Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный университет имени М.В. Ломоносова»

Факультет фундаментальной физико-химической инженерии

Центр теоретических проблем физико-химической фармакологии РАН
Лаборатория внутриклеточной сигнализации и системной биологии

КУРСОВАЯ РАБОТА

«Применение свёрточных нейросетей для классификации клеток крови на изображениях.»

Выполнил студент
2 курса 201 группы
Домрачев Иван Сергеевич

_____
(подпись студента)

Научный руководитель
доктор физико-математических наук
Свешникова Анастасия Никитична

_____
(подпись руководителя)

Москва, 2023 г.

Оглавление

1. Введение.

1.1 Актуальность работы

В различных патологических состояниях, таких как бактериальная инфекция или рак, наблюдается сложное взаимодействие между системой свертывания крови, иммунной системой и эндотелием, известное как тромбовоспаление. Основным фактором, вызывающим тромбовоспаление, является взаимодействие между нейтрофилами и тромбоцитами. Этот процесс хорошо изучен in vivo, а также исследуется с помощью ex vivo моделей, позволяющих анализировать движение нейтрофилов вокруг растущего тромба. Для диагностики заболеваний, связанных с кровью, широко применяется идентификация и характеристика образцов крови пациента. Микроскопические данные, связанные с хемотаксисом лейкоцитов и ростом тромбов, обрабатываются и интерпретируются с помощью методов машинного обучения.

Машинное обучение является широко используемым инструментом для решения различных задач в разных областях человеческой деятельности, начиная от рекомендации рекламы и заканчивая управлением автоматическими заводами. В настоящее время искусственные нейронные сети, включая сверточные нейронные сети, широко применяются для решения сложных задач, таких как распознавание речи и оптическое распознавание символов, а также в медицине, включая исследование тромбовоспаления.

1.2 Поставленные задачи

В данной работе были поставлены следующие задачи:

1.2.1 Создание автоматизированного метода обнаружения и классификации подтипов клеток крови с использованием методов компьютерного зрения. Это позволит значительно сократить время и усилия, затрачиваемые на ручное распознавание и классификацию кровяных клеток, а также снизить вероятность ошибок, связанных с человеческим фактором.

1.2.2 Использование сверточных нейронных сетей (CNN) для анализа микроскопических данных крови. CNN являются эффективными инструментами для обработки изображений и позволяют автоматически извлекать признаки из входных данных. Это поможет улучшить точность и эффективность обнаружения и классификации кровяных клеток.

1.2.3 Оценка производительности и точности разработанного метода на данных с электронного микроскопа. Это позволит оценить применимость метода в клинической практике и определить его потенциальные ограничения и возможности для дальнейшего улучшения.

1.2.4 Исследование возможности применения разработанного метода для анализа данных, связанных с тромбовоспалением. Это позволит получить дополнительные выводы о взаимодействии нейтрофилов и тромбоцитов, а также расширить наше понимание механизмов тромбовоспаления.

В результате выполнения этих задач ожидается получение автоматизированного метода для обнаружения и классификации подтипов клеток крови, который может значительно улучшить эффективность диагностики и лечения заболеваний, связанных с кровью, а также способствовать более глубокому исследованию тромбовоспаления и его механизмов.

2. Литературный обзор.

2.1 Анализ объекта и предмета исследования.

Тромбовоспаление и роль различных кровяных клеток.

Тромбовоспаление представляет собой сложное взаимодействие между системой свертывания крови, иммунной системой и эндотелием, которое может возникать в различных патологических состояниях. Возможные причины возникновения тромбовоспаления включают инфекции, опухоли, травмы и хирургические вмешательства. Нейтрофилы, эозинофилы, тромбоциты, лимфