Product Quantization を用いた近似 k 最近傍探索の高速化

湯川 皓太† 天笠 俊之‡

†筑波大学システム情報工学研究科 〒 305-8573 茨城県つくば市天王台 1-1-1
‡筑波大学計算科学研究センター 〒 305-8573 茨城県つくば市天王台 1-1-1
E-mail: †yukawa@kde.cs.tsukuba.ac.jp, ‡amagasa@cs.tsukuba.ac.jp

あらまし 近似最近傍（ANN）探索は画像検索，クラスタリングなどの分野で幅広く用いられている．一方で，ANN 探索の多くの手法では時間，空間計算量が大きく高次元，大規模なデータセットに対して適用するのが難しいという問題点がある．ANN 探索の代表的な手法の一つとして，Product Quantization(PQ)がある．PQ では高次元ベクトルを複数のサブベクトルに分割し，量子化を行うことで，空間計算量と検索精度のトレードオフの向上を達成している．また，PQ のアルゴリズムではルックアップテーブルを用いることによって近似距離の計算時間削減を行なっている．本研究では PQ アルゴリズムのルックアップテーブルを用いた近似距離計算において,ルックアップテーブルを参照する回数を削減することで検索時間を短縮するアルゴリズムを提案する．実験ではテキストと画像のデータセットを用いて，従来の PQ アルゴリズムと比べて，精度を落とすことなく検索時間が削減できていること示した．

キーワード 情報検索，最近傍探索，近似最近傍探索，Product Quantization

1. はじめに

近年インターネットの普及やその利用の増加によって，情報検索の重要性も増している．情報検索において需要の多い検索方法が与えられたデータに対して類似したデータを検索，推薦するという検索方法である．このような検索方法は閲覧しているニュース記事に類似した記事を推薦したり，ある画像に類似した画像を検索したりするといったタスクに活用されている．

そのような類似したデータを検索する際によく用いられる手法が最近傍探索法である．最近傍探索はデータを距離空間に射影してベクトルに変換し，あるデータ間の類似度を計算することによって最も類似したデータを検索することができる。ここで使用する類似度はユークリッド距離やコサイン類似度などが用いられる．また，最近傍探索を拡張し，近傍の k 件のランキングされたデータを結果として出力する k 最近傍探索もよく用いられている．最近傍探索は比較的単純なアルゴリズムなため実装が簡単である．一方で，次元数 D，データ数 N のデータに対して探索を行う場合，計算量は$O(ND)$となるため，高次元，大規模なデータに対して実行するのは現実的でない．

そこで，現実的には近似最近傍探索が用いられている．近似最近傍探索では最近傍探索で求められるような厳密解を求める代わりに，高速に近似解を求める．

近似最近傍探索の代表的な手法の一つに Product Quantization（PQ） [1]がある．PQ では精度，検索速度，計算時のメモリ消費量のトレードオフが取れた手法ということで近年 PQ を元にした様々な発展手法が発表されている．

PQ はルックアップテーブルを用いることで高速に検索を行うことができる．一方で，ルックアップテーブルを用いた検索方法は原論文の PQ から変わっていない．そこで，本研究ではルックアップテーブルを用いた検索方法において，近似最近傍探索において無駄な参照回数を減らすことによって検索時間を効率化する手法を提案する．それに加え，ルックアップテーブルを並び替えることによってさらに検索を高速化する手法も合わせて提案する．

実験ではテキスト，画像の２種類のデータセットを用いて提案手法が PQ で用いられている方法に比べて検索の精度を落とすことなくより短い時間で検索できていることを示した．

本稿の構成は以下の通りである．まず，２章で ANN 探索の関連研究を紹介し，３章で PQ に関する前提知識について説明する．次に４章では本研究の提案手法について説明し，５章で提案手法に対する実験について述べる．最後に６章で本研究の結論と今後の課題について述べる．

2. 関連研究

2.1. ANN 探索の関連研究

ANN 探索の手法は多くの場合，検索の精度，計算コスト，メモリコストという３つの観点のトレードオフから評価する．ANN 探索のための手法は数多く提案されているが，ベクトルの変換方法，データ構造などの観点から大きく３種類に分類することができる．３種類の手法の特徴を表１に示す．

