МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

02.03.03 Математическое обеспечение и администрирование информационных систем
(код и наименование направления подготовки, специальности)

Мобильные и сетевые технологии
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Разработка ПО, основанного на алгоритмах машинного обучения для работы с медицинскими данными»

Студент П.И. Баринов
(И.О. Фамилия) (личная подпись)

Руководитель канд. тех. наук, доцент, В.С. Климов
(Ученая степень, звание, И.О. Фамилия)

Консультант М.В. Дайнеко
(Ученая степень, звание, И.О. Фамилия)

Тольятти 2021

Аннотация

Темой данной выпускной квалификационной работы является «Разработка ПО, основанного на алгоритмах машинного обучения для работы с медицинскими данными».

Целью данной выпускной квалификационной работы является разработка программного обеспечения, которое позволяет визуализировать циркулирующие опухолевые клетки на образцах крови, используя методы машинного обучения.

Объектом исследования является процесс визуализации циркулирующих опухолевых клеток на образце крови, с применением методов машинного обучения.

Предметом исследования является программное обеспечение, которое визуализирует циркулирующие опухолевые клетки на образце крови, используя методы машинного обучения.

Задачами ВКР являются:

– провести анализ уже существующих методов определения клеток в крови;

– определить метод визуализации клеток и метод машинного обучения;

– реализовать методы в виде программного обеспечения;

– провести испытания реализованного алгоритма;

– сделать вывод об эффективности реализованного алгоритма;

Первая глава содержит описание уже существующих методов обработки медицинских данных. Во второй главе содержится описание методов и их последующая реализация. В третье главе проводится тестирование полученного программного обеспечения. В заключении содержатся выводы, которые были сделаны в результате проделанной работы.

В работе было использовано 22 формулы, 16 рисунков и 20 источников. Общий объем квалификационной работы составил 43 страницы.

Abstract

The title of the graduation work is *Developing software based on machine learning algorithms for working with medical data*.

The graduation work is devoted to analyzing the problem, designing mathematical models with subsequent implementation, as well as testing the developed software.

The object of the study is the process of visualizing the circulating tumor cells on a blood sample using machine learning techniques.

The goal of this research is to develop software enabling us to visualize circulating tumor cells on blood samples using machine learning techniques.

To achieve the goal, the following tasks have to be performed:

– to analyze the existing methods of detecting tumor cells in blood;

– to choose the method of cell visualization and machine learning method;

– implement the methods chosen in the form of software;

– to test the implemented algorithm;

– to make a conclusion about the effectiveness of the implemented algorithm;

The process of visualizing the circulating tumor cells is time-consuming. Therefore, a specialist is supposed to be precise when performing this task. The development and implementation of software based on machine learning algorithms are supposed to simplify the process of visualizing the circulating tumor cells and increase its efficiency.

The first chapter of the research reveals the existing methods of medical data processing. The second chapter dwells on the model to which we aspire when implementing the software. The third chapter presents the results of testing the program performance. In conclusion, it should be highlighted that the obtained results show that the developed software for visualization of the circulating tumor cells works effectively and accurately.

Содержание

Введение

На сегодняшний день медицина становится наукоемкой и быстроразвивающейся благодаря активному развитию технологий и внедрению их в работу здравоохранительных организаций. В медицинской практике существует ряд заболеваний, для лечения которых принципиально важно своевременное диагностирование. Рак, диабет, болезни сердца, пневмония – это лишь ряд диагнозов, которые требуют срочного вмешательства. Сегодня многие исследователи, в том числе биоинформатики, выдвигают теорию о целесообразности использования ресурсов искусственного интеллекта в области принятия решений в медицинской сфере. Машинное обучение считается одним из самых интенсивно развивающихся направлений искусственного интеллекта, о чем свидетельствуют научные труды последних десятилетий. Однако, многие предлагаемые решения работают на специфических выборках данных, что делает созданные продукты неприменимыми во многих медицинских учреждениях. Большинство сложных моделей требует огромного количества ресурсов, как вычислительных, так и временных, поэтому часто не могут быть использованы на практике. Простейшие алгоритмы зачастую дают низкую точность, имеют непростительно большой для медицинских задач процент ошибки или большую вероятность ложного срабатывания. Отдельной проблемой является вопрос целесообразности использования интеллектуальных систем в медицинских учреждениях. Высокая стоимость приобретения, сложность внедрения, нежелание врачей осваивать работу с компьютером, неудобство использования системы медперсоналом и её ограниченная функциональность часто не позволяют массово внедрять системы по интеллектуальному анализу данных в медицинских учреждениях.

Целью данной выпускной квалификационной работы стало использование методов визуализации и машинного обучения для определения циркулирующих опухолевых клеток в крови.

Для достижения поставленной цели необходимо решить определенные задачи:

1. Провести анализ научной и технической литературы.
2. Провести анализ уже существующих методов определения клеток в крови.
3. Определить метод визуализации клеток.
4. Определить метод машинного обучения.
5. Реализовать методы в виде программного обеспечения.
6. Провести испытания реализованного алгоритма.
7. Определить эффективность программного обеспечения.
8. Сделать вывод об эффективности реализованного алгоритма.

1 Анализ задания на исследование

1.1 Описание поставленной задачи

Рак становится ведущей причиной смерти во всем мире. В 2020 году, по оценкам, почти 10 миллионов человек по всему миру умерли от рака, и было зафиксировано 19,3 миллионов новых случаев заболевания. Таким образом, почти каждая шестая смерть была вызвана раком, и для каждого из нас риск развития рака в течение нашей жизни составляет около 38%, что влияет на большую часть нашей жизни.

Для большинства видов рака биопсия, то есть удаление и микроскопический анализ раковой ткани, является основным инструментом диагностики. Биопсия во время постановки диагноза является инвазивной и часто невоспроизводимой, с помощью которой невозможно получить представление о прогрессировании заболевания и эффективности лечения. Другая проблема заключается в том, что большинство смертей, связанных с раком, вызваны не самой первичной опухолью, а метастазами во вторичных участках. Хотя метастазы разделяют большинство характеристик первичной опухоли, у них есть мутации, которые не фиксируются биопсией первичной опухоли, но могут быть важны для принятия решений о лечении.

Решающим моментом в формировании метастазов являются клетки, которые отделяются от первичной опухоли и проникают в кровоток. Эти клетки называются циркулирующими опухолевыми клетками (далее ЦОК). После попадания в кровоток ЦОК циркулируют по сердечно-сосудистой системе. Большинство из них будет уничтожено, но некоторые ЦОК могут экстравазировать кровоток во вторичном месте и образовывать новые опухоли.

ЦОК могут быть обнаружены в крови всех больных метастатическим раком при условии достаточно большого объема крови в качестве образцов и, по прогнозам, будут присутствовать до образования отдаленных метастазов.

Генетический и фенотипический состав ЦОК демонстрирует большую гетерогенность.

Примечательно, что ЦОК отражают не только характеристики первичной биопсии, но и изменения, наблюдаемые в местах метастазирования, и, таким образом, могут рассматриваться как “жидкая биопсия”, охватывающая все участки опухоли. В отличие от биопсии тканей, получение ЦОК от пациента неинвазивно, и эти жидкие биопсии могут повторяться. Таким образом, ЦОК можно использовать для мониторинга состояния заболевания пациента, а также для проверки работоспособности терапии. Потенциально ЦОК можно даже использовать для принятия решений о лечении, тем самым прокладывая путь к персонализированной медицине для больных раком.

Хотя ЦОК обладают большим потенциалом в качестве биомаркера рака, использование этого потенциала частично затруднено из-за их сложного обнаружения. По сравнению с другими клетками крови, ЦОК встречается редко, и у 50% пациентов с метастазами в 1 мл крови можно обнаружить только от 1 до 10 ЦОК. Однако стоит учесть, что в 1 мл крови содержится около 5×10^6 лейкоцитов и около 5×10^9 эритроцитов.

1.2 Обзор реализованных алгоритмов

Чтобы обнаружить и подсчитать ЦОК в крови пациентов, их необходимо отделить от основной массы других клеток в крови. В настоящее время единственной системой, которая разрешена министерством здравоохранения Российской федерации, для мониторинга пациентов, является система «Онкопаспорт», которая идентифицирует ЦОК у пациентов с раком легких, предстательной железы, толстой кишки, молочной железы и поджелудочной железы. В США, аналогичная система называется CellSearch и также, является единственной в своем роде системой, которую одобрили в министерстве здравоохранения США. В данной системе, для отслеживания

ЦОК, используют ферромагнитные жидкости (магнитные наночастицы длиной 175 нанометров), покрытые антителами клеточной адгезии эпителия, применяют для отделения ЦОК от большинства других клеток с помощью магнитов. После этого ЦОК и оставшиеся лейкоциты (примерно 100×10^3 клетки) флуоресцентно окрашиваются, выделяя клеточное ядро и цитоскелет. Автоматический флуоресцентный сканирующий микроскоп используется для отображения всех клеток, генерируя около 180 изображений на флуорофор, для образца. На основе их флуоресцентной экспрессии кандидаты ЦОК (объекты, окрашенные красителем нуклеиновой кислоты и цитокератином) представляются пользователю в галерее миниатюр и проверяются вручную, что делает подсчеты ЦОК субъективными. То же самое справедливо и для большинства других методов – хотя в последнее время разрабатывается все больше и больше технологий для визуализации ЦОК, большинство из которых не имеют (полу) автоматизированного конвейера обработки изображений для увеличения количества замеченных ЦОК.

Тем не менее, сообщенные показатели ЦОК по-прежнему субъективны и подвержены различиям между специалистами. Было исследовано, что обширное обучение может уменьшить вариации в назначении объектов в качестве ЦОК, но полностью устранить их невозможно, пока процесс не будет полностью автоматизирован и объективен. Это сильно мотивирует разработку полностью автоматизированных компьютерных алгоритмов для обнаружения и подсчета ЦОК в пробах крови пациентов с метастазами.

1.3 Постановка задачи

Для выполнения задачи дипломной работы, требуется обучить алгоритм анализировать изображение на наличие клеток, после чего обучить сегментированию этих клеток, с последующей реализацией алгоритма на языке программирования. Для реализации программного обеспечения был выбран язык программирования MATLAB.

В результате реализованная программа должна иметь ряд функциональных возможностей, а именно:

– реализованная программа должна распознавать циркулирующие опухолевые клетки с изображений образцов крови;

– реализованная программа, в случае успеха, должна сегментировать клетки, возвращая образец с красным контуром вокруг циркулирующих опухолевых клеток;

– реализованная программа, после завершения процесса сегментации, должна предоставлять информацию об образце, используя свой интерфейс.

При разработке программного обеспечения, необходимо будет учесть вышеперечисленные требования к программе.

