Neighbor-To-Neighbor Search for Fast Coding of Feature Vectors

Nakamasa Inoue and Koichi Shinoda
Tokyo Institute of Technology, Tokyo, 152-8552, Japan.
inoue@ks.cs.titech.ac.jp, shinoda@cs.titech.ac.jp

Abstract

Assigning a visual code to a low-level image descriptor, which we call code assignment, is the most computationally expensive part of image classification algorithms based on the bag of visual word (BoW) framework. This paper proposes a fast computation method, Neighbor-to-Neighbor (NTN) search, for this code assignment. Based on the fact that image features from an adjacent region are usually similar to each other, this algorithm effectively reduces the cost of calculating the distance between a codeword and a feature vector. This method can be applied not only to a hard codebook constructed by vector quantization (NTN-VQ), but also to a soft codebook, a Gaussian mixture model (NTN-GMM). We evaluated this method on the PASCAL VOC 2007 classification challenge task. NTN-VQ reduced the assignment cost by 77.4% in super-vector coding, and NTN-GMM reduced it by 89.3% in Fisher-vector coding, without any significant degradation in classification performance.

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 assigning visual words to low-level image descriptors [1], finding the closest matches for image mosaicing [2], or searching for nearest neighbor shapes to 3D shape models [3].

Most of them are simplified into either one of two problems: a hard-vector-quantization (VQ) problem or a soft-VQ problem. In hard VQ, each input vector is assigned to its closest codeword. 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. 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. One of the advantages of soft VQ is that it reduces quantization errors. However, there is a trade-off between speed and accuracy: soft VQ is more accurate but slower than hard VQ.

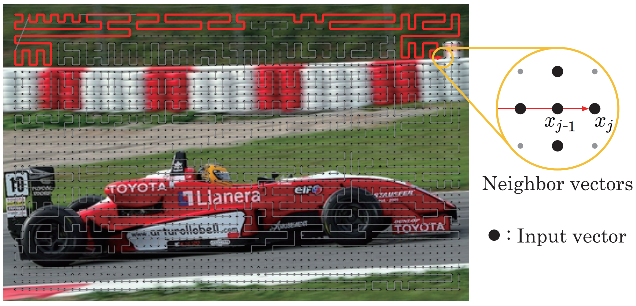

Figure 1. **Neighbor-to-neighbor (NTN) search.** NTN search assigns a code to an input vector from a neighbor vector to a neighbor vector. A typical example of a neighbor vector is a descriptor x_j adjacent to a descriptor x_{j-1} where image descriptors are densely sampled from an image. The red path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

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 [4], randomized kd-trees [5], hierarchical k-means tree [6] 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 [7] extends hierarchical k-means to a GMM framework.

These previous studies assume input feature vectors are independent from each other. However, input vectors are often strongly depend on each other when they are extracted from the same region in an image. Their typical examples are densely-sampled image descriptors such as dense SIFT, which have been proven to be effective in image classification [8, 9]. In dense SIFT, two adjacent descriptors are often assigned to the same codeword since they are similar to each other. This observation brings us to an idea to speed up VQ by skipping calculations for such similar input vectors.

This paper proposes a fast computation method for code assignment, which we call Neighbor-to-Neighbor (NTN) search. 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. We apply NTN search to hard VQ (NTN-VQ) and a GMM based soft VQ (NTN-GMM). In our experiments on image classification, we demonstrate the effectiveness of NTN-VQ on super-vector coding, and NTN-GMM on Fisher-vector coding.

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. Section 4 describes how to extend it to a GMM based soft VQ. Section 5 reports the result of our evaluation, and Section 6 concludes the paper.

2. Related work

Hard VQ is one of the most widely used methods in computer vision algorithms including bag-of-visual-words (BoW) [1]. A codebook is typically trained by using k-means algorithm. 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 a tree structure. Nistér *et al.* [10] propose a "vocabulary tree" which uses a hierarchical k-means tree. Lowe [2] uses a kd-tree for nearest neighbor search of SIFT descriptors. Muja and Lowe [6] have proposed an automatic selection method from the recent two approximate nearest neighbor (ANN) algorithms: randomized kd-trees [5], and hierarchical k-means tree [6]. They have provided it as a fast software library for approximate nearest neighbors (FLANN). These methods are often used with dimension reduction techniques such as product quantization [11].