Locally Sensitive Hashing（LSH） [3]などの Hash 系

の手法では，ハッシュ関数を用いて類似のベクトルを一定の確率で類似のハッシュコードにマッピングする．このようにハッシュコードにマッピングすることによって，多数のベクトルを少数のハッシュコードで近似表現できるため計算するべきベクトル数を減らすことができる．またハッシュコードではハミング距離によってベクトル間の類似度を計算する事ができるので，変換前のベクトル間の距離を計算するのに比較して大幅に計算コスト,メモリコストを削減できる.一方で，多数のベクトルを単一のハッシュコードで表現するので変換誤差が大きくなり，検索の精度は大きく低下する．

kd-tree [4]などの Tree 系の手法では，検索対象のベクトルを距離空間内の距離に基づいて木構造で保持しておく．このようなデータ構造にすることによって近似最近傍探索の際に，事前に分割されている近傍のデータのみに対して距離計算を行うことで探索を行うことができる．一方で，ベクトルをコードに変換することなく直接ベクトルから距離を計算するため大規模，高次元なデータに対しては依然として計算コスト，メモリコストは高くなる．

Product Quantization [1]などの Quantization 系の手法ではベクトル量子化と呼ばれる技術を用いてベクトルを複数のコードワードの組み合わせによって表現する．この手法では Hash 系の手法と比較して単一のコードではなく，複数のコードの組み合わせのため誤差が小さくなるため検索の精度も高い．Tree 系の手法と比較するとコードに変換して距離計算を行うため計算コスト,メモリコストは小さくなる.よって Quantization 系の手法はこれら３つの観点から見た場合，最も良いトレードオフが取れた手法ということになる．

表１　ANN 関連研究の分類ごとの特徴

	Hash 系	Tree 系	Quantization 系
精度	△	◎	○
計算コスト	◎	△	○
メモリコスト	◎	△	○

2.2. PQ の関連研究

現在 PQ を発展させた様々な発展手法が提案されている.事前変換を用いた手法 [5], [6]では回転行列を用いてベクトルを事前に最適な回転を行うことで，量子化誤差を減少させ検索精度を向上させている．サブベクトルを複数のコードブックで表現する手法 [7], [8]では，サブベクトルを複数のコードブックからなるコードワードの和で表すことで検索精度を向上させている．また，Additive Quantization [9]，Composite Quantization [10]では PQ の量子化を一般化して定式化している．一般化された量子化前アルゴリズムを用いて新しい問題設定に対応させた，エンコーディング速度を向上させた手法 [11]，教師付きデータを用いた手法 [12]，マルチモーダルモデルを用いた手法 [13]，ストリームデータに対してオンライン学習アルゴリズムを加えた手法 [14]などがある．また，近年はデータを CNN などのニューラルネットワークを用いてベクトル化し，コードブックとネットワークの損失を一貫して最小化する手法 [15], [16] 手法なども提案されている．

2.3. 既存手法の問題点

既存手法では PQ を元にして，コードブックを用いた量子化誤差を小さくすることによって検索精度を向上させている．一方で，検索の際の距離計算においてルックアップテーブルを用いてサブベクトルごとの距離を読み出し，すべてのベクトルに対して距離を計算するという手法は変わっていない．このような手法では近似 k 最近傍探索において計算の必要のないベクトルに対しても距離計算を行ってしまっている．これによって無駄なルックアップテーブルの参照が行われているため，メモリアクセスの回数が増大し，検索時間も増大してしまうという問題点がある．

3. 前提知識

3.1. ベクトル量子化

ベクトル量子化 [2]ではベクトル$X = \{x_0, \dots, x_{N-1}\} \in \mathbb{R}^D$を k-means 法によってクラスタリングする．クラスタリングした各ベクトルをクラスター重心$\boldsymbol{C} \in c_i (0 \leq i \leq k-1)$に近似する．このとき，クラスター重心$c_i$をコードワード，クラスター重心の集合$\boldsymbol{C}$をコードブックと呼ぶ．コードブック最適化のための目的関数は(1)のようになる．

$$\min_{C} \sum_{X} \|x - c_i(x)\|^2 \tag{1}$$

このように多数のベクトルを少数のコードワードに近似することによって，近似最近傍探索において距離を計算するべきベクトルの数を減らすことができる．

3.2. PQ によるベクトル量子化

Product Quantization [1]ではベクトルをM個のサブベクトルに分割する．このとき，元のベクトルxは，

$$x = x^1 \times \dots \times x^M \tag{2}$$

のように，分割されたサブベクトルの直積で表現することができる．次に分割されたM個のサブベクトルそれぞれに対してベクトル量子化を行う．コードブック最適化のための目的関数は(**3**)のようになる．

$$\min_{\boldsymbol{C}^1, \dots, \boldsymbol{C}^M} \sum_{X} \|x^m - c_i^m(x^m)\|^2, s.t\, \boldsymbol{C} = \boldsymbol{C}^1 \times \dots \times \boldsymbol{C}^M \tag{3}$$