2 Математическое описание задачи

2.1 Анализ и выбор вычислительного метода для реализации алгоритма

2.1.1 Методы математического анализа изображений

Область математической визуализации — это довольно новая междисциплинарная область, возникшая с изобретением цифровой визуализации. С огромным количеством цифровых изображений, которые производятся в настоящее время, важность обработки изображений постоянно растет. Основные приложения можно найти в биомедицинских науках, робототехнике, спутниковых снимках, науках о земле, материаловедении, но в настоящее время также в автономном вождении и приложениях для смартфонов для обработки изображений и видео. Основными задачами, решаемыми обработкой изображений, являются автоматизированное шумоподавление изображений, сегментация, реконструкция, отслеживание и оценка потоков. Подполе математического анализа изображений сосредоточено как на разработке теоретической основы, так и на разработке новых инструментов обработки изображений, основанных на базовых математических моделях. Изображение либо рассматривается как непрерывная функция $u: \Omega \to \mathbb{R}$ с $\Omega \subset \mathbb{R}^d$ или как дискретная матрица $U \in \mathbb{R}^{n_1 \times \ldots \times n_d}$. Для RGB-изображений $d = 3$, а для изображений в оттенках серого $d = 2$. Оставшаяся часть этого раздела используется для краткого представления наиболее важных концепций математической визуализации, которые мы использовали в нашей работе. Начнем с введением вариационных методов для задач обработки изображений в непрерывной постановке.

2.1.1.1 Вариационные методы

Пусть $u \in U$ - изображение, которое мы хотим реконструировать, $K: U \to V$ – прямое отображение оператора из функционального пространства U в другое функциональное пространство V и $f \in V$ – функция, моделирующая измеренные данные, потенциально поврежденные шумом. Мы предполагаем, что:

$$f \approx Ku, \tag{1}$$

в результате чего возникает обратная задача. Эта обратная задача, как правило, некорректна, поскольку предположение о стабильности, в соответствии с Адамаром, нарушается из-за зашумленных или неполных данных. Прямой оператор K моделирует процесс формирования изображения, для данных микроскопии это может быть тождество $K = I$. Чтобы решить обратную задачу для u, мы минимизируем функционал J, состоящий из члена расхождения данных, измеряющего насколько хорошо выполняется равенство $f = Ku$, и функционала регуляризации $R(u)$, накладывающего некоторые предварительные ограничения на u, то есть:

$$J(u) = D(f, Ku) + \alpha R(u) \to \min_u, \tag{2}$$

где α – параметр регуляризации, уравновешивающий влияние обоих членов;

D – параметр выбора основывающийся на ожидаемом шуме в f и задаче обработки [20].

Данные, а также термин регуляризации адаптированы как к задаче обработки, так и к данным. Общими терминами несоответствия данных являются L^p-нормы $||Ku - f||_{L^p}$, в особенности для $p = 1$ и $p = 2$. Функционал регуляризации выбирается на основе предварительной информации, которая должна быть включена в модель. Распространенными вариантами являются кусочно-постоянная (общее изменение шумоподавления), гладкость (регуляризация градиента изображения по норме L^2) или разреженность (регуляризация по норме L^1). Чтобы

минимизировать функционал J, требуется некоторая форма производной от J. Если J не является дифференцируемой, для решения (2) потребуются более продвинутые алгоритмы оптимизации.

2.1.1.2 Сегментация с использованием активных контурных методов

В математическом изображении задача сегментации описывает задачу автоматического обнаружения областей, представляющих интерес в изображении u. В биомедицинской визуализации, сегментация часто используется либо в качестве этапа предварительной обработки для извлечения и отслеживания признаков, либо в качестве альтернативного этапа постобработки после реконструкции изображения для обнаружения областей, которые представляют интерес в реконструированном изображении. В то время как простые модели сегментации, такие как пороговое значение гистограммы, часто терпят неудачу для экспериментальных наборов данных, нелинейные вариационные методы обладают большим потенциалом. Существуют два способа определить регион – либо через опись края, либо через опись внутренней части. Ниже мы опишем экземпляр модели сегментации на основе регионов.

В 1989 году Мамфорд и Шах представили следующую вариационную модель сегментации изображений:

$$J^{MS}(u,C)\int_{\Omega}|f(x)-u(x)|^2dx + \\ +\alpha\times Length(C)+\beta\times\int_{\Omega\setminus C}|\nabla u(x)|^2dx\rightarrow\min_{u,C}, \quad (3)$$

где u – дифференцируемая функция, определяющая сегментацию;

C – контур изображения.

Данная модель позволяет решению u содержать острые края между областями изображения. Помимо контура, регуляризация градиента по L^2-норме обеспечивает гладкость u. Эта модель была адаптирована Чаном и Везем в 2001 году [4]. Они ограничивают u наличием только двух областей

изображения Ω_1 и Ω_2, имеющих постоянные значения интенсивности c_1 и c_2 соответственно. Опять же, C описывает контур, разделяющий обе области и тем самым определяющим сегментацию. Это можно формализовать следующим образом:

$$J^{CV}(c_1, c_2, C) = \lambda_1 \times \int_{\Omega_1} (f(x) - c_1)^2 dx + \lambda_2 \times$$
$$\times \int_{\Omega_2} (f(x) - c_2)^2 dx + \alpha \times Length(C) + \beta \times Area(\Omega_1), \quad (4)$$

где λ_1, λ_2, α, β – константы, взвешивающие соответствующий член;

c_1, c_2 – константы, которые необходимо определить при минимизации $J^{CV}(c_1, c_2, C)$.

Эта модель называется моделью “активный контур без ребер”. Во время минимизации (4) контур C развивается до тех пор, пока не достигнет минимума, который в идеальном случае описывает границы интересующего объекта. Визуализация этого процесса показана на рисунке 1.

Рисунок 1 – Визуализация контура, движущегося к конечной сегментации

2.1.1.3 Регуляризация полной вариации

Как упоминалось ранее, функционал регуляризации может быть использован для включения предварительных знаний о решении u в вариационную модель. Общее предположение в случае зашумленных данных состоит в том, что лежащее в основе изображение $u \in U$ почти везде гладкое. Поэтому методы линейной фильтрации, которые штрафуют,

например, L^2-норму градиента, предполагают, что изображение не содержит острых краев. Тем не менее, в большинстве изображений острые края необходимы для описания границ объекта. Поэтому Рудин, Ошер и Фатеми ввели понятие нелинейной регуляризации полной вариации (TV регуляризация) для обработки изображений [19]. Функционал TV определяется как:

$$TV(u) \coloneqq \sup_{\substack{g \in C_0^\infty(\Omega;\mathbb{R}^2) \\ ||g||_\infty < 1}} \int_\Omega u \nabla \times g d\mu = \int_\Omega |Du|, \tag{5}$$

где *Du* – градиент *u* в распределительном смысле.

Соответствующее пространство функций с ограниченной вариацией (от англ. Bounded Variation, сокр. BV), определяется как $BV(\Omega) \coloneqq \{u \in L^1(\Omega) | TV(u) < \infty\}$. Для функций u, которая содержится в пространстве Соболева $W^{1,1}$, справедливо, что:

$$TV(u) = ||\nabla \mathrm{u}||_{L^1}. \tag{6}$$

Это создает интуитивное понимание того, что TV штрафует колебания в u и обеспечивает разреженный градиент. Небольшие колебания будут сглажены, но большие значения градиента, возникающие на краях, сохраняются в решении, что приводит к кусочно-постоянным изображениям. Это, в свою очередь, мотивирует использование TV регуляризации в вышеупомянутой модели сегментации (4), где мы стремимся найти кусочно-постоянное изображение только с двумя значениями интенсивности. Используя формулу совместной площади, можно даже показать, что общее изменение характеристики точно соответствует длине контура *C* в (4).

Минимизация полной вариации с помощью метода самого крутого спуска приводит к следующему уравнению в частных производных, которое называется TV-потоком [1]:

$$u_t(t,x) = -\nabla \times \left(\frac{Du}{|Du|}\right) \text{ в } (0,\infty) \times \Omega;\ u(0,x) = f(x) \text{ для } x \in \Omega, \qquad (7)$$

где *Du* – градиент;

∇ – дивергенция.

Для этой формулы, мы предполагаем граничные условия Неймана [3]. Для $f(x) = 1_C(x)$ единственное решение (7) является:

$$u(t,x) = (1 - \lambda_C t) + 1_C(x) \text{ с } \lambda_C = \frac{Length(C)}{Area(C)}. \qquad (8)$$

Интуитивную интерпретацию того, почему TV-регуляризатор приводит к “мультяшным” изображениям, можно увидеть в (7). Если x лежит в гладкой области, а |*Du*| мал, то $\nabla * \left(\frac{Du}{|Du|}\right)$ разгладится достаточно сильно. С другой стороны, если x лежит на краю, а |*Du*| относительно велик это приведет лишь к незначительному сглаживанию. Поэтому минимизация функционала TV приводит к нелинейному процессу диффузии.

2.1.2 Машинное обучение

Целью машинного обучения, является разработка алгоритмов, которые учатся делать прогнозы на основе входных данных, не будучи явно запрограммированными на то, как делать эти прогнозы. Алгоритмы машинного обучения используют статистический анализ для своих прогнозов, который должен улучшиться, когда появятся новые входные данные, на которых алгоритм может “учиться”. Методы обучения можно разделить на контролируемые и неконтролируемые методы.

Для контролируемых методов нам требуется в качестве входных данных набор S из N экземпляров $x^i \in \mathbb{R}^n$ с требуемыми выходными данными $y^i \in \mathbb{R}^I$, поэтому $S = \{(x^i, y^i) \in \mathbb{R}^n \times \mathbb{R}^I, i = 1, \dots, N\}$. В случае проблемы классификации, y^i будет меткой, которая была дана входу x^i. Для контролируемого обучения, набор доступных примеров *S*, разбивается, на так называемый, обучающий набор S_{train}, который алгоритм использует для

обучения правильному прогнозированию, а также тестовый набор S_{test} для оценки производительности алгоритма на примерах, которые он ранее не видел. Если $f : \mathbb{R}^n \to \mathbb{R}^I$ является базовым, функция f сделает отображение входных данных в выходные данные, цель которых состоит в том, чтобы аппроксимировать f оптимальным способом, зная только экземпляры $f(x^i) = y^i$. Чтобы решить, какое отображение является оптимальным, функция потерь L определяется аналогично J в (2). Функция потерь зависит от отображения f и принимает значения, близкие к нулю, если f является близким приближением истинного базового отображения между входными и выходными данными. Поэтому в задачах машинного обучения решается следующая задача минимизации:

$$y^i J(f) = L(f) + \alpha R(f) = \sum_{i=1}^{N} L\left(f(x^i), y^i\right) + \alpha R(f) \to \min_f, \qquad (9)$$

где $R(f)$ – регуляризационный функционал, который включает априорную информацию о f и α-весах.

