МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ 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 кафедры)

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

Технология программирования
(направленность (профиль)/специализация)

БАКАЛАВРСКАЯ РАБОТА

на тему Разработка системы анализа результатов диагностики кожных патологий для выявления злокачественных новообразований на основе нейронных сетей

Студент К.А. Сержантов
(И.О. Фамилия) (личная подпись)

Руководитель С.В. Баумгертнер
(И.О. Фамилия) (личная подпись)

Консультанты АВ Прошина
(И.О. Фамилия) (личная подпись)

(И.О. Фамилия) (личная подпись)

Допустить к защите

Заведующий кафедрой к.т.н, доцент кафедры ПМИ,
А.В. Очеповский _____
(ученая степень, звание, И.О. Фамилия) (личная подпись)

«_____»_____2018г.

Тольятти 2018

АННОТАЦИЯ

Рассматривается проблема неинвазивной диагностики раковых заболеваний тканей кожи человека. Проводится анализ результатов спектроскопии и автофлуоресценции(АФ) и комбинационного рассеивания(КР), для выявления признаков дифференциации кожных патологий с использованием нейронной сети.

На начальном этапе было проведено исследование возможности обучения нейронной сети на спектрах автофлуоресценции. Основой для исследования стали стохастические методики дифференциации патологий, обосновывающие наличие в спектрах АФ классифицирующих признаков патологий.

Для решения данной задачи была выбрана нейронная сеть прямого распространения, основанная на многослойном персептроне и состоящая из 1 скрытого слоя. Анализируя спектры, мы получили признаки пригодные для обучения многослойного персептрона. При данном подходе существует вероятность упустить скрытые зависимости, в результате чего было принято решение перейти к сверточным нейронным сетям(CNN)

При данном подходе рассматривался спектр комбинационного рассеивания. Входные данные представляется в виде каскада горизонтально расположенных частей спектра, сохраняя зависимость значений интенсивности от длины волны, позволяя нейронной сети самой составляет карты признаков регистрируя флаги, отвечающие за дифференциацию патологии.

Эффективным решением в этой области может стать использование предобученных нейронных сетей: Xception, VGG16/19. Созданные на их базе архитектуры были обучены на подготовленных наборах данных. Для обобщения результатов обучения был разработан алгоритм принятия взвешенного решения.

Работа состоит из введения, трех глав и заключения. Объем работы составляет 65 страниц, на которых размещены 30 рисунка, 5 таблицы и 3

приложения. При написании данной работы было использовано 20 литературных источников, в том числе 7 источников на иностранном языке.

ABSTRACT

The title of the thesis is «Development of a system for analyzing the results of skin diagnosis pathologies for the detection of malignant neoplasms based on neural networks».

The object of the thesis is the process of differentiation of biotissue samples from the results of Raman spectroscopy and autofluorescence(AF). The subject of the thesis is artificial neural networks.

The thesis deals with problems of noninvasive diagnostics of oncological diseases of human skin. The analysis of the results of spectroscopy AF and Raman scattering is carried out to identify signs of differentiation of cutaneous pathologies using a neural network.

We first consider the study of the possibility of training a neural network on the spectra of AF. The basis for the study was stochastic methods of differentiation of pathologies, which justifies the presence in the AF spectral of classifying signs of pathologies. We then examine convolutional neural networks. When working with them, the spectrum of AF was considered in conjunction with the spectrum of Raman scattering. With this approach, the neural network itself creates a feature map by registering the flags responsible for the pathology differentiation. Much attention is given to the processing of input data processing. The use of pre-prepared neural networks limits the input matrix, obliging to use preprocessing, aimed at increasing the dimension of input data. We also report the results of experiments on the detection of skin pathologies by pre-trained neural networks.

An effective solution in this area can be the use of pre-conditioned neural networks: Xception, VGG16 / 19. The architectures created on their basis were trained on the prepared data sets. To generalize the learning outcomes, an algorithm for making a weighted decision was developed.

The graduation work consists of an explanatory note on 65 pages, an introduction, including 30 figures, 5 tables, the list of 20 references including 7 foreign sources and 3 appendices.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Заболеваемость раковыми образованиями, в России составляет в общем 330 случаев на сто тысяч населения в год. Лишь по Самарской области показатель уровня заболеваемости злокачественными новообразованиями – составляет около 446,6 новых случая на 100 000 населения и около 18,6% из них именно рак кожи. Смертельный исход происходит в двухстах случаях из трехсот и за последние несколько десятилетий показатели статистики остаются неизменны [7].

Рак кожи – это обобщенное название большого количества разновидностей злокачественных опухолей. Каждая опухоль имеет свои специфические биологические особенности: клиническое проявление, тканевую структуру, метастазирование и т.д. [1].

Существует три основных типа рака кожи: базальноклеточный рак, плоскоклеточный рак и злокачественная меланома. Среди всех раков кожи злокачественная меланома кожи является наиболее опасным видом рака, так как в подавляющем большинстве случаев приводит к гибели пациентов, особенно при обнаружении патологии на поздней стадии. При этом заболеваемость и смертность от меланомы кожи увеличивается в большинстве стран по всему миру [2]. Принимая во внимание низкую (около 50%) диагностическую точность пигментных опухолей на ранней стадии врачами общей практики, требуется создание новых методов контроля опухолей.

Трудности диагностики меланом врачами общей практики связаны со сложностью в интерпретации клинических признаков опухоли и невозможностью отличить меланому от доброкачественных пигментных образований (таких как невус) на ранней стадии развития. Также при подозрении на наличие меланомы медицинский персонал лишён возможности использования инвазивных методов исследования, таких как биопсия с гистологическим или цитологическим исследованием, в связи с повышенным риском прогрессирования поражений. В этой связи оптические методы имеют

огромный потенциал для неинвазивного выявления и определения конкретного типа опухолевых образований в тканях кожи с применением инструментальных методов.

Наиболее широко развивающимися в этой области являются методы оптической спектроскопии, они позволяют неинвазивно диагностировать раковые опухоли. Сегодня в арсенале ученых существует несколько способов спектроскопического анализа биологических сред. Это: спектроскопия обратного рассеяния, автофлуоресценция, спектроскопия комбинационного рассеяния (КР), отражательная спектроскопия и другие [14].

При выполнении бакалаврской работы использовались высокоуровневые языки программирования MATLAB 9.4 (R2018a), Python 3.6, предоставляющие широкий инструментарий для решения задач машинного обучения по данным результатов спектроскопии с помощью алгоритмов машинного обучения производилось обучение нейронных сетей различных архитектур.

Проект разрабатывается при поддержке Самарского национального исследовательского университета имени академика С.П. Королева, материалы для исследования предоставлены ГБУЗ Самарским областным клиническим онкологическим диспансером.

Цель выпускной квалификационной работы: исследование точности дифференциации патологий по результатам АФ и КР спектроскопии с помощью нейросетевого алгоритма.

Объект исследования – процесс дифференциации образцов биоткани по результатам спектроскопии комбинационного рассеивания и флуоресценции.

Предмет исследования – искусственные нейронные сети.

Выпускная квалификационная работа состоит из введения, трех глав и заключения.

Во введении описывается актуальность рассматриваемой темы, определяются объект и предмет выпускной квалификационной работы, ставится цель и выявляются задачи.

В первой главе обосновывается актуальность проделанной работы, представляются теоретические сведенья о машинном обучении, в частности нейронные сети.

Во второй главе описываются существующие нейросетевые модели, так же подготавливаются наборы данных для их обучения.

В третьей главе производится обучения разработанных нейросетевых архитектур на основе подготовленных наборов данных.

В заключении подводятся итоги исследования, формируются окончательные выводы по рассматриваемой теме.

1 АНАЛИЗ МЕТОДОВ СПЕКТРОСКОПИИ И МАШИННОЕ ОБУЧЕНИЯ

1.1 Анализ проблемы неинвазивной диагностики кожно-раковых заболеваний

Рак кожи – это обобщенное название большого количества разновидностей злокачественных опухолей. Каждая опухоль имеет свои специфические биологические особенности: клиническое проявление, тканевую структуру, метастазирование и т.д.

Чаще всего рак кожи развивается у людей, длительное время находящихся на солнце. Частота возникновения этой патологии у людей с белым цветом кожи нарастает по мере приближения к экватору. Отрицательную роль в росте заболеваемости играет истончение озонового слоя. Установлено, что истончение озонового слоя на 1% приводит к увеличению заболеваемости на 3–4%.

Доказательством канцерогенного эффекта воздействия ионизирующей радиации на кожные покровы являются случаи возникновения раковой опухоли у рентгенологов при отсутствии у них средств защиты [14].

Рак кожи может возникать при иммунодефицитных состояниях. Например, при приеме иммунодепрессантов пациентами, перенесших операцию по пересадке органов.

Хроническая травматизация кожных покровов особенно термическими или химическими факторами может приводить к появлению раковых опухолей кожи.

Основными видами рака кожи являются:

1. Базалиома не является истинно злокачественной опухолью, поскольку имеет ряд морфологических и клинических черт злокачественной опухоли: упорный инфильтративный рост с разрушением нижележащих тканей и склонность к рецидивированию даже после радикального иссечения. Однако эта опухоль крайне редко метастазирует. Базалиому в настоящее время

рассматривают как полу злокачественную опухоль с местно-деструирующим ростом.

2. Плоскоклеточный рак кожи характеризуется не только агрессивным местным ростом, но и способностью к диссеминации (лимфогенному и гематогенному метастазированию). Рак кожи возникает чаще всего на открытых участках тела – кожа лица, головы, шеи (до 80%) и преимущественно у пожилых людей (возраст старше 50 лет). При чем, в 10% случаев могут возникать 2 и более очагов злокачественного роста.

3. Меланома в структуре злокачественных опухолей кожи составляет 6 – 7%, однако ввиду очень агрессивного течения заболевания занимает первое место в структуре смертности. Развивается меланома на неизмененной коже или на месте пигментного невуса. Наиболее часто локализуется на туловище (у мужчин) и на голени (у женщин). Характеризуется ранним лимфогенным и гематогенным метастазированием [4].

Меланома является наиболее агрессивным видом рака кожи и без своевременного лечения является смертельной. Ее удаление на ранних стадиях почти всегда ведет к выздоровлению, и поэтому ранняя диагностика меланомы имеет огромное значение для спасения жизни пациента.

Трудности ее диагностике возникают из-за того, что доброкачественные поражения, такие как пигментные невусы, себорейный кератоз и другие типы рака кожи, такие как базальноклеточная карцинома, могут напоминать меланому.

Удаление каждого пигментированного новообразования неприемлемо для пациента. Сообщалось, что 80% биопсий, полученных с подозрением на злокачественные поражения кожи, были доброкачественными, и поэтому неуместная хирургическое вмешательство является довольно частым случаем.

Несмотря на многочисленные попытки реализовать различные инструментальные методы, надежного неинвазивного метода диагностики кожно-раковых заболеваний пока не было найдено.

Наиболее широко развивающимися в этой области являются методы оптической спектроскопии, они позволяют неинвазивно диагностировать раковые опухоли. Биофотоника применяется в различных областях жизнедеятельности: для определения компонентного состава газов, жидкостей, порошков, твердых тел и используется в фармакологии, материаловедении, контроле продукции и других сферах.

Принцип автофлуоресценции заключается в использовании переизлучения поглощенной молекулой энергии в более длинноволновой части спектра, что позволяет получить информацию о биохимическом составе вещества [14].

Спектроскопия комбинационного рассеяния (рамановская спектроскопия англ. Raman scattering spectroscopy) – это спектральный метод изучения вещества, основанный на явлении комбинационного рассеяния монохроматического света. Суть метода заключается в регистрации спектральных линий излучения, рассеянного образцом (в твердой, жидкой или газообразной фазе). Эти спектральные линии, отсутствующие в спектре первичного (возбуждающего) излучения, соответствуют определенным колебаниям групп атомов. Это позволяет определить наличие определенных функциональных групп по характеристическим частотам колебаний их фрагментов.

Каждый из данных методов обладает рядом преимуществ и недостатков. Так, например, спектроскопия комбинационного рассеяния отличается спектром с ярко выраженными пиками, которые соответствуют наличию определенного вещества в исследуемой среде, но при этом при КР исследовании затрачивается большее количество времени, для накопления сигнала. Это не позволяет использовать КР для массового скрининга и анализа обширных областей кожи. А автофлуоресценция же наоборот может использоваться для быстрого анализа больших областей биотканей, но это сказывается на точности метода. Она ниже, нежели чем у КР. У метода отражательной спектроскопии отличительной особенностью является то, что он

позволяет оперировать только с одним измеряемым параметром – коэффициентом диффузного отражения биоткани [14].

Спектры комбинационного рассеяния и автофлуоресценции сложны. Традиционное распознавание с помощью визуального осмотра является субъективным методом, отнимающим много времени.

Использование машинного обучения в задачах дифференциации образцов кожи по результатам спектрального анализа позволит создать модель способную находить скрытые зависимости извлекать из необработанных данных шаблоны, отвечающие за классификацию кожнораковых заболеваний, увеличивающую точность распознавания патологий и в перспективе способную регистрировать смертельные новообразования на ранних стадиях, помогая врачам своевременно оказывать квалифицированную помощь.

1.2 Машинное обучение и общая характеристика искусственных нейронных сетей

За последнее десятилетие интерес к машинному обучению невероятно возрос. Сейчас оно применяется повсеместно: компьютерных программах, отраслевых конференциях, фондовых биржах, машиностроении, а также медицине. По своей сути, машинное обучение – это использование алгоритмов для извлечения информации из необработанных данных, ее представления в некоторой типизированной модели, а за тем использование этой модели для вывода информации о новых данных.

Артур Самуэль, первопроходец в области искусственного интеллекта (ИИ) в IBM и Stanford, в 1959 году в своей инновационной работе по компьютерным шашкам, определил машинное обучение следующим образом:

Машинное обучение – это процесс, в результате которого машина (компьютер) способна показывать поведение, которое в нее не было явно заложено (запрограммировано).

Формальное определение машинного обучения дал Том Митчелл – американский ученый, профессор Университета Карнеги-Меллон.

Говорят, что компьютерная программа обучается на основе опыта E по отношению к некоторому классу задач T и меры качества P, если качество решения задач из T, измеренное на основе P, улучшается с приобретением опыта E.

В бакалаврской работе для решения задачи дифференциации патологий использовались модель машинного обучения – искусственная нейронная сеть.

Искусственные нейронные сети – это класс моделей машинного обучения, в основе которых лежат исследования центральной нервной системы млекопитающих. Нейросеть состоит из нескольких взаимосвязанных нейронов, организованных в слои, которые обмениваются между собой сообщениями (возбуждаются) при выполнении определенных условий.

Общая структура нейронной сети продемонстрирована на рисунке 1.1. Элементарным преобразователем в данных сетях является искусственный нейрон или просто нейрон, названный так по аналогии с биологическим прототипом. К настоящему времени предложено и изучено большое количество моделей нейроноподобных элементов и нейронных сетей.

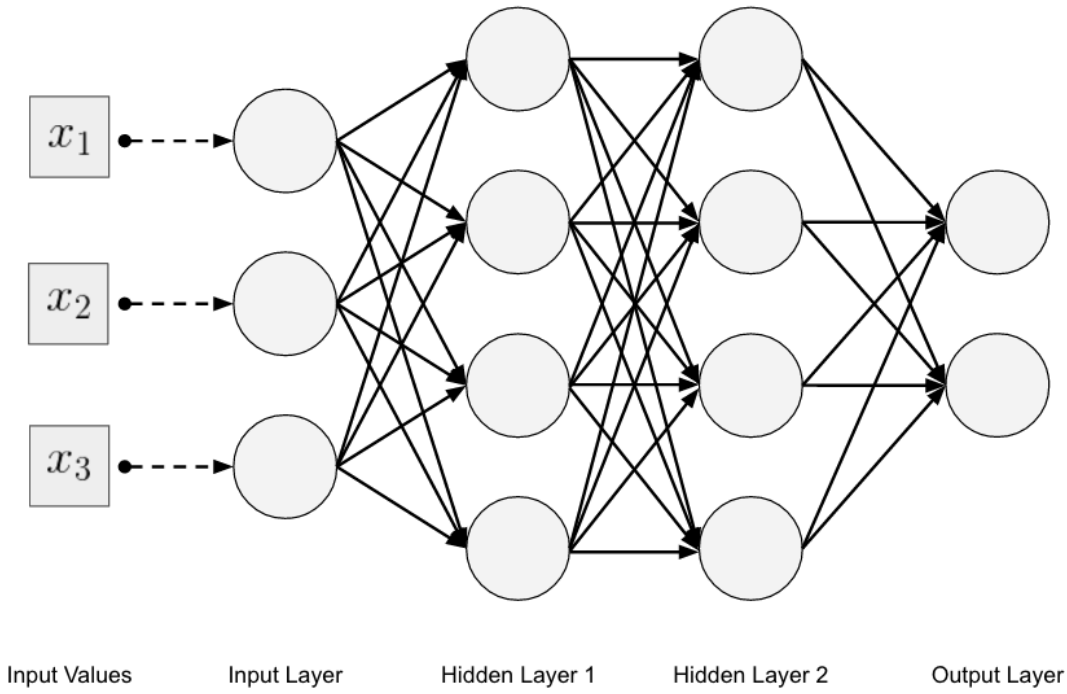


Рисунок 1.1 – Классическая топология нейросети, со входным (Input Layer), выходным (Output Layer), принимающим решение о классе, и ассоциативным (Hidden Layer) слоем

Нейрон является составной частью нейронной сети. Он состоит из элементов трех типов: умножителей (синапсов), сумматора и нелинейного преобразователя. Синапсы осуществляют связь между нейронами, умножают входной сигнал на число, характеризующее силу связи (вес синапса). Сумматор выполняет сложение сигналов, поступающих по синоптическим связям от других нейронов, и внешних входных сигналов. Нелинейный преобразователь реализует нелинейную функцию одного аргумента – выхода сумматора. Эта функция называется функцией активации или передаточной функцией нейрона. Общая структура нейрона продемонстрирована на рисунке 1.2.