このようにベクトルを複数のサブベクトルに分割してからベクトル量子化を行うことによって，ベクトルを一つのベクトルを複数のコードワードの組み合わせで

表現するので,量子化した際の誤差が小さくなるので,近似最近傍探索における精度を向上させることができる.

3.3. PQ による近似k最近傍探索

PQ を用いた近似最近傍探索では,ルックアップテーブルを用いて量子化されたベクトル間の距離を計算することで実現している.検索対象となるデータベースに格納されているベクトルは事前に最適化されたコードブックが生成されており,その生成されたコードブックを用いて各ベクトルはM個のコードワードの組み合わせによって量子化されているとする.

まず,与えられたクエリベクトル$\boldsymbol{q}$をM分割して得られる各サブベクトルとそれに対応するコードブック内のコードワードとの間の距離をすべて計算し,その計算結果をルックアップテーブルに保存する.よってルックアップテーブル$\boldsymbol{l}(\boldsymbol{q})$は,

$$\boldsymbol{l}(\boldsymbol{q}) = \begin{bmatrix} d_{0,0} & \cdots & d_{0,k-1} \\ \vdots & \ddots & \vdots \\ d_{M-1,0} & \cdots & d_{M-1,k-1} \end{bmatrix} \tag{4}$$

のように,$k \times M$の行列で表すことができる.

次にデータベースベクトルとクエリベクトルとの間の距離を計算する.データベースサブベクトルはコードワードのインデックスによって量子化されているので,データベースベクトル$\boldsymbol{V}=\{\boldsymbol{v}_0, \ldots \boldsymbol{v}_i, \ldots \boldsymbol{v}_{N-1}\}$は,(5)のようにコードワードの組み合わせによって近似されている.

$$\boldsymbol{v}_i \approx \boldsymbol{c}(k_i^0) \times \ldots \times \boldsymbol{c}(k_i^{M-1}) \tag{5}$$

ここで,k_i^mは$\boldsymbol{v}_i$のm番目のサブベクトルにおけるコードワードのインデックスで,$\boldsymbol{c}(k_i^m)$はインデックスk_i^mに該当するコードワード（クラスター重心）を表している.

このとき各データベースベクトル$\boldsymbol{v}_i$と,クエリベクトル$\boldsymbol{q}$との距離はルックアップテーブルから読みだしたコードワードとの距離を用いて,

$$dist(\boldsymbol{v}_i, \boldsymbol{q}) = \sum_{m=0}^{M-1} d_{m,k_i^m} \tag{6}$$

のように,読みだしたM個の距離の和で計算することができる.このような計算をN個のデータベースベクトル全てに行うことで,近似最近傍ベクトルを得ることができる.

このようにルックアップテーブルを用いて事前にクエリベクトルと,コードワードとの距離を計算することによってN個のデータベースベクトルとの距離を計算するときには読みだした距離の総和を求めるだけで近似距離を計算することができる.この計算は次元数Dに依存していない.また,ルックアップテーブルの計算は$k \times M$個のコードワードとクエリとの距離は距離を計算すれば良いので,データベースのベクトルの数Nに依存しない.

4. 提案手法

4.1. 提案手法1

必要のないルックアップテーブルの参照回数を削減することによって PQ を用いた近似 k 最近傍探索法の検索時間を短縮させる手法を提案する.提案手法 1 では分割した部分空間に対して近傍データと推定したデータよりも距離が大きいと判断した時点で,そのベクトルは近似 k 最近傍ベクトルになり得ないので距離計算を中断する.前提条件としてすでに近似 k 最近傍データの候補として k 個のデータが検索されているとする.PQ のアルゴリズムと同様にクエリベクトルとしてデータベースベクトル間の距離を計算するには,分割されたサブベクトル間の距離を順にルックアップテーブルから順に読み出す.このとき,読みだした距離の和がベクトル全体の距離となる.このような手順で読みだしたサブベクトル間の距離で順に和を取るときに k 最近傍候補ベクトル候補との k 番目に距離が近いベクトルとクエリ間の距離と比較する.このとき計算中のベクトルの距離の和が k 番目の最近傍ベクトルとの距離以上であれば,そのベクトルは最近傍ベクトルの候補にはなりえない.よって,注目しているベクトルの距離計算を中断し,それ以降のすべてのサブベクトル間の距離をルックアップテーブルから読み出す処理を行わない.このように近似 k 最近傍ベクトルになりえないベクトルに対して途中で距離計算を中断することによって,必要のないルックアップテーブルへのメモリアクセスの回数を削減する.