Несмотря на то, что эта задача минимизации очень похожа на задачу минимизации в предыдущем подразделе, фундаментальное отличие заключается в том, что отображение K в (2) известно, и мы ищем оптимальное u, но для задачи машинного обучения (9), входные данные x известны, но мы ищем оптимальное отображение f между входными и выходными данными. Наиболее распространенной задачей в контролируемом обучении является классификация, где мы пытаемся сопоставить x^i с дискретной меткой y^i. Для анализа ячеек это может быть сопоставление входных объектов x^i с классом ячеек. Примерами методов машинного обучения, используемых для классификации, являются методы опорных векторов [5], деревья решений [17] и нейронные сети, которые описаны ниже.

Для неконтролируемого машинного обучения у нас есть только входные данные x^i, но нет необходимых выходных данных $S = \{x^i \in$

$\mathbb{R}^n, i, \ldots, N\}$. В этом случае цель состоит в том, чтобы найти скрытые закономерности во входных данных, которые описывают структуру различных входных данных. Одним из классов неконтролируемых методов обучения являются метод кластеризации, цель которого состоит в том, чтобы найти группы объектов, которые имеют общие характеристики, отличающие их от других примеров. Хорошо известными примерами алгоритмов кластеризации являются, например, алгоритм K-средних [14], метод ближайших соседей [18] и спектральная кластеризация [15]. Другая группа неконтролируемых алгоритмов обучения состоит из методов уменьшения размерности, цель которых состоит в том, чтобы найти представление входных данных в пространстве с более низкой размерностью. Яркими примерами данной группы являются методы главных компонент [10], сингулярная декомпозиция [8] и нейронные сети.

2.1.2.1 Предварительная обработка изображений в качестве входных данных

Для применения машинного обучения на данных изображений мы можем выделить два различных класса входных данных. Традиционно, алгоритмы обучения (как контролируемые, так и неконтролируемые) работают не непосредственно с самим изображением, а с функциями, которые были извлечены ранее. Это означает, что мы преобразуем исходное изображение $x \in \mathbb{R}^n$ в пространство объектов $\mathbb{R}^m (m < n)$ с помощью функции извлечения объектов $F: \mathbb{R}^n \to \mathbb{R}^m$. Алгоритм обучения впоследствии использует вектор признаков *F(x)* в качестве входных данных вместо самого изображения *x*. В биомедицинских приложениях пространство функций часто состоит из созданных вручную, интуитивно понятных функций, таких как края или физические свойства, а именно округлость, размер или интенсивность. Таким образом, мы определяем, какие, по нашему мнению, наиболее информативные функции необходимы алгоритму для выполнения учебной задачи. Недостатком является то, что мы уже смещаем

алгоритм, используя функции ручной работы. Скрытая функция, которая напрямую не связана с физическими свойствами или свойствами изображения, может быть пропущена таким образом. Поэтому последние достижения в области нейронных сетей используют исходное изображение *x* в качестве входных данных. Методы состоят из двух частей – части извлечения объектов, которая также извлекается из входных данных, и функции, специфичной для конкретной задачи. Это означает, что алгоритм сам узнает, какие функции наиболее полезны для извлечения для желаемой задачи, и поэтому не подвержен влиянию человеческой интерпретируемости входных функций.

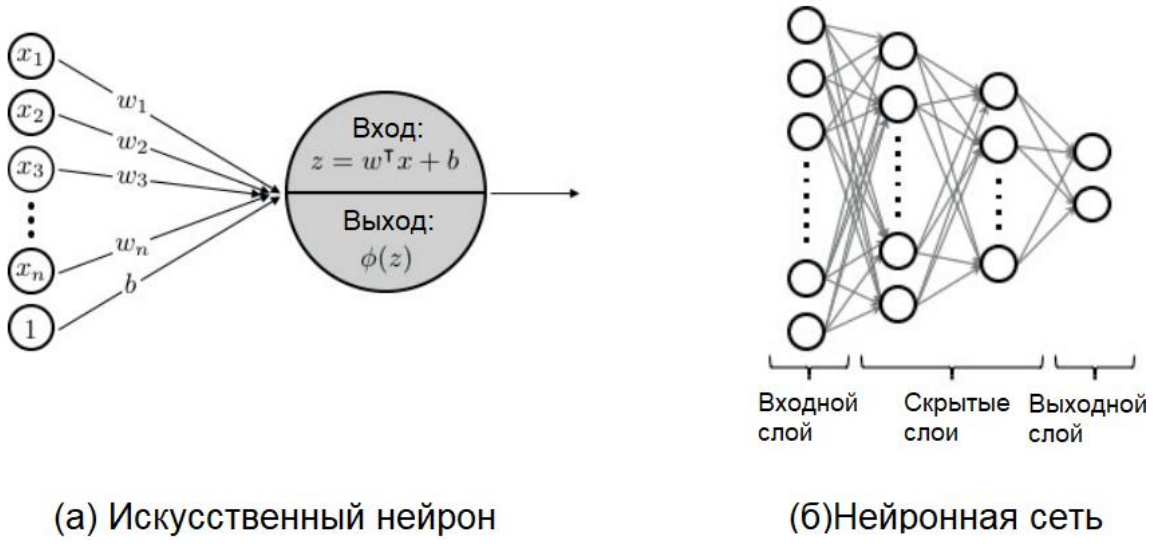

Рисунок 2 – Эскиз искусственного нейрона (а) и пример нейронной сети с двумя скрытыми слоями (б)

2.1.2.2 Нейронные сети и глубокое обучение

Нейронные сети являются графовыми структурами, в которых каждый узел представляет собой искусственный нейрон, вдохновленный нейронами в мозге. Искусственный нейрон с входным сигналом $x \in \mathbb{R}^n$ определяется как:

$$\eta\ (x;\ \varphi,\ w,\ b) = \varphi\ (w^{\mathrm{T}}x + b), \tag{10}$$

где *w* – вектор веса;

b – смещение;

φ – функция активации.

Таким образом, нейрон получает входы $x_1, \ldots, x_1$, присваивает каждому из них вес w_i и суммирует взвешенные входы. Кроме того, к сумме $w^{\mathrm{T}}x$ добавляется смещение b. Наконец, нейрон выводит (нелинейную) функцию активации φ, применяемую к так называемым баллам $w^{\mathrm{T}}x + b$. Схема искусственного нейрона представлена на рисунке 2(а). Комбинация нескольких нейронов в ациклическом ориентированном графе называется нейронной сетью, которая представлена на рисунке 2(б). Первый слой нейронов называется входным слоем, последний слой – выходным слоем, а все промежуточные слои называются скрытыми слоями. Сеть, в которой нет или очень мало скрытых слоев, называется мелкой сетью, в то время как сеть с несколькими скрытыми слоями называется глубокой. Нейронные сети — это класс отображений f от входных данных к выходным. При решении задачи минимизации (9) все веса и смещения оптимизируются по отношению к входным данным и желаемому выходу.

Как упоминалось ранее, минимизация функционала J(f) требует производной по параметрам для оптимизации $\frac{\partial J}{\partial f}$. Поскольку f сама по себе является композицией сложенных слоев, для определения производной можно использовать правило цепочки. Градиенты по параметрам в слое (k – 1) могут быть выражены в терминах градиентов по параметрам в следующем слое k. Поэтому, чтобы вычислить производную, мы начинаем вычислять производную функции потерь по параметрам последнего слоя, а затем распространяемся назад по всем слоям. Этот алгоритм называется алгоритмом обратного распространения. Это можно рассматривать как противоположное отображение прямого пути через сеть, который сопоставляет x с выходом y. Таким образом, алгоритм обратного распространения позволяет эффективно обучать и более сложные сети, если функции активации дифференцируемы, чтобы можно было вычислить все частные производные.

Ключевым элементом гибкости нейронных сетей являются нелинейные функции активации φ. Они позволяют аппроксимировать произвольные функции с любой заданной точностью. В реальных задачах это имеет решающее значение, поскольку большинство базовых функций, отображающих входные данные на желаемый выход, требуют расширенных нелинейностей.

Как показано на примере рисунка 2(б), нейронные сети представляют собой сложенные слои искусственных нейронов. Нейроны в каждом слое (кроме входных и выходных слоев) соединены с каждым нейроном в слое до и после. Эта архитектура называется полностью подключенным слоем. Более формально, для нейронов N_{k-1} в (k – 1) слое и нейронов N_k в слое k мы определяем полностью связанный слой I_k^{full} как функцию $I_k^{full}: \mathbb{R}^{N_{k-1}} \to \mathbb{R}^{N_k}$, которая определяется как:

$$I_k^{full}(\mathrm{x};\ \mathrm{W}, \mathrm{b}) := \varphi(Wx + b), \tag{11}$$

где $\varphi = [\varphi_1, \varphi_2 \ldots, \varphi_{N_k}]$ – точечная функция активации;

$W = \left[w_1, w_2 \ldots, w_{N_k}\right] \in \mathbb{R}^{N_k} \times \mathbb{R}^{N_{k-1}}$ – весовая матрица, хранящая веса для каждого нейрона в I_k^{full};

$b \in \mathbb{R}^{N_k}$ – вектор смещения.

Для простоты мы суммируем все параметры сети, оптимизированные на этапе обучения (т. Е. все весовые матрицы W и все смещения b), в одной переменной θ.

Многоуровневые сетевые сопоставления можно разделить на две различные функции: функцию извлечения объектов и функцию, специфичную для конкретной задачи. Первая часть, определенная как F, отвечает за извлечение оптимальных объектов из входных данных для желаемой задачи и, таким образом, заменяет ручное извлечение объектов, описанное выше. Извлеченные объекты затем вводятся во вторую, специфичную для конкретной задачи функцию $T : \mathbb{R}^m \to \mathbb{R}^I$. В случае

контролируемой задачи классификации это нелинейный классификатор, в котором количество нейронов в последнем слое соответствует количеству классов. Таким образом, функция $f : \mathbb{R}^n \rightarrow \mathbb{R}^I$, отображающая вход x на желаемый выход y, может быть описана как:

$$f(x;\ \theta)\ =\ T(F(x;\theta_F);\ \theta_T). \qquad (12)$$

Функции F и T параметризуются с помощью θ_F и θ_T, что означает, что веса и смещения всех слоев, используемых в F или T соответственно, суммируются в соответствующих θ.