Soft VQ, which assigns more than one codewords to an input vector, has been proposed to reduce quantization errors in hard VQ. For example, a Gaussian mixture model (GMM) [12] provides soft weighting based on the ratios of Gaussian probabilities. Tree-GMM [7] extends the hierarchical k-means to a GMM framework in order to calculate Gaussian probabilities quickly. Sparse coding [13] assigns several tens of codewords to an input vector by solving a constrained least square fitting problem. J. Wang el al. [14] introduced k-nearest neighbor (k-NN) search as the preprocessing to the sparse coding.

K. Chatfield et al. [15] compared recent image representations using these hard/soft VQ algorithms and reported that the Fisher-vector (FV) coding [9] is the best and the super-vector (SV) coding [8] is the second in terms of image classification accuracy. FV and SV use a GMM based soft VQ and hard VQ to assign codewords to image descriptors, respectively. The speed of these methods is faster than that of the other methods since their codebook size is relatively small ($256 \leq K \leq 1024$). However, the coding step is still the time-bottleneck of a pipeline for extracting image representations.

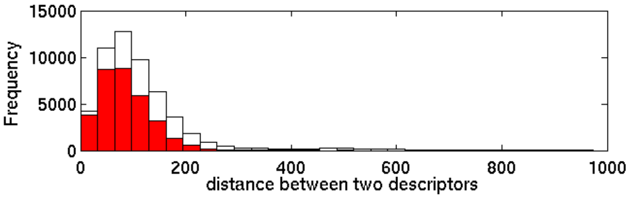

Figure 2. **A histogram of descriptors.** Red bars: descriptors that have the same visual word as a neighbor descriptor. White bars: all descriptors. SIFT descriptors are extracted from every 4 pixels at 5 scales on the PASCAL VOC 2007 training images. The codebook size is 512. 61.3% of two adjacent descriptors have the same visual word.

In our experiments, the coding step occupies 85.3% and 88.4% of computational time in FV coding and SV coding, respectively (see Section 5, Figure 11).

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 and $B(x)$ be a set of neighbor vectors for an input vector $x \in X$. The NTN search assumes that a neighbor vector in $B(x)$ is similar to x, and that the number of neighbor vectors is smaller than the codebook size. A typical example that satisfies this assumption is densely-sampled SIFT descriptors for image classification . Here, $B(x)$ is a set of the four descriptors adjacent to a descriptor x (Figure 1 and Figure 2) or a set of descriptors in the same pre-segmented region.

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.

3.2. Algorithm

Let $\{\mu_k\}_{k=1}^{K}$ be a codebook. In the initialization step for $j = 1$, x_j is randomly selected from X. Its code v_j is determined as

$$v_j = \operatorname*{argmin}_k d_{jk}, \tag{1}$$

where distance

$$d_{jk} = \|x_j - \mu_k\|, \tag{2}$$

is calculated for each $k = 1, 2, \cdots, K$. This process is the same as the straightforward hard VQ.

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

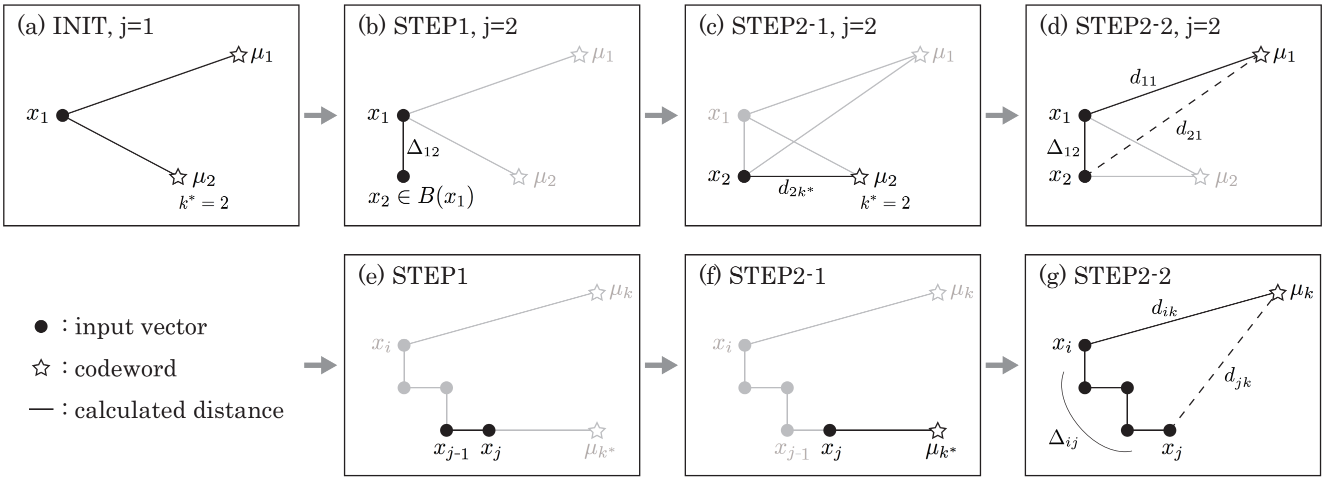

Figure 3. **Algorithm overview.** (a) Initialization step: distance from an input vector x_1 to each codeword is calculated. (b) STEP 1: the next input vector x_2 which minimizes Δ_{12} is selected from neighbor vectors. (c) STEP 2-1: d_{2k^*} is calculated where k^* is the code for x_1. (d) STEP 2-2: a lower bound $\underline{d}_{21} = d_{11} - \delta\Delta_{12}$ is calculated where δ 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 Δ_{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).

(STEP 1: Select the next input vector)
For each $x \in B(x_{j-1})$, calculate $\Delta(x) = \|x - x_{j-1}\|$, and set

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

where $\bar{X} = X \setminus \{x_1, \cdots, x_{j-1}\}$ 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 $k^* = v_{j-1}$.
2-1) Calculate distance d_{jk^*}.
2-2) For $k = 1, 2, \cdots, k^* - 1, k^* + 1, \cdots, K$, calculate a lower bound $\underline{d}_{jk}$ for d_{jk} as follows.

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

where i is the index of the input vector whose distance d_{ik} has been calculated, δ is a parameter, and Δ_{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

$$v_j = \operatorname*{argmin}_{k \in E} d_{jk}, \tag{5}$$

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

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 3 (g)). Take the maximum such index i and let Δ_{ij} be an accumulated distance between x_i and x_j given by

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

The triangle inequality gives

$$d_{ik} - \Delta_{ij} \leq d_{jk} \leq d_{ik} + \Delta_{ij}. \tag{7}$$

It implies

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

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 δ

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

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

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.

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

Algorithm 1 NTN-VQ

```
Input: input vectors $X$ $(N = |X|)$,
        codebook $\{\mu_k\}_{k=1}^K$, parameter $\delta$.
Output: codes $\{v_i\}_{i=1}^N$
$x_1 \leftarrow \mathrm{Rand}(X)$
$\underline{d}_k \leftarrow \|x_1 - \mu_k\|$ for all $k$
$v_1 \leftarrow \mathrm{argmin}_k\, \underline{d}_k;\ k^* \leftarrow v_1$
for $i = 2, \cdots, N$ do
  $x_i \leftarrow \mathrm{argmin}_{x \in B(x_{i-1}) \cap \bar{X}} \|x - x_{i-1}\|$
  $\underline{d}_{k^*} \leftarrow \|x_i - \mu_{k^*}\|$
  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
  $v_i \leftarrow k^*$
end for
```

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 μ_k, Σ_k and w_k $(k = 1, 2, \cdots K)$ 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 $x_j (j = 1, 2, \cdots, N)$ to the k-th codeword is given by

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

Here p_{jk} is a Gaussian probability given by

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

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

Our empirical observation shows that the distribution of p_{jk} over all the codewords is peaky, i.e., for each input vector x_j, a few p_{jk}'s have a large value and the others do not. 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

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

where p_{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 $\overline{p}_{jk}$ of a probability p_{jk} (see Appendix for details) given by

$$\overline{p}_{jk} = p_{ik} \exp\left(\delta_{ik} \Delta_{ij}\right). \tag{13}$$

Here, Δ_{ij} is the accumulated distance given by Eq. (6) and δ_{ik} is given by

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

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 Σ_k^{-1}. Note that this upper bound is obtained efficiently from a previous upper bound by

$$\overline{p}_{jk} = \overline{p}_{j-1,k} \exp\left(\delta_{ik} \|x_j - x_{j-1}\|\right). \tag{15}$$

Finally, instead of the intersection of bottom sets $G_i^{(b)} \cap G_j^{(b)}$, its subset U_{ij} given by

$$U_{ij} = \{k : \overline{p}_{jk} < p_{\text{th}}\}, \tag{16}$$

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

The threshold p_{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

$$p_{\text{th}} = p_{jk^*}, \quad k^* = \operatorname*{argmax}_k p_{j-1,k}. \tag{17}$$

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 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

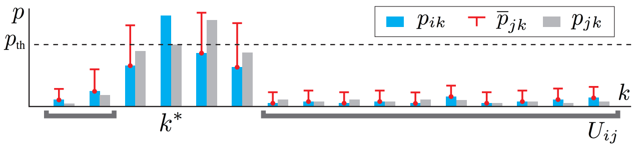

Figure 4. **Distribution of p_{ik} and $p_{jk}(i < j)$.** Calculation of a Gaussian probability p_{jk} is skipped for $k \in U_{ik}$.

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

$$c_{jk} \simeq \begin{cases} 1, & \text{if } k = \operatorname*{argmax}_k \log p_{jk}, \\ 0, & \text{otherwise.} \end{cases} \tag{18}$$

5. Experimental evaluation

5.1. Experimental setup

We perform image classification experiments on the PASCAL VOC 2007 classification challenge [16]. The dataset 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. A single core of a 2.93 GHz Intel Xeon CPU with an 8 GB memory is used for measuring computational costs.

We implement NTN-VQ (Alg. 1) on super-vector (SV) coding [8] and NTN-GMM (Alg. 2) on Fisher-vector (FV) coding [9]. We compare them with two standard methods, VQ, GMM, and two tree-based methods, ANN-VQ, Tree-GMM. The ANN-VQ uses a fast library for approximate nearest neighbor (ANN) search in [6, 15] for SV coding. The Tree-GMM [7] is an extension of the hierarchical k-means to a GMM framework for FV coding. In addition, NTN-LM-GMM applies the LM approximation to NTN-GMM.

In all experiments, 2×2 SIFT descriptors are extracted from every 4 pixels [1] 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. We omit Gaussian weighting for SIFT descriptors. The averaged number of descriptors per image is 49580. A codebook is trained on randomly sampled 1 million descriptors by using k-means algorithm for VQ or EM algorithm for a GMM. 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.

