**Анотация**

Одной из востребованных областей поиска с Интернете является поиск по изображениям. Современные методы индексации данных показывают отличный результат поиска в многомиллиардных коллекциях и повсеместно применяются для данной задачи. В этой этой работе предлагается изучить существующие подходы индексации больших коллекций изображений и посмотреть на поведение этих алгоритмов в задаче распознавания лиц. Будем считать, что современные нейросетевые алгоритмы достаточно хорошо выделяют признаки лица на изображении и лишь немного затронем этот вопрос. Основной задачей ставим сравнение скорости поиска индексных методов на наборах данных лиц. Начнем рассмотрение с простейших алгоритмов поиска, таких как поиск по точному расстоянию и простая индексная структура. Затем на более продвинутых алгоритмах будем увеличивать скорость поиска и исследовать ухудшение точности. Эксперименты показывают, что в больших коллекция изображений лиц можно добиться приемлемой скорости поиска с хороших показателем точности. Сильное же ускорение приводит к большим потерям.

**Введение**

В последнее десятилетие визуальный поиск стал широко распространенной функцией многих поисковых систем. В связи с этим эффективный поиск ближайших соседей стал серьезной исследовательской проблемой [1-6]. Потребность быстрого поиска похожих изображений занимает большую нишу в современных приложениях компьютерного зрения, в том числе в задаче распознавания лиц [8]. Социальные сети, правоохранительные органы имеют огромные коллекции изображений лиц, среди которых надо уметь быстро извлекать нужную информацию. Для решения данной проблемы требуются эффективные и масштабируемые алгоритмы поиска с низкими временными затратами. Ожидается, что ответ на запросы к базам данных из миллиардов элементов будет занимать несколько миллисекунд.

Решение данной задачи можно рассматривать с двух сторон. Во-первых, даже самые современные алгоритмы поиска и обработки лиц на изображении не идеальны. Это открывает просторы для исследований. Во-вторых, подходящая структура данных для поиска может давать многократное увеличение скорости. Среди всех алгоритмов распознавания сверточные нейронные сети (CNN) показывают лучшие результаты поиска лиц на изображении и используются в большинстве исследованиях этой области [8,10].В связи с этим основной задачей исследования будем считать проблему выбора поисковой структуры данных, а дескрипторы лиц для экспериментов будем строить по одной из общедоступных CNN.

Все существующие крупномасштабные поисковые системы избегают исчерпывающего поиска путем ограничения конечного набора кандидатов, который рассматривается для запроса. Данный подход называют приближенным поиском ближайших соседей (ANN). Современные алгоритмы ANN имеют три основных реализации: инвертированная индексация [1-6], хеширование [9], многомерная инвертированная индексация основанная на квантовании произведения (PQ) [4,5]. В этой работе основное внимание будет уделено инвертированному индексу и его оптимизации с помощью PQ.

Структуры индексации разбивают пространство поиска на большое количество непересекающихся областей, и в процессе поиска используются только малая часть коллекции, наиболее близкая к конкретному запросу. Отобранная часть данных образует короткий список кандидатов, и поисковая система исчерпывающе рассчитывает расстояния между запросом и всеми кандидатами. На этом этапе важно, чтобы список кандидатов был коротким, так как вычисление расстояния имеет линейную сложность по данной длине. Метод PQ для ANN используется в двух видах: для построения многомерного инвертированного индекса для приближенного поиска или для кодирования векторов в компактные коды для точного поиска. Идея этих подходов состоит в том, чтобы разложить пространство векторов на большое количество подпространств и обучить запросы получать доступ к ближайшим подпространствам.

Первая структура индексации, способная работать с миллиардным набором данных, была представлена в [1]. Она была основана на структуре инвертированного индекса, которая разбивает пространство признаков на диаграмму Вороного. Каждая область задается свои центроидом, который предварительно обучили алгоритмом K-средних. Было показано, что эта система достигает разумных скоростей поиска порядка нескольких десятков миллисекунд. Позже обобщение структуры инвертированного индекса было предложено в [3]. В этой работе был представлен инвертированный многомерный индекс или мультииндекс (IMI), который разбивает пространство признаков на несколько ортогональных подпространств и разбивает каждое подпространство на диаграммы Вороного независимо. Декартово произведение такого разбиения образует неявное разбиение всего пространства поиска. Обе эти структуры обладают своими недостатками, которые можно устранить с помощью различных оптимизаций PQ [6,7].

В данной работе будет описано несколько современных архитектур индексирования и путем экспериментов исследована применимость данного подходя для задачи распознавания лиц.

**Постановка задачи**

Главной задачей данной работы является проверка современных индексных структур быстрого поиска в задаче распознавания лиц. Считается, что выделением лица на изображении и построением его признаков занимаются алгоритмы общедоступных сверточных нейронных сетей. Работа будет производится в рамках заранее подготовленных 128-мерных векторов, по одному для каждого изображения. Время построения каждого вектора учитывать не будем, так как во всех алгоритмах будет использоваться одна и та же нейронная сеть. Также не будем учитывать, но обратим внимание на время обучения индексных структур, так как для разных алгоритмов оно может отличаться на порядки.

Проверка заключается в измерении среднего времени поиска похожих лиц в большой коллекции изображений. Время, затраченное на поиск k (k = 1, 5, 10, 30, 50, 100) ближайших соседей будет основным критерием скорости алгоритма. Помимо скорости надо учитывать точность поиска. Во всех приведенных статьях точность измерялась как процент истинных ближайших соседей среди k найденных, где истинные ближайшие соседи определялись точным евклидовым расстоянием. В связи со спецификой нашей задачи, наиболее правильным вариантом измерения точности будет получение процента лиц, отмеченных так же, как и запрос, среди k найденных ближайших соседей.

Для применимости данного подхода к реальным задачам требуемая скорость поиска не должна превышать нескольких миллисекунд. А точность должна быть в пределе допустимой для выбранного алгоритма, то есть не сильно отличаться от приводимой в статьях. В качестве вывода следует оценить эти показатели и сформулировать возможные пути улучшения.

**Обзор существующих решений**

В данной части работы будут кратко описаны несколько идей из существующих работ и попутно введены обозначения для дальнейшего исследования.

Начнем с того, что во всех упомянутых ниже алгоритмах решается одна и та же задача. А именно, вычисление евклидовых расстояний между векторами высокой размерности.

Сосредоточим внимание на D-мерном евклидовом пространстве R^D. Задача состоит в том, чтобы найти элемент NN(x) в конечном наборе Y ⊂ R^D, минимизируя расстояние до вектора запроса x ∈ R^D:

NN (x) = argmin(d (x, y)). (1)

y∈Y

Поиск ближайшего соседа по своей природе дорогой из-за влияния размерности.

Для сокращения времени поиска было предложено несколько методов многомерного индексирования, таких как популярное KD-дерево или другие методы ветвей и границ. Однако для больших размерностей оказывается [12], что такие подходы не намного эффективнее, чем исчерпывающий расчет расстояний, сложность которого составляет O(nD). Также одним из самых популярных алгоритмов ANN является евклидово локально-чувствительное хеширование (LSH) [9]. Однако этот подход не учитывает требования к памяти структуры индексации. В случае LSH использование памяти может быть даже выше, чем у исходных векторов. Этот факт серьезно ограничивает число векторов, которые могут обрабатываться этим алгоритмом.

В связи с этим более подробно обсудим алгоритмы индексирования основанные на квантовании. Так как полное сравнение вектора запроса со всеми векторами коллекции невозможно, список кандидатов нужно сокращать. Поэтому была придумана инвертированная индексная структура для быстрого доступа к наиболее значимым векторам. Общая идея в том, что инвертированый индекс содержит связные списки векторов, где каждый список является отображением некоторого вектора-центроида. На первом этапе запроса определяется центроид, затем по определенному этим вектором списку выполняется исчерпывающий поиск.

Простейший инвертированный индекс основан на векторном квантовании (VQ) [1,2,7]. Цель VQ уменьшить количество элементов представления пространства поиска. Формально VQ - это функция q, отображающая d-мерный вектор x ∈ R^D на вектор q (x) ∈ C = {ci; i ∈ I}, где I — конечное индексное множество: I = 0 , ... , k - 1. Вектора ci называются центроидами. Множество Vi векторов, отображаемых в данный индекс i, называется ячейкой Вороного и определяется как

Vi = {x ∈ R^D: q (x) = ci}. (2)

Ячейки VQ образуют разбиение пространства поиска. Все векторы, лежащие в одной и той же ячейке Vi, представляются одним и тем же центроидом ci. Качество VQ обычно измеряется по среднеквадратичной ошибке между вектором запроса x и его значением q(x):

MSE (q) = Ed (q (x), x)^2

где d (x, y) = || x - y || - евклидово расстояние между x и y.

Стандартным методом обучения центроидов VQ является алгоритм кластеризации k-средних, который находит оптимальное разбиение путем итеративного сопоставления векторов набора центроидам и перестройки этих центроидов по сопоставленным векторам.

Поиск по данной структуре производится в два этапа:

1) поиск ближайшего к запросу центроида по короткому списку центроидов;

2) поиск по списку кандидатов соответствующих этому центроиду.

Чтобы обеспечить высокую скорость поиска с использование инвертированного индекса количество ячеек должно быть достаточно велико, что сильно снижает скорость обучения алгоритма на больших объемах данных.

Повысить эффективность этапа обучения можно с помощью инвертированного иерархического индекса (HKM) [11], который помимо первичного разбиения на ячейки Вороного разбивает пространство каждой ячейки повторно. В огромных датасетах уровень вложенности разбиения выбирается достаточно большим, что гарантирует небольшой список кандидатов. Помимо скорости обучения, при правильном подборе параметров можно добиться ускорения поиска [11].

Еще одна идея использования индексов для быстрого поиска предлагает разбивать пространство векторов на несколько подпространств меньшей размерности и обучить каждое подпространство малой размерности отдельно. Это позволит разбить датасет на огромное количество ячеек и с помощью декартова произведения центроидов быстро получать доступ к ним. Данная структура называется инвертированным мульти-индексом (IMI) [3].

Более формально IMI основан на квантовании произведения (PQ) [2], которое представляет из себя схему сжатия с потерями. PQ кодирует каждый вектор x ∈ R^D как объединение M кодовых слов из M D/M-мерных списков C1,..., CM, каждый из которых содержит K кодовых слов. Другими словами, PQ разбивает вектор на M отдельных подвекторов и применяет векторное квантование (VQ) к каждому подвектору, используя при этом отдельный список центроидов. Следовательно процесс обучения IMI схож с процессом обучения обычного инвертированного индекса, за тем лишь исключением, что обучать приходится несколько списков центроидов. В результате каждый вектор x кодируется набором индексов [i1,. , , , iM] и аппроксимируется как x ≈ [C1 (i1),. , , , CM (iM)]. Быстрое вычисление евклидова расстояния становится возможным благодаря эффективному асиметричному вычислению расстояния [2] с использованием обученных центроидов:

||q - x||^2 ≈ ||q - [C1 (i1),…. , CM (iM)]||^2 = СУММА(m = 1, M) ||qm - Cm (im)||^2, (2)

