次元縮小射影の量子化像を用いる近似最近傍検索

樋口 直哉[1,a)]　今村 安伸[2,b)]　篠原 武[3,c)]　平田 耕一[3,d)]　久保山 哲二[4,e)]

概要：大規模データセットにおける高精度かつ高速な最近傍検索（NN）のために，長短 2 種類の ANN 索引を用いた 2 重フィルタリングを利用する近似最近傍検索（ANN）を提案する．両索引は，量子化された次元縮小射影の量子化像とみなすことができ，検索する質問データに対する優先度に基づくフィルタリングによって NN 候補が選択される．短い方の索引はスケッチと呼ばれ，本論文では並列処理による優先順位付きスケッチ列挙を用いてフィルタリングを高速化する手法を提案する．スケッチを用いたフィルタリングにより得られた NN 候補は，長い ANN 索引を用いてさらに少ない候補数に絞り込まれる．3 種類の公開大規模データセットに対する標準的なデスクトップ PC を用いた実験を通して，提案する 2 重フィルタリングが効率的な NN 検索を実現し，各質問に対して非常に短い応答時間で高い再現率で回答できることを示す．

Approximate Nearest-Neighbor Search Using Quantized Images of Dimension Reduction Mapping

1. はじめに

1.1 研究の背景と目的

大規模データセットの最近傍検索（NN 検索）は，様々な分野で重要な基盤技術の一つである．例えば，著作権保護の観点から動画共有プラットフォーム上の違法動画を監視するシステムや契約通りにコマーシャルが放送されているかを確認するコマーシャル監視システム，類似画像検索や楽曲識別システムなどがその応用分野である．特に，これらのシステムはリアルタイム性が求められ，高速かつ高精度な検索が不可欠である．

近年の AI 技術の発展により，大規模データセットを直接利用することでシステムの信頼性や汎用性が向上することが分かってきた．一方，パターン認識の分野では，NN 検索の重要性を認識しつつも，高い計算コストを回避するために少数の典型的な例を選択するアプローチが取られることが多い．しかし，統計学習理論や AI 応用の成功例からも分かるように，データセットの多様性と規模を活用することで，未知の入力に対する汎化能力の向上が期待できる．本研究では，代表的な例を少数選択することなく，大規模データセットをそのまま活用する近似最近傍検索（ANN 検索）のアプローチを支える基盤技術の一つとして，効率的で信頼性の高い検索を実現することを目指す．

本研究では，提案手法の実用性や汎用性を考慮し，標準的なデスクトップ PC（例えば，8 コア CPU，128GB メモリ，SSD）を想定する．これにより，高価な専用ハードウェアやクラウド環境を必要とせず，幅広いユーザーが利用できる手法の開発を可能にすることを目指す．また，そのような環境での性能評価を通じて，リソース制約のあるシステムでも実用性が高いことを示す．

1.2 近似最近傍検索

大規模なデータセットで正確な最近傍を見つけるコストは非常に大きいため，他の研究と同様に，ある程度の誤差を許容する近似最近傍検索 (ANN 検索) の可能性を検討する．ANN 検索の精度の定性的な指標として再現率を使用する．本論文では，NN 検索に焦点を当てているため，再

[1] 崇城大学
Sojo University
[2] THIRD Inc
[3] 九州工業大学
Kyushu Institute of Technology
[4] 学習院大学
Gakushuin University
[a)] nac24nh@gmail.com
[b)] imamura.kit@gmail.com
[c)] shino.kyutech@gmail.com
[d)] hirata@ai.kyutech.ac.jp
[e)] kuboyama@tk.cc.gakushuin.ac.jp

現率は，質問に対して真の最近傍が見つかる割合である．

ANN 検索のための索引を ANN 索引という．ANN 索引を用いる方法では，索引によって複数の最近傍候補を選択し，候補中の最近傍を回答として返す，候補の選択を**フィルタリング**という．ANN 索引を使用した検索の**再現率**は，フィルタリングによって選択された候補に真の最近傍が含まれている割合である．

1.3 これまでの研究の進捗

ANN 索引の基礎として，次元縮小射影 S-Map [15] を使用する．**スケッチ**は，コンパクトなビット列であり，多くの研究者によって ANN 索引として議論されてきた [3], [10], [11], [12], [13], [17]．球面分割によるスケッチは，S-Map の量子化像と見なすことができる点に注目し，スケッチを使用したフィルタリングの候補選択基準として従来使用されているハミング距離よりも信頼性の高い非対称距離（後に部分復元距離 $\widetilde{D}_p$ と呼ばれる）を提案した [6]．また，選択基準の照合や比較を行うことなく，選択基準の上位からスケッチを一つずつ列挙することを用いて，より効率的なフィルタリング手法を提案し [4], [5], [9]，大規模なデータセットでも効率的なフィルタリングが実現できることを示した [7]．さらに，並列処理によるフィルタリングの高速化の方法も提案した [8]．しかし，スケッチ列挙によるフィルタリングでは，スケッチのサイズ（**幅**という）が小さいものに限定されており，フィルタリングに必要な候補の数を大幅に削減することはできていない．そのため，フィルタリングで得た候補から最近傍を選択するには大きなコストがかかり，検索プロセス全体の高速化には依然として課題が残されている．

