag-of-words representations [5] over pyramid schemes [3, 4] to recent kernel combinations [6] these methods have greatly advanced the field. The key principle these methods operate on is a similarity measure between data points that is induced by a kernel function. While this unifying view might be perceived as one of the greatest strengths of this approach it comes also with a weakness. Similarities are measured globally between the examples. One direct consequence is that the model cannot easily incorporate any assumptions on the underlying data distribution.

Naive-Bayes-Nearest-Neighbor models (NBNN) [1] on the other hand make one of the most radical assumptions – which is independence between each feature dimension. This is right at the other end of the spectrum when compared with the kernel learning approach as they allow for an explanation of an observation at test time by a recombination of all features in the training using an image-to-class distance. While this approach has a strong generalization performance, it may loose consistency of the prediction as it takes feature sharing to the extreme. In order to balance strength and weaknesses of both approaches recent work aimed at combining both schools of thought [2]. Our work aims at exploring the spectrum in-between these two extremes.

There is a rich literature on employing hierarchies and/or taxonomies to aid classification. Various approaches are based on a semantic ontology as defined by models like wordnet [7, 8]. While these ideas appeal on a computational as well as conceptual level, the results often lag behind the expected performance gains.

Apart from hierarchies defined by human domain knowledge, hierarchies and tree structures have a long standing tradition in the machine learning community (e.g. [9, 10]). Here, the structure that supports classification is typically build in a data-driven way – and in fact it is unclear in which cases human or data-driven class groupings will give better support for classification. Recent methods followed up on this issue and suggest a relaxed hierarchy that accounts for semantic and classification cost [11]. Our approach follows this line of thought by forming subclasses and feature clusters that still obey the basic category labeling.

Recent work on pooling shows that the classification accuracy can be improved by choosing more carefully the pooling regions. Traditionally the pooling regions were the cells of the spatial pyramid in different levels. The work of [12] introduced spatial regions that do not follow the grid division of spatial pyramid and capture better dataset-specific spatial information. More close to our approach is the work of [13] where they exploit the partitioning in the feature space together with a spatial pyramid partition. However both these methods rely on a visual vocabulary.

3 Background

3.1 NBNN

For the NBNN classification we use the formulation of [14], which is based on [1]. The sampling of features is motivated by saliency and is similar to how the human eye process an image: starting from a feature g_1, the sampling continues to other features g_t based on the saliency distribution $p(g_t|C = k)$ for a specific class k. We assume statistical independence between the points given the class label:

$$P(\{g_t\}_1^T|C = k) = \prod_{t=1}^{T} P(g_t|C = k), \tag{1}$$

with T the number of features. By applying Bayes rule we obtain:

$$P(C = k|\{g_t\}_1^T) = P(C = k) \prod_{t=1}^{T} P(g_t|C = k). \tag{2}$$

with $P(C = k)$ considered uniform across classes. For the term in the product, the NBNN approach suggests the use of kernel density estimation that is approximated by the 1-nearest-neighbor term as follows [1]:

$$P(g_t|C = k) \propto \max_i \exp(-\|w_{k,i} - g_t\|_2^2), \tag{3}$$

where $w_{k,i}$, g_t are the original feature vectors of training and test features respectively. As described in [1], the spatial coordinates of each feature can be added to the feature vector, which has shown to improve results for object centered datasets like Caltech 101.

Finally, taking the log of the class posterior leads to the well know sum over minimum distances (nearest neighbors) of NBNN:

$$\begin{aligned} \log P(C = k|\{g_t\}_1^T) &\propto \sum_{t=1}^{T} \max_i(-\|w_{k,i} - g_t\|_2^2) & (4)\\ &\propto \sum_{t=1}^{T} \min_i \|w_{k,i} - g_t\|_2^2 & (5) \end{aligned}$$

3.2 Kernelized NBNN

In [2], the NBNN formulation has been extended to a discriminative scheme, the *NBNN* kernel. In this way, they were able to introduce the benefits of NBNN in a multiple kernel framework, where it can be combined with other kernels. In this paper we follow their approach for kernelizing the NBNN. However, in this work we focus on the exploitation of the information that is hidden inside the classes themselves and thus, we do not apply the combination with a bag-of-words scheme or phow kernels.

For the implementation of the NBNN kernel, we follow the formulation of [2]. The comparison of two feature sets $X = \{\mathbf{x}\}$ and $Y = \{\mathbf{y}\}$ is based on the *normalized sum match kernel* [15–17]:

$$K(X,Y) = \Phi(X)^T\Phi(Y) = \frac{1}{|X||Y|}\sum\sum k(\mathbf{x},\mathbf{y}). \tag{6}$$

The local kernel $k(\mathbf{x},\mathbf{y})$ compares local features $\mathbf{x}$ and $\mathbf{y}$ and is computed as the dot-product between two vectors $\phi(\mathbf{x})$ and $\phi(\mathbf{y})$ with dimension equal to the number of classes and each element focusing on one particular class. $\phi(\mathbf{x})$ is a function of the distances from the feature $\mathbf{x}$ to its nearest neighbors in the different classes $c \in C$:

$$k(\mathbf{x},\mathbf{y}) = \phi(\mathbf{x})^T\phi(\mathbf{y}), \tag{7}$$

$$\phi(\mathbf{x}) = [\phi^1(\mathbf{x}) \quad \phi^2(\mathbf{x}) \quad \dots \quad \phi^{|C|}(\mathbf{x})], \tag{8}$$

$$\phi^c(\mathbf{x}) = d^c_{\mathbf{x}} - d^{\bar{c}}_{\mathbf{x}}, \tag{9}$$

with $d^c_{\mathbf{x}}$ the distance to the nearest neighbor of $\mathbf{x}$ belonging to class c (calculated using the exponent shown in the right hand side of equation 3) and $d^{\bar{c}}_{\mathbf{x}}$ the distance to the nearest neighbor from all classes except c. As shown in equation 9, we follow the suggestion of [2] and do not use the nearest-neighbor distances directly but rather the original distance minus the distance to the nearest other class.

4 The pooled NBNN kernel

Our new image representation is based on two observations. First, the NBNN formulation in equation 4 can be understood as first performing max pooling over a code that is formed by computing L_2 distances to all features in the training set and then performing spatial sum pooling over all features in the test image. In this paper, we investigate more general pooling strategies for the max operator. While the NBNN approach performs max pooling (finds nearest neighbors) within the features of a certain class, we propose different pooling strategies that are more general. They can be more specific by only pooling over sub-classes or more general by not necessarily obeying class boundaries. The choices that we investigate for such generalized pooled NBNN features are detailed in section 4.1.

Second, while these NBNN features are not necessarily directly useful for classification anymore, we realize that the NBNN kernel [2] uses the class distance as a type of features. Therefore, we propose to include our generalized NBNN features into this kernel learning framework, so that the learning algorithm can exploit the different views on the data. Our experimental results support our claim that our new representation indeed captures complementary information which is useful for visual recognition

Finally, different partitions employed in the pooling scheme can be combined in a multikernel framework, resulting in a new kernel that effectively exploits different levels of granularity (from very specific to very generic). Our experiments show a consistent improvement by incorporating more partitions. Moreover, as we show later, evaluating this kernel is not much more expensive than evaluating the original NBNN kernel.

4.1 Partitioning the feature space

As described above the NBNN scheme implies a coding of features in terms of L_2 distances to the features in the training set. In contrast to the typical spatial pooling employed for visual recognition (which would relate to variations to the sum operator in equation 4), our method targets the max pooling operator in equation 4 that pools in appearance space. The NBNN suggests to pool over all features of the same class. In this section, we propose more general strategies.

The basic *structure* that features belong to is by definition the image, since the feature distribution is populated by features extracted from (training) images. However, we can generalize and assume that in a particular image one or more processes generated the data. This is also the assumption behind the visual word notion: that structures with similar appearance come from the same process.

We derive additional structures at different levels of granularity by clustering the features (or sets of features). In this process, we define an **atom** as an indivisible feature group that a cluster of features builds upon, and the **cluster-level** as the set of features to which the clustering is applied. In case of hierarchical clustering, the atoms correspond to the leaf nodes, while the cluster-level determines the root node. In order to further clarify our framework, we start by describing the image-to-class model for object classification according to our new terminology.

