JCSE Journal of Computing Science and Engineering

Regular Paper

Journal of Computing Science and Engineering,
Vol. 11, No. 1, March 2017, pp. 1-8

Memory-Efficient NBNN Image Classification

YoonSeok Lee and Sung-Eui Yoon*
School of Computing, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea
ys.lee@kaist.ac.kr, sungeui@gmail.com

Abstract
Naive Bayes 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. However, it has a distinct disadvantage. Its memory requirement can be prohibitively high while processing a large amount of data. To deal with this problem, we apply a spherical hashing binary code embedding technique, to compactly encode data without significantly losing classification accuracy. We also propose using an inverted index to identify nearest neighbors among binarized image descriptors. To demonstrate the benefits of our method, we apply our method to two existing NBNN techniques with an image dataset. By using 64 bit length, we are able to reduce memory 16 times with higher runtime performance and no significant loss of classification accuracy. This result is achieved by our compact encoding scheme for image descriptors without losing much information from original image descriptors.

Category: Smart and intelligent computing

Keywords: Image classification; NBNN; Hashing; Memory efficiency; Indexing

I. INTRODUCTION

Image classification assigns 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, naïve Bayes nearest neighbor (NBNN) [1] is one popular image classifier that does not require an explicit learning process. NBNN is designed based on the naïve Bayes assumption and using nearest neighbor search. It usually uses local descriptors (e.g., SIFTs), which are 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 since 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.

The NBNN approach has advantages over other learning based techniques for image classification. NBNN is theoretically simple and easy to implement. 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, follow-up studies

Open Access http://dx.doi.org/10.5626/JCSE.2017.11.1.1 http://jcse.kiise.org

Received 13 September 2016; **Accepted** 13 December 2016
*Corresponding Author

 pISSN: 1976-4677 eISSN: 2093-8020

are being actively conducted [5, 6]. Interestingly, there is a recent study [7] for combining CNNs and NBNN to achieve even higher accuracy. Furthermore, NBNN techniques can be used in situations where 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 requirements. Accuracy and slow performance have been addressed by many prior approaches [3, 8-10], 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 an approximate, yet fast, nearest neighbor search 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. Using our method, we are able to observe faster running performance and lower required memory without a significant loss of classification accuracy. Especially, when we use 64 bit length for binary codes, we are able to achieve a 16 times memory reduction over those two NBNN approaches, while achieving 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, the source code of our approach will be available (http://sglab.kaist.ac.kr/projects/NBNN_Memory).

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 local descriptors similar 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 a data adaptation problem [2] and image retrieval [7].

In order to address a high runtime overhead in querying, McCann and Lowe [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 using unquantized original descriptors has not yet been given much consideration.

Nearest neighbor search. Exact or approximate nearest neighbor search has been widely studied. One of the most common acceleration data structures for the search is kd-trees [12]. The kd-trees have also been widely adopted in many computer vision and various optimization techniques [13]. Other 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 these 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 higher accuracy for the search problem, by computing hashing functions considering input data. Unfortunately, most of these techniques tend to rely on learning techniques or require high computation time. Therefore, 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 in many different directions including supporting 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 has been proposed [22]. This technique can generate more closed regions in high dimensional spaces, resulting in high accuracy for an approximate neighbor search. This property can preserve the distances between the

original data with their corresponding binary codes. Due to this high accuracy, we used it for encoding image descriptors and used their binary codes for NBNN techniques.

III. MEMORY-EFFICIENT NBNN

In this section, we first explain the original NBNN technique. We then explain the 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., $I = \{d_1, d_2, \ldots, d_n\}$. 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 naïve 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, there is a high memory requirement for storing local descriptors and tree-based indexing structure.

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 well studied 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 distances 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 also. This results 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, $I_b = \{b_1, b_2, \ldots, b_n\}$, 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 Eqs. (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 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 Section IV.

C. Indexing

We can reduce the memory requirement by applying binary hashing to the image descriptors. However, we still have the issue of query time scalability. Since 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 searches. 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 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 the 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

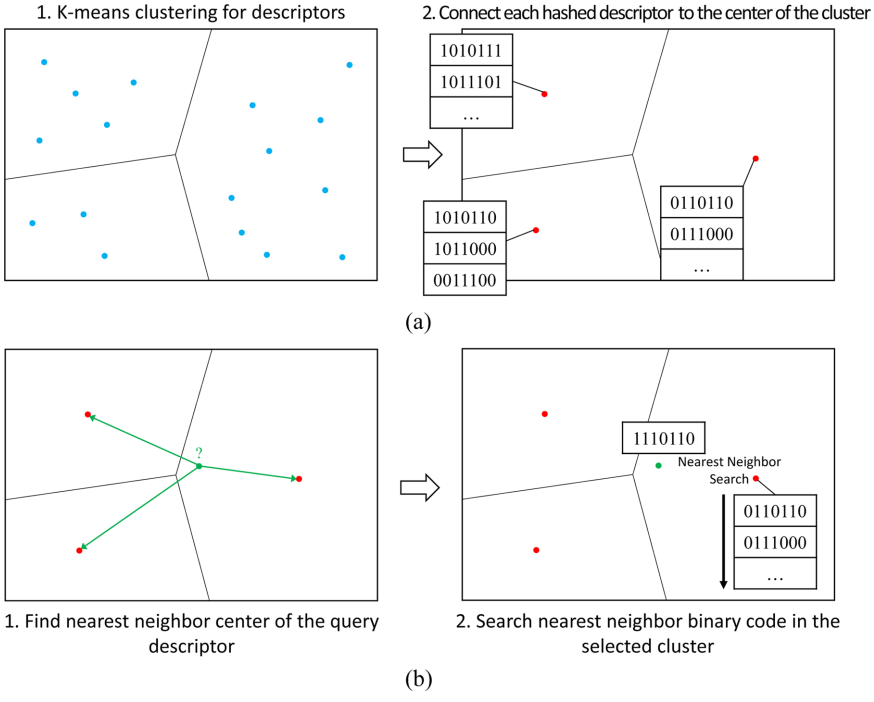

Fig. 1. The top row shows the inverted indexing structure for our method, while the bottom row shows how to access the structure to identify nearest neighbors. (a) Build the inverted index and (b) query the input descriptor. Blue, red, and green dots represent training image descriptors, cluster centers,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

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.

Using the inverted index, we can efficiently identify potential candidates of k nearest neighbors from the query data. This inverted index requires the number of clusters for computing center clusters. Depending on the number of clusters, we can control the number of descriptors per cluster. In Section IV-B, we discuss the effects of varying the 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. The 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 the guidelines mentioned in their corresponding papers. These methods utilize a fast approximate nearest neighbor search method, FLANN [16], to efficiently identify nearest neighbors based on kd-trees. We used L1 and L2 distances to calculate the distance between original image descriptors, and used the Hamming distance to measure the distance between binary codes for our method.

The memory requirement of our method can be controlled by changing the number of bits used for the hashing function. For example, if we use 64 bit 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. This is due to the drastic difference of accessing speed between main memory and auxiliary memory like a hard disk [27].

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 2,000 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. This is 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 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 tech-

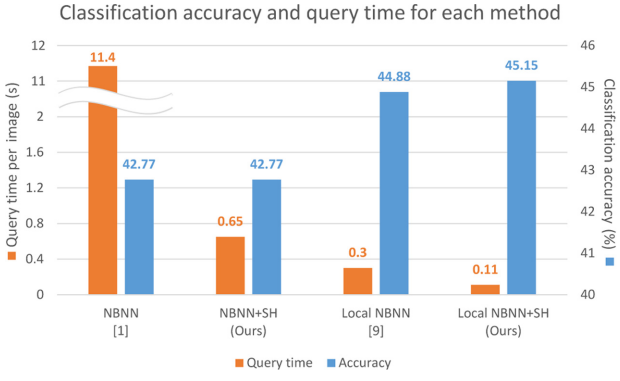

Fig. 2. Average classification accuracy and query time of different methods. Our methods (NBNN+SH and Local NBNN+SH) show better query performance over their corresponding methods (NBNN and Local NBNN, respectively), while maintaining or showing even higher classification accuracy. For the test, we use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

niques, 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, as a type of dimension reduction technique, 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, our method 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.

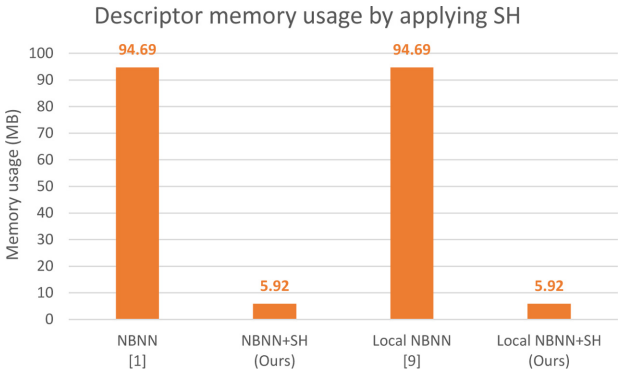

Fig. 3. Memory requirement of image descriptors used in different methods. Our method uses 8 bytes for binary codes, while prior NBNN methods use SIFTs, which are encoded by 128 bytes. In terms of image descriptors, there is no difference between NBNN and local NBNN methods.

We also investigated the effects of having different numbers of clusters in our indexing structure used with the binary codes (Table 1). For the test, we use the local NBNN combined with spherical hashing. In all the tested cases, the classification accuracies are similar to one another, ranging between 42% and 46%. The overall query time gets 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 therefore requires a long computation time to find the nearest image among them to the given 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 large-scale data. In the case of ILSVRC2010 [28], which is a popular large-scale image dataset consisting of 1,000 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 the 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

Table 1. Effects of having varying numbers of clusters for Local NBNN+SH

	Number of clusters				
	50	**100**	**1,000**	**2,000**	**4,000**
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

We achieve the best performance when we have 2 k clusters for the local NBNN combined with spherical hashing (SH).

to compactly represent descriptors used for classifiers. We have also tested the inverted index for efficiently performing the approximate nearest neighbor search with those computed binary codes. To demonstrate the benefits of our methods, we tested them in a well-known benchmark, the Caltech-101 image dataset. When we used 64 bit length, 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 work with global features is an interesting research problem. Since we have verified the benefits of the inverted index structure, it would be worthwhile to investigate other advanced techniques such as multi-index and the recent shortlist selection method [24] designed for efficient, high-dimensional nearest neighbor search. We believe that this line of research helps to improve the scalability of NBNN based approaches, which is an important data driven classification method.

ACKNOWLEDGMENTS

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

REFERENCES

1. O. Boiman, E. Schechtman, and M. Irani, "In defense of nearest neighbor based image classification," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Anchorage, AK, 2008, pp. 1-8.
2. T. Tommasi and B. Caputo, "Frustratingly easy NBNN domain adaptation," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Sydney, 2013, pp. 897-904.
3. I. Kuzborskij, F. Maria Carlucci, and B. Caputo, "When naive Bayes nearest neighbors meet convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, 2016, pp. 2100-2109.
4. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097-1105, 2012.
5. R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Columbus, OH, 2014, pp. 580-587.
6. N. Zhang, J. Donahue, R. Girshick, and T. Darrell, "Part-based R-CNNs for fine-grained category detection," in *Proceedings of European Conference on Computer Vision*, Zurich, Switzerland, 2014, pp. 834-849.
7. L. Xie, R. Hong, B. Zhang, and Q. Tian, "Image classification and retrieval are one," in *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval (ICMR)*, Shanghai, China, 2015, pp. 3-10.
8. T. Tuytelaars, M. Fritz, K. Saenko, and T. Darrell, "The NBNN kernel," in *Proceedings of 2011 IEEE International Conference on Computer Vision (ICCV)*, Barcelona, Spain, 2011, pp. 1824-1831.
9. S. McCann and D. G. Lowe, "Local naive Bayes nearest neighbor for image classification," in *Proceedings of 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* Providence, RI, 2012, pp. 3650-3656.
10. M. Sun, Y. Lee, and S. E. Yoon, "Relation based Bayesian network for NBNN," *Journal of Computing Science and Engineering*, 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 *Proceedings of European Conference on Computer Vision*, Crete, Greece, 2010, pp. 171-184.
12. J. H. Friedman, J. L. Bentley, and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Transactions on Mathematical Software (TOMS)*, 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*, vol. 116, no. 9, pp. 991-998, 2012.
14. C. Silpa-Anan and R. Hartley, "Optimised KD-trees for fast image descriptor matching," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Anchorage, AK, 2008, pp. 1-8.
15. Y. Jia, J. Wang, G. Zeng, H. Zha, and X. S. Hua, "Optimizing kd-trees for scalable visual descriptor indexing," in *Proceedings of 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* San Francisco, CA, 2010, pp. 3392-3399.
16. M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *Proceedings of the International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISAPP)*, Lisbon, Portugal, 2009, pp. 331-340.
17. J. Wang, S. Kumar, and S. F. Chang, "Semi-supervised hashing for scalable image retrieval," in *Proceedings of 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* San Francisco, CA, 2010, pp. 3424-3431.
18. Y. Gong and S. Lazebnik, "Iterative quantization: a procrustean approach to learning binary codes," in *Proceedings of 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Providence, RI, 2011, pp. 817-824.
19. P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," in *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, Dallas, TX, 1998, pp. 604-613.
20. M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni,

"Locality-sensitive hashing scheme based on p-stable distributions," in *Proceedings of the 20th Annual Symposium on Computational Geometry*, Brooklyn, NY, 2004, pp. 253-262.
21. J. Pan and D. Manocha, "Fast GPU-based locality sensitive hashing for k-nearest neighbor computation," in *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, Chicago, IL, 2011, pp. 211-220.
22. J. P. Heo, Y. Lee, J. He, S. F. Chang, and S. E. Yoon, "Spherical hashing," in *Proceedings of 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Providence, RI, 2012, pp. 2957-2964.
23. H. Jegou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 1, pp. 117-128, 2011.
24. J. P. Heo, Z. Lin, X. Shen, J. Brandt, and S. E. Yoon, "Shortlist selection with residual-aware distance estimator for k-nearest neighbor search," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, 2016, pp. 2009-2017.
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 *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Washington, DC, 2004, pp. 1-9.
26. D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91-110, 2004.
27. S. E. Yoon, C. Lauterbach, and D. Manocha, "R-LODs: fast LOD-based ray tracing of massive models," *The Visual Computer*, vol. 22, no. 9, pp. 772-784, 2006.
28. O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, et al., "ImageNet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211-252, 2015.

YoonSeok Lee

YoonSeok Lee is a software engineer at the Digital Content & Audio Platform team of NAVER Corp., Seongnam, 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

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 in 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.