А. С. Вешкин[1], *А. В. Хвостиков*[2]

МНОГОМАСШТАБНЫЙ МЕТОД ПОИСКА ПО СОДЕРЖИМОМУ ДЛЯ ПОЛНОСЛАЙДОВЫХ ГИСТОЛОГИЧЕСКИХ ИЗОБРАЖЕНИЙ*

Введение

Данная работа посвящена разработке метода поиска по содержимому для полнослайдовых гистологических изображений и оценка эффективности его применения в качестве вспомогательного средства в работе врачей гистологов.

Современные гистологические сканеры позволяют врачам-гистологам получать полнослайдовые изображения (*англ. Whole Slide Images, WSI*) образцов ткани в очень высоком разрешении. Типичное разрешение подобных изображений крайне велико — порядка 100 тыс. $\times$ 100 тыс. пикселей, но может достигать и больших значений.

Процесс ручного анализа таких изображений кажется трудозатратным не только из-за огромного количества информации, но и потому что с такими изображениями можно работать только при помощи специального программного обеспечения.

В задачах, связанных с анализом большого числа изображений часто применяют методы поиска по содержимому (*англ. Content Based Image Retrieval, CBIR*). Эти медоты позволяют автоматически анализировать имеющийся набор изображений и осуществлять среди него поиск похожих изображений. Данные методы получили широкое распространение. Например, они используются при поиске картинок в интернете.

Если абстрагироваться от таких особенностей, как, например, выбор способа выделения признаков, характеризующих область интереса, то общая схема работы CBIR методов выглядит следующим образом (Рис. 1):

[1]Техник-программист факультета ВМК МГУ имени М. В. Ломоносова, e-mail: artem.veshkin@gmail.com.

[2]К.ф.-м.н. м.н.с. факультета ВМК МГУ имени М. В. Ломоносова, e-mail: khvostikov@cs.msu.ru.

*Работа выполнена при поддержке гранта РНФ 22-41-02002.

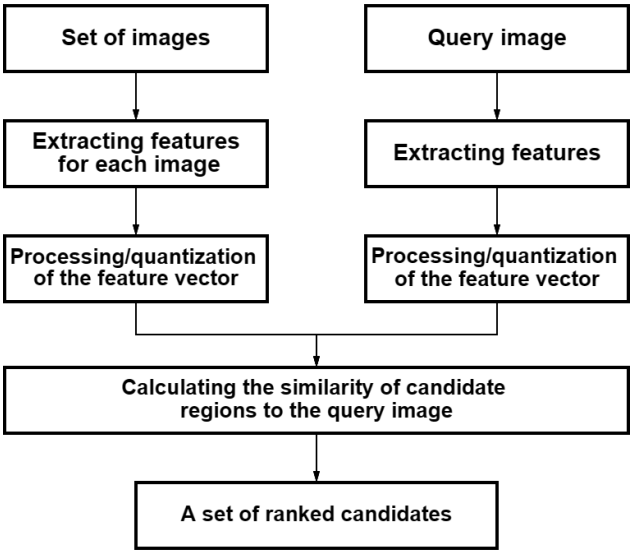

Рис. 1. Общая схема работы методов CBIR

1. Для каждого изображения из набора изображений выделяется набор признаков. В случае данной работы, особенностью системы является работа с полнослайдовыми изображениями, позволяющими рассматривать изображение на большом количестве масштабов. Поэтому признаки извлекаются со всех используемых для поиска масштабов;
2. После, признаки извлекаются и для фрагмента-запроса;
3. Вектор признаков характеризует изображение-запрос, и его можно сравнивать определенным образом с другими векторами признаков из набора изображений. По результатам сравнений определяются наиболее похожие фрагменты изображений из набора изображений и возвращаются пользователю в качестве результата.

Метод поиска на основе бинаризации

В качестве первого подхода при разработке метода было решено использовать бинаризацию снимков [1]. На Рис. 2 приведены этапы работы алгоритма, подробное объяснение приведено ниже.

Сначала происходит предобработка набора изображений. На каждый снимок накладывается сетка, которая разбивает его на равные квадраты T_{ij}. В данной работе размер квадратов выбран 224×224 пикселя.