1.4 本論文で得られた結果

本論文では，2 種類の ANN 索引を使用した**2 重フィルタリング**手法を提案する．1 次フィルタリングは，スケッチ列挙による高速フィルタリングを利用する．2 次フィルタリングでは量子化した S-Map 像（QSMAP）を使用するが，スケッチの場合と同様に，部分復元距離 $\widetilde{D}_p$ を QSMAP でも使用する．提案手法の有効性を検証するために，三つの公開データセットを使用する．DEEP1B [2]（96 次元，10 億点），LAION5B [14] の部分集合である LAION100M（768 次元，1 億 200 万点），および YFCC100M [1]（4,096 次元，9700 万点）．

この論文の主な貢献は次の二つである．

- $p = 1$ や $p = 2$ のときの $\widetilde{D}_p$ 順の狭幅スケッチ列挙によるフィルタリングの並列処理による高速化
- 2 重フィルタリングによる ANN 検索の高速化

$p = \infty$ 以外の $p = 1$ または $p = 2$ のときは，$\widetilde{D}_p$ 順の列挙を並列化することが困難であるため，ハミング距離順序列挙を用いた近似列挙を並列処理することによってフィルタリングの高速化を実現した [8]．本論文では，$p = 1$ または $p = 2$ の場合，$\widetilde{D}_p$ 順の列挙を分割し，分割した各列挙に対するフィルタリングを並列処理することで高速化できることを示す．

スケッチを用いた 1 次フィルタリングで得られた候補から最近傍を直接選択するのではなく，S-Map の量子化像である QSMAP を 2 次索引としてフィルタリングすることで候補を絞り込む 2 重フィルタリング法を提案した．

本論文で提案した方法は，標準的なデスクトップ PC（8 コア CPU，128GB RAM）でも高精度かつ高速な NN 検索を実現できることを示した．例えば，DEEP1B に対して提案法を適用すると，16 スレッド並列処理，使用メモリ 52GB で，再現率 80%であれば，1 質問あたりの応答時間 1 ミリ秒，再現率 90%であれば，2 ミリ秒未満で，ANN 検索を行うことができる．

NeurIPS'21 [16] では，DEEP1B のような 10 億点の巨大なデータセットの ANN 検索を高速化する問題にも取り組んでいる．しかし，主な課題は，ANN 索引の精度と検索速度 QPS（Queries Per Second，1 秒あたりに処理できる質問数）であり，記録された最高の検索性能は，DEEP1B で 10,000QPS で再現率が約 70%である．本研究における速度は，ANN 索引によるフィルタリングで見つかった候補の中から質問に最も近い点を選択するプロセスを含む検索の全体的なコストと応答時間である．この規模のデータセットのオンライン検索の応答時間（または応答遅延）に関するレポートが，Data Engineering Bulletin にある．そこでは，1.5TB という極めて巨大な主記憶を搭載した高性能コンピューティング環境でグラフベース手法 [18] を用いた実験の結果では，DEEP1B を 16 スレッドで並列処理して高速化した場合でも，リコール率 90%で応答時間は 1 質問あたり数十ミリ秒以上である．以上により，本論文で提案する方法は，高速かつメモリ効率が良いといえる．

2. 準備

2.1 距離空間での最近傍検索

この論文で使用する次元縮小射影とスケッチは，検索対象が距離空間内の有限の点集合であると想定している．$\mathcal{U}$ を距離空間とし，距離関数を D とする．この論文で提案する手法の有効性を確認するための実験で使用されているすべてのデータセットはユークリッド空間のものであるが，この論文で使用する索引技法はユークリッド空間の仮定を必要としないことを強調しておく．

この論文の主な課題は，距離空間での ANN 検索を高速化することである．データセット ds からの NN 検索は，指定された質問点 $q \in \mathcal{U}$ に対する ds 内の最近傍，つまり $\mathrm{argmin}_{x \in ds} D(q, x)$ を見つけることである．しかし，$\mathcal{U}$ が高次元で ds が巨大な場合，最近傍を正確かつ迅速に見つけることは困難である．そのため，最近傍を正確に見つけ

る必要はなく，ある程度の再現率で最近傍を見つけることが求められる ANN 検索を使用する．ここで，**再現率**とは，正しい最近傍が見つかる割合である．本論文で使用する主な概念と表記法を表 1 に示す．

2.2 次元縮小射影 S-Map とスケッチ

ANN は，効率的な検索のために距離情報をある程度保持しながら，次元縮小による点のコンパクトな表現を索引として使用する．代表的な次元縮小手法である主成分分析（PCA）は，ユークリッド空間を前提としている．本論文では，S-Map [15] を基本とする．S-Map 自体は，PCA とは異なり，座標表現のない非ユークリッド空間や距離空間にも適用できる．S-Map のピボットは，任意の点 $c \in \mathcal{U}$ であり，ピボット c による点 x の S-Map 像 $\varphi_c(x)$ は，x と c 間の距離として定義される．

