Regular Paper
Journal of Computing Science and Engineering
Vol.00, No.0, Jan 0000, pp. 000-000

Memory-Efficient NBNN Image Classification

YoonSeok Lee
School of Computing, KAIST, Korea **ys.lee@kaist.ac.kr**

Sung-eui Yoon*
School of Computing, KAIST, Korea **sungeui@gmail.com**

Abstract
Naive Bays Nearest Neighbor (NBNN) is a simple image classifier based on identifying nearest neighbors. NBNN uses original image descriptors (e.g., SIFTs) without vector quantization for preserving the discriminative power of descriptors and has a powerful generalization characteristic. It, however, has a distinct disadvantage; its memory requirement can be prohibitively high as we have a large amount of data. We identify this problem of NBNN techniques and apply a binary code embedding technique, i.e., spherical hashing, to encode data compactly without a significant loss of classification accuracy. We also propose to use an inverted index to identify nearest neighbors among those binarized image descriptors. To demonstrate benefits of our method, we apply our method to two of existing NBNN techniques with an image dataset. By using 64 bit lengths, we are able to observe 16 times memory reduction with a higher runtime performance without a significant loss of classification accuracy. This result is achieved by our compact encoding scheme of image descriptors without losing much information of original image descriptors.

Category: Smart and Intelligent Computing

Keywords: Image classification; NBNN; hashing; memory efficiency; indexing

I. INTRODUCTION

Image classification is a task of assigning an appropriate class label to a query image, and has been studied as an important task in the computer vision field for a long time.

Among many available classification techniques, Naive Bayes Nearest Neighbor (NBNN) [1] is one of popular image classifiers that does not require an explicit learning process. NBNN is designed based on the naive Bayes assumption and using nearest neighbor search. It usually uses local descriptors (e.g., SIFTs), which is densely extracted from a query image. Unlike many other conventional image classifiers, NBNN does not perform descriptor quantization like bags-of-words for compact representation. Instead, NBNN utilizes original image descriptors as they maintain discriminative power. As the distance metric, NBNN uses "Image-to-class" distances measured with all the available classes by identifying the nearest neighbor for each local descriptor, and assigns a class that has the minimum sum of distances to the class type of a query image.

Thanks to characteristics of NBNN, the NBNN approach has advantages over other learning based techniques for image classification. NBNN is theoretically simple and easy to implement. As a result, it is also easy to modify NBNN for a particular purpose. For example, NBNN is adjusted for solving domain adaptation problems [2]. Furthermore, NBNN shows high generalization power [3], since it works mainly in a data-driven way without tuning parameters for a particular dataset.

Recently, Convolutional Neural Networks (CNNs) [4] are achieving high classification accuracy and thus receiving significant attention. Also, its follow-up studies are being actively conducted [5, 6]. Interestingly, there is a recent study [7] in the direction of combining CNNs and NBNN for achieving even higher accuracy. Furthermore, NBNN techniques can be used in situations where

Open Access yy.5626/JCSE.2011.5.2.xxx http://jcse.kiise.org

Received 00 Month 2011, Accepted 00 Month 2011, Revised 00 Month 2011
* Corresponding Author

using deep CNNs is not appropriate due to their long training time.

Nonetheless, NBNN has certain drawbacks such as low accuracy compared to recent convolutional neural net based approaches, slow runtime performance, and high memory requirement. Accuracy and slow performance have been addressed by many prior approaches [8, 9, 10, 3], but the memory issue has not been well addressed, according to the best of our knowledge.

Main contributions. In this paper, we propose a memory-efficient NBNN technique. To compactly represent image descriptors, we apply a binary code embedding technique to map original local image descriptors into short binary codes. We then perform approximate, yet fast nearest neighbor search by using an inverted index structure associated with those binary codes.