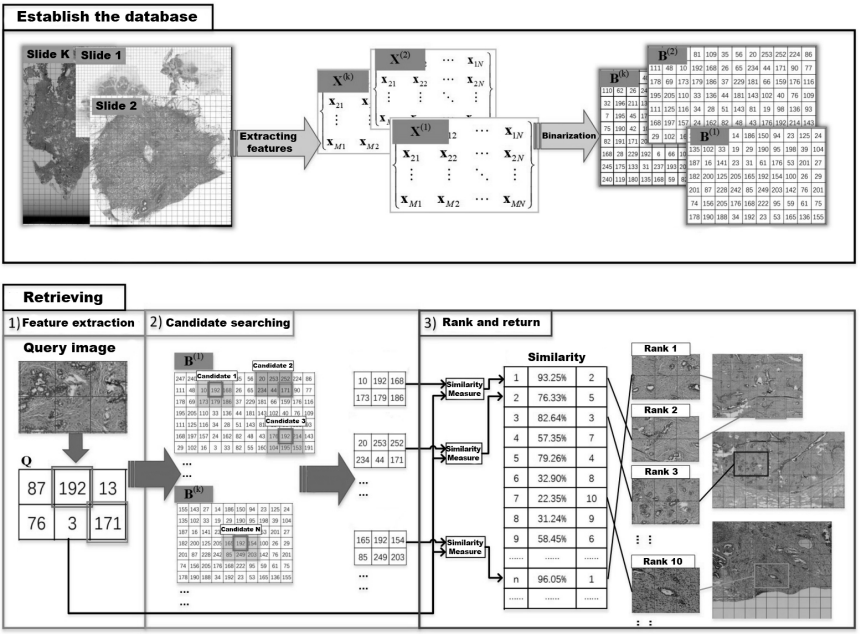

Рис. 2. Схема работы метода CBIR при использовании бинаризации

Для каждого такого фрагмента T_{ij} вычисляется вектор признаков $x_{ij} = f(T_{ij})$. Способы получения признаков описаны в разделе *"Извлечение признаков"*.

К каждому вектору признаков x_{ij} применяется специальная бинарная хэш-функция, устройство которой описано в разделе *"Выбор хэш-функции"*. Эта функция возвращает 32 битное двоичное число $b_{ij} = h(x_{ij})$.

Таким образом каждое i-тое изображение из набора изображений можно охарактеризовать матрицей B_i вида:

$$B_i = \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1N} \\ b_{21} & b_{22} & \dots & b_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ b_{M1} & b_{M2} & \dots & b_{MN} \end{pmatrix},$$

а весь набор данных D как набор матриц $D = \{B_1, B_2, \dots, B_L\}$, где L - число изображений в наборе данных.

Аналогичные действия производятся с изображением, которое поступает в качестве запроса. На него накладывается сетка тех же размеров, для фрагментов извлекаются вектора признаков. После к ним применяются те же хэш-функции, что и к набору изображений. Таким

образом входное изображение характеризуется набором 32-х битных чисел:

$$Queue = \begin{pmatrix} q_{11} & q_{12} & \cdots & q_{1n} \\ q_{21} & q_{22} & \cdots & q_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ q_{m1} & q_{m2} & \cdots & q_{mn} \end{pmatrix}$$

Затем происходит сравнение матрицы *Queue* со всеми подматрицами из базы данных D, в которых есть хоть один одинаковый элемент $b_{i'j'} = q_{ij}$. Между подматрицами и матрицей запроса *Queue* вычисляетя специальное расстояние, вводимое на основе расстояния по Хэммингу. О нем также будет рассказанно в разделе *"Вычисление расстояния между хэшами"*.

В итоге пользователю возвращаются фрагменты из набора изображений, соответствующие подматрицам, которые оказались ближе всего к матрице *Queue*.

Извлечение признаков

В рамках разработки метода с использованием бинаризации, для извлечения признаков из изображения было испробовано 2 подхода. Первый использовал в качетсве признаков статистические характеристики изображения. Во втором случае признаки получались при помощи сверточной нейронной сети. Качество работы метода с использованием обоих экстракторов измерено на отдельном размеченном наборе данных.

Статистические признаки

В качестве первого экстрактора признаков использовались статистические характеристики изображения.

В данной работе использовались следующие признаки:

- $MEAN = \frac{1}{N}\sum_i \sum_j I[i,j]$ - среднее значение;
- $SD = \frac{1}{N}\sqrt{\sum_i \sum_j (I[i,j] - MEAN)^2}$ - среднеквадратичное отклонение;
- $VAR = \frac{SD}{MEAN}$ - вариация;
- $SKEW = \frac{1}{N*SD^3}\sum_i \sum_j (I[i,j] - MEAN)^3$ - асимметрия;
- $KURT = \frac{1}{N*SD^4}\sum_i \sum_j (I[i,j] - MEAN)^4$ - эксцесс;
- $ENT = -\sum_i \sum_j I[i,j] \log_2 I[i,j]$ - энтропия.