Image to Class. In the case of the image-to-class (I2C) distance framework [1], the atoms are images and the cluster-level is the class level. The reason why we consider the images as atoms and not the features directly is because the set of features from a training image can not be separated. In other words, we can not assign features from the same image to different clusters. Note that in this case no clustering algorithm is actually performed and the assignment of atoms to clusters is determined solely based on the annotation (a cluster contains all the features from images with the corresponding class label).

In this paper we focus on two extensions:

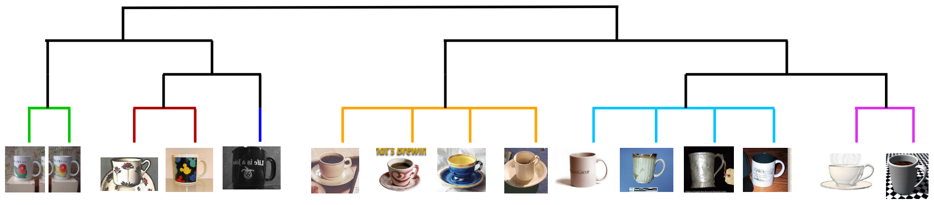

Fig. 2. Visualization of the hierarchical partition of class *cup* into 2,4 and 6 subclasses.

1. The atoms are individual images and the cluster-level is the class. This case is similar to the I2C case, but clustering is performed at the class level, resulting in a number of subclasses for each class. We refer to this case as *Image to Subclass* (I2Sub).
2. The atoms are individual features and the cluster-level is the class. The cluster in the feature space can be seen as a large visual word. Therefore, for a new image we calculate the *Image-to-Word* distance (I2Word).

Image to Subclass. For the first case, where we are looking in every class separately and the building block is the image, we need to define a measure of similarity between the image feature distributions. Here we use the image-to-image distance, i.e. the sum of the distances of features in image 1 to their nearest neighbors in image 2. This way we obtain a distance matrix of all images belonging to the same class. Based on the distance matrix we apply agglomerative clustering with complete linkage. The result of the clustering shows which images are clustered to the same subclass. Figure 2 illustrates two classes with two of their subclasses from the Caltech-101 dataset.

Each subclass contains images from the original class that are similar in feature space, resulting in a more coherent set of features in feature space. A new test image is then compared to all the subclasses separately. This does not only provide more information to the final classifier, but also reduces the chance of false positives ('accidentally' obtaining a small distance-to-(sub)class is less likely since the subclasses are more compact in feature space). The computation of the distances to subclasses is similar to Eq (8)[3].

Image to Word. For the second case, we partition our features based on their location in feature space, without imposing features from the same image to be assigned to the same cluster. The cluster level is the class, and therefore features from images of different classes will never end up in the same cluster. As a result, the clusters we obtain are somehow similar to class specific visual words. However, note that we do not perform vector quantization, so this is only a partial analogy. Clusters are created by applying K-means on all features from

[3] During training, if a subclass consists of only 1 image, its distance to its own subclass is the average of distances of all images within the class.

a class. After that, the original class label is no longer used. For a new image, we then calculate the distance to all class specific "visual-word-like" clusters [4].

4.2 Computational considerations

The clustering of features into subclasses and word clusters is performed once during training. For a new test image the procedure is similar to [1, 2]. For each feature in the image we find its nearest neighbor in every class, subclass or word cluster and calculate the distance to that cluster of features using equation (3). Note however that all the partitions of the feature space are performed in the class level. Therefore, we know for each training feature to which partition it belongs (class, subclasses, clusters). Now during testing, the small number of NIMBLE features per image (100) allow us to compute the distance of a test feature to all training features in a class. Thus, by performing a search algorithm we can efficiently find the 1st nn for every semantic partition, without the need for nn search for every kernel case (the nn search time only increases from 7.7 s per image to 9.2 s per image for 15 scenes).

5 Experiments

In this section we provide quantitative results for the proposed framework. We investigate different partitions of our feature sets as described in section 4 and combine the resulting kernel representations in a multi-kernel framework. For combining the kernels, we simply use an average kernel as previous results [6] have shown that typical multi-kernel learning schemes tend to results in the same performance range as the average kernel with the cost of additional parameter selection. Our investigations have confirmed this observation for our setting too.