$$\varphi_c(x) = D(x, c)$$

一般に，m 次元への S-Map は，m 個のピボット $\Pi = (c_0, \ldots, c_{m-1})$ を使用して定義される．

$$\varphi_\Pi(x) = (\varphi_{c_0}(x), \cdots, \varphi_{c_{m-1}}(x))$$

本論文では，S-Map の量子化像を使用して，S-Map 像の表現サイズを圧縮する．

スケッチ は，点をビット文字列としてコンパクトに表現したもので，厳密には局所性敏感ハッシュ（LSH）ではないが，LSH のような特性を持ち，ANN 索引として使用される．スケッチを使用した検索では，点間の距離関係をスケッチ間の距離で近似して ANN 検索を実現する．ハミング距離は，スケッチ間の標準的な距離である．球面分割によって作成されたスケッチは，S-Map の射影像の各次元を 1 ビットに量子化したものと考えることができること，また，ハミング距離の代わりに質問点 q とスケッチ ς 間の非対称な部分復元距離 $\widetilde{D}_p(q, \varsigma)$ を使用すると，ハミング距離を使用するものよりも再現率の高い ANN 検索が実現できることを示した [6]．同様の非対称距離は，Dong ら [3] によっても導入されている．この非対称距離は，量子化によって失われた距離情報を部分的に復元したものと見なすことができる．

ここでは，1 次元の例を使用して，部分復元距離の原理を説明する．a と b を区間 $[0, 2]$ 内の実数とする． a と b の間の距離は $|a - b|$ であり，二つの量子化された整数間の距離は，$||\lfloor a \rfloor - \lfloor b \rfloor||$ である．量子化された a と元の b の間の距離は，$||\lfloor a \rfloor - b|$ である．$a = 1.2$，$b = 0.9$ の場合，$|a - b| = 0.3$，$||\lfloor a \rfloor - \lfloor b \rfloor|| = 1$ となり，量子化によって生じる誤差は 0.7 となる．一方，$||\lfloor a \rfloor - b| = 0.1$ であるので，量子化によって生じる誤差は，0.2 に減少する．部分復元距離 $\widetilde{D}_p$ は，$p \geq 1$ のとき，Minkowski の L_p 距離と同様に，スケッチの各ビットについて得られた部分復元距離を集計して計算される．スケッチなどの圧縮表現を索引として使用する検索では，データセット内の点の元の特徴データは大きいため，2 次記憶に残し，圧縮表現を主記憶に読み込んで使用する．しかし，検索の質問については，圧縮表現だけでなく元の特徴データも使用できる．これを考慮すると，部分復元距離を検索に使用して，より多くの距離情報を使用したより正確な（より高い再現率の）検索を可能にすることができる．

本論文では，スケッチと量子化 S-Map 像（QSMAP）の 2 種類の索引を使用した 2 重フィルタリングを提案する．スケッチには，20 ビットや 24 ビットなどの比較的短い長さ，狭い幅を使用する．スケッチを用いた 1 次フィルタリングにより，候補は素早く大まかに選択される．2 次フィルタリングでは，1 次フィルタリングで得られた候補を少数の候補に絞り込む．2 次フィルタリングでは，スケッチより大きく，元の特徴データより小さい QSMAP を使用する．QSMAP では，スケッチと同様に，部分的に復元された距離 $\widetilde{D}_p$ を使用できる．

2.3 スケッチ列挙によるフィルタリング

16 ビットや 20 ビットなどの幅が狭いスケッチの場合，質問とスケッチを比較せずに，質問への優先順にスケッチを列挙することでフィルタリングを高速化できることを示した [4], [5]．優先順位については，ハミング距離や $\widetilde{D}_1$，$\widetilde{D}_2$，$\widetilde{D}_\infty$ のいずれも列挙に使用できる．

列挙によるフィルタリングは，あまり馴染みのない方法であるため，Algorithm 1 で概要を説明する．列挙法を効率的に使用するために，データセット内の点は，スケッチ順にソートされていると仮定する．

$$ds = \{x_0, \ldots, x_{n-1}\}, \sigma(x_0) \leq \cdots \leq \sigma(x_{n-1})$$

スケッチ順での番号 i を点 x_i の参照番号として用いる．スケッチ ς に対して，スケッチをキーとするバケット表を使用すれば，ς を持つ点にすばやくアクセスすることができる．狭幅スケッチのビット列は，整数として扱うことができるため，バケット表は，大きさ 2^w の非負整数配列 bkt によってあらわすことができる．そこでは，ς を持つ点が存在するとき，

$$bkt[\varsigma] = \text{スケッチ順の最初の位置},$$

そうでないとき，

$$bkt[\varsigma] = bkt[\varsigma + 1]$$

となるようにしておく．そうすると，ς を持つ点が存在する場合，それらは，

$$x_{bkt[\varsigma]}, \ldots, x_{bkt[\varsigma+1]-1}$$

となる．スケッチ ς を持つ点が存在しない場合は，つまり，ς のバケットが空の場合は，

$$bkt[\varsigma] = bkt[\varsigma + 1]$$

になっている．実際の検索では，スケッチ全体のごく一部のみが列挙されるため，フィルタリングが非常に高速になることに注意されたい．

表 1 表記法

Notation	Description
$\mathcal{U}$	距離空間
$x, y, x_0, \ldots$	$\mathcal{U}$ 内の点
$D(x, y)$	x と y 間の距離
$ds \subseteq \mathcal{U}$	データセット $\{x_0, x_1, \ldots, x_{n-1}\}$（スケッチ順に並んでいる仮定する）
$q \in \mathcal{U}$	質問
$k_{1^{\text{st}}}$, $k_{2^{\text{nd}}}$	1 次および 2 次フィルタリングで求める候補数
w	スケッチの幅（長さ，ビット数）
$\sigma(x), \varsigma$	x のスケッチ, 不特定点のスケッチ
$e_i(q, \varsigma), e_i$	ς の第 i ビットの分割球と q の最小距離，距離下限
$\widetilde{D}_p(x, \varsigma)$	x と ς 間の部分復元距離
$a \oplus b$	a と b のビット毎の排他的論理和

// q, k, bkt: 質問，選択する候補数，バケット表
function FilteringBySketchEnumeration(q, k)
  q への優先順のスケッチ列挙を開始する;
  $C \leftarrow \emptyset$;
  repeat
    $\varsigma \leftarrow$ 列挙のつぎのスケッチ;
    if $bkt[\varsigma] < bkt[\varsigma + 1]$ **then**
      for $i = bkt[\varsigma]$ **to** $bkt[\varsigma + 1] - 1$ **do**
        $C \leftarrow C \cup \{i\}$;
  until $|C| \geq k$;
  return C;

Algorithm 1: スケッチ列挙によるフィルタリング

2.4 量子化 S-Map によるフィルタリング

2 重フィルタリングの 2 次フィルタリングでは，スケッチよりも大きく選別能力の高い QSMAP を ANN 索引として用いて，1 次フィルタリングで得られた候補を絞り込む．前の小節で説明したように，データセット内の点はスケッチ順に並んでいると仮定されている．したがって，1 次候補中にスケッチが同じ点が複数ある場合，スケッチ順の参照番号が連続している．2 次フィルタリングでは，1 次候補に対して質問に対する QSMAP の優先度 $\widetilde{D}_p$ を計算し，その上位 $k_{2^{\text{nd}}}$ 個を選択する．このため，データセットの QSMAP もスケッチ順にソートして，主記憶に読み込んでおく．

3. スケッチと QSMAP を用いた 2 重フィルタリング

2 重フィルタリングを使用した ANN 検索の概要を Algorithm 2 に示す．ここで，与えられた質問は，q であり，1 次フィルタリングで，狭幅スケッチを使用して中間的に取得される候補の個数は，$k_{1^{\text{st}}}$ である．事前に準備する必要がある主なものは，スケッチによるバケット表 bkt とスケッチ順に並べた QSMAP 像である．

function NNSearchByDoubleFiltering(q, $k_{1^{\text{st}}}$, $k_{2^{\text{nd}}}$)
  $C_1 \leftarrow$ FilteringBySketchEnumeration($q, k_{1^{\text{st}}}$);
  C_1 から $\widetilde{D}_p(q, \cdot)$ が小さい $k_{2^{\text{nd}}}$ 個の候補を C に求める;
  return $\text{argmin}_{x \in C} D(q, x)$;

Algorithm 2: 2 重フィルタリングによる ANN 検索

3.1 並列処理による高速化

まず，並列処理による 1 次フィルタリングの高速化について考えよう．質問 q からスケッチ ς の i 番目のピボットによる分割球面境界までの最短距離を $e_i(q, \varsigma)$ とする．分割球の中心と半径を c_i と r_i すると，$e_i(q, \varsigma)$ は，つぎで求めることができる．

$$e_i(q, \varsigma) = |D(q, c_i) - r_i|$$

この $e_i(q, \varsigma)$ は，質問のスケッチ $\sigma(q)$ とスケッチ ς の i 番目のビットが不一致であるときの q とスケッチが ς である任意の点 x 間の距離の下限になっている．つまり，

$$(\forall x \in \mathcal{U} \text{ such that } \sigma(x) = \varsigma)\ e_i(q, \varsigma) \leq D(q, x)$$

であるので，$e_i(q, \varsigma)$ を第 i ビットの**距離下限**と呼ぶことにする．文脈で q と ς が明らかなときは，単に e_i と略記する．距離下限を集計したものが部分復元距離である．$\widetilde{D}_p$ 順の列挙自体を並列化することは，$p = 1$ や 2 のときは難しいため，列挙されたスケッチを，最も低い low 個の距離下限に対応する low ビットによって，2^{low} 個のクラスに分類する．ここでは，簡単のために，$e_0, e_1, \ldots$ が最下位ビットから最上位ビットまで昇順に並んでいると仮定するが，実際の実装では，距離下限の順序を用いた相対位置のビット操作が必要である．

例として，$w = 4, low = 2, (e_3, e_2, e_1, e_0) = (3, 2, 2, 1)$ であるときの $\widetilde{D}_1$ 順の列挙を考えよう．このとき，$\widetilde{D}_1(q, \varsigma)$ は，質問のスケッチ $\sigma(q)$ と ς の不一致ビットに対応する e_i を足し合わせたものである．質問 q への $\widetilde{D}_1$ 距離で順序付けられたスケッチの列挙を考えることは，図 1 の Entire enumeration に示しているように，$\sigma(q)$ と ς の排他論理和 $\sigma(q) \oplus \varsigma$ の列挙を考えることと同等になる．$\sigma(q) \oplus \varsigma$ の最下位 2 ビットは，$\widetilde{D}_1$ 順で 00，01，10，11 である．最下位