(I - изображение, i и j в диапазоне длины и ширины изображения, N - число пикселей изображения).

Перед вычислением данных статистических характеристик изображения переводились в градации серого и нормализовались [2].

Также использовался ряд характеристик для матриц смежности (*англ. Grey Level Co-occurrence Matrix, GLCM*). Матрица смежности строится по фиксированному вектору $\overline{d} = (dr, dc)$, задающему смещение между пикселом яркостью i и пикселом яркостью j:

$$C_d[i,j] = |\{(r,c)|I[r,c] = i, I[r+dr, c+dc] = j\}|.$$

Для компактного описания GLCM используются числовые признаки:

- $ASM = \sum_i \sum_j C_d^2[i,j]$ - энергия;
- $ENT = -\sum_i \sum_j C_d^2[i,j] \log_2 C_d^2[i,j]$ - энтропия;
- $CON = \sum_i \sum_j (i-j)^2 C_d^2[i,j]$ - контрастность;
- $HOM = \sum_i \sum_j \frac{C_d^2[i,j]}{1+|i-j|}$ - однородность;
- $DIS = \sum_i \sum_j C_d^2[i,j]|i-j|$ - различие;
- $COR = \sum_i \sum_j \frac{(i-\mu_x)(j-\mu_j)}{\sqrt{\sigma_x^2 * \sigma_y^2}} * C_d[i,j]$ - корреляция.

В работе использовалась GLCM со смещением в 5 пикселей в направлении 0 градусов. Итого получалось 6 признаков для исходного изображения и 6 признаков для матрицы смежности. Таким образом вычисляется вектор признаков длины 12.

Нейросетевой экстрактор признаков

В качестве второго экстрактора признаков использовалась предобученная сверточная нейронная сеть. Сеть имела структуру, похожую на AlexNet [3] и была изначально обучена для классификации фрагментов ткани гистологических изображений на 9 классов. Обучение велось на наборе данных NCT-CRC-HE-100K [4]. Точность классификации сети составила более 95% на тестовой выборке.

Далее у сети были отрезаны полносвязные слои и оставлены только сверточные слои, отвечающие за получение внутреннего представления. К ним был применен слой субдискретизации с функцией усреднения (AvgPool), усредняющий результирующую карту признаков до размера 1×1.

В итоге, модифицированная сеть возвращает набор из 64 значений. Интересующий фрагмент изображения, для которого нужно вычислить признаки, подавался на вход сети и результат рассматривался как вектор признаков, характеризующий его, и использовался далее в алгоритме.

Выбор хэш-функции

Вектора признаков, полученные на предыдущем этапе, должны быть бинаризованы, то есть преобразованы в двоичные числа. Для этого используется специальная бинарная хэш-функция.

В данной работе качестве хэш-функции была использована функция Locality-Sensitive Hash (LSH) [5, 6]. Особенностью этой функции является то, что она учитывает близость векторов признаков в признаковом пространстве. Таким эффектом, как и LSH, обладают, например, thresholded PCA (tPCA)[7], Iterative Quantization (ITQ) [8], binary factor analysis (BFA) [9] и Binary Autoencoders (BA) [9]. LSH была выбрана как наиболее простая в интерпретации и реализации функция.

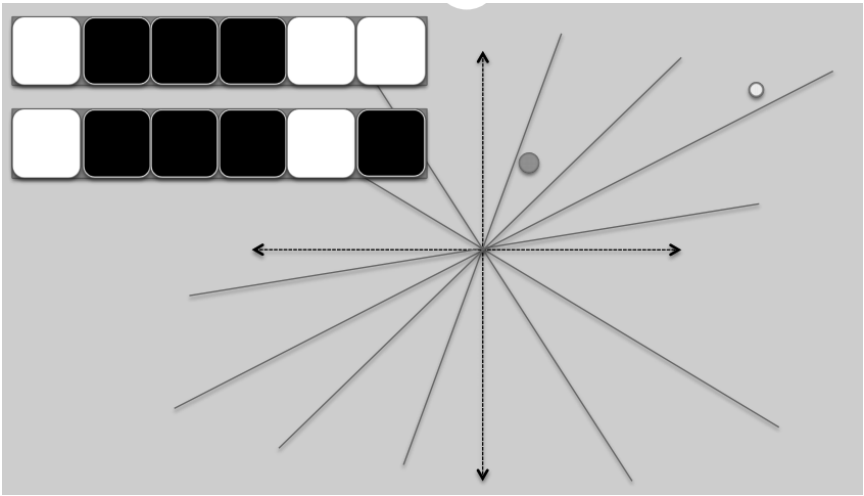

Рис. 3. Пример работы функции LSH для вектора признаков длины 2 и семи разделяющих прямых. Строятся 7 разделяющих прямых, выходящих из начала координат. Для каждой прямой выполняется операция: если вектор признаков находится над i-ой прямой, то на i-ой позиции результата ставится 0, иначе 1

На Рис. 3 приведен пример применения функции LSH. В примере на рисунке результатом работы будет 7-ми значное двоичное число, а работа происходит для вектора из двух признаков.

