

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное образовательное
учреждение высшего образования
ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ

Институт математики, механики и компьютерных наук
имени И. И. Воровича

Направление подготовки Прикладная математика и информатика

Кафедра Информатики и вычислительного эксперимента

МЕТОДЫ СЕГМЕНТАЦИИ ИЗОБРАЖЕНИЙ

Курсовая работа

Студента 3 курса
А. А. Мезга

Научный руководитель:
кандидат физико-математических наук, доцент В. А. Нестеренко

оценка (рейтинг) подпись руководителя

Ростов-на-Дону

2019

Содержание

Введение	3
1. Задача сегментации изображений	5
2. Методы сегментации изображений	6
2.1. Классические методы сегментации изображений	7
2.1.1. Морфологические методы	7
2.1.2. Пороговые методы	9
2.1.3. Метод водоразделов	11
2.1.4. Текстурные методы	13
2.2. Методы глубокого обучения	14
2.2.1. Fully Convolutional Network	14
2.2.2. U-Net	15
2.2.3. Модель тирамису	17
3. Сегментация клеточных ядер на биомедицинских снимках	18
3.1. Входные данные	18
3.2. Программная реализация и технические детали	19
3.2.1. Инициализация данных	20
3.2.2. Архитектура сети и обучение модели	23
3.2.3. Визуализация результатов	26
Заключение	29
Список литературы	31

Введение

Изображение является одним из главных способов передачи информации, поэтому идентификация, извлечение и преобразование этой информации- важные задачи в области обработки цифровых изображений. Такая обработка может производиться как для получения выходного изображения, так и для получения контекстной информации (например, распознавание текста или поиск объектов на изображении). Разделение изображения на значимые структуры также часто является этапом анализа, визуализации и многих других задач обработки изображений.

Человек способен легко распознавать объекты, выделять их края и понимать контекст изображения, однако значительно сложнее охарактеризовать и выделить такие свойства с помощью точных параметров, которые позволяли бы компьютеру выполнять подобные задачи. Одной из них и является сегментация изображений, представляющая собой разбиение изображения на области. Она входит в целый ряд практических задач, таких как распознавание образов, нахождение дефектов при производстве, обработка спутниковых и медицинских снимков, что делает её актуальной.

В настоящий момент не существует метода, который мог бы показывать одинаково высокие результаты для различных типов задач. Наиболее часто на выбор конкретного метода влияет область, для которой необходимо его применение.

Методы сегментации в области медицинских снимков являются одними из наиболее развивающихся, что неудивительно, ведь качественное выполнение такой задачи позволит, например, обнаруживать опухоли и другие патологии на ранних этапах, проводить диагностику и планировать лечение.

В рамках курсовой работы была рассмотрена задача автоматического обнаружения клеточных ядер, что обусловлено важностью данной проблемы. Её решение позволит быстрее производить исследования, сокращая тем самым десятилетний период создания и выхода на рынок новых ле-

карственных препаратов от таких тяжелых заболеваний, как рак, болезни сердца, болезнь Альцгеймера и диабет.

Таким образом, можно выделить следующие задачи, рассматриваемые в данной курсовой работе:

1. Формально описать задачу сегментации изображений.
2. Рассмотреть существующие методы сегментации изображений.
3. Реализовать алгоритм сегментации изображений на примере задачи автоматического обнаружения клеточных ядер на биомедицинских изображениях.

1. Задача сегментации изображений

Задача сегментации ставится на начальных этапах обработки изображений и заключается в выделении непересекающихся областей, покрывающих данное изображение, для которых выполняется некий критерий однородности. В качестве данного критерия может выступать цвет, яркость, текстура и пр (см. рисунок 1).

Процесс сегментации позволяет облегчить сбор характеристик изображения, разбив его на отдельные области. Можно выделить несколько подходов к решению такой задачи:

- Анализ низкого уровня - процесс разбиения изображения на области "похожих" друг на друга пикселей.
- Анализ высокого уровня - процесс отделения находящихся на изображении объектов от фона и друг от друга.

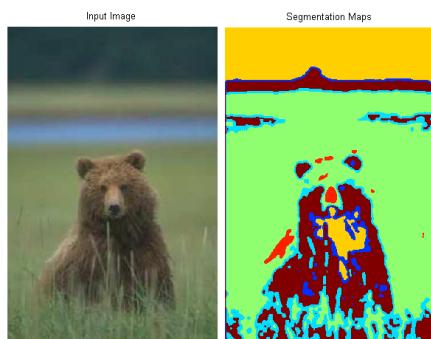


Рисунок 1 — Пример сегментации по цвету

Помимо классической сегментации существуют несколько подвидов данной задачи, так называемые Instance и Semantic Segmentation.

В случае задачи Semantic Segmentation (семантическая сегментация), изображение разделяется на отдельные группы пикселей, которые объединяются в области, принадлежащие одному объекту. При этом определяются типы объектов в этих областях.

Главным отличием Instance Segmentation от Semantic Segmentation является необходимость обработки разных экземпляров одного класса в

виде различных объектов, что усложняет задачу. Более наглядно разницу этих подходов можно увидеть на примере рисунка 2.

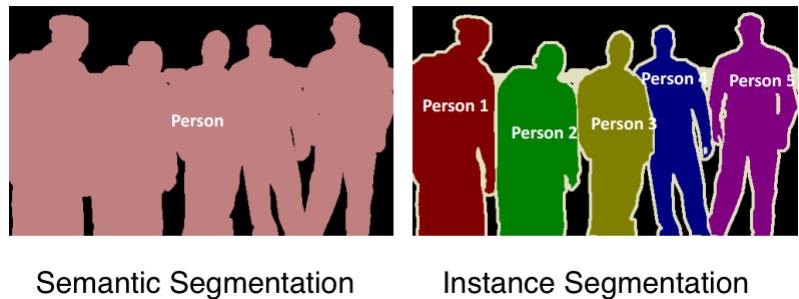


Рисунок 2 — Сравнение Instance и Semantic Segmentation

Данные задачи тесно связаны с задачами классификации и поиска объектов на изображении, однако являются более комплексными и сложными, так как помимо классификации найденных объектов, существует задача правильного выделения их частей на изображении.