2 ビットで分類すると，図の Class 0，...，Class 3 のようになる．最上位 2 ビットは全クラスに共通で，00，01，10，11 であることに注意されたい．このように，最下位 2 ビットで分類された 2^2 個のクラスの列挙は，最上位 2 ビットの $\widetilde{D}_1$ 順のビット列と分類に対応する最下位 2 ビットのビット列との排他論理和で与えられる．ただし，部分列挙の先頭は，列挙全体の先頭と異なる場合があることに注意されたい．つまり，このように分類された列挙の先頭は，列挙全体の先頭の近似にすぎない．ただし，共通の最上位 2 ビットパターンの数は列挙全体の $\frac{1}{2^2}$ であるため，列挙によるフィルタリングの全体的なコストは小さくなる．

ここでは，Algorithm 3 に示した実装の詳細を説明する．下位の距離下限に対応する low ビットの部分が 2^{low} 通りあり，それらの β_0，...，$\beta_{2^{low}-1}$ で分類し，共通部分の上位 w–low ビットのみを変更した列挙 $\gamma_0, \gamma_1, \ldots$ を用いて，各部分で並列にフィルタリングを行う．しかし，列挙されたスケッチの点の数は一定ではなく，点が全くない場合もあり，同じ数の共通部分列挙を用いても，分類された部分ごとに得られる候補の数は異なる．全体としてフィルタリングに必要な候補の数が目標数 $k_{1^{\mathrm{st}}}$ に達したかどうかを監視しながら，効率的に並列にフィルタリングを行うことは難しい．そこで，共通部分の列挙を複数（Algorithm 3 では ls）取得し（8 行），それらを使用して各部分のフィルタリングを並列実行する（11 行から 14 行のループ）．目標よりも多くの候補が取得された場合，超過した部分を縮小する．この場合，最大の $\widetilde{D}_1$ を持つものから削除する（17 行）．また，各部分の候補数を均一にして，2 次フィルタリングの並列処理を効率的にする（18 行）．2 次フィルタリングを 1 次フィルタリングに織り込むと無駄が多くなるため，1 次フィルタリングの後に 2 次フィルタリングを実行する．1 次フィルタリングと 2 次のフィルタリングの間で渡される候補の表現サイズを削減する方が効率的であるため，個々のデータ参照のリストではなく，同じスケッチを持つ連続したデータ参照を表す区間のリストを候補表現として使用する（14 行）．このようにすることで，1 次フィルタリングで使用するスケッチの幅は比較的に小さく，同じスケッチを共有する点が複数あることが多いため，候補表現のサイズを削減できる．ここで注目すべきは，各部分で見つかった候補の区間が $\widetilde{D}_p$ 順に並んでいることを利用することで，不要な候補の除去と候補数のバランスをとるという作業を全体のフィルタリングコストに比べて比較的小さなコストで実行できることである．また，Algorithm 3 では，候補を表すリストに，列挙されたスケッチ ς の優先度 $\widetilde{D}_p(q,\varsigma)$ と区間 $(bkt[\varsigma], bkt[\varsigma+1]-bkt[\varsigma])$ が含まれているが，$\widetilde{D}_p$ 順に並んでいることを利用して実装すれば，優先度を不要にすることができる．

3.2 索引サイズの選択

スケッチ幅 w を大きくすると選別能力が高まり，目標再現率を達成するために必要な候補数が減る．ただし，スケッチ列挙による 1 次フィルタリングでは，個々のスケッチを持つ点の数が減るため，訪問するスケッチの数が増える．2 次フィルタリングで参照する QSMAP 像をスケッチ順に並べておくと，同じスケッチを持つ候補へのアクセスコストを削減できるが，スケッチ幅を大きくするとこの効果は減少する．

QSMAP サイズが大きいほど選別能力は高くなるが，主記憶に格納できる範囲内である必要があり，部分復元距離の計算コストも増加する．そのため，選別能力とフィルタリングコストのトレードオフを考慮して QSMAP サイズを慎重に選択する必要がある．

4. 実験

この節では，提案する ANN 検索手法を使用した実験結果を示す．特徴データは，8 ビット整数で表現するように変換したものを用いる．部分復元距離 $\widetilde{D}_p$ を 1 次と 2 次の両方のフィルタリングで使用する．1 次フィルタリングでは，$p=1$，2 次フィルタリングでは，$p=2$ である．いずれも，いろいろな p を用いた実験により，選別能力が高くなる p を選んだ．

4.1 データセットとコンピューター環境

三つのデータセットを使用する．点数，次元，大きさ，質問数を表 2 に示す．

実験には，標準的デスクトップ PC を使用する: CPU AMD Ryzen 7 3700X 3.6GHz，チップセット B550，RAM DDR4 128GB，SSD INTEL M.2 2TB．128GB の RAM が実装されているが，実験では，64GB 以下しか使用されていない．

4.2 2 重フィルタリングによる最近傍検索の応答時間

検索応答時間（応答遅延）の実験結果を示す．前述のように，1 次フィルタリングにおけるスケッチ幅と 2 次フィルタリングにおける QSMAP のサイズは慎重に選択する必要がある．ここでは，本研究の実験環境でほぼ最適であると考えられる組み合わせを使用した結果のみを示す．

2 重フィルタリングを使用した最近傍検索の応答時間は，1 次フィルタリングコストと 2 次フィルタリングコスト，最近傍選択コストの合計によって決まる．ここで，フィルタリングによって得られた候補から最近傍を選択する際，特徴データは，2 次記憶に置いているものと想定する．このような 2 次記憶からの検索コストの実測値を安定して得ることは，ディスクキャッシュの影響を受けるため非常に困難である．そこで，ディスクキャッシュによる高速化の影響が多少あると考えられる実測値とともに，別途測定し