В общем случае мы имеем дело с n-мерным пространством признаков. В нем строятся k случайных разделяющих гиперплоскостей. Выходом бинарной хэш-функции функции LSH будет k-битное двоичное число, в котором на i-том месте будет стоять 0 в случае нахождения вектора признаков над i-ой гиперплоскостью, и 1 иначе. Это позволяет хэшам близких в признаковом пространстве векторов быть близкими по расстоянию Хэмминга.

Вычисление расстояния между хэшами

Для вычисления расстояния между двоичными хэшами можно использовать стандартное расстояние Хэмминга (число позиций, в которых соответствующие значения двоичных чисел различны). Но целью является вычислить расстояние между двумя матрицами, состоящими из двоичных хэшей. Если поэлементно развернуть эти матрицы в вектора, получим два вектора из двоичных чисел, соответствующих запрашиваемому изображению и фрагменту-кандидату.

Существует несколько способов вычисления расстояния между ними. В данной работе лучше всего показал себя следующий.

Пусть есть вектор запроса $Q = \{q_1, q_2, \ldots, q_l\}$ и вектор кандидата $P = \{p_1, p_2, \ldots, p_l\}$.

Для каждого $q_i \in Q$ можно найти ближайший по Хэммингу элемент $p_{i_{min}} = \min\limits_{p_j \in P} d_h(q_i, p_j)$.

Далее вычисляется величина d_{near}, определяемая, как среднее арифметическое по всем $p_{i_{min}}$:

$$d_{near}(Q,P) = \frac{1}{l} \sum_{q_i \in Q} p_{i_{min}}.$$

Итоговую меру $D(Queue, B')$, где B' это подматрица - кандидат, а *Queue* - матрица, соответствующая запросу, определяем следующим образом:

$$D(Queue, B') = d_{near}(Queue, B') + d_{near}(B', Queue).$$

Все это позволяет ввести расстояние между матрицей, характеризующей запрос и матрицами кандидатов из набора данных с изображениями.

Метод поиска на основе вариационного автокодировщика

Для повышения качества извлекаемых из фрагментов признаков была предпринята попытка использовать вариационный автокодировщик [10]. На Рис. 4 приведена его схема.

Вариацонный автокодировщик - это нейросетевая модель, состоящая из двух основных частей: кодировщика и декодировщика. Кодировщик в ходе обучения учится компактно описывать входное изображение, а декодировщик - по этому описанию восстанавливать исходное изображение.

На выходе кодировщика обучаются два вектора: μ_x и σ_x. По ним с использованием репараметризации генерируется вектор $z = \mu_x + \exp(\frac{\sigma_x}{2}) \cdot \varepsilon$, где $\varepsilon \sim \mathcal{N}(0,1)$. Таким образом получается, что $z \sim \mathcal{N}(\mu_x, \sigma_x)$. Этот вектор z передается на вход декодировщику и может использоваться в методе поиска по содержимому в качестве вектора признаков.

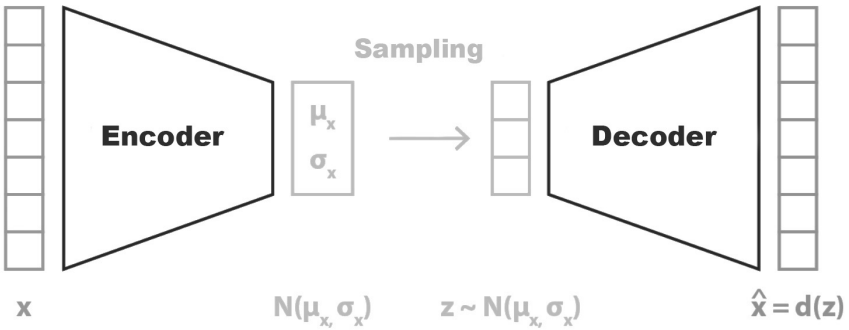

Рис. 4. Схема работы вариационного автокодировщика. x - входное изображение, $\hat{x}$ - изображение на выходе декодировщика, μ_x, σ_x - вектора на выходе кодировщика

В данной работе на вход вариационному автокодировщику подается трехканальное изображение размера 224×224 пикселя. Такое изображение эквивалентно вектору признаков из почти 150 тысяч значений. При этом модель кодирует такой объем информации в вектор z, длина которого составляет от 32 до 512 в зависимости от архитектуры. Таким образом получается сжать входное изображение почти в 500 раз. И вектором такой длины удается довольно точно описать содержимое входного изображения.