2. Методы сегментации изображений

В первую очередь, методы сегментации изображений можно разделить на две группы:

- Автоматические методы (позволяют получить результат без взаимодействия с пользователем).
- Интерактивные методы (требуют непосредственного участия пользователя в процессе обработки входного изображения).

Несмотря на активное развитие автоматических методов, они все еще не могут сравняться по точности с интерактивными. Поэтому в случаях, когда нужна максимальная точность обработки, необходимо участие человека.

Нужно отметить, что для сегментации изображений создано множество алгоритмов, однако общего решения не существует. В связи с этим для достижения лучших результатов они часто совмещаются. Кроме то-

го, выбор определенного метода следует производить с учетом конкретной задачи.

В данном разделе будут рассмотрены как классические подходы, применяющиеся для решения задачи сегментации, так и методы, основанные на различных методах глубокого обучения искусственных нейронных сетей.

2.1. Классические методы сегментации изображений

В качестве классических методов сегментации выделим следующие классы (по возрастанию сложности) и рассмотрим их подробнее:

1. морфологические методы;
2. пороговые методы;
3. методы наращивания областей;
4. текстурные методы.

2.1.1. Морфологические методы

Данный класс методов применяется для обработки бинарных (черно-белых) изображений. Они помогают извлекать некоторые характеристики, которые позже применяются для получения формы объекта [1]. Основой для данных методов служат логические операции

- AND
- OR
- XOR
- NOT

Кроме вышеперечисленных операций, часто применяют эрозию и дилатацию.

Основную роль в сегментации изображений играет компонента связности соседних пикселей. В контексте рассматриваемого метода, элементы

изображения называются связными, если они соседние и их уровни яркости удовлетворяют критерию сходства.

Будем рассматривать некоторое подмножество элементов S входного изображения и элементы $p, q \in S$. Говорят, что p, q связны в S , если между ними существует путь, все элементы которого принадлежат S . Соответственно, компонентой связности элемента p является множество связных с ним в S элементов. При этом множество S , содержащее только одну компоненту связности, называется связным множеством.

Рассмотрим связную компоненту Y из A . Пусть $p \in Y$, а отсюда можем получить все элементы компоненты Y из следующего соотношения, положив $X_0 = p$:

$$X_k = (X_{k-1} \oplus B) \cap A, \quad k = 1, 2, 3, \dots \quad (1)$$

B - подходящий примитив. A выступает в качестве ограничивающего примитива, что обусловлено значением разыскиваемых пикселей, равным 1. Пересечение с A на каждом шаге помогает исключить из результата дилатации значения, которые выпадают на нулевые элементы. Данный алгоритм возможно применить, если множество A состоит из конечного числа связных компонент и при этом известна как минимум одна точка для каждой из них. Работу алгоритма можно увидеть на рисунке 3.

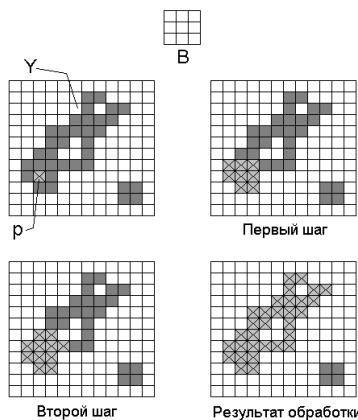


Рисунок 3 — Работа алгоритма для компоненты p

2.1.2. Пороговые методы

Класс методов, разделяющих изображение на части, основывающийся на пороговых значениях. Выделяют два основных типа таких методов:

- методы с глобальным порогом;
- методы с адаптивным порогом

Рассмотрим гистограммы изображений на рисунке 4, которые демонстрируют пример, когда большая часть пикселей сосредоточена вокруг двух центров. Для того, чтобы отделить объекты от фона, необходимо выбрать значение T и ввести следующее правило, для которого 1 - принадлежность объекту, а 0 - фону:

$$g(x, y) = \begin{cases} 1, & \text{if } f(x, y) > T \\ 0, & \text{if } f(x, y) \leq T \end{cases} \quad (2)$$

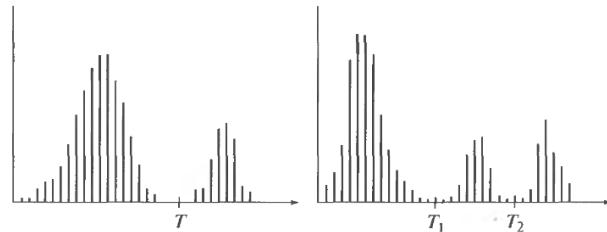


Рисунок 4 — Гистограмма изображений

В случае, когда можно выбрать общее значение T для всех пикселей изображения, порог называется глобальным, а если T зависит от пространственной ориентации по x, y , порог называют динамическим. Кроме того, если выбор значения T производится на основе значения функции $f(x, y)$, порог называют адаптивным.

При применении метода, основанного на глобальным пороге, процедура разделяет входное изображение на две области: объект и фон. Данный метод показывает хорошие результаты только в случае контролируемого

освещения. На рисунке 5 видим исходное изображение, его гистограмму и результат работы алгоритма пороговой сегментации с глобальным порогом.

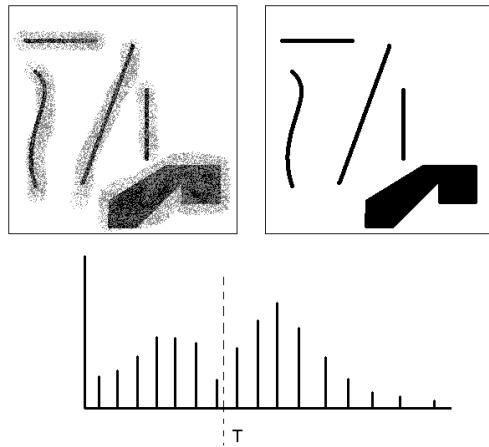


Рисунок 5 — Использование глобального порога

Для данного метода существует возможность реализовать алгоритм автоматического выбора порога, например, следующим образом:

1. выбрать начальную оценку порога;
2. произвести сегментацию на области G_1 и G_2 , используя данную оценку T ;
3. вычислить средние значения яркостей μ_1, μ_2 областей G_1 и G_2 соответственно;
4. пересчитать порог по правилу $T = \frac{1}{2}(\mu_1 + \mu_2)$;
5. повторять выполнение шагов 2-4, пока результаты соседних вычислений не окажутся меньше некоторого ε .