Datasets. In order to embed our results in the context of previous methods and compare to the state-of-the-art we have chosen the following three, widely used datasets: Caltech 101 [18], 15 Scenes [4] and Oxford Flowers 17 [19]. Caltech 101 provides a large number of classes and large intra-class variation, which we believe can be captured well by our distance-to-subclass approach (see also figure 2). 15 Scenes dataset on the other hand contains less classes and the intra-class variation is smaller but each class contains a larger number of images, which makes it better suited for svm training. The Oxford Flowers 17 dataset presents high intra-class variation and difficult separation across classes. Moreover, each class contains a large number of images.

Implementation details. We use 4 random train/test splits in order to infer the results presented in this paper (every result is the average over 4 splits). For generating the train/test splits in all datasets we use the framework of [14], that picks random train and test images from each class. Note that, unlike [20, 2], we did not add jittered images.

[4] But note that the number of clusters is typically kept low, and therefore features within a single cluster show more variability than the variability usually observed in visual words.

The image representation in our paper is based on the NIMBLE features proposed in [14]. For the experiments reported in this work we use the publicly available code of [14] for extracting the features and the same training/testing setup. We sample 100 features per image, and each of these is described with a 500 dimensional descriptor. This allows for fast and exact nearest-neighbor matching. While NIMBLE features are a convenient choice in the context of NBNN methods, thanks to their sparsity, we would like to stress that the method proposed in this paper is generic and by no means limited to one particular type of features. The location information was used as additional dimension in the feature vectors similar to [1, 14], weighted with a scalar α (1 for Caltech and 17 Flowers and 0.5 for 15 Scenes). We compute the NBNN distances using the code of [21] (FASTANN) for exact nearest neighbor. For training and testing the support vector machines we use the liblinear package [22] with L_2-regularized L_2-loss support vector classification (primal).

When computing the I2C and I2Subclass distances for the training images, we exclude the image itself from the class set. This way the distance of a training image to the class it belongs to better represents the distance that a testing image will have to the particular class (even if it results in a slightly lower number of class features in those cases, since one image is removed). An alternative could be to remove only the feature under consideration. This approach is followed for the I2Words experiments.

Note that we also report results for vanilla NBNN (i.e., non-kernelized). Since the clusters in our experiments are always class-specific, this is indeed possible, using the following routine: for a new image, we compute the distances to each of the partitions for all features, select the cluster with the smallest cumulative distance, and assign the label of the corresponding class.

15 Scenes. This dataset contains images from indoor and outdoor scenes

Method	NBNN	NBNN kernel
I2C (baseline)	72.0±0.3	76.0 ± 0.8
I2Sub-2	72.3±1.5	76.4 ± 0.4
I2Sub-4	68.9±1	76.3 ± 2.3
I2Sub-6	68.1±3.1	76.6 ± 1.0
I2Sub-10	66.1±8	77.0 ± 0.5
I2Sub-20	66.2±1.7	76.4 ± 0.4
I2Word-2	70.7±0.8	74.8 ± 1.3
I2Word-4	67.0±0.6	74.8 ± 0.5
I2Word-8	63.5±1.3	75.9 ± 0.3
I2Word-16	61.0±0.6	74.4 ± 0.7
I2Word-32	60.2±3.2	71 ± 3
Average Kernel	-	**79.7±1.5**

Fig. 3. Classification accuracy of NBNN and NBNN kernel for different pooling strategies in 15 Scenes.

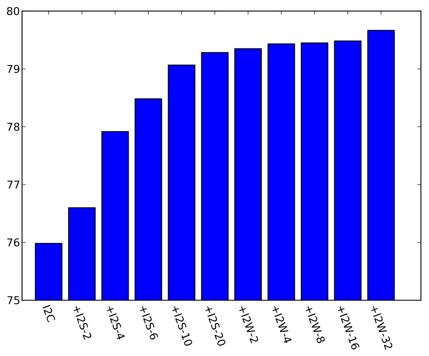

Fig. 4. Performance gain for 15 Scenes when adding kernels. The order of addition is the same order as in Table 3.

belonging to 15 different classes. Both for training and testing we use 100 images. Figure 3 shows the results for the individual partitions as we increase the number of subclasses (I2Sub*) or the number of words (I2Word-*). Using complete agglomerative clustering, we generate for every class 2, 4, 6, 10 and 20 subclasses. For the generation of the word clusters, we use K means with 2, 4, 8, 16 and 32 clusters.

