University of Amsterdam

Artificial Intelligence

Master Thesis

Naive Bayes Nearest-Neighbor Object Detection

Using Local NBNN and Exemplar Models to Perform Object Detection in Web Images

Author:
Maarten van der Velden
Student ID: 5743087
maarten.vandervelden@student.uva.nl

Supervisor:
dr. Jan van Gemert
j.c.vangemert@uva.nl

May 8, 2013

Abstract

This thesis explores the possibilities of using the Naive Bayes Nearest-Neighbor (NBNN) image classification method as a detection algorithm to find objects in digital photos. It improves and elaborates on similar earlier attempts by combining several related theories and methods. NBNN has been proven to be a good parameter-free image classification algorithm, that works without a learning phase. Extensions into the field of object detection have been involving the clustering of exemplar-models (exemplar-NBNN), whereas others improved NBNN by investigating the local nearest-neighbor search space (LNBNN).

Exemplar-NBNN's agglomerative clustering approach was evaluated and exchanged with the theoretically more appealing mode finding algorithm called quickshift. The result was combined with LNBNN to involve multiple nearest neighbors.

Tests on the TUD Motorbikes and Pascal VOC 2007 benchmark sets show an improved object detection algorithm, that performs reasonably well compared with other approaches. Further research is needed to explore its possibilities in more detail. Not only with respect to performance, but also to its efficiency, as time complexity appear to be prohibitive for large data sets at the moment.

1 Introduction

Object retrieval is a main topic in the research area of computer vision. Most methods for finding objects are designed specifically for certain subtopics, such as retrieving one type of objects only, categorizing the scene of the image, or discerning foreground objects from the background. For each of these subtopics, the object finding task is modeled in a certain way. Three very common models are image classification, image segmentation, and object detection.

In image classification (Figure 1a), the object finding task is represented as the task of predicting the class of a whole image. The prediction is some measure of how likely it is for an image to belong to a certain class. This task is useful when the image as a whole is of importance, for example when a user is looking for images of a certain topic.

Image segmentation is different from image classification, because for each pixel in the image, a class label has to be found.(Figure 1c) This task is more complex than classification, because there is less information available for a small patch of the image than for the image as a whole. Therefore, the context of each segment becomes more important for a good classification of objects. Image segmentation might be useful when trying to find the distinction between foreground and background, for example.

Object detection (Figure 1b) is similar to both image classification and segmentation, and lies somewhere in between these tasks. Detection is the task of pinpointing areas on an image where objects are, and of which class this object is. Detection is more specific than classification, because the objective is not only to give a class label, but also an indication of the object's location. This indication is not as specific however as in the segmentation task, because the goal is not to define a class for all pixels in the images, but only for the foreground objects. the indication of the object's location can be given in a number of ways, but the most common one is to give a rectangular bounding box that

envelops the object [12]. Object detection is useful when you are interested in only specific parts of the image, for example in tasks like face detection.

(a) *Classification* **(b)** *Detection* **(c)** *Class-segmentation*

Figure 1: *Three Computer Vision tasks for the same image. If the question is to find persons and airplanes, in a classification task* ***(a)*** *it would be enough to classify the image as a whole as containing both classes. In a detection task* ***(b)*** *bounding boxes need to be given that neatly envelop the objects and have the correct class label. In a class-segmentation task* ***(c)****, every pixel should be labeled correctly "airplane", "person" or "background" (transparent in this case). Note that this segmentation image has a blank label for areas that are either on the boundary between different classes, or difficult to classify (at the left side of the image). [12]*

These kinds of algorithms can be used in a broad range of applications. Image retrieval from the internet can be improved by using high level visual data next to textual data such as the filename or context of an image in a web page, especially as the amount of images on the internet grows much faster than they can be for example tagged by useful keywords. Various dedicated systems might use object recognition. For example, in robotics object detection might be used to determine where in the range of the robot certain objects are (and to interact with them accordingly). Another application field is object tracking in video footage. If objects can be successfully detected in single images, it is possible to use this information in an image sequence to learn the trajectories of objects within the sequence. This is useful in surveillance settings, or to gather individual statistics of players in team sports. [3, 11, 19]

While most humans have little difficulty categorizing visual input, recognizing objects or classes of objects, and tracking objects in continuous sequences through time, object recognition is regarded a difficult task. A computer basically "sees" an image as a bunch of numbers representing color values for each image pixel. When the image is not an exact copy of a known image, let alone an image of an object not seen before by the computer, it becomes hard to see sometimes crucial similarities. In an average recognition setting, this is the case almost all the time. When comparing images of a certain object of interest, the images tends not to be at the exact same location all the time, the lighting conditions vary, just like the viewpoint and distance of the camera, the camera quality, the image size. The object might be cut off by the edge of the image, or may be occluded by another object. When multiple objects of the same class need to be compared the issues become even bigger. The intra-class variety might be huge (think of all different kinds of chairs you have ever encountered), causing a large variance of appearance.

Therefore, higher level representations are needed to compare images, objects, and classes of objects. To this end, several kinds of image features have been developed that can describe parts of images at a higher level. These features usually cover a small part of the image, and describe numerically what this

part looks like. For example, sharp color contrast, or the occurrence of lines, corners or blobs might be of interest, just like the size and direction of these properties. The features should usually be invariant to a number of (irrelevant) variables, and be covariant to important ones. These features make it easier to compare images and objects, because the question becomes: "Which class' features do the features of this image resemble most?"

A number of different approaches have been developed to find a way to answer this question, or basically to classify images or objects. Usually these involve the analysis of a number of labeled images (the training phase), in which a model is learned for classifying new images. in the test phase, unlabeled query images are subjected to this model, and a decision is made which class is most likely.

For image classification, recently the Naive Bayes Nearest Neighbor (NBNN) [5] approach has gained popularity. [1, 2, 22, 27, 28, 38, 40] Boiman *et al.* apply the simple nature of nearest-neighbor classification to build a state-of-the-art image classification method. They show that a Nearest-Neighbor based approach for image classification has a number of appealing properties. In the first place, Nearest Neighbor is a so-called "lazy" classification method, meaning that it does not need a separate phase in which it learns a model. It does not build a model for a decision boundary based on the evidence given, but is just compares each query directly with this evidence. This also makes the method parameter-free, meaning that there are no parameters needed that influence the model that is made.

Because no model is learned, the raw evidence is of much importance in NBNN, therefore the more evidence is available, the better. Parametrized methods benefit from little but good evidence, but without a learning step, the quantization error this gives is very harmful. Furthermore, Boiman shows that query images should not be compared to evidence images, but to the aggregated evidence classes, image-to-class distance.

Among improvements by others, McCann & Lowe [22] have come up with an adaptation of NBNN which looks at the local surroundings of image features in the NN-space across all classes in stead of a class-by-class approach. This Local NBNN (LNBNN) approach uses more than one single nearest neighbor to find object-to-class distances to multiple classes at once, making the process more efficient and the performance better.

In this thesis I will explore the possibilities of extending the NBNN method from image classification to object detection. I combine McCann & Lowe's LNBNN based object-to-class distance estimation with exemplar-model object detection [1, 9]. To do this, each object descriptor taken during training is regarded as an exemplar. It refers to a certain part of the object it was sampled from. In this way, bounding box hypotheses can be made from descriptors in a test image and their nearest neighbor exemplars. These hypotheses can be clustered to form detections. In this regard, single-link agglomerative clustering as used by Becker *et al.* [1] is compared with quickshift mode finding clustering. [34]

The experiments are performed on both a composed dataset of motorbike images (TUD Motorbikes) [1, 15], and on the more challenging VOC2007 object detection task [12].

The results show that exemplar-LNBNN detection is an improvement over earlier attempts to apply NBNN to an object detection task [1]. Furthermore,

the use of the quickshift algorithm a more theoretically sound clustering algorithm is shown to be justified. Nevertheless, the experiments also expose some vulnerabilities of the approach. The benefit of not having a learning phase, and therefore a short "training" time is compensated by a larger memory consumption (no efficient model to compress the evidence data) and a longer time to perform detection (comparison of each image feature to a large set of evidence features). Regardless of some efficiency improving measures, this remains a problem when it comes to large tests.

Section 2 gives an overview of related work on the various parts of this task. In Section 3 I will discuss the details of the NBNN method, and the assumptions under which it works. In Section 4 the theory behind exemplar-based modeling will be explained. The link between the two methods will be made in Section 5. In Section 6 the experiments will be elaborated, after which the results are given. In Section 7 these results are discussed and analyzed. Finally, in Section 8 conclusions will be drawn and discussed.

2 Related Work

There has been a lot of research into image classification and object detection lately. A number of competitions have sparked development of a broad range of approaches to get the best results. There is not one correct way of performing object recognition, there is a multitude of methods that all have their merits and disadvantages. The NBNN approach also got more attention recently.

2.1 Image Classification

A lot of different approaches to image classification exist, and achieve competitive results. Most of these algorithms are somewhat related to NBNN, but most of them involve a learning phase where a model of the data is constructed, which is unnecessary in NBNN.

Most state-of-the-art image classification methods are based somehow on an approach known as visual codebook, Bag of Words (BoW) or Bag of Features (BoF). [7, 17, 20, 32, 31, 36] In this approach, local image features are regarded as visual words, which are stored in a dictionary, or visual codebook. The visual words in the codebook usually are a representation of a local cluster of features. In this way, each feature in an image is mapped onto a certain visual word, and the image as a whole can be seen as an unordered collection (or bag) of visual words. In this way, images can be represented as histograms of visual words, binning each image feature into a visual word, and counting the numbers of visual words.

Now, the task of classification is to compare these histograms between labeled training images and unlabeled test images. Often [7], support vector machines are used to learn a discriminative model of how to classify images into classes.

BoW has achieved many good results in image classification, and many adaptations of the method have been proposed. An example is the use of a spatial pyramids to keep the spatial relation between features intact [17], instead of discarding all relations between features. Much effort has been put in encoding the visual words into image descriptors. [7] Instead of making histograms,

(a) *Histogram Encoding*

(b) *Kernel Encoding*

(c) *Fisher Kernel Encoding*

(d) *Locality-constrained Linear Encoding*

Figure 2: *Various encoding formalizations of visual words into a codebook.*

methods such as soft-assignment kernel codebooks [20, 32], Fisher kernels and locality-contrained linear encoding (LLC) [36] have been proposed. see Figure 2.

Kernel codebooks and soft-assignment codebooks improve results by taking into account the ambiguity of some features [20, 32]. Instead of assigning each feature to a single visual word, this approach assigns the features to visual words weighted by their distance to each other (usually through a kernel), see Figure 2b

A different encoding of BoW is with the use of Fisher kernels. [26] This method replaces the discriminative SVM learning phase by learning a Fisher kernel based on a Gaussian Mixture Model. The Fisher kernel does not only take into account the distance between feature and visual word for binning purposes, but encodes the feature distribution explicitly, as can be seen in Figure 2c.

A fourth alternative used is Locality-constrained Linear Coding (LLC), [36] which represents each feature as a linear combination of a number of nearby visual words, pooling these into an image representation. This approach is somewhat similar to soft-assignment.

A main difference between BoW and NBNN is that the former method compares images to images comparing image histograms, whereas NBNN compares images to all features of a certain class. Furthermore, BoW is a model-learning method, while NBNN is model-free.

2.2 Object Detection

The most intuitive way of thinking about object detection is probably to apply image classification at various windows within the image instead of on the image as a whole. This involves iterating over possible window locations, sizes and aspect ratios for the whole image, and determining the likelihood of each window of representing an object. This so called sliding window approach marks early

detection methods [35]. The applicability of this approach however is fairly limited, because of the large number of possible windows to check. Therefore, many methods find a way to make this window search more efficient. Viola & Jones [35] propose a cascade approach, where a very simple classification method is used on the full set of hypotheses for bounding boxes in order to cast most of them away early. For difficult hypotheses a more sophisticated classification is done to narrow down the search, each step using a better, and much slower classification algorithm. In contrast, Efficient Sub-window Search methods [2, 16, 25, 39] model the problem into a branch-and-bound search method. They recursively split the window in two, find the response for the class on the current scale, and continue with the most promising leaf. When the response of both windows after a split is lower than the one above, the correct window is assumed to be on the above level.

Another approach that recently gained more attention because of the promising results it getting, is that of detection by segmentation. [30, 42] These methods rely on the fact that segmentation methods are meant to subdivide the image into segments that represent a semantic unity, like parts of objects or full objects. The resulting segments can be used as hypotheses for detecting objects. This means the amount of possible windows can be reduced heavily. Van de Sande *et al.* [30] use a hierarchical segmentation algorithm to make the detection scale invariant, and train discriminatively by focusing on hard examples. Zhang *et al.* [42] do not explicitly segment the image, but just like many segmentation algorithms they do look for edges that enclose an object as a restraint for selecting it as a possible detection.

Part-based models form a different approach on effectively finding hypothesis windows for objects [14]. These methods learn object models based on a combination and spatial organization of a number of designated, but unlabeled, parts. These parts are learned as a hidden variable during training, being groups of features reoccurring in the same formation in a certain area of bounding boxes of a class. Furthermore, the difference in scale between the full object window (the root) and its parts is fixed. In comparison with sliding-window approaches, this means a restriction in the number of possibilities for detection of objects. The relative scale of the parts should comply with that of the root scale.

Figure 3: *From [14].*

2.3 NBNN Based Methods

Naive Bayes Nearest Neighbor (NBNN) was introduced as an image classification algorithm by Boiman *et al.* [5] Its details will be explained in Section 3

Figure 4: *"Given an image (a) our aim is to find its objects for which the ground truth is shown in (b). To achieve this, we adapt segmentation as a selective search strategy: We aim for high recall by generating locations at all scales and account for many different scene conditions by employing multiple invariant color spaces. Example object hypotheses are visualized in (d)." From [30].*

In recent years, it has gained recognition as an alternative approach to BoW classifications, and many adaptations have been made since. [1, 2, 22, 27, 28, 38]

Some of these authors have been pointing out shortcomings of the NBNN method, and have come up with solutions. [2, 22, 27, 38] Some authors [2, 38] stress that NBNN is highly sensitive to differences in descriptor density over classes. Behmo *et al.* for example state that this is caused by dropping the factor from the classification rule which normalizes for the descriptor density in the train set for a given class. This term is dropped because an equal kernel estimation is assumed, which does not hold for large differences in descriptor density over classes. [2] Therefore, they propose to learn the density estimation parameters per class as a linear problem. This new formulation models an affine transformation on the distance measure used in NBNN. These two parameters can be seen as corrections for each class on the bias that arises when classes with the same priors are not equally sampled. Wang *et al.* [38] take a slightly different approach to solve the same problem. They try to correct the distances found by replacing the euclidean distance of NN by a learned Mahalonobis distance for each class.

McCann *et al.* [22] have been focusing on the meaning of retrieving the nearest neighbor of each class involved. They improved performance of NBNN by looking at the local neighborhood of each descriptor over all classes, taking into account not only a single NN per class, but the kNN overall. They show that this gives a better estimate of the local surroundings of each descriptor. This approach is also used in the thesis, see Section 3.4.1.

Other research has focused on using NBNN as input for existing methods [28] or to apply it to different tasks [1]. Tuytelaars has been modifying NBNN

into a kernelized version, using it as input for SVM classification. To this end, each image (a set of features) is represented as a vector with, for each class, the sum of each feature's NN distances (Eq. (1) – (3)). The kernel is defined as the the dot product of two image vectors (Eq. (4)).

$$\forall x \in X, \forall c \in C, d_x^c = ||\mathbf{x} - NN^c(\mathbf{x})||^2 \tag{1}$$

$$\Phi^c(X) = \sum_x \in X d_x^c \tag{2}$$

$$\Phi(X) = [\Phi^1(X) \dots \Phi^{|C|}(X)]^T \tag{3}$$

$$K(X, Y) = \Phi(X)^T \Phi(Y) \tag{4}$$

This kernel can be used as input of any kernel based classification method, such as SVM. Using this approach, even though image-to-image distance is used in the kernel, the images themselves are represented using image-to-class distances, fulfilling Boiman's constraint. Tuytelaars shows the NBNN kernel can also be a useful complementary kernel next to BoW kernels.

3 Naive Bayes Nearest Neighbor

The focus of this thesis is on the Naive Bayes Nearest Neighbor (NBNN) image classification method, first presented by Boiman *et al.* [5] In this section the theory behind the method, its sources and its later adaptations will be discussed.

3.1 k-Nearest Neighbor Classification

The k-Nearest-Neighbor algorithm is one of the earliest approaches for classification in machine learning. Given a set of items $n \in \mathcal{N}$, labeled with a designated number of classes $c_n \in \mathcal{C}$, an unlabeled query item q and a distance measure to calculate pairwise differences between items $d(x_1 \| x_2)$, the nearest neighbors of q are the k items in $\mathcal{N}$ for which $d(q \| n)$ is lowest. In turn, sets $\mathrm{NN}_c(q)$ contain those nearest neighbors of q that belong to class c. This way, kNN classification comes down to

$$\hat{c}_q = \arg\max_c |\mathrm{NN}_c(q)|, \tag{5}$$

being the class of which the most items are within the k nearest neighbors of q.

In contrast to other classification algorithms, kNN is relatively simple. It does not require any training to make a model from the labeled images. The algorithm only estimates the probability density of the classes locally, within k, which means it does not assume a global probability density distribution underneath the data, which makes it a non-parametric way of making a non-linear classifier.

3.2 Naive Bayes Classification

Naive Bayes is another rather simple classification algorithm, which is based on the idea that a simplification of Bayes' Theorem can be made using the

assumption that all features of an item $d_{n,i} \in \mathcal{D}_n$ appear conditionally independent of each other. This means that $p(d_{n,i}|c_n, d_{n,j}) = p(d_{n,i}|c_n)$ is assumed for all $n, c, i, j, i \neq j$.