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 $\{c_{ik}\}_{i=1}^{N}{}_{k=1}^{K}$
$x_1 \leftarrow \text{Rand}(X)$
$p_k, \overline{p}_k \leftarrow w_k \mathcal{N}(x_1|\mu_k, \Sigma_k)$ 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 $k$; $k^* \leftarrow \operatorname*{argmax}_k p_k$
for $i = 2, \cdots, N$ do
    $x_i \leftarrow \operatorname*{argmin}_{x \in B(x_{i-1}) \cap \bar{X}} \|x - x_{i-1}\|$
    $p_{k^*}, \overline{p}_{k^*} \leftarrow w_{k^*} \mathcal{N}(x_i|\mu_{k^*}, \Sigma_{k^*})$
    for all $k \neq k^*$ do
        $\overline{p}_k \leftarrow \overline{p}_k \exp(\delta_k \|x_i - x_{i-1}\|)$
        if $p_{k^*} < \overline{p}_k$ then
            $p_k, \overline{p}_k \leftarrow w_k \mathcal{N}(x_i|\mu_k, \Sigma_k)$
            $\delta_k \leftarrow S_k \delta \|x_i - \mu_k\|_{\Sigma_k^{-1}}$
        end if
    end for
    $c_{ik} \leftarrow \frac{p_k}{\sum_{k'=1}^{K} p_{k'}}$ for all $k$; $k^* \leftarrow \operatorname*{argmax}_k p_k$
end for
```

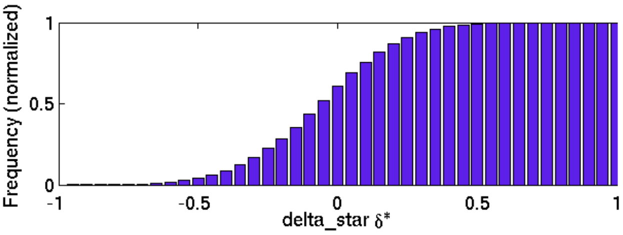

Figure 5. **Cumulative histogram of δ^*.** Statistics of the true δ^* in Eq. (8) on PASCAL VOC 2007 training images is reported for NTN-VQ.

5.2. Experimental Results

5.2.1 Speed of coding

In Table 1, we compare speed of coding at the fixed accuracy level. Overall, our NTN methods are faster than the others while keeping the classification accuracy. We observe that NTN-VQ and NTN-LM-GMM reduce the assignment cost by 77.4% and by 89.3%, respectively. 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 1. Here, a parameter δ 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 $\underline{d}_{jk}$ gives a correct lower bound when $\delta = 0.20$ for NTN-VQ as shown in Figure 5.

Figure 6 shows the speed-accuracy trade-off for NTN

[1] Mean APs were 0.582, 0.582 and 0.574 for density of 3, 4, and 5, respectively.

Method	δ	Mean AP [test/val]	$\lvert E\rvert$	Time (sec)	Reduction rate (%)
VQ	-	***0.568 / 0.519**	512.0	856.2	0.0
ANN-VQ [6]	-	0.563 / 0.518	-	475.7	44.4
NTN-VQ	0.20	0.563 / 0.519	**57.8**	193.2	**77.4**
GMM	-	***0.582 / 0.532**	512.0	2595.7	0.0
Tree-GMM [7]	-	0.582 / 0.532	295.2	1496.8	42.3
NTN-GMM	0.09	0.580 / 0.531	88.0	642.0	75.3
NTN-LM-GMM	0.09	0.579 / 0.531	**88.0**	**276.8**	**89.3**

Table 1. **Speed comparison at the fixed accuracy level.** VQ: standard hard vector quantization (VQ), ANN-VQ: approximate nearest neighbor search [6], 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 [7] NTN-GMM: our NTN search for a GMM (Alg. 2), NTN-LM-GMM: NTN-GMM with log-max approximation. δ: a parameter of our NTN methods, Mean AP: image classification accuracy on the testing set and the validation set of the VOC 2007 classification challenge. $|E|$: the number of distance or probability calculations per input vector, Time: assignment time in sec, Reduction rate: reduction rate of the assignment 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 ($p < 0.05$).