После построения глубокой сети, задача состоит в том, чтобы найти оптимальные веса с учетом входных и выходных данных. Это требует решения задачи минимизации (9) с f, определенной как в (12). С ростом сложности и глубины сетевых архитектур процесс оптимизации стал настолько сложным, что эффективное обучение глубоких нейронных сетей стало самостоятельной областью исследований, называемой глубоким обучением [13]. Хотя технически алгоритм обратного распространения может быть использован для обучения каждой глубокой сети с дифференцируемыми функциями активации, может возникнуть несколько практических проблем. Основным недостатком являются исчезающие градиенты, которые препятствуют эффективному обучению ранних слоев. Алгоритм обратного распространения основан на цепном правиле, в котором умножается несколько производных. Если все производные меньше единицы, этот продукт становится очень маленьким для ранних слоев в глубоких сетях, так что веса этих слоев обновляются лишь незначительно. В качестве альтернативы, если все производные больше единицы, произведение всех частных производных становится очень большим, что приводит к серьезным изменениям в ранних слоях и нестабильному обучению. Методы предотвращения обоих случаев включают новые сетевые архитектуры, называемые остаточными сетями [9], адаптированные функции активации [7] или регуляризации.

Выбор функции потерь L, используемой для оптимизации параметров нейронной сети, тесно связан с решаемой задачей. Как и в (2), общим выбором является L^2-расстояние:

$$L^2\left(f\left(x^i;\theta\right),y^i\right)=\left|\left|f\left(x^i;\theta\right)-y^i\right|\right|_{L^2}. \quad (13)$$

2.1.2.3 Сверточные нейронные сети

В области обработки изображений, где $x \in \mathbb{R}^{n_1\times\ldots\times n_d}$, использование полностью подключенных сетей, представленных в предыдущем параграфе, часто затруднено огромным количеством необходимых параметров. Предположим, что у нас есть изображение в оттенках серого с размером 100 × 100 = 10 000 пикселей, неглубокая сеть всего с 2 скрытыми слоями по 200 нейронов в каждом и выходной слой с 2 узлами. Это приводит к 10000 × 200 + 200 × 200 + 200 × 2 = 2040400 веса, плюс одно смещение на узел для оптимизации. Если изображение представляет собой цветное изображение RGB вместо оттенков серого, количество весов напрямую увеличивается до 6040400. Это показывает, что для достаточно больших изображений более глубокие сети вычислительно неосуществимы, и для оптимизации всех параметров требуется огромное количество обучающих данных.

Один из способов обойти эту проблему – использовать повторяющуюся структуру изображений и использовать так называемую сверточную нейронную сеть. Вместо того, чтобы соединять каждый пиксель в одном слое с каждым пикселем в следующем слое, пиксель соединяется только с пикселями, окружающими его, с помощью свертки (рисунок 3). Таким образом, количество подключенных пикселей определяется размером ядра s. Вместо того, чтобы изучать все веса полностью связанного слоя, изучаются только веса сверточных ядер. Затем, то же самое ядро идет по изображению, и результирующая свертка вместе с смещением вводятся в функцию активации. Сверточный слой соответствует полностью связному

слою с разреженной весовой матрицей, где только диагонали $(s^2 - 1)$ ненулевые. Более того, для каждой ненулевой диагонали все записи имеют одинаковое значение, отражающие одни и те же сверточные шаги по всему изображению. Тем не менее, в отличие от полностью связанного слоя, в каждом слое изучается не только одно ядро, то есть, не только разреженная матрица веса, но и фиксированное количество N_k-ядер. Это число обычно увеличивается с увеличением глубины сети. Каждое ядро приводит к новому представлению изображения, и в последующем слое эти представления объединяются в третьем измерении. Формально сверточный слой может быть записан как функция:

$$I_k^{\text{сверт}}(x; K, b) := [\varphi(z_1), \ldots, \varphi(z_{N_k})]. \quad (14)$$

В данной функции, каждая карта объектов $z_j (j = 1, \ldots, N_k)$ определяется как:

$$z_j(x; K_j) = K_j \times x + b_j, \quad (15)$$

где $*$ – дискретная свертка x с ядром K_j и смещением b_j.

Простая сверточная нейронная сеть показана на рисунке 4.

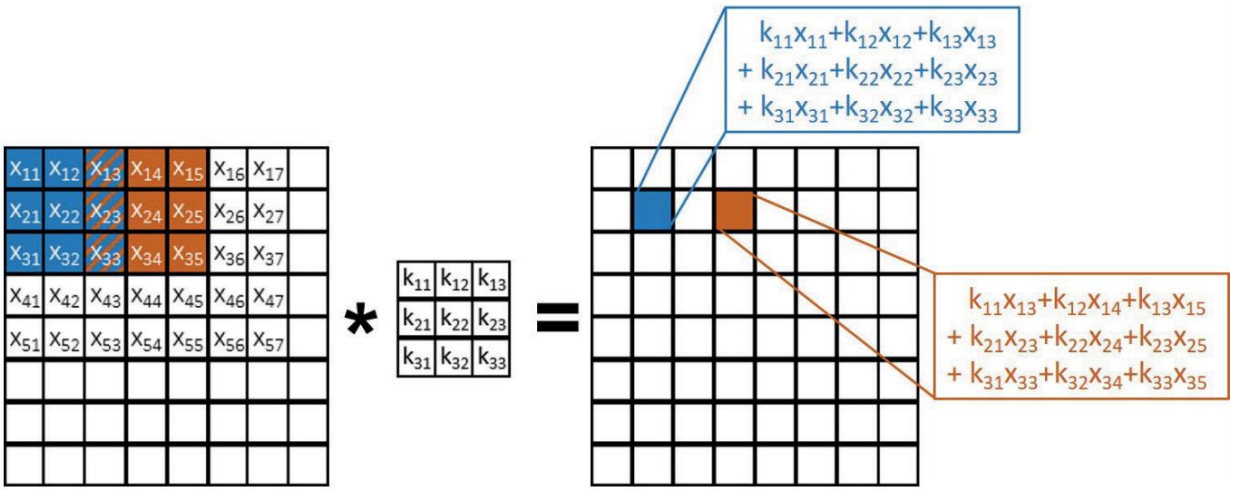

Рисунок 3 – Визуализация свертки в нейронной сети. Каждый пиксель соединен со всеми окружающими пикселями, покрытыми ядром свертки

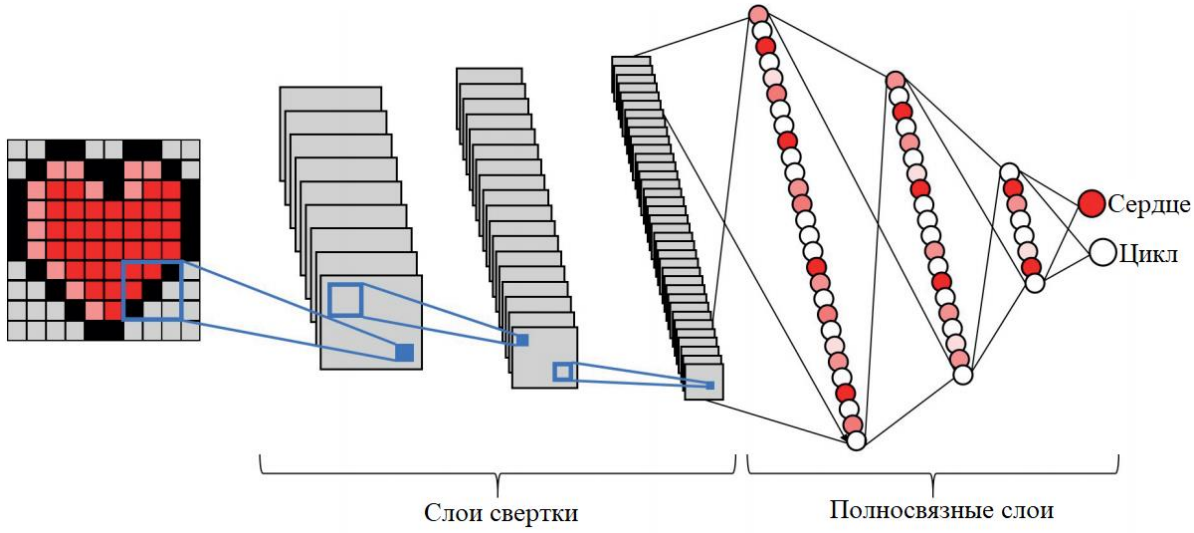

Рисунок 4 – Визуализация сверточной нейронной сети для бинарной классификации

Один из способов понять, как сверточные нейронные сети смотрят на входные данные — это визуализировать изученные функции. На рисунке 5 показан пример визуализации объектов. Здесь происходит визуализация функции, обученной сверточной нейронной сетью, на наборе данных ImageNet. Для фиксированной карты объектов они показали девять наиболее заметных активаций (слева), а справа – изображения, которые активировали карту объектов. Мы видим, что для каждой карты объектов девять изображений очень похожи, показывая, например, весь полосатый узор (слой два, строка один, столбец два) или все колесо автомобиля (слой три, строка два, столбец два). Это отражает то, что обученные ядра в сверточной нейронной сети выделяют специфические особенности изображения. Более того, мы видим, что масштаб объектов увеличивается с глубиной слоя. В первом слое изучаются только маленькие объекты, такие как ребра, в то время как в третьем слое карта объектов уже выделяет гораздо более сложные узоры (первая строка, первый столбец) или верхнюю часть тела и голову человека (третья строка, третий столбец). Это показывает свойство извлечения иерархических объектов сверточной сети. Интуитивное представление о том, почему более глубокие слои изучают более

масштабные объекты входного изображения, можно увидеть на рисунке 6. Мы выбираем один пиксель в слое k (выделен синим цветом) и размер ядра 3 × 3. Если мы затем посмотрим в предыдущем слое (k – 1), какие пиксели повлияли на активацию выбранного пикселя в слое k, мы увидим, что это окно 3 × 3 вокруг исходного пикселя. Если мы затем посмотрим на один слой раньше, то есть, в слой (k – 2), это уже окно размером 5×5 пикселей, которое влияет на активацию одного пикселя в слое k. Поле пикселей, влияющих на активацию (обозначено для каждого слоя синим цветом на рисунке 6), называется восприимчивым полем. Мы видим, что он увеличивается с каждым слоем, поэтому чем глубже слой, тем больше восприимчивое поле во входном изображении. Таким образом, более глубокие слои изучают крупномасштабные объекты входного изображения, а ранние слои имеют очень локализованное представление о входном изображении и, следовательно, изучают объекты малого масштаба.