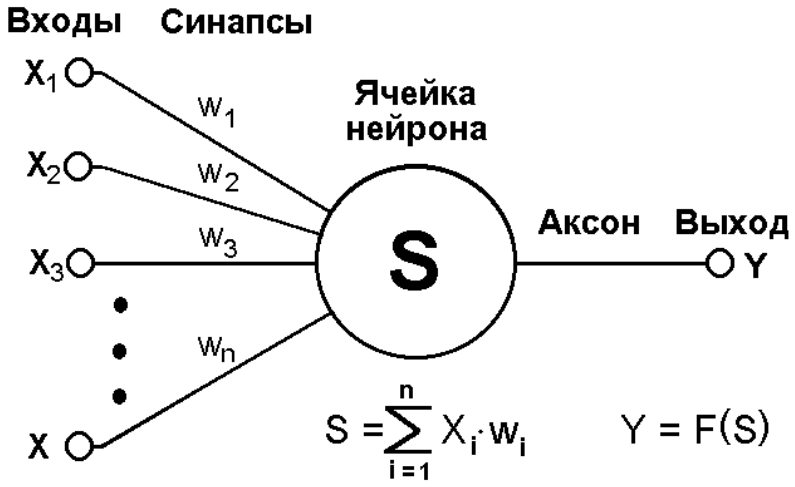


Рисунок 1.2 – Структура искусственного нейрона

Нейрон в целом реализует скалярную функцию векторного аргумента. Математическая модель нейрона показана на формуле 1.1:

$$S = \sum_{i=1}^{n} w_i x_i + b;\ \ y = f(s) \qquad (1.1)$$

где S – результат суммирования (sum);

$w(i)$ – вес (weight) синапса, $i = 1, n$;

x – компонент входного вектора (входной сигнал), $i = 1, n$;

b – значение смещения (bias);

n – число входов нейрона;

y – выходной сигнал нейрона;

$f(s)$ – нелинейное преобразование (функция активации) [4].

В общем случае входной сигнал, весовые коэффициенты и смещение могут принимать действительные значения, а во многих практических задачах – некоторые фиксированные значения. Выход y определяется видом функции активации и может быть, как действительным, так и целым.

В исследовании рассматривается проблема неинвазивной диагностики раковых заболеваний тканей кожи человека. Проводится анализ результатов спектроскопии комбинационного рассеивания и автофлуоресценции, для выявления признаков дифференциации кожных патологий с использованием различных архитектур нейронной сети.

Исходя из объекта и предмета исследованиядля достижения поставленной цели были определены следующие задачи:

– Анализ способов оптической спектроскопии.

– Проектирование нейросетевых архитектур для решения задач дифференциации патологий по результатам спектроскопии.

– Препроцессинг входных данных и создание наборов для обучения нейронных сетей.

– Тестирование разработанных архитектур.

В ходе выполнения выпускной квалификационной работы должна быть исследование точности дифференциации патологий по результатам АФ и КР спектроскопии с помощью нейросетевых алгоритмов.

В первой разделе дипломной работы была обоснована актуальность выбранной темы, а также рассмотрены теоретические сведения необходимые для проектирования искусственных нейронных сетей для задачи дифференциации кожных патологий.

2 ПРОЕКТИРОВАНИЕ НЕЙРОСЕТЕВЫХ АРХИТЕКТУР

2.1 Используемое программное обеспечения

При выполнении бакалаврской работы использовались высокоуровневые языки программирования MATLAB 9.4 (R2018a), Python 3.6, предоставляющие широкий инструментарий для решения задач машинного обучения.

MATLAB (matrix laboratory) – представляет собой многофункциональную вычислительную среду, работа в которой осуществляется на одноименном запатентованном языке программирования, разработанным в MathWorks. MATLAB позволяет манипулировать матрицами, строить функции и данные, реализовывать алгоритмы, создавать пользовательские интерфейсы и взаимодействовать с программами, написанными на других языках, включая C, C ++, C #, Java, Fortran и Python.

Среда разработки MATLAB также включает в себя пакет расширения Neural Network Toolbox, содержащий средства для проектирования, моделирования, разработки и визуализации нейронных сетей. Пакет обеспечивает всестороннюю поддержку типовых нейросетевых парадигм и имеет открытую модульную архитектуру. Пакет содержит функции командной строки и графический интерфейс пользователя для быстрого пошагового создания нейросетей.

Ядро MATLAB позволяет максимально просто работать с матрицами реальных, комплексных и аналитических типов данных и со структурами данных и таблицами поиска.

MATLAB содержит встроенные функции линейной алгебры, быстрого преобразования Фурье, функции для работы с полиномами, функции базовой статистики и численного решения дифференциальных уравнений, а также расширенные математические библиотеки для Intel MKL.

Все встроенные функции ядра MATLAB разработаны и оптимизированы специалистами и работают быстрее их эквивалентов на C/C++ или Java [17].

Python – это интерпретируемый язык программирования высокого уровня для программирования общего назначения. Созданный Guido van Rossum и впервые выпущенный в 1991 году, Python придерживается философии, которая подчеркивает читаемость кода, выделяя блоки отступами (significant whitespace). Он предоставляет конструкции, которые обеспечивают четкое программирование как в малых, так и в больших масштабах [18].

Python имеет систему динамической типизации и автоматическое управление памятью. Он поддерживает несколько парадигм программирования, в том числе объектно-ориентированную, императивную, функциональную, процедурную, и имеет большую и всеобъемлющую стандартную библиотеку [18].

В последнее время, язык программирования Python стал качественным инструментом для научных вычислений, включая анализ и визуализацию больших наборов данных. Язык программирования Python пригоден для науки о данных в основном благодаря большой и активно развивающейся экосистеме пакетов, созданных сторонними разработчиками:

NumPy – библиотека для работы с однородными данными в виде массивов.

Pandas – библиотека для работы с неоднородными и поименованными данными.

SciPy – библиотека для общих научных вычислительных задач.

Написания Python кода проводились в бесплатной кросплатформенной IDE Spyder, имеющей открытый исходный код. В Spyder интегрированы библиотеки NumPy, SciPy, Matplotlib и IPython.

В данной бакалаврской работе особое внимание уделяется работе с библиотекой Keras, версии 2.1.6. Это – нейросетевая библиотека с открытым исходным кодом, написанная на Python. Она представляет собой надстройку над фреймворками TensorFlow, Microsoft Cognitive Toolkit, Theano [19]. В данный момент по умолчанию Keras использует Tensorflow. С которым мы и будем работать.

Разработанный для быстрой работы с глубокими нейронными сетями, он фокусируется на удобстве, модульности и расширяемости. Библиотекой Keras была разработана в рамках исследовательской работы проекта ONEIROS (Open Neun-Electronic Intelligent Robot Operating Systemи ее основным автором является инженер Франсуа Холле) [19].

Tensorflow – фреймворк для глубокого машинного обучения, разработанная компанией Google для решения задач построения и тренировки нейронной сети с целью автоматического нахождения и классификации образов. Основное API для работы с библиотекой реализовано для Python, также существуют реализации для C++, Haskell, Java и Go. В 2015 году система была переведена в свободный доступ с открытой лицензией Apache 2.0

2.2 Анализ результатов автофлуоресценции для дифференциации образцов кожи

Для формирования обучающей выборки использовались результаты автофлуоресценции, проведённой на 21 образце нормальной ткани кожи и 21 измерение образцов злокачественных образований.

Для анализа выбирался участок 550-750 нм в регистрируемом спектре для синего лазера (457 нм), пример спектов изображен на рисунке 2.1. Нормализация спектра осуществлялась относительно главного экстремума каждого образца. Формы спектров биотканей обуславливаются наличием коллагенов, эластина, фосфолипидов и других компонентов, также в ходе анализа экспериментальных данных был выявлен характерный пик на длине волны 640 нм, обусловленный наличием порфиринов [5].

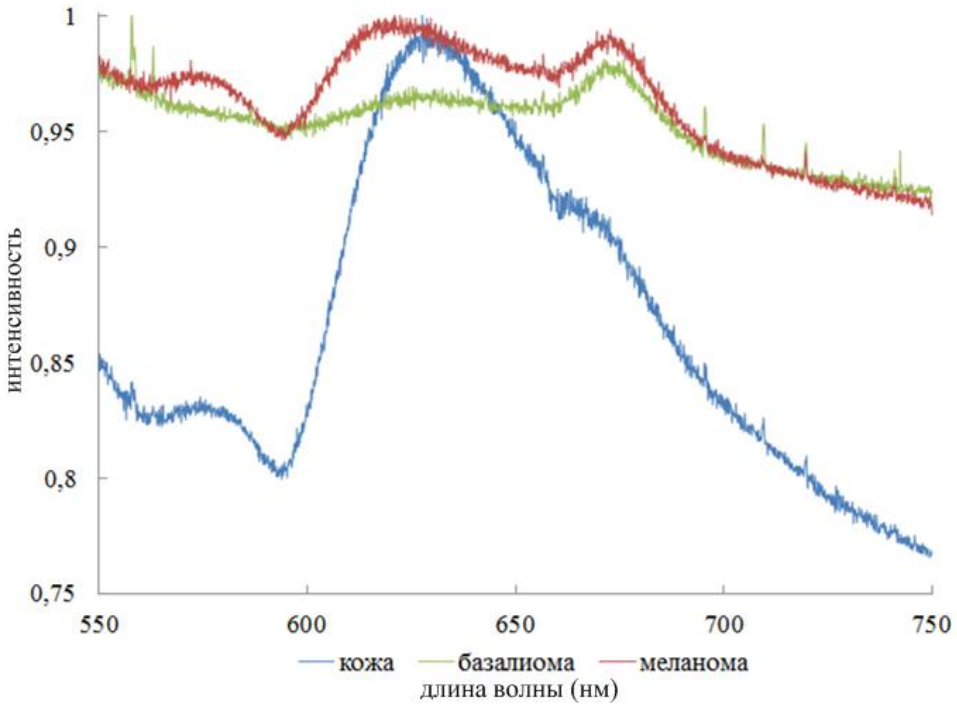


Рисунок 2.1 – Спектры АФ здоровой кожи, базалиомы и меланомы

Анализ экспериментальных данных позволил выявить основные полосы 580 и 620 нм, характеризующие патологически измененные биоткани. Форма спектра и интенсивность излучения в выбранных полосах позволили провести дифференциацию нормальной и патологически измененной кожи.

Для определения наличия злокачественного образования был введен коэффициент $I = I580/I620$, равный отношению интенсивностей спектров авто- флуоресценции в выбранных областях. Если значение коэффициента I было меньше 0,81, то считалось, что была исследована здоровая кожа, если значение коэффициента I было между 0,81 и 0,96, то считалось что мы исследовали базалиому, если значение I было больше 0,96, то считалось, что мы промерили меланому. На рисунке 2.2 представлены зависимости положения значения коэффициента I. [14]

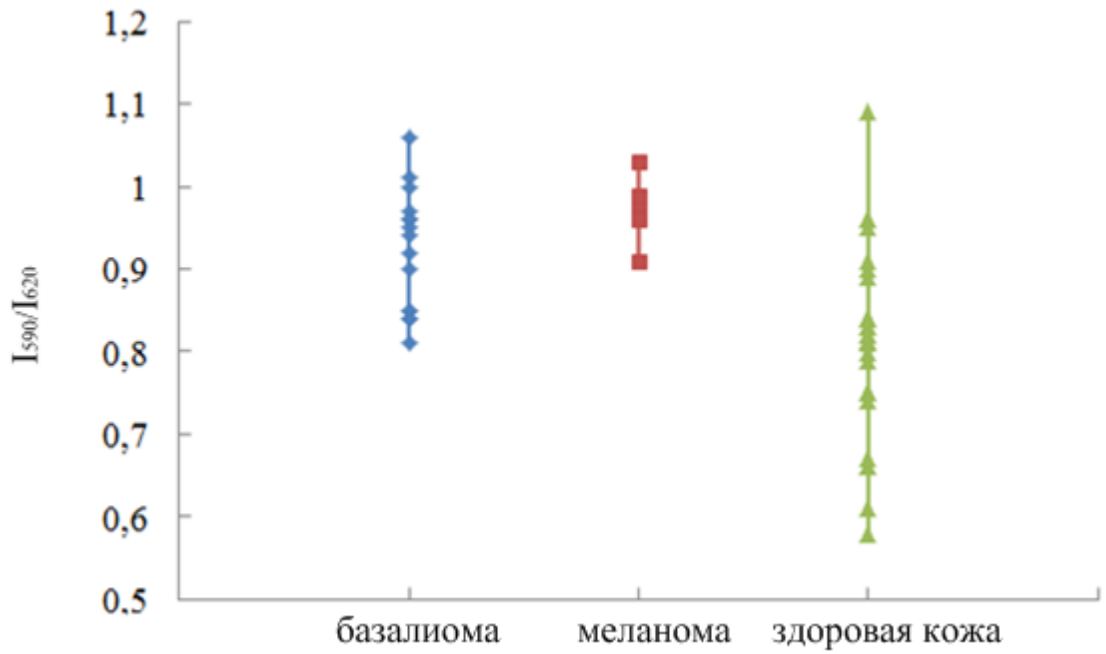


Рисунок 2.2 – Положение коэффициента I АФ здоровой кожи, базалиомы и меланомы

В ходе анализа экспериментальных исследований при регистрации спектров были получены данные о положения основных максимумов, они располагались в границах 550 – 575 нм 600 – 625 нм.

На рисунке 2.1 видно, что в АФ спектре здоровой ткани и патологии есть два явных максимума. Положения максимумов меняются не только по интенсивности, но и по длине волны. Это происходит из-за изменения количественного состава компонентов образца. При этом происходит смещение основных максимумов спектра АФ по длине волны. Это можно использовать для более детального анализа дифференциации спектра здоровой кожи и патологий. Для данных спектров характерны два основных максимума. На рисунке 2.3 представлены длины волн, соответствующие положениям максимумов АФ, измеренных образцов в диапазоне 550 – 640 нм.

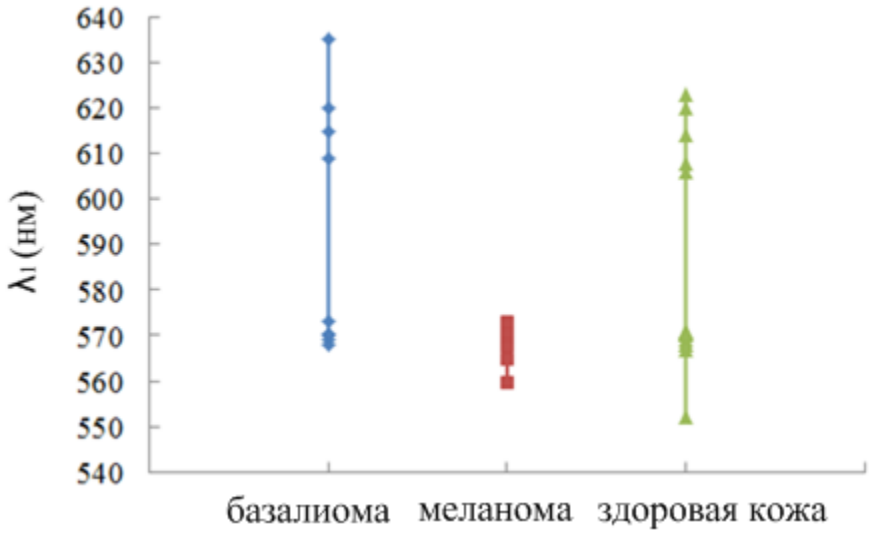


Рисунок 2.3 – Положения максимумов АФ здоровой кожи, базалиомы и меланомы в диапазоне 550 – 640 нм

На рисунке 2.4 представлены основные положения максимумов АФ в диапазоне 600 – 690 нм.