In accordance with previous findings [2], when using image-to-class comparisons, the NBNN kernel outperforms NBNN in this setting with a performance of 76% versus 72.0%. For NBNN we see a gradual decrease when going to the more fine grained representations. For the NBNN kernel the results seem to be more stable around $75 - 77\%$. We attribute this to the discriminative training. The kernel framework allows us to further boost performance by combining all the representations in a multi-kernel setting. With a total performance of 79.7% we outperform the best individual kernel by over 2.5% and the NBNN baseline by almost 8%. We consider this as a strong indication that our approach indeed derives representations that capture different aspects of the object classes.

Figure 4 further illustrates the performance gain by adding a single kernel at a time according to the ordering in Table 3 (top to bottom). We observe a consistent improvement by gradually enriching the representation and we achieve strong performance even compared to methods that use a more elaborate spatial model (*e.g.* best results reported in [23, 2] rely on a Spatial Pyramid). In 15 Scenes NIMBLE seems not to preform as well as on the other datasets due to the small number of features and saliency map based sampling instead of dense sampling.

17 Flowers. This dataset contains 17 different flower classes with each of them consisting of 80 images. For our experiment we use 60 images for training and 20 for testing. The results are summarized in Table 5. Similar to the previous experiment, we observe better performance for the kernelized NBNN compared to NBNN, since the number of training images is large enough.

This experiment is in line with the previous one regarding overall trends and the relative performance of the different kernels. The individual kernels do not exceed the performance of the I2C kernel but their combination again results in an increase in performance. In particular, our final result achieves state-of-the-art results (to the best of our knowledge) compared to other methods that only use one feature. ([24] Color-opponent-SIFT 80%, result taken from [25]).

Caltech-101. Finally, we report results on Caltech-101 [18]. While criticized by many, it is still a widely used dataset, that allows us to compare our method against a wide range of methods in the literature.

First, we study the behavior of different representations derived by our pooled NBNN kernel on Caltech101 using only 15 training images (see figures 7 and 10). For this experiment we investigate 2,4 and 6 subclasses and 2,4,8 and 16 word clusters. We stop at this granularity due to the limited number of training images and hence lack of data in this setting. The findings are consistent with the previous two datasets (adding kernels increase the accuracy, Figure 10). We see a steady decrease in performance for the NBNN method while the NBNN

Method	NBNN	NBNN kernel
I2C (baseline)	81.9±1.9	83.08±1.5
I2Sub-2	81.9±1.9	82.9±4.2
I2Sub-4	81.1±0.6	82.5±1
I2Sub-6	79.6±1.6	82.6±2
I2Sub-10	78.3±3.8	82.7±2.1
I2Sub-20	77.2±2.1	81.7±4.1
I2Word-2	80.6±0.1	82.3±0.4
I2Word-4	77.8±0.8	83.5±1.6
I2Word-8	72.3±7.5	80.3±1
I2Word-16	66.9±14	82.3±1.2
I2Word-32	66.0±11	80.8±0.7
Average Kernel	-	**85.3±3.1**

Fig. 5. Classification accuracy of NBNN and NBNN kernel for different pooling strategies in 17 Flowers.

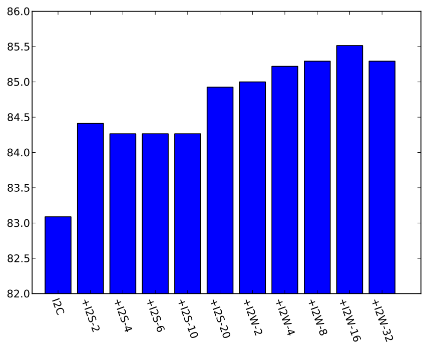

Fig. 6. Performance gain for 17 Flowers when adding kernels. The order of addition is the same order as in Table 5.