methods for different values of δ. 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 7: a tree-based ANN-VQ performs better than RAND-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 8. If we ignore computation time for Δ_{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).

5.2.2 Result examples

We examine the effectiveness of NTN-VQ for several different images in Figure 9. The reduction rate of the assignment cost by NTN-VQ is 84.9% for the image (a) and 66.8% for the image (d). Here, (a) and (d) 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.

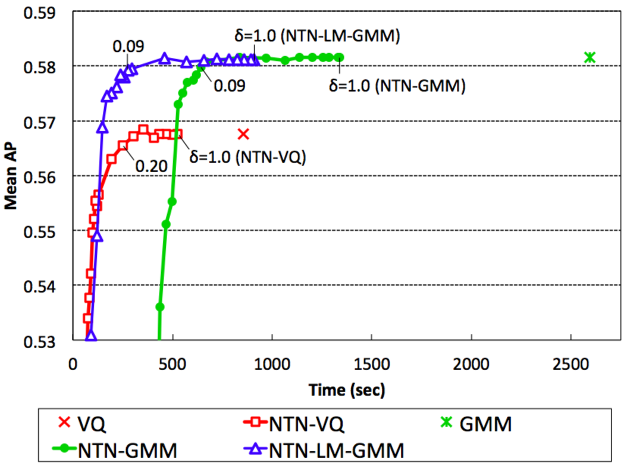

Figure 6. **Speed-accuracy trade-off for different values of δ.** Trade-off between assignment time and Mean AP is reported. All plots are for $\delta = 1.0, 0.9, \cdots 0.1, 0.09, \cdots, 0.01$. VQ: standard hard vector quantization (VQ), NTN-VQ: neighbor-to-neighbor (NTN) search for VQ, GMM: standard Gaussian mixture model (GMM), NTN-GMM: NTN search for a GMM, NTN-LM-GMM: NTN-GMM with the log-max approximation.

5.2.3 Codebook size

The simplest idea to reduce the assignment cost is to reduce the codebook size. In Figure 10, 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 $K = 2048, 1024, \cdots, 16$.

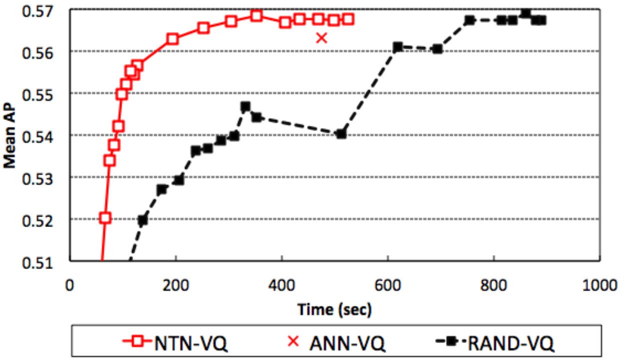

Figure 7. **Comparison with RAND-VQ.** Trade-off between assignment time and Mean AP is reported. NTN-VQ: neighbor-to-neighbor (NTN) search for VQ, this is the same plot as Figure 6, ANN-VQ: approximate nearest neighbor search [6], RAND-VQ: NTN-VQ in which a neighbor vector is replaced by a randomly sampled vector.

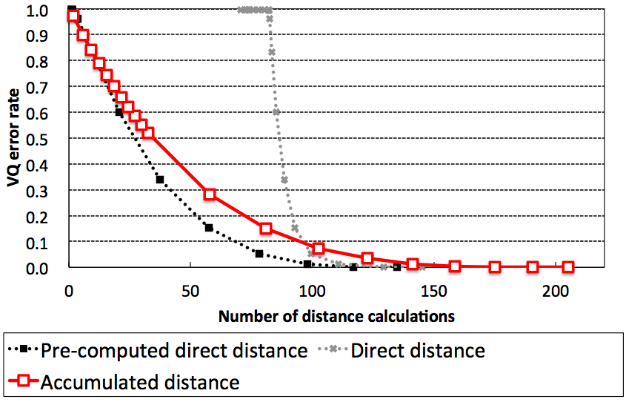

Figure 8. **Comparison of the accumulated distance and the direct distance.** VQ error rate in NTN-VQ for different values of δ is reported. All plots are for $\delta = 1.0, 0.9, \cdots 0.1, 0.09, \cdots 0.01$. Accumulated distance: Δ_{ij} is defined by Eq. (6). Direct distance: Δ_{ij} is replaced by the direct distance $\|x_i - x_j\|$. Pre-computed direct distance: the direct distance is used but distance calculations for it are not counted.

5.2.4 Relative computational time in a pipeline

Figure 11 shows the relative cost of coding with respect to the cost of the other steps of processing pipeline for extracting SV and FV representations. 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 coding and SV coding, 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 coding, which generate a final FV representation, is also reduced by the LM approximation since we can skip some summations in pooling if c_{ik} is equal to zero.