где qm - m-й подвектор запроса q. С точки зрения геометрии PQ эффективно разделяет исходное векторное пространство на K^M ячеек, каждая из которых является декартовым произведением M ячеек меньшей размерности. Однако стоит учитывать, что из-за огромного количества кластеров в IMI некоторые области могут содержать содержать сравнительно малое количество кандидатов или не содержать совсем. Следовательно, IMI в процессе поиска тратит много времени на посещение пустых областей. Причина этого недостатка состоит в том, что IMI при обучении не учитывает зависимость подпространств при разделении, которые зачастую зависимы на практике. В частности, существуют значительные корреляции между различными подпространствами дескрипторов, построенных с помощью сверточных нейронных сетей, которые наиболее актуальны в наши дни. Для решения проблемы адаптации алгоритмов к коррелированным данным можно предварительно производить ортогональные преобразования над данными [4] или локальные оптимизации PQ [6].

**Квантование произведения (PQ)** представляет собой схему сжатия с потерями для многомерных векторов [2]. PQ кодирует каждый вектор x ∈ RD как объединение M кодовых слов из M D M -мерных кодовых книг C1,. , , CM, каждое из которых содержит K кодовых слов. Другими словами, PQ разбивает вектор на M отдельных субвекторов и применяет векторное квантование (VQ) к каждому субвектору, используя при этом отдельную кодовую книгу. В результате каждый вектор x кодируется набором индексов кодовых слов [i1,. , , , iM] и аппроксимируется x ≈ [C1 (i1),. , , , CM (iM)]. Быстрое вычисление евклидова расстояния становится возможным благодаря эффективной процедуре АЦП [12] с использованием справочных таблиц: kq - xk 2 ≈ kq - [C1 (i1),. , , , CM (iM)] k 2 = (1) X M m = 1 kqm - Cm (im) k 2, где qm - m-й подвектор запроса q. Эта сумма может быть вычислена в M сложениях и поисках, учитывая, что расстояния от подвекторов запроса до кодовых слов предварительно вычисляются. С точки зрения геометрии PQ эффективно разделяет исходное векторное пространство на ячейки KM, каждая из которых является декартовым произведением M ячеек меньшего размера. Такое приближение на основе произведения работает лучше, если D M -мерные компоненты векторов имеют независимые распределения. На степень зависимости влияет выбор расщепления, и она может быть дополнительно улучшена путем ортогонального преобразования, примененного к векторам в качестве предварительной обработки. Поэтому в двух недавних работах было найдено оптимальное преобразование [10, 16]. После одного из них модификация PQ, соответствующая такому преобразованию предварительной обработки, упоминается ниже как Оптимизированное квантование продукта (OPQ).

**Неортогональные квантования**. В нескольких работах [8, 3, 21, 6] были предложены модификации PQ, которые не разбивают пространство данных на ортогональные подпространства. Фактически, они обобщают идею квантования продуктов, аппроксимируя каждый вектор суммой M кодовых слов вместо конкатенации. В этом случае процедура АЦП все еще эффективна, а точность аппроксимации увеличивается.

Первый подход, остаточное векторное квантование [8], квантует исходные векторы, а затем итеративно квантовает остатки аппроксимации из предыдущей итерации. [8] также вводит систему IVFRVQ, которая позволяет выполнять неисчерпывающий поиск. IVFRVQ сокращает большие части точек базы данных, основываясь на расстояниях от запроса до «грубых приближений», полученных первыми несколькими квантователями. Другой подход, Аддитивное квантование (AQ) [3], является наиболее общим, поскольку он не накладывает никаких ограничений на кодовые слова из разных кодовых книг. Обычно AQ обеспечивает наименьшие ошибки сжатия, однако он намного медленнее, чем другие методы, особенно для больших M. Композитное квантование (CQ) [21] изучает кодовые книги с фиксированным значением скалярного произведения между кодовыми словами из разных кодовых книг. Наконец, квантование дерева (TQ) [6] выполняет декомпозицию на подпространства, ограничивая отношения неортогональности между различными кодовыми книгами для формирования графа дерева. При этом ограничении возможно эффективное кодирование. В целом, неортогональные квантования достигают более высокой точности аппроксимации, чем (O) PQ, особенно для негистограммных данных. В то же время они медленнее и требуют более сложных процедур обучения.

**IVFADC.** Одной из первых систем, способных эффективно работать с наборами данных в миллиардах, была IVFADC, представленная в [13]. Эта система объединяет инвертированный индекс [19] в качестве структуры индексации и квантования продукта для сжатия базы данных. IVFADC сначала разделяет пространство на K-ячейки с помощью стандартных K-средних, а затем кодирует смещения каждой точки от центра тяжести ячейки, которой он принадлежит. Кодирование выполняется через квантование продуктов, в котором используются глобальные кодовые книги, совместно используемые всеми ячейками. **Перевернутый мультииндекс и мульти-D-АЦП.**

Inverted Multi-Index (IMI) [2] - это алгоритм индексации для многомерных пространств и очень больших наборов данных. IMI обобщает инвертированный индекс с помощью кодовых книг продукта для построения центроидов клеток (обычно рассматривается только два компонента в продукте). Таким образом, Inverted Multi-Index имеет две D 2-мерные кодовые книги произведений для разных половин вектора, каждая из которых содержит K субкодовых слов, что позволяет эффективно создавать ячейки K2, которые, как правило, на порядки превышают количество ячеек в IVFADC. система или другие системы, использующие инвертированные индексы. Большое количество ячеек обеспечивает очень плотное разделение пространства, а это означает, что для достижения высокого уровня отзыва (например, истинного ближайшего соседа) необходимо пройти небольшую часть набора данных. Для сжатия набора данных [2] следовал системе IVFADC и использовал Квантование продуктов с глобальными кодовыми книгами, общими для всех ячеек, чтобы кодировать смещения векторов от центроидов (эта система называется Multi-D-ADC).

**Исследование и построение решений задачи**

A. Вычисление расстояний с использованием квантованных кодов Рассмотрим вектор запроса x и вектор базы данных y. Мы предлагаем два метода вычисления приблизительного евклидова расстояния между этими векторами: симметричный и асимметричный. Смотрите рисунок 2 для иллюстрации. Вычисление симметричного расстояния (SDC): оба вектора x и y представлены соответствующими центроидами q (x) и q (y). Расстояние d (x, y) аппроксимируется расстоянием ˆd (x, y), dq (x), q (y), которое эффективно получается с помощью квантователя произведения ˆd (x, y) = dq (x), q (y) = sX jd qj (x), qj (y) 2, (12) где расстояние d cj, i, cj, i0 2 считывается из справочной таблицы, связанной с j-м субквантователем , Каждая справочная таблица содержит все квадраты расстояний между парами центроидов (i, i0) субквантователя или (k ∗) 2 квадратов расстояний1. Вычисление асимметричного расстояния (ADC): вектор базы данных y представлен q (y), но запрос x не закодирован. Расстояние d (x, y) аппроксимируется расстоянием ˜d (x, y), dx, q (y), которое вычисляется с использованием разложения ˜d (x, y) = dx, q (y) = sX jd uj (x), qj (uj (y)) 2, (13) где квадраты расстояний d uj (x), cj, i 2: j = 1. , , м, я = 1. , , k ∗, вычисляются до поиска.

Единственное преимущество SDC перед ADC - это ограничение использования памяти, связанного с запросами, поскольку вектор запроса определяется кодом. В большинстве случаев это не актуально, и тогда следует использовать асимметричную версию, которая позволяет получить искажения с меньшим расстоянием для аналогичной сложности. Мы сосредоточимся на АЦП в остальной части этого

Индексирование вектора y происходит следующим образом:

1) квантование y в qc (y)

2) вычисление остаточного r (y) = y - qc (y)

3) квантование r (y) в qp (r (y)), которое для квантователя продукта - это присвоение uj (y) qj (uj (y)) для j = 1. , , м.