If Bayes' Theorem is modeled to predict the probability of a class c of an item q, we can say

$$p(c|q) = \frac{p(q|c)\,p(c)}{p(q)} \tag{6}$$

$$\propto p(q|c)\,p(c), \tag{7}$$

because the probability is to be compared among all classes $\mathcal{C}$, for the same item q, which makes $p(q)$ constant. Furthermore, if q is represented by m features d_q, (6) can be rewritten as

$$p(c|d_{q,1}, \ldots, d_{q,m}) = p(d_{q,1}, \ldots, d_{q,m}|c)\,p(c), \tag{8}$$

which is equivalent to the joint probability

$$\begin{aligned} p(c|d_{q,1}, \ldots, d_{q,m}) &\propto p(c, d_{q,1}, \ldots, d_{q,m}) \\ &\propto p(c)\,p(d_{q,1}|c)\,p(d_{q,2}|c, d_{q,1}), \ldots, p(d_{q,m}|c, d_{q,1}, d_{q,2}, \ldots, \\ &\quad d_{q,m-1}). \end{aligned} \tag{9}$$

By the conditional dependence assumption made above, this can be reduced into

$$p(c|q) \propto p(c) \prod_{i=1}^{m} p(d_{q,i}|c). \tag{10}$$

With this equation, a decision rule can be made as follows

$$\hat{c}_q = \arg\max_c \frac{1}{m}\,p(c) \prod_{i=1}^{m} p(d_{q,i}|c), \tag{11}$$

which is called a maximum a-posteriori (MAP) classifier. In this formulation of a classification problem, a probability density estimation is needed to model all features. All kinds of distributions can be used for this.

3.3 Boiman's NBNN

Boiman *et al.* [5] coined the term Naive Bayes Nearest Neighbor (NBNN) for their image classification algorithm, that used a combination of Nearest-Neighbor (NN) and Naive Bayes to classify images in a multi-class setting. The approach works well because of the non-parametric character of the approach, which means no parameter learning is required. This makes it easy to use the method on a problem with a large number of classes, while parametric methods typically model multi-class problems as multiple 2-class problems. It also means that the risk of overfitting is smaller because there are no model parameters to be tuned. The only parameters are in the (type and amount of) features, which is the same for other classification methods. The authors achieved results competitive to the state of the art Bag of Features (BoF) methods because the method benefits from two requirements it meets: (*i*) Avoiding feature quantization and (*ii*) the use of image-to-class distance instead of image-to-image distances. They

Figure 5: *Visualization of the quantization error. (a) An image from the Face class in Caltech101. [13] (b) "Quantization error of densely sampled SIFT descriptors using a large codebook of Caltech101. Red = high error; Blue = low error. The most informative descriptors (eye, nose, etc.) have the highest quantization error." (c) "Green marks the 8% of the descriptors in the image that are most frequent in the database (simple edges)." (d) "Magenta marks the 8% of the descriptors in the image that are least frequent in the database (mostly facial features)". From [5]*

theorize that earlier attempts to use NN [4, 41] for image classification failed because these do not meet both requirements.

Feature quantization is a means of creating compact image descriptions, such as the ones used in the BoF methods. In BoF, all descriptors of an image are clustered into visual words, and histograms are constructed counting the occurrence of each word in an image. Image matching is based on comparing these histograms. Quantization is harmful for a nearest neighbor approach because the most informative descriptors get the highest quantization error while being necessary for finding nearest neighbors. This becomes clear when the observation is made that for a descriptor to be more informative, it should appear only at certain circumstances, ideally only in images of one class. They should be fairly well recognizable among other descriptors, and therefore they tend to be outliers in descriptor space. When quantizing, these descriptors will mostly be incorporated into visual words that are rather unlike these descriptors, resulting in a large quantization error. In Figure 5, the effects of this are illustrated.

In learning-based methods, like SVM, this quantization error is outweighed by the benefit of dimensionality reduction which allows training on a large data set. Furthermore, the problem itself is mitigated in BoF methods. The learning phase in these methods makes sure that only descriptors that are good predictors, regardless of their quantization error, are considered in the test phase. Because NBNN does not have a training phase nor dimensionality reduction, it has to use every single descriptor of each image to perform NN on, to avoid the quantization error.

Kernel methods such as SVM are based on image-to-image distances to create decision boundaries. For each image in the training set, a histogram of visual words is built, based on the features that appear in the image, and are matched with the visual words that resulted from quantizing all training features. In a nearest neighbor approach, image-to-image distances do not enable much generalization, as no inference is done from the training images and their features. This would mean only test images close to known images will be classified correctly. Therefore, image-to-class distances should be used. While an image might be far removed from all others of a certain class, the individual features might all be close to features of different images of this class, making

the image-to-image distances to a class large, but the image-to-class distance short. [37] Figure 6 shows the difference between image-to-image distance and image-to-class distance.

Figure 6: *Image-to-class distance versus image-to-image distance. Given three labeled images of the "ballet"-class (bottom row), and a query image (top left). Even though the "Query-to-image" distance is large to each individual labeled image, represented by the KL-divergences, the "Query-to-class" distance is small. The top right image shows the query image, with each descriptor having the color of the labeled image which gave it the highest descriptor likelihood. It shows that this query configuration is more likely given the three images, than each individual image separately. (Image taken from [5, 6].)*

Using the main points of no quantization and image-to-class distances, k-nearest neighbor can be performed for individual features, per class. Note that in this way, each individual feature could be classified as its NN, but a second step is required to classify images as a whole. Therefore the set of individual distances is used as the input of a Naive Bayes classifier.

The NBNN decision rule is defined under the Naive Bayes assumption of Section 3.2. The MAP classifier from (11), when assuming uniform priors $p(c)$ over all classes, simplifies to Maximum Likelihood (ML) classifier

$$\hat{c} = \arg\max_c \frac{1}{m} \prod_{i=1}^{m} p(d_{q,i}|c) \tag{12}$$

$$\propto \arg\max_c \frac{1}{n} \sum_{i=1}^{n} \log p(d_{q,i}|c), \tag{13}$$

Where the last formulation as a sum of log-probabilities is used in practice to compensate for the low probabilities typical for this classifier, which may cause rounding errors in computer applications.

Using the notion that NN classification tends to the Bayes optimal classifier when the sample size tends to infinity[5, 10], dense sampling can be used to get a high number of descriptors per class from the training set $Z = |\mathcal{D}_c|$. The

class conditional probability of a feature $p(d_q|c)$ can be modeled using Parzen likelihood estimation, where

$$\hat{p}(d_q|c) = \frac{1}{Z}\sum_{j=1}^{L} K(d_q - d_{c,j}). \tag{14}$$

Parzen kernel function $K(\cdot)$, typically Gaussian, defines the distance between query descriptor d_q and labeled descriptor d_c. When Z goes to infinity, $\hat{p}(f|c)$ approaches $p(f|c)$.

This approach entails calculating the distance of d to all Z descriptors of each class, which would be very costly. Because only a small minority of the descriptors can be expected to be significantly close to d_q, taking into account only the nearest descriptors is a safe approximation, which enables using nearest neighbors (NN) to find these descriptors. Even more so, Boiman shows that the performance loss of taking only the 1 nearest neighbor is very small. Because of this the Parzen estimate of d_q to class c reduces to the distance of d_q to its nearest neighbor in c: $\|d_q - \mathrm{NN}_c(d_q)\|^2$, resulting in the following log likelihood and classifier:

$$\log P(q|c) \propto -\sum_{i=1}^{m} \|d_{q,i} - \mathrm{NN}_c(d_{q,i})\|^2 \tag{15}$$

$$\hat{c} = \arg\min_c \sum_{i=1}^{m} \|d_{q,i} - \mathrm{NN}_c(d_{q,i})\|^2 \tag{16}$$

Now, classification boils down to calculating descriptors for all images in each class and for the query image, estimating the nearest neighbor of each query descriptor for each class, calculating the sum of distances for each class for the query image and selecting the lowest distance. This approach is both very simple and intuitive. It also enables use of different kinds and combinations of descriptors.

3.4 Limitations and Extensions

Even though the algorithm of Boiman *et al.* [5] is very simple and requires no learning phase, it does have scalability issues because of its dependence on as many individual descriptors as possible. Because all densely computed descriptors for each image in the training set have to be stored, and the nearest neighbor for each descriptor of each query image on each class has to be found, the time and memory usage is much higher than for example BoF methods, which use a more compact representation of images. Calculating NN can be sped up by using sophisticated approximations of the Nearest Neighbor algorithm, such as FLANN [24], which uses randomized kd-trees to provide a quick search in a k-dimensional search space. This relieves some time issue, but is not a remedy for the memory issues.

3.4.1 Local NBNN

An adaptation of NBNN by McCann & Lowe called local NBNN (LNBNN)[22] has another approach to solve some of the limitations of the original algorithm.

Their main idea is that it is not necessary to find a descriptor's nearest neighbor in each class, but that is sufficient to find the k nearest neighbors over all classes. It will perform better, in fact. See Figure 7 for a visualization of the conceptual difference.

The simplification of the basic query implies a more efficient algorithm, because it is much faster to search for k nearest neighbors in one collection, than to search for 1 NN for each class, even when k is much larger than the number of classes.

Besides the time-complexity benefit, LNBNN also enhances the results of NBNN. Because only the kNN are taken into account, it might well be that not all classes will be involved for each descriptor in an image. In the original NBNN, these distances would still be taken into account, lowering the chance of the class to get the highest probability for the image by a possibly large amount. LNBNN takes the $k+1$-th distance for these classes, making this effect much less significant. Liu *et al.*[20] have hypothesized that in a sparse manifold space, such as the descriptor space, larger distance give a much worse estimate of membership to a class. This might be the cause for the improved results LNBNN gets compared to Boiman's NBNN. Another argument is that each descriptor should at most add a bit evidence to the probability of an object's class. It should not lower this probability. Otherwise the influence of background clutter could be too large.

LNBNN will be used as a basis for the object detection algorithm, because it is an efficient variant of NBNN, and extends to the $k > 1$ case, which proved to be important in the detection case.

(a) *The original NBNN asks, "Does this descriptor look like a keyboard? a car? ... a dog"*

(b) *Local NBNN asks, "What does this descriptor look like?"*

Figure 7: *Difference in conceptual approach of regular NBNN and Local NBNN. From [22].*

4 Object Detection Using Exemplar Models

As mentioned before (Section 2.2), there is a broad range of object detection approaches. Because NBNN classification uses individual descriptors, it seems natural to look for a detection method that uses these descriptors directly, when you want to adapt NBNN to detection. Exemplar models do just that, and have been used in similar attempts before [1].

4.1 Exemplar Models

Exemplar models are related to part-based models.[18, 9] In this approach, for each object feature found during training, an exemplar is stored. The exemplar represents the size and location of the bounding box relative to the feature. The exemplars are aggregated into part models [18] or visual words [9]. In this way a class is modeled by a set of exemplars that have a high probability of occurring in a certain location and at a certain scale of the image.

At test time, each feature found in the test image is matched with the exemplars stored, and from this combination of the location and scale of the feature found, and the exemplar it is matched with, a hypothesis can be formed for the object's location in the test image. This means that each feature in the test image gets a vote, weighted by the quality of its match, for a bounding box. These hypotheses can be clustered into detection windows (See Figure 8). Experiments [33] show that this method performs well too as a first step in a cascade setting.

Figure 8: *Given four images of car sides, their bounding boxes (yellow) and 8 features of a car wheel (above), for each feature an exemplar is made, where its position and size relative to the bounding box is used. So when a feature is found during test time, that is similar to a car wheel, a number of hypotheses is formed by applying the exemplars to the new feature (below left, in black). These hypotheses can be clustered into a small amount of highly likely detections. (Image taken from Chum and Zisserman [9].)*

4.2 Clustering Algorithms

In exemplar-model detection, hypotheses of bounding boxes need to be clustered into detections. These detections represent the classes, which are inherently unknown, and depend fully on the image at hand, and the hypotheses that have been derived from it. The number of objects in the image is unknown, so the number of clusters should not be fixed.

K-means clustering is a well known clustering algorithm. It divides all objects in a pre-defined number of k clusters, minimizing the sum of squared distances of all objects within each separate cluster. An important reason why it is not appropriate for this task, is that we do not have an indication of the number of clusters we want. Therefore, other algorithms have to be considered.

4.2.1 Agglomerative Clustering

It is possible to view a clustering clustering algorithm as a tree, where the leaves represent the objects to be clustered. If the pairwise distances between

(a) **(b)** **(c)** *The set of hypotheses.*

(d) **(e)**

Figure 9: *The main difference between agglomerative clustering as used by Becker, and quickshift clustering is the way the detection is determined from the cluster. In single-link clustering, the inclusion of a single hypothesis into a cluster can have a large effect on the detection formed from this cluster: The blue hypothesis in (a) is included in the red cluster (b), making the red detection (bold line) almost twice as big. Using quickshift (d,e) this inclusion does not have any effect on the detection itself, because the local maximum of the red cluster is unchanged.*

all objects are known, the leaves can be joined into clusters iteratively, each time merging the two closest objects into a cluster. When these clusters are again clustered based on distance, merging will continue until there is only one single cluster left. Afterwards, this tree could be split into clusters by cutting branches that exceed a certain threshold distance, θ_m. This clustering algorithm is called Agglomerative clustering. It is a subtype of hierarchical clustering, other hierarchical algorithms not being agglomerative, but divisive, starting with one single cluster.

There is some variety in the ways to merge leaves into clusters. Single link clustering means that the minimum distance between members of each cluster is the basis of further clustering, complete linkage and mean linkage clustering take the maximum distance and the mean distance, respectively.

When clustering hypotheses, a substantial part of clustering consists of defining a detection from a cluster of hypotheses. In agglomerative this is not clearly defined, because, even though there is a hierarchy in clustering, it does not follow from the algorithm where the cluster center is located. Therefore, for this thesis, the values of cluster members are averaged to find a detection. [1] In a Euclidean clustering space, this is a reasonable approach, even there is no theoretical reason to do it this way.

4.2.2 Mode-seeking Algorithms

Mainly because of the arbitrary definition of a cluster center in agglomerative algorithms, it is useful to look for other clustering algorithms that have a better founded way of determining their center. Good candidates are mode finding algorithms, like mean-shift clustering and its variants. [8, 34]

Mode finding is the search for local density maxima in the clustering space. The mean-shift algorithm [8] Uses a Parzen density estimate to calculate this, and, starting at each data point, ascends the gradient of this estimate to find its maximum, shown in Figure 10. In this way, each data point's mode is found, and all data points can be clustered into these modes, the local maxima being the final detections.

Figure 10: *Mean shift clustering (left) vs. Median Shift clustering (center) vs. Quick-shift clustering (right) (Image taken from [34])*

Quickshift clustering takes a slightly different approach, mainly used to speed up the clustering process. [34] This approach does not model a global density estimate, but calculates the density for each data point. Next, for each point its neighbor with the highest density estimate (higher than its own) within a certain window is found. This results in a tree-like structure, where the data points at local density maxima become the head nodes, and define cluster and also its detection, the difference between mean-shift and quickshift is shown in Figure 10.

The window parameter τ in quickshift defines what range around each data point is used both to calculate its density estimate, and to look for data points available for clustering.

5 Exemplar-NBNN, an Exemplar-based Detection Model

NBNN classification and Chum's exemplar model detection [9] can be combined into a new detection method, because the exemplar model has more in common with the NBNN approach than most other detection methods. Exemplar models are based on image-to-class distance, like NBNN. This provides a natural way of linking both methods.

Chum *et al.*[9] uses interest point detectors and a codebook approach for their exemplars. The descriptor space is quantized like regular codebook classification methods do, even though they take the location of the object relative to the descriptor as basis for clustering the visual words, and not the descriptors themselves.

In contrast to codebook methods however, in Exemplar-NBNN the exemplar models use image-to-class distance when testing. Like explained in Section 3.3, while most codebook methods compare images to each other, in exemplar model detection the individual features are compared to the quantized exemplars. Because these exemplars have a designated class the distance of a query image

to each class forms the basis for classification, and not the distance between images.

In this way, the un-quantized descriptors of NBNN can be used as exemplars. In the detection phase, using the NBNN approach, each descriptor in a query image will get a NN from a class, but also an exemplar associated with this neighbor descriptor. When the query descriptor and its NN-exemplar are combined, a hypothesis can be formed for the location of an object, in the form of a bounding box.

These hypotheses could be used as detections, using the distance of the query descriptor to the nearest class as weight. This does not totally agree however with the requirement within the NBNN approach that the classification (detection in this case) should be based on as many descriptors as possible. In this case a single detection is based on a single descriptor only. Therefore, the hypotheses should be aggregated into clusters first. Detections based on these clusters have a much larger evidence base as the cluster size gets larger. Therefore, the cluster size can be regarded as a good weight for the resulting detections.

In their implementation, Becker *et al.* [1] rank their detections in this way, breaking ties for detections of the same cluster size by comparing the mean of some relative distance measure Q_H over all hypotheses that are member of the cluster. Q_H is defined as follows: $Q_H = \frac{d^- - d^+}{d^+}$, where d^+ and d^- represent the NN distance to the foreground and background class, respectively. In a multiple class setting, the background distance is the minimum over all non-class NN distances.

5.1 Descriptor Aliasing

An issue that arises using exemplar-NBNN detection is that certain descriptors may be very similar to not one, but multiple neighbors. These neighbors might refer to totally different exemplars, generating a variety of hypotheses. This effect can be called descriptor aliasing, and is shown in Figure 11.

This problem can be avoided by taking more than one neighbor into account, which translates to setting $k > 1$ in the kNN phase of exemplar-NBNN. This would mean that all $n \leq k$ neighbors that are closer to the foreground than the nearest background descriptor will be used to make detection hypotheses. This is not only an argument for introducing LNBNN detection, but also in general, for taking into account more neighbors in regular NBNN detection. This phenomenon explains some of the results obtained in the experiments.