Однако при случае, когда мы не можем контролировать освещение, что бывает значительно чаще, вышеописанный метод работает некорректно. Для улучшения результатов в таких условиях, был разработан метод с применением адаптивного порога. Ключевым нововведением является разбиение изображение на подобласти, в которых ищутся пороговые значения,

не зависящие друг от друга. Для такого разбиения используется дисперсия освещения, которая вычисляется по формуле:

$$\sigma^2(z) = \sum_{i=0}^{L-1} (z_i - m)^2 p(z_i) \quad (3)$$

Визуализация применения алгоритмов с глобальным и адаптивным порогами, как и их сравнение, приведены на рисунке 6.

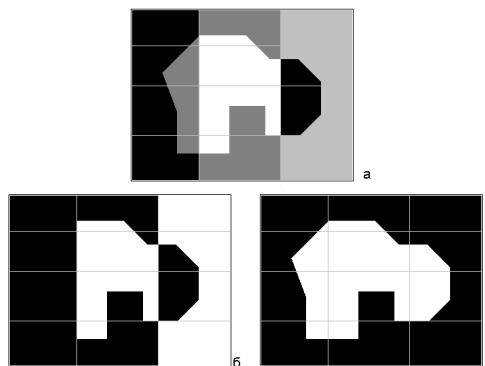


Рисунок 6 — Сравнение обработки изображения глобальным и адаптивным порогами

2.1.3. Метод водоразделов

Метод водоразделов является одним из основных алгоритмов наращивания областей [2]. Он рекурсивно группирует пиксели в области по заданным критериям.

В ходе работы данного алгоритма производится обнаружение и устранение разрывов, пороговая обработка и обработка областей, что позволяет ему показывать более высокие результаты в сравнении с предыдущими подходами. Название алгоритма было получено из представления изображения в виде трёхмерной поверхности, а воды в виде уровня яркости пикселей. Таким образом на "поверхности" оказывается несколько видов точек: точки локального минимума, точки на "склоне" точки на "гребнях возвышенностей".

Основной алгоритм представим в таком виде:

1. "вода" заполняет трёхмерную поверхность начиная с локальных минимумов;
2. в момент, когда "вода" с разных сторон гребня готова слиться, устанавливаем водораздел;
3. алгоритм завершает свою работу, когда над "поверхностью воды" остаются только водоразделы.

Пример работы данного алгоритма можно увидеть на рисунке 7.



Рисунок 7 — Метод водоразделов

Кроме классического метода, существует метод водораздела с маркерами, которые разделяют на принадлежащие объекту и принадлежащие фону. Для него необходима предобработка изображения и выбор критериев для маркеров. Например, пусть внутренний маркер- область, окруженная точками с большим значением яркости, её точки образуют компоненту связности (см. п. 2.1.1) и имеют одинаковую яркость. После этого применяется алгоритм сегментации методом водоразделов с условием, что локальные маркеры рассматриваются в качестве локальных минимумов. Пример сегментации методом маркерных водоразделов приведён на рисунке 8.



Рисунок 8 — Метод маркерных водоразделов

2.1.4. Текстурные методы

Текстурные методы используют диффузные свойства объект, над которым производится анализ [3]. В качестве характеристики структуры возьмём элементарную структурную единицу, которая повторяется в пространстве. Исходя из этого описать текстуру можно заданием такой элементарной единицы и правилом повторения. Анализ усложняется тем, что в структуре, как и в периодическом повторении, могут существовать случайные отклонения. Текстуры могут выглядеть по-разному при изменении масштаба, что приводит к необходимости многомасштабного анализа текстуры.

Для текстурного анализа применяются фундаментальные текстурные операторы: среднее значение, дисперсия, масштаб, ориентация. Они используются на разных уровнях обработки изображений. Например, при вычислении локальной ориентации и масштаба можно еще раз применить операторы усреднения и дисперсии, но не к среднему значению и дисперсии уровней яркости, а к локальной ориентации и масштабу.

Данные четыре оператора можно выделить в два класса: дисперсия и среднее значение не зависят от масштаба и поворота, а в то время как масштаб и ориентация как раз и определяют их.

Текстуру можно описать с использованием статистических характеристик, которые определяются по гистограмме яркости изображения или его областей.

Если z - случайная величина, определяющаяся яркостью элементов изображения, $p(z_i), i = 0, 1, 2, \dots, L - 1$ - гистограмма данного изображения с L числом уровней яркости, то центральный момент порядка n такого z вычисляется по формуле :

$$\mu_n(z) = \sum_{i=0}^{L-1} (z_i - m)^n p(z_i) \quad (4)$$

m - среднее значение яркости изображения $m = \sum_{i=0}^{L-1} z_i p(z_i)$ $\mu_0 = 1, \mu_1 = 0$ из соотношений выше. Дисперсия (второй момент) $\sigma^2(z) = \mu_2(z)$ выступает

в качестве меры яркостного контраста, что используется для реализации дескрипторов относительно гладкости. Следующая величина равна 0 в местах постоянной яркости (нулевая дисперсия) и стремится к 1 для больших значений $\sigma^2(z)$:

$$R = 1 - \frac{1}{1 + \sigma^2(z)} \quad (5)$$

Третий момент характеризует асимметрию гистограммы

$$\mu_3(z) = \sum_{i=0}^{L-1} (z - m)^3 p(z_i) \quad (6)$$

Помимо прочего, можно выделить среднюю энтропию:

$$e = \sum_{i=0}^{L-1} p(z_i) \log_2 p(z_i) \quad (7)$$

И однородность:

$$U = \sum_{i=0}^{L-1} p^2(z_i) \quad (8)$$

2.2. Методы глубокого обучения

Вместе с развитием методов глубокого обучения, улучшились и совершенствовались методы сегментации изображений с их применением. В настоящий момент существует множество архитектур нейронных сетей, позволяющих решать задачи сегментации. Бегло рассмотрим некоторые из них.

2.2.1. Fully Convolutional Network

FCN используется для преобразования входного изображения до меньших разрешений. Для этого производится серия свёрток. Данный набор свёрточных операций также называют кодировщиком (encoder). На следующем этапе изображение декодируется с применением билинейную интерполяцию или через серию транспонированных свёрток, носящих название декодер (decoder) [4]. Смотреть рисунок 9.