Algorithm 1 提案手法 1 のアルゴリズム

Input : N // 検索対象のベクトル数
M // サブベクトルへの分割数
k // k 件の近似最近傍ベクトルを取得
codes // 量子化された検索対象ベクトル
LookupTable // ルックアップテーブル
Output : AnnIdx // k 個の近似最近傍ベクトルのインデックス

```
1: AnnDist ← []
2: Annidx ← []
3: for n ← 1 to N do
4:     d ← 0
5:     flag ← True
6:     for m ← 1 to M do
7:         if AnnDist.size == k && d >= AnnDist[k-1] then
8:             flag ← False
9:             break
10:        else
```

```
        d ← d + LookupTable[m][codes[n][m]]
    if AnnDist.size < k || flag then
        insert(AnnDist, d, AnnIdx, n, k)
```

提案手法１のアルゴリズムを Algorithm 1 に示す．アルゴリズム中で，insert(list1, x1, list2, x2, k)関数は list1 中の値が降順になるように保って list1 に新たな値 x1 を挿入し，list2 中の x1 を挿入した位置と同じ位置に x2 を挿入し，サイズが k 以上となる場合には，list1,list2 中の k 個以上の値を削除する関数である．

4.2. 提案手法2

提案手法２では提案手法１に加えてルックアップテーブルの並び替え処理を行う．

並び替えの基準としては

(1) 合計値

(2) 中央値

の２種類の基準に従ってルックアップテーブルの並び替え処理を行う．

ルックアップテーブル生成時に各サブベクトルにクエリサブベクトルとコードワード間の k 個の距離の上記２種類の基準値を計算する．この計算された距離の基準値の降順になるようにルックアップテーブルの行ごとにソートを行う．よって，ルックアップテーブルのソート操作はそれぞれ式（７），（８）のようになる．$sum()$, $med()$はそれぞれ合計値，中央値の計算を表している．

$$argsort_{\boldsymbol{M}}\{sum(d_{0,0},\dots,d_{0,k-1}),sum(d_{M-1,0},\dots,d_{M-1,k-1})\} \quad (7)$$

$$argsort_{\boldsymbol{M}}\{med(d_{0,0},\dots,d_{0,k-1}),med(d_{M-1,0},\dots,d_{M-1,k-1})\} \quad (8)$$

距離計算時にはソートされたルックアップテーブルを上から（最もコードワードの距離の合計値，中央値が大きかった行から）順に読み出して，距離の計算を行う．このようなルックアップテーブルのソート処理を事前に行うことによって，距離計算時に，順にサブベクトルの距離を読み出す際に，より読み出し回数が少ないときに提案手法の計算の打ち切りが発生する確率が大きくなるので，さらにルックアップテーブルの参照回数を減らす事ができる．

5. 実験

5.1. データセット

今回の実験ではテキストデータセットとして News20 データセット [17]，画像データセットとして Caltech101 データセット [18]の２種類のデータセットを使用した．News20 データセットは２０クラスのニュース記事のデータセットで，データ数は 18,846 である．また，各データは BERT [19]を用いてベクトル化しており，次元数は 768 となっている．Caltech101 データセットは 101 クラスの画像データセットで，データ数は 9,144 である．また，各データは GIST 特徴量記述子[20]を用いてベクトル化し，次元数は 960 となっている．またそれぞれのデータセットをシャッフルし，60%を train データ，40%を test データとして分割している．train データはコードブックを生成し，検索対象となるデータベースベクトルとして使用し，test データは実際に検索を行うことがクエリベクトルとして使用する．各データセットの詳細を表２に示す．