To verify benefits of our method, we test our method against a standard image dataset, and compare our method against two well-known NBNN approaches, the original NBNN and the local NBNN that improves the performance of the original NBNN. By using our method, we are able to observe faster running performance and lower memory requirement without a significant loss of classification accuracy. Especially, when we use 64 bit lengths for binary codes, we are able to achieve 16 times memory reduction over those two NBNN approaches, while 12 times and 1.125 times faster running performance over the original and local NBNNs, respectively. These results are achieved by accurately embedding original descriptors into compact binary codes. To encourage further research, source codes of our approach will be available[1].

II. RELATED WORK

In this section, we review prior approaches that are directly related to our method.

A. NBNN

NBNN[1] uses original image descriptors to preserve the discriminative power of the features instead of using descriptor quantization method like bag-of-words, which is used in many other image classifiers. In addition, NBNN utilizes the image-to-class distance metric in order to generalize the characteristics of each class in contrast to other methods using the classical image-to-image distance. Therefore, it can classify images successfully by searching similar local descriptors to the query descriptors among all the descriptors, even if there are no matching images to the query image in the dataset.

To address drawbacks of the original NBNN and extend it to other related problems, many studies have been proposed. Optimal NBNN [11] studied parameters to consider the assumptions that were made for designing NBNN, and dependencies among the local features are also studied [10]. Recently, NBNN was utilized for data adaptation problem [2] and image retrieval [7].

In order to address a high runtime overhead in querying, McCann et al. [9] proposed local NBNN, which only calculates the distance from the query descriptors to others in a single time, instead of performing the search iteratively with all the classes. However, the memory scalability problem arisen from using unquantized original descriptors is not well considered yet.

Nearest neighbor search. Exact or approximate nearest neighbor search has been widely studied. One of most common acceleration data structures for the search is kd-trees [12]. kd-trees were also widely adopted in many computer vision techniques and various optimization techniques with kd-trees have been proposed [13]. Some of well-known optimization techniques in the computer vision field include randomized kd-trees [14] and relaxed orthogonality of partition axes [15]. Muja and Lowe [16] have proposed an automatic parameter selection algorithm of some of the aforementioned techniques (e.g., [14]). Nonetheless, many hierarchical techniques including ones based on kd-trees have been known to work ineffectively for high dimensional problems.

B. Hashing

As an approximate, yet scalable nearest neighbor search approach, hashing techniques have been extensively studied recently. These techniques can be broken into two categories: data-independent and data-dependent techniques. Data-dependent techniques [17, 18] can produce more high accuracy for the search problem, by computing hashing functions considering input data. Unfortunately, most these techniques tend to rely upon learning techniques or to require high computation time and thus we focus on data-independent techniques, which are more suitable for NBNN approaches.

The most well-known technique under the data-independent category is locality sensitive hashing [19]. This technique draws hyperplanes randomly from a certain distribution function, and uses them for hashing functions. This technique has been generalized into many different directions including ones to support different distance metrics [20] and GPU acceleration [21].

These hashing functions can be used for encoding input data into binary codes. Recently, hypersphere based hashing function and binary code embedding technique is proposed [22]. This technique can

[1]http://sglab.kaist.ac.kr/projects/NBNN_Memory

generate more closed regions in high dimensional spaces, resulting in a high accuracy for approximate neighbor search. This property can preserve the distances between the original data well with their corresponding binary codes. Thanks to this high accuracy, we adopt to use it for encoding image descriptors and using their binary codes for NBNN techniques.

III. MEMORY-EFFICIENT NBNN

In this section, we first explain the original NBNN technique. We then explain two main components of our method: binarization and inverted indexing.

A. NBNN based classification

Let us represent an image I as a set of local descriptors, i.e., $I = \{d_1, d_2, \cdots, d_n\}$. In order to classify the image with NBNN, we define and measure the image-to-class distance, D_{ItC}, which uses a descriptor-to-class distance, D_{DtC}. We also define $NN_c(d)$ to be the nearest neighbor descriptor to the given descriptor d among descriptors assigned to the class c. The descriptor-to-class and image-to-class distances can be then defined as follows:

$$D_{DtC}(d, c) = ||d - NN_c(d)||, \tag{1}$$

$$D_{ItC}(I, c) = \sum_{i=1}^{n} D_{DtC}(d_i, c), \tag{2}$$

where n is the number of the local descriptors extracted from the image I.

NBNN identified a class of an image I according to the following equation, which is derived by simplifying the maximum likelihood classifier based on the naive Bayes probabilistic model [1]:

$$\hat{c} = \underset{c}{argmin} D_{ItC}(I, c). \tag{3}$$

NBNN technique relies on computing the nearest neighbor given a descriptor. This nearest neighbor search is efficiently supported by Approximate Nearest Neighbor (ANN) search methods using kd-trees [12]. By utilizing kd-trees, we can achieve fast search performance. Nonetheless, we found that this nearest neighbor search is still the main bottleneck of NBNN and can take 85% of the total computation of the NBNN method in our experiment. Furthermore, the memory requirement of storing local descriptors and such tree-based indexing structure is high.

B. Binarization of descriptors

Our main goal is to perform nearest neighbor search in a memory-efficient manner, which is the main computational component of NBNN techniques. Fortunately, nearest neighbor search has been studied well even for high-dimensional data such as our image descriptors. Especially, for such high-dimensional problems, hashing techniques have been demonstrated to work well and well-known examples include locality sensitive hashing [19]. These hashing techniques can work as binary code embedding that compactly represents data points based on hashing functions.

In order to present image descriptors as a binary code for our problem, we utilize spherical hashing [22]. Spherical hashing is one of the state-of-the-art methods to represent high dimensional points into compact binary codes. Most prior works used hyperplanes to partition data into two sets and to encode those partitioned data with one bit (0 for one set or 1 for the other set).

On the other hand, spherical hashing computes binary codes based on hyperspheres, each of which tightly bounds input data. While $D + 1$ hyperplanes are required to define a closed region in a D dimensional space, one hypersphere is enough to define such a closed region. In other words, the average of the maximum distance among points with the same binary code can be bounded, and thus errors caused by representing original data into such binary codes can be bounded too, resulting in better approximate nearest search while compactly representing data. Thanks to this property, spherical hashing has been demonstrated to show higher accuracy over other hyperplane based techniques given the same number of bit lengths. Nonetheless, any binary code embedding techniques can be used instead of spherical hashing, our chosen method for this work.

Suppose that we represent an image descriptor, d, to a binary code, b, by using a binary code embedding or hashing method, $h(\cdot)$; i.e. $b = h(d)$. The image I is then represented as a set of binary codes, $I_b = \{b_1, b_2, \cdots, b_n\}$, which are computed by applying the hashing function to the original image descriptors.

Once we represent descriptors into binary codes, we cannot use distance functions defined with those original image descriptors. Instead, we define a distance function between a binary code and a class, D_{BtC}, as the following:

$$D_{BtC}(b, c) = HD(b, NN_c(b)), \tag{4}$$

where $HD(\cdot, \cdot)$ is the Hamming distance between two binary codes. By replacing D_{DtC} by D_{BtC} in Eq. 2 and 3, we have the classification function for our method using binary codes:

$$D_{I_b tC}(I, c) = \sum_{i=1}^{n} D_{BtC}(b_i, c), \tag{5}$$

$$\hat{c} = \underset{c}{argmin} D_{I_b tC}(I, c). \tag{6}$$

While we can represent image descriptors with binary codes, we lose information of original image

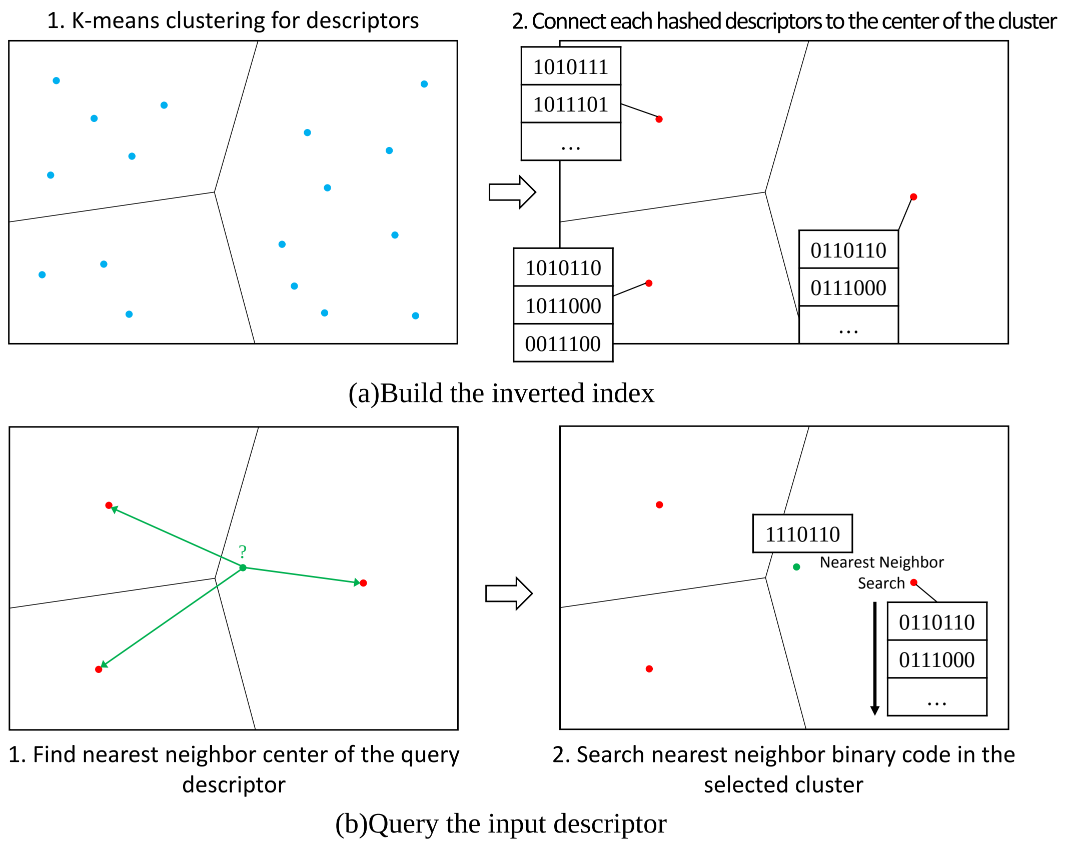
1. K-means clustering for descriptors
2. Connect each hashed descriptors to the center of the cluster
1010111
1011101
...
1010110
1011000
0011100
0110110
0111000
...
(a)Build the inverted index
?
1110110
Nearest Neighbor Search
0110110
0111000
...
1. Find nearest neighbor center of the query descriptor
2. Search nearest neighbor binary code in the selected cluster
(b)Query the input descriptor

Fig. 1. The top row (a) shows the inverted indexing structure for our method, while the bottom row (b) shows how to access the structure to identify nearest neighbors. Blue, red, green dots represent training image descriptors, cluster centers, and a query image descriptor, respectively.

descriptors during binary code embedding. As a result, the accuracy of our approximate nearest neighbor search goes down, as a smaller bit is used for encoding binary codes. We discuss behaviors of accuracy and memory requirement of bit lengths in Sec. IV.

C. Indexing

We can reduce the memory requirement by applying binary hashing to the image descriptors. We still, however, have the issue of query time scalability. As the nearest neighbor operation on original image descriptors causes a time scalability problem taking most of the query time in the original NBNN, the nearest neighbor operation on binary codes can also cause a similar problem if we use a linear search algorithm to find the closest code.

To address this time scalability problem, we need a proper indexing method to perform accurate, yet fast nearest neighbor search. Unfortunately, ANN using kd-trees can be applied even to the nearest neighbor search on binary codes, but its performance would be very inefficient, since kd-trees have been to work well mainly for low-dimensional problems.

To support an efficient search of identifying nearest neighbors to the given query, we adopt an inverted indexing structure as illustrated in Fig. 1. To build the inverted index, we perform the following steps:

1. **Computing clusters.** We perform k-means clustering on the original descriptors to build clusters. Any clustering methods can be used instead of the simple k-means clustering. Especially, product quantization has been demonstrated to work well with binary codes and high-dimensional descriptors [23].
2. **Assigning to the closet cluster.** For each original descriptor, we identify its closest cluster by computing the distance between the descriptor and centers of clusters. Instead of storing the original descriptor, we compute a binary code of the descriptor and associate the binary code with the cluster. We can then efficiently organize our inverted index with our binary codes. When we want to access their original descriptors and images, we also store these data associated with each cluster in a secondary memory space (e.g., disk).

For simplicity, we explained the simple, inverted index. Recently, multi-index has been proposed [24], and can be more complex, yet more efficient for large-scale problems.

At a query time, we use the computed inverted index as the following:

1. **Finding the nearest cluster.** Given a query, we identify the nearest cluster among the cluster centers.
2. **Identifying k nearest neighbors.** Given the nearest cluster, we access binary codes of image descriptors associated with the cluster. We first convert the image descriptor of the query into a binary code. We then measure the Hamming distances between binary codes of the query and others associated with the cluster. By performing sorting according to the computed Hamming distance, we can identify k nearest neighbors.

By using the inverted index, we can efficiently identify potential candidates of k nearest neighbors from the query data. The aforementioned inverted index requires the number clusters for computing center clusters. Depending on the number of clusters, we can control the number of descriptors per each cluster. In Sec. IV-B, we discuss effects of varying number of clusters.

IV. EXPERIMENTS

In this section, we performed experiments to compare the performance of our memory efficient NBNN to those of original NBNN and local NBNN methods. Especially, we focused on the query time, classification accuracy, and memory usage of different NBNN based image classification methods.

A. Implementation and Datasets

We used 101 classes of Caltech-101 image dataset [25], excluding the background class. We utilized densely extracted SIFTs [26] as the local descriptors. For extracting SIFTs densely, we divide an image into multi-resolution grids instead of using an ordinary keypoint extracting algorithm, and extracted features in a multi-scale manner.

We followed the experiment protocol laid out by the prior work [1] to set the experiment environment for our paper. We randomly choose 15 training images and 15 test images for each class. 64 bit code length is used, unless mentioned otherwise, when binary code embedding is applied to descriptors.

We implemented NBNN [1] and local NBNN [9] based on guidelines mentioned in their corresponding papers. These methods utilize a fast approximate nearest neighbor search method, FLANN [27], to efficiently identify nearest neighbors based on kd-trees. We use L1 and L2 distances to calculate the distance between original image descriptors, and use the Hamming distance to measure the distance between binary codes for our method.

The memory requirement of our methods can be controlled by changing the number of bits used

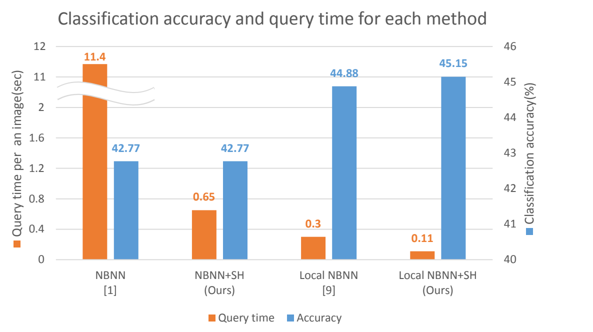

Fig. 2. Average classification accuracy and query time of different methods. Our methods (NBNN+SH and Local NBNN + SH) show better query performance over their corresponding methods (NBNN and Local NBNN, respectively), while maintaining or showing even higher classification accuracy. For the test, we use 64 bit code lengths for our methods.

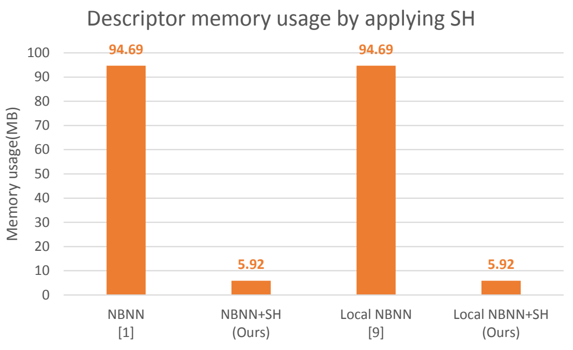

Fig. 3. Memory requirement of image descriptors used in different methods. Our method uses 8 bytes for binary codes, while prior NBNN methods use SIFTs, which are encoded by 128 bytes. In terms of image descriptors, there is no difference between NBNN and local NBNN methods.

for the hashing function. For example, if we use 64 bits code length for our hashing function, which is long enough to maintain the classification accuracy in most cases, a single SIFT descriptor whose size is 128 B can be reduced by a factor of 16 times. Even though the order of the space complexity remains unchanged, reducing the memory requirement even with a constant factor is highly effective. Especially when the raw data size that is bigger than hardware memory capacity is reduced and fit into the available memory capacity, we can observe drastic performance improvement, due to the drastic difference of accessing speed between main memory and auxiliary memory like a hard disk [28].

B. Results

We performed experiments to compare the performance of our memory-efficient NBNN classifier combined with spherical hashing (NBNN+SH) with the original NBNN. We also apply spherical hashing to the Local NBNN method (Local NBNN+SH) [9] and compare its performance with the original local NBNN (Local NBNN). The classification accuracy and query time of tested classification methods are shown in Fig 2. We measure classification accuracy as the ratio of the correctly classified query images over all the test images. The average query time per image is calculated by measuring the total time taken for classifying all test images and dividing it by the number of test images.

We set the number of clusters to be 30 in NBNN+SH and 2000 in Local NBNN+SH; our methods also adopt the same numbers of clusters. We set these parameters differently because when the number of descriptors in an indexing scheme becomes bigger, the number of clusters should be bigger for better performance. For NBNN, the number of descriptors in a single indexing scheme is much smaller than that of Local NBNN, because NBNN builds an indexing structure for each class, while Local NBNN manages all the descriptors in a single indexing structure.

First of all, we observe faster running time and higher accuracy by using the local NBNN over the original NBNN, as demonstrated by the paper of local NBNN [9]. Furthermore, by using our method applying the spherical hashing and the inverted index to those prior NBNN techniques, we are able to observe higher query performance without a significant loss of accuracy. For the case of NBNN, the query performance of NBNN+SH is more than 10 times faster than that of NBNN even with a slight accuracy improvement. This drastic performance improvement is achieved mainly because computing the Hamming distance between binary codes is much faster than the Euclidean computation between the original SIFT descriptors. We also conjecture that hashing, a type of dimension reduction techniques, cancels variance of image descriptors of the same object, resulting in a slightly higher accuracy in this case.

We observe the similar trend even between Local NBNN+SH and Local NBNN. Comparing Local NBNN+SH to Local NBNN, ours shows about three times performance improvement, while the classification accuracies are also similar.

We also measure the memory requirement of different methods. Our methods combined with spherical hashing show a significant advantage over the original NBNN methods, because only 8 bytes are used to represent a binary code, while 128 bytes are needed to represent one SIFT descriptor. This difference results in 16 times less memory usage excluding the overhead for constructing the indexing scheme, while preserving classification accuracy and improving query time. Fig 3 shows the memory requirements of different methods tested in our experiment environment. Considering that the SIFT descriptor is relatively lower dimensional data among available image descriptors, more advantage can be observed in higher dimensional spaces such as features from convolutional neural nets.

We also investigate effects of having different numbers of clusters of our indexing structure used with the binary codes (Table I). For the test, we use the local NBNN combined with spherical hashing. In all the tested cases, the classification accuracies are similar to each other, ranging between 42% and 46%. The overall query time is getting smaller when the number of clusters gets larger, but gets longer when the number of clusters becomes too large (e.g., 4 k clusters) for the tested dataset. When we have a small number of clusters, finding the nearest cluster is fast, but the cluster is associated with many images and thus it requires a long computation time to find the nearest image among them given the query image. On the other hand, when the number of clusters is too high, finding the nearest cluster takes a long time, resulting in a longer computation time. Given this trade-off, the best performance is achieved when we have 2 k clusters for the tested benchmark.

While our methods are not directly tested on large-scale data consisting of more than one million images, we discuss the memory requirement briefly here when different NBNN methods are applied to such a large-scale data. In the case of ILSVRC2010 [29], which is one of the popular large-scale image dataset and consists of 1000 classes, the memory requirement for descriptors is less than 2 GB for the local NBNN+SH, when 500 training images are used for each class. On the other hand, more than 30 GB is required for using the local NBNN. We assume that 512 SIFT features are extracted from each image as same as the case of the other tested experiment with the Caltech image dataset.

V. CONCLUSION & FUTURE WORKS

In this paper, we have applied a binary code embedding, spherical hashing, to NBNN based image classifiers to compactly represent descriptors used for classifiers. We have also tested the inverted index for efficiently perform approximate nearest neighbor search with those computed binary codes. To demonstrate benefits of our methods, we have tested them in a well-known benchmark, Caltech-101 image dataset. When we use 64 bit lengths, we were able to observe that the proposed methods show similar classification accuracy and query speed, while reducing the memory requirement by a factor of 16 over prior NBNN methods. This is mainly achieved thanks to accurate binary code embedding adopted together with the inverted index structure.

Many interesting research directions lie ahead. We would like to utilize global image features such as features from convolution neural net [4]. Because NBNN classifiers assume local image descriptors as local features, extending NBNN classifiers to

Table I. Effects of having varying numbers of clusters for Local NBNN+SH. We achieve the best performance when we have 2 k clusters for the local NBNN combined with spherical hashing.

	The number of clusters				
	50	100	1000	2000	4000
Query time (ms)	2144	1060	146	123	322
Time for finding nearest center(ms)	16	33	31	33	34
Time for finding nearest binary code(ms)	1610	791	99	69	50
Accuracy	44.09	42.90	44.88	44.42	42.97

work with global features is an interesting research problem. Since we have verified benefits of the inverted index structure, it would be worthwhile to investigate other advanced techniques such as multi-index and recent shortlist selection method [24] designed for efficient, high-dimensional nearest neighbor search. We believe that this line of research helps to improvement the scalability of NBNN based approaches, which is one of important data-driven classification methods.

ACKNOWLEDGEMENTS

This work is performed in part by MSIP/NRF No. 2013-067321 and MSIP/IITP [R0126-16-1108].

BIOGRAPHIES

YoonSeok Lee is a software engineer at the Digital Content&Audio Platform team of NAVER Corp., Seongnam in South Korea. He received the B.S. and M.S. degrees in computer science from KAIST in 2014 and 2016, respectively. His research interest lies in image classification, image representation and hashing techniques.

Sung-Eui Yoon is currently an associate professor at KAIST. He received the B.S. and M.S. degrees in computer science from Seoul National University in 1999 and 2001, respectively. His main research interest is on designing scalable graphics, image search, and geometric algorithms. He gave numerous tutorials on proximity queries and large-scale rendering at various conferences including ACM SIGGRAPH and IEEE Visualization. Some of his work received a distinguished paper award at Pacific Graphics, invitations to IEEE TVCG, an ACM student research competition award, and other domestic research-related awards. He is a senior member of IEEE, and a member of ACM and KIISE.

REFERENCES

[1] O. Boiman, E. Schechtman, and M. Irani, “In defense of nearest neighbor based image classification,” in *CVPR*, 2008.

[2] T. Tommasi and B. Caputo, “Frustratingly easy nbnn domain adaptation,” in *ICCV*, 2013.

[3] I. Kuzborskij, F. M. Carlucci, and B. Caputo, “When nave bayes nearest neighbors meet convolutional neural networks,” in *CVPR*, 2016.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *NIPS*, 2012.

[5] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *CVPR*, 2014.

[6] N. Zhang, J. Donahue, R. Girshick, and T. Darrell, “Part-based R-CNNs for fine-grained category detection,” in *ECCV*, 2014.

[7] L. Xie, R. Hong, B. Zhang, and Q. Tian, “Image classification and retrieval are one,” in *ICMR*, 2015.

[8] T. Tuytelaars, M. Fritz, K. Saenko, and T. Darrell, “The nbnn kernel,” in *ICCV*, 2011.

[9] S. McCann and D. G. Lowe, “Local naive bayes nearest neighbor for image classification,” in *CVPR*, 2012.

[10] M. Sun, Y. Lee, and S.-E. Yoon, “Relation based bayesian network for nbnn,” *JCSE*, vol. 9, no. 4, pp. 204–213, 2015.

[11] R. Behmo, P. Marcombes, A. Dalalyan, and V. Prinet, “Towards optimal naive bayes nearest neighbor,” in *ECCV*, 2010.

[12] J. H. Friedman, J. L. Bentley, and R. A. Finkel, “An algorithm for finding best matches in logarithmic expected time,” *ACM Trans. Math. Softw.*, vol. 3, no. 3, pp. 209–226, 1977.

[13] K. Kim, M. K. Hasan, J.-P. Heo, Y.-W. Tai, and S.-E. Yoon, “Probabilistic cost model for nearest neighbor search in image retrieval,” *Computer Vision and Image Understanding (CVIU)*, 2012.

[14] C. Silpa-Anan, R. Hartley, S. Machines, and A. Canberra, “Optimised kd-trees for fast image descriptor matching,” in *CVPR*, 2008.

[15] Y. Jia, J. Wang, G. Zeng, H. Zha, and X.-S. Hua, “Optimizing kd-trees for scalable visual descriptor indexing,” in *CVPR*, 2010.

[16] M. Muja and D. G. Lowe, “Fast approximate nearest neighbors with automatic algorithm configuration,” in *Int’l Conf. Computer Vision Theory and Applications*, 2009.

[17] J. Wang, S. Kumar, and S.-F. Chang, "Semi-supervised hashing for scalable image retrieval," in *CVPR*, 2010.

[18] Y. Gong and S. Lazebnik, "Iterative quantization: a procrustean approach to learning binary codes," in *CVPR*, 2011.

[19] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *STOC*, 1998.

[20] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *SoCG*, 2004.

[21] J. Pan and D. Manocha, "Fast gpu-based locality sensitive hashing for k-nearest neighbor computation," in *ACM SIGSPATIAL GIS*, 2011.

[22] J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S.-E. Yoon, "Spherical hashing," in *CVPR*, 2012.

[23] H. Jegou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE TPAMI*, vol. 33, no. 1, pp. 117–128, 2011.

[24] J.-P. Heo, Z. Lin, X. Shen, J. Brandt, and S. eui Yoon, "Shortlist selection with residual-aware distance estimator for k-nearest neighbor search," in *CVPR*, 2016.

[25] 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," in *CVPRW*, 2004.

[26] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, vol. 60, no. 2, pp. 91–110, 2004.

[27] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *VISSAPP*, 2009.

[28] S.-E. Yoon, C. Lauterbach, and D. Manocha, "R-LODs: Interactive LOD-based Ray Tracing of Massive Models," *The Visual Computer (Pacific Graphics)*, vol. 22, no. 9–11, pp. 772–784, 2006.

[29] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *IJCV*, vol. 115, no. 3, pp. 211–252, 2015.