

論文 / 著書情報  
Article / Book Information

Title	Fast Coding of Feature Vectors using Neighbor-To-Neighbor Search
Author	Nakamasa Inoue, Koichi Shinoda
Journal/Book name	IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), vol. 38, no. 6, pp. 1170-1184
Issue date	2015, 9
DOI	<a href="http://dx.doi.org/10.1109/TPAMI.2015.2481390">http://dx.doi.org/10.1109/TPAMI.2015.2481390</a>
URL	<a href="http://www.ieee.org/index.html">http://www.ieee.org/index.html</a>
Copyright	(c)2015 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other users, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works for resale or redistribution to servers or lists, or reuse of any copyrighted components of this work in other works.
Note	このファイルは著者（最終）版です。 This file is author (final) version.

# Fast Coding of Feature Vectors using Neighbor-To-Neighbor Search

Nakamasa Inoue, *Member, IEEE*, and Koichi Shinoda, *Senior Member, IEEE*,

**Abstract**—Searching for matches to high-dimensional vectors using hard/soft vector quantization is the most computationally expensive part of various computer vision algorithms including the bag of visual word (BoW). This paper proposes a fast computation method, Neighbor-to-Neighbor (NTN) search [1], which skips some calculations based on the similarity of input vectors. For example, in image classification using dense SIFT descriptors, the NTN search seeks similar descriptors from a point on a grid to an adjacent point. Applications of the NTN search to vector quantization, a Gaussian mixture model, sparse coding, and a kernel codebook for extracting image or video representation are presented in this paper. We evaluated the proposed method on image and video benchmarks: the PASCAL VOC 2007 Classification Challenge and the TRECVID 2010 Semantic Indexing Task. NTN-VQ reduced the coding cost by 77.4%, and NTN-GMM reduced it by 89.3%, without any significant degradation in classification performance.

**Index Terms**—Neighbor-to-Neighbor Search, Image Classification, Video Semantic Indexing, Vector Quantization, Gaussian Mixture Model

## 1 INTRODUCTION

Searching for matches to high-dimensional vectors is the most computationally expensive part of various computer vision algorithms. Examples of these algorithms include coding of feature vectors in image classification [2], image mosaicing [3], event detection in video [4], [5], [6], action recognition with 3D cameras [7], [8], and 3D shape modeling [9].

Most of them are simplified into either one of two problems: a hard-vector-quantization (VQ) problem or a soft-VQ problem [10], [11]. In hard VQ, each input vector is assigned to its closest codeword [12], [13]. In soft VQ, each input vector is assigned to more than one codewords in a soft weighting manner typically depending on distance between the input vector and a codeword [14], [15]. Probabilistic models are often used for soft weighting. A typical example is a Gaussian mixture model (GMM) in which each codeword has a covariance matrix and a weighting coefficient [15]. Soft VQ has smaller quantization errors but it is computationally costly [16].

Many studies have been done to develop fast hard/soft VQ algorithms. Most of them use a tree structure to reduce the computational cost. For hard VQ, approximate nearest neighbor (ANN) algorithms such as the best bin first search [18], randomized  $kd$ -trees [19], hierarchical  $k$ -means tree [20] are known to provide speed-ups with only minor loss in accuracy. Some other studies extend them to a probabilistic model for soft VQ. For example, Tree-structured GMM in [16], [17] extends hierarchical  $k$ -means to a GMM framework.

These previous studies assume input feature vectors are independent from each other. In contrast, our motivation is in the fact that input vectors are often dependent on each other.

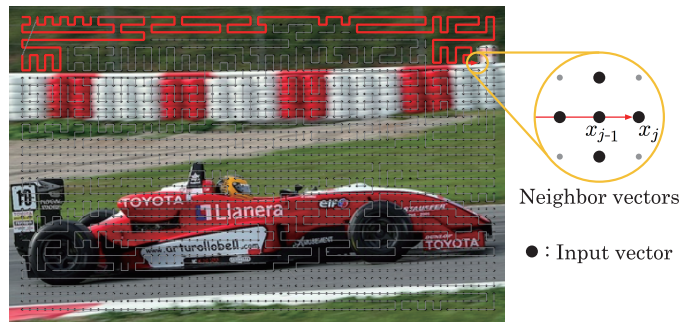


Fig. 1. **Neighbor-to-neighbor (NTN) search.** NTN search assigns a code to an input vector from a neighbor vector to a neighbor vector. A typical example of a neighbor vector is a descriptor  $x_j$  adjacent to a descriptor  $x_{j-1}$  where image descriptors are densely sampled from an image. The red path on the image shows the ordering of computation.

Their typical examples are time-series features in video and spatially related features such as dense SIFT [21], [22]. In dense SIFT, two adjacent descriptors are often assigned to the same codeword since they are similar to each other. This brings us to an idea to speed up coding by skipping calculations for such similar input vectors.

For large-scale image retrieval, deep architectures such as deep convolutional neural networks [23] and deep Fisher networks/kernel [24], [25] have shown to be effective to learn features in a supervised way. Recent studies have introduced coding techniques to deep architectures. For example, image representation based on aggregation of deep learned features is obtained by applying VQ or GMMs in [26], [27]. A hybrid architecture of GMM-based Fisher vectors and deep neural networks has been proposed in [28]. Since deep learning and bag-of-words with dense sampling have a lot in common as

• N. Inoue and K. Shinoda are with the Department of Computer Science, Tokyo Institute of Technology, Japan.  
E-mail: inoue@ks.cs.titech.ac.jp, shinoda@cs.titech.ac.jp

shown in [25], [28], it is also important for deep architectures to improve the speed of coding features.

For video retrieval, it has shown that deep neural networks and bag-of-words benefit from each other [29]. For example, to capture human actions from video, bag-of-words using dense sampling of local descriptors on multiple frames has shown to be effective.

This paper proposes a fast computation method for coding of feature vectors, which we call Neighbor-to-Neighbor (NTN) search [1]. This algorithm, assuming that 1) a set of neighbor vectors of each input vector are defined and 2) an input vector and its neighbor vector are similar, skips some distance calculations between a neighbor vector and a codeword. This algorithm effectively utilizes a triangle inequality for the distances between neighbor vectors. In addition to NTN search for hard VQ (NTN-VQ) and for GMM based soft VQ (NTN-GMM) presented in our previous work [1], we apply NTN search to sparse coding using locality-constrained linear coding (NTN-LLC) [30] and a kernel codebook (NTN-KCB) [14] in this paper. We demonstrate the effectiveness of the NTN search methods in our experiments on image and video benchmarks of the PASCAL VOC 2007 Classification Challenge and the TRECVID 2010 Semantic Indexing Task.

This paper is organized as follows. The next section reviews the related studies. Section 3 explains the NTN algorithm in a simple framework, hard VQ. Sections 4 to 6 describe how to extend it to soft VQ using a GMM, sparse coding and a kernel codebook. Section 7 reports the results of our evaluation, and Section 8 concludes the paper.

## 2 RELATED WORK

### 2.1 Hard VQ

Hard VQ is a classical technique to compress signals such as image, video, and speech [12]. It is used in many computer vision algorithms including bag-of-visual-words (BoW) [2]. A codebook is typically trained by using  $k$ -means algorithm [13]. Hard VQ costs  $O(K)$  to assign one of  $K$  codewords to an input vector in a straightforward way.

Many previous studies reduce the costs from  $O(K)$  to  $O(\log K)$  by using tree structure. The  $kd$ -tree [31], [32], which splits a vector space into rectangular regions, is a traditional way to construct a tree structure on given data. Metric trees [33], [34] and ball trees [35], [36] construct a tree structure, whose nodes represent spherical regions. They are effective for range search to find samples within a given radius around an input query vector, since triangle inequalities are effectively applied to vectors in a database with them. Note that since our NTN search applies triangle inequalities to input vectors in contrast with them, NTN search and tree-based methods are complementary to each other.

In computer vision, Lowe [3] uses a  $kd$ -tree for nearest neighbor search of SIFT descriptors. Nistér *et al.* [37] propose a “vocabulary tree” which uses a hierarchical  $k$ -means tree. Muja and Lowe [20] have proposed an automatic selection method from the recent two approximate nearest neighbor (ANN) algorithms: randomized  $kd$ -trees [19], and a hierarchical  $k$ -means tree [20]. They have provided it as a

fast software library for approximate nearest neighbor search (FLANN). To construct a more precise tree structure, random forests [38], [39] are effective since they learn a tree structure in a supervised way. These tree based methods effectively represent data structure of image descriptors. However, since they assume that input vectors are independent from each other, re-calculation cannot be avoided even if input vectors are the same or similar to the previous ones.

For high dimensional vectors, dimension reduction techniques such as principal component analysis (PCA) [40] and linear discriminant analysis (LDA) [41] are often applied as preprocessing of the above methods. Product quantization [42], which divide a input vector into several sub-vectors in VQ process, is also known to be effective for high dimensional vectors. Note these dimension reduction techniques can be introduced to our proposed NTN search by applying them to input vectors, since our idea to skip calculation based on dependency of input vectors is independent from them.

### 2.2 Soft VQ

To reduce quantization errors in hard VQ, soft VQ assigns more than one codewords to an input vector. For example, a Gaussian mixture model (GMM) [15] provides soft weighting based on the ratios of Gaussian probabilities. The kernel codebook (KCB) [14] uses kernel density estimation for soft VQ. Sparse coding [43] assigns several tens of codewords to an input vector by solving a constrained least square fitting problem. Bag-of-visual-words frameworks incorporated with these methods improves the accuracy in object recognition and scene classification.

Some studies have proposed techniques to reduce the computational cost of soft VQ while keeping the accuracy. Tree-GMM [16] extends the hierarchical  $k$ -means to a GMM framework to calculate Gaussian probabilities quickly. Wang *et al.* [30] introduced  $k$ -nearest neighbor ( $k$ -NN) search as the preprocessing to the sparse coding. Fast sparse coding algorithms which simplify the fitting problem, are also proposed for learning image filters [44] and action recognition [45], [46].

### 2.3 Image representation

Here we review recent image representation based on hard or soft VQ. Bag-of-visual-words (BoW) [2] represents an image by a histogram of visual words. Visual words are obtained by applying hard VQ or soft VQ to image descriptors such as SIFT [3], HOG [47], Color SIFT [48], and LBP [49]. In locality-constrained linear (LLC) coding [30], max pooling with sparse coding is used instead of taking a histogram of visual words.

Super-vector representation (SV) [21] uses the first order differences between input vectors and codewords in addition to the histogram. Hard VQ is often used in SV. Fisher-vector representation (FV) [22] based on a Fisher kernel [50], [51] on a GMM represents an image by a vector of derivation of a log-likelihood function. The final representation of FV is similar to that of SV with the exception that it captures not only the first order but also the second order differences between input

Notation	Description
$X \subset \mathbb{R}^d$	A set of input vectors
$d$	Dimensionality of input vectors
$x_j \in X$	An input vector ( $j = 1, 2, \dots, N$ )
$N$	The number of input vectors
$B(x_j)$	A set of neighbor vectors of $x_j \in X$
$\mu_k$	A codeword vector ( $k = 1, 2, \dots, K$ )
$K$	The number of codewords
$\delta$	A parameter to control speed
$c_{jk}$	A code of $x_j$ for the codeword $\mu_k$

TABLE 1  
Summary of notation.

vectors and codewords in a probabilistic framework. GMM-based soft VQ is used in FV.

K. Chatfield et al. [52] compared recent image representation and reported that the FV representation [22] is the best and the SV representation [21] is the second in terms of image classification accuracy on the PASCAL VOC Classification Challenge. They have noticed that FV and SV are efficient since they work well with a relatively small ( $256 \leq K \leq 1024$ ) codebook. However, the coding step using hard/soft VQ is still the time-bottleneck of a pipeline for extracting image representation.

## 2.4 Video representation

Recent studies have shown that the above image representation methods are also effective for video classification [53], [54]. For example, GMM supervector representation [55], which can be viewed as a simplification of the Fisher-vector representation, have been proven to be effective for semantic indexing of video data [54]. It uses a robust parameter estimation technique, maximum-a-posteriori (MAP) estimation, for a GMM.

By extracting a key-frame from a video clip, a video classification problem can be solved as an image classification problem. However, analyzing multiple frames in video significantly improves the performance of video classification. For example, extracting low-level image descriptors from several hundreds of image frames in video to make a BoW representation of video with hard/soft VQ is effective for semantic concept detection on video data [53]. This shows that improving the speed of hard/soft VQ is an important issue in video classification.

## 3 NEIGHBOR-TO-NEIGHBOR (NTN) SEARCH FOR VECTOR QUANTIZATION

### 3.1 Outline

This section presents our neighbor-to-neighbor (NTN) search in a simple framework, hard VQ. Let  $X$  be a set of input vectors. The NTN search assumes that a set of neighbor vectors  $B(x) \subset X$  for each input vector  $x \in X$  is given. For example,  $B(x)$  is defined as a set of the four descriptors adjacent to a descriptor  $x$  for densely-sampled image descriptors (Figure 1 and Figure 6). We expect that a neighbor vector in  $B(x)$  is

similar to  $x$ , and that the number of neighbor vectors is smaller than the codebook size.

Let  $\{\mu_k\}_{k=1}^K$  be a codebook for VQ. The goal here is to compute codes  $c_{jk}$  of each input vector  $x_j$  for each codeword  $\mu_k$ . Codes are given by

$$c_{jk} = \begin{cases} 1, & \text{if } k = \operatorname{argmax}_k d_{jk}, \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

where

$$d_{jk} = \|x_j - \mu_k\|, \quad (2)$$

is the the Euclidean distance between an input vector  $x_j$  and a codeword  $\mu_k$ . The mathematical notification is summarized in the Table 1.

### 3.2 Algorithm

In NTN search, input vectors are ordered from a neighbor vector to a neighbor vector to skip distance calculations for some input vectors based on a triangle inequality. We first explain the structure of our algorithm and then explain our speeding-up idea.

In the initialization step for  $j = 1$ ,  $x_j$  is randomly selected from  $X$ . To obtain its code  $c_{jk}$ , distance  $d_{jk}$  in Eq. (2) is exhaustively calculated for each  $k = 1, 2, \dots, K$ . This process is the same as the straightforward hard VQ.

For  $j = 2, 3, \dots, N$ , the following three steps are iterated (Figure 2).

#### (STEP 1: Select the next input vector)

For each  $x \in B(x_{j-1})$ , calculate  $\Delta(x) = \|x - x_{j-1}\|$ , and set

$$x_j = \operatorname{argmin}_{x \in B(x_{j-1}) \cap \bar{X}} \Delta(x), \quad (3)$$

where  $\bar{X} = X \setminus \{x_1, \dots, x_{j-1}\}$  is a set of remaining input vectors. If  $B(x_{j-1}) \cap \bar{X} = \emptyset$  then  $x_j$  is randomly picked from  $\bar{X}$ .

#### (STEP 2: Calculate distance)

Set  $k^* = \operatorname{argmax}_k c_{j-1,k}$ .

2-1) Calculate distance  $d_{jk^*}$ .

2-2) For  $k = 1, 2, \dots, k^* - 1, k^* + 1, \dots, K$ , calculate a lower bound  $\underline{d}_{jk}$  for  $d_{jk}$  as follows.

$$\underline{d}_{jk} = d_{ik} - \delta \Delta_{ij}, \quad (4)$$

where  $i$  is the index of the input vector whose distance  $d_{ik}$  has been calculated,  $\delta$  is a parameter, and  $\Delta_{ij}$  is an accumulated distance from  $x_i$  to  $x_j$ . This process will be explained in detail in the next paragraph. If  $\underline{d}_{jk} \geq d_{jk^*}$  then skip calculation of  $d_{jk}$ , otherwise calculate  $d_{jk}$ .

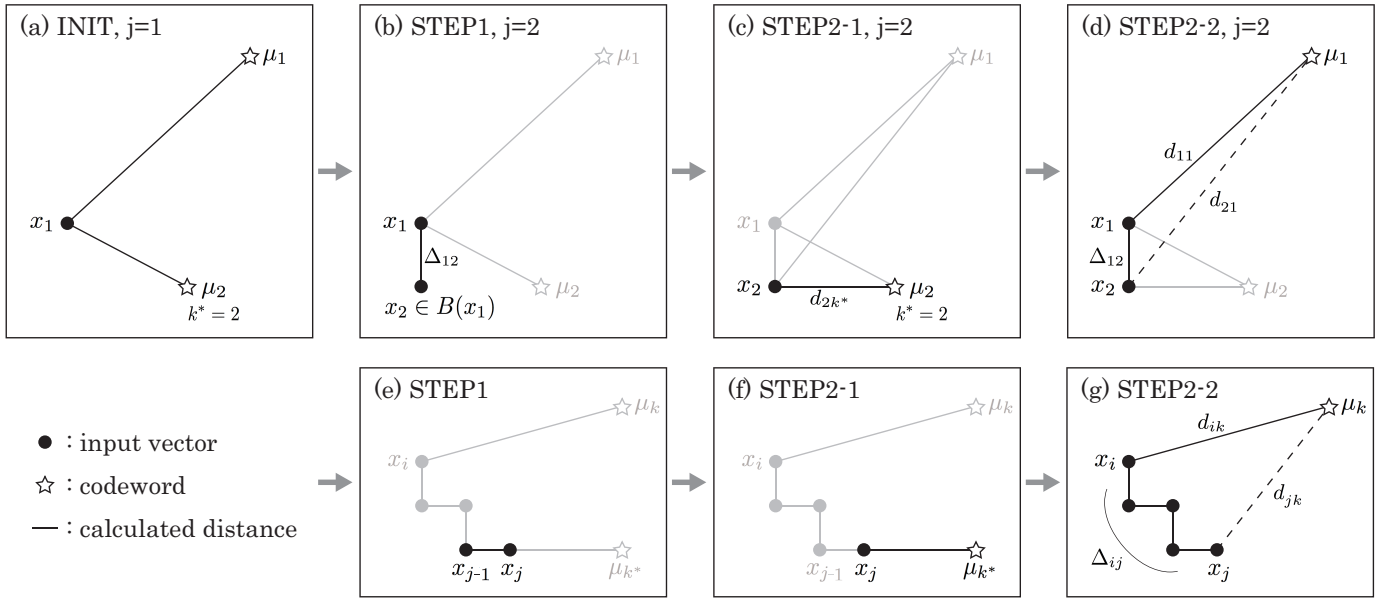
#### (STEP 3: Output a code)

Calculate

$$c_{jk} = \begin{cases} 1, & \text{if } k = \operatorname{argmax}_{k \in E} d_{jk}, \\ 0, & \text{otherwise.} \end{cases} \quad (5)$$

where  $E$  is a set of indices of codewords whose distance to  $x_j$  is calculated in STEP 2.





**Fig. 2. Algorithm overview.** (a) Initialization step: distance from an input vector  $x_1$  to each codeword is calculated. (b) STEP 1: the next input vector  $x_2$  which minimizes  $\Delta_{12}$  is selected from neighbor vectors. (c) STEP 2-1:  $d_{2k^*}$  is calculated where  $k^*$  is the code for  $x_1$ . (d) STEP 2-2: a lower bound  $\underline{d}_{21} = d_{11} - \delta\Delta_{12}$  is calculated where  $\delta$  is a parameter, calculation of  $d_{21}$  is skipped if  $\underline{d}_{21} \geq d_{2k^*}$ . (e),(f),(g): STEP 1, 2-1, and 2-2 for  $x_j (j > 2)$ , respectively. In (g), accumulated distance  $\Delta_{ij}$  between  $x_i$  and  $x_j$  is used to obtain a lower bound  $\underline{d}_{jk} = d_{ik} - \delta\Delta_{ij}$  in Eq. (4).

Here we explain Eq. (4) in STEP 2. For a given  $x_j$ , let's go back to the previous input vector  $x_i$  ( $i < j$ ) whose distance  $d_{ik}$  has been calculated (Figure 2 (g)). Take the maximum such index  $i$  and let  $\Delta_{ij}$  be an accumulated distance between  $x_i$  and  $x_j$  given by

$$\Delta_{ij} = \sum_{p=i+1}^j \|x_p - x_{p-1}\|. \quad (6)$$

The triangle inequality gives

$$d_{ik} - \Delta_{ij} \leq d_{jk} \leq d_{ik} + \Delta_{ij}. \quad (7)$$

It implies

$$\exists \delta^* \in [-1, 1] \text{ s.t. } d_{jk} = d_{ik} - \delta^* \Delta_{ij}. \quad (8)$$

Thus, for  $\delta \geq \delta^*$ ,  $\underline{d}_{jk}$  in Eq. (4) is a lower bound of distance  $d_{jk}$ . Note that the result of coding by this algorithm is exactly the same as that by the original hard VQ in this case.

### 3.3 The parameter $\delta$

Our idea to improve the speed of the algorithm is to regard  $\delta$  as a constant and use it as a parameter. Then, the lower bound is efficiently updated from the previous lower bound by

$$\underline{d}_{jk} = \underline{d}_{j-1,k} - \delta \|x_j - x_{j-1}\|. \quad (9)$$

The lower bound is obtained by only one distance calculation from  $x_{j-1}$  to  $x_j$ , which is already calculated in STEP 1. By relaxing the restriction  $\delta \geq \delta^*$ , we can further reduce the computational cost though the exact solution may not be obtained in such cases. To optimize the parameter  $\delta$ , cross validation is used in practice.

Alg. 1 summarizes the neighbor-to-neighbor search for hard VQ (NTN-VQ) which outputs assigned codes for each input vector quickly.

## 4 NTN SEARCH FOR GAUSSIAN MIXTURE MODELS

A Gaussian mixture model (GMM) is an extension of hard VQ to a probabilistic framework since it provides a soft assignment of codewords to an input vector. Here we extend the NTN search to a GMM framework (NTN-GMM). The algorithm structure of NTN-GMM is the same as NTN-VQ, but instead of a lower bound of distance for NTN-VQ, an upper bound of a Gaussian probability is calculated for NTN-GMM.

Let  $\mu_k, \Sigma_k$  and  $w_k$  ( $k = 1, 2, \dots, K$ ) be the mean vector, the covariance matrix, and the mixture weight of the  $k$ -th mixture component (codeword) of a GMM, respectively. A code  $c_{jk}$  for an input vector  $x_j$  ( $j = 1, 2, \dots, N$ ) to the  $k$ -th codeword is given by

$$c_{jk} = \frac{p_{jk}}{\sum_{k'=1}^K p_{jk'}}. \quad (10)$$

Here  $p_{jk}$  is a Gaussian probability given by

$$p_{jk} = \frac{w_k}{(2\pi)^{\frac{d}{2}} |\Sigma_k|^{\frac{1}{2}}} \exp\left(-\frac{1}{2} \|x_j - \mu_k\|_{\Sigma_k^{-1}}^2\right), \quad (11)$$

where  $\|x\|_A = \sqrt{x^T A x}$ . Note that  $NK$  probability calculations are required in the standard GMM.

Empirical observation in [22], [17] shows that the distribution of  $p_{jk}$  over all the codewords is peaky on the PASCAL VOC image dataset and the TRECVID video dataset, respectively, i.e., for each input vector  $x_j$ , a few  $p_{jk}$ 's have

**Algorithm 1** NTN-VQ

---

**Input:** input vectors  $X$  ( $N = |X|$ ),  
 codebook  $\{\mu_k\}_{k=1}^K$ , parameter  $\delta$ .

**Output:** codes  $\{c_{ik}\}_{i=1}^N$

$x_1 \leftarrow \text{Rand}(X)$   
 $\underline{d}_k \leftarrow \|x_1 - \mu_k\|$  **for all**  $k$   
 $c_{1k} \leftarrow 0$  **for all**  $i, k$   
 $k^* \leftarrow \underset{k}{\text{argmin}} \underline{d}_k; c_{1k^*} \leftarrow 1$

**for**  $i = 2, \dots, N$  **do**  
 $x_i \leftarrow \underset{x \in B(x_{i-1}) \cap \bar{X}}{\text{argmin}} \|x - x_{i-1}\|$   
 $\underline{d}_{k^*} \leftarrow \|x_i - \mu_{k^*}\|$   
**for all**  $k \neq k^*$  **do**  
 $\underline{d}_k \leftarrow \underline{d}_k - \delta \|x_i - x_{i-1}\|$   
**if**  $\underline{d}_{k^*} > \underline{d}_k$  **then**  
 $\underline{d}_{k^*} \leftarrow \|x_i - \mu_k\|$   
**if**  $\underline{d}_{k^*} > \underline{d}_k$  **then**  $k^* \leftarrow k$  **end if**  
**end if**  
**end for**  
 $c_{ik^*} \leftarrow 1$   
**end for**

---

a large value and the others do not. We also confirm this observation in Figure 3. If  $x_j$  is similar to  $x_{j-1}$ , the “peak” is shifting gradually as  $j$  increments. Conversely, the changes in the “bottom (no-peak)” of the distribution are generally small. This observation brings us to an idea that we may ignore the change of Gaussian probabilities in the “bottom” of the distribution.

For a given  $x_j$ , let  $x_i$  ( $i < j$ ) be the previous input vector whose Gaussian probability  $p_{ik}$  has been calculated. The idea is to ignore the difference between  $p_{jk}$  and  $p_{ik}$  and assume  $p_{jk} = p_{ik}$  for  $k \in G_i^{(b)} \cap G_j^{(b)}$  to skip calculation of  $p_{jk}$ . Here,  $G_j^{(b)}$  is a set of mixture components in the “bottom” of the distribution, which we call a bottom set, given by

$$G_j^{(b)} = \{k : p_{jk} < p_{\text{th}}\}. \quad (12)$$

where  $p_{\text{th}}$  is a threshold to categorize the mixture components into “peak” and “bottom”.

A bottom set  $G_j^{(b)}$  cannot be directly observed without computing  $p_{jk}$ . Thus, we introduce an upper bound  $\bar{p}_{jk}$  of a probability  $p_{jk}$  (see Appendix for details) given by

$$\bar{p}_{jk} = p_{ik} \exp(\delta_{ik} \Delta_{ij}). \quad (13)$$

Here,  $\Delta_{ij}$  is the accumulated distance given by Eq. (6) and  $\delta_{ik}$  is given by

$$\delta_{ik} = S_k \delta \|x_i - \mu_k\|_{\Sigma_k^{-1}}, \quad (14)$$

where  $\delta \in [0, 1]$  is a parameter to control the speed of our algorithm (as Subsec. 3.3) and  $S_k$  is the square root of the spectral radius of  $\Sigma_k^{-1}$ . Note that this upper bound is obtained efficiently from a previous upper bound by

$$\bar{p}_{jk} = \bar{p}_{j-1,k} \exp(\delta_{ik} \|x_j - x_{j-1}\|). \quad (15)$$

Finally, instead of the intersection of bottom sets  $G_i^{(b)} \cap G_j^{(b)}$ ,

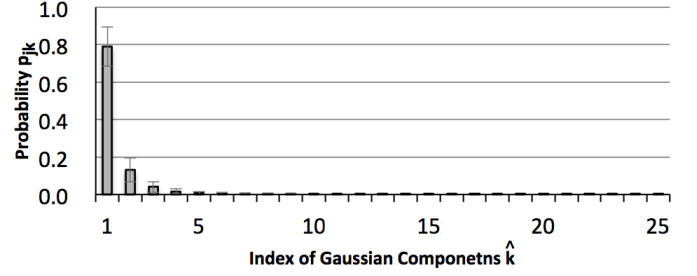


Fig. 3. Values of Gaussian probability  $p_{jk}$ . The indexes of Gaussian components on the horizontal axis are sorted by the probability values, e.g., the maximum value of  $p_{jk}$  over  $k$  is plotted on  $\hat{k} = 1$ . Results for the top 25 of 512 components are used for illustration. On average, five components have a probability value larger than 0.01. This result is obtained on randomly sampled 100 thousand descriptors on PASCAL VOC 2007 training images.

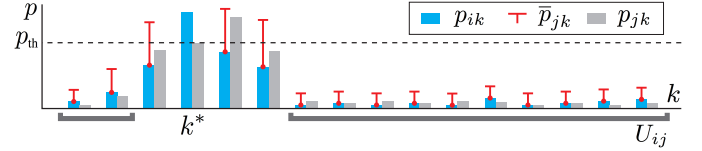


Fig. 4. Distribution of  $p_{ik}$  and  $p_{jk}$  ( $i < j$ ). Calculation of a Gaussian probability  $p_{jk}$  is skipped for  $k \in U_{ij}$ .

its subset  $U_{ij}$  given by

$$U_{ij} = \{k : \bar{p}_{jk} < p_{\text{th}}\}, \quad (16)$$

is used for determining mixture components to skip calculation of  $p_{jk}$  (Figure 4).

The threshold  $p_{\text{th}}$  should depend on the maximum value of Gaussian probabilities at  $j$ , i.e.,  $\max_k p_{jk}$ . However, this value also cannot be observed without computing all Gaussian probabilities at  $j$ . Since two adjacent descriptors are expected to be similar to each other, the value at the previous maximum point is used to determine the threshold as

$$p_{\text{th}} = p_{jk^*}, \quad k^* = \underset{k}{\text{argmax}} p_{j-1,k}. \quad (17)$$

Note that, by this thresholding, Gaussian probabilities at the previous maximum point and the current maximum point (at least) will be calculated for each input vector.

Alg. 2 summarizes the NTN search for a GMM (NTN-GMM) which outputs soft codes for each input vector quickly.

To further improve the speed of NTN-GMM, avoiding the  $\exp$  computation is effective since our observation shows that 63.0% of the computational cost in coding using a GMM is spent for it. An  $\exp$  operator is deleted by taking a log of Gaussian probabilities and introducing log-max (LM) approximation to approximate Eq. (10) by

$$c_{jk} \simeq \begin{cases} 1, & \text{if } k = \underset{k}{\text{argmax}} \log p_{jk}, \\ 0, & \text{otherwise.} \end{cases} \quad (18)$$

**Algorithm 2** NTN-GMM

---

**Input:** input vectors  $X$  ( $N = |X|$ ),  
 GMM  $\{w_k, \mu_k, \Sigma_k\}_{k=1}^K$ , parameter  $\delta$ .

**Output:** soft codes  $\{c_{ik}\}_{i=1}^N \{k=1}^K$

$x_1 \leftarrow \text{Rand}(X)$   
 $p_k, \bar{p}_k \leftarrow w_k \mathcal{N}(x_1 | \mu_k, \Sigma_k)$  **for all**  $k$   
 $\delta_k \leftarrow S_k \delta \|x_1 - \mu_k\|_{\Sigma_k^{-1}}$  **for all**  $k$   
 $c_{1k} \leftarrow \frac{p_k}{\sum_{k'=1}^K p_{k'}}$  **for all**  $k$ ;  $k^* \leftarrow \text{argmax}_k p_k$

**for**  $i = 2, \dots, N$  **do**  
 $x_i \leftarrow \text{argmin}_{x \in B(x_{i-1}) \cap \bar{X}} \|x - x_{i-1}\|$   
 $p_{k^*}, \bar{p}_{k^*} \leftarrow w_{k^*} \mathcal{N}(x_i | \mu_{k^*}, \Sigma_{k^*})$   
**for all**  $k \neq k^*$  **do**  
 $\bar{p}_k \leftarrow \bar{p}_k \exp(\delta_k \|x_i - x_{i-1}\|)$   
**if**  $p_{k^*} < \bar{p}_k$  **then**  
 $p_k, \bar{p}_k \leftarrow w_k \mathcal{N}(x_i | \mu_k, \Sigma_k)$   
 $\delta_k \leftarrow S_k \delta \|x_i - \mu_k\|_{\Sigma_k^{-1}}$   
**end if**  
**end for**  
 $c_{ik} \leftarrow \frac{p_k}{\sum_{k'=1}^K p_{k'}}$  **for all**  $k$ ;  $k^* \leftarrow \text{argmax}_k p_k$

**end for**

---

## 5 NTN SEARCH FOR SPARSE CODING

This section presents NTN search for locality-constrained linear (LLC) coding [30], which is a fast sparse coding algorithm proposed for image classification. In [30], a fast approximation algorithm for LLC using nearest neighbor search is also proposed. Here, our NTN-LLC replaces its nearest neighbor search step with NTN search. In the following, we first review the LLC and its approximation, and then explain our NTN-LLC algorithm.

Let  $\{\mu_k\}_{k=1}^K$  be a codebook. LLC assigns codewords to an input vector  $x_j$  by solving the following fitting problem

$$c_j = \text{argmin}_{c_j} \|x_j - Bc_j\|^2 + \lambda \|d_j \odot c_j\|^2 \text{ s.t. } \sum_k c_{jk} = 1 \quad (19)$$

where  $B = [\mu_1, \dots, \mu_K]$  is a codeword matrix,  $c_j \in \mathbb{R}^K$  is a code vector,  $c_{jk}$  is the  $k$ -th element of  $c_j$ ,  $\odot$  denotes the element-wise multiplication, and  $d_i \in \mathbb{R}^K$  is given by

$$d_{ik} = \exp\left(\frac{\|x_i - \mu_k\|}{\sigma}\right). \quad (20)$$

As shown in [30], since a code obtained by solving Eq.(19) has a significant value only for several nearest codewords of  $x_j$ , the fast approximation of LLC simply uses the  $M$  nearest codewords of  $x_j$  and solves the following simplified problem

$$c_j = \text{argmin}_{c_j} \|x_j - B_j c_j\|^2 \text{ s.t. } \sum_k c_{jk} = 1 \quad (21)$$

where  $B_j$  is a matrix of  $M$ -nearest codewords of  $x_j$  given by

$$B_j = [\mu_{\sigma_1}, \mu_{\sigma_2}, \dots, \mu_{\sigma_M}]. \quad (22)$$

Here,  $\sigma_1, \dots, \sigma_M$  are indices of the  $M$ -nearest codewords. Since the computational cost of approximated LLC is  $O(K + M^2)$ , we assume  $M \ll K$  in practice.

**Algorithm 3** NTN-LLC

---

**Input:** input vectors  $X$  ( $N = |X|$ ), parameter  $M$   
 codebook  $\{\mu_k\}_{k=1}^K$ , parameter  $\delta$ .

**Output:** code vectors  $\{c_i\}_{i=1}^N$

$x_1 \leftarrow \text{Rand}(X)$   
 $\underline{d}_k, \bar{d}_k \leftarrow \|x_1 - \mu_k\|$  **for all**  $k$   
 $B_1 \leftarrow [\mu_{\sigma_1}, \mu_{\sigma_2}, \dots, \mu_{\sigma_M}]$  by sorting  $\bar{d}_k$   
 $c_1 = \text{argmin}_{c_1} \|x_1 - B_1 c_1\|^2$  s.t.  $\sum_k c_{1k} = 1$   
 $k^* \leftarrow \text{argmax}_k c_{1k}$

**for**  $i = 2, \dots, N$  **do**  
 $x_i \leftarrow \text{argmin}_{x \in B(x_{i-1}) \cap \bar{X}} \|x - x_{i-1}\|$   
 $\underline{d}_{k^*}, \bar{d}_{k^*} \leftarrow \|x_i - \mu_{k^*}\|$   
**for all**  $k \neq k^*$  **do**  
 $\underline{d}_k \leftarrow \underline{d}_k - \delta \|x_i - x_{i-1}\|$   
 $\bar{d}_k \leftarrow \bar{d}_k + \delta \|x_i - x_{i-1}\|$   
**if**  $\underline{d}_{k^*} > \underline{d}_k$  **then**  
 $\underline{d}_k, \bar{d}_k \leftarrow \|x_i - \mu_k\|$   
**if**  $\bar{d}_{k^*} > \bar{d}_k$  **then**  $k^* \leftarrow k$  **end if**  
**end if**  
**end for**  
 $B_i \leftarrow [\mu_{\sigma_1}, \mu_{\sigma_2}, \dots, \mu_{\sigma_M}]$  by sorting  $\bar{d}_k$   
 $c_i = \text{argmin}_{c_i} \|x_i - B_i c_i\|^2$  s.t.  $\sum_k c_{ik} = 1$

**end for**

---

Our idea of NTN-LLC is to use approximate  $M$ -nearest codewords based on NTN search instead of the exact  $M$ -nearest codewords. The algorithm structure of NTN-LLC is the same as NTN-VQ, but it finds not only the nearest codeword but also  $M$ -nearest codewords by approximating distance between a codeword and an input vector by its upper bound.

The following is the NTN-LLC algorithm. The initialization step for  $j = 1$  uses the exact  $M$ -nearest neighbors to obtain a code in Eq. (21). The following three steps are iterated for  $j = 2, 3, \dots, N$ .

**(STEP 1: Select the next input vector)**

Select  $x_j$  in the same way as STEP 1 of NTN-VQ.

**(STEP 2: Calculate distance)**

Set  $k^* = \text{argmax}_k c_{j-1,k}$ .

2-1) Calculate distance  $d_{jk^*}$ .

2-2) For  $k = 1, 2, \dots, k^* - 1, k^* + 1, \dots, K$ , calculate a lower bound  $\underline{d}_{jk}$  and an upper bound  $\bar{d}_{jk}$  for  $d_{jk}$  as follows.

$$\underline{d}_{jk} = d_{ik} - \delta \Delta_{ij}, \quad (23)$$

$$\bar{d}_{jk} = d_{ik} + \delta \Delta_{ij}, \quad (24)$$

where  $i$  is the index of the input vector whose distance  $d_{ik}$  has been calculated,  $\delta$  is a parameter, and  $\Delta_{ij}$  is an accumulated distance from  $x_i$  to  $x_j$ . If  $\underline{d}_{jk} \geq d_{jk^*}$  then skip calculation of  $d_{jk}$ , otherwise calculate  $d_{jk}$ .

**(STEP 3: Output a code)**

Let

$$\tilde{B}_j = [\mu_{\sigma_1}, \mu_{\sigma_2}, \dots, \mu_{\sigma_M}]. \quad (25)$$

be a matrix of approximated  $M$ -nearest codewords of  $x_j$  in

**Algorithm 4** NTN-KCB

---

**Input:** input vectors  $X$  ( $N = |X|$ ),  
 codebook  $\{\mu_k\}_{k=1}^K$ , parameter  $\delta$ .  
**Output:** soft codes  $\{c_{ik}\}_{i=1}^N \{k=1}^K$   
 $x_1 \leftarrow \text{Rand}(X)$   
 $\mathcal{K}_k, \bar{\mathcal{K}}_k \leftarrow \mathcal{K}(x_1, \mu_k)$  **for all**  $k$   
 $c_{1k} \leftarrow \frac{\mathcal{K}_k}{\sum_{k'=1}^K \mathcal{K}_{k'}}$  **for all**  $k$ ;  $k^* \leftarrow \underset{k}{\text{argmax}} p_k$   
**for**  $i = 2, \dots, N$  **do**  
 $x_i \leftarrow \underset{x \in B(x_{i-1}) \cap \bar{X}}{\text{argmin}} \|x - x_{i-1}\|$   
 $\mathcal{K}_{k^*}, \bar{\mathcal{K}}_{k^*} \leftarrow \mathcal{K}(x_i, \mu_{k^*})$   
**for all**  $k \neq k^*$  **do**  
 $\bar{\mathcal{K}}_k \leftarrow \bar{\mathcal{K}}(x_i, \mu_k)$   
**if**  $\mathcal{K}_{k^*} < \bar{\mathcal{K}}_k$  **then**  
 $\mathcal{K}_k, \bar{\mathcal{K}}_k \leftarrow \mathcal{K}(x_i, \mu_k)$   
**end if**  
**end for**  
 $c_{ik} \leftarrow \frac{\mathcal{K}_k}{\sum_{k'=1}^K \mathcal{K}_{k'}}$  **for all**  $k$ ;  $k^* \leftarrow \underset{k}{\text{argmax}} \mathcal{K}_k$   
**end for**

---

terms of the following approximated distance

$$\tilde{d}_{jk} = \begin{cases} d_{jk} & k \in E \\ \bar{d}_{jk} & k \notin E. \end{cases} \quad (26)$$

Here,  $E$  is a set of indices of codewords whose distance to  $x_j$  is calculated in STEP 2.

Solve

$$c_j = \underset{c_j}{\text{argmin}} \|x_j - \tilde{B}_j c_j\|^2 \quad \text{s.t.} \quad \sum_k c_{jk} = 1 \quad (27)$$

to obtain a code vector.

NTN-LLC also has a parameter  $\delta$  as NTN-VQ in Sec.3.3 to control the trade-off between speed and accuracy: the number of exact nearest codewords increases as  $\delta$  increases, and the speed increases as  $\delta$  decreases. Note that, since exact distance is obtained in the STEP 3 for  $|E|$  codewords in Eq. (26), NTN-LLC returns the exact solution if  $|E| \geq M$ . Alg. 3 summarizes the NTN-LLC algorithm.

## 6 NTN SEARCH FOR A KERNEL CODEBOOK

The kernel codebook (KCB) [14] uses kernel density estimation to obtain a code of a image descriptor in the bag-of-visual-words framework. Here, we present an NTN search algorithm for KCB (NTN-KCB).

Let  $\{\mu_k\}_{k=1}^K$  be a codebook. A code obtained from KCB is given by

$$c_{jk} = \frac{\mathcal{K}(x_j, \mu_k)}{\sum_{k'=1}^K \mathcal{K}(x_j, \mu_{k'})}. \quad (28)$$

where  $\mathcal{K}(\cdot, \cdot)$  is a kernel function.

Since Eq. (28) replaces the probability  $p_{jk}$  in Eq. (11) in NTN-GMM by a kernel function, NTN-KCB utilizes the same algorithm structure as NTN-GMM, in which we replace  $p_{jk}$  and  $\bar{p}_{jk}$  by  $\mathcal{K}(x_j, \mu_k)$  and  $\bar{\mathcal{K}}(x_j, \mu_k)$ , respectively. Here,  $\bar{\mathcal{K}}(x_j, \mu_k)$  is an upper bound of the kernel function.

As in NTN-GMM (Eq. (15)), having the upper bound  $\bar{\mathcal{K}}(x_j, \mu_k)$ , which can be calculated from 1) the previous upper bound  $\bar{\mathcal{K}}(x_{j-1}, \mu_k)$  and 2) the distance between the current and the previous inputs  $\|x_j - x_{j-1}\|$  without any other computation with  $x_j$ , is effective to speed-up the NTN-KCB algorithm.

For instance, for the following Gaussian-shaped kernel

$$\mathcal{K}(x_j, \mu_k) = \exp\left(-\frac{\gamma}{2}\|x_j - \mu_k\|^2\right), \quad (29)$$

which is shown to be effective in [14], its upper bound is given by

$$\begin{aligned} \bar{\mathcal{K}}(x_j, \mu_k) &= \mathcal{K}(x_i, \mu_k) \exp(\delta\gamma^{-1}\|x_i - \mu_k\|\|x_i - x_j\|) \\ &= \bar{\mathcal{K}}(x_{j-1}, \mu_k) \exp(\delta\gamma^{-1}\|x_i - \mu_k\|\|x_j - x_{j-1}\|). \end{aligned} \quad (30)$$

where  $\gamma$  is a parameter of the kernel and  $\delta$  is the parameter to control the trade-off between speed and accuracy.

Alg. 4 summarizes the NTN-KCB algorithm.

## 7 EXPERIMENTAL EVALUATION

We perform image and video classification experiments on the PASCAL VOC 2007 Classification Challenge [56] and the TRECVID 2010 Semantic Indexing Task, respectively.

### 7.1 Experiments on PASCAL VOC 2007

#### 7.1.1 Experimental setup

The dataset of PASCAL VOC 2007 consists of 9,963 images, which are divided into a training set (5011 images) and a testing set (4952 images). We use Mean Average Precision (Mean AP) over the 20 object categories for evaluating classification accuracies.

We implement 1) NTN-VQ (Alg. 1) with histogram representation and super-vector (SV) representation [21], 2) NTN-GMM (Alg. 2) with Fisher-vector (FV) representation [22], 3) NTN-LLC (Alg. 3) with LLC representation [30], and 4) NTN-KCB (Alg. 4) with Histogram representation. The definition of each representation is summarized in Appendix.

We compare our NTN methods with standard VQ, GMM, LLC and KCB, and tree-based ANN-VQ, RF-VQ and Tree-GMM. The ANN-VQ uses a fast library for approximate nearest neighbor (ANN) search [20], [52]. The RF-VQ uses random forests used in [38], [39]. The Tree-GMM [16] is an extension of the hierarchical  $k$ -means to a GMM framework. In addition, NTN-LM-GMM applies the LM approximation to NTN-GMM. The parameter  $\delta$  for NTN methods and parameters for random forests and kernel codebooks are optimized on the validation set, where half of training images are used for training models and others are used for validating the models.

The following are our standard settings in all experiments.  $2 \times 2$  SIFT descriptors are extracted from every 4 pixels at 5 scales. We set a set of neighbor vectors  $B(x)$  to a set of the four SIFT descriptors adjacent to a descriptor  $x$ . The averaged number of descriptors per image is 49580. We omit Gaussian weighting for SIFT descriptors. A codebook is trained on randomly sampled 1 million descriptors by using the  $k$ -means



Method	Representation	$\delta$	Mean AP	$ E $	Time (msec)	Reduction rate (%)
VQ	SV	-	* <b>0.568</b>	512.0	856.2	0.0
ANN-VQ [20]	SV	-	0.563	-	475.7	44.4
RF-VQ [38], [39]	SV	-	0.559	-	292.6	65.8
NTN-VQ	SV	0.20	0.563	<b>57.8</b>	<b>193.2</b>	<b>77.4</b>
GMM	FV	-	* <b>0.582</b>	512.0	2595.7	0.0
Tree-GMM [16]	FV	-	0.582	295.2	1496.8	42.3
NTN-GMM	FV	0.09	0.580	88.0	642.0	75.3
NTN-LM-GMM	FV	0.09	0.579	<b>88.0</b>	<b>276.8</b>	<b>89.3</b>
VQ	Hist	-	*0.404	512.0	856.2	0.0
NTN-VQ	Hist	0.20	<b>0.405</b>	<b>57.8</b>	<b>193.2</b>	<b>77.4</b>
LLC	LLC	-	* <b>0.439</b>	-	2428.8	0.0
NTN-LLC	LLC	0.50	0.429	-	<b>688.8</b>	<b>71.6</b>
KCB	Hist	-	* <b>0.469</b>	512.0	2757.1	0.0
NTN-KCB	Hist	0.20	<b>0.469</b>	<b>214.9</b>	<b>681.8</b>	<b>75.3</b>

TABLE 2

**Speed comparison at the fixed accuracy level.** VQ: standard hard vector quantization (VQ), ANN-VQ: approximate nearest neighbor search [20], RF-VQ: random forests [38], [39], NTN-VQ: our neighbor-to-neighbor (NTN) search for VQ (Alg. 1), GMM: standard Gaussian mixture model (GMM), Tree-GMM: an extension of the hierarchical  $k$ -means to a GMM framework in [16] NTN-GMM: NTN search for a GMM (Alg. 2), NTN-LM-GMM: NTN-GMM with log-max approximation, LLC: Locality-constrained linear (LLC) coding [30], NTN-LLC: NTN search for LLC (Alg. 3), KCB: Kernel codebook (KCB) [14], NTN-KCB: NTN search for KCB (Alg. 4). SV: Super-vector representation, FV: Fisher-vector representation, Hist: Histogram representation.  $\delta$ : a parameter of our NTN methods, Mean AP: image classification accuracy on the testing set of the PASCAL VOC 2007 Classification Challenge.  $|E|$ : the number of distance or probability calculations per input vector, Time: coding time in msec, Reduction rate: reduction rate of the coding cost. Note that there are no statistically significant differences in Mean AP between the method marked “\*” and each other method in the same split table on randomization test ( $p < 0.05$ ).