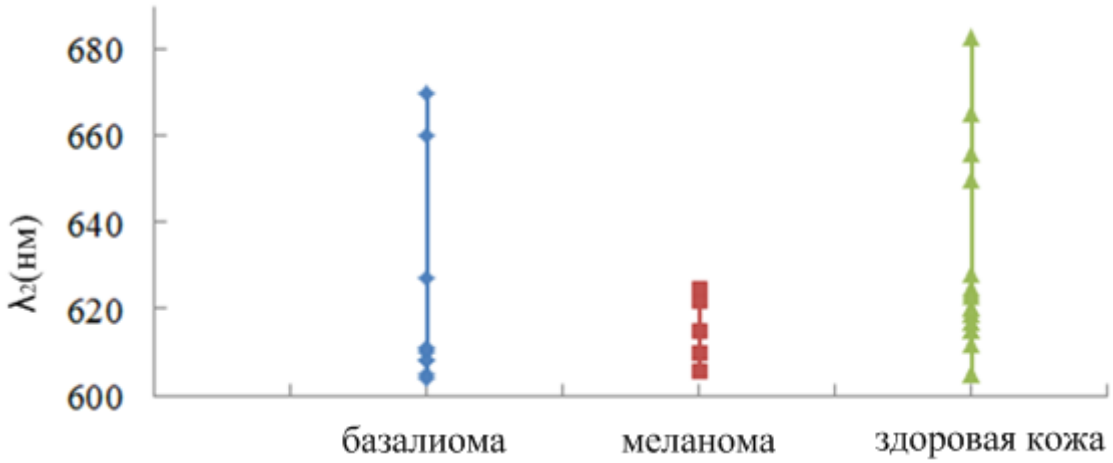


Рисунок 2.4 – Положения максимумов АФ здоровой кожи, базалиомы и меланомы в диапазоне 600 – 690 нм

Вид зависимости на рисунках 2.3 и 2.4 позволяет сформулировать новую методику дифференциации меланомы от других типов тканей. Критерием дифференциации меланомы от других новообразований является расположение ее максимумов в диапазонах 555 – 575 нм и 600 – 625 нм.

Для повышения точности методики дифференциации патологии от здоровой ткани была использована нейронная сеть, обученная на классифицирующих признаках спектров АФ.

$I580$ – значение минимума спектра в диапазоне 580-620нм;

$I620$ – значение максимума спектра в диапазоне 580-620нм;

I – отношение интенсивности спектров АФ на длинах волн равных 580 и 620нм.

λ_1 – расположение максимума спектра в диапазоне 555-575нм;

λ_2 – расположение максимума спектра в диапазоне 600-625нм.

Значение признаков уникально для каждого отдельного спектра АФ, а их совокупность позволяет дифференцировать патологию от кожи.

Для поиска максимумов/минимумов из заданных диапазонах спектра, использовался алгоритм нахождения точки при которой значение интенсивности спектров, расположенных по обе стороны от проверяемой, ниже/выше проверяемой на *delta*(пиковый порог). [5]

Для реализации данного метода использовалась функция *peakdet*, листинг которой представлен на рисунке 2.5. Обнаружение пиков в векторе *[maxtab, mintab] = peakdet (v, delta)* находит локальные максимумы и минимумы («пики») в векторе v. Maxtab и mintab состоит из двух столбцов. Столбец 1 содержит индексы вектора V, а столбец 2 - найденные значения.

С помощью *[maxtab, mintab] = peakdet (v, delta, x)* индексы в maxtab и MINTAB заменяются соответствующими значениями X.

Точка считается максимальным пиком, если она имеет максимальное значение, и ей предшествовало (слева) значение, меньшее значением delta.

```
function [maxtab, mintab]=peakdet(v, delta, x)
maxtab = [];
mintab = [];
if nargin < 3
  x = (1:length(v))';
else
  x = x(:);
  if length(v)~= length(x)
    error('Входные векторы v и x должны иметь одинаковую длину');
  end
end
if (length(delta(:)))>1
  error('Входной аргумент DELTA должен быть скаляром');
end
if delta <= 0
  error('Входной аргумент DELTA должен быть положительным');
end
mn = Inf; mx = -Inf;
mnpos = NaN; mxpos = NaN;
lookformax = 1;
for i=1:length(v)
  this = v(i);
  if this > mx, mx = this; mxpos = x(i); end
  if this < mn, mn = this; mnpos = x(i); end
  if lookformax
    if this < mx-delta
      maxtab = [maxtab ; mxpos mx];
      mn = this; mnpos = x(i);
      lookformax = 0;
    end
  else
    if this > mn+delta
      mintab = [mintab ; mnpos mn];
      mx = this; mxpos = x(i);
      lookformax = 1;
    end
  end
end
```

Рисунок 2.5 – Листинг функции поиска максимумов/минимумов из заданных диапазонах спектра

Пиковый порог был установлен выше максимального размаха шума спектра. Из множества найденных пиков по всей длине волны в диапазонах 555-575нм, 600-625нм выбирается максимальный/минимальный пик. На рисунке 2.6 продемонстрирован результат работы данного алгоритма. Красным отображается спектр здоровой кожи, бирюзовым патология. Максимумы спектров выделены зеленой звездой, минимумы показынный синей звездой.

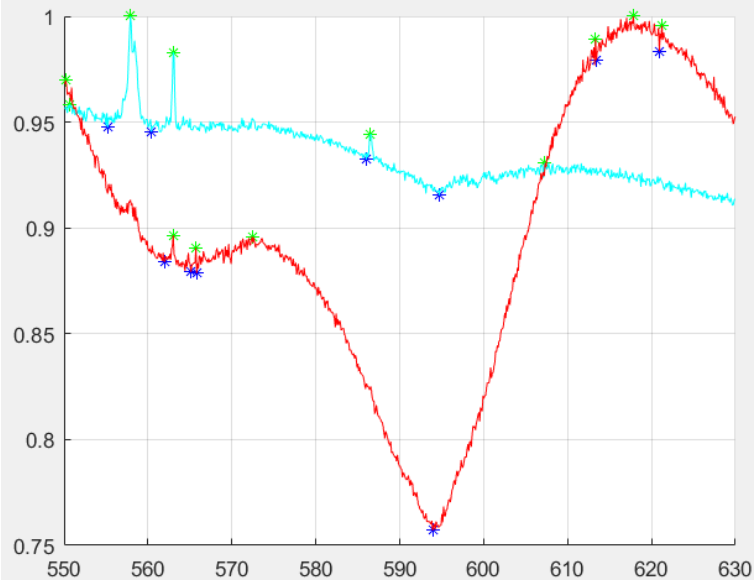


Рисунок 2.6 – Нормированные спектры кожи и патологии с выделенными максимумами и минимумами.

Основной проблемой для вычисления классифицирующих признаков спектра является образцы в которых отсутствуют явные максимумы в диапазонах 555-575нм, 600-625нм.

Для этих случаев использовалось рекурсивное уменьшение значения пикового порога, до тех пор, пока в проверяемых диапазонах не появятся значения пиков из которых будет выбран максимальный.

```
function lambda=Fiend_lambda_MIN(semple,alfa,betta,n)
delta=0.01;
id_lambda=[];
   while isempty(id_lambda)
   [~, min_peaks]=peakdet(semple(2,:), delta, semple(1,:));%пики АФ (паталогия)
   if ~isempty(min_peaks)
      if min_peaks(1,1)==semple (1,2)||min_peaks(1,1)==semple (1,1)
            min_peaks (1,:)=[];
      end
   end
  if ~isempty(min_peaks)
   id_lambda=find(min_peaks(:,1)>alfa&min_peaks(:,1)<betta);%адрес пиков 555-575
(паталогия)
  end
   delta=delta-0.0009;
   end        [lambda_N1_1,~]=find(min_peaks==max(min_peaks(id_lambda(:),2)));
   lambda=min_peaks(lambda_N1_1,n);
end
```

Рисунок 2.7 – Листинг функции рекурсивное уменьшение значения пикового порога

Данный способ позволяет решить проблему отсутствия значения классифицирующего признака, но при этом появляется вероятность искажения значений, используемых в обучающей выборке.

Задачи дифференциации патологий от здоровых биотканей, является задачей бинарной классификации. На основе проведенного исследования была составлена обучающая выборка, состоящая из двух наборов векторов-признаков размерность 1x5.

Признаки, помещенные в вектор, не имеют пространственной связи и для их дифференциации подойдет сеть прямого распространения выполняющая роль классификатора.

Но при данном подходе существует вероятность упустить скрытые зависимости, в результате чего было принято решение перейти к архитектуре, объединяющей в себе поиск классифицирующих признаков и их последующую классификацию – сверточные нейронные сети

2.3 Сверточные нейронные сети

Сверточная нейронная сеть (CNN или ConvNet) представляет собой класс глубоких искусственных нейронных сетей прямого распространения, которые были успешно применены для анализа визуальных образов с высокой степенью инвариантности к масштабированию, смещению, повороту, смене ракурса и прочим пространственным искажениям[4].

Целью CNN является глубокое изучение данных по средствам использования сверток. Они эффективны при распознавании объектов на изображениях, а также успешно выполняют задачи по их классификации. CNN могут распознавать лица, уличные знаки, собак и множество других визуальных данных. Область применения CNN так же затрагивает анализ текста с помощью оптического распознавания символов.

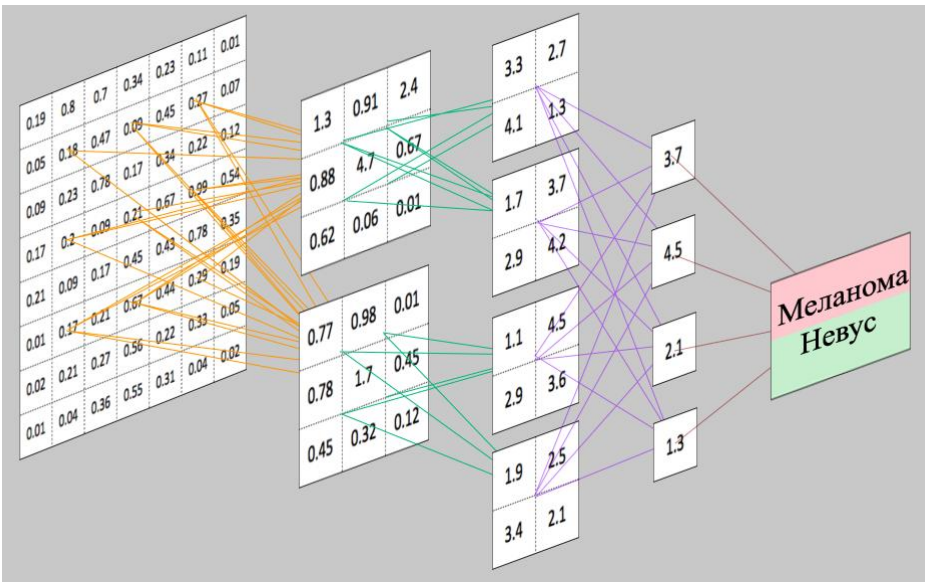


Рисунок 2.8 – Визуализация работы CNN

CNN, как правило, наиболее эффективны, когда у входных данных прослеживается некоторая структура. Примером может служить, как изображения, так и аудиоданные, имеющие набор повторяющихся шаблонов, и поступающие друг с другом входные значения имеют некую пространственную связь.

2.4 Обзор архитектуры CNN

CNN преобразуют входные данные поступающие на входной слой, проводя их через все подключенные слои в набор оценок классов, задаваемых выходным слоем. Существует много вариантов архитектур CNN, но все они основаны на архитектуре, представленной на рисунке 2.9. На нем показаны три основные группы слоев:

1. Входной слой (input layer).
2. Признаковый слой (feature-extraction layer).
3. Классификатор (Classification layer).

Структура входных данных обычно имеет трехмерный вид: ширина, высота изображения, а также глубина, представляющая собой цветовые каналы (к примеру, три для цветовой модели RGB). [9]

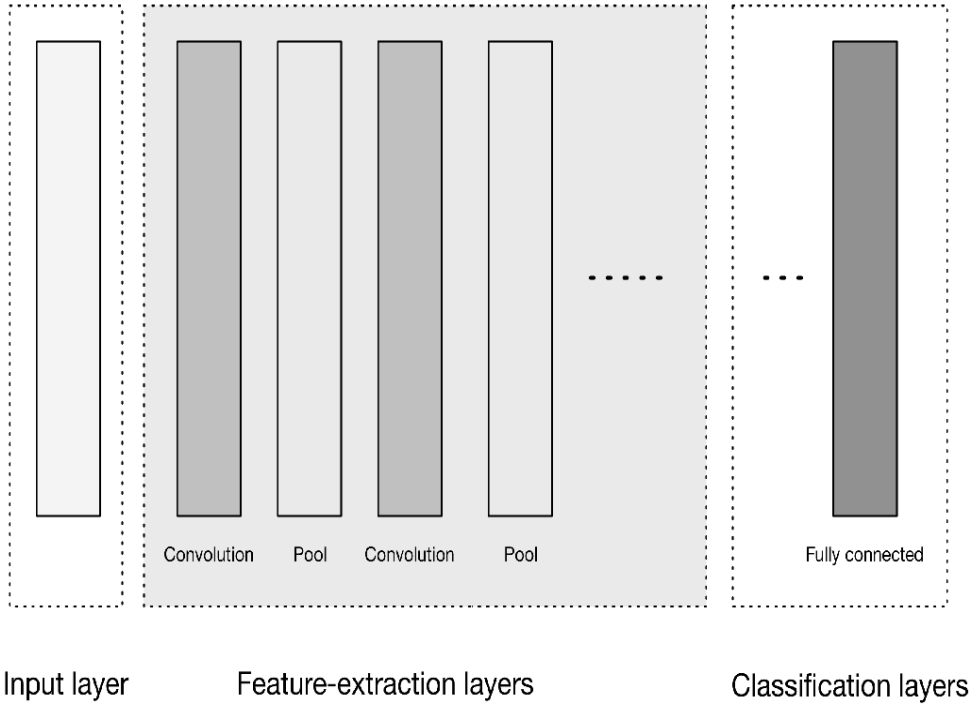


Рисунок 2.9 – Общая архитектура CNN высокого уровня

Входной слой – это место, где мы загружаем и сохраняем исходные входные данные изображения для их дальнейшей обработки сетью. Входные данные определяют ширину, высоту и количество каналов слоя.

Слои выделения признаков имеют общую повторяющуюся структуру, Слой свертки, Слой пула. Эти слои находят признаки на изображениях постепенно составляя карты.

Слой классификации состоит из единого или несколько полносвязных слоев, для дифференцировать карты признаков и вычисления вероятности их принадлежности к заданным классам. Размеры выходных данных слоя равны [1 × 1 × N], где N – количество оцениваемых классов.

Сверточные слои считаются основными строительными блоками архитектур CNN. Как показано на рисунке 2.10, сверточные слои преобразуют входные данные, используя заплатку из локально связанных нейронов предыдущего слоя. Слой будет вычислять скалярное произведение между областью нейронов во входном слое и весами, с которыми они локально связаны в выходном слое. [9]

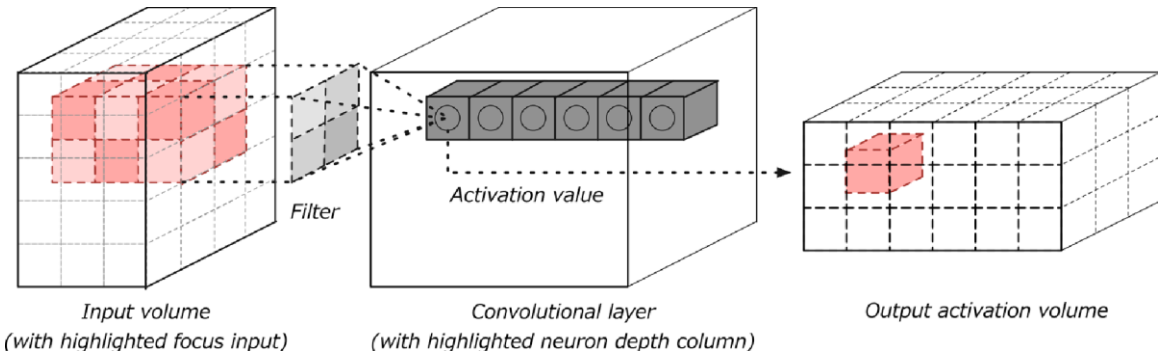


Рисунок 2.11 – Слой свертки с входными и выходными томами

Полученный результат обычно имеет одинаковые (или уменьшенные) пространственные размеры, но иногда увеличенное количество элементов третьего измерения, глубины. Рассмотрим концепцию свертки немного подробнее.

Свертка – это математическая операция, описывающая правило для объединения двух наборов информации. Операция свертки, показанная на рисунке 2.12, известна как детектор признаков CNN. На вход свертки могут посылаться как необработанные данные, так и вывод из другой свертки. Она часто интерпретируется как фильтр, в котором ядро фильтрует входные данные для определенных видов информации.