Method	NBNN	NBNN kernel
I2C	70.6±0.9	66.6±0.8
I2Sub-2	68.8±1.3	67.7±0.9
I2Sub-4	66.5±1	67.3±1.3
I2Sub-6	65.3±1.5	67.4±0.1
I2Word-2	64.8±0.9	68.6±0.6
I2Word-4	59.7±2.2	68±3
I2Word-8	54.5±2	61±7
I2Word-16	51.0±1	43±3
Average Kernel	-	**71.9±0.3**

Fig. 7. Classification accuracy in Caltech 101 (15 images).

Method	NBNN	NBNN kernel
I2C	77.8±0.3	76.7±0.5
I2Sub-5	73.5±1	77.1±1.1
I2Sub-10	71.9±1.6	77.2±2.6
I2Sub-15	71.1±2.2	76.2±5.2
I2Word-2	72.2±1.3	75.3±1.4
I2Word-4	67.3±1.8	75.4±1.7
I2Word-8	61.2±0.6	68.8±1.4
I2Word-16	57.5±1.6	50.8±13
Average Kernel	-	**81.0±1**

Fig. 8. Classification accuracy in Caltech 101 (30 images).

kernel roughly remains constant (or slightly increases). We observe that the NBNN kernel performance at 66.6% is lower than the NBNN baseline at 70.6%. This is an issue already encountered in previous work [2] and is owed to the small number of available samples that makes holding out samples for NBNN representation and classifier learning difficult. However, when combining all the kernels we again manage to (slightly) outperform the NBNN approach with a total performance of 71.9% – even in this unfavorable setting. Hereby, we manage to cure a problem that previous settings were troubled with in the low sample regime. In the second experiment on Caltech 101 we use 30 images for training (Fig. 8). When the number of images is increased the NBNN kernel performs better.

Finally, in Table 9 we compare our best results (the average kernels including I2C, I2Sub and I2Word) to other approaches in the literature using a single feature type, on Caltech 101 with 15/30 images. To the best of our knowledge, for 15 training images our system gives the highest performance among all methods using a single descriptor.

Method	15 images	30 images
Gehler *et al.*. [6]	61.0 ± 0.2	69.4 ± 0.4
Griffin *et al.*. [26]	59.4	67.6 ± 1.4
NBNN [1]	65.0 ± 1.1	~ 72.4
LLE [27]	65.43	73.4
Vedaldi *et al.*[20]	66.3 ± 1.1	-
ScSPM [28]	67.0 ± 0.5	73.2 ± 0.5
Pinto *et al.*. [29]	61.4	67.4
NIMBLE [14]	70.8±0.7	78.5±0.4
NBNN kernel+phow [2]	69.2 ± 0.9	75.2 ± 1.2
GLP+LLC+SPM [23]	70.3	**82.6**
ours	**71.9±0.3**	81±1.0

Fig. 9. Comparison of the proposed method with state-of-the-art (using only one feature type).

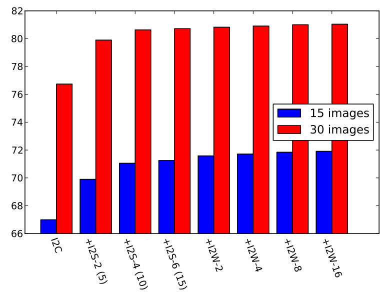

Fig. 10. Performance gain for Caltech 101 when adding kernels.

6 Conclusion

We have presented a series of object representations that extend previous image-to-image and image-to-class distances to a more general framework that models an object class with a set of distances to partitions of the features space. This representation has an interesting dual interpretation as pooling over features from the reference images. In particular it provides new insights to the NBNN classification scheme in the context of feature coding and pooling. Therefore we call it the pooled NBNN kernel. We explore a variety of partition schemes including subclasses (obeying image constraints and class boundaries) and 'class-specific visual word'-like clusters (obeying class boundaries only) and combine them in a multi-kernel framework.

A combination of all the kernels we derive leads to consistent performance improvements which lets us conclude that the associated representations indeed capture different aspects of the object classes. The resulting kernel achieves strong performance across 3 widely used recognition benchmarks and in particular outperforms the state of the art on Caltech101 and 17 Flowers using a single feature.