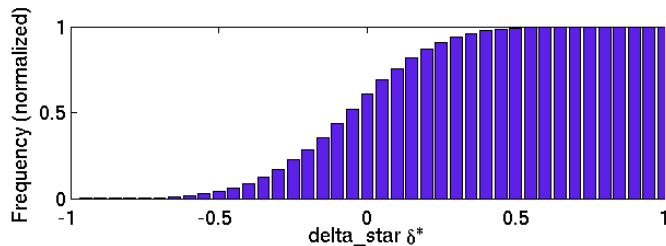


Fig. 5. **Cumulative histogram of  $\delta^*$ .** Statistics of the true  $\delta^*$  in Eq. (8) on PASCAL VOC 2007 training images is reported for NTN-VQ.

algorithm or the EM algorithm. Covariance matrices for a GMM are assumed to be diagonal. The codebook size is set to 512. A one-vs-rest linear SVM is used for a classifier for each of 20 object categories, where the regularization parameter is fixed to 1.0. A single core of a 2.93 GHz Intel Xeon CPU with an 8 GB memory is used for measuring computational costs. Note that some influence on changing these settings is reported in the following experimental results.

7.1.2 Speed of coding

In Table 2, we compare speed of coding at the fixed accuracy level. Overall, our NTN methods are faster than the others while keeping the classification accuracy. For example, we observe that NTN-VQ and NTN-LM-GMM reduce the coding

cost by 77.4% and by 89.3%, respectively. It is also confirmed that FV representation and SV representation perform significantly better than the others as reported in [52]. Note that there are no significant differences in Mean AP on randomization test ( $p < 0.05$ ) between methods in the same split table in Table 2.

Here, a parameter  $\delta$  is optimized on the validation set, where half of training images are used for training models and others are used for validating the models. As described in Subsec. 3.3, the restriction of  $\delta \geq \delta^*$  is relaxed in our methods. However, more than 90% of  $d_{jk}$  gives a correct lower bound when  $\delta = 0.20$  for NTN-VQ as shown in Figure 5. Note that 61.3% of two adjacent descriptors have the same visual word as shown in Figure 6.

Figure 7 shows the speed-accuracy trade-off for NTN methods for different values of  $\delta$ . We observe that NTN-LM-GMM outperforms NTN-GMM and NTN-VQ in terms of both speed and accuracy. This is because NTN-LM-GMM has the advantages of both of NTN-VQ and NTN-GMM: it only requires distance calculations without using an  $exp$  operator as NTN-VQ, but it has a weight coefficient and a covariance matrix for each codeword as NTN-GMM.

Compared with tree-based methods, a disadvantage of NTN methods is that they are not very effective if neighbor vectors are not similar to each other. We confirm this in Figure 8: tree-based ANN-VQ and RF-VQ perform better than RAND-



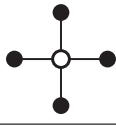

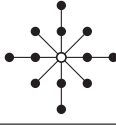
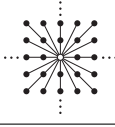
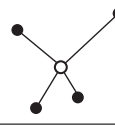
	(a)	(b)	(c)	(d)	(e)	(f)	(g)
Neighbors $B(x)$							
$ B(x) $	1	2	4	8	12	ALL	4 (random)
$\min \Delta(x)$	173.6	154.2	132.8	132.7	131.1	120.8	443.9
Time (msec)	223.7	210.3	193.2	195.0	201.0	12330.6	476.5
Reduction (%)	73.9	75.4	<b>77.4</b>	77.2	76.5	-	44.4

TABLE 3

**Speed comparison using different types of neighbor vectors on PASCAL VOC 2007.** The row of “Neighbors  $B(x)$ ” illustrates the definition of neighbor vectors. A white node and black nodes denote a point on a grid for a descriptor  $x$  and its neighbor points, respectively. (a) to (e) use points adjacent to  $x$ , (f) uses all grid points on an image, (g) uses randomly sampled 4 points.  $|B(x)|$ : the number of neighbor vectors,  $\min \Delta(x)$ : the minimum distance between  $x$  and its neighbor vector, Time: coding time in msec, Reduction: reduction rate of the coding cost by NTN-VQ compared with the standard VQ (856.2 msec).

Stride $S$	Mean AP	Number of descriptors	Time using GMM (msec)	Time using NTN-LM-GMM (msec)	Reduction (%)
7	0.557	14438	785.5	80.2	<b>89.8</b>
6	0.565	20412	1100.7	112.5	<b>89.8</b>
5	0.574	30653	1630.2	170.5	89.5
4	<b>0.582</b>	49580	2595.7	276.8	89.3
3	<b>0.582</b>	91284	4785.8	527.0	89.0

TABLE 4

**Accuracy and speed comparison using different strides in dense sampling.** Stride  $S$ : the density of sampling (every  $S$  pixels), Mean AP: Mean AP on the PASCAL VOC 2007 Classification Challenge, Number of descriptors: the averaged number of sampled descriptors per image, Time (msec): coding time using GMM/NTN-LM-GMM in msec, GMM: standard Gaussian mixture model (GMM), NTN-LM-GMM: NTN search for a GMM with log-max approximation, Reduction (%): reduction rate of the coding cost.

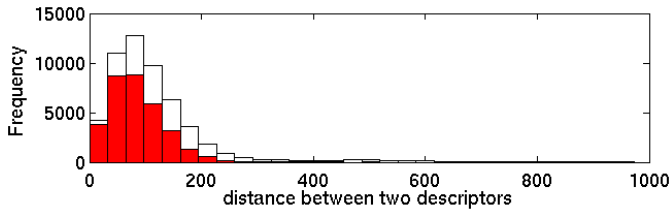


Fig. 6. **A histogram of descriptors.** Red bars: descriptors that have the same visual word as a neighbor descriptor. White bars: all descriptors. SIFT descriptors are extracted from every 4 pixels at 5 scales on the PASCAL VOC 2007 training images. The codebook size is 512. 61.3% of two adjacent descriptors have the same visual word.

VQ which replaces a neighbor vector by a randomly sampled vector in each iteration of NTN-VQ. Notably, the average distance between two neighbor descriptors ( $x_{j-1}$  and  $x_j$ ) is 132.8 and 507.7 for NTN-VQ and RAND-VQ, respectively. This confirms that the assumption that neighbor vectors are similar to each other, is necessary for NTN methods.

In addition, we confirm the necessity of the accumulated distance in Eq. (6) by replacing it with direct distance, i.e.,

$\Delta_{ij} = \|x_i - x_j\|$  in Figure 9. If we ignore computation time for  $\Delta_{ij}$ , for example in the case where a distance matrix on input vectors is pre-computed in some way, the direct distance is better than the accumulated distance. In general, the accumulated distance is computationally effective since it derives efficient update rules of a lower/upper bound in Eq. (9) and (15).

### 7.1.3 Definition of neighbor vectors

Table 3 shows reduction rates of the coding cost using different types of neighbor vectors  $B(x)$ . As can be seen, using the four adjacent points on a grid gives the best reduction rate. Since it is needed to calculate distance between  $x$  and each neighbor vector in  $B(x)$  in STEP 1 of NTN-VQ, the number of neighbor vectors is better to be small rather than large. For example, if we set  $B(x)$  to all the other descriptors than  $x$  as Table 3 (f), the computational time for the STEP 1 becomes longer than that for applying the standard VQ. We conclude that looking at four neighbors is a reasonable definition for NTN search on densely sampled image descriptors.

### 7.1.4 Sampling density

Table 4 compares results using different strides to extract SIFT descriptors. A short stride improves the accuracy as can be seen and the improvement saturates at the stride of 4, which

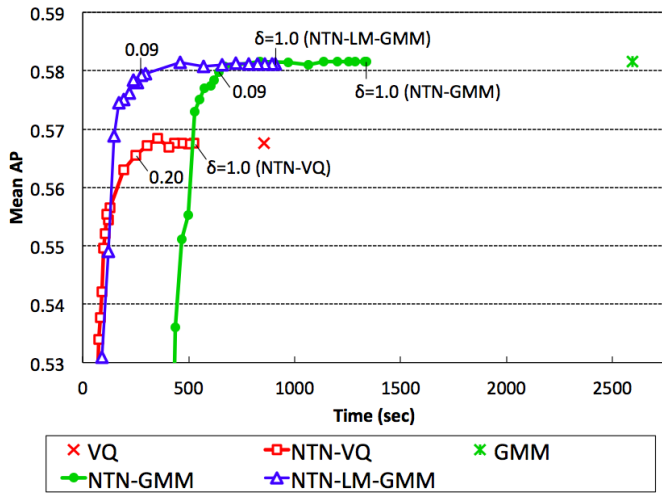


Fig. 7. **Speed-accuracy trade-off for different values of  $\delta$ .** Trade-off between coding time and Mean AP is reported. All plots are for  $\delta = 1.0, 0.9, \dots, 0.1, 0.09, \dots, 0.01$ . VQ: standard hard vector quantization (VQ), NTN-VQ: neighbor-to-neighbor (NTN) search for VQ, GMM: standard Gaussian mixture model (GMM), NTN-GMM: NTN search for a GMM, NTN-LM-GMM: NTN-GMM with the log-max approximation.

is the standard setting in our experiments. It is also confirmed that the reduction rate of the coding cost obtained by using NTN-LM-GMM is stable for the all strides.

### 7.1.5 Result examples

We examine the effectiveness of NTN-VQ for several different images in Figure 10. The reduction rate of the coding cost by NTN-VQ is 84.9% for the image (a) and 66.8% for the image (h). Here, (a) and (h) are the best and the worst cases on PASCAL VOC 2007, respectively. This shows that NTN methods are more effective for images which can be segmented into several uniform regions. Notably, NTN-VQ is still better than ANN-VQ even in the worst case.

### 7.1.6 Codebook size

The simplest idea to reduce the coding cost is to reduce the codebook size. In Figure 11, which shows the speed-accuracy trade-off for different codebook sizes, we confirm that using NTN methods is better than reducing the codebook size. This also confirms that NTN-LM-GMM is the best in both speed and accuracy. Note that there is no significant difference in Mean AP between a standard method and a NTN methods for each codebook size  $K = 2048, 1024, \dots, 16$ .

### 7.1.7 Relative computational time in a pipeline

Figure 12 shows the relative cost of coding with respect to the cost of the other steps of processing pipeline for extracting SV and FV representation. As can be seen, the coding step is the majority of the whole processing pipeline: 85.3% and 88.4% of computational time are occupied from it in FV and SV, respectively. NTN-VQ, NTN-GMM and NTN-LM-GMM reduces the total computational cost by 66.0%, 66.5%, and 85.3%, respectively. Note that the cost of pooling in FV, which generate a final FV representation, is also reduced by the LM

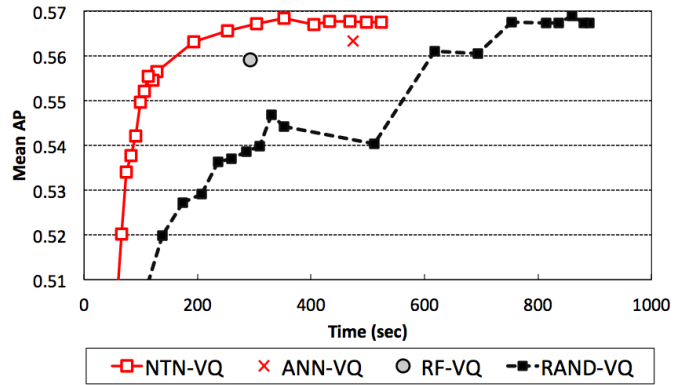


Fig. 8. **Comparison with RAND-VQ.** Trade-off between coding time and Mean AP is reported. NTN-VQ: neighbor-to-neighbor (NTN) search for VQ, this is the same plot as Figure 7, ANN-VQ: approximate nearest neighbor search [20], RF-VQ: random forests [38], [39], RAND-VQ: NTN-VQ in which a neighbor vector is replaced by a randomly sampled vector.