Функция потерь, используемая при обучении вариационного автокодировщика, состоит из двух частей:

$$Loss = Loss_1 + Loss_2,$$

$$Loss_1 = (x - \hat{x})^2, Loss_2 = \log\frac{1}{\sigma_x} + \frac{\sigma_x^2 + \mu_x^2}{2} - \frac{1}{2},$$

Где x - изображение на входе автокодировщика, $\hat{x}$ - изображение на выходе автокодировщика, μ_x и σ_x - выходы кодировщика.

$Loss_1$ - квадрат отклонения исходного изображения от воссозданного. $Loss_2$ - дивергенция Кульбака-Лейблера между $\mathcal{N}(\mu_x, \sigma_x)$ и $\mathcal{N}(0,1)$. Таким образом изображение на входе автокодировщика приближается к изображению на выходе, а распределение векторов в скрытом представлении - к нормальному с $\mu = 0, \sigma = 1$.

На Рис. 5 представлен пример работы вариационного автокодировщика применительно к фрагментам гистологических изображений.

Из-за нормального распределения компонент векторов признаков, генерируемых вариационным автокодировщиком, эти компоненты расположены близко к нулю. Поэтому использование бинаризации при

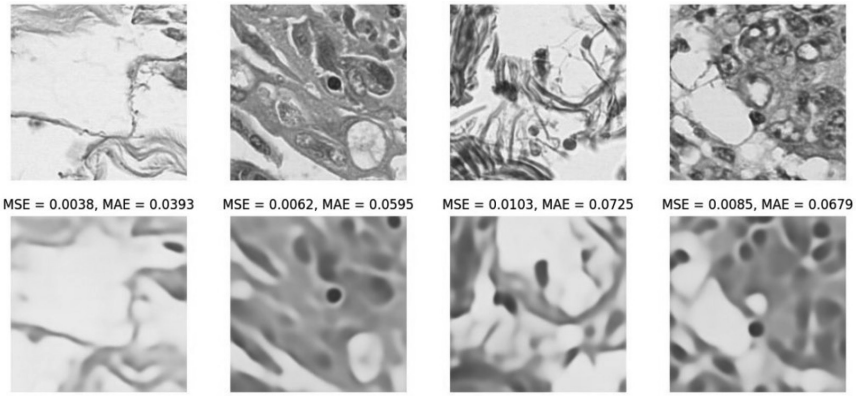

Рис. 5. Работа вариационного автокодировщика на гистологических изображениях. В первой строке - изображения, подаваемые на вход кодировщику, во второй - восстановленные декодировщиком по вектору z. Входные изображения эквивалентны вектору признаков из почти 150 тысяч значений. Кодировщик осуществляет их сжатие почти в 500 раз

работе с такими признаками не дает удовлетворительных результатов из-за ошибок квантования.

Поэтому при использовании вариационного автокодировщика от бинаризации пришлось отказаться и в качестве меры близости векторов использовать квадрат отклонения $d(z_1, z_2) = (z_1 - z_2)^2$.

Автоматическое определение масштаба

Особенностью метода, разрабатываемого в данной работе, является работа с полнослайдовыми изображениями, позволяющими рассматривать изображение на большом количестве масштабов. Таким образом приходится осуществлять поиск по всем возможным масштабам, так как масштаб, для которого взят запрос, зачастую неизвестен.

Использование бинаризации позволяет при помощи "дешевых"вычислений сократить число кандидатов для поиска. Но при использовании же вариационного автокодировщика воспользоваться бинаризацией для ускорения поиска не получится.

Однако, если метод сможет автоматически определять масштаб, на котором стоит осуществлять поиск, а не будет перебирать все возможные варианты, то это существенно сократит время ответа на запрос.