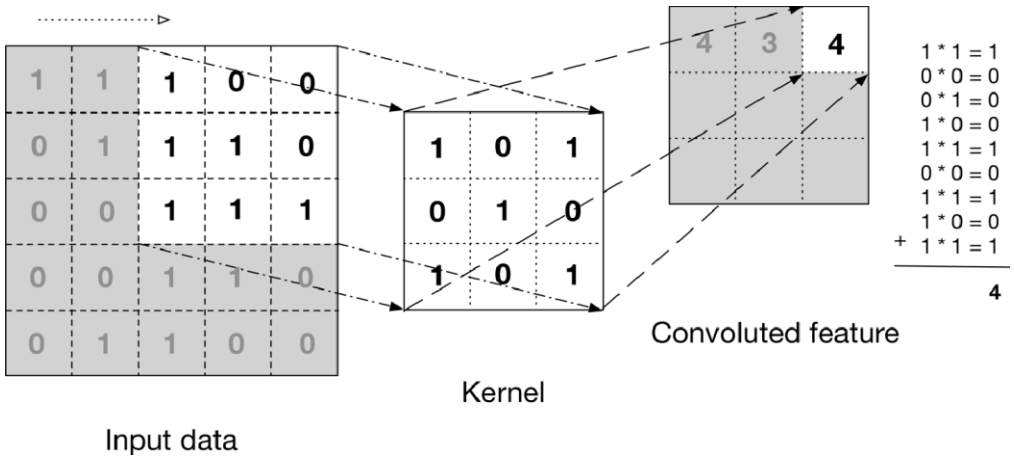


Рисунок 2.12 – Операция свертки

На рисунке показано, как ядро скользит по входным данным для получения данных с запутанными характеристиками (выводами). На каждом

шаге ядро умножается на значения входных данных в пределах его границ, создавая единую запись на выходной карте признаков.

Введем понятие оператора свертки. Имея двумерное изображение I и небольшую матрицу K размерности $h \times w$ (так называемое ядро свертки), построенная таким образом, что графически кодирует какой-либо признак, мы вычисляем свернутое изображение $I * K$, накладывая ядро на изображение всеми возможными способами и записывая сумму произведений элементов исходного изображения и ядра:

$$(I * K)_{xy} = \sum_{i=1}^{h} \sum_{j=1}^{w} K_{ij} \times I_{x+i-1,y+j-1}, \quad (2.1)$$

где I двумерное изображение;

K – ядро свертки;

$I * K$ – искомое свернутое изображение;

x, y, h, w – индексы матриц.

Оператор свертки составляет основу сверточного слоя (convolutional layer) в CNN. Слой состоит из определенного количества ядер K (с аддитивными составляющими смещения b для каждого ядра) и вычисляет свертку выходного изображения предыдущего слоя с помощью каждого из ядер, каждый раз прибавляя составляющую смещения. В конце концов ко всему выходному изображению может быть применена функция активации σ. Обычно входной поток для сверточного слоя состоит из d каналов, например, red/green/blue для входного слоя, и в этом случае ядра тоже расширяют таким образом, чтобы они также состояли из d каналов; получается следующая формула для одного канала выходного изображения сверточного слоя, где K – ядро, а b – составляющая смещения:

$$conv(I, K)_{x,y} = \sigma(b + \sum_{i=1}^{h} \sum_{i=1}^{h} \sum_{k=1}^{d} K_{ijk} \times I_{x+i-1,y+j-1,k}), \quad (2.2)$$

где σ – функция активации;

K – ядро свертки;

b – составляющая смещения.

2.5 Слои субдискретизации

Слои субдискретизации – слой подвыборки, пулинговые слои (от англ. downsampling или pooling layer) обычно устанавливаются между последовательными сверточными слоями, с целью постепенного уменьшения пространственных размеров (ширины и высоты) данных. Объединение слоев постепенно уменьшает размерность данных в сети и помогает контролировать переобучение.

Самыми распространёнными операциями субдискретизации являются:

- операция max pooling;
- операция average pooling.

Слои субдискретизации использует операцию *max()* для изменения пространственных характеристик (ширина, высота) входных данных. Эта операция называется max pooling. При размере фильтра *2×2* операция *max()* принимает значение наибольшего числа попавшего в фильтр. Эта операция не влияет на глубину данных. В пуллинговых слоях используются фильтры для выполнения процесса понижающей дискретизации входных данных. Эти слои выполняют операции уменьшения размерности по пространственному размеру входных данных [9].

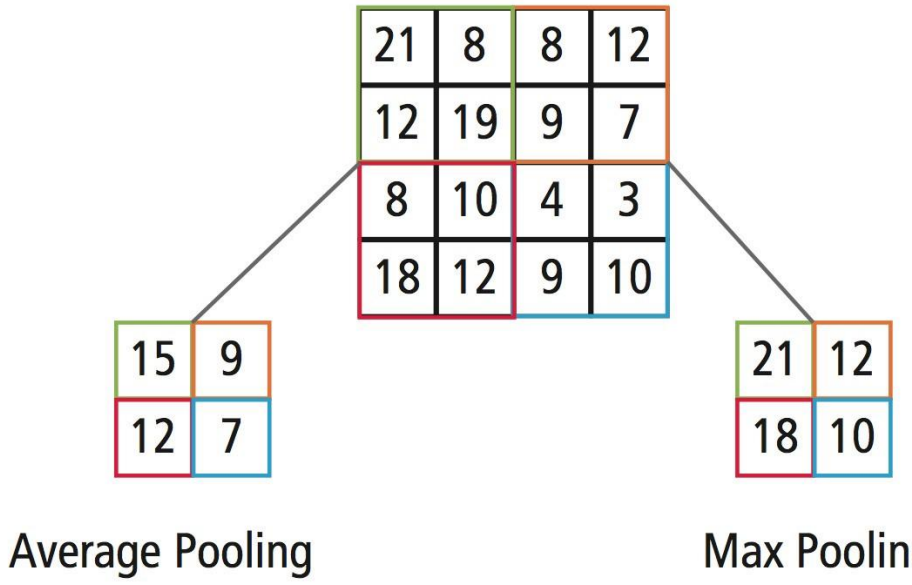


Рисунок 2.13 – Визуализация работы слоя субдискретизации

Чаще всего в пуллинговых слоях используется фильтр 2×2 с шагом 2. Данные характеристи позволяют уменьшать каждый фрагмент глубины входных данных в два раза по пространственным размерам (ширина и высота).

2.6 Предобученные сверточные нейронные сети

Для работы со сверточными сетями в качестве входных данных использовалась выборка из 363 образцов патологии среди которых: меланома, базалиома, невус.

Эффективным решением в области распознавания патологий на Раманавской спектрограмме, может стать использование предобученных нейронных сетей, таких как: Xception, VGG16, VGG19, ResNet50, DenseNet. В частности, применение их сверточных слоев, совместно с кастомным классификатором.

Достаточно похожих, с помощью технологии *переноса обучения (transfer learning)*. Для этого от предварительно обученной сети “обрезается” классификатор и вместо него добавляется новый, приспособленный для нашей задачи. Например, вместо классификатора, обученного на наборе данных ImageNet с 1000 классов, мы добавляли в нейронную сеть собственный

классификатор, в котором всего два класса. Затем этот классификатор обучается на новых данных.

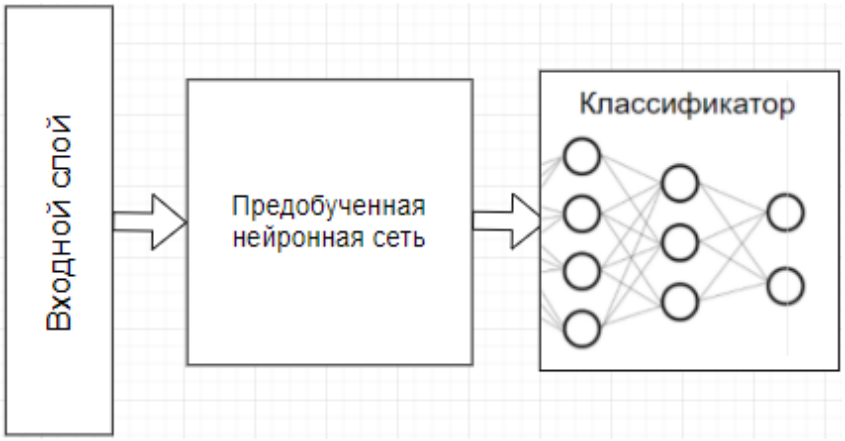


Рисунок 2.14 – Общая архитектура нейронной сети на основе предобученных классификаторов

Keras предоставляет нейронные сети обучавшиеся в течении долгого времени. Такие предобученные нейронные сети могут быть доученны для решения частных задач.

Все нейронные сети предобучались на наборе данных ImageNet. Это проект по созданию и сопровождению массивной базы данных аннотированных изображений, предназначенная для отработки и тестирования методов распознавания образов и машинного зрения.

Таблица 2.1 – Список предобученных нейронных сетей доступных в Keras

Модель	Размер	Минимальный размер входного тензора	Top-5 Accuracy	Количество параметров
Xception	88 MB	71x71	0.945	22,910,480
VGG16	528 MB	48x48	0.901	138,357,544
VGG19	549 MB	48x48	0.910	143,667,240
ResNet50	99 MB	197x197	0.929	25,636,712
InceptionV3	92 MB	139x139	0.944	23,851,784
InceptionResNetV2	215 MB	139x139	0.953	55,873,736
MobileNet	17 MB	128x128	0.871	4,253,864

Использование данного метода ставит ограничение на входную матрицу, обязуя использовать предобработку, направленную на увеличения размерности

входных данных до требуемых размеров предобученных нейронных сетей. Для тестирования рассматривались сети с наименьшим размером входного тензора: Xception, VGG16, VGG19. Рассмотрим их архитектуры.

Рассмотрим нейронную сеть Xception. Xception предлагает сверточную нейронную сетевую архитектуру, полностью основанную на разделяймых по уровню слоях свертки. Основанную на следующей гипотезе: сопоставления межканальных корреляций и пространственных корреляций в характеристических картах сверточных нейронных сетей может быть полностью расципленно.

Полное описание спецификаций сети приведено на рисунке 2.15. Архитектура Xception: данные сначала проходят через поток ввода, затем через средний поток, который повторяется восемь раз, и, наконец, через выходной поток, она имеет 36 сверточных слоев, формирующих базу извлечения объектов сети. 36 сверточных слоев структурированы в 14 модулей, все из которых имеют линейные остаточные соединения вокруг них, за исключением первого и последнего модулей. Иначе говоря, архитектура Xception представляет собой линейный стек разделяемых по глубине слоев свертки с остаточными соединениями [12].

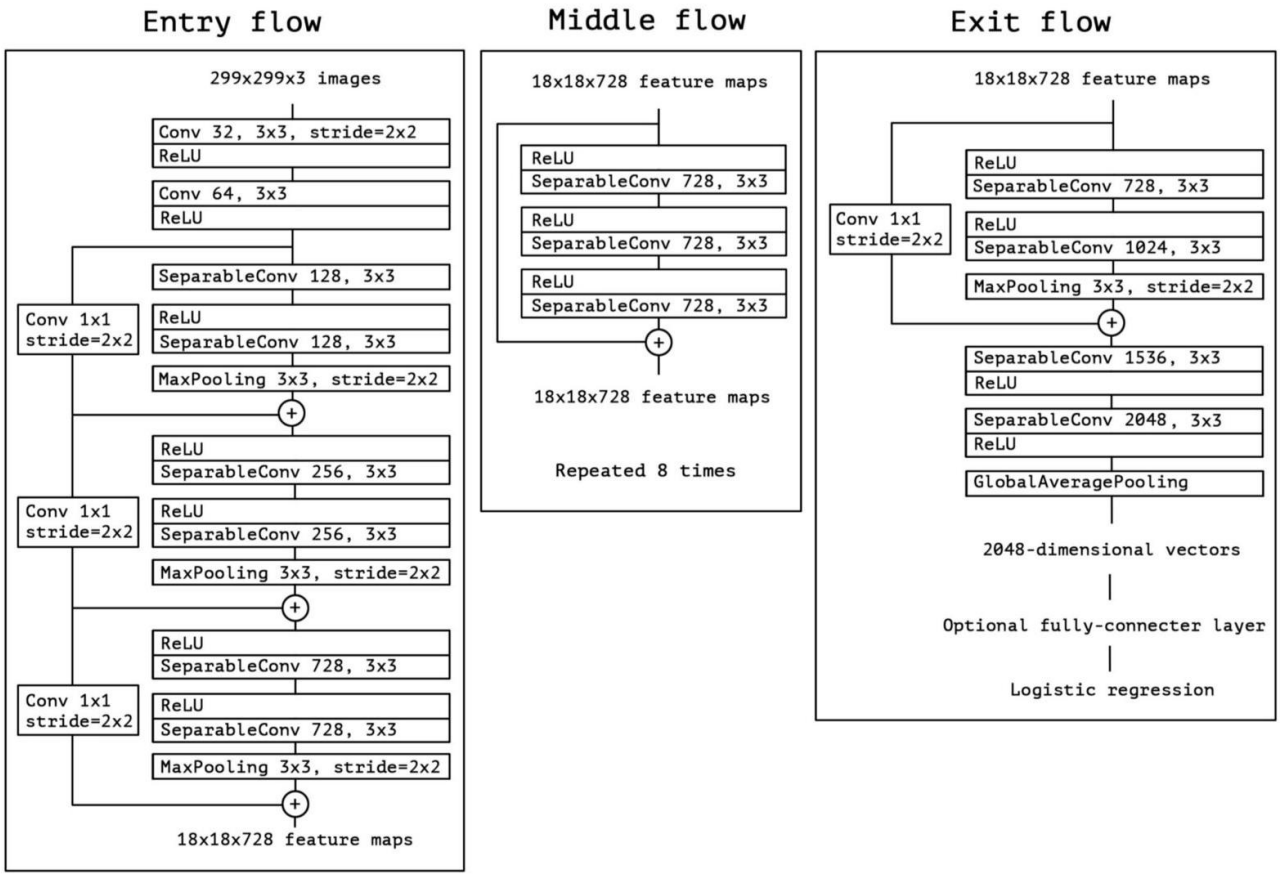


Рисунок 2.15 – архитектура сети Xception

Архитектура сети Xception доступна для загрузки из фреймворка Keras:

keras.applications.Xception.Xception(include_top=True, weights='imagenet', input_tensor=None, pooling=None, classes=1000)

Размер входных данных по умолчанию для этой модели – 299x299. Аргументы:

– include_top: Подключение полносвязных слоев верхнего уровня сети

– weights: значения весов нейронной сети. *'None'* (случайная инициализация) или *'imagenet'* (предобученная на наборе данных ImageNet).

– input_tensor: optional Keras tensor (i.e. output of *layers.Input()*) to use as image input for the model.

– input_shape: optional shape tuple, only to be specified if *include_top* is False (иначе форма ввода должна быть (299, 299, 3) (с форматом данных *'channels_last'*). Он должен иметь ровно 3 входных канала, а ширина и высота

должны быть не меньше 71. Например. (150, 150, 3) будет одним из допустимых значений.

– pooling: опциональный аргумент извлечения признаков, когда аргумент include_top - False.

– *None* означает, что выходом модели будет 4D тензором на выходе последнего сверточного слоя.

– *avg* означает что для сети будет применять операция average pooling

– *max* означает что для сети будет применять операция max pooling

– classes: необязательный аргумент обозначающий количества классов классифицируемых изображений. Указывается если, *include_top* имеет значение *True* или если аргумент *weights* не указан.[16]

Так же рассмотрим архитектура предобученная нейронная сеть VGG16. VGG16 – сеть Visual Geometry Group из университета Оксфорда для распознавания объектов на изображениях, состоит из 16 слоев.

Архитектура сети VGG16 доступна для загрузки из фреймворка Keras:

keras.applications.vgg16.VGG16 (include_top=True, weights='imagenet', , input_shape=None, pooling=None, classes=1000)

Размер входных данных по умолчанию для этой модели – 224x224.

Аргументы:

– include_top: Подключение 3 полно связных слоев верхнего уровня сети

– weights: значения весов *None* (случайная инициализация) или *'imagenet'* (предобученная на наборе данных ImageNet).

– input_shape: опциональная форма кортежа, указывается только если *include_top – False.*

– input_shape: optional shape tuple, only to be specified if *include_top* is False (иначе форма ввода должна быть (224, 224, 3) (с форматом данных *'channels_last'*) или (3, 224, 224) (с форматом данных *'channels_first'*). Он должен иметь ровно 3 входных канала, а ширина и высота должны быть не меньше 48. Например. (200, 200, 3) будет одним из допустимых значений.

– pooling: опциональный аргумент извлечения признаков, когда аргумент include_top - False.

– *None* означает, что выходом модели будет 4D тензором на выходе последнего сверточного слоя.