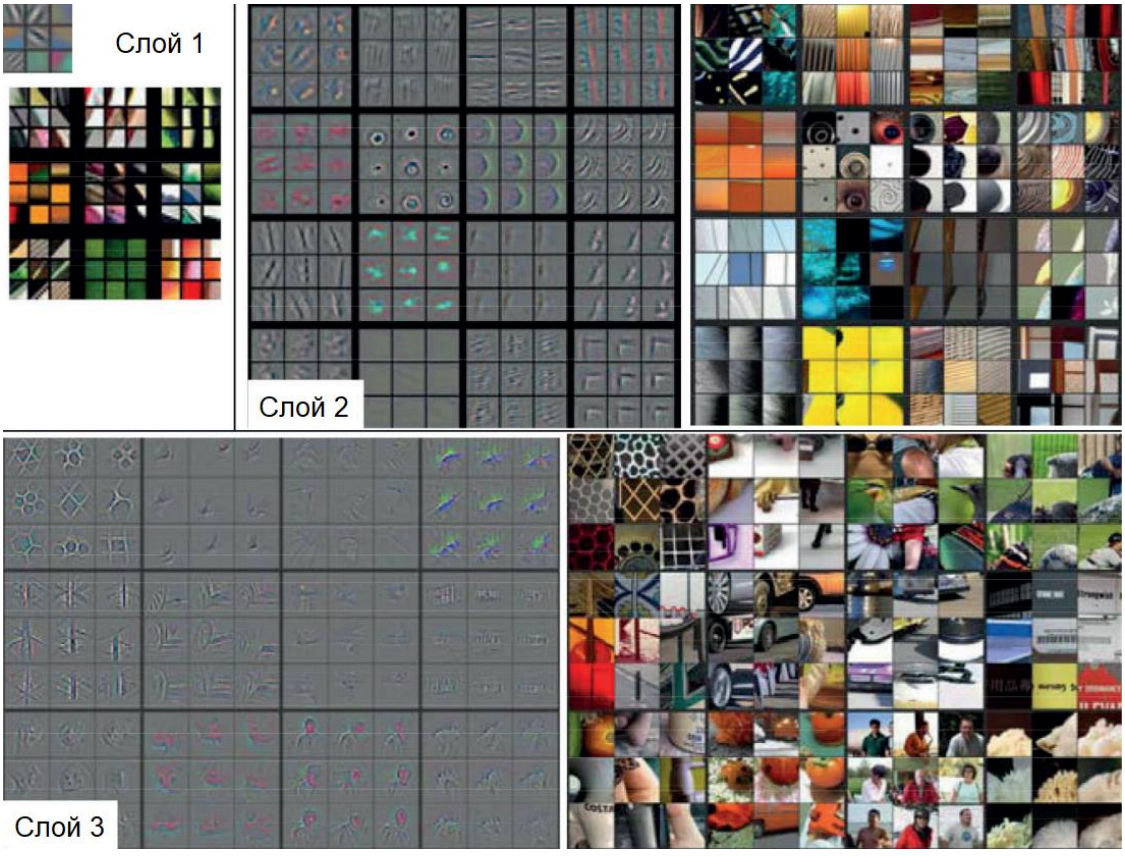

Рисунок 5 – Визуализация функций в полностью обученной сети

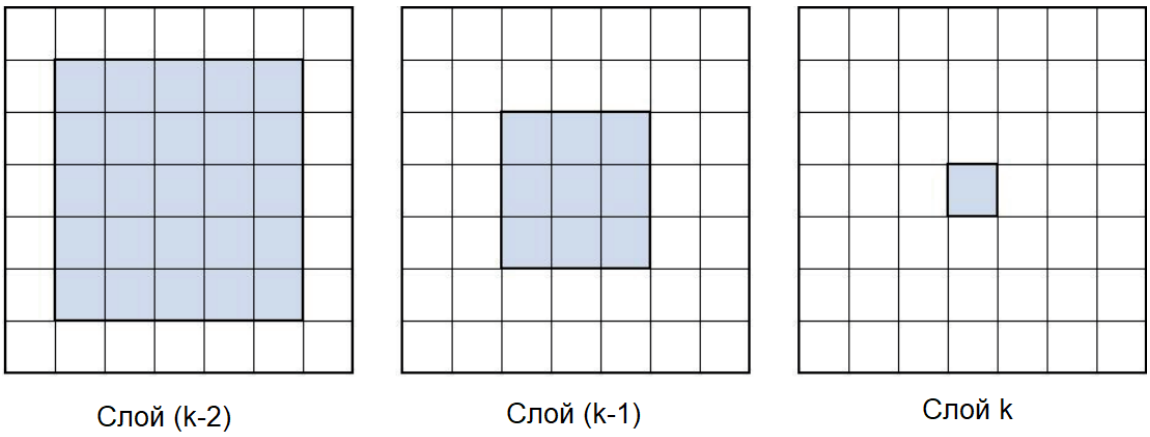

Рисунок 6 – Визуализация восприимчивого поля трех последующих слоев

2.1.2.4 Автокодировщики

В случае неконтролируемого обучения, выходные метки $y^i \in \mathbb{R}^I$ недоступны. Тем не менее, чтобы найти структуру в данных, мы можем сопоставить наши входные данные x^i с самими x^i, но мы ограничиваем функцию отображения $f(x;\theta)$, чтобы она изучала больше, чем простое сопоставление идентификаторов. Такая архитектура нейронной сети называется автокодировщиком [2]. Поскольку автокодировщики отображают $f : \mathbb{R}^n \to \mathbb{R}^n$ (или в случае изображений $f : \mathbb{R}^{n_1 \times \ldots \times n_d} \to \mathbb{R}^{n_1 \times \ldots \times n_d}$, они имеют существенно иную структуру, чем классификационные сети, которые отображают $x^i \in \mathbb{R}^n$ в $y^i \in \mathbb{R}^I$ с $I << n$. Сравнение обеих архитектур в случае полностью подключенной сети показано на рисунке 7. Для автоавтокодировщиков, специфичная для задачи функция $T : \mathbb{R}^m \to \mathbb{R}^n$ представляет из себя функцию реконструкции, которая отображает объекты $F(x;\theta_F)$ обратно во входное пространство. Для того чтобы сжать входные данные и изучить наиболее важные функции, необходимые для восстановления входных данных как можно лучше, пространство объектов $\mathbb{R}^m$ в большинстве случаев должно иметь меньшую размерность, то есть, $m < n$. Тем не менее, подобно переполненным словарям при изучении словарей, существуют также подходы, в которых выбирается пространство объектов более высокой размерности. Мы называем пространство признаков скрытым

пространством и называем $F(x;\theta_F) \in \mathbb{R}^m$ скрытыми переменными. Хотя архитектуры кодирования и декодирования для автокодировщиков могут быть выбраны независимо, обычно используются симметричные автокодировщики.

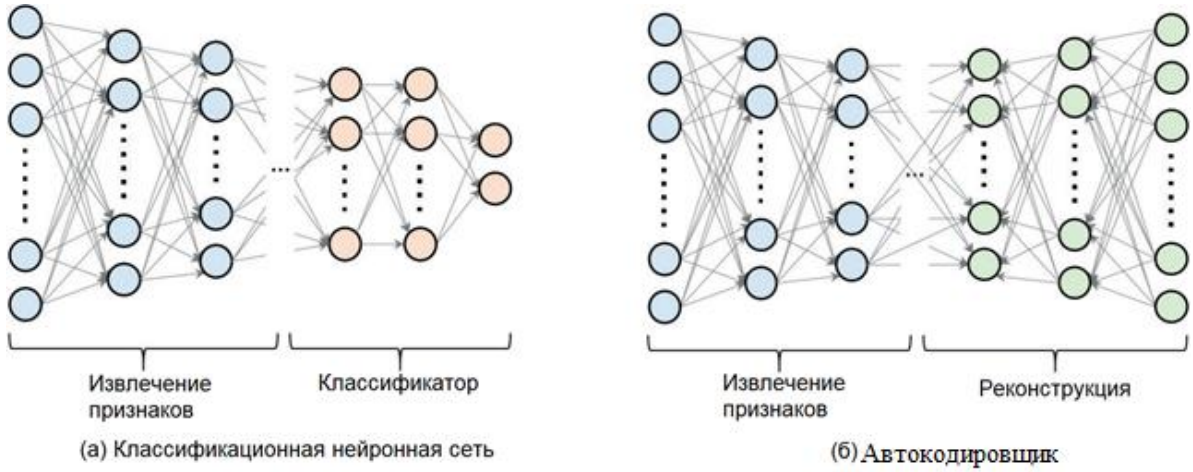

Рисунок 7 – Пример нейронной сети для классификации (а) и автокодировщика (б) с одинаковой архитектурой кодирования

2.1.2.5 Связь нейронных сетей с фильтрационными и вариационными методами

В предыдущих разделах мы уже видели, что вариационные методы и методы глубокого обучения имеют проблемы с минимизацией энергопотребления. Нелинейное уравнение диффузии (7), возникающее в результате минимизации полной вариации (6), может быть интерпретировано как прямой путь через сверточный слой с фиксированными весами и фиксированной функцией активации. Это наталкивает на вопрос о том, действительно ли регуляризация полной вариации является наилучшей регуляризацией для данных или другие веса и функции активации улучшат результаты? Этот вопрос решается классом архитектур нейронных сетей, называемых вариационными сетями [11]. Идея состоит в том, чтобы заменить фиксированный функционал регуляризации вариационного метода (2) обучаемой сверточной нейронной сетью. Таким образом, вариационные сети объединяют идею обоих полей и приводят к двухуровневой задаче

оптимизации, в которой мы минимизируем как решение u, так и параметры сети θ. Можно показать, что существует энергетический функционал E, для которого один шаг в градиентной схеме минимизации E соответствует прямому пути через автокодировщик с привязанными весами (автокодировщик, в котором матрица веса декодирования является транспонированием матрицы веса кодирования). Это снова показывает тесную связь между оптимальными функционалами регуляризации и нейронными сетями.

2.2 Реализация программного обеспечения

Нижеописанные алгоритмы были написаны при помощи языка MATLAB.

На основании изученных методов, был реализован алгоритм сегментации. Он основан на сходстве функционала полной вариации (5) и функционала Чана-Веза (4). Однако, в нашем случае мы заменяем регуляризацию полной вариации на итерационную регуляризацию, которая основана на расстояниях Брегмана. Таким образом, полученная новая модель сегментации задается как:

$$u_{k+1} = \underset{\substack{\mathrm{u}\in \mathrm{BV}(\Omega) \\ \mathrm{u}(\mathrm{x})\in[0,1]}}{argmin} \int_{\Omega} \mathrm{u}((f-c_1)^2 - (f-c_2)^2) + \alpha D_{TV}^{Pk}(\mathrm{u}, u_k). \qquad (16)$$

Вставив определение расстояния Брегмана $D_{TV}^{Pk}(\mathrm{u}, u_k) \coloneqq TV(u) - TV(u_k) - \langle p_k, u - u_k \rangle$ и игнорируя части независимые от u, мы получаем следующую модель:

$$u_{k+1} = \underset{\mathrm{u}\ \in \mathrm{BV}(\Omega)}{argmin} \int_{\Omega} \mathrm{u}((f-c_1)^2 - (f-c_2)^2) + \\ + \chi_{[0,1]}(u) + \alpha(TV(u) - \langle u, p_k \rangle). \qquad (17)$$

Диапазон шкал, присутствующих в u_{k+1}, снова определяется индексом итерации k. Эта модель представляет собой подход с обратным масштабным пространством, поэтому малое k соответствует крупномасштабной сегментации и наоборот.