Entire enumeration
$(e_3, e_2, e_1, e_0) = (3, 2, 2, 1)$

i	$\sigma(q) \oplus \varsigma_i$	$\widetilde{D}_1(q, \varsigma_i)$
0	0000	0
1	0001	$e_0 = 1$
2	0010	$e_1 = 2$
3	0100	$e_2 = 2$
4	0011	$e_1 + e_0 = 3$
5	0101	$e_2 + e_0 = 3$
6	1000	$e_3 = 3$
7	0110	$e_2 + e_1 = 4$
8	1001	$e_3 + e_0 = 4$
9	0111	$e_2 + e_1 + e_0 = 5$
10	1010	$e_3 + e_1 = 5$
11	1100	$e_3 + e_2 = 5$
12	1011	$e_3 + e_1 + e_0 = 6$
13	1101	$e_3 + e_2 + e_0 = 6$
14	1110	$e_3 + e_2 + e_1 = 7$
15	1111	$e_3 + e_2 + e_1 + e_0 = 8$

Classified by lowest 2-bits

Class 0 ($\beta_0 = 00$)

i	$\sigma(q) \oplus \gamma_i \oplus \beta_0$	$\widetilde{D}_1(q, \varsigma_i)$
0	0000	0
1	0100	2
2	1000	3
3	1100	5

Class 1 ($\beta_1 = 01$)

i	$\sigma(q) \oplus \gamma_i \oplus \beta_1$	$\widetilde{D}_1(q, \varsigma_i)$
0	0001	1
1	0101	3
2	1001	4
3	1101	6

Class 2 ($\beta_2 = 10$)

i	$\sigma(q) \oplus \gamma_i \oplus \beta_2$	$\widetilde{D}_1(q, \varsigma_i)$
0	0010	2
1	0110	4
2	1010	5
3	1110	7

Class 3 ($\beta_3 = 11$)

i	$\sigma(q) \oplus \gamma_i \oplus \beta_3$	$\widetilde{D}_1(q, \varsigma_i)$
0	0011	3
1	0111	5
2	1011	6
3	1111	8

図 1 距離下限の下位 2-bit による列挙の分類

```
// q: 質問, k: 選択する候補数, low: 分類のためのビット数
// ls: まとめて列挙する共通部分のスケッチ数, bkt: バケット表
// C[0], ..., C[2^low − 1]: 候補を表す優先度付き区間の集合, |C|: C の総候補数
1  function FilteringBySketchEnumerationInParallel(q, k)
2      距離下限の下位 low ビットのパターン β_0, ..., β_{2^low − 1} を計算する;
3      距離下限の下位 low ビットを固定した q に対する D̃_p 順のスケッチ列挙を開始する;
4      for t = 0 to 2^low do
5          C[t] ← ∅;
6      i ← 0;
7      repeat
8          つぎの ls 個の列挙 γ_i, ..., γ_{i+ls−1} を求める;
9          do in parallel
10             t ← スレッド番号;
11             for j = 0 to ls − 1 do
12                 ς ← β_t ⊕ γ_{i+j};
13                 if bkt[ς] < bkt[ς + 1] then
14                     C[t] ← C[t] ∪ {(D̃_p(q, ς), (bkt[ς], bkt[ς + 1] − 1))};
15         i ← i + ls;
16     until |C| ≥ k;
17     |C| < k でない限り, 最大の D̃_p の区間を取り除く;
18     C[0], ..., C[2^low − 1] の候補数が均一になるように区間を移動する;
19     return C;
```

Algorithm 3: スケッチ列挙による並列フィルタリング

表 2 データセット

データセット	点数	次元	大きさ	質問数
YFCC100M	0.97×10^8	4,096	400GB	5,000
DEEP1B	1.00×10^9	96	100GB	10,000
LAION100M	1.02×10^8	768	77GB	10,000

た 2 次記憶に置いた特徴データから最近傍を検索するランダムアクセス時のコストを用いた予測値を示す．

フィルタリングで求める候補数については，$k_{2^{\text{nd}}}$ が小さいほど検索が高速になるわけではない．これは，フィルタリングコストと候補から最近傍を選択するコストのバランスによる．候補数 $k_{2^{\text{nd}}}$ が多いほどリコールは高くなるが，候補からの最近傍検索コストは高くなる．また，1 次フィルタリングで選択する候補数 $k_{1^{\text{st}}}$ についても，$k_{1^{\text{st}}}$ を増やすとリコールは高くなるが，2 次フィルタリングコストは高くなる．そのため，適切な $k_{1^{\text{st}}}$ と $k_{2^{\text{nd}}}$ の組み合わせを使う必要がある．表 3 は，実験によって選んだほぼ最適と考えられる組み合わせを使った場合の結果である．フィルタリングコストの 1st と 2nd の列には，それぞれ 1 次と 2 次のフィルタリングのコストを 1 質問当たりのミリ

秒 (ms/q) で示す．NN-search の列には，フィルタリングによって選択された候補から最近傍を選択するコストが含まれている．Total 列には，応答時間（応答遅延，latency）を表す検索全体のコストが含まれている．NN-search* 列と total* 列には，予測値が含まれている．実測値と予測の差は，データセットによって違いはあるものの，それほど大きくない．