– *avg* означает что для сети будет применять операция average pooling

– *max* означает что для сети будет применять операция max pooling

– classes: необязательное значение количества классов классифицируемых изображений если, *include_top* имеет значение *True* и не если аргумент *weights* не указан [16].

Модель VGG-19 является улучшенной версией VGG16. Она состоит из 144 миллионов параметров и добавляет в архитектуру, помимо 84 миллионов параметров, еще одну простую идею. Возьмем для примера свертку 5 × 5, это отображение $f: R^{25} \rightarrow R$, оно содержит 25 параметров. Если заменить ее стеком из двух слоев со свертками 3 × 3, то мы получим такое же отображение, но количество параметров будет меньше: 3 × 3 + 3 × 3 = 18, а это на 22 % меньше. Если же заменить 11 × 11 на четыре свертки 3 × 3, то это уже на 70 % меньше параметров.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Рисунок 2.16 – Архитектура сети VGG19

Архитектура сети так VGG19 доступна для загрузки из фреймворка Keras:

keras.applications.vgg19.VGG19 (include_top=True, weights='imagenet', , input_shape=None, pooling=None, classes=1000)

Размер входных данных по умолчанию для этой модели – 224x224.

Аргументы:

– include_top: Подключение 3 полно связных слоев верхнего уровня сети

– weights: значения весов *None* (случайная инициализация) или *'imagenet'* (предобученная на наборе данных ImageNet).

– input_shape: опциональная форма кортежа, указывается только если *include_top – False.*

– input_shape: optional shape tuple, only to be specified if *include_top* is False (иначе форма ввода должна быть (224, 224, 3) (с форматом данных

'channels_last') или (3, 224, 224) (с форматом данных *'channels_first'*). Он должен иметь ровно 3 входных канала, а ширина и высота должны быть не меньше 48. Например. (200, 200, 3) будет одним из допустимых значений.

– pooling: опциональный аргумент извлечения признаков, когда аргумент include_top - False.

– None означает, что выходом модели будет 4D тензором на выходе последнего сверточного слоя.

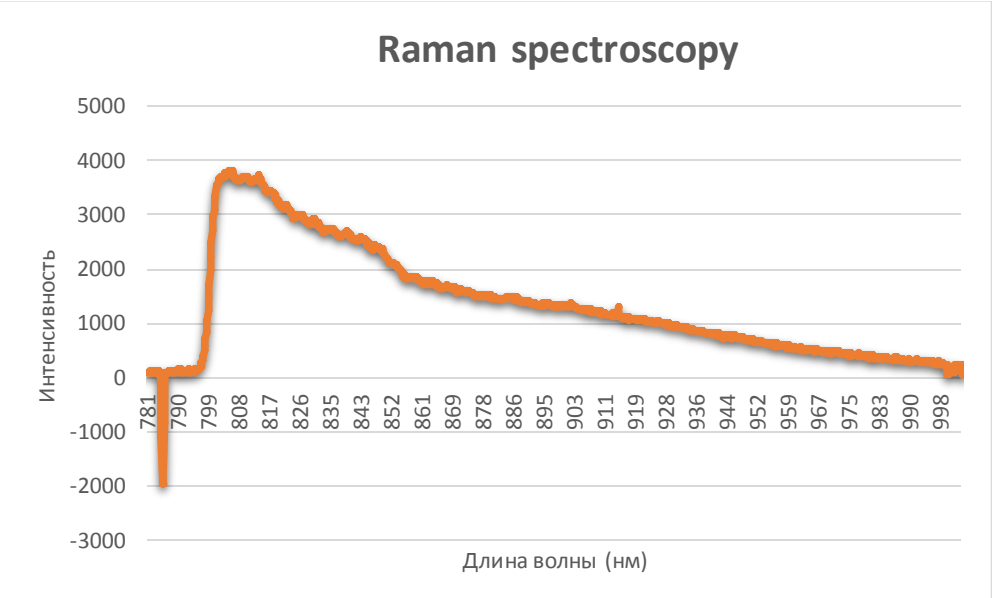
– avg означает что для сети будет применять операция average pooling

– max означает что для сети будет применять операция max pooling

– classes: необязательное значение количества классов классифицируемых изображений если, include_top имеет значение True и не если аргумент weights не указан. [16]

2.7 Препроцессинг и создание различных наборов данных на основе спектров комбинационного рассеивания

Сырые данные представляют собой матрицу размерность 2x1024, состоящие из значений длин волны и соответствующих им значений интенсивности.



780.64	83.67
780.87	101.33
781.11	80.33
781.34	118.67
781.58	92.33
781,81	111
782,05	90
782,28	98
…	
1001,71	129
1001,91	235
1002.10	229.33
1002.29	228.00
1002.48	216.67
1002.68	217.67
1002.87	193.33
1003.06	69.67

Рисунок 2.17 – Пример сырых данных и их визуализация

Для их обработки предобученными сверхточными нейронными сетями необходимо представить их в виде каскада горизонтально расположенных частей спектра, при этом сохраняя пространственную зависимость значений интенсивности от длины волны. Полученные матрицы обладает размерностью 44x44.

В таблице 2.1 показано что минимальными размерами входных тензоров сверточных сетей являются, 48x48 для VGG16, 48x48 для VGG19 и 71x71 для Xception.

Изменение размера входных данных приведет к появлению шума и артефактов, влияющих на точность нейронной сети. Если коэффициент масштабирования для сетей VGG составляет всего 1.19, то для архитектуры Xception необходимо увеличить входные данные примерно в 2.6038 раз.

Для увеличения размера входных данных представленной в виде квадратной матрицы использовался метод двумерной сплайн-аппроксимации по прямоугольной сетке (bivariate spline approximation over a rectangular mesh). Данный метод может использоваться как для аппроксимации, так и для интерполяции данных. Суть интерполяции заключается в использовании имеющихся данных для получения ожидаемых значений в неизвестных точках.

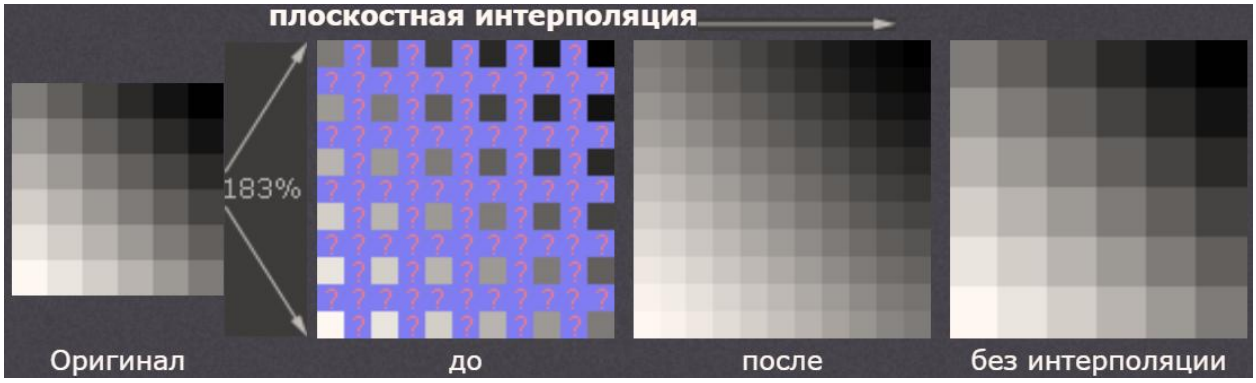


Рисунок 2.18 – Визуализация работы алгоритма интерполяции.

Альтернативой этому методу является интерполяция спектра перед его представлением в матричной форме. Для этого использовался метод линейной интерполяции.

На основе этой методики будет создано два набора данных для обучения нейронных сетей с предобученными сверточными слоями VGG16/19 и Xception.

Для эффективного использования сверточной нейронной сети Xception воспользуемся методом дублирования информативных частей спектра. Данный способ, помимо уменьшения коэффициента масштабирования, позволит нейронной сети акцентировать внимание на отдельных частях спектра, увеличивая значимость отдельных ее областей при дифференциации патологии.

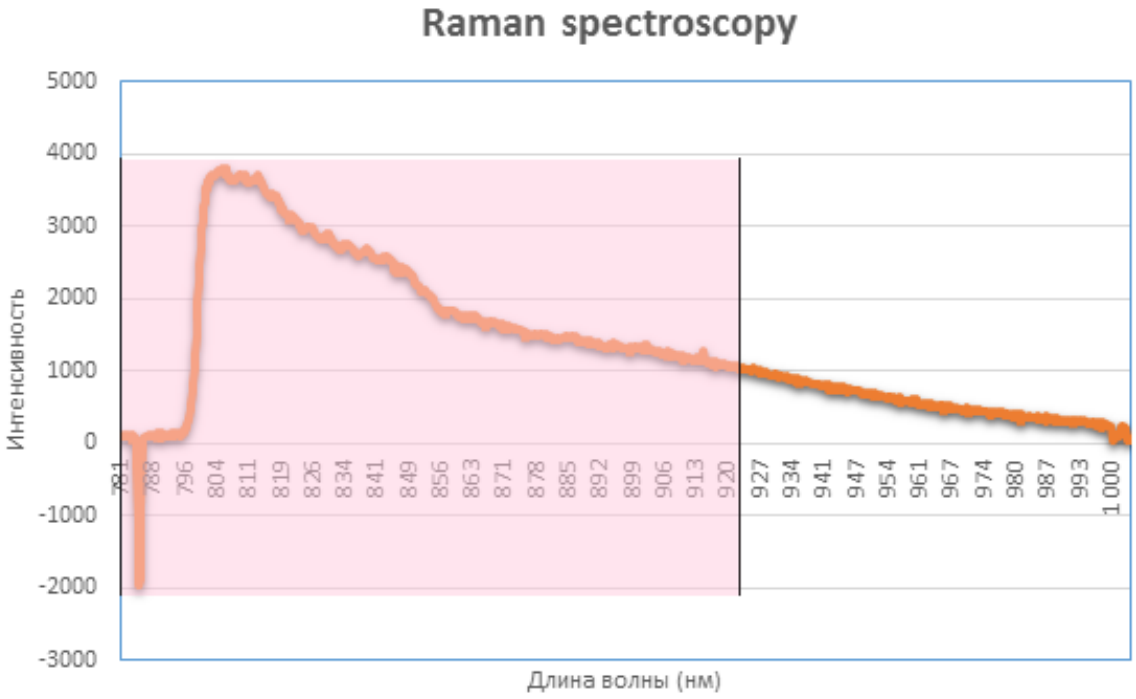


Рисунок 2.19 – Визуализация дублируемого участка спектра

Спектроскопия комбинационного рассеяния отличается спектром с ярко выраженными пиками, которые соответствуют наличию определенного вещества в исследуемой среде. В качестве дублируемого участка спектра возьмем область с длиной волны 781 до 923 нм. На ней зарегистрировано 648 значений интенсивности. В совокупности со значениями длин волн входные данные можно представить в виде квадратной матрицы размерностью 36x36. Для использования в сверточной сети Xception продублируем матрицу 4 раза.

Так же воспользуемся подходом дублирования информативных участков спектра для создания набора данных для сети VGG16/19. Длинна волны

дублируемого участка составляет 781 до 909 нм с зарегистрировано 576 значениями интенсивности. Для использования в сверточных сетях VGG16/19 достаточно продублировать матрицу 2 раза.

Набор данных представляет собой список списков numpy массивов определенного формата. Индексы списка соответствую данному списку

Спектроскопия меланомы 0, спектроскопия образца кожи – 1, спектроскопия базалиомы – 2, спектроскопия образца кожи – 3, спектроскопия невуса – 4, спектроскопия образца кожи – 5, Спектроскопия кератомы – 6, спектроскопия образца кожи – 7, Спектроскопия других паталогий – 8, спектроскопия образца кожи – 9, спектроскопия образцов, не имеющих диагноза - 10.

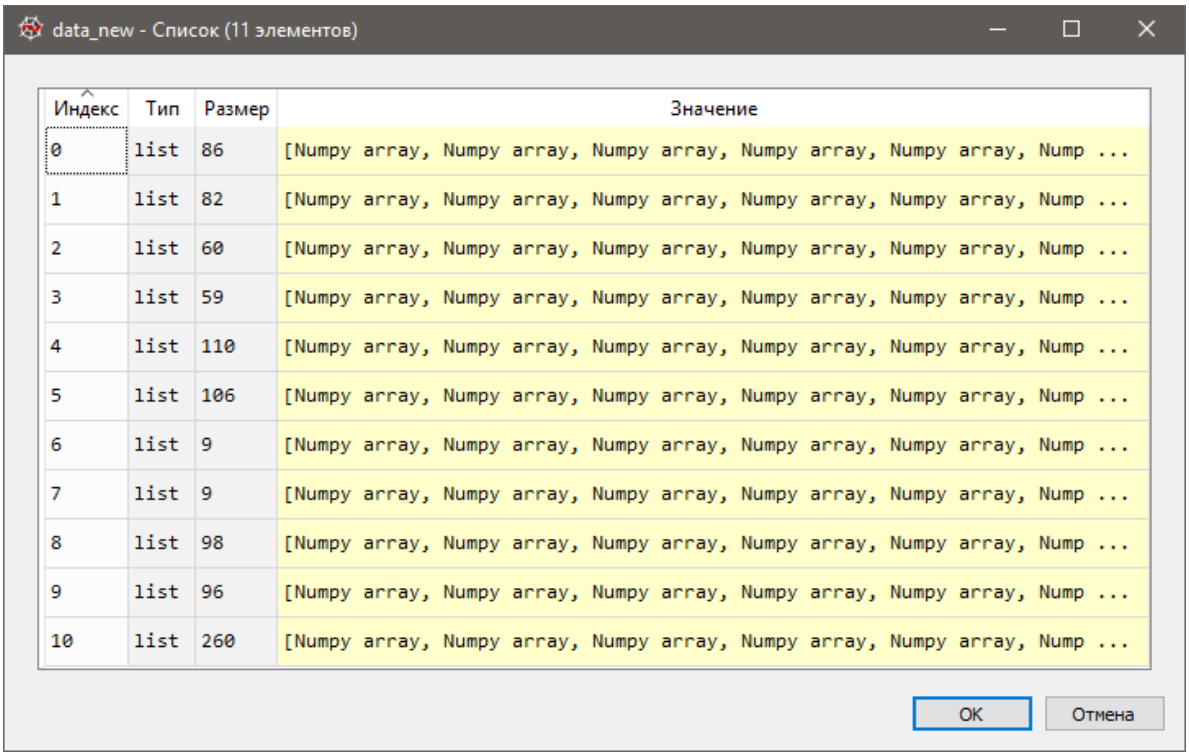


Рисунок 2.20 – Общий вид наборы данных

Всего было создано 5 наборов данных:

- VGG16/19_dataSet_48x48_968x2;
- VGG16/19_dataSet_48x48(24x48x2)_2x576x2;
- VGG16/19_dataSet_48x48_1152x2;
- Xception _dataSet_72x72(32x32x4)_1x1024;

– Xception_dataSet_72x72(36x36x4)_2x648.

Название содержит краткую информацию о содержащихся в наборе данных. Информация по всем созданным наборам представлена в таблице 2.2.

Таблица 2.2 Наборы данных, основанные на спектрах комбинационного рассеивания.

Входные данные	Размерность	Интерполяция данные	Коэффициент масштабирования	Квадратное представление	Сеть
интенсивность + длина волны	2x968	48x48	1.19	48x48	VGG16/19
интенсивность+ длина волны	2x1024	1152x2	1.125	48x48	VGG16/19
интенсивность+ длина волны	2x576x2	без интерполяции	1	24x48x2	VGG16/19
интенсивность+ длина волны	2x648	без интерполяции	1	36x36x4	Xception
интенсивность	1x1024	1296	1.27	36x36x4	Xception

Созданные наборы данных сохранены с помощь модуля pickle, в формате pickle. Pickle – это модуль реализующий двоичные протоколы для сериализации и десериализации объекта Python. «Pickling» - это процесс, в соответствии с которым иерархия объектов Python преобразуется в поток байтов, а «unpickling» - это обратная операция, в результате которой поток байтов (из двоичного файла или байтоподобного объекта) преобразуется обратно в объекты языка Python. Каждый из наборов данных доступен для загрузки в программу, с помощью функции open.

Листинг кода создания набора данных предоставленн в приложении Б.

Во второй главе был проведен анализ спектров АФ. Была создана обучающая выборка, состоящая из векторов характеристик спектров полученных с помощью стохастических методик дифференциации патологии

Кроме того, во второй главе был проведен анализ различных нейростевых архитектур, в ходе которого были выбраны сети VGG16/19, Xception, а также созданы наборы данных используемые для их дальнейшего обучения.

3 ТЕСТИРОВАНИЕ НЕЙРОСЕТЕВЫХ АРХИТЕКТУР

3.1 Дифференциация результатов автофлуоресценции.