6 Experiments

The theory on local exemplar-NBNN detection was tested in a number of experiments. First, the NBNN detection experiments of Becker [1] were replicated to validate the implementation, see Section 6.4. These experiments were altered in a number of ways, to see the effects of different descriptors, feature selection rules, clustering algorithms and using Behmo's optimized NBNN approach. Next to this, exemplar-kNBNN was tested, both with and without McCann's Local NBNN approach (Section 6.5). Most tests were performed on both the TUD motorbikes and VOC2007 benchmark detection sets.

(a) *Training image* **(b)** *Test image with hypotheses...* **(c)** *...and with resulting detections (Single-link)*

Figure 11: *During training (a), each descriptor gets an exemplar from ground truth, in this case the descriptors for the car's wheels. During the test phase (b), each descriptor found (car wheels again), gets hypotheses based on its neighbors for each class. Here, the wheel's two neighbors are those of (a), and their hypotheses are given in blue and red respectively (ground truth is given in yellow). When only one neighbor is taken into account, there is a risk of not favoring the two more or less correct bounding boxes because of repetition of descriptors in the training images. With $k = 2$ and using single-link clustering, the resulting detections would look something like (c), with the green detection getting a higher rank because of its larger cluster size.*

6.1 Features

In most experiments, feature selection is done by dense sampling. Each point was sampled at various scales, to allow for scale invariance. The feature that has been used was the scale-invariant feature transform (SIFT). [21]

For some experiments, SIFT features were detected using Harris-Laplacian interest point detection. [23, 29] This detector finds potential interest points based on Harris corner detection, and then selects the most appropriate scale for each by taking finding a maximum over a Laplacian-of-Gaussians. The SIFT features are taken at these locations and at these scales. Compared to dense sampling, interest point detection gets relatively little features, but rather distinctive ones. Boiman claims dense sampling is necessary for NBNN (*cf.* Section 3 [5]), but its performance/efficiency tradeoff for exemplar-NBNN might be different.

6.2 PASCAL VOC Data Set

The Visual Object Classes (VOC) 2007 challenge of the PASCAL network [12] provides a data set annotated for image detection. The data set consists of 20 object classes plus a background class, on a total of 9,963 images containing 24,640 annotated objects. The images are all taken from Flicker, and show a large variety of image quality and a large intra-class variety of objects. The data set is subdivided into 50% test set, 25% training set and 25% validation set, with approximately equal distribution of class object distribution over the sets.

The VOC 2007 dataset is partly annotated for class segmentation, 632 images subdivided into 210 test images, 213 validation images and 209 train images, approximately equally distributed over the classes. For the experiments below, the combined segmentation training and validation sets (422 images) were used to train, and the test set was mostly used to measure performance. The main reason for this training set is that segmentation proved to be a good ground

truth for class assignment of features. On top of that, the full VOC07 training set appeared prone to exceed time and memory constraints. Testing was done both on the segmentation test set and the full test set.

For performance comparison the results of the Pascal VOC 2007 detection challenge were used. For each class, the score of the best performing contender was taken and this provided the baseline scores. These scores give a good measure of the state-of-the-art at the time. While being outdated, it gives good comparison material for the new results.

6.3 TUD Motorbikes Data Set

The TUD Motorbikes data set is a selection of motorbikes from different benchmark sets. [15] The training set consists of 153 images of motorbikes on a uniform background, and are segmented into 2 classes: *motorbike* and *background*. The test set consists of 115 images, all of motorbikes, with various background, and ground truth bounding boxes. Becker *et al.* [1] use this benchmark to test their setup. In the training phase they train on the motorbike features from the TUD train set, and they add features from non-object areas of 300 random images of the PASCAL VOC 2007 set, as background features.

This same setup was used here, as a comparison to Becker's experiments. Even though this was not reported by Becker, I decided to perform in 5-fold, each time with a different randomized set of background images. Results of the experiments ware compared to the performance of Becker, who got an average precision of 83.20%, and an equal error rate of 84.1%.

6.4 Exemplar-NBNN Detection

The first experiment is used to verify the results of Becker *et al.* [1]. In addition to this, some parameters of this approach were altered to see what effects they have on the overall performance.

This experiment was performed on the TUD Motorbike set in the setup used by Becker (c.f. Section 6.3). The SIFT descriptors were collected using dense sampling, using a step size of 8 pixels, and patch sizes of 32×32, 48×48 and 64×64. I repeated each test 5 times with a different random selection of background images from the VOC 2007 data set, to avoid possible artifacts. FLANN was used to perform approximate NN, using the kd-tree algorithm with 4 trees and 1000 checks to assure high enough accuracy.

The similarity measure between hypotheses, used as a basis for the clustering algorithm, was defined as the area of overlap (AO) between two hypotheses:

$$AO(H_a, H_b) = \frac{|H_a \cap H_b|}{|H_a \cup H_b|} \tag{17}$$

Clustering was done using the single-link agglomerative clustering algorithm (Section 4.2.1), using 2 parameters: θ_m, which defines the minimal overlap between hypotheses to be clustered together, and the maximal overlap between detections: θ_p. I set the parameters such that $\theta_m = 0.8$ and $\theta_p = 0$, conform to Becker.

Detections were ranked according to their "detection quality", Q_D. This measure is defined as the amount of hypotheses being clustered into one detection. Ties being broken by a second ordering, called "hypothesis quality", Q_H,

which is defined as the distance of a hypothesis to the foreground class, relative to its distance to the background class:

$$Q_H = \frac{d_{bg} - d_{fg}}{d_{fg}} \tag{18}$$

Performance was measured with the average precision (AP) of the ranked detections, and visualized in a precision-recall curve.

6.4.1 NBNN Detection Using Quickshift clustering

I compared the results of detection using single link clustering with quickshift clustering. Because quickshift is a mode finding algorithm, it is guaranteed to find local maxima in the hypothesis density space. Therefore, it gives a more theoretically sound basis for clustering hypotheses, perhaps improving results along the way. Quickshift has one parameter, τ. This represents the expected variance within the clustering space, and defines the maximal variance to be clustered, somewhat like θ_m in single link clustering.

Quickshift has no bound for overlap in detections, which should not be a problem as the test images might have multiple overlapping objects. To make this comparison fair towards single link clustering, I varied the value of θ_p in the latter algorithm. For quickshift I settled on $\tau = 1.2$, for single link clustering $\theta_p = 0.4$ seemed ideal.

6.4.2 Exemplar-NBNN Results

Figure 12 and Table 1 show the results on the TUD set, of the implementation of Becker's algorithm using Single-link clustering compared to the results reported by Becker [1]. It also shows the results using the quickshift algorithm and using single-link clustering with $\theta_p = 0.4$, instead of Becker's $\theta_p = 0.0$. It shows that quickshift performs slightly better than single-link clustering on the TUD test, and that dense sampling is indeed crucial, seeing the major decline in performance when harris-laplacian feature selection was used.

Figure 12: *Precision-Recall curves of the TUD Motorbikes detection task. Comparing Becker's [1] setup with various settings, keeping $k = 1$. Curves provided are all based on a test using the same random background set (except the reference Equal Error Rate reported by Becker).*

Table 1: *Comparison of test statistics of TUD Motorbikes tests using various clustering algorithms and settings. "Becker Baseline" represents the result reported by Becker [1]. All other results are averaged over 5 randomized tests.*

Method	duration	max. recall	detections	ap
Becker Baseline	—	0.8450	—	0.8320
dense, k=1, $SL_{0.0}$	0.34	0.7632	390	0.7259
dense, k=1, $SL_{0.4}$	0.39	0.9936	3412	0.8314
dense, k=1, $QS_{1.2}$	0.46	0.9920	7874	0.8438
harrislaplace, k=1, $QS_{1.2}$	0.39	0.9440	6062	0.3421
dense, k=20, $SL_{0.4}$	2.30	0.9856	3673	0.8478
dense, k=4, $QS_{1.2}$	1.09	0.9776	8189	0.8788
harrislaplace, k=4, $QS_{1.2}$	0.43	0.9440	6768	0.4357
dense, k=20, local, $SL_{0.4}$	87.27	0.8080	6997	0.5236
harrislaplace, k=4, local, $QS_{1.2}$	0.69	0.7760	13052	0.2124
dense, k=4, local, $QS_{1.2}$	7.59	0.9360	12467	0.7948

Table 1 shows some more statistics of the experiments. The results reported by Becker are somewhat higher than my own basic replication. This might be caused by a fortunate set of random background descriptors in the reference test, while my results are averaged over 5 random sets. Becker does not report the randomization process involved, so I was not able to replicate the results exactly. The results clearly show that raising the θ_P threshold for the maximal overlap between detections is beneficial, mainly because the recall gets higher. It also shows this change raises the total number of detections found by a factor of > 10, quickshift generating more detections than single-link using similar settings. Interestingly, interest point detection seems to generate almost all correct detections with maximal recall of 94%. The low AP can only be explained by a bad ranking of detections.