4) добавить новую запись в инвертированный список, соответствующий qc (y). Он содержит идентификатор вектора (или изображения) и двоичный код (индексы квантователя продукта).

Поиск ближайшего соседа (ей) запроса x состоит из

1) квантования x его w ближайшим соседям в кодовой книге qc; Для наглядности на следующих двух шагах мы просто обозначим через r (x) остатки, связанные с этими w-назначениями. Два шага применяются ко всем w назначениям.

2) вычислить квадрат расстояния d uj (r (x)), cj, i 2 для каждого субквантизатора j и каждого из его центроидов cj, i;

3) вычислить квадрат расстояния между r (x) и всеми индексированными векторами инвертированного списка. Используя расстояния между субвекторами и центроидами, вычисленные на предыдущем шаге, это заключается в суммировании m искомых значений, см.

4) выберите K ближайших соседей x на основе расчетных расстояний. Это эффективно реализуется за счет поддержания структуры Maxheap с фиксированной емкостью, которая хранит наименьшие значения K, которые мы когда-либо видели. После каждого расчета расстояния идентификатор точки добавляется в структуру, только если ее расстояние меньше наибольшего расстояния в Maxheap

**Описание практической части**

**Заключение**

**ПОЛЕЗНЫЕ ВЫДЕРЖКИ**

**Статья (2)**

В этой статье мы строим короткие коды, используя квантование. Цель состоит в том, чтобы оценить расстояния с использованием расстояний вектор-центроид, то есть вектор запроса не квантуется, коды назначаются только векторам базы данных. Это уменьшает шум квантования и впоследствии улучшает качество поиска. Чтобы получить точные расстояния, ошибка квантования должна быть ограничена. Следовательно, общее число k центроидов должно быть достаточно большим, например, k = 264 для 64-битных кодов.

Квантователи решетки предлагают лучшие свойства квантования для равномерного распределения векторов, но это условие редко выполняется векторами реального мира. На практике эти квантователи выполняют значительно хуже, чем k-средних в задачах индексации [22]

В нашем методе используется аналогичное количество операций поиска в таблице, что обеспечивает несравнимую эффективность. Полное сравнение вектора запроса со всеми кодами запрещено для очень больших наборов данных. Поэтому мы вводим модифицированную инвертированную файловую структуру для быстрого доступа к наиболее значимым векторам. Грубый квантователь используется для реализации этой инвертированной файловой структуры, где векторы, соответствующие кластеру (индексу), сохраняются в связанном списке. Векторы в списке представлены короткими кодами, вычисленными нашим квантователем продуктов, который используется здесь для кодирования остаточного вектора относительно центра кластера

Б. Квантователи произведений. Рассмотрим 128-мерный вектор, например, дескриптор SIFT [23]. Квантизатор, создающий 64-битные коды, то есть «только» 0,5 бит на компонент, содержит k = 264 центроида. Следовательно, невозможно использовать алгоритм Ллойда или даже HKM, так как количество требуемых выборок и сложность изучения квантователя в несколько раз превышают k. Даже невозможно сохранить значения с плавающей запятой D × k, представляющие k центроидов. Квантование продукта является эффективным решением для решения этих проблем. Это обычная методика в исходном кодировании, которая позволяет выбирать количество компонентов, подлежащих квантованию совместно (например, группы из 24 компонентов могут быть квантованы с использованием мощной решетки пиявки). Входной вектор x разбит на m различных подвекторов uj, 1 ≤ j ≤ m размерности D ∗ = D / m, где D кратно m. Субвекторы квантуются отдельно с использованием m различных квантователей. Поэтому данный вектор x отображается следующим образом: x1, ..., xD ∗ | {z} u1 (x), ..., xD − D ∗ + 1, ..., xD | {z} um (x) → q1 u1 (x)), ..., qm (um (x), (8) где qj - квантователь с низкой сложностью, связанный с j-м субвектором. связать набор индексов Ij, кодовую книгу Cj и соответствующие значения воспроизведения cj, т. е. значение воспроизведения квантователя продукта идентифицируется элементом набора индексов продукта I = I1 × ... × Im. Поэтому кодовая книга определяется так как декартово произведение C = C1 ×.. × Cm, (9) и центроид этого множества является конкатенацией центроидов субквантизаторов m, то теперь мы предполагаем, что все субквантователи имеют одинаковое конечное число k ∗ значения воспроизводства. В этом случае общее число центроидов определяется как k = (k ∗) m. (10) Обратите внимание, что в экстремальном случае, когда m = D, компоненты вектора x все квантованы по отдельности. продукт квантования оказывается скалярным квантователем, где функция квантования, связанная с каждым компонентом, может быть разной.