Для решения задачи дифференциации кожи от патологии была выбрана нейронная сеть, основанная на многослойном персептроне и состоящая из 1 скрытого слоя.

Имея в наличие 42 усредненных образца, из которых 21 образец кожи и 21 патологии, была спроектирована и обучена нейронная сеть.

При обучении нейронной сеть использовался алгоритм Левенберга-Маркварта при котором обучающая выборка делится в случайном порядке на три набора: 60% используются для обучения (Training), 20% для проверки достоверности полученных результатов и избежание переобучения (Validation), 20% используются для независимого испытания сети(Test).

Процесс обучения нейронной сети прекратился при достижении среднеквадратичной ошибки значения в 0.19 процесс изображен на Рисуноке 3.1. Это произошло на 13 эпохе обучения нейронной сети. На последующих эпохах значение ошибки лишь увеличивалось.

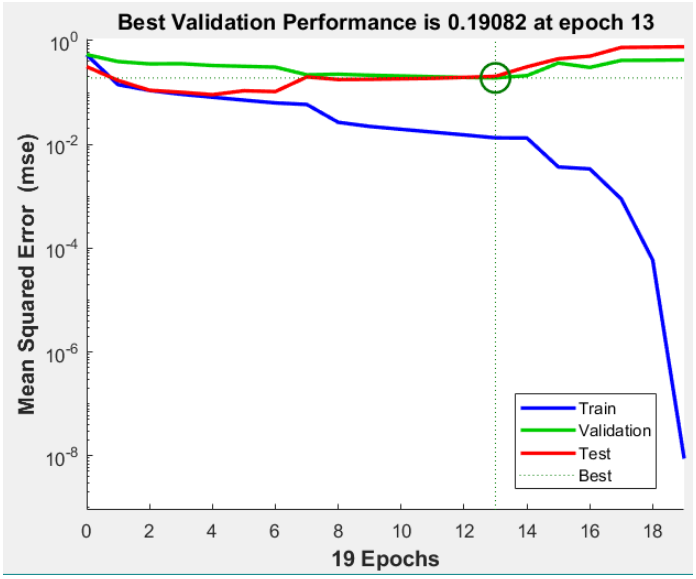


Рисунок 3.1 – График обучения нейронной сети

По завершению обучения была построена регрессионная модель выходных результатов нейронной сети, изображенная на рисунке 3.1.

Коэффициент корреляции(R) показывает эффективность работы нейросетевого алгоритма при работе с входными данными.

Несмотря на эффективную работу нейронной сети на обучающей выборке(R=0.97) ее работа с тестовой выборкой оказалась неудовлетворительной [5].

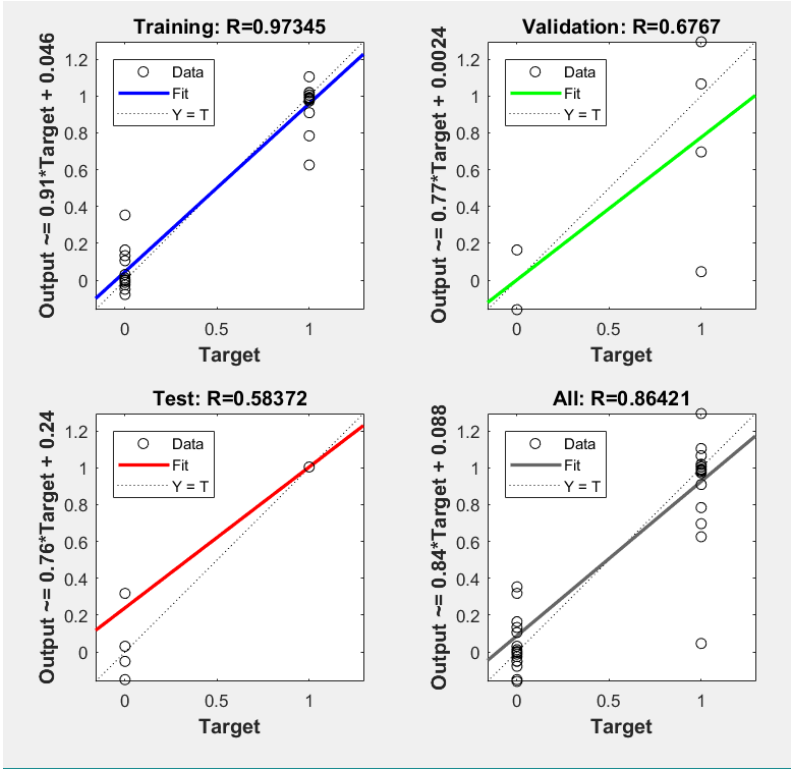


Рисунок 3.2 – Регрессионная модель нейронной сети

Это является следствием использования малой размерности обучающей выборки. Для подтверждения, сеть, была обучена повторно на неусреднённых данных спектров АФ. 126 образцов из которых 63 образца кожи и 63 образца патологии.

Значение коэффициента корреляции при проверке на обучающей выборке уменьшилось, но при этом наблюдается значительное повышение значения классификации кожи и патологии при работе с тестовой выборкой. [5]

В медицине для характеристики информативности диагностических методов исследования служат объективные параметры, именуемые операционными характеристиками исследования (теста). В частности, нас интересуют такие характеристики как чувствительность и специфичность.

Чувствительность (Se) – это способность диагностического метода давать правильный результат, который определяется как доля истинно положительных результатов среди всех проведенных тестов.

$$Se = \frac{TP}{TP+FN} \times 100\%, \quad (3.1)$$

где TP – количество истинно отрицательных результатов;

FP – количество ложноположительных результатов.

Специфичность (Sp) – это способность диагностического метода не давать при отсутствии заболевания ложноположительных результатов, который определяется как доля истинно отрицательных результатов среди здоровых лиц в группе исследуемых. Данный показатель определяется по формулам:

$$Sp = \frac{TN}{TN+FP} \times 100\%, \quad (3.2)$$

где TH – количество истинно отрицательных результатов

FP – количество ложноположительных результатов

В нашем случае значения чувствительности обученной нейронной сети равно 80%, в то время как специфичность всего 40%.

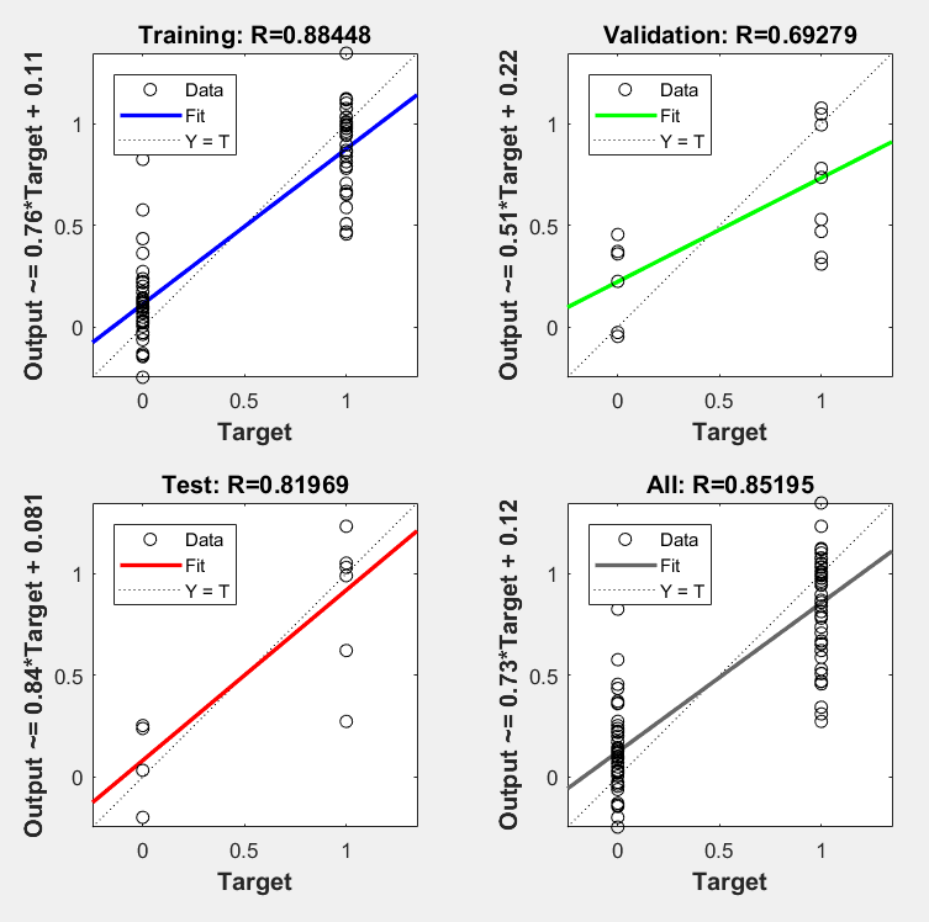


Рисунок 3.3 – Регрессионная модель нейронной сети

Коэффициент корреляции работы на тестовой выборке возрос с ростом обучающей выборки, но классифицирующие признаки, полученные при обработке не усреднённых данных, могут иметь отклонение от фактического результата, тем самым увеличивая ошибку определение патологии до загрузки обучающей выборки в сеть.

3.2 Тестирование сверточных архитектур

Формально, задача дифференциации патологий была поделена на 4 части. Сверточных нейронные сети обучались для решения задач бинарной классификации:

1. Меланома/базалиома (м/б).
2. Меланома/невус (м/н).
3. Базалиома/невус (б/н).

4. Доброкачественные/злокачественные опухоли (д/з).

Из подгруженных наборов данных извлекались выборки необходимые для обучения нейронной сети. В соответствии с задачей бинарной классификации им присваивались метки класса 0, 1, представленные в виде массива. Для чистоты эксперимента данные перемешивались по функции представленной на рисунке 3.4, а за тем делились на 2 группы Test и Training в соотношении 25%, 75% соответственно. Листинг кода обучения нейронной сети предоставлен в приложении А.

```
outputData=np.asarray(
            [["1","0"]for  _ in
range(len(data_new[0])+len(data_new[2]))]+
            [["0","1"]for  _ in range(len(data_new[4]))]
            )
outputData=outputData.astype('float32')

inputData=np.vstack((
                np.asarray(data_new[0]),
                np.asarray(data_new[2]),
                np.asarray(data_new[4]),
                ))
randomize  = np.arange(len(outputData))
np.random.shuffle(randomize)
inputData  = inputData[randomize]
outputData = outputData[randomize]

X_train,X_test = np . array_split ( inputData ,  2 )
buff,X_test = np . array_split ( X_test ,  2 )
X_train=np.vstack((X_train,buff))

Y_train,Y_test = np . array_split (outputData ,  2 )
buff,Y_test = np . array_split ( Y_test ,  2 )
Y_train=np.vstack((Y_train,buff))
```

Рисунок 3.4 – Листинг кода формирования выборки

Результаты тестирования предобученных нейронных сетей представлены в таблицах 3.1, 3.2, 3.3.

Таблица 3.1 – Результаты тестирования VGG16

Набор данных	Задача	Метод масштабирования	Количество	Точность Test, %	Точность Training, %
dataSet_48x48_968x2	м/б	Интерполяция	86/60	63	57
dataSet_48x48_968x2	м/н	Интерполяция	86/110	55	58
dataSet_48x48_968x2	н/б	Интерполяция	110/60	60	67
dataSet_48x48_968x2	д/ з	Интерполяция	86+60/110	54	57
dataSet_48x48(24x48x2)_2x576x2	м/б	Интерполяция +Дублирование спектра	86/60	58,33	59,09
dataSet_48x48(24x48x2)_2x576x2	м/н	Интерполяция +Дублирование спектра	86/110	52	59
dataSet_48x48(24x48x2)_2x576x2	н/б	Интерполяция +Дублирование спектра	110/60	67	69
dataSet_48x48(24x48x2)_2x576x2	д/ з	Интерполяция +Дублирование спектра	86+60/110	46	54
dataSet_48x48_1152x2e	м/б	Дублирование спектра	86/60	61	58
dataSet_48x48_1152x2	м/н	Дублирование спектра	86/110	54	56
dataSet_48x48_1152x2	н/б	Дублирование спектра	110/60	62	65
dataSet_48x48_1152x2	д/ з	Дублирование спектра	86+60/110	49	50

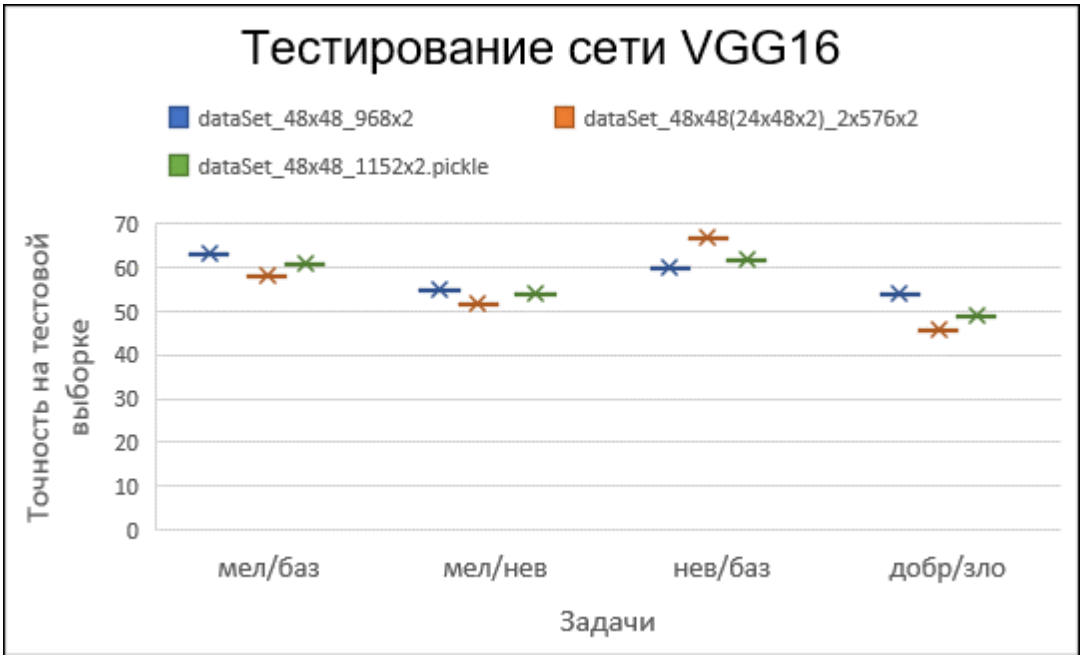


Рисунок 3.5 – Сравнительная гистограмма результатов обучения сети Xception на разных тестовых выборках

Таблица 3.2 – Результаты тестирования VGG19

Набор данных	Задача	Метод масштабирования	Количество	Точность Test, %	Точность Training, %
dataSet_48x48_968x2	м/б	Интерполяция	86/60	64	60
dataSet_48x48_968x2	м/н	Интерполяция	86/110	61.22	53.48
dataSet_48x48_968x2	н/б	Интерполяция	110/60	69	69
dataSet_48x48_968x2	д/з	Интерполяция	86+60/110	58	57
dataSet_48x48(24x48x2)_2x576x2	м/б	Интерп.+ Дублирование	86/60	52,78	61,82
dataSet_48x48(24x48x2)_2x576x2	м/н	Интерп.+ Дублирование	86/110	55,10	56,12
dataSet_48x48(24x48x2)_2x576x2	н/б	Интерп.+ Дублирование	110/60	59	66
dataSet_48x48(24x48x2)_2x576x2	д/ з	Интерп.+ Дублирование	86+60/110	51	56
dataSet_48x48_1152x2	м/б	Дублирование	86/60	58	62
dataSet_48x48_1152x2	м/н	Дублирование	86/110	60	55
dataSet_48x48_1152x2	н/б	Дублирование	110/60	64	65
dataSet_48x48_1152x2	д/з	Дублирование	86+60/110	53,46	55,77

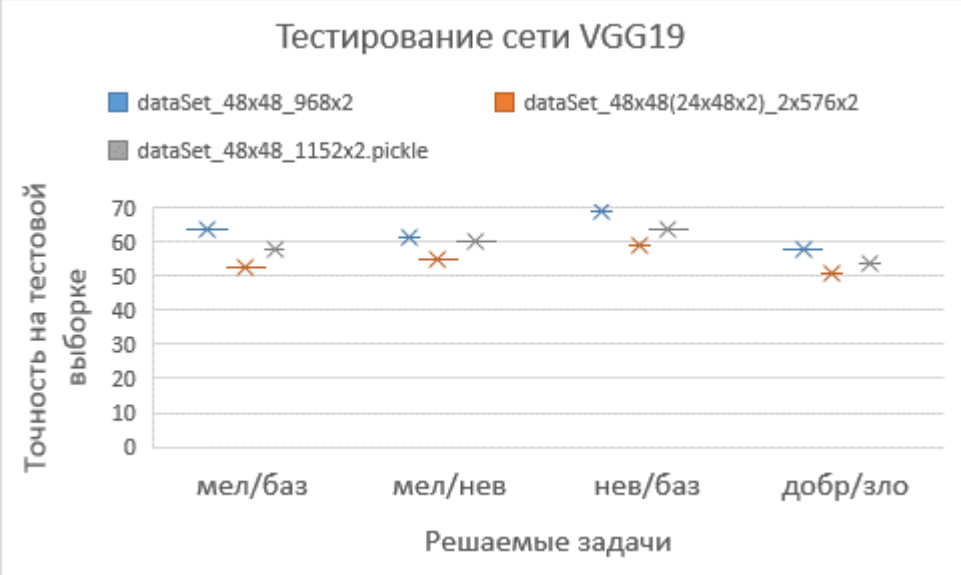


Рисунок 3.6 – Сравнительная гистограмма результатов обучения сети Xception на разных тестовых выборках