Figure 13: *AP for each VOC07 class of various tests using $k = 1$, sorted by the median of AP over all classes.*

On the VOC07 dataset, overall performance is lower than on the TUD set, which was expected. Figure 13 and Table 2 show that the overall performance of exemplar NBNN (with $k = 1$) is lower than that of the baseline. The results

however vary greatly over the classes. In 8 classes, the experiment using Becker's settings ("dense, k=1,$SL_{0.0}$, segtest") gets a higher AP than the baseline. At the same time, some classes get APs which are much lower than the baseline, notably *bottle*, *car*, *chair*, *person*, *pottedplant*, *sofa* and *tvmonitor*. Looking at the median of the APs over all classes, taken as a measure to compare overall performance, it shows that on average the $k = 1$ methods perform somewhat less than the baseline. It also confirms the results on the TUD experiments, quickshift seems to perform better than single-link clustering, in single-link clustering overlap between detections should be allowed, and dense sampling gives better performance than interest point detection.

Looking at some other statistics of these VOC experiments in Table 3 shows that it is true that interest point detection results in large time savings. Again, the increase of the amount of detections is quite steep as overlap between detections is allowed.

6.5 Exemplar-kNBNN Detection

A downside of the exemplar-NBNN method is the descriptor aliasing problem explained in Section 5.1. This can be solved by taking the k of kNN to be greater than 1. This means that the k closest neighbors over all classes (the object class and the background class) are taken into account when constructing detection hypotheses. Within these k nearest neighbors, the ones belonging to the object class that are closer than any neighbor in the background class, are transformed into detection hypotheses. This kNBNN approach was tested in the detection algorithm and compared to the previous setup.

6.5.1 Local NBNN Detection

Another disadvantage of exemplar-NBNN detection is it returning many false positive detections for images not having any objects of the current object class. The reason for this is obvious, as all descriptors closer to the class than to the background will be regarded as hypotheses, and therefore images without any objects get reasonably well ranked detections. This problem gets larger as the data sets contains more object classes.

A solution for this is to not only compare the current object class with the background class, but to compare it with all other classes, to see what each descriptor in the test image looks most like. This results in a hypothesis selection step that chooses only descriptors that are closer to the current class than to any other.

This however would create another problem not present in the original approach. Some descriptors could be an indication for multiple classes, because certain parts of objects are quite similar. While the original exemplar-NBNN approach allows for this, regarding each class independently, this multi-class approach prevents this. Think of the similarity of a bicycle wheel with that of a motorbike or a car. To disregard all but one of these classes might cause very little evidence for most classes in an image, giving less stable detections. This reminds of Boiman's argument that much evidence is vital for the NBNN method, because the Naive Bayes assumption is only met towards infinity and there is no training phase to compensate for it. [5]

At the same time, the goal to compare all classes simultaneously is similar to McCann's formulation of local NBNN's query. [22] (*cf.* Section 3.4.1). Merging all class indexes into one, and taking into account not only the first NN per class, but the *overall* kNN prevents having too many false positive results, while keeping the possibility of a descriptor to be used for a hypothesis for multiple classes like in the original problem.

As a bonus, LNBNN detection also incorporates a way to prevent the aliasing problem (*cf.* Section 5.1). Among the k closest neighbors, there might be multiple ones from the same class. Instead of disregarding these neighbors like local NBNN does, they can be taken into account by creating hypotheses for all non-background nearest neighbors.

6.5.2 Exemplar kNBNN & LNBNN Results

Figure 14: *Precision-Recall curves of the TUD Motorbikes detection task, of various tests with $k > 1$.*

As shown in Figure 14 and Table 1, kNBNN detection on the TUD task improves the performance even further. The optimal amount of neighbors appeared to be at $k = 4$ for quickshift clustering, while the optimum for single-link runs was at $k = 20$. Interestingly, LNBNN seems to perform less well than regular kNBNN. This might be due to the task, having only one class and all test images containing at least one motorbike, while LNBNN was built specifically for multiple-class detection and the probability of a class appearing on an image being less than one.

Again, interest point detection harms performances greatly compared to dense sampling, and quickshift always performs better than single-link clustering. Furthermore, the amount of detections seems to get quite big in quickshift-LNBNN experiments. The "dense, k=20,local,$SL_{0.4}$" experiment took 87 hours, which is extraordinary. It is probably due to a memory issue.

On the VOC test, performances are somewhat comparable to the $k = 1$ experiments as Figure 15 and Table 2 show. The methods can keep up fairly well with the baseline, but again there is much variance between classes and experiments, both in absolute performance and relative to the baseline.

Highest average performance was reached in the "dense, local, k=4, $QS_{1.2}$" experiment, the median AP being 21.9%, very slightly above the baseline. The same pattern as before emerges in dense sampling getting higher results than

Figure 15: *AP for each VOC07 class of various tests using $k > 1$, compared with both the baseline and the best results with $k = 1$.*

harris-laplacian interest point detection. The results interestingly show that while the median AP's of kNBNN and LNBNN detection are higher than single NN detection, the differences between classes are also bigger in the new experiments. Taken over all experiments, 4 classes get highest AP using kNBNN, 6 classes using LNBNN and only 2 classes using single NBNN (the other 8 classes getting highest AP in the baseline). On the other hand, the second or third largest AP for most classes are among the single NBNN experiments. Experiment "harrislaplace, local, k=20, $SL_{0.4}$, segtest" is interesting, because even though it performs mediocre on average, it scores best in 4 classes: *aeroplane*, *bird*, *horse* and *train*. Further analysis is needed to find out what causes these results.

Table 3 shows that duration of VOC tests can get much higher for kNBNN and LNBNN experiments, just like the number of detections. Again, one test clearly stands out: "dense, local, k=4, $QS_{1.2}$, segtest". Just like in the TUD test, this is probably caused by memory deficiency issues.

Take in mind that I trained on a subset of the available training images only, and I tested on a subset of the test set too. It would be interesting to see results on a larger train and test set, even though much of the train set was not segmented and therefore not available for comparison.

The two last experiments mentioned in Table 3, Table 2 and the last one in Figure 15: "harrislaplace, k=4, $QS_{1.2}$, full test" and "dense, k=1, $SL_{0.0}$, full test", were performed on the full VOC07 detection test set. Performance is much lower than using the smaller segmentation test set, duration and number of detections are much higher than their counterpart experiments on the smaller set. This shows some complexity issues of the method. The next section will elaborate on the possible consequences of these results.

7 Analysis of Results

The experiments and their results show a number of things: the approach of exemplar-NBNN detection works not only on the TUD dataset, but also fairly well on the VOC2007 set. Furthermore, the substitution of of the agglomerative clustering algorithm with a mode finding algorithm seems to have a positive

Table 2: *Comparison of Average Precision on the VOC 2007 test for local NBNN, and a baseline of the best performances achieved in the VOC 2007 challenge. Per class the highest three AP's are colored gold, silver, and bronze respectively.*

	Pascal VOC07 Challenge Baseline	dense, k=1, $SL_{0.0}$, segtest	dense, k=1, $SL_{0.4}$, segtest	dense, k=1, $QS_{1.2}$, segtest	harrislaplace, k=1, $QS_{1.2}$, segtest	dense, k=4, $QS_{1.2}$, segtest	harrislaplace, k=4, $QS_{1.2}$, segtest	dense, local, k=4, $QS_{1.2}$, segtest	harrislaplace, local, k=4, $QS_{1.2}$, segtest	harrislaplace, local, k=20, $SL_{0.4}$, segtest	harrislaplace, k=4, $QS_{1.2}$, full test	dense, k=1, $SL_{0.0}$, full test
aeroplane	26.2	11.2	13.3	29.4	31.8	21.3	37.0	29.8	38.8	44.0	17.5	13.7
bicycle	40.9	21.8	35.7	37.2	18.1	37.0	12.4	39.6	20.3	22.5	11.8	17.3
bird	9.8	32.2	33.3	24.0	13.6	25.1	14.0	27.0	22.7	33.7	5.4	7.0
boat	9.4	10.0	14.6	12.6	1.0	15.0	1.6	12.6	3.2	1.7	6.2	1.9
bottle	21.4	6.1	8.1	5.2	1.0	4.5	0.8	4.1	4.4	2.5	0.6	1.0
bus	39.3	24.4	33.1	31.1	17.6	18.0	17.6	28.8	14.6	20.1	13.2	23.3
car	43.2	14.5	17.7	14.5	10.9	12.9	11.6	17.0	13.6	6.1	17.6	15.1
cat	24.0	47.1	46.9	32.2	33.3	32.2	33.3	27.9	27.0	43.0	34.2	37.5
chair	12.8	0.0	2.0	1.7	0.7	1.4	1.8	1.2	2.0	1.3	1.6	0.9
cow	14.0	14.7	16.9	21.7	9.6	22.5	10.9	20.4	11.8	16.9	8.0	7.2
diningtable	9.8	27.1	25.8	19.5	4.1	32.5	0.7	23.5	9.1	18.4	25.3	20.9
dog	16.2	26.2	24.3	17.6	5.3	18.5	3.9	16.8	12.6	9.6	22.9	23.2
horse	33.5	21.0	21.4	29.4	22.2	28.7	21.2	31.7	13.9	34.7	14.6	14.9
motorbike	37.5	41.5	49.3	48.4	44.1	54.7	43.4	42.8	42.6	28.1	18.1	25.2
person	22.1	9.4	11.5	7.7	4.0	7.8	4.4	6.4	3.5	6.2	5.3	7.3
pottedplant	12.0	0.3	1.8	2.7	0.4	2.7	0.2	4.1	0.7	0.9	1.6	0.5
sheep	17.5	12.4	15.4	17.2	16.2	19.6	15.5	20.3	17.4	16.5	6.3	4.7
sofa	14.7	0.0	3.3	5.7	0.2	10.0	0.2	26.2	1.1	6.8	10.0	4.4
train	33.4	52.3	48.0	48.4	48.7	57.6	48.5	51.9	40.5	58.7	19.9	20.3
tvmonitor	29.8	7.5	16.8	9.2	19.9	5.8	19.8	8.4	11.8	12.3	6.1	2.5
MEDIAN	21.8	14.6	17.3	18.6	12.2	19.0	12.0	21.9	13.1	16.7	10.9	10.5