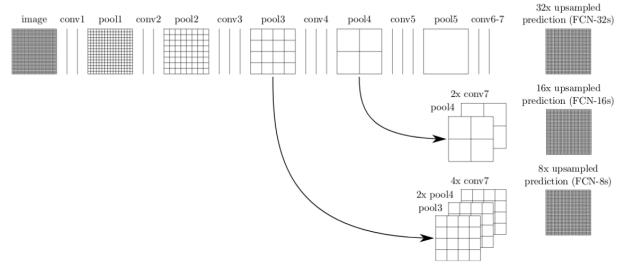


Рисунок 9 — Прямое и обратное преобразование FCN

Базовая архитектура хоть и эффективно, но имеет ряд недостатков, среди которых — наличие артефактов, которые расположены в шахматном порядке и связаны с неравномерным перекрытием выходов при операции транспонированной свертки (см. рисунок 10).

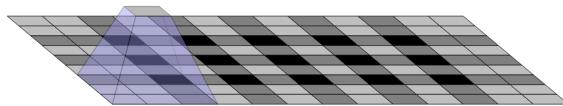


Рисунок 10 — Формирование артефактов FCN

Следующий недостаток- низкая разрешающая способность на краях изображения. Она возникает из-за потери информации в процессе кодирования.

Для решения данных проблем, присущих базовой модели FCN, было предложено несколько решений. Некоторые из них представлены ниже.

2.2.2. U-Net

Рассмотрим данную архитектуру более подробно, так как именно она применялась для задачи автоматического обнаружения клеточных ядер на биомедицинских изображениях, решаемой в рамках данной курсовой работы.

U-Net является архитектурой CNN (англ. Convolutional Neural Network), применяющейся в области сегментации изображений. Она включает в себя сужающийся путь, который служит для получения контекста,

и симметричный расширяющийся путь, позволяющий добиться точной локализации объекта [5]. Архитектура сети представлена на рисунке 11.

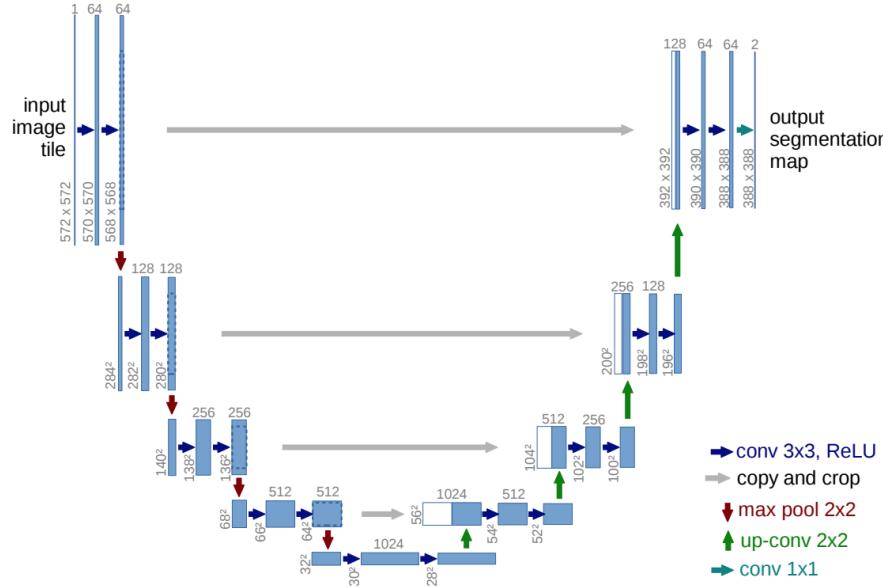


Рисунок 11 — Архитектура сети U-Net

На приведенном выше изображении мы видим пример архитектуры U-net для изображения, имеющего минимальное разрешение 32x32 пикселя. Синие поля соответствуют многоканальным картам свойств объектов, где количество каналов обозначено над ними. Размер $x - y$ указан в левом нижнем углу полей. Белые прямоугольники представляют собой скопированные карты свойств. Стрелки служат для обозначения различных операций.

Слева на схеме изображен сужающий путь, который характерен для свёрточных нейронных сетей. В данном случае он состоит из двух применяющихся свёрток размером 3x3, после которых следует активационная функция ReLU и операция Max Pooling 2x2 с шагом 2, которая позволяет понижать разрешение изображения.

Каналы свойств вырастают в два раза каждый раз, когда проводится процедура понижающей дискретизации. В расширяющем же пути, на каждом шаге происходит сверхдискретизация карты свойств, после чего

применяется ряд операций: свёртка 2×2 , уменьшающая количество каналов признаков, объединение с прообразом карты свойств из сужающего пути и две свёртки 3×3 , после которых снова следует функция ReLU. В конце 64-компонентный вектор признаков приводится к требуемому количеству классов с помощью свёртки 1×1 .

2.2.3. Модель тирамису

Данная модель схожа с вышеописанной архитектурой U-Net, но для прямой и транспонированной были использованы плотные блоки. Они состоят из нескольких свёрточных слоёв, а входами служат отображения признаков всех предыдущих слоёв (см. рисунок 12). Результирующая сеть эффективна в смысле параметров и лучше работает с полученными признаками на старых слоях.

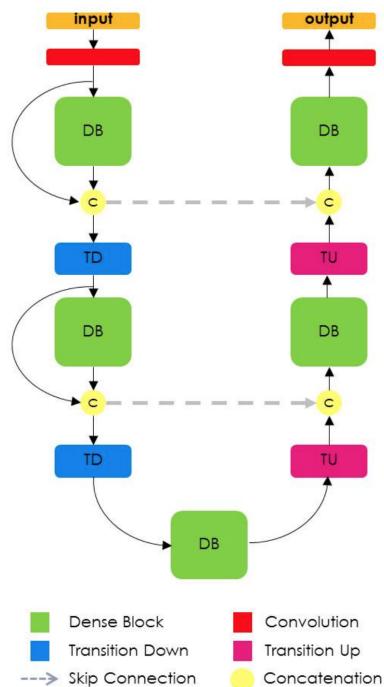


Рисунок 12 — Архитектура сети Тирамису

Несмотря на высокую эффективность, главным недостатком такой сети является низкая эффективность работы с памятью, что обусловле-

но операций объединения нескольких фреймворков машинного обучения, Поэтому для её работы требуются мощные кластеры GPU.