Таблица 3.3 – Результаты тестирования Xception

Набор данных	Задача	Метод масштабирования	Размер выборки	Точность Test, %	Точность Training, %
dataSet_72x72(36x36x4)_2x648	м/б	Дублирование спектра	86/60	54	61
dataSet_72x72(36x36x4)_2x648	м/н	Дублирование спектра	86/110	54	60
dataSet_72x72(36x36x4)_2x648	н/б	Дублирование спектра	110/60	63	69

Продолжение таблицы 3.3

Набор данных	Задача	Метод масштабирования	Размер выборки	Точность Test, %	Точность Training, %
dataSet_72x72(36x36x4)_2x648	Д/ з	Дублирование спектра	86+60/110	63	65
dataSet_72x72(32x32x4)_1x1024	м/б	Интерпаляция +Дублирование спектра	86/60	55.56%	60.45%
dataSet_72x72(32x32x4)_1x1024	н/б	Интерпаляция +Дублирование спектра	110/60	71,41	72
dataSet_72x72(32x32x4)_1x1024	Д/ з	Интерпаляция +Дублирование спектра	86+60/110	69,53	65.10

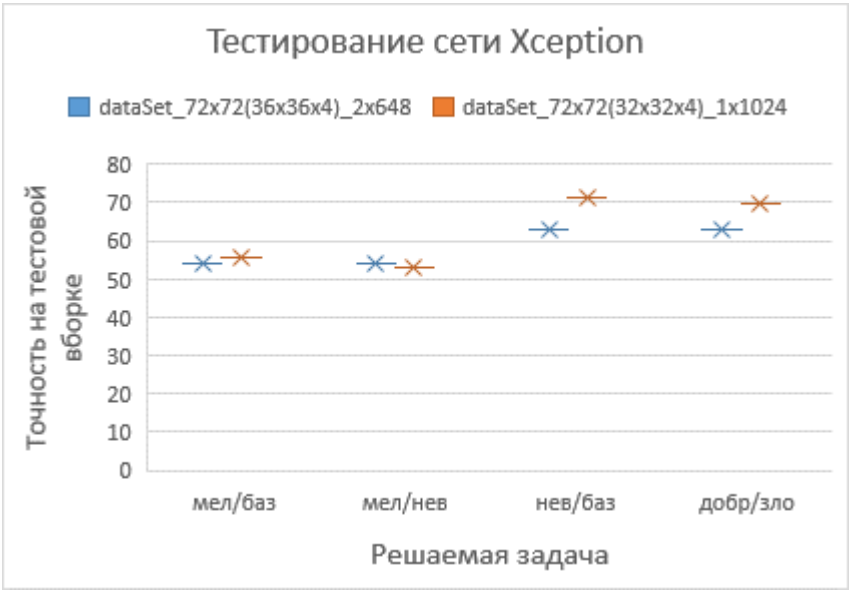


Рисунок 3.7 – Сравнительная гистограмма результатов обучения сети Xception на разных тестовых выборках

В среднем наибольшую точность показала модель нейронной сети основанная на сверточных слоях сети Xception и обученная на наборе данных формата 72x72(32x32x4)_1x1024.

Используя модель основывающуюся на сети Xception, веса полученные при ее обучении на наборе данных 72x72(32x32x4)_1x1024, проведем тонкую настрой для каждой задачи бинарной классификации.

Тонкая настройка сети (fine-tuning) состоит в разблокировании нескольких верхних слоев базовой модели, используемых для извлечения признаков, и совместного обучения совместно с вновь добавленной частью сети

(в данном случае, полносвязного классификатора). Этот метод называется тонкой настройкой потому, что оно корректирует абстрактное представления предобученной сети, делая его более актуальными для текущей проблемы.

Тонкую настройку можно проводить только после обучения нового классификатор. Если классификатор еще не обучен, тогда сигнал ошибки, распространяющийся по сети во время тренировки, будет слишком большим, и данные, ранее изученные с помощью тонкой настройки, будут уничтожены.

Алгоритм тонкой настройки включает в себя следующие действия:

1. Добавьте свою собственный классификатор поверх уже обученной базовой сети.

2. Блокировка обучения базовой сеть.

3. Обучение добавленной части сети

4. Разблокировка несколько слоев ранее блокированной модели

5. Совместная тренировка базовую сеть совместно с добавленной частью.

На рисунке 3.8 показан пример диагностического вывода в Keras, где:

– loss – значение ошибки на обучающей выборке

– acc - точность на обучающей выборке

– vall_loss – значение ошибки на валидационных(проверочных) данных

– val_acc – точность на валидационных(проверочных) данных

Значение ошибки вычисляется как значение бинарной кросс-энтропии (binary cross-entropy):

$$BinariCrossEntropy = - \quad (y_{true} log\ y_{pred} + (1 - y_{true})\log(1 - y_{pred})), \quad (3.3)$$

где $BinariCrossEntropy$ – бинарной кросс-энтропии;

y_{true} – фактический результат;

y_{pred} – предполагаемый.

```
Epoch 1/15
102/102 [==============================] - 44s 434ms/step - loss: 0.5912 - acc: 0.7647 - val_loss: 0.6028 - val_acc:
0.7500
Epoch 2/15
102/102 [==============================] - 10s 96ms/step - loss: 0.5375 - acc: 0.7598 - val_loss: 0.5730 - val_acc:
0.8077
Epoch 3/15
102/102 [==============================] - 10s 96ms/step - loss: 0.5257 - acc: 0.7892 - val_loss: 0.5527 - val_acc:
0.8077
Epoch 4/15
102/102 [==============================] - 10s 96ms/step - loss: 0.4958 - acc: 0.8088 - val_loss: 0.5214 - val_acc:
0.7885
Epoch 5/15
102/102 [==============================] - 10s 96ms/step - loss: 0.4727 - acc: 0.7990 - val_loss: 0.5161 - val_acc:
0.8462
Epoch 6/15
102/102 [==============================] - 10s 96ms/step - loss: 0.4611 - acc: 0.8235 - val_loss: 0.5070 - val_acc:
0.8654
Epoch 7/15
102/102 [==============================] - 10s 96ms/step - loss: 0.5033 - acc: 0.8088 - val_loss: 0.4843 - val_acc:
0.8654
Epoch 8/15
102/102 [==============================] - 10s 98ms/step - loss: 0.4787 - acc: 0.8137 - val_loss: 0.4756 - val_acc:
0.8462
Epoch 9/15
102/102 [==============================] - 10s 96ms/step - loss: 0.5076 - acc: 0.7598 - val_loss: 0.4626 - val_acc:
0.8462
Epoch 10/15
102/102 [==============================] - 10s 97ms/step - loss: 0.4673 - acc: 0.7990 - val_loss: 0.4558 - val_acc:
0.8462
Epoch 11/15
102/102 [==============================] - 10s 96ms/step - loss: 0.3937 - acc: 0.8676 - val_loss: 0.4491 - val_acc:
0.8462
Epoch 12/15
102/102 [==============================] - 10s 97ms/step - loss: 0.4008 - acc: 0.8725 - val_loss: 0.4449 - val_acc:
0.8462
Epoch 13/15
102/102 [==============================] - 10s 101ms/step - loss: 0.3795 - acc: 0.8922 - val_loss: 0.4399 - val_acc:
0.8462
Epoch 14/15
102/102 [==============================] - 10s 102ms/step - loss: 0.4041 - acc: 0.8578 - val_loss: 0.4329 - val_acc:
0.8269
Epoch 15/15
102/102 [==============================] - 10s 99ms/step - loss: 0.4601 - acc: 0.8284 - val_loss: 0.4382 - val_acc:
0.8462
```

Рисунок 3.8 – Диагностический вывод тонкой настройке модели Xception на задаче невус/базалиома

Таблица 3.4 – Результаты тонкой настройки

Название набора данных	Задача	Точность на тестовой выборке	Точность на обучающей выборке	Чувствительность, %	Специфичность, %
dataSet_72x72(32x32x4)_1x1024	н/б	83.33	89.84	73.33	98.15
dataSet_72x72(32x32x4)_1x1024	н/м	59.32	68.12	40.48	95.74
dataSet_72x72(32x32x4)_1x1024	м/б	65.31	75.05	90.43	57,16
dataSet_72x72(32x32x4)_1x1024	з/д	64.84	76.82	85.91	54,21

Все весовые коэффициенты сохранены в форма h5, модели нейронных сетей сохранены в формате json и доступны для дальнейшей работы.

На основе нейронных сетей, обученных решать задачи бинарной классификации был разработан алгоритм принятия взвешенного решения. Блок-схема которого представлена на рисунке 3.9.

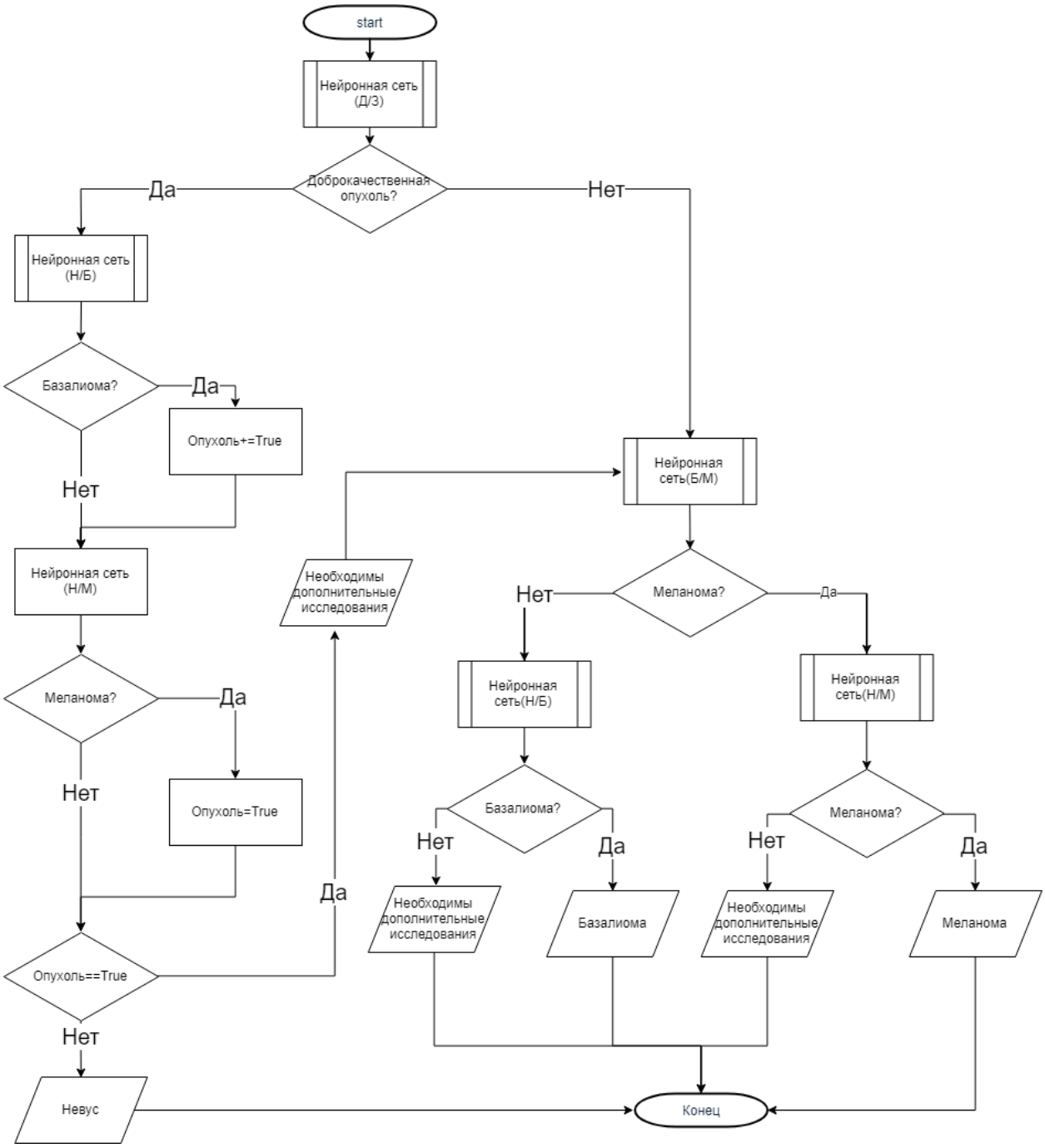


Рисунок 3.9 – Алгоритм принятия взвешенного решения

Листинг алгоритма, представленного на рисунке 3.9 дается в приложении В.

Точность работы данного алгоритма составляет 70%, специфичность – 53%, чувствительность – 93,64.

На выходе нейронная сеть предоставляет вероятностный вывод принадлежности входного значения к классам, на которых она обучалась. Вопросы разработки алгоритма основанного на вероятностном выводе нейронных сетей а так же улучшение существующего будут рассматриваться при работе над магистерской работой.

В третьей главе проводится обучения нейронных сетей с помощью подготовленных наборов данных. Для сверточной архитектуры применяются методики переноса обучения, а также тонкой настройки. Точность дифференциации патологий: Невус/Базалиома – 83.33, Невус/Меланома – 63, Меланома/Базалиома – 65.31. Так же для увеличения точности дифференциации патологий и обобщения нейронных сетей обученных для решения задач бинарной классификации разрабатывается алгоритм принятия взвешенного решения. Точность работы данного алгоритма составляет 70%, специфичность – 53%, чувствительность – 93,64.

ЗАКЛЮЧЕНИЕ

В ходе выпускной квалификационной работы была описана актуальность рассматриваемой темы, определены объект и предмет выпускной квалификационной, поставлена цель и выявлены задачи. Так же, было рассмотрено два различных подхода к дифференциации патологи. Первый подход основан на выявлении признаков спектров автофлуоресценции с помощью стохастических методик дифференциации патологий, второй же – это слепой метод дифференциации спектров комбинационного рассеивания.

На начальном этапе было проведено обучения нейронной сети на спектрах автофлуоресценции. Основой для исследования стали стохастические методики дифференциации патологий. При данном подходе входными данными в нейронную сеть был вектор признаков размерность 1x5. Для решения данной задачи подходила нейронная сеть прямого распространения, состоящая из 1 скрытого слоя, представляющая собой классификатор – входными данными в которой являются классифицирующие признаки, вычисленные с использованием стохастических методик.

Но при данном подходе существует вероятность упустить скрытые зависимости, в результате чего было принято решение перейти к архитектуре, объединяющей в себе поиск классифицирующих признаков и их последующую классификацию – сверточные нейронные сети. В качестве входных данных стали рассматриваться спектры комбинационного рассеивания.

Эффективным решением в этой области может стать использование предобученных нейронных сетей: Xception, VGG16, VGG19. В частности, использование их сверточных слоев, совместно с новым классификатором.

Используя методы интерполяции и дублирования спектров было составлено пять наборов данных по которым в дальнейшем обучались сверточные нейронные сети. После обучения сети с лучшими результатами подверглись тонкой настройки. Точность дифференциации патологий: Невус/Базалиома – 83.33, Невус/Меланома – 63, Меланома/Базалиома – 65.31.

Для объединения всех обученных нейронных сетей был составлен алгоритм принятия взвешенного решения, основанный на логическом. Точность работы данного алгоритма составляет 70%, специфичность – 53%, чувствительность – 93,64.

Апробация данной работы была представлена в финале Всероссийского студенческого фестиваля «СТАРТАП», который проводилась в Тольяттинском государственном университете с 4 по 7 октября 2017 года в очной форме.

Опубликована на I Всероссийской научной конференции «Информационные технологии в моделировании и управлении: подходы, методы, решения», которая проводилась в Тольяттинском государственном университете с 12 по 14 декабря 2017 года в заочной форме.

И так же работа была представлена и опубликована на научной конференции «Студенческие дни науки ТГУ», которая проводилась в Тольяттинском государственном университете с 23 по 25 апреля 2018 года в очной форме.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

Научная и методическая литература

1. Быков В.Л. Цитология и общая гистология: учеб. пособие / В.Л. Быков – СПб.: СОТИС, 2015. – 520 с.