Acknowledgment. We acknowledge the support from ERC grant Cognimund.

References

1. Boiman, O., Shechtman, E., Irani, M.: In defense of nearest-neighbor based image classification. In: CVPR. (2008)
2. Tuytelaars, T., Fritz, M., Saenko, K., Darrell, T.: The nbnn kernel. In: ICCV. (2011)
3. Grauman, K., Darrell, T.: The pyramid match kernel: Efficient learning with sets of features. In: JMLR. Volume 8. (2007) 725–760
4. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: CVPR. (2006)

5. Csurka, G., Dance, C.R., Fan, L., Willamowski, J., Bray, C.: Visual categorization with bags of keypoints. In: In Workshop on Statistical Learning in Computer Vision, ECCV. (2004) 1–22
6. P.V.Gehler, S.Nowozin: On feature combination for multiclass object classification. In: ICCV. (2009)
7. Marszałek, M., Schmid, C.: Constructing category hierarchies for visual recognition. In: ECCV. (2008)
8. Deng, J., Berg, A.C., Li, K., Fei-Fei, L.: What does classifying more than 10,000 image categories tell us? In: ECCV. (2010)
9. Zhigang, L., Wenzhong, S., Qianqing, Q., Xiaowen, L., Donghui, X.: Hierarchical-support vector machines. In: IGARSS. (2005)
10. Xia, S., Li, J., Xia, L., Ju, C.: Tree-structured support vector machines for multi-class classification. In: ISNN. (2007)
11. Gao, T., Koller, D.: Discriminative learning of relaxed hierarchy for large-scale visual recognition. In: ICCV. (2011)
12. Jia, Y., Huang, C., Darrell, T.: Beyond spatial pyramids: Receptive field learning for pooled image features. In: CVPR. (2012)
13. Boureau, Y., Le Roux, N., Bach, F., Ponce, J., LeCun, Y.: Ask the locals: multi-way local pooling for image recognition. In: ICCV. (2011)
14. Kanan, C., Cottrell, G.: Robust classification of objects , faces , and flowers using natural image statistics. In: CVPR. (2010)
15. Lyu, S.: Mercer kernels for object recognition with local features. In: CVPR. (2005)
16. Bo, L., Sminchisescu, C.: Efficient match kernel between sets of features for visual recognition. In: Advances in Neural Information Processing Systems. (2009)
17. Haussler, D.: Convolution kernels on discrete structures. In: Technical Report. (1999)
18. Fei-Fei, L., Fergus, R., Perona, P.: Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories. In: Workshop on Generative-Model Based Vision. (2004)
19. Nilsback, M.E., Zisserman, A.: A visual vocabulary for flower classification. In: CVPR. (2006)
20. Vedaldi, A., Gulshan, V., Varma, M., Zisserman, A.: Multiple kernels for object detection. In: ICCV. (2009)
21. Philbin, J., Chum, O., Isard, M., Sivic, J., Zisserman, A.: Object retrieval with large vocabularies and fast spatial matching. In: CVPR. (2007)
22. Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., Lin, C.J.: Liblinear: A library for large linear classification. In: JMLR. Volume 9. (2008) 1871–1874
23. Feng, J., Ni, B., Tian, Q., Yan, S.: Geometric lp-norm feature pooling for image classification. In: CVPR. (2011)
24. van de Sande, K.E.A., Gevers, T., Snoek, C.G.M.: Evaluating color descriptors for object and scene recognition. PAMI **32** (2010) 1582–1596
25. Fernando, B., Fromont, E., Muselet, D., Sebban, M.: Discriminative feature fusion for image classification. In: CVPR. (2012)
26. Griffin, G., Holub, A., Perona, P.: Caltech-256 object category dataset. (2007)
27. Wang, J., Yang, J., Yu, K., Lv, F., Huang, T., Gong, Y.: Locality-constrained linear coding for image classification. In: CVPR. (2010)
28. Yang, J., Yu, K., Gong, Y., Huang, T.: Linear spatial pyramid matching using sparse coding for image classification. In: CVPR. (2009)
29. Pinto, N., Cox, D.D., DiCarlo, J.J.: Why is real-world visual object recognition hard? In: PLoS Computational Biology. (2008)