По определению расстояния Брегмана в $p_k \in \partial \mathrm{TV}(u_k)$ субдифференциал многозначен. Поэтому нам нужно определить правило для выбора p_k. Один из способов - вывести стратегию обновления, основанную на условии оптимальности (17) [16]:

$$0 = ((f - c_1)^2 - (f - c_2)^2) + q_{k+1} + \alpha\, p_{k+1} - \alpha\, p_k, \qquad (18)$$

где q_{k+1} – элемент в субдифференциале характеристической функции $\chi_{[0,1]}(u_{k+1})$.

Таким образом, мы можем выбрать $q_k = 0$ и с этого момента пренебречь им. В следствии чего, стратегия обновления для p_{k+1} задается как:

$$p_{k+1} = p_k - \frac{1}{\alpha}((f - c_1)^2 - (f - c_2)^2) = -\frac{k+1}{\alpha}((f - c_1)^2 - (f - c_2)^2). \qquad (19)$$

Это обновление не зависит от u_k, поэтому оно является поточечным постоянным обновлением в каждой итерации.

После того как все методы были описаны, пришло время их реализовать в виде программного алгоритма.

На рисунке 8 представлена инициализация метода Брегмана (19), используя функционал MATLAB.

```
bregman_cv_core.m
function u = bregman_cv_core(f,nx,ny,lambda,breg_it,inner_it,...
                             tol,p,u,u_bar,b,sigma,tau,theta,...
                             init,mu_update,mu0,mu1,useMask,mask)
    dim = ndims(f);

    i = 1; j = 1;
    while i <= breg_it %инициализация итераций Брегмана
        stat_u = [];
        while j <= inner_it && (isempty(stat_u) || ~isempty(stat_u) && stat_u(end) >= tol) %регуляризация полной вариации
            arg1 = p + sigma * grad(u_bar,'shift'); %обновление p по (19)
            p = arg1 ./ max(1,repmat(sqrt(sum(arg1.^2,3)),[1 1 dim]));
            u_old = u; %обновление u по (17)
            arg2 =  (u + tau * div(p,'shift')) - tau/lambda * ((f - mu1).^2 - (f - mu0).^2 - lambda * b);
            u = max(0, min(1,arg2));
            stat_u(j) = (nx*ny)^(-1) * (sum((u(:) - u_old(:)).^2)/sum(u_old(:).^2));
            theta = 1/((1+2*tau)^(1/2)); tau = theta * tau; sigma = sigma/theta;
            u_bar = u + theta * (u - u_old);
            j = j + 1; %обновление внутреннего индекса

        end

        b = b + 1/lambda * ((f - mu0).^2 - (f - mu1).^2);  %обновление границ для метода Брегмана
        i = i + 1; j = 1; %обновление внешнего индекса

    end
```

Рисунок 8 – Инициализация метода Брегмана

Для нахождения значения *u*, требуется вычислить значение TV-потока (7), для этого, реализуем формулу в виде программного алгоритма (рисунок 9).

```
% реализация формулы (7), с применением функционала Matlab
function div_v = div(v,method)
    persistent Dx_div Dy_div
    N = size(v,2);
    M = size(v,1);
    if size(Dx_div,1) ~= N || size(Dy_div,1) ~= M
        Dx_div = [];
        Dy_div = [];
    end
    if isempty(Dx_div) && strcmp(method,'lr')
        Dx_div = spdiags([-ones(N,1) ones(N,1)],[0 1],N,N); Dx_div(N,:) = 0;
        Dy_div = spdiags([-ones(M,1) ones(M,1)],[0 1],M,M); Dy_div(M,:) = 0;
    end
    if strcmp(method,'shift')
        v = single(v);
        [nx, ny, ~] = size(v);
        hx = 1; hy = 1;
        div_v = hx^(-1) * cat(1, v(1,:,1), v(2:nx-1,:,1) - v(1:nx-2,:,1), -v(nx-1,:,1));
        div_v = div_v + hy^(-1) * cat(2, v(:,1,2), v(:,2:ny-1,2) - v(:,1:ny-2,2), -v(:,ny-1,2));
    elseif strcmp(method,'lr')
        div_v = v(:,:,1)*Dx_div + Dy_div'*v(:,:,2);
    end
```

Рисунок 9 – Реализация формулы (7) в виде программного алгоритма

Следующую переменную, которую требуется найти, является значение TV-регуляризации. Нахождение данной переменной при помощи программного алгоритма можно наблюдать на рисунке 10.

```
% реализация формулы (5), с применением функционала Matlab
function grad_u = grad(u,method)
    persistent Dx_grad Dy_grad

    N = size(u,2);
    M = size(u,1);
    if size(Dx_grad,1) ~= N || size(Dy_grad,1) ~= M
        Dx_grad = [];
        Dy_grad = [];
    end
    if isempty(Dx_grad) && strcmp(method,'lr')
        Dx_grad = spdiags([-ones(N,1) ones(N,1)],[0 1],N,N); Dx_grad(N,:) = 0;
        Dy_grad = spdiags([-ones(M,1) ones(M,1)],[0 1],M,M); Dy_grad(M,:) = 0;
    end
    if strcmp(method,'shift')
        u = single(u);
        [nx, ny] = size(u);
        hx = 1; hy = 1;
        grad_u(:,:,1) = hx^(-1) * cat(1, u(2:nx,:) - u(1:nx-1,:), zeros(1,ny));
        grad_u(:,:,2) = hy^(-1) * cat(2, u(:,2:ny) - u(:,1:ny-1), zeros(nx,1));
    elseif strcmp(method,'lr')
        if issparse(u*Dx_grad') || issparse(Dy_grad*u)
            grad_u = cat(3,full(u*Dx_grad'),full(Dy_grad*u));
        else
            grad_u = cat(3,u*Dx_grad',Dy_grad*u);
        end
    end
end
```

Рисунок 10 – Нахождение значения TV-регуляризации

Результат работы вышеописанного алгоритма можно наблюдать на рисунке 11.

Рисунок 11 – Пример работы алгоритма

Реализовав алгоритм сегментации, следующий шагом, является применение методов машинного обучения к результатам работы данного алгоритма. В прошлом разделе мы в вкратце расписали алгоритм сверточной нейронной сети, в данном разделе предстоит его реализация. Для реализации сверточной нейронной сети я использовал нейросетевую библиотеку Keras на

языке Python, с применением архитектуры U-net с пропускными соединениями.

Несмотря на то, что в среде разработки MATLAB нет возможности использовать язык программирования Python, в данной среде имеется возможность импортировать предварительные подготовленные сети Keras при помощи команды “net = importKerasNetwork(modelfile)”.

Для автоматической классификации флуоресцентных миниатюр клеток по отдельным классам клеток мы использовали сверточные нейронные сети [12]. Мы использовали две различные архитектуры. Первая архитектура представляет собой стандартную сверточную нейронную сеть с тремя слоями свертки с размером ядра 3 × 3 и увеличивающимся числом фильтров. Первый сверточный слой имеет 16 фильтров, два следующих: 32 и 64 фильтра. Количество фильтров определяет количество каналов в выводе сверточного слоя. За каждым сверточным слоем следует слой максимального объединения, чтобы вдвое уменьшить разрешение входных данных. Таким образом, входное изображение преобразуется из изображения 80×80×3 в представление входных данных 10×10×64 путем прямой передачи его через три блока свертки и слоя максимального объединения.

За сверточной частью следует полностью связанный слой из 208 нейронов, которые соединены другим полностью связанным слоем с 50 нейронами, образующими скрытое пространство. В этом слое функция активации не применяется. Это 50-мерное представление скрытого пространства вводится в последние два полностью связанных выходных слоя с шестью нейронами каждый, где последний имеет функцию активации в виде сигмоиды. Чтобы обучить эту сеть, мы минимизировали функцию потерь перекрестной энтропии:

$$L_{\text{снс}}(y_{\text{ист}}, y_{\text{прог}}) = -\sum_i y_{\text{ист}}(i)\ln(y_{\text{прог}}(i)), \quad (20)$$

где $y_{\text{ист}}$ – однократно закодированная истинная меткой для входного сигнала $x_{\text{вход}}$;

$y_{\text{прог}}$ – вектор прогнозируемых вероятностей классов.

Для второй сетевой архитектуры мы использовали сверточные автоэнкодеры для улучшения результатов [6]. Автоэнкодеры сначала кодируют входные данные, а затем как можно лучше восстанавливают входной сигнал из закодированного представления. В нашем случае скрытое пространство, содержащее закодированные входные данные, имеет гораздо меньшую размерность, чем входное пространство. Таким образом, сеть изучает представление более низкой размерности, которое все еще содержит большую часть вариаций входных данных, чтобы восстановить входные данные из закодированной скрытой переменной. Для задач классификации автоэнкодеры часто используются для предварительной подготовки сетей, особенно если количество помеченных примеров эталонных данных слишком мало для оптимизации очень глубоких сетей. В этом случае веса сначала оптимизируются на немаркированных данных, а на втором этапе сеть настраивается на изучение классификации на основе помеченных входных данных. В отличие от этого подхода, мы использовали автоэнкодер и классификационную часть одновременно. Архитектура кодирования и классификации такая же, как и для стандартной используемой сверточной нейронной сети, но представление скрытого пространства теперь также используется в качестве входных данных для сверточной сети декодирования. Сеть декодирования начинается с полностью связанного слоя из 6400 нейронов, активация которых преобразуется в тензор размером 10×10×64, который вводится в сверточную часть сети декодирования. Каждая сеть содержит три блока, состоящих из слоя восходящей выборки с коэффициентом два, за которым следует сверточный слой с ядрами размером 3 × 3. Для всех трех сетей первый сверточный слой имеет 32 фильтра, второй - 16 фильтров, а последний слой имеет один фильтр. Выходные слои используют функцию активации сигмоиды, а все другие слои, активация которых до сих пор не определена, используют активацию выпрямленных

линейных единиц. Чтобы обучить сеть классификации с автоматическим кодированием, мы минимизировали следующую функцию потерь:

$$L_{\text{аа}-\text{снс}}\left(\left(x_{\text{вход}}, y_{\text{ист}}\right), \left(x_{\text{рек}}, y_{\text{ист}}\right)\right) =$$
$$= \sum_{c=1}^{3}\left(-\sum_{i,j} x_{\text{вход}}(i,j,c)\ln\left(x_{\text{рек}}(i,j,c)\right)\right) + \beta \cdot L_{\text{снс}}\left(y_{\text{ист}}, y_{\text{прог}}\right), \quad (21)$$

где $x_{\text{рек}}$ – реконструкция входного сигнала на основе сжатого представления в скрытом пространстве.