2. Демидов Л.В. Меланома кожи: стадирование, диагностика и лечение: науч. работа / Л.В Демидов – Москва, 2013. – 658-665 с

3. Дьяконов В.П., Круглов В.В. MATLAB 6.5 SP1/7/7 SP1/7 SP2 +Simulink 5/6. Инструменты искусственного интеллекта и биоинформатики./ В.П. Дьяконов, В.В Круглов – Москва: СО-ЛОН-ПРЕСС, 2011. – 456 с.

4. Медведев В.С., Потемкин В.Г. Нейронные сети. MATLAB 6 / В.С. Медведев, В.Г. Потемкин – Москва: ДИАЛОГ-МИФИ, 2013. – 496 с

5. Сержантов К.А. Автофлуоресцентный анализ кожных патологий, с использованием нейросетевого алгоритма: науч. работа /К.А. Сержантов, М.Г. Лисовская, В.П. Захаров, А.А. Морятов, С.В. Козлов / сборник статей «Информационные технологии в моделировании и управлении: подходы, методы, решения», 2017. – 256-263 с.

6. Тучин. В.В. Оптическая биомедицинская диагностика: учеб. пособие / В.В. Тучин - Москва: ФИЗМАТЛИТ, 2015. – 13 с.

7. Тюляндин С.А., Моисеенко В.М. Практическая онкология: учеб. пособие/ С.А. Тюляндин, В.М. Моисеенко – СПб.: Центр ТОММ, 2014. - 23 с.

8. Beazley, D. Python_Cookbook_3rd_Edition_2015 / David Beazley, Brain K. Jones: O'Reilly Media - June 2016. – 706 p.

9. Buduma N. Fundamentals of Deep Learning. Designing Next-Generation Machine Intelligence Algorithms / Nikhil Buduma: O'Reilly Media - June 2017. – 298 p.

10. Chollet, F. Deep Learning with Python / Francois Chollet: – Manning Publications - December 22, 2017. – 386 p.

11. Kuhlman, Dave. A Python Book: Beginning Python, Advanced Python, and Python Exercises. / Kuhlman, Dave: Archived from the original on 23 June 2012. – 673 p

12. Lutz, M. Learning Python, 5th Edition Fifth Edition / Mark Lutz: O'Reilly Media – June 2015. – 1648 p.

13. Shukla N. Manning Early Access Program Machine Learning with TensorFlow/ Nishant Shukla Manning Publications 2017. – 244 p

14. M. Vrakova, I. Bratchenko, V. Zakharov Autofluorescence analysis of skin cancer pathologies in the visible region / Proceedings of Saratov Fall Meeting - SFM'14, International Symposium Optics and Biophotonics-IV September Saratov, 2014. – 22-26 p

Электронные ресурсы

15. Andrei Sozykin [Электронный ресурс]. – Режим доступа: https://www.asozykin.ru/deep_learning

16. Keras Documentation. keras.io. [Электронный ресурс]. – Режим доступа: https://keras.io/applications/

17. Matlab Documentation [Электронный ресурс]. – Режим доступа: https://nl.mathworks.com/help/matlab/

18. Python Documentation [Электронный ресурс]. – Режим доступа: https://www.python.org/doc/

19. Python Software Foundation. Retrieved [Электронный ресурс]. – Режим доступа: https://en.wikipedia.org/wiki/Python_(programming_language)

20. Welcome to Neural Network Toolbox [Электронный ресурс] – Режим доступа: http://matlab.exponenta.com/neuralnetwork/index.php

ПРИЛОЖЕНИЕ А

Листинг кода обучения нейронной сети

```
import win32com.client,pickle
import numpy as np
with open('dataSet_72x72(32x32x4)_1x1024.pickle', 'rb') as f:
    data_new = pickle.load(f)

outputData=np.asarray(
            [["1","0"]for _ in range(len(data_new[0]))]+
            [["0","1"]for _ in range(len(data_new[2]))]
            )
outputData=outputData.astype('float32')

inputData=np.vstack((
                np.asarray(data_new[0]),
                np.asarray(data_new[2]),
                ))
randomize = np.arange(len(outputData))
np.random.shuffle(randomize)
inputData = inputData[randomize]
outputData = outputData[randomize]

X_train,X_test = np . array_split ( inputData , 2 )
buff,X_test = np . array_split ( X_test , 2 )
X_train=np.vstack((X_train,buff))

Y_train,Y_test = np . array_split (outputData , 2 )
buff,Y_test = np . array_split ( Y_test , 2 )
Y_train=np.vstack((Y_train,buff))

from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Activation, Dropout, Flatten, Dense
from keras.applications import Xception
from keras.optimizers import Adam
from keras.layers import Conv2D, MaxPooling2D, Conv2DTranspose
import numpy as np
xception_net = Xception(weights='imagenet', include_top=False, input_shape=(72,
72, 3), pooling=None)
xception_net.trainable = False

model = Sequential()
model.add(xception_net)
```

```
model.add(Flatten())
model.add(Dense(256))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(2))
model.add(Activation('sigmoid'))
model.compile(loss='binary_crossentropy',
              optimizer=Adam(lr=1e-5),
              metrics=['accuracy'])
# Обучаем сеть
model.fit(X_train,  Y_train, batch_size=15, epochs=15, validation_split=0.2,
verbose=1)
scores = model.evaluate(X_test,  Y_test, verbose=0)
print("Точность работы на тестовых данных: %.2f%%" % (scores[1]*100))
scores = model.evaluate(X_train,  Y_train, verbose=0)
print("Точность работы на обучающих данных: %.2f%%" % (scores[1]*100))

xception_net.trainable  = True
trainable=  False
for layer  in  xception_net.layers:
    if layer.name  == 'block12_sepconv1_act':
        trainable=True
    layer.trainable  = trainable
xception_net.summary()
model.compile(loss='binary_crossentropy',
              optimizer=Adam(lr=1e-5),
              metrics=['accuracy'])
model.fit(X_train,  Y_train, batch_size=15, epochs=15, validation_split=0.2,
verbose=1)
# Оцениваем качество обучения сети на тестовых данных
scores = model.evaluate(X_test,  Y_test, verbose=0)
print("Точность работы на тестовых данных: %.2f%%" % (scores[1]*100))
scores = model.evaluate(X_train,  Y_train, verbose=0)
print("Точность работы на обучающих данных: %.2f%%" % (scores[1]*100))
#сохранение
model_json=model.to_json()
json_file=open("fine_tuning_dataSet_72x72(32x32x4)_1x1024.pickle(MvsB).json","
w")
json_file.write(model_json)
json_file.close()
model.save_weights("fine_tuning_dataSet_72x72(32x32x4)_1x1024.pickle(MvsB).h
5")
```

ПРИЛОЖЕНИЕ Б

Листинг кода создания наборов данных

```
# -*- coding: utf-8 -*-
"""
Created on Wed Mar 16 03:22:32  2018
@author: Kirill
"""
import win32com.client,  os, re, pickle
import numpy as np
import shutil
def isint(s):
    try:
        int(s)
        return  False
    except ValueError:
        return True

def CreateDir(nameDirs,path='C:\\Users\\Karter\\Desktop\\samle\\'):
    try:
        for num in range(len(nameDirs)):
            os.makedirs(path+str(nameDirs[num]))
        return print("Каталоги созданны")
    except FileExistsError:
        return print("Каталоги уже существуют")
def ClassObject(name):
    i=0
    if '-к' in name.lower():
        i+=1
    reg = re.compile('[^0-9]')
    x=reg.sub('', name)
    try:
        x=int(bibl[x][0])
    except KeyError:
        fq=open('errorId.txt','a')
        fq.write(name+"\n")
        fq.close()
        return 10
    return x+i
def readExel(path,inputID="b7:b375",inputDiagnosis="aa7:aa375"):
        Excel = win32com.client.Dispatch("Excel.Application")
        wb = Excel.Workbooks.Open(path)
        sheet = wb.ActiveSheet
```

```
        print('вв /n пример:b7:b375')

        vals = [r[0].value  for r in sheet.Range(inputID)],[r[0].value  for r in
sheet.Range(inputDiagnosis)]
        wb.Close()
        Excel.Quit()
        for num in range(len(vals[1])):
          if 'меланом' in vals[1][num].lower()  :
             vals[1][num]='0'             #'melanoma'0
          elif  'базал' in vals[1][num].lower():
             vals[1][num]='2'             #'basalioma' 2
          elif  'невус'  in vals[1][num].lower():
             vals[1][num]='4'             #'nevus'4
          elif  'кератома'  in vals[1][num].lower():
             vals[1][num]='6'             #'keratoma'6
          else:
             vals[1][num]='8'             #'other'8
        return  list(vals)
def copyFile(classSemple,path,name,newPath='C:\\Users\\Karter\\Desktop\\samle'):
    try:
        shutil.copyfile(path+name,newPath+'\\'+str(classSemple)+'\\'+name)
    except IOError:
        print('дирректории ',newPath+'\\'+str(classSemple)+'\\'+name,' не
существуют')
def resize(Z,a,b,k=1):
    from scipy.interpolate import RectBivariateSpline
    # Regularly-spaced, coarse grid
    dx, dy = 1, 1
    xmax, ymax = len(Z[0])/2,  len(Z)/2
    x = np.arange(-xmax,  xmax,  dx)
    y = np.arange(-ymax,  ymax,  dy)
    X, Y = np.meshgrid(x,  y)
    interp_spline  = RectBivariateSpline(y,  x, Z, kx=k)
    dx2, dy2 = len(Z[0])/a,  len(Z)/b
    x2 = np.arange(-xmax,  xmax, dx2)
    y2 = np.arange(-ymax,  ymax, dy2)
    X2, Y2 = np.meshgrid(x2,y2)
    Z2 = interp_spline(y2,  x2)
    Z2 = interp_spline(y2,  x2)
    return Z2

path='C:\\Users\\Karter\\Documents\\Python\\bibl.xls'
print('введите путь до эксель файла (Пример:
C:\\Users\\Karter\\Documents\\Python\\bibl.xls)')
vals=readExel(path)
```

```
fs=open('bibls.txt','w')
for num in range(len(vals[1])):
    fs.write( str(int(vals[0][num]))+'\t'+str(vals[1][num])+'\n'  )
fs.close()
bibl = {}
path='C:\\Users\\Karter\\Documents\\Python\\bibls.txt'
with open(path) as file:
    for line in file:
        key, *value = line.split()
        bibl[key] = value
nameDirs=[i  for i in range(0,11)]
CreateDir(nameDirs)
path ="C:\\Users\\Karter\\Documents\\Python\\Raman\\"
n=11
l=[[] for i in range(n)]
dateFile = os.listdir(path)
for word in dateFile:
    fileInDirect=os.listdir(path+word)
    for word2 in fileInDirect:
            with open(path+word+'\\'+ word2) as f:
                buff=[float(line.split()[0])  for line in f]
                f.seek(0)
                buff2=[float(line.split()[1])  for line in f]
                while len(buff)>576:
                    buff.pop()
                    buff2.pop()
                buff=np.asarray(buff)
                buff.shape=(12,48)
                buff2=np.asarray(buff2)
                buff2.shape=(12, 48)
                buffArr=np.hstack((buff,buff2))
                buffArr.shape=(24, 48)
                buff2=np.concatenate((buffArr,buffArr),axis=0)
                buffArr3= buffArr= buff2.reshape(48, 48,1)
                buffArr3=np.append(buffArr3,buffArr,axis=2)
                buffArr3=np.append(buffArr3,buffArr,axis=2)
                buffArr3=buffArr3/buffArr3.max()
                l[ClassObject(word2)].append(buffArr3)#/buffArr.max())
                copyFile(ClassObject(word2),path+word+'\\',word2)
with open('dataSet_48x48(24x48x2)_2x576x2.pickle', 'wb') as temp:
    pickle.dump(l, temp)
```

ПРИЛОЖЕНИЕ В

Листинг алгоритма принятия взвешенного решения

```
# -*- coding: utf-8 -*-
"""
Created on Fri May 12 1:17:16  2018
@author: Кирилл Сержантов

dataset:
"fine_tuning_dataSet_72x72(32x32x4)_1x1024.pickle(NvsB).h5"
class: 1 0 - базалиома(2)  0 1 - невус(4)
dataset:
"fine_tuning_dataSet_72x72(32x32x4)_1x1024.pickle(MvsB).h5"
class:    1 0 - базалиома(2)    0 1 - Меланома(0)
dataset:
"fine_tuning_dataSet_72x72(32x32x4)_1x1024.pickle(NvsM).h5"
class:    1 0 - меланома(0)    0 1 - невус(4)
dataset:
"fine_tuning_dataSet_72x72(32x32x4)_1x1024.pickle(DvsZ).h5"
class:    1 0 - зло(0+2)    0 1 - добро(4)
"""
from keras.optimizers  import Adam
import numpy as np
import pickle
from keras.models import model_from_json

dataSet='DataSet_72x72(32x32x4)_1x1024.pickle'
model_1="fine_tuning_dataSet_72x72(32x32x4)_1x1024.pickle(NvsB).json"
model_2="fine_tuning_dataSet_72x72(32x32x4)_1x1024.pickle(MvsB)_2.json"

weightsMvsB="fine_tuning_dataSet_72x72(32x32x4)_1x1024.pickle(MvsB)_2.h5"
weightsMvsN="fine_tuning_dataSet_72x72(32x32x4)_1x1024.pickle(NvsM).h5"
weightsZvsD="fine_tuning_dataSet_72x72(32x32x4)_1x1024.pickle(ZvsD).h5"
weightsNvsB="fine_tuning_dataSet_72x72(32x32x4)_1x1024.pickle(NvsB).h5"

with open(dataSet, 'rb') as f:
    data_new = pickle.load(f)


json_file=open(model_1,"r")
loaded_model_json_1=json_file.read()
json_file.close()
```

```
json_file=open(model_2,"r")
loaded_model_json_2=json_file.read()
json_file.close()

cnnModelMvsB=model_from_json(loaded_model_json_2)
cnnModelMvsN=model_from_json(loaded_model_json_1)
cnnModelZvsD=model_from_json(loaded_model_json_1)
cnnModelBvsN=model_from_json(loaded_model_json_1)


cnnModelMvsB.load_weights(weightsMvsB)
cnnModelMvsB.compile(loss='binary_crossentropy',optimizer=Adam(lr=1e-
5),metrics=['accuracy'])
cnnModelMvsN.load_weights(weightsMvsN)
cnnModelMvsN.compile(loss='binary_crossentropy',optimizer=Adam(lr=1e-
5),metrics=['accuracy'])
cnnModelZvsD.load_weights(weightsZvsD)
cnnModelZvsD.compile(loss='binary_crossentropy',optimizer=Adam(lr=1e-
5),metrics=['accuracy'])
cnnModelBvsN.load_weights(weightsNvsB)
cnnModelBvsN.compile(loss='binary_crossentropy',optimizer=Adam(lr=1e-
5),metrics=['accuracy'])
k=0.1
for counter in enumerate(data_new[0][:]):
    print("Begin")
    dataNewSempl=counter[1]
    dataNewSempl=dataNewSempl.reshape(1,72,  72,3)
    dataSempMvsB = cnnModelMvsB.predict(dataNewSempl)
    dataSempMvsN = cnnModelMvsN.predict(dataNewSempl)
    dataSempZvsD = cnnModelZvsD.predict(dataNewSempl)
    dataSempBvsN = cnnModelBvsN.predict(dataNewSempl)
    neopl=False
    if np.argmax(dataSempZvsD[:][0])==  1:
        if np.argmax(dataSempBvsN[:][0])==  0:
            neopl=True
        if np.argmax(dataSempMvsN[:][0])==  0:
            neopl=True
        if neopl==False:
            print("Невус")
        else:
            print("Дополнительные исследования")
            if np.argmax(dataSempMvsB[:][0])==  1 and abs(dataSempMvsB[0][0]-
dataSempMvsB[0][1])>k:
                if np.argmax(dataSempBvsN[:][0])==  0 and abs(dataSempBvsN[0][0]-
dataSempBvsN[0][1])>k:
```

```
                print("Базалиома")
            else:
                print("Базалиома, Требуются дополнительные исследования")
        else:
            if np.argmax(dataSempMvsN[:][0])== 0 and abs(dataSempMvsN[0][0]-
dataSempMvsN[0][1])>k:
                print("Меланома")
            else:
                print("Меланома, Требуются дополнительные исследования")
    else:
        if np.argmax(dataSempMvsB[:][0])== 1 and abs(dataSempMvsB[0][0]-
dataSempMvsB[0][1])>k:
            if np.argmax(dataSempBvsN[:][0])== 0 and abs(dataSempBvsN[0][0]-
dataSempBvsN[0][1])>k:
                print("Базалиома")
            else:
                print("Базалиома, Требуются дополнительные исследования")
        else:
            if np.argmax(dataSempMvsN[:][0])== 0 and abs(dataSempMvsN[0][0]-
dataSempMvsN[0][1])>k:
                print("Меланома")
            else:
                print("Меланома, Требуются дополнительные исследования")
```