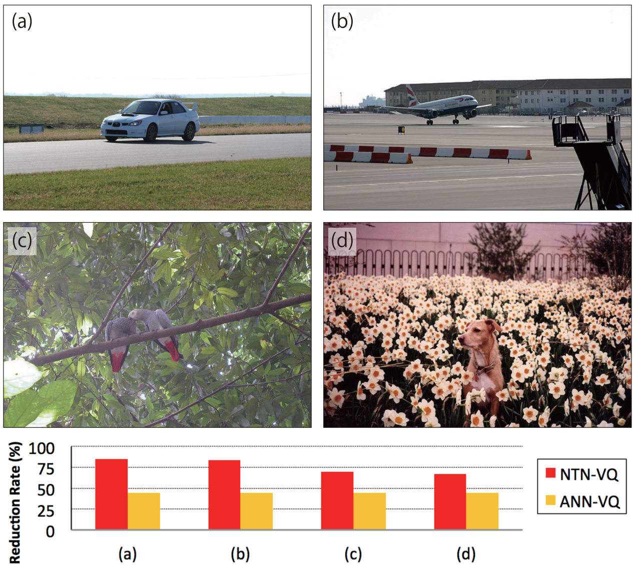

Figure 9. **The computational cost reduction by NTN-VQ for different images.** Four images (a), (b), (c), and (d) are from PASCAL VOC 2007. The reduction rate of the assignment cost by NTN-VQ and ANN-VQ for each image is reported.

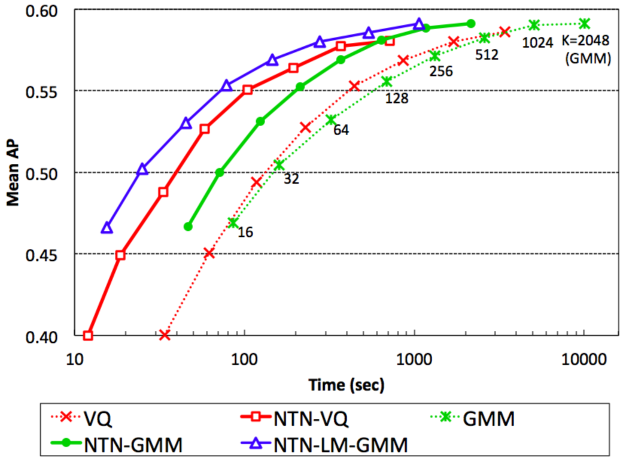

Figure 10. **Speed-accuracy trade-off for different codebook sizes.** Trade-off between assignment time and Mean AP for codebook sizes of $K = 2048, 1024, 512, \cdots, 16$. is reported. VQ: standard hard vector quantization (VQ), NTN-VQ: neighbor-to-neighbor (NTN) search for VQ $\delta = 0.20$, GMM: 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 $\delta = 0.09$.

For large-scale image classification, for example on the ImageNET with more than 20,000 object categories, 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 [11] to the final image representation can be effectively utilized.

6. 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) and a Gaussian mixture model (GMM). We tested NTN-VQ and NTN-GMM on super-vector coding and Fisher-vector coding, respectively. Our experiments on the PASCAL VOC 2007 classification challenge showed that NTN-VQ, NTN-GMM, and NTN-LM-GMM reduced the assignment cost by 77.4%, 75.3%, and 89.3%, respectively, without any significant degradation in the image classification performance. This result confirms the effectiveness of our proposed algorithms.

In future work, we will focus on the speeding up of the feature extraction step that we didn't discuss in this paper. Approximation of dense sampling and SIFT descriptors would be interesting as promising next steps.

Acknowledgement

This work was supported by JSPS KAKENHI Grant Number 11J04223.

References

[1] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. *Proc. ECCV SLCV workshop*, pages 59–74, 2004. 1, 2

[2] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004. 1, 2

[3] M. Ovsjanikov, W. Li, L. Guibas, and N. J. Mitra. Exploration of continuous variability in collections of 3d shapes. *ACM Trans. Graph.*, 30(4):1–10, July 2011. 1

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

[5] C. Silpa-anan and R. Hartley. Optimised kd-trees for fast image descriptor matching. *Proc. CVPR*, pages 1–8, 2008. 1, 2

[6] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *Proc. VISAPP*, pages 331–340, 2009. 1, 2, 5, 6, 7

[7] N. Inoue and K. Shinoda. A fast map adaptation technique for gmm-supervector-based video semantic indexing systems. *Proc. ACM Multimedia*, pages 1357–1360, 2011. 1, 2, 5, 6