Во время обучения мы заметили, что алгоритм часто сходился к локальному минимуму весов, что приводило к очень высокой активации в скрытом пространстве, и при реконструкции все значения были близки к нулю. Поэтому мы добавили L_2-регуляризацию на активацию скрытого пространства, которая предотвращает сходимость алгоритма к этому набору весов.

2.3 Обучение алгоритма визуализации в целях реализации программы по распознаванию циркулирующих опухолевых клеток

Чтобы успешно обучить алгоритм, потребуется база с положительной и отрицательной выборкой. В положительной выборке должны присутствовать изображения ЦОК. В отрицательной выборке, должны присутствовать изображения “не-ЦОК”, учитывая характер направления работы, это могут быть другие клетки, которые присутствуют в крови, а именно:

– внеклеточные везикулы;

– белые кровяные тельца;

– голые ядра;

– дифференцировочный антиген.

Для выборки был использован уже готовый датасет, который был найден на сайте портала общих данных о геномных данных (Genomic Data Commons Data Portal, сокращенно GDC). Выборка представляет из себя

25827 оригинальных изображений флуоресцентно подсвеченных клеток в формате .tiff, в которой 6505 изображений представляют из себя изображения клеток с ЦОК, все остальные изображения относятся к изображениям клеток с “не ЦОК”. Выборка была случайным образом разделена на 2 набора: обучающий и проверочный. Доля обучающего набора составляет 80%, доля проверочного набора составляет 20%.

Для обучения сверточной нейронной сети мы использовали итеративную схему обучения: сначала мы обучили сеть с $\beta = 0$, так что обучался только автокодировщик, а веса в слоях классификации оставались случайными. Полученные веса были использованы в качестве инициализации для другой фазы обучения с $\beta = 10$. Затем мы снова использовали эти веса в качестве инициализации и увеличили β до 100. Таким образом, на каждой итерации мы увеличивали влияние производительности классификации на веса. Полное время обучение алгоритма составило 1214 часов.

3 Тестирование и анализ эффективности реализованной программы

3.1 Описание реализованной программы

Результатом выполнения данной работы является реализованное программное обеспечение, которое позволяет визуализировать ЦОК на изображении с образцом.

Основные функции программы:

– работа с изображениями образцов;

– детектирование клеток;

– распознавание “ЦОК” и “не ЦОК” с последующей сегментацией ЦОК при помощи красного контура;

– вывод результатов в виде изображений и статистики;

Для запуска приложения нужно запустить файл Start.m при помощи среды разработки MATLAB.

Запуск программы демонстрируется на рисунке 12, где демонстрируется интерфейс, который выводится при запуске программы.

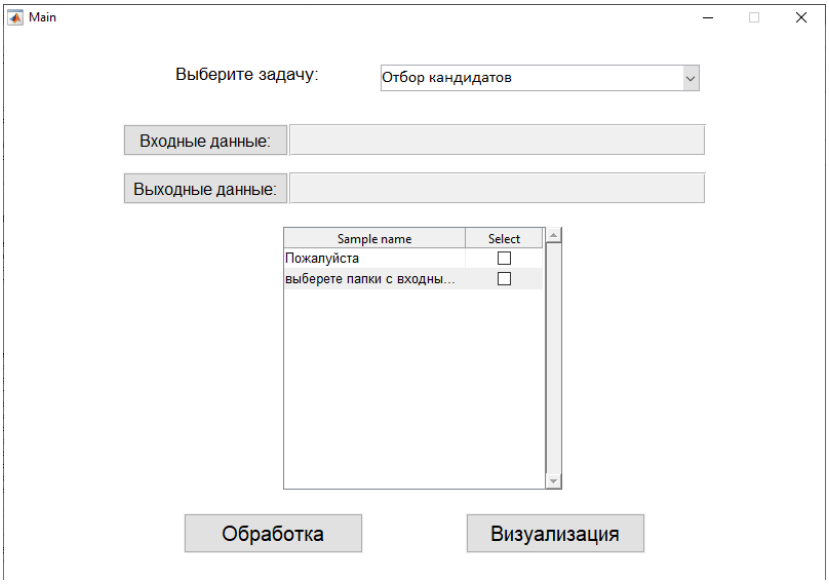

Рисунок 12 – Вывод интерфейса программы при запуске

После этого, нужно выбрать папку с входными и выходными данными. В папке с входными данными должны содержаться изображения с образцами. Изображения, которые анализирует программа, должна содержать автоматическое сканирование всей области (без миниатюр клеток), а также не должна состоять из нескольких сканов. Также, изображения должны быть в формате .tiff. Если все сделано правильно, в поле образцов появятся строки с образцами. Это продемонстрировано на рисунке 13.

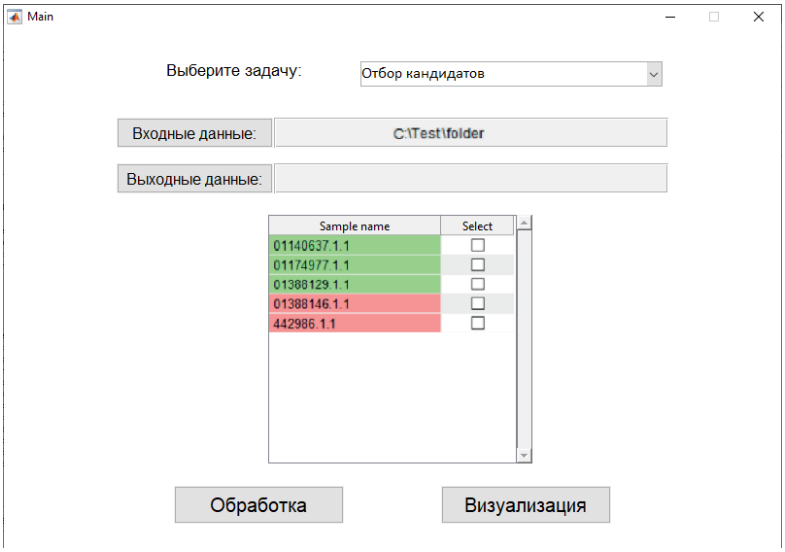

Рисунок 13 – Вывод образцов в поле выбора образцов для обработки

Если образец подсвечивается зеленым, это обозначает что образец уже проходил обработку, если красным, это значит, что образец ещё не проходил процесс визуализации.

После того, как образец был выбран, можно начинать процесс обработки. Процесс обработки осуществляет детектирование клеток, распознавание их и выделение ЦОК, сохраняя полученные результаты в папку выходных данных. Данный процесс занимает 10-40 минут на обработку одного образца.

Для того чтобы увидеть результат работы программы, нужно использовать функцию «Визуализация». Данная функция выводит в интерфейсе программы столбик с обработанными входными данными и

данными после обработки. Результат работы этой функции продемонстрирован на рисунке 14.

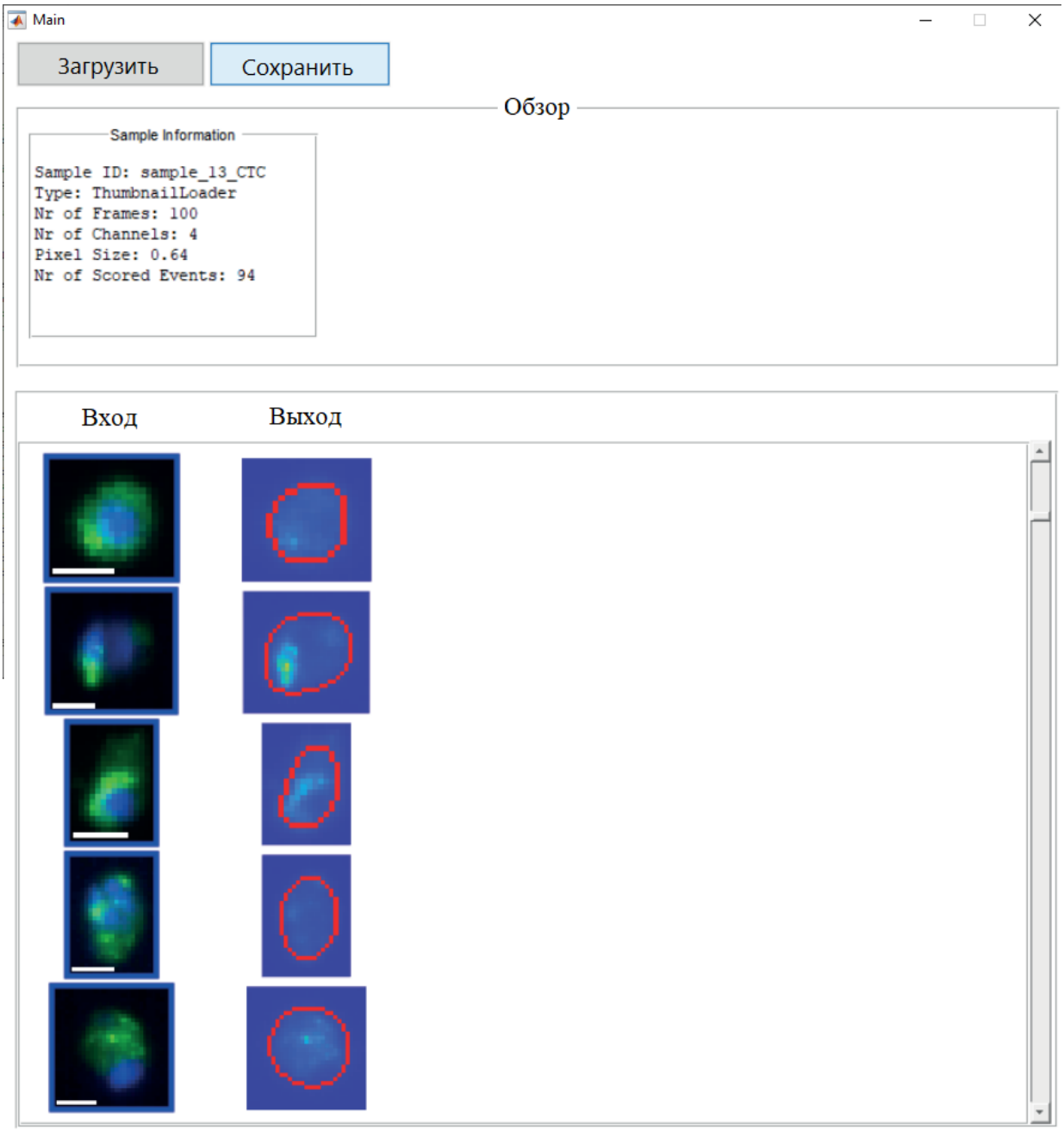

Рисунок 14 – Результат работы функции «Визуализация»

3.2 Тестирование реализованной программы

В целях получения оценки качества работы программы, необходимо учесть несколько параметров. Сначала оценим процесс обучения алгоритма.