Для решения этой задачи было решено обучить нейронную сеть, на вход которой подаются два изображения, а на выходе возвращается вероятность того, что это изображения, взятые с одного масштаба. Далее выбирая несколько фрагментов с каждого мастшаба полнослайдового

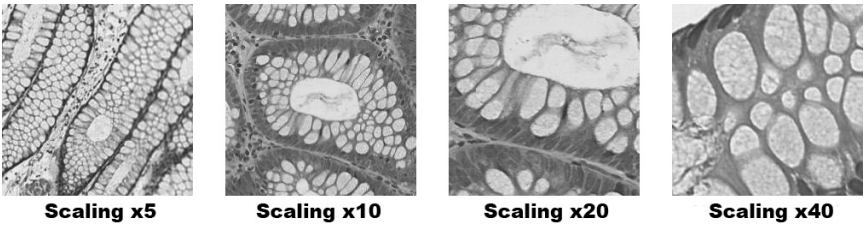

Рис. 6. Пример фрагментов одного и того же полнослайдового изображения, взятого на разных масштабах

изображения и сравниваяь их с изображением-запросом, в зависимости от ответа нейронной сети можно выбрать масштабы для поиска.

На Рис. 7 приведена архитектура описанной выше нейронной сети. На вход сети подаются два трехканальных RGB изображения, сложенных в стопку. Таким образом вход модели имеет 6 каналов. Далее идут комбинации из сверточного слоя, слоя пакетной нормализации [11] и функции активации LeakyReLU. После следуют 3 линейных слоя, последний из которых имеет один выход. Значение этого выхода переводится функцией $\sigma(x) = \frac{1}{1+e^{-x}}$ в диапазон от 0 до 1 и интерпретируется как вероятность того, что входные изображения взяты с одного масштаба.

Для обучения модели брались случайные фрагменты полнослайдовых изображений из набора данных PATH-DT-MSU, взятых с масштабов 5, 10, 15, 20, 25, 30, 35 и 40. 40 - максимально возможное увеличение, 20 - увеличение в 2 раза меньше и так далее. Далее из этих фрагментов составлялись пары. Так как задача определения того, взяты ли фрагменты с одного масштаба или с разных, эквивалентна задаче бинарной классификации, требовалось соблюдать баланс пар фрагментов с одного масштаба и с разных.

Обучение модели велось на графическом ускорителе GeForce RTX 3060 Ti. После $N = 1.000.000$ шагов (трех дней обучения) точность сети на валидационной выборке составила более 96%.

На Рис. 8 демонстрируется работа метода автоматического определения масштаба для изображений разных масштабов. На визуализациях представлены отклики нейронной сети на каждый из масштабов. Зеленым выделен столбец с масштабом, с которого было взято изображение. Во всех случаях отклик на истинный масштаб или на соседние с ним является максимальным, что демонстрирует работоспособность подхода.

Метод автоматического определения масштабов позволяет вместо поиска по всем доступным масштабам ограничится поиском лишь по нескольким из них, что ускоряет работу метода в несколько раз. Также,

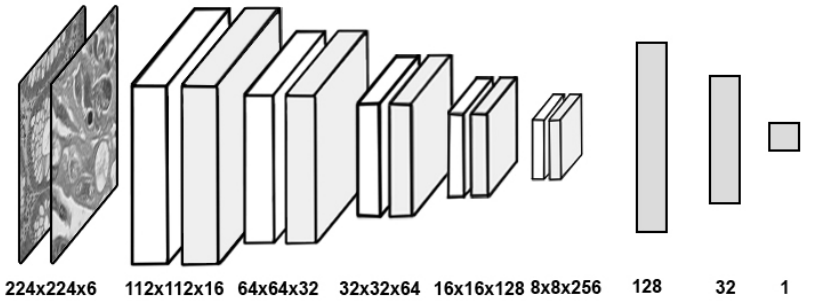

Рис. 7. Архитектура нейронной сети для сравнения масштабов двух изображений. К исходным двум изображениям поочередно применяются свёрточные слои и слои пакетной нормализации, затем полносвязные слои. Для каждого тензора снизу подписана его размерность

при необходимости поиска на других масштабах, сеть можно относительно легко дообучать.

Оценка качества работы метода

Для оценки качества работы разработанного метода поиска по содержимому использовались поднаборы WSS1, WSS2 набора данных PATH-DT-MSU (`https://imaging.cs.msu.ru/en/research/histology/path-dt-msu`), в которых фрагменты полнослайдовых изображений соответствуют различным типам тканей (всего 4 класса: AT, LP, MM, TUM).

При выборе способа оценки качетсва работы метода было решено исходить из предположения, что при корректной работе метода, он в ответ на запрос определенного класса будет преимущественно возвращать области, принадлежащие к тому-же классу.

При это больший вклад в меру качества должны были вносить ошибки на первых позициях выдачи, чем на последних.