DEEP1B は，三つのデータセットの中で点の数が最大であるが，次元数は最小である．DEEP1B に対する検索応答時間は，再現率 70%，80%，90% の場合，それぞれ，1 質問当たりのミリ秒 (ms/q) で，実測値と予測値ともに，0.66，0.90，1.8 である．LAION100M の点の数は，DEEP1B の 10 分の 1 で，次元数は約 7 倍である．LAION100M に対する検索応答時間は，DEEP1B よりも少し遅くなっている． YFCC100M は，LAION100M とほぼ同じ点の数であるが，次元数は DEEP1B の約 40 倍である．検索応答時間は，再現率が 70%，80%，90%の場合，それぞれ，実測値 (ms/q) で，4.5，8.4，19 と大幅に遅くなっている．それでも，再現率が 90%の場合，1 秒あたり約 50 質問に応答できる速さである．

本研究の速度に関する課題は，リアルタイム性にとって重要なオンライン検索における応答時間（応答遅延）である．表 3 に示した応答時間は，全質問の平均である．リアルタイム性を重視している場合には，平均だけでなく，ばらつきが問題となるので，表には標準偏差も示した．

4.3 ANN 検索の解の品質

ANN 検索で得た解は，最近傍でない可能性がある．しかし，ANN の適用問題によっては，得られた解が最近傍でなくても，ある程度近いものが求められていれば十分であることがある．ANN 検索で得られた解が最近傍である割合が再現率である．最近傍でなくても k 近傍に含まれている割合を recall$^+$@k として，表 3 で示した目標の再現率となる候補数 $k_{1^{\text{st}}}$ および $k_{2^{\text{nd}}}$ の設定での検索結果に対する recall$^+$@k を図 2 に示す．この図から得られた解が最近傍でなくても，6 近傍以内であればよいのであれば，再現率が 70%の検索結果でも，90%を超える品質であることが分かる．

5. おわりに

本論文では，長い ANN 索引と短い ANN 索引の 2 種類を用いた 2 重フィルタリングによる近似最近傍検索法を提案し，三つの公開大規模データセットを使用した実験を通じてその有効性を確認した．主目標は，ANN 索引の効率だけでなく，ANN 索引によって得られた複数の候補から最近傍を選択するコストを含む検索全体の応答速度を向上させることであり，オンライン質問処理の応答速度に焦点を当てている．

オンライン質問処理においてリアルタイム性が求められる場合には，応答速度の平均だけではなく，そのばらつきが小さい必要がある．本論文で提案した手法の質問応答速度のばらつきは，それほど大きいものではないが，さらに均質な応答時間となる工夫が必要である．重要な今後の課題の一つである．

フィルタリングで使用する 2 種類の索引は，いずれも次元縮小射影 S-Map に基づいており，その性能は使用するピボットによって大きく左右される．S-Map のピボット選択には，PCA のように確立された解析的手法が存在しないため，データセットと質問のサンプルを使用した発見的手法で最適化されている．スケッチと QSMAP，ピボット選択，射影法などの詳細については，本論文の範囲外とし，別の機会に報告する．

参考文献

[1] Amato, G., Falchi, F., Gennaro, C. and Rabitti, F.: YFCC100M-HNfc6: A Large-Scale Deep Features Benchmark for Similarity Search, *Proc. SISAP'16, LNCS 9939, Springer*, pp. 196–209 (2016).

[2] Babenko, A. and Lempitsky, V.: Efficient Indexing of Billion-Scale Datasets of Deep Descriptors, *Proc. CVPR'16, IEEE Computer Society*, pp. 2055–2063 (2016).

[3] Dong, W., Charikar, M. and Li, K.: Asymmetric distance estimation with sketches for similarity search in high-dimensional spaces, *Proc. ACM SIGIR'08*, pp. 123–130 (2008).

[4] Higuchi, N., Imamura, Y., Kuboyama, T., Hirata, K. and Shinohara, T.: Fast Nearest Neighbor Search with Narrow 16-bit Sketch, *Proc. ICPRAM'19*, pp. 540–547 (2019).

[5] Higuchi, N., Imamura, Y., Kuboyama, T., Hirata, K. and Shinohara, T.: Fast Filtering for Nearest Neighbor Search by Sketch Enumeration Without Using Matching, *Proc. AusAI'19, LNCS 11919, Springer*, pp. 240–252 (2019).

[6] Higuchi, N., Imamura, Y., Kuboyama, T., Hirata, K. and Shinohara, T.: Nearest neighbor search using sketches as quantized images of dimension reduction, *Proc. ICPRAM'18*, pp. 356–363 (2018).

[7] Higuchi, N., Imamura, Y., Mic, V., Shinohara, T., Hirata, K. and Kuboyama, T.: Nearest-Neighbor Search from Large Datasets Using Narrow Sketches, *Proc. ICPRAM'22*, pp. 401–410 (2022).

[8] Higuchi, N., Imamura, Y., Mic, V., Shinohara, T., Hirata, K. and Kuboyama, T.: Fast Filtering for Similarity Search Using Conjunctive Enumeration of Sketches in Order of Hamming Distance, *Proc. ICPRAM'24*, pp. 499–510 (2024).