[8] X. Zhou, K. Yu, T. Zhang, and T. S. Huang. Image classification using super-vector coding of local image descriptors. *Proc. ECCV*, pages 141–154, 2010. 1, 2, 5

[9] F. Perronnin, S. Jorge, and T. Mensink. Improving the fisher kernel for large-scale image classification. *Proc. ECCV*, pages 143–156, 2010. 1, 2, 5

[10] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. *Proc. CVPR*, pages 2161–2168, 2006. 2

[11] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011. 2, 8

[12] F. Perronnin, C. Dance, G. Csurka, and M. Bressan. Adapted vocabularies for generic visual categorization. *Proc. ECCV*, pages 464–475, 2006. 2

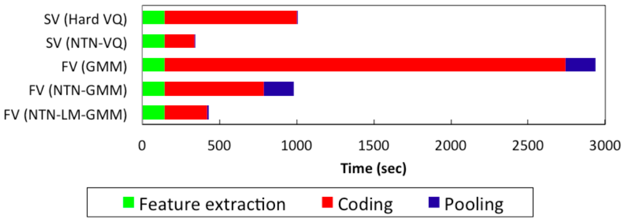

Figure 11. **Relative computational cost.** Computational cost for each step of super-vector (SV) coding and Fisher-vector (FV) coding is reported. The codebook size is 512. Feature extraction: SIFT descriptors are extracted from every 4 pixels at 5 scales, Coding: each descriptor is assigned to codeword(s), 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.

[13] T. Huang. Linear spatial pyramid matching using sparse coding for image classification. *Proc. CVPR*, pages 1794–1801, 2009. 2

[14] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. *Proc. CVPR*, pages 3360–3367, 2010. 2

[15] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. *Proc. BMVC*, pages 1–12, 2011. 2, 5

[16] M. Everingham, A. Zisserman, C. Williams, and L. V. Gool. The pascal visual obiect classes challenge 2007 (voc2007) results. *http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html*, 2007. 5

Appendix

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

$$\exists\delta^* \in [-1,1] \text{ s.t. } \|x_j - \mu_k\|^2_{\Sigma_k^{-1}} \tag{19}$$
$$= \|x_i - \mu_k\|^2_{\Sigma_k^{-1}} + \|x_i - x_j\|^2_{\Sigma_k^{-1}} - 2\delta^*\|x_i - x_j\|_{\Sigma_k^{-1}}\|x_i - \mu_k\|_{\Sigma_k^{-1}}.$$

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

$$\begin{aligned} \|x_j - \mu_k\|^2_{\Sigma_k^{-1}} &\geq \|x_i - \mu_k\|^2_{\Sigma_k^{-1}} - 2S_k\delta\|x_i - x_j\|\|x_i - \mu_k\|_{\Sigma_k^{-1}} \\ &\geq \|x_i - \mu_k\|^2_{\Sigma_k^{-1}} - 2S_k\delta\Delta_{ij}\|x_i - \mu_k\|_{\Sigma_k^{-1}}, \end{aligned} \tag{20}$$

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

$$\begin{aligned} p_{jk} &= \frac{w_k}{Z_k}\exp\left(-\frac{1}{2}\|x_j - \mu_k\|^2_{\Sigma_k^{-1}}\right) && (21) \\ &\leq \frac{w_k}{Z_k}\exp\left(-\frac{1}{2}\|x_i - \mu_k\|^2_{\Sigma_k^{-1}} + S_k\delta\Delta_{ij}\|x_i - \mu_k\|_{\Sigma_k^{-1}}\right) && (22) \\ &= p_{ik}\exp\left(S_k\delta\Delta_{ij}\|x_i - \mu_k\|_{\Sigma_k^{-1}}\right) && (23) \\ &= p_{ik}\exp\left(\delta_{ik}\Delta_{ij}\right) = \overline{p}_{jk}, && (24) \end{aligned}$$

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