Для учета этих моментов и численного измерения качества работы метода использовалась фукнция $MAP@k$ - Mean Average Precision для top k [12, 13]:

$$MAP@k = \frac{\sum_{n=1}^{k} \frac{1}{n} * [class_{target} = class_n]}{\sum_{n=1}^{k} \frac{1}{n}},$$

$class_{target}$ - класс изображения, поданного в метод,

$class_n$ - класс изображения, находящегося на позиции с номером n в выдаче метода,

k - число отранжированных кандидатов.

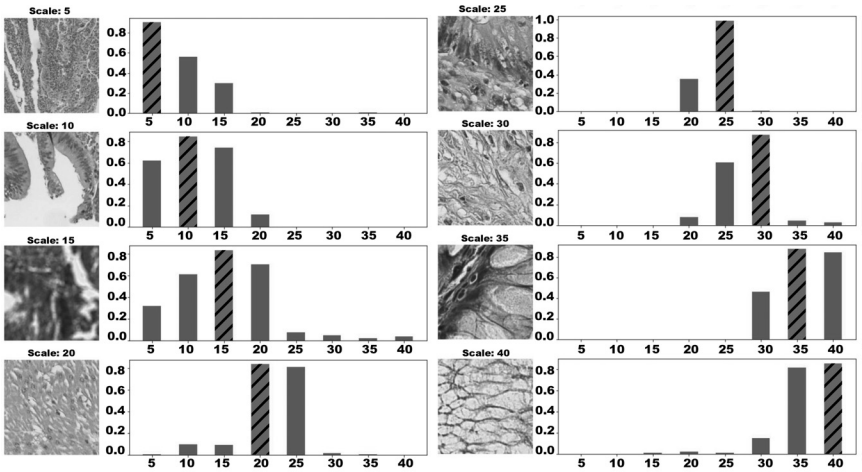

Рис. 8. Работа метода автоматического определения масштаба изображения. На визуализации представлены фрагменты изображений, взятых на разных масштабах и соответствующие этим фрагментам отклики модели определения масштаба. Столбец со штриховкой соответствует реальному масштабу изображения. По оси обсцисс отложен масштаб в диапазоне от 5 до 40 с шагом 5

Фукнция $MAP@k$ учитывает соответствие класса запроса классу результата на n-том месте с весом $\frac{1}{n}$. Таким образом, ошибки на первых результатах выдачи сильнее сказываются на итоговом значении функции, чем ошибки на последних результатах. Также значение функции приводится к диапазону [0, 1] путем деления на $\sum_{n=1}^{k} \frac{1}{n}$.

На Рис. 9 показаны значения функции $MAP@10$ для трех типов экстракторов признаков. Значения представлены как для каждого класса изображений по отдельности, так и усредненные по всем классам. При построении визуализации использовалось 10 запросов для каждого класса ткани. Поиск осуществлялся по полному набору размеченных фрагментов тканей.

Из визуализации видно, что метод с использованием признаков, полученных при помощи вариационного автокодировщика, в среднем оказывается лучше других вариантов метода и достигает среднего значения $MAP@10$ равного 0.293.

$MAP@10$ - это метрика ранжирования и с большим весом она учитывает успех или ошибку на первых результатах выдачи. Поэтому значение $MAP@10 = 0.293$ не означает, что метод в среднем находит только 3 из 10 фрагментов того же класса, что и изображение-запрос. Его можно трактовать как то, что либо метод ошибается только в небольшом

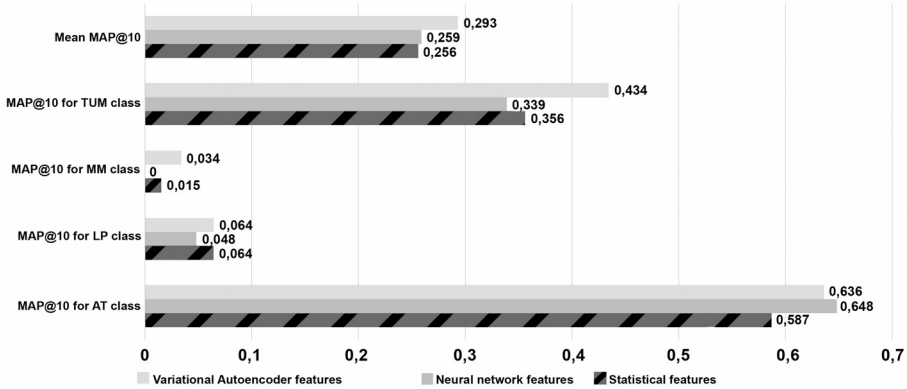

Рис. 9. Значение функции *MAP@*10 при работе с 4 классами изображений для трех типов экстракторов признаков на разных классах гистологических изображений

числе первых результатов выдачи, либо преимущественно в последних результатах. Поэтому метод с такими показателями может быть использован на практике в работе врачей в качестве вспомогательного средства.

Заключение

Метод, описаный в этой работе, реализован на языке Python 3.8 в виде отдельного модуля. Весь исходный код доступен по ссылке: https://github.com/ArtemVeshkin/Content-Based-Image-Retrieval.

Этап загрузки и индексации полнослайдовых изображений в метод является наиболее длительным по времени и занимает от 3 до 10 минут в зависимости от используемого экстрактора признаков. Этот этап выполняется только один раз для каждого изображения.

При тестировании скорости поиска по полнослайдовым изображениям из набора данных PATH-DT-MSU, скорость поиска составиляла от 2 до 10 секунд в зависимости от использования опции автоматического определения масштаба запроса и от масштаба, на котором осуществлялся поиск.

Из приведенных результатов видно, что метод действительно ищет похожие фрагменты гистологических изображений.

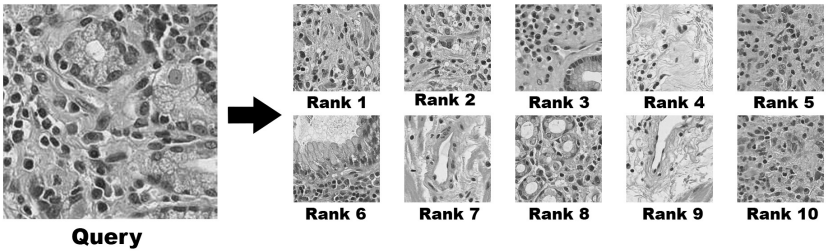

Рис. 10. Пример работы метода поиска по содержимому. Слева - изображение-запрос, справа - найденные методом области

Благодарности

Работа выполнена при поддержке гранта РНФ 22-41-02002.

Литература

1. Size-scalable content-based histopathological image retrieval from database that consists of WSIs / Zheng Y., Jiang Z., Zhang H., Xie F., Ma Y., Shi H., and Zhao Y. // IEEE journal of biomedical and health informatics. — 2017. — Vol. 22, no. 4. — P. 1278–1287.

2. Kanan C., Cottrell G. W. Color-to-grayscale: does the method matter in image recognition? // PloS one. — 2012. — Vol. 7, no. 1. — P. e29740.

3. Yuan Z.-W., Zhang J. Feature extraction and image retrieval based on AlexNet // Eighth International Conference on Digital Image Processing (ICDIP 2016) / International Society for Optics and Photonics. — 2016. — Vol. 10033. — P. 100330E

4. Kather, Jakob Nikolas, Halama, Niels, Marx, Alexander. (2018). 100,000 histological images of human colorectal cancer and healthy tissue (v0.1) [Data set]. Zenodo. https://doi.org/10.5281/zenodo.1214456

5. Slaney M., Casey M. Locality-sensitive hashing for finding nearest neighbors [lecture notes] // IEEE Signal processing magazine. — 2008. — Vol. 25, no. 2. — P. 128–131.

6. Kulis B., Grauman K. Kernelized locality-sensitive hashing // IEEE Transactions on Pattern Analysis and Machine Intelligence. — 2011. — Vol. 34, no. 6. — P. 1092–1104

7. Wang J., Kumar S., Chang S.-F. Semi-supervised hashing for scalable image retrieval // 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition / IEEE. — 2010. — P. 3424–3431.

8. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval / Gong Y., Lazebnik S., Gordo A., and Perronnin

F. // IEEE transactions on pattern analysis and machine intelligence. — 2012. — Vol. 35, no. 12. — P. 2916–2929.

9. Carreira-Perpinán M. A., Raziperchikolaei R. Hashing with binary autoencoders // Pro- ceedings of the IEEE conference on computer vision and pattern recognition. — 2015. — P. 557–566.
10. Kingma D. P., Welling M. Auto-encoding variational bayes // arXiv preprint arXiv:1312.6114. — 2013.
11. How does batch normalization help optimization? / Santurkar S., Tsipras D., Ilyas A., and Madry A. // Advances in neural information processing systems. — 2018. — Vol. 31.
12. Learning with average precision: Training image retrieval with a listwise loss / Revaud J., Almazán J., Rezende R. S., and Souza C. R. d. // Proceedings of the IEEE/CVF International Conference on Computer Vision. — 2019. — P. 5107–5116.
13. Lim D., Lanckriet G., McFee B. Robust structural metric learning //International conference on machine learning. – PMLR, 2013. – P. 615-623.