effect, just like taking a value of $k > 1$ and using LNBNN. Nevertheless there is much variance within the results, and therefore it is necessary to give a deeper analysis of the results, to find out how they can be interpreted.

A first glance at the results might show which objects are relatively easy to detect, which ones are hard, and what images succeed in "deceiving" the LNBNN. Figure 16 shows the four highest ranked "True Positive" detections of persons in the experiment with the best median AP for VOC07 (cf. Table 2). Strikingly, all objects are large, covering over half the image. Compared to the highest ranked "False Positives" in Figure 17, it shows that all highly ranked detections cover all or most of the image. Figure 19 confirms this suspicion that highly ranked detections tend to be larger on average than the lower ranked ones. Taking into account the ranking method, which gives larger clusters a higher rank (Q_D), this means that the larger the cluster, the more likely it is that the detections are large in image coverage.

Taking a closer look at Figure 17, that besides a near miss of a person

Table 3: *Test duration and total number of detections found per VOC07-test. This gives an indication of time and memory complexity.*

	duration	no. of detections
dense, k=1, $SL_{0.0}$, segtest	3.27	10016
dense, k=1, $SL_{0.4}$, segtest	3.39	86158
dense, k=1, $QS_{1.2}$, segtest	4.30	165356
harrislaplace, k=1, $QS_{1.2}$, segtest	0.95	54906
dense, k=4, $QS_{1.2}$, segtest	6.39	173901
harrislaplace, k=4, $QS_{1.2}$, segtest	1.00	57774
dense, local, k=4, $QS_{1.2}$, segtest	80.05	379439
harrislaplace, local, k=4, $QS_{1.2}$, segtest	2.23	159427
harrislaplace, local, k=20, $SL_{0.4}$, segtest	4.36	299750
harrislaplace, k=4, $QS_{1.2}$, full test	43.49	3630083
dense, k=1, $SL_{0.0}$, full test	72.08	231669

detection, the other false positives involve large objects, but of the wrong class.

On the other end of the detection ranking are hard-to-guess persons, that have nevertheless been detected. Note that because the recall is lower than 1, some persons have not been detected at all. The hard-to-guess persons shown in Figure 18 are mostly small, occluded, or cut-off. besides, in some images large objects of other classes are pictured, which may have influenced the number of hypotheses for the person-class in these particular images.

In the next sections, certain aspects of the experiments are analyzed in more depth.

7.1 The Datasets

The differences between the datasets are striking. The TUD Motorbikes set has a single class, and the test set contains only positive images (i.e. images with at least one motorbike). The training set has a clear-cut segmentation, which improves the feature quality. The VOC07 dataset is much harder. Every class appears in only 5% of the test images, excluding the *person* class, which occurs in 40% of the images. To put it another way, in the TUD Motorbike test the task is for every image to locate the motorbike object(s), while in de VOC2007 test the task is for every image and every class to assess the possibility of this class having an object in the image, and if yes, to locate it.

Of course, in practice these tasks are modeled in the same way. The detections within a single class are all compared to one another in terms of their value (cluster size), across all images, both class and non-class images (*Cf.* the PASCAL VOC evaluation method [12].)

Another important aspect is that the TUD dataset tends to have motorbike objects that are large, centered in the image, and with little clipping and occlusion. Some of the classes of VOC2007 consist of objects that are generally small (*bottle*), not in the focal point of the image (*chair, pottedplant*), or often occluded by other objects (*diningtable*). This makes it obvious that the scores overall are higher on the TUD Motorbikes test than on the VOC2007 test.

(a) **(b)** **(c)** **(d)**

Figure 16: *"True Positives", correct detections with the highest rank, using LNBNN, $k = 4$, interest point detection and quickshift ($\tau = 1.2$) on VOC07's person class.*

7.2 Clustering Algorithms

Overall, the suspicion seems to be correct that quickshift should perform better than agglomerative clustering because of the more natural way of selecting a cluster center, i.e. a detection out of a cluster of hypotheses. The results however depend a lot on class, and are not always very obvious.

To see whether the clustering of hypotheses is needed at all, for some tests a comparison was made between the performance of the ranked detections versus ranking the hpyotheses. As a ranking method, Becker's Q_H measure was used (Equation 18). Figure 20 shows the effects of clustering on a small test.

The recall of the detections will usually be lower than the recall when using hypotheses, because the detections are more or less a subset of hypotheses. However, ranking the hypotheses appears to be much harder than ranking detections, resulting in a much lower AP. This shows the main benefit of clustering.

7.3 The Influence of k

The experiments show that performance increases when multiple neighbors are taken into account, confirming the theory of descriptor aliasing in Section 5.1. There is however a quite narrow sweet spot: setting k too high hurts performance and, to a larger extend, efficiency. On the TUD set, with only two classes (motorbike and background), the ideal value for k will be lower than on the VOC07 set, having 21 classes. As McCann [22] shows, k should be a multiple of

Figure 17: *"False Positives", incorrect detections with the highest rank, using LNBNN, $k = 4$, interest point detection and quickshift ($\tau = 1.2$) on VOC07's person class.*

the number of classes to give a good estimate of the feature's local surroundings, and of the classes which are relatively close to it. With k as low as 5, at most 5 classes can be included as nearest neighbors, which might not give a fair estimate for the other classes.

Unfortunately, setting k higher has been proven prohibitive when running quickshift tests on the VOC07 set. A more efficient algorithm could explore the influence of k much better.

7.4 Time and Memory

Detection algorithms can be compared not only for their qualitative performance, but also in terms of complexity and efficiency. Exemplar-NBNN is quite heavy in both terms. Given t training images, s test images, d descriptors per image, on average, c classes, and k nearest neighbors, this is the time complexity breakdown of the main stages of the pipeline:

1. Extracting features from the training images, splitting them into the classes, adding them to their respective NN indexes and saving their exemplars: $\frac{td}{c} \log \frac{td}{c} = O(n \log n)$, using FLANN kd-trees for building NN indexes.

2. Extracting features from the test images: $s \times d$

Figure 18: *"Difficult", correct detections with the lowest ranks, using LNBNN, $k = 4$, interest point detection and quickshift ($\tau = 1.2$) on VOC07's person class.*

3. Finding k nearest neighbors from each class, for each descriptor of each test image: $dsc \log \frac{td}{c} = O(n \log n)$, using kd-trees.

4. Performing detection. For each test image and each class, get its hypotheses, calculated their pairwise overlap, and cluster them: $scx(dk)^2 = O(n^2)$, factor $x > 1$ depending on the clustering algorithm, but at least once for the pairwise overlap.

This shows that the training phase has complexity $O(n \log n)$, while the test phase has complexity $O(n^2)$. This means training is quite efficient, mainly because no learning is involved. Testing takes much more time however, and proves to be the prohibitive step, especially calculating pairwise distances between all hypotheses in an image, for every image. Figure 21 and 22 show the duration of the experiments discussed in the previous section. The combination of LNBNN and dense sampling seems to increase duration greatly. Dense sampling results in a much larger value d per image, causing both training and testing to take more time, especially the factor $(dk)^2$ during the detection phase, because $d >> k$. LNBNN also takes into account more NN than plain kNBNN, but should not have this great an effect.

Increasing time complexity is somewhat correlated to memory complexity, especially in the biggest factor involved, the number of descriptors per image d. Both the FLANN indexes and the test image representations get bigger. This effect may cause additional issues when the experiment runs out of memory (memory swapping). This may be the cause of the extremely large durations

Figure 19: *The experiments show that the detections with a higher rank (from a larger cluster), also have a tendency to generate larger bounding boxes. This graph comes from the TUD motorbike experiment using $k = 4$, quickshift ($\tau = 1.2$) and dense sampling.*

Figure 20: *Comparison of Average Precision using the clustered detections versus the hypotheses before clustering, for a small test on 5 VOC07 classes, using LNBNN with $k = 5$, quickshift clustering, and interest point detection. It clearly shows the effect of clustering.*

reported in Figures 21 and 22.