[9] 樋口直哉，今村安伸，久保山哲二，平田耕一， 篠原武：狭い 16 ビットのスケッチを用いた高速最近傍検索，情報処理学会論文誌「数理モデル化と応用」，pp. 13–22 (2020).

[10] Lv, Q., Josephson, W., Wang, Z. and and K. Li, M. C.: Efficient filtering with sketches in the ferret toolkit, *Proc. MIR'06*, pp. 279–288 (2006).

[11] Mic, V., Novak, D. and Zezula, P.: Improving sketches for similarity search, *Proc. MEMICS'15*, pp. 45–57 (2015).

表 3　2 重フィルタリングによる ANN 検索の性能

データセット	再現率 (%)	候補数 $k_{1^{st}}$ ($\times 10^6$)	候補数 $k_{2^{nd}}$	フィルタリング 1st (ms/q)	フィルタリング 2nd (ms/q)	NN-search (ms/q)	Total (ms/q)	Total 標準偏差	NN-search* (ms/q)	Total* (ms/q)
DEEP1B	70	0.13	15	0.055	0.45	0.063	0.66	0.66	0.068	0.66
	80	0.24	20	0.072	0.65	0.090	0.90	0.57	0.090	0.90
	90	0.62	40	0.150	1.40	0.170	1.80	0.88	0.180	1.80
LAION100M	70	0.08	20	0.039	0.44	0.072	0.61	0.51	0.140	0.67
	80	0.19	40	0.058	0.82	0.120	1.10	0.61	0.280	1.20
	90	0.60	100	0.140	2.20	0.280	2.70	0.85	0.700	3.20
YFCC100M	70	0.33	50	0.160	3.90	0.230	4.50	0.87	0.900	5.20
	80	0.64	90	0.280	7.50	0.350	8.40	1.10	1.600	9.60
	90	1.50	200	0.700	17.00	0.580	19.00	2.10	3.600	22.00

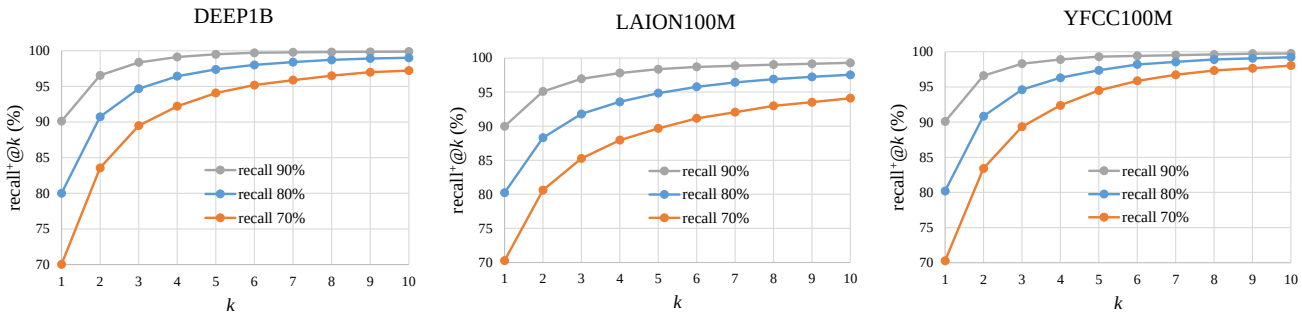

図 2　ANN の解の品質

[12] Mic, V., Novak, D. and Zezula, P.: Speeding up similarity search by sketches, *Proc. SISAP'16*, pp. 250–258 (2016).
[13] Müller, A. and Shinohara, T.: Efficient similarity search by reducing I/O with compressed sketches, *Proc. SISAP'09*, pp. 30–38 (2009).
[14] Schuhmann, C., Beaumont, R., Vencu, R., Gordon, C., Wightman, R., Cherti, M. and Jitsev, . J.: LAION-5B: An open large-scale dataset for training next generation image-text models, *arXiv preprint arXiv:2210.08402* (2022).
[15] Shinohara, T. and Ishizaka, H.: On dimension reduction mappings for approximate retrieval of multi-dimensional data, *Progress of Discovery Science, LNCS 2281, Springer*, pp. 89–94 (2002).
[16] Simhadri, H. V., Williams, G., Aumüller, M., Douze, M., Babenko, A., Baranchuk, D., Chen, Q., Hosseini, L., Krishnaswamy, R., Srinivasa, G., Subramanya, S. J. and Wang, J.: Results of the NeurIPS'21 Challenge on Billion-Scale Approximate Nearest Neighbor Search, *CoRR*, Vol. abs/2205.03763 (online), DOI: 10.48550/arXiv.2205.03763 (2022).
[17] Wang, Z., Dong, W., Josephson, W., Q. Lv, M. C. and Li, K.: Sizing sketches: A rank-based analysis for similarity search., *Proc. ACM SIGMETRICS'07*, pp. 157–168 (2007).
[18] Zhang, M., Ren, J., Peng, Z., Jin, R., Li, D. and Ren, B.: iQAN: Fast and Accurate Vector Search with Efficient Intra-Query Parallelism on Multi-Core Architectures, *IEEE Data Eng. Bull.*, Vol. 46, No. 3, pp. 22–38 (online), available from ⟨http://sites.computer.org/debull/A23sept/p22.pdf⟩ (2023).