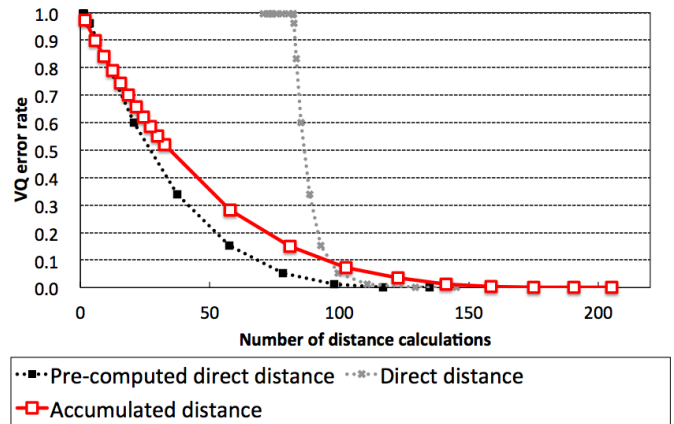


Fig. 9. **Comparison of the accumulated distance and the direct distance.** VQ error rate in NTN-VQ for different values of  $\delta$  is reported. All plots are for  $\delta = 1.0, 0.9, \dots, 0.1, 0.09, \dots, 0.01$ . Accumulated distance:  $\Delta_{ij}$  is defined by Eq. (6). Direct distance:  $\Delta_{ij}$  is replaced by the direct distance  $\|x_i - x_j\|$ . Pre-computed direct distance: the direct distance is used but distance calculations for it are not counted.

approximation since we can skip some summations in pooling if  $c_{ik}$  is equal to zero.

Here we consider the costs to extract image representation. However, for large-scale image classification, we should consider the SVM-classification cost, which is negligible in our experiments on PASCAL VOC with 20 categories. To reduce the SVM-classification cost, applying dimension reduction techniques such as product quantization [42] to the final image representation can be effectively utilized.



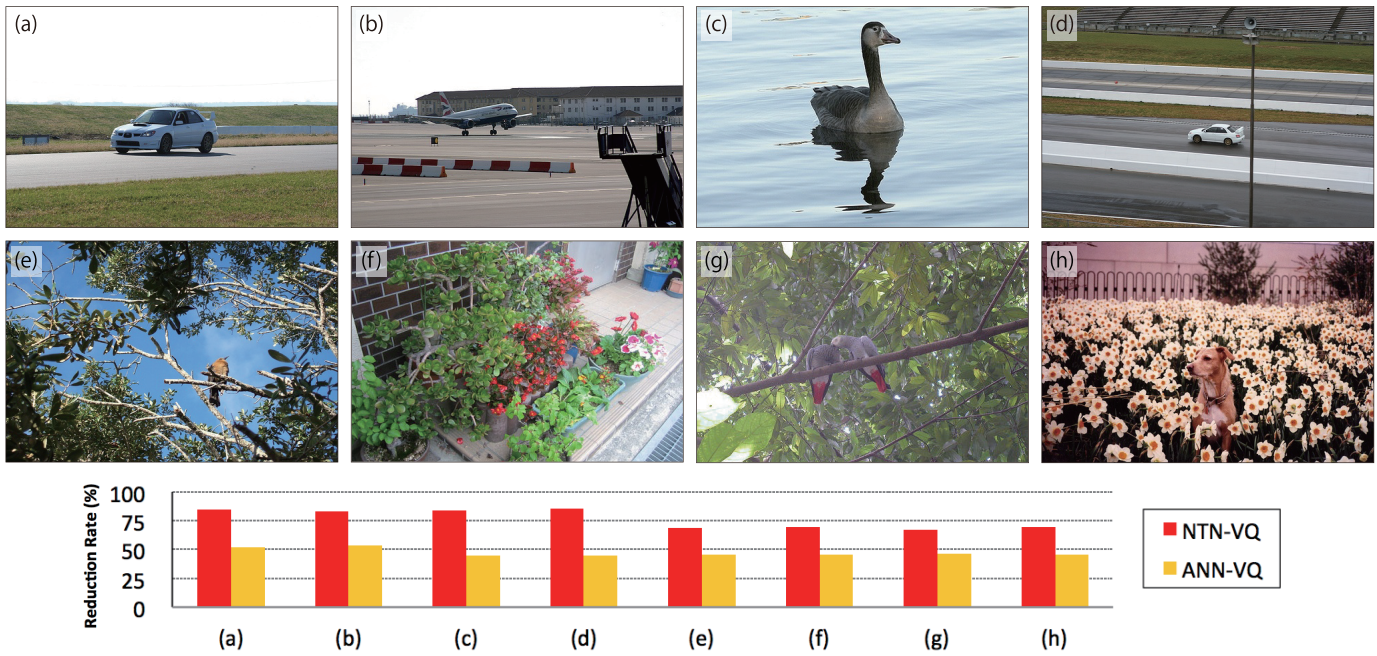


Fig. 10. The computational cost reduction by NTN-VQ for different images. Eight images are from PASCAL VOC 2007. The reduction rate of the coding cost by NTN-VQ and ANN-VQ for each image is reported.

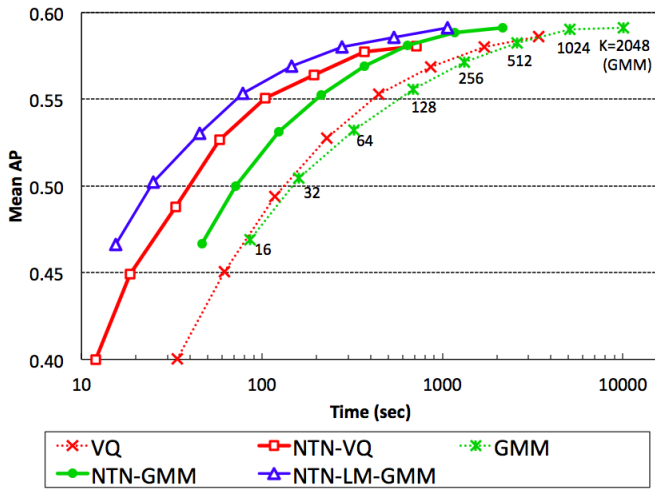


Fig. 11. Speed-accuracy trade-off for different codebook sizes. Trade-off between coding time and Mean AP for codebook sizes of  $K = 2048, 1024, 512, \dots, 16$ . is reported. VQ: standard hard vector quantization (VQ), NTN-VQ: neighbor-to-neighbor (NTN) search for VQ  $\delta = 0.20$ , GMM: standard Gaussian mixture model (GMM), NTN-GMM: NTN search for a GMM  $\delta = 0.09$ , NTN-LM-GMM: NTN-GMM with the log-max approximation  $\delta = 0.09$ .

## 7.2 Experiments on TRECVID 2010

### 7.2.1 Experimental setup

The dataset of the TRECVID 2010 Semantic Indexing Task consists of 400 hours of Internet archive videos with creative commons licenses. Shot boundaries are automatically detected and provided with video data. The task is to detect 30 semantic

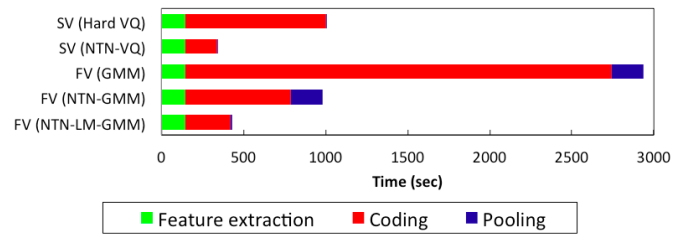


Fig. 12. Relative computational cost. Computational cost for each step to extract super-vector (SV) representation and Fisher-vector (FV) representation is reported. The codebook size is 512. Feature extraction: SIFT descriptors are extracted from every 4 pixels at 5 scales, Coding: each descriptor is assigned to codeword(s), Pooling: an SV or FV image representation is generated. 85.3%, 56.6%, 88.4%, 65.4% and 64.2% of computational time is occupied from coding by VQ, NTN-VQ, GMM, NTN-GMM, and NTN-LM-GMM, respectively. Total computational cost is reduced by 66.0%, 66.5% and 85.3% by NTN-VQ, NTN-GMM, and NTN-LM-GMM, respectively.

concepts including objects, events, and scenes from video shots. The number of video shots is 119,685 for training, and 146,788 for testing. We use Mean Average Precision (Mean AP) over the 30 semantic concepts for evaluating detection accuracies. The AP is estimated by using a method called inferred average precision, which is the official evaluation measure of the TRECVID 2010 Semantic Indexing Task.

We implement NTN-GMM (Alg. 2) and NTN-LM-GMM with GMM-supervector (GS) representation, which has shown to be effective for video semantic indexing in [54]. See Appendix for the definition of the GS representation.

Method	Mean AP	Time (msec)	Reduction (%)
GMM	*0.0887	18088.7	0.0
NTN-GMM	<b>0.0893</b>	3484.5	80.7
NTN-LM-GMM	0.0881	<b>1542.9</b>	<b>91.5</b>

TABLE 5

**Speed comparison on TRECVID 2010.** GMM: standard Gaussian mixture model (GMM), NTN-GMM: NTN search for a GMM (Alg. 2,  $\delta = 0.09$ ), NTN-LM-GMM: NTN-GMM with log-max approximation  $\delta = 0.09$ .

We extract the same image descriptors as the PASCAL VOC 2007 from at most 100 image frames per a video shot. A GS representation is obtained from all descriptors in a video shot. The other experimental conditions are the same as the PASCAL VOC 2007.

### 7.2.2 Speed of coding

From the TRECVID dataset, comparable results with those on the PASCAL VOC dataset (Subsec 7.1.2) are obtained as shown in Table 5, which compares speed of coding using GMM, NTN-GMM, and NTN-LM-GMM. The reduction rate of the coding cost is 80.7% and 91.5% for NTN-GMM and NTN-LM-GMM, respectively. While NTN-GMM or NTN-LM-GMM works slightly better than the standard GMM in terms of accuracy for some semantic concepts, there is no statistically significant difference between them (randomization test,  $p < 0.05$ ). We conclude that our NTN algorithm effectively reduces the computational cost without any significant degradation.

### 7.2.3 Definition of neighbor vectors

Since video has two spatial and one time dimensions, we extend the definitions of neighbor vectors in Table 3 to those in the three dimensional space in Table 6. The best choice is to use the six points adjacent to a point for a descriptor  $x$  as a set of neighbor vectors  $B(x)$ , since two descriptors on adjacent points in the time axis are often very similar to each other if they are extracted from a static background in video.

## 8 CONCLUSION

We have proposed a fast computation method for searching for the matches, neighbor-to-neighbor (NTN) search, and its applications to vector quantization (VQ), a Gaussian mixture model (GMM), sparse coding, and a kernel codebook. Our experiments on the PASCAL VOC 2007 classification challenge showed that NTN-VQ and NTN-LM-GMM reduced the coding cost by 77.4%, and 89.3%, respectively, without any significant degradation in the image classification performance. We also confirmed the effectiveness of the NTN search on the TRECVID 2010 Semantic Indexing Task.

In future work, we will focus on applications of NTN search to deep architectures such as deep convolutional neural networks, and deep learning using Fisher vectors. Approximation of deep learned features densely sampled in convolutional layers would be interesting as a promising next step.

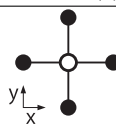
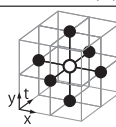
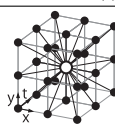
	(c)	(h)	(i)
Neighbors $B(x)$			
$ B(x) $	4	6	26
$\Delta_{i,i+1}$	116.2	82.0	101.5
Time (msec)	1542.9	1267.3	1512.9
Reduction (%)	91.5	<b>93.0</b>	91.6

TABLE 6

**Speed comparison using different types of neighbor vectors on TRECVID 2010.** The row of “Neighbors  $B(x)$ ” illustrates the definition of neighbor vectors. A white node and black nodes denote a point on a grid for a descriptor  $x$  and its neighbor points, respectively. (c) uses four points adjacent to  $x$  which is the best choice in Table 3, (h) uses six points adjacent to  $x$  in a spatial-temporal space on video data, (i) uses twenty six points adjacent to  $x$ .  $|B(x)|$ : the number of neighbor vectors,  $\min \Delta(x)$ : the minimum distance between  $x$  and its neighbor vector, Time: coding time in msec, Reduction: reduction rate of the coding cost by NTN-LM-GMM.