8 Conclusion

Since the introduction of NBNN in 2008, much efforts are made improving the method for image classification, and to a lesser extend, object detection. Other approaches have been developing too, while NBNN still has the attention of a number of researchers, it has not (yet?) become a major approach in object recognition. This thesis explores the possibilities of NBNN into the domain of object detection, and confirms the appealing traits of the method also stated by others: training is quick, the pipeline is relatively straightforward and intuitive, with a number of theoretical benefits.

This thesis extends NBNN while keeping these traits intact. In the training phase, the only extension is keeping track of exemplars, which can be calculated easily. The pipeline is still intuitive, forming bounding-box hypotheses from NN features, clustering the hypotheses into detections, and ranking these. Introduc-

Figure 21: *Obviously duration increases as the experiments get more complex. LNBNN takes much more time, while the use of interest points causes a significant time reduction on tests with $k > 1$.*

ing quickshift, a sound way of defining detections from clusters of hypotheses is devised. The theoretical basis of Local NBNN, in asking the question "What do the features look like?" instead of iteratively asking "Does this feature look like class A? Like class B? ..." is also in line with the simplicity of the NBNN approach, and at the same time it is theoretically more sound to look at the local neighborhood of features, instead of a per-class basis.

The resulting method improves over earlier NBNN-like methods for detection, and is capable of achieving good results on the popular benchmark image set of Pascal VOC 2007.

Having said this, the main disadvantage of this method, and earlier NBNN-based methods, is the slow test phase. Because no model is being learned, test images have to be compared with more or less raw training features, which is both complex in time and memory. Optimizations like FLANN relieve this complexity a bit, but the test phase remains a bottleneck. In detection, this disadvantage becomes even larger than in classification, because of the more extensive test phase. More research is needed to discover the practical boundaries of LNBNN detection and see what efficiency and performance improvements can be made.

The focus on further research should be on finding efficient ways of performing object detection using NBNN. There might be ways of improving the clustering algorithm without hurting performance. For example, it would improve efficiency greatly if not every hypothesis would have to be compared to every single other one, by exploiting certain structural properties of bounding boxes.

It could also be interesting to try and combine the advantages of the NBNN approach with those of other detection methods, such as part-based models, efficient sub-window search, or hierarchical approaches. The object-to-class nature of NBNN might be a useful addition to these methods, that are often based

Figure 22: *The same trend is visible when testing on the VOC07 set. Especially the combination of dense sampling and Local NBNN is prohibitive. The three rightmost columns show tests run on the full VOC test set ($\sim$ 5000 images), whereas the other experiments were run on the segmentation test set of 210 images*

on object-to-object distance measures. On the other hand, LNBNN detection might benefit from a more discriminative approach, such as support vector machines or conditional random fields.

References

[1] J.H. Becker, T. Tuytelaars, and L. Van Gool. Codebook-free exemplar models for object detection. In *13th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS), 2012*, pages 1–4. IEEE, 2012.

[2] R. Behmo, P. Marcombes, A. Dalalyan, and V. Prinet. Towards optimal naive bayes nearest neighbor. In *ECCV 2010*, pages 171–184. Springer, 2010.

[3] B. Benfold and I. Reid. Stable multi-target tracking in real-time surveillance video. In *CVPR, 2011*, pages 3457–3464. IEEE, 2011.

[4] A.C. Berg, T.L. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondences. In *CVPR, 2005*, volume 1, pages 26–33. IEEE, 2005.

[5] O. Boiman, E. Shechtman, and M. Irani. In defense of nearest-neighbor based image classification. In *CVPR 2008*, pages 1–8. IEEE, 2008.

[6] O. Boiman M., Irani. Similarity by composition. In *NIPS 2006*, volume 19, page 177. The MIT Press, 2006.

[7] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. In *Proc. BMVC*, pages 76.1–76.12, 2011. ISBN 1-901725-43-X. http://dx.doi.org/10.5244/C.25.76.

[8] Y. Cheng. Mean shift, mode seeking, and clustering. *PAMI*, 17(8):790–799, 1995.

[9] O. Chum and A. Zisserman. An exemplar model for learning object classes. In *CVPR 2007*, pages 1–8. IEEE, 2007.

[10] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[11] A. Ekin, A.M. Tekalp, and R. Mehrotra. Automatic soccer video analysis and summarization. *IEEE Transactions on Image Processing*, 12(7):796–807, 2003.

[12] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[13] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, 106(1): 59–70, 2007.

[14] P.F. Felzenszwalb, R.B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

[15] M. Fritz, B. Leibe, B. Caputo, and B. Schiele. Integrating representative and discriminant models for object category detection. In *ICCV 2005*, volume 2, pages 1363–1370. IEEE, 2005.

[16] C.H. Lampert, M.B. Blaschko, and T. Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *CVPR 2008*, pages 1–8. IEEE, 2008.

[17] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR 2006*, volume 2, pages 2169–2178. IEEE, 2006.

[18] B. Leibe, A. Leonardis, and B. Schiele. Combined object categorization and segmentation with an implicit shape model. In *Workshop on Statistical Learning in Computer Vision, ECCV*, pages 17–32, 2004.

[19] A.J. Lipton, H. Fujiyoshi, and R.S. Patil. Moving target classification and tracking from real-time video. In *Workshop on Applications of Computer Vision, 1998*, pages 8–14. IEEE, 1998.

[20] L. Liu, L. Wang, and X. Liu. In defense of soft-assignment coding. In *ICCV 2011*, pages 2486–2493. IEEE, 2011.

[21] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[22] S. McCann and D.G. Lowe. Local naive bayes nearest neighbor for image classification. In *CVPR 2012*, pages 3650–3656. IEEE, 2012.

[23] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *International journal of Computer Vision*, 65(1-2):43–72, 2005.

[24] M. Muja and D.G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Applications (VISSAPP09)*, pages 331–340, 2009.

[25] M. Pedersoli, A. Vedaldi, and J. Gonzalez. A coarse-to-fine approach for fast deformable object detection. In *CVPR 2011*, pages 1353–1360. IEEE, 2011.

[26] F. Perronnin, J. Sánchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. In *Computer Vision–ECCV 2010*, pages 143–156. Springer, 2010.

[27] R. Timofte and L. Van Gool. Iterative nearest neighbors for classification and dimensionality reduction. In *CVPR 2012*. IEEE, 2012.

[28] T. Tuytelaars, M. Fritz, K. Saenko, and T. Darrell. The nbnn kernel. In *ICCV 2011*, pages 1824–1831. IEEE, 2011.

[29] K.E.A. van de Sande, T. Gevers, and C.G.M. Snoek. Evaluating color descriptors for object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1582–1596, 2010.

[30] K.E.A. van de Sande, J.R.R. Uijlings, T. Gevers, and A.W.M. Smeulders. Segmentation as selective search for object recognition. In *ICCV 2011*, pages 1879–1886. IEEE, 2011.

[31] J.C. Van Gemert. Exploiting photographic style for category-level image classification by generalizing the spatial pyramid. In *Proceedings of the 1st ACM International Conference on Multimedia Retrieval*, page 14. ACM, 2011.

[32] J.C. van Gemert, C.J. Veenman, A.W.M. Smeulders, and J.-M. Geusebroek. Visual word ambiguity. *PAMI 2010*, 32(7):1271–1283, 2010.

[33] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *ICCV 2009*, pages 606–613. IEEE, 2009.

[34] A. Vedaldi and S. Soatto. Quick shift and kernel methods for mode seeking. In *Computer Vision–ECCV 2008*, pages 705–718. Springer, 2008.

[35] P. Viola and M.J. Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004.

[36] J Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *CVPR 2010*, pages 3360–3367. IEEE, 2010.

[37] Z. Wang, Y. Hu, and L.T. Chia. Learning instance-to-class distance for human action recognition. In *ICIP 2009*, pages 3545–3548. IEEE, 2009.

[38] Z. Wang, Y. Hu, and L.T. Chia. Improved learning of i2c distance and accelerating the neighborhood search for image classification. *Pattern Recognition*, 2011.

[39] T. Yeh, J.J. Lee, and T. Darrell. Fast concurrent object localization and recognition. In *CVPR 2009*, pages 280–287. IEEE, 2009.

[40] D. Zhang, B. Liu, C. Sun, and X. Wang. Random sampling image to class distance for photo annotation. *Working Notes of CLEF*, 2010.

[41] H. Zhang, A.C. Berg, M. Maire, and J. Malik. Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In *CVPR, 2006*, volume 2, pages 2126–2136. IEEE, 2006.

[42] Z. Zhang, Y. Cao, D. Salvi, K. Oliver, J. Waggoner, and S. Wang. Free-shape subwindow search for object localization. In *CVPR 2010*, pages 1086–1093. IEEE, 2010.