表２　データセット

名称	種類	ベクトル化手法	次元数	データ数
News20	テキスト	BERT	768	18,846 (train:11,307, test:7,539)
Caltech101	画像	GIST	960	9,144 (train:5,486, test:3,658

5.2. パラメータと評価指標

評価指標として Recall@R と検索時間を採用している．Recall@R は近似最近傍データとして検索された上位 R 個以内に真の最近傍データが含まれている確率として計算される．Recall@1 は accuracy と同じ値になる．本研究では R=20 として実験を行った．また比較する PQ，提案手法１，提案手法２で共通して，サブベクトルへの分割数 M=16，各サブベクトルのコードワード数 k=256 としてパラメータを設定した．

5.3. 精度と検索時間

PQ，提案手法１，提案手法２（合計値），提案手法２（中央値）の Recall@20，検索精度の比較実験の結果を表３に示す．News20 データセット，Caltech101 データセットの両方において提案手法は PQ と比べて Recall@20 は低下していない．一方で，提案手法１は最大で 1/4 程度にまで検索時間を短縮できる．これは PQ ではすべてのデータベースベクトルに対してルックアップテーブルを用いて距離を計算しているのに対して，提案手法１では近似 k 最近傍データになりえないデータに対しては距離計算が中断できているためである．また，Caltech101 データセットでは提案手法２は提案手法１よりも更に検索時間を短縮できていることがわかる．これはルックアップテーブルの並び替えによっていくつかのデータでより早期に提案手法１の計算中断が発生しているためである．今回の使用したデータセットでは提案手法２（合計値）と提案手法２（中央値）で大きな速度の差は見られなかった．

表 3 Recall@20 と検索時間

データセット	手法	Recall@20	検索時間[s] (1 クエリあたり)
News20	PQ	0.8036	0.02914
	提案手法 1	0.8036	0.00961
	提案手法 2 (合計値)	0.8036	0.00878
	提案手法 2 (中央値)	0.8036	0.00914
Caltech101	PQ	0.9289	0.01380
	提案手法 1	0.9289	0.00804
	提案手法 2 （合計値）	0.9289	0.00604
	提案手法 2 （中央値）	0.9289	0.00612

5.4. ルックアップテーブルの参照回数

1 クエリあたりにルックアップテーブルを参照する回数の平均値の分布を提案手法 1 と提案手法 2 で比較した. News20 データセットの結果を図 1 , Caltech101 データセットの結果を図 2 に示す．図中で横軸の method1 は提案手法 1，method2 sum は提案手法 2（合計値），method2 med は提案手法 2 （中央値），を示している．縦軸は 1 クエリあたりのルックアップテーブル参照回数を示している．今回は M=16 に設定しているので，参照回数の最大値は 1 6 になる．提案手法 1 は News20 データセットでは平均で 3 程度, Caltech101 データセットでは 7 程度にまで削減できている．提案手法 2 では Caltech101 データセットで，提案手法 1 に比べて提案手法 2 では更に 5 程度にまで削減できている．これはデータセットによっては提案手法 2 のルックアップテーブルの並び替えによってルックアップテーブルの参照回数を大きく削減できることを示している．