3. Сегментация клеточных ядер на биомедицинских снимках

В качестве практической части данной курсовой работы была взята задача автоматического обнаружения и сегментации клеточных ядер на биомедицинских снимках. Такой выбор обуславливается важностью решаемой проблемы, так как идентификация клеточных ядер является отправной точкой для большинства анализов, ведь большинство из 30 триллионов клеток человеческого тела содержат ядра, в которых содержится ДНК, программирующая каждую клетку. Нахождение ядер позволяет исследователям идентифицировать каждую отдельную клетку в образце, и измеряя, как клетки реагируют на различные обработки, исследователь может понять основные биологические процессы.

3.1. Входные данные

В качестве входных данных был использован data set, предоставленный Kaggle в рамках "Data Science Bowl 2018". Этот набор данных содержит большое количество изображений сегментированных ядер. Изображения были получены в различных условиях и различаются по типу клеток, масштабу и модальности изображения (светлое поле или флуоресценция). Структура описанного data set'a приведена на рисунке 13.

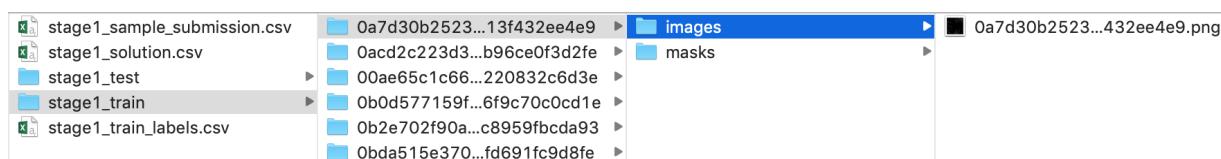


Рисунок 13 – Структура набора данных

Каждое изображение имеет уникальное название ImageId. Файлы, принадлежащие изображению, содержатся в папке с таким же ImageId, в которой находятся две подпапки:

- images - содержит файл изображения;
- masks - содержит сегментированные маски каждого ядра. Эта папка включена только в тренировочный набор. Каждая маска содержит одно ядро. Маски не могут перекрываться (ни один пиксель не принадлежит двум маскам).

Пример тренировочного изображения с обработанными масками, изначально представляющими собой набор бинарных изображений, можно видеть на рисунке 14.

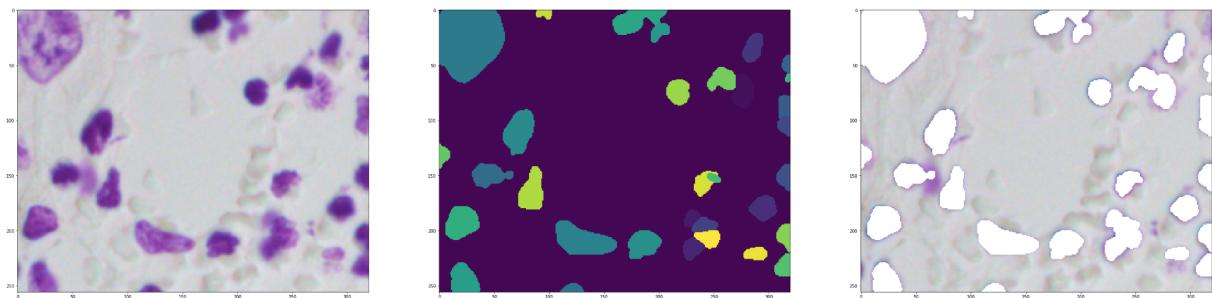


Рисунок 14 — Пример тренировочного изображения с его масками

3.2. Программная реализация и технические детали

Ввиду отсутствия оборудования с NVidia GPU, для повышения скорости обучения модели было принято решение производить вычисления в облачном сервисе Google Colaboratory, который предоставляет возможность использовать GPU NVIDIA Tesla T4. В связи с этим программный код был написан на языке Python с использованием оболочки Jupyter Notebook.

Для реализации проекта были использованы следующие библиотеки:

- NumPy - для работы с массивами и математическими функциями над ними;
- Scikit-image - для обработки изображений;

- TensorFlow - для построения и тренировки нейронной сети;
- Keras - для работы с сетями глубинного обучения (надстройка над фреймворками DeepLearning4j, TensorFlow и Theano).

3.2.1. Инициализация данных

Начальная инициализация данных состоит в считывании всех уникальных ImageId изображений и задании некоторых параметров, необходимых для обучения модели.

Листинг 3.1. Начальная инициализация данных

```
TRAIN_PATH = 'stage1_train/'
TEST_PATH = 'stage1_test/'

train_ids = next(os.walk(TRAIN_PATH))[1]
test_ids = next(os.walk(TEST_PATH))[1]

# Set some parameters
warnings.filterwarnings('ignore', category=UserWarning, module=''
skimage')
seed = 42
random.seed = seed
smooth = 1.
epochs = 50
```

Инициализация изображений из набора данных производится посредством вызова реализованных функций `get_train_images()` (листинг 3.2) и `get_test_images()` (листинг 3.3). Для тренировочной выборки функция возвращает массивы матриц `IMG_train` и `MASK_train`, содержащие исходные изображения и их маски. Для тестовой выборки маски отсутствуют, поэтому ограничиваемся инициализацией изображений и сохранением их исходных размеров, так как исходные изображения имеют разный размер и существует необходимость его изменения до стандартного, в данном случае был выбран 256×256 пикселей как оптимальный.

Листинг 3.2. Инициализация тренировочной выборки

```
def get_train_images(train_image_ids, path, aug=0, img_width=256,
                     img_height=256, img_channels=3):
    IMG_train = np.zeros((len(train_image_ids), img_height,
                          img_width, img_channels), dtype=np.uint8)
    MASK_train = np.zeros((len(train_image_ids), img_height,
                          img_height, 1), dtype=np.bool)

    print('Loading and resizing train images/masks')
    sys.stdout.flush()
    for n, image_id in tqdm(enumerate(train_image_ids), total=
                           len(train_image_ids)):
        #Image loading and resizing
        image_path = os.path.join(path, image_id, 'images',
                                  '{}.png'.format(image_id))
        image = imread(image_path)[:, :, :img_channels]
        image = resize(image, (img_height, img_width), mode=
                       'constant', preserve_range=True)
        IMG_train[n] = image

        #Masks loading and resizing
        path_mask = path + image_id
        mask = np.zeros((img_height, img_width, 1), dtype=np.
                        bool)
        for mask_file in next(os.walk(path_mask + '/masks/'))[2]:
            mask_ = imread(path_mask + '/masks/' + mask_file)
            mask_ = np.expand_dims(resize(mask_, (img_height,
                                                 img_width), mode='constant',
                                         preserve_range=True),
                                  axis=-1)
            mask = np.maximum(mask, mask_)
        MASK_train[n] = mask

    return IMG_train, MASK_train
```

Листинг 3.3. Инициализация тестовой выборки

```
def get_test_images(test_image_ids, path, img_width=256, img_height=256, img_channels=3):
    # Get and resize test images
    IMG_test = np.zeros((len(test_image_ids), img_height,
                         img_width, img_channels), dtype=np.uint8)
    sizes_test = []
    print('Loading and resizing test images')
    sys.stdout.flush()
    for n, image_id in tqdm(enumerate(test_image_ids), total=len(
        test_image_ids)):
        image_path = os.path.join(path, image_id, 'images', '{}.
            png'.format(image_id))
        image = imread(image_path)[:, :, :img_channels]
        sizes_test.append([image.shape[0], image.shape[1]])
        image = resize(image, (img_height, img_width), mode='
            constant', preserve_range=True)
        IMG_test[n] = image
    return IMG_test, sizes_test
```

Пример полученных данных приведен на рисунке 15 (маски продемонстрированы в нескольких режимах).

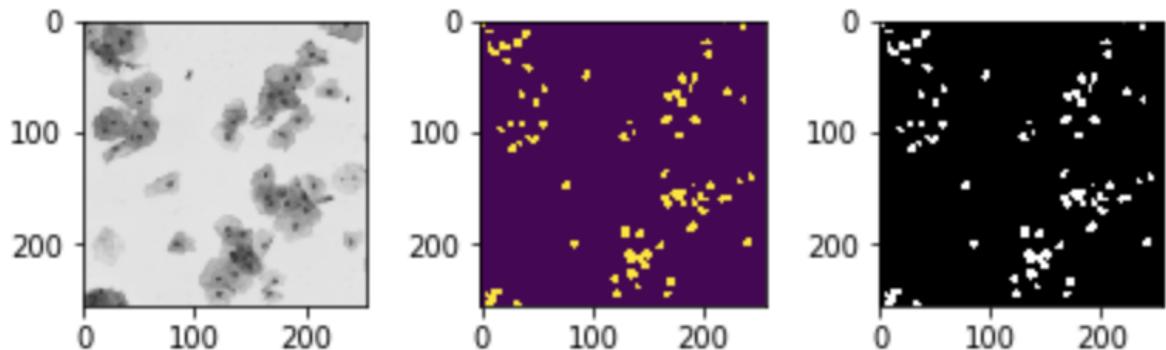


Рисунок 15 — Визуализация результата работы `get_train_images()`

3.2.2. Архитектура сети и обучение модели

Для решения поставленной задачи было принято решение использовать архитектуру U-Net, которая была рассмотрена в разделе 2.2.2.

Получение модели было выделено в отдельную функцию `get_unet()`, которую можно разделить на несколько смысловых частей и рассмотреть отдельно.

Сужающий путь (см. листинг 3.4) содержит ряд свёрток размером 3×3 и функции MaxPooling для перехода к следующему слою. Также использовался дропаут, который помогал предотвратить переобучение сети на тренировочных данных. В качестве активационной функции выбрана ReLU, а потому инициализатором весов был выбран НЕ-инициализатор [3].

Далее был реализован расширяющий путь (см. листинг 3.5), который содержит транспонированные свёртки 2×2 и свёртки 3×3 . Чтобы согласовать размерности получаемых матриц, используется функция `concatenate()`. На последнем шаге используется свёртка 1×1 для преобразования многокомпонентного вектора признаков в конечный результат.

Листинг 3.4. Сужающий путь U-Net функции get_unet()

```
c1 = Conv2D(16, (3, 3), activation='relu', kernel_initializer=
    'he_normal', padding='same') (s)
c1 = Dropout(0.1) (c1)
c1 = Conv2D(16, (3, 3), activation='relu', kernel_initializer=
    'he_normal', padding='same') (c1)
p1 = MaxPooling2D((2, 2)) (c1)

c2 = Conv2D(32, (3, 3), activation='relu', kernel_initializer=
    'he_normal', padding='same') (p1)
c2 = Dropout(0.1) (c2)
c2 = Conv2D(32, (3, 3), activation='relu', kernel_initializer=
    'he_normal', padding='same') (c2)
p2 = MaxPooling2D((2, 2)) (c2)

c3 = Conv2D(64, (3, 3), activation='relu', kernel_initializer=
    'he_normal', padding='same') (p2)
c3 = Dropout(0.2) (c3)
c3 = Conv2D(64, (3, 3), activation='relu', kernel_initializer=
    'he_normal', padding='same') (c3)
p3 = MaxPooling2D((2, 2)) (c3)

c4 = Conv2D(128, (3, 3), activation='relu', kernel_initializer
    ='he_normal', padding='same') (p3)
c4 = Dropout(0.2) (c4)
c4 = Conv2D(128, (3, 3), activation='relu', kernel_initializer
    ='he_normal', padding='same') (c4)
p4 = MaxPooling2D(pool_size=(2, 2)) (c4)

c5 = Conv2D(256, (3, 3), activation='relu', kernel_initializer
    ='he_normal', padding='same') (p4)
c5 = Dropout(0.3) (c5)
c5 = Conv2D(256, (3, 3), activation='relu', kernel_initializer
    ='he_normal', padding='same') (c5)
```

Листинг 3.5. Расширяющий путь U-Net функции get_unet()

```
u6 = Conv2DTranspose(128, (2, 2), strides=(2, 2), padding='same') (c5)
u6 = concatenate([u6, c4])
c6 = Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same') (u6)
c6 = Dropout(0.2) (c6)
c6 = Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same') (c6)

u7 = Conv2DTranspose(64, (2, 2), strides=(2, 2), padding='same') (c6)
u7 = concatenate([u7, c3])
c7 = Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same') (u7)
c7 = Dropout(0.2) (c7)
c7 = Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same') (c7)

u8 = Conv2DTranspose(32, (2, 2), strides=(2, 2), padding='same') (c7)
u8 = concatenate([u8, c2])
c8 = Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same') (u8)
c8 = Dropout(0.1) (c8)
c8 = Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same') (c8)

u9 = Conv2DTranspose(16, (2, 2), strides=(2, 2), padding='same') (c8)
u9 = concatenate([u9, c1], axis=3)
c9 = Conv2D(16, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same') (u9)
c9 = Dropout(0.1) (c9)
c9 = Conv2D(16, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same') (c9)

outputs = Conv2D(1, (1, 1), activation='sigmoid') (c9)
```

В ходе обучения существует необходимость оценки качества работы алгоритма, поэтому был использован коэффициент подобия Серенсена-Дайса, позволяющий оценить качество сегментации. Он выражается следующей функцией:

$$D = \frac{2P(A \cap B)}{P(A) + P(B)} \quad (9)$$

Результатом будет значение в диапазоне $[0; 1]$, где 1 - абсолютное сходство сегментации на двух изображениях. Программная реализация вычисления коэффициента Серенсена-Дайса представлена листинге 3.6.

Листинг 3.6. Расширяющий путь U-Net функции get_unet()

```
def dice_coef(mask_true, mask_pred):
    mask_true_f = K.flatten(mask_true)
    mask_pred_f = K.flatten(mask_pred)
    intersection = K.sum(mask_true_f * mask_pred_f)
    return (2. * intersection + smooth) / (K.sum(mask_true_f) +
    K.sum(mask_pred_f) + smooth)
```

Для тренировки построенной сети были выбраны следующие параметры (использовалась кросс-валидация):

- количество эпох: 50;
- размер валидационной выборки: 10%;
- размер пакета: 16.

Кроме того, был установлен параметр ранней остановки, если прохождение последующих 5 эпох не дает прироста точности относительно последнего лучшего результата. Лучшая модель сохранялась в файл model-dsbowl.h5.

3.2.3. Визуализация результатов

В процессе обучения были получены следующие результаты:

- пройдено эпох: 45;
- loss на тренировочной выборке: 0.0622;

- loss на валидационной выборке: 0.0722;
- коэффициент Дайса на тренировочной выборке: 0.8800;
- коэффициент Дайса на валидационной выборке: 0.8721;

Изменение данных параметров с ростом эпох обучения можно увидеть на рисунке 16.

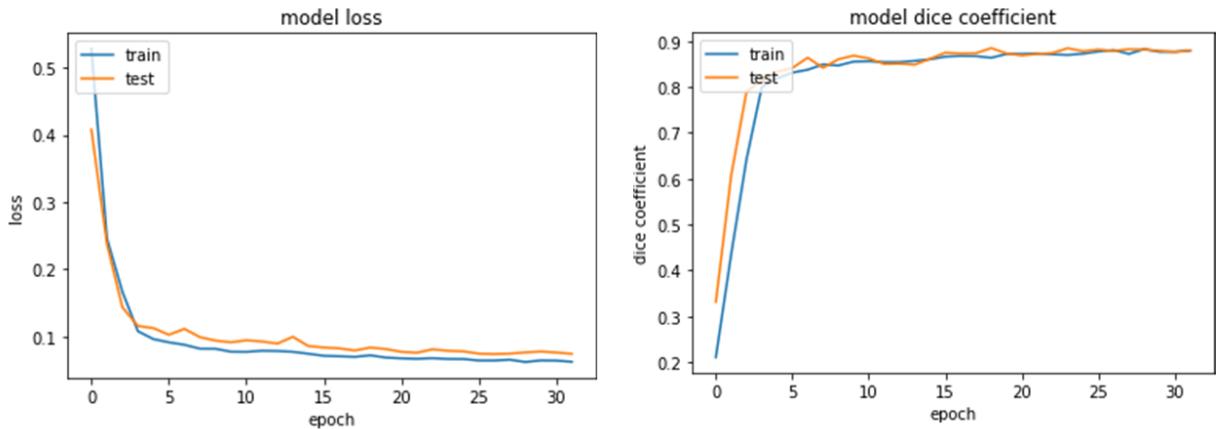


Рисунок 16 — Визуализация истории обучения модели

После процесса обучения модели было произведено предсказание для всех изображений в наборе данных. Для предсказанных масок был установлен порог 0.5, после чего выполнена процедура Thresholding'a по данному порогу. На рисунке 17 показаны результаты предсказаний для разных классов изображений:

- изображение из тренировочной выборки (а);
- изображение из валидационной выборки (б);
- изображение из тестовой выборки (в).

Для тренировочного и валидационного изображения помимо предсказанной маски (справа) продемонстрирована эталонная сегментация (в центре).

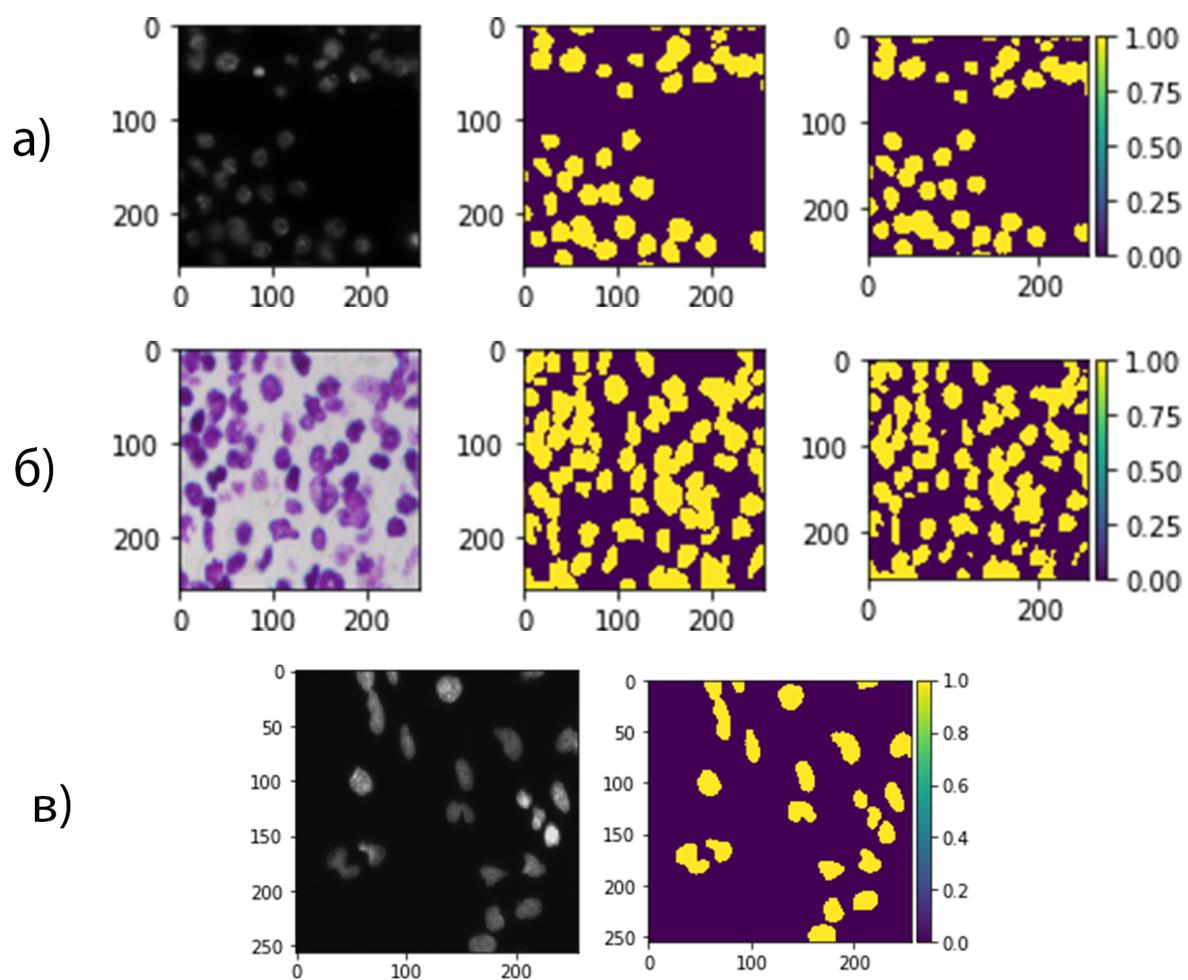


Рисунок 17 — Предсказания модели

Как видно из представленных выше результатов, качество сегментации изображений, схожих с изображениями в обучающей выборке, довольно высокий. Потери относительно эталонной сегментации лежат в пределе 8%.

Заключение

Таким образом, в рамках курсовой работы была формально описана задача сегментации изображений, рассмотрены основные методы и реализован алгоритм сегментации биомедицинских снимков.

Для практической части была взята задача семантической сегментации, которая заключалась в автоматическом обнаружении клеточных ядер. Выбор обусловлен тем, что решение данной проблемы помогает быстрее проводить исследования в области медицины. Было протестировано несколько вариаций решения. Так, классический метод водоразделов давал излишнюю сегментацию, а вариант с маркерами не удовлетворяет основному условию- автоматическое выполнение. Наилучшие результаты были получены с применением метода глубокого обучения, основывающегося на сети U-Net. Это обусловлено тем, что dataset был относительно небольшого размера, а сами изображения в маленьком разрешении, что свойственно для задач, использующих медицинские изображения. Сама же архитектура сети U-Net ориентирована на работу с биомедицинскими снимками.

Рассмотрев полученные изображения мы видим, что loss сегментации на валидационной выборке составляет примерно 8%, однако этот результат может быть ухудшен, если при teste будут подаваться изображения клеточных ядер, существенно отличающихся от таковых в обучающей выборке. Кроме того, результат может быть улучшен, если удастся подобрать лучшие параметры обучения или применив более сложную архитектуру. Например, вместо классического U-Net возможно применение его рекуррентного варианта. Главные изменения будут заключаться в следующем:

- ConvLSTM слой;
- добавлен компонент функции потерь, который штрафует CNN в случае, если было "выучено" слишком много ядер;
- применение Венгерского алгоритма для оптимального совмещения разметки с предсказанными объектами.

Помимо этого возможен синтез с алгоритмом Deep Watershed Transform, который решает некоторые проблемы классического метода, связанные со слишком парцелярной сегментацией [6]. Для этого используются 2 разные CNN, которые определяют энергию (высоту ландшафта) и единичные векторы, направленные к границам или от границ объектов, что помогает CNN "выучить" энергию и границы объектов.

Из рассмотренного варианта решения мы видим, что классические методы не утратили своей актуальности и могут быть эффективно объединены с методами глубокого обучения, что значительно повышает эффективность как самих классических методов, так и методов машинного обучения, в состав которых они входят, а большое количество задач, требующих их применения, служит весомым доказательством актуальности и целесообразности дальнейшего проведения исследований в данной области.

Список литературы

- [1] Уалиева И. М. Сегментация микроскопических изображений эпителиальных клеток / И. М. Уалиева, Ж. К. Жукешева. — Москва 2016. — 96 с.
- [2] Тропченко, А. Ю. Методы вторичной обработки и распознавания изображений: учеб. пособие / А. Ю. Тропченко, А.А. Тропченко.— СПб: Университет ИТМО, 2015. – 215 с.
- [3] Sergios Theodoridis, Rama Chellappa. Academic Press Library in Signal Processing, Volume 4. - 2013. - T. 12.
- [4] Long J., Shelhamer E., Darrell T. Fully Convolutional Networks for Semantic Segmentation, arXiv preprint arXiv:1411.4038 – 2016
- [5] Ronneberger O., Fischer P., Brox T. U-Net: Convolutional Networks for Biomedical Image Segmentation, arXiv preprint arXiv:1505.04597 – 2015
- [6] Bai M., Urtasun R. Deep Watershed Transform for Instance Segmentation, arXiv preprint arXiv:1611.08303 – 2016