## REFERENCES

- [1] N. Inoue and K. Shinoda. Neighbor-to-neighbor search for fast coding of feature vectors. *Proc. ICCV*, pp. 1233–1240, 2013. 1, 2
- [2] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. *Proc. ECCV SLCV workshop*, pp. 59–74, 2004. 1, 2
- [3] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *In IJCV*, vol. 60(2), pp. 91–110, 2004. 1, 2
- [4] H. Wang, A. Klaser, C. Schmid, and C.-L. Liu. Dense trajectories and motion boundary descriptors for action recognition. *In IJCV*, vol. 103(1), pp. 60–79, 2013. 1
- [5] A. Habibiyan, T. Mensink, and C.G.M. Snoek. VideoStory: a new multimedia embedding for few-example recognition and translation of events. *Proc. ACM Multimedia*, 2014. 1
- [6] C. Gao, D. Meng, W. Tong, Y. Yang, Y. Cai, H. Shen, G. Liu, S. Xu, and A. G. Hauptmann. Interactive surveillance event detection through mid-level discriminative representation. *Proc. ICMR*, 2014. 1
- [7] C. Ellis, S.Z. Masood, M.F. Tappen, J.J. Laviola, and R. Sukthankar. Exploring the trade-off between accuracy and observational latency in action recognition. *In IJCV*, vol. 101(3), pp. 420–436, 2013. 1
- [8] J. Luo, W. Wang, and H. Qi. Group sparsity and geometry constrained dictionary learning for action recognition from depth maps. *Proc. ICCV*, 2013. 1
- [9] M. Ovsjanikov, W. Li, L. Guibas, and N. J. Mitra. Exploration of continuous variability in collections of 3d shapes. *ACM Trans. Graph.*, vol. 30(4), pp. 1–10, 2011. 1
- [10] R. Xu, and D. Wunsch II. Survey of clustering algorithms. *IEEE Trans. on Neural Networks*, vol. 16(3), pp. 645–678, 2005. 1
- [11] C.G.M. Snoek, and M. Worring. Concept-based video retrieval. *Foundations and Trends in Information Retrieval*, 2009. 1
- [12] A. Gersho, and R. M. Gray. Vector quantization and signal compression. Kluwer Academic Publishers, 1992. 1, 2
- [13] S. P. Lloyd. Least squares quantization in PCM. *IEEE Trans. on Information Theory*, vol. 28(2), pp. 129–137, 1982. 1, 2
- [14] J. C. V. Gemert, J.-m. Geusebroek, C. J. Veenman, and A. W. M. Smeulders. Kernel codebooks for scene categorization. *Proc. ECCV*, pp. 696–709, 2008. 1, 2, 7, 8
- [15] F. Perronnin, C. Dance, G. Csurka, and M. Bressan. Adapted vocabularies for generic visual categorization. *Proc. ECCV*, pp. 464–475, 2006. 1, 2
- [16] N. Inoue and K. Shinoda. A fast map adaptation technique for GMM-supervector-based video semantic indexing systems. *Proc. ACM Multimedia*, pp. 1357–1360, 2011. 1, 2, 7, 8
- [17] N. Inoue, and K. Shinoda. q-Gaussian mixture models for image and



video semantic indexing. *Elsevier JVCI*, vol. 24(8), pp. 1450–1457, 2013. [1](#), [4](#)

[18] J. S. Beis and D. G. Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. *Proc. CVPR*, pp. 1000–1006, 1997. [1](#)

[19] C. Silpa-anan and R. Hartley. Optimised kd-trees for fast image descriptor matching. *Proc. CVPR*, pp. 1–8, 2008. [1](#), [2](#)

[20] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *Proc. VISAPP*, pp. 331–340, 2009. [1](#), [2](#), [7](#), [8](#), [10](#)

[21] X. Zhou, K. Yu, T. Zhang, and T. S. Huang. Image classification using super-vector coding of local image descriptors. *Proc. ECCV*, pp. 141–154, 2010. [1](#), [2](#), [3](#), [7](#), [14](#)

[22] F. Perronnin, S. Jorge, and T. Mensink. Improving the fisher kernel for large-scale image classification. *Proc. ECCV*, pp. 143–156, 2010. [1](#), [2](#), [3](#), [4](#), [7](#), [14](#)

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. *Proc. NIPS*, 2010. [1](#)

[24] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep Fisher networks for large-scale image classification. *Proc. NIPS*, 2013. [1](#)

[25] V. Sydorov, M. Sakurada, and C.H. Lampert. Deep Fisher kernels – end to end learning of the Fisher kernel GMM parameters. *Proc. CVPR*, 2014. [1](#), [2](#)

[26] Y. Gong, L. Wang, R. Guo, and S. Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. *Proc. ECCV*, 2014. [1](#)

[27] J. Y. Ng, F. Yang, and L. S. Davis. Exploiting local features from deep networks for image retrieval. *Proc. CVPR workshop on Deep Vision*, 2015. [1](#)

[28] F. Perronnin, and D. Larlus. Fisher vectors meet neural networks: a hybrid classification architecture. *Proc. CVPR*, 2015. [1](#), [2](#)

[29] C.G.M. Snoek, K.E.A. van de Sande, D. Fontijne, S. Cappallo, J. van Gemert, A. Habibiyan, T. Mensink, P. Mettes, R. Tao, D.C. Koelma, and A.W.M. Smeulders. Video concept detection by deep nets with FLAIR (Mediamill at TRECVID 2014: searching concepts, objects, instances and events in video). *Proc. TRECVID workshop*, 2014. [2](#)

[30] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. *Proc. CVPR*, pp. 3360–3367, 2010. [2](#), [6](#), [7](#), [8](#)

[31] J. H. Freidman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *In ACM Trans. on Mathematical Software*, vol. 3(3), pp. 209–226, 1977. [2](#)

[32] R. F. Sproull. Refinements to nearest-neighbor searching in k-dimensional trees. *In Algorithmica*, vol. 6(1), pp. 579–589, 1991. [2](#)

[33] P. Ciaccia, M. Patella, and P. Zezula. M-tree: an efficient access method for similarity search in metric spaces. *Proc. VLDB*, 1997. [2](#)

[34] J. Uhlmann. Satisfying general proximity/similarity queries with metric trees. *In Elsevier Information Processing Letters*, vol.40(4), pp.175–179, 1991. [2](#)

[35] S. M. Omohundro. Efficient algorithms with neural network behavior. *In Complex Systems*, vol.1(2), pp.273–347, 1987. [2](#)

[36] S. M. Omohundro. Five balltree construction algorithms. *ICSI Technical Report*, TR-89-063, 1989. [2](#)

[37] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. *Proc. CVPR*, pp. 2161–2168, 2006. [2](#)

[38] F. Moosmann, E. Nowak, and F. Jurie. Randomized clustering forests for image classification. *IEEE Trans. on PAMI*, vol. 30(9), pp. 1632–1646, 2008. [2](#), [7](#), [8](#), [10](#)

[39] J. R. R. Uijlings, A. W. M. Smeulders, and R. J. H. Scha. Real-Time Visual Concept Classification. *IEEE Trans. on Multimedia*, vol. 12(7), pp.665–681, 2010. [2](#), [7](#), [8](#), [10](#)

[40] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, vol. 24(6), pp. 417–441, 1933. [2](#)

[41] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics Journal*, vol. 7(7), pp. 179–188, 1936. [2](#)

[42] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE Trans. on PAMI*, vol. 33(1), pp. 117–128, 2011. [2](#), [10](#)

[43] T. Huang. Linear spatial pyramid matching using sparse coding for image classification. *Proc. CVPR*, pp. 1794–1801, 2009. [2](#)

[44] H. Bristow, A. Eriksson, and S. Lucey. Fast convolutional sparse coding. *Proc. CVPR*, pp. 391–398, 2013. [2](#)

[45] T. Guha and R. Ward. Learning sparse representations for human action recognition. *IEEE Trans. on PAMI*, vol. 34(8), pp. 1576–88, 2012. [2](#)

[46] X. Zhao, X. Li, C. Pang, X. Zhu, and Q. Z. Sheng. Online human gesture recognition from motion data streams. *Proc. ACM Multimedia*, pp. 23–32, 2013. [2](#)

[47] N. Dalal and W. Triggs. Histograms of oriented gradients for human detection. *Proc. CVPR*, pp. 886–893, 2004. [2](#)

[48] K. van de Sande, T. Gevers, and C. Snoek. Evaluating color descriptors

for object and scene recognition. *IEEE Trans. on PAMI*, vol. 32(9), pp. 1582–1596, 2010. [2](#)

[49] X. Wang, T. X. Han, and S. Yan. An HOG-LBP human detector with partial occlusion handling. *Proc. ICCV*, pp. 32–39, 2009. [2](#)

[50] T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. *Proc. NIPS*, pp. 487–493, 1998. [2](#)

[51] F. Perronnin and et al. Fisher kernels on visual vocabularies for image categorization. *Proc. CVPR*, 2007. [2](#)

[52] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. *Proc. BMVC*, pp. 1–12, 2011. [3](#), [7](#), [8](#)

[53] C.G.M. Snoek, K.E.A. van de Sande, D. Fontijne, A. Habibiyan, M. Jain, S. Kordumova, Z. Li, M. Mazloom, S.L. Pintea, R. Tao, D.C. Koelma, and A.W.M. Smeulders. The mediamill at trecvid 2013: searching concepts, objects, instances and events in video. *Proc. TRECVID workshop*, 2013. [3](#)

[54] N. Inoue, Y. Kamishima, T. Wada, K. Shinoda, and S. Sato. Semantic indexing using gmm supervectors and tree-structured GMMs (TokyoTech+Canon at TRECVID 2011). *Proc. TRECVID workshop*, 2011. [3](#), [11](#)

[55] W. M. Campbell, D. E. Sturim, and D. A. Reynolds. Support vector machines using gmm supervectors for speaker verification. *IEEE Signal Processing Letters*, vol. 13, pp. 308–311, 2006. [3](#), [14](#)

[56] S. M. A. Eslami, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The Pascal visual object classes challenge: a retrospective. *In IJCV*, vol. 111(1), pp. 98–136, 2014. [7](#)

## APPENDIX

### Upper bound of a Gaussian probability

The upper bound  $\bar{p}_{jk}$  of the probability (Eq. (13)) is delivered as follows. The law of cosines gives

$$\begin{aligned} \exists \delta^* \in [-1, 1] \text{ s.t. } \|x_j - \mu_k\|_{\Sigma_k^{-1}}^2 & \quad (32) \\ = \|x_i - \mu_k\|_{\Sigma_k^{-1}}^2 + \|x_i - x_j\|_{\Sigma_k^{-1}}^2 - 2\delta^* \|x_i - x_j\|_{\Sigma_k^{-1}} \|x_i - \mu_k\|_{\Sigma_k^{-1}}. \end{aligned}$$

For  $\delta \geq \max(\delta^*, 0)$ , it implies

$$\begin{aligned} \|x_j - \mu_k\|_{\Sigma_k^{-1}}^2 & \geq \|x_i - \mu_k\|_{\Sigma_k^{-1}}^2 - 2S_k\delta \|x_i - x_j\|_{\Sigma_k^{-1}} \|x_i - \mu_k\|_{\Sigma_k^{-1}} \\ & \geq \|x_i - \mu_k\|_{\Sigma_k^{-1}}^2 - 2S_k\delta\Delta_{ij} \|x_i - \mu_k\|_{\Sigma_k^{-1}}, \end{aligned} \quad (33)$$

where  $S_k$  is the square root of the spectral radius of  $\Sigma_k^{-1}$  and  $\Delta_{ij}$  is the accumulated distance given by Eq. (6). Thus, we have

$$p_{jk} = \frac{w_k}{Z_k} \exp\left(-\frac{1}{2}\|x_j - \mu_k\|_{\Sigma_k^{-1}}^2\right) \quad (34)$$

$$\leq \frac{w_k}{Z_k} \exp\left(-\frac{1}{2}\|x_i - \mu_k\|_{\Sigma_k^{-1}}^2 + S_k\delta\Delta_{ij} \|x_i - \mu_k\|_{\Sigma_k^{-1}}\right) \quad (35)$$

$$= p_{ik} \exp\left(S_k\delta\Delta_{ij} \|x_i - \mu_k\|_{\Sigma_k^{-1}}\right) \quad (36)$$

$$= p_{ik} \exp(\delta_{ik}\Delta_{ij}) = \bar{p}_{jk}, \quad (37)$$

where  $Z_k = (2\pi)^{\frac{d}{2}} |\Sigma_k|^{-\frac{1}{2}}$  and  $\delta_{ik}$  is given in Eq.(14).