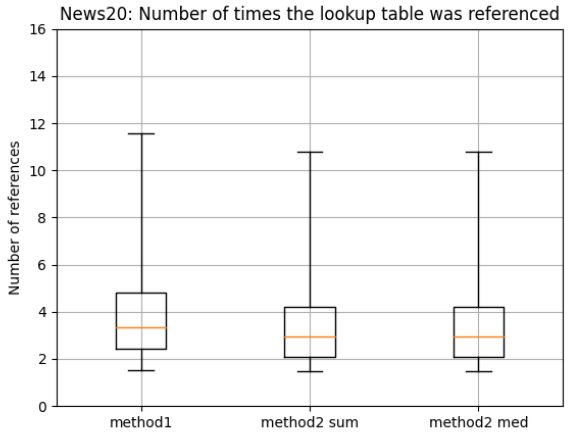

図 1　News20 のルックアップテーブル参照回数

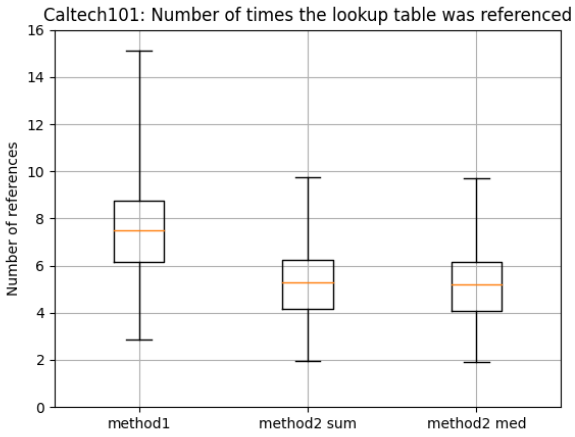

図 2　Caltech101 のルックアップテーブル参照回数

6. まとめと今後の課題

本研究では PQ を用いた近似 k 最近傍探索において，距離計算時の必要のないルックアップテーブルの参照を減らすことによって，検索精度を減少させることなく検索時間を短縮できる手法を示した．またそれに加えてルックアップテーブルの並び替えによって更に検索時間を短縮できる手法も提案した．

今後の計画としては今回提案した手法をより大規模，高次元なデータセットに対して実験を行い，有効性を検証する必要がある．また，時系列データセットに対する有効性など，異なった問題設定に対する提案手法の検証も課題となっている．

7. 謝辞

この成果は，国立研究開発法人新エネルギー・産業技術総合開発機構（NEDO）の委託業務（JPNP20006）の結果得られたものです．

参考文献

[1] H. Jegou, M. Douze , C. Schmid, “Product quantization for nearest neighbor search,” IEEE Transactions on Pattern Analysis and Machine Intelligence, 2011.

[2] R. M. Gray,“Vector quantization,”IEEE ASSP Magazine, 1984.

[3] M. Datar, N. Immorlica, P. Indyk , V. S. Mirrokni, “Locality- sensitive hashing scheme based on p-stable distributions,” Proceedings of the Symposium on Computational Geometry, 2004.

[4] C. Silpa-Anan , R. Hartley, “Optimised KD-trees for fast image descriptor matching, ” Proceedings of the IEEE Con- ference on Computer Vision and Pattern Recognition, 2008.

[5] M. Norouzi , D. J. Fleet, “Cartesian k-means,” Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2013.

[6] T. Ge, K. He, Q. Ke , J. Sun, “Optimized Product Quantization for Approximate Nearest Neighbor Search,” Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2013.

[7] J. Wang, J. Wang, J. Song, X.-S. Xu, H. T. Shen , S. Li, “Optimized Cartesian K-Means,” IEEE Transactions on Knowledge and Data Engineering.

[8] A. Babenko , V. Lempitsky, “Tree Quantization for Large-Scale Similarity Search and Classification,” Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2015.

[9] A. Babenko , V. Lempitsky, “ Additive Quantization for Extreme Vector Compression,” Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2014.

[10] T. Zhang, C. Du , J. Wang, “ Composite Quantization for Approximate Nearest Neighbor Search,” 31st International Conference on Machine Learning, ICML, 2014.

[11] T. Zhang, G.-J. Qi, J. Tang , J. Wang, “Sparse Composite Quantization,” Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2015.

[12] X. Wang, T. Zhang, G.-J. Qi, J. Tang , J. Wang, “Supervised Quantization for Similarity Search,” IEEE Conference on Computer Vision and Pattern Recognition, 2016.

[13] T. Zhang , J. Wang, “ Collaborative Quantization for Cross-Modal Similarity Search,” Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016.

[14] D. Xu, I. W. Tsang , Y. Zhang, “Online Product Quantization,” IEEE Transactions on Knowledge and Data Engineering, 2018.

[15] Y. Cao, M. Long, J. Wang, H. Zhu , Q. Wen, “Deep quantization network for efficient image retrieval,” 2016.

[16] Y. Cao, M. Long, J. Wang , S. Liu, “Deep Visual-Semantic Quantization for Efficient Image Retrieval,” 2017.

[17] K. Lang, “NewsWeeder: Learning to Filter Netnews,” International Conference on Machine Learning, 1995.

[18] L. Fei-Fei, R. Fergus , P. Perona, “Learning Generative Visual Models from Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories,” 著: *Conference on Computer Vision and Pattern Recognition Workshop*, 2004.

[19] J. Devlin, M.-W. Chang, K. Lee , K. Toutanova, “ BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” arXiv preprint arXiv:1810.04805, 2018.

[20] A. Friedman, “Framing Pictures: The Role of Knowledge in Automatized Encoding and Memory for Gist,” Journal of Experimental Psychology: General, 1979.