, Преимущество квантователя продукта состоит в том, чтобы производить большой набор центроидов из нескольких небольших наборов центроидов: связанных с субквантователями. При изучении субквантователей с использованием алгоритма Ллойда используется ограниченное число векторов, но кодовая книга в некоторой степени все еще адаптирована к представлению данных для представления. Сложность изучения квантователя в m раз больше сложности выполнения кластеризации k-средних с k ∗ центроидами размерности D ∗

Поскольку подпространства ортогональны, квадрат искажения, связанный с квантователем произведения, равен MSE (q) = X j MSE (qj), (11) где MSE (qj) - искажение, связанное с квантователем qj. На рисунке 1 показано MSE как функция длины кода для различных (m, k ∗) наборов, где длина кода равна l = m log2 k ∗, если k ∗ - степень двойки. Кривые получены для набора 128-мерных дескрипторов SIFT, подробности см. В разделе V. Можно заметить, что для фиксированного числа битов лучше использовать небольшое количество субквантователей со многими центроидами, чем иметь много субкванизаторов с несколькими битами. В крайнем случае, когда m = 1, квантователь продукта становится обычной кодовой книгой k-средних.

Литература (По номерам статей из файла TextNotes):

[1] Jegou, H., Tavenard, R., Douze, M., Amsaleg, L.: Searching in one billion vectors: Re-rank with source coding. In: ICASSP. (2011)

[2]H. Jegou, M. Douze, and C. Schmid, “Product quantization for ´ nearest neighbor search,” TPAMI, vol. 33, no. 1, 2011.

[3]A. Babenko and V. Lempitsky, “The inverted multi-index,” 2014 IEEE

[4]Ge, T., He, K., Ke, Q., Sun, J.: Optimized product quantization. Technical report (2013)

[5]Kalantidis, Y., Avrithis, Y.: Locally optimized product quantization for approximate nearest neighbor search. In: in Proceedings of International Conference on Computer Vision and Pattern Recognition (CVPR 2014), IEEE (2014)

[6]Babenko, A., Lempitsky, V.S.: Efficient indexing of billion-scale datasets of deep descriptors. In: CVPR. (2016)

[7]Revisiting the Inverted Indices for Billion-Scale Approximate Nearest Neighbors Dmitry Baranchuk1,2 , Artem Babenko1,3 , Yury Malkov4

[8]Wang, D., Otto, C., Jain, A.K.: Face search at scale. TPAMI (2017)

[9] W. Dong, Z. Wang, W. Josephson, M. Charikar, and K. Li, “Modeling LSH for performance tuning,” CIKM, 2008. 2, 4

[10]D. Yi, S. Liao, and S. Z. Li, “Learning face representation from scratch,” arXiv:1411.7923v1, 2014

Двухуровневая индексация

[11] D. Nister and H. Stew ´ enius, “Scalable recognition with a vocab- ´ ulary tree,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2161–2168, 2006

[12]A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces Weber, Roger and Schek, Hans-Jrg and Blott, Stephen