Во время обучения алгоритма, были сняты показатели нагрузки на систему, а именно:

– максимальная температура центрального процессора;

– средняя температура центрального процессора;

– максимальная температура ядер процессора;

– средняя температура ядер процессора;

– максимальная нагрузка на ОЗУ.

Также, по моему мнению, к данным показателям следует добавить время обучения алгоритма.

Показатели получились такими:

– максимальная температура центрального процессора: 80 °C;

– средняя температура центрального процессора: 76 °C;

– максимальная температура ядер процессора: 79 °C;

– средняя температура ядер процессора: 76 °C;

– максимальная нагрузка на ОЗУ: 85% (10,1 ГБ / 11,9 ГБ);

– полное время обучения: 1214 часов.

На основе этих данных, можно сделать вывод о том, что алгоритм сверточных нейронных сетей с применением автокодировщика достаточно сильно нагружают систему.

Мы обучили сверточную нейронную сеть для классификации 2-х классов флуоресцентных изображений ЦОК по сравнению с уменьшенными изображениями всех других объектов, найденных на флуоресцентных изображениях. Производительность классификации в нашем наборе валидации из 1301 циркулирующих опухолевых клеток и 8618 других клеток, обобщена на рисунке 15. В целом, мы получили точность в 93,9% с чувствительностью 93,2% и специфичностью 99,0%. Сеть присваивает каждому объекту вероятность принадлежности к классу, и для 1221 правильно классифицированных ЦОК, только 91 клетка имела вероятность ниже 90%, из которых 25 имели вероятность менее 75%. Для 89 объектов, классифицированных как ЦОК, несмотря на то что фактически, выбранные

клетки ЦОК не являются, 67 имели вероятность определения ЦОК выше 75% и 50 из них имели вероятность выше 90%. 22 клетки были неправильно классифицированы сетью и были отмечены как ЦОК.

	Реальность	
Прогноз	1221	22
	89	8529

Рисунок 15 – Матрица ошибок

Чтобы оценить качество работы бинарного классификатора, было решено использовать ROC-кривую, ROC-кривая изображена на рисунке 16. Также был высчитан показатель AUC, данный показатель представляет из себя площадь, которая ограничена ROC-кривой и осью доли ложных положительных классификаций. Чем выше показатель AUC, тем качественнее реализован классификатор. Значение AUC можно увидеть в выделенной жирной ячейке.

ROC-кривая		
False Positive	True Positive	AUC
1	1	0,59091
0,409090909	0,938508839	0,17064
0,227272727	0,049192929	0,00224
0,181818182	0,008455035	0,00038
0,136363636	0,006149116	0,00056
0,045454545	0,004611837	0,00021
0	0	0
		0,76494

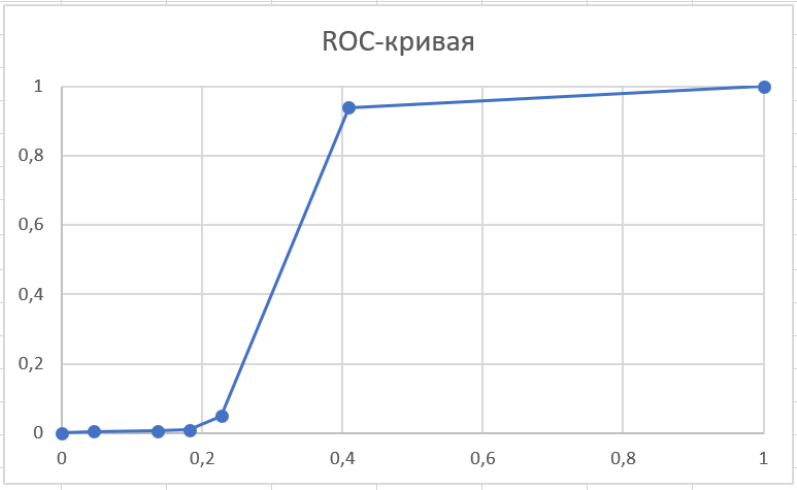

Рисунок 16 - ROC-кривая

Заключение

Рак является одной из ведущих причин смерти во всем мире. Он начинается с образования первичной опухоли и может распространяться по всему организму, что в конечном итоге приводит к большинству смертей, связанных с раком. Опухолевые клетки, которые отделяются от опухоли и проникают в кровоток, называются циркулирующими опухолевыми клетками. Экстравазация этих клеток может привести к возникновению новых опухолей на отдаленных участках, и этот процесс обычно называют образованием метастазов. Количество ЦОК сильно коррелирует со временем выживаемости онкологических больных, и количество ЦОК может быть использовано для мониторинга терапии рака. Тем не менее, данные клетки очень редки, и точное обнаружение, характеристика и подсчет их на изображениях флуоресцентно меченных клеток очень сложен и часто выполняется вручную. Этот процесс подвержен ошибкам и предубеждениям пользователей. В следствии этого, данная выпускная квалификационная работа направлена на разработку программного обеспечения для работы с медицинскими данными, с использованием методов машинного обучения.

В процессе решение вопроса по разработке программного обеспечения, на основе методов машинного обучения для работы с медицинскими данными, в рамках данной работы были достигнуты некоторые результаты, а именно:

1) изучены основные принципы машинного обучения;

2) изучены основные принципы визуализации;

3) реализован алгоритм на основе вышеописанных принципов на языке программирования MATLAB;

4) обучена сверточная нейронная сеть, с применением метода автокодировщика;

5) протестирована реализованная программа;

6) собрана статистика об эффективности работы программы, а также об энерго- и ресурсоемкости программы, в процессе её работы.

Как уже было написано выше, несмотря на достаточную высокую точность полученных результатов, для работы алгоритма затрачивается большое количество вычислительной мощности.

Такой вывод был сделан на основе той информации, которая была получена во время тестирования работы программы.

Данный метод по распознаванию ЦОК в крови пациента, очевидно не является единственным. Существует большое количество известных методов визуализации с применением технологий машинного обучения, и ещё больше, методов, которые до сих пор не являются изученными.

Однако, в рамках данной задачи, где основной целью являлось создать программное обеспечение, для работы с медицинскими данными, данный алгоритм показывает себя достаточно хорошо, поскольку позволяет ценой вычислительных мощностей, точно находить ЦОК в крови пациента, что может привести к более эффективным системам по борьбе с раком.

Таким образом, на основе ранее поставленных задач, можно сделать вывод о том, что цель, заданная в данной выпускной квалификационной работе, была выполнена успешно.

Список используемой литературы

1. Andreu F., Ballester C., Caselles V., Mazón J. M. Minimizing Total Variation Flow // Comptes Rendus de l`Académie des Sciences. 2000. Series 1: Mathematics. С. 867–872.

2. Baldi P. Autoencoders, Unsupervised Learning, and Deep Architectures // ICML Unsupervised and Transfer Learning. 2012. С. 37–49.

3. Bellettini G., Caselles V., Novaga M. The Total Variation Flow in RN // Journal of Differential Equations. 2002. С. 475–525.

4. Chan T. F., Vese L. A. Active Contours Without Edges // IEEE Transactions on Image Processing. 2001. № 10. С. 266–277.

5. Cortes C., Vapnik V. Support-Vector Networks // Machine Learning. 1995. №20. С. 273–297.

6. Ghifary M., Kleijn W. B., Zhang M., Balduzzi D., Li W. Deep Reconstruction-Classification Networks for Unsupervised Domain Adaptation // European Conference on Computer Vision. 2016. С. 597–613.

7. Glorot X., Bordes A., Bengio Y. Deep Sparse Rectifier Neural Networks // International Conference on Artificial Intelligence and Statistics. 2011. С. 315–323.

8. Golub G. H., Reinsch C. Singular Value Decomposition and Least Squares Solutions // Numerische Mathematik. 1970. № 14. С. 403–420.

9. He K., Zhang X., Ren S., Sun J. Deep Residual Learning for Image Recognition // IEEE Conference on Computer Vision and Pattern Recognition. 2016. С. 770–778.

10. Jolliffe I. T. Principal Component Analysis // International Encyclopedia of Statistical Science. 2011. С. 1094–096.

11. Kobler E., Klatzer T., Hammernik K., Pock T. Variational Networks: Connecting Variational Methods and Deep Learning // Pattern Recognition. 2017. С. 281–293.

12. Krizhevsky A., Sutskever I., Hinton G. E. ImageNet Classification with Deep Convolutional Neural Networks // Advances in Neural Information Processing Systems. 2012. С. 1097–1105.

13. LeCun Y. A., Bengio Y., Hinton G. E. Deep Learning // Nature. 2015. № 521. С. 436–437.

14. Macqueen J. Some Methods for Classification and Analysis of Multivariate Observations. California: Berkeley Symposium on Mathematical Statistics and Probability, 1967. С. 281–297.

15. Ng A. Y., Jordan M. I., Weiss Y. On Spectral Clustering: Analysis and an Algorithm // Advances in Neural Information Processing Systems. 2002. № 15. С. 849–856.

16. Osher S. J., Burger M., Goldfarb D., Xu J., Yin W. An Iterative Regularization Method for Total Variation-Based Image Restoration // Multiscale Modeling & Simulation. 2005. № 4. С. 460– 489.

17. Quinlan J. R. Induction of Decision Trees // Machine Learning. 1986. №1. С. 81–106.

18. Roweis S. T., Saul L. K. Nonlinear Dimensionality Reduction by Locally Linear Embedding // Science. 2000. №290. С. 2323–2326.

19. Rudin L. I., Osher S. J., Fatemi E. Nonlinear Total Variation Based Noise Removal Algorithms // Physica D: Nonlinear Phenomena. 1992. С. 259–268.

20. Scherzer O., Grasmair M., Grossauer H., Haltmeier M., Lenzen F., Grasmair M., Haltmeier M. Variational Methods in Imaging. Berlin : Springer Science+ Business Media, 2009. 320 с.