### Image representations

Let  $X = \{x_i\}_{i=1}^N$  be a set of  $d$ -dimensional input vectors. Here we review the definition of each representation method used in the experiments. In the following,  $\{\mu_k\}_{k=1}^K$  and  $\{w_k, \mu_k, \Sigma_k\}$  denote codebooks for VQ and a GMM, respectively.

### Histogram representation

Histogram representation is a histogram of quantized input vectors given by

$$\phi_{\text{Hst}}(X) = \begin{pmatrix} C_1 \\ C_2 \\ \vdots \\ C_K \end{pmatrix}. \quad (38)$$

Here  $C_k$  is a count for the  $k$ -th visual word given by

$$C_k = \sum_{i=1}^N c_{ik}, \quad (39)$$

where

$$c_{ik} = \begin{cases} 1, & \text{if } k = \underset{k}{\operatorname{argmin}} \|x_i - \mu_k\|, \\ 0, & \text{otherwise.} \end{cases} \quad (40)$$

The dimension of the Histogram representation is  $K$ .

### LLC coding

LLC coding, which is obtained by max-pooling, is given by

$$\phi_{\text{LLC}}(X) = \begin{pmatrix} M_1 \\ M_2 \\ \vdots \\ M_K \end{pmatrix}. \quad (41)$$

where

$$M_k = \max_i c_{ik}. \quad (42)$$

Here  $c_{ik}$  is a code given by Eq.(21). The dimension of this representation is  $K$ .

### Super-vector representation

Super-vector representation [21], which captures the first order differences between input vectors and codewords in addition to the histogram, is given by

$$\phi_{\text{sv}}(X) = \left[ s\sqrt{C_k/N}; \frac{1}{\sqrt{C_k/N}} \sum_{i=1}^N c_{ik}(x_i - \mu_k) \right]_{k=1}^K \quad (43)$$

where  $s$  is a parameter,  $c_{ik}$  and  $C_k$  are given by Eq. (40) and (42), respectively. Here,  $[\cdot]_{k=1}^K$  denotes concatenation of vectors in the bracket for  $k = 1, 2, \dots, K$ . The dimension of this vector is  $K(d + 1)$ .

### Fisher-vector representation

Fisher-vector representation [22] captures the first and second order differences between input vectors and codewords by using a GMM. It is given by

$$\phi_{\text{fv}}(X) = \left[ \frac{1}{N\sqrt{w_k}} \sum_{i=1}^N c_{ik} \left( \frac{x_i - \mu_k}{\sigma_k} \right); \right. \quad (44)$$

$$\left. \frac{1}{N\sqrt{2w_k}} \sum_{i=1}^N c_{ik} \left( \frac{(x_i - \mu_k)^2}{\sigma_k^2} - 1 \right) \right]_{k=1}^K \quad (45)$$

where  $c_{ik}$  is a code given by Eq. (10),  $\sigma_k$  is a vector of diagonal elements of  $\Sigma_k$ , division between vectors is an element-wise operation. The dimension of Fisher vector is  $2Kd$ .

### GMM-Supervector representation

GMM-supervector representation [55] is similar to the Fisher-vector representation, but it uses a robust parameter estimation technique, maximum-a-posteriori (MAP) estimation. It is given by

$$\phi_{\text{GS}}(X) = \left[ \frac{\sqrt{w_k}}{\tau + C_k} \sum_{i=1}^N c_{ik} \Sigma_k^{-\frac{1}{2}} (x_i - \mu_k) \right]_{k=1}^K \quad (46)$$

where  $\tau$  is a parameter of the MAP estimation,  $c_{ik}$  is a code given by Eq. (10), and

$$C_k = \sum_{i=1}^N c_{ik}. \quad (47)$$

The dimension of GMM supervector is  $Kd$ .



**Nakamasa Inoue** received the B.S., M.S. and D.Eng. degrees in computer science from Tokyo Institute of Technology, Tokyo, Japan, in 2009, 2011, and 2014, respectively. He is currently an assistant professor with Tokyo Institute of Technology. His research interests include multimedia information retrieval, statistical pattern recognition, visual and audio categorization, and large-scale benchmark evaluations. He is a member of the IEEE, IEICE and ASJ.



**Koichi Shinoda** received the B.S. and M.S. degrees from the University of Tokyo, Tokyo, Japan in 1987 and 1989, respectively, both in physics, and the D. Eng. Degree in computer science from the Tokyo Institute of Technology, Japan, in 2001. In 1989, he joined NEC Corporation, Japan, where he was involved in research on automatic speech recognition. From 1997 to 1998, he was a Visiting Scholar with Bell Labs, Lucent Technologies, Murray Hill, NJ. From June 2001 to September 2001, he was a Principal Researcher with Multimedia Research Laboratories, NEC Corporation. From October 2001 to March 2002, he was an Associate Professor with the University of Tokyo, Japan. He is currently a Professor with the Tokyo Institute of Technology. His research interests include speech recognition, video information retrieval, statistical pattern recognition, and human interfaces. He received the Awaya Prize from the Acoustic Society of Japan in 1997 and the Excellent Paper Award from the IEICE in 1998. He was Publicity Chair in INTERSPEECH2010, Video Program Co-Chair in ACM Multimedia 2012. Dr. Shinoda is a senior member of IEEE, IEICE. He is a member of ACM, IPSJ, JSAI, and ASJ. He is currently an associate editor of Computer Speech and Language and Speech Communication, Elsevier.

**SUPPLEMENTARY MATERIAL**

**Comparison with Harris-Laplace Detector**

To reduce the computational cost of image/video classification, another way is to introduce an interest-point detector. For example, by introducing the Harris-Laplace detector, which extracts corner points from an image, we obtain a smaller number of image descriptors than that in dense sampling. Table 7 compares NTN-VQ with Harris-Laplace detector on the PASCAL VOC 2007. The threshold of the detector is optimized to compare results at the same computational cost. We see that the NTN-VQ performs better than the Harris-Laplace detector in terms of Mean AP. This shows that it is better to reduce the coding cost than to reduce the number of descriptors by applying an interest-point detector. Note that since NTN search and interest-point detectors are complementary to each other, NTN search can be introduced to detector-based applications.

**Average Precision by Objects/Concepts**

Table 8 and 9 show Average Precision for each object/concept on PASCAL VOC and TRECVID datasets, respectively.

Method	$N$	Time (msec)	Mean AP
Dense sampling	49580	856.2	0.568
Dense sampling + NTN	11188	193.2	0.563
Harris-Laplace	11680	201.7	0.453

TABLE 7

**Comparison with Harris-Laplace detector.**  $N$ : the number of image descriptors (the average number of processed descriptors for NTN). Time: coding time in msec. Mean AP: mean of Average Precisions on PASCAL VOC 2007 obtained by using super-vector coding.

Object	VQ	NTN-VQ	GMM	NTN-GMM	NTN-LM-GMM
Aeroplane	0.782	0.778	0.790	<b>0.793</b>	0.790
Bicycle	0.643	0.642	<b>0.638</b>	0.632	0.634
Bird	0.493	0.485	<b>0.506</b>	0.503	0.496
Boat	0.679	0.677	<b>0.703</b>	0.699	0.699
Bottle	0.261	0.215	0.277	0.284	<b>0.285</b>
Bus	0.615	0.621	<b>0.640</b>	0.637	0.634
Car	0.757	0.754	<b>0.768</b>	0.765	0.764
Cat	<b>0.590</b>	0.587	0.579	0.571	0.575
Chair	0.534	0.541	0.545	0.544	<b>0.545</b>
Cow	0.441	0.445	<b>0.463</b>	0.458	0.461
DiningTable	<b>0.552</b>	0.533	0.536	0.535	0.542
Dog	0.392	0.368	<b>0.439</b>	0.439	0.438
Horse	0.757	0.750	0.768	<b>0.771</b>	0.769
Motorbike	0.654	0.648	<b>0.685</b>	0.677	0.674
Person	0.814	0.813	<b>0.828</b>	0.826	0.827
PottedPlant	0.263	0.246	<b>0.276</b>	0.273	0.270
Sheep	0.415	<b>0.451</b>	0.415	0.422	0.424
Sofa	0.464	0.472	<b>0.524</b>	0.523	0.515
Train	0.749	0.748	<b>0.773</b>	0.768	0.769
TvMonitor	<b>0.495</b>	0.488	0.478	0.476	0.473
Mean AP	0.568	0.563	<b>0.582</b>	0.580	0.579

TABLE 8

**Average Precision by objects on the PASCAL VOC 2007 Classification Challenge.** Super-vector representation is used for VQ and NTN-VQ. Fisher-vector representation is used for GMM, NTN-GMM, and NTN-LM-GMM. VQ: standard hard vector quantization (VQ), NTN-VQ: our neighbor-to-neighbor (NTN) search for VQ (Alg. 1), GMM: standard Gaussian mixture model (GMM), NTN-GMM: NTN search for a GMM (Alg. 2), NTN-LM-GMM: NTN-GMM with log-max approximation.

Semantic Concept	GMM	NTN-GMM	NTN-LM-GMM
Airplane_Flying	<b>0.1085</b>	0.1007	0.1029
Animal	<b>0.0577</b>	0.0530	0.0512
Asian_People	0.0162	<b>0.0272</b>	0.0242
Bicycling	0.0676	<b>0.0690</b>	0.0644
Boat_Ship	0.0945	<b>0.0999</b>	0.0906
Bus	0.0027	0.0027	<b>0.0027</b>
Car_Racing	0.0265	<b>0.0275</b>	0.0271
Cheering	0.0119	<b>0.0131</b>	0.0130
Cityscape	0.1457	0.1460	<b>0.1468</b>
Classroom	0.0136	<b>0.0141</b>	0.0139
Dancing	<b>0.0425</b>	0.0362	0.0364
Dark-skinned_People	<b>0.1376</b>	0.1219	0.1161
Demonstration_Protest	0.1349	<b>0.1507</b>	0.1374
Doorway	0.0998	0.0996	<b>0.0999</b>
Explosion_Fire	<b>0.0309</b>	0.0298	0.0271
Female_Face	0.1407	0.1477	<b>0.1506</b>
Flowers	0.0267	<b>0.0286</b>	0.0269
Ground_Vehicles	0.2176	<b>0.2201</b>	0.2111
Hand	0.0409	<b>0.0524</b>	0.0520
Mountain	<b>0.2368</b>	0.2306	0.2346
Nighttime	<b>0.1136</b>	0.1039	0.0931
Old_People	0.0417	0.0448	<b>0.0450</b>
Running	0.0729	0.0719	<b>0.0736</b>
Singing	0.0842	0.0923	<b>0.0950</b>
Sitting_Down	0.0002	0.0005	<b>0.0005</b>
Swimming	0.3351	0.3403	<b>0.3440</b>
Telephones	<b>0.0063</b>	0.0053	0.0054
Throwing	0.0457	0.0467	<b>0.0479</b>
Vehicle	<b>0.2145</b>	0.2064	0.2135
Walking	0.0927	0.0960	<b>0.0964</b>
Mean AP	0.0887	<b>0.0893</b>	0.0881

TABLE 9

**Average Precision on the TRECVID 2010 Semantic Indexing Task.** GMM: standard Gaussian mixture model (GMM), NTN-GMM: NTN search for a GMM (Alg. 2,  $\delta = 0.09$ ), NTN-LM-GMM: NTN-GMM with log-max approximation  $\delta = 0.09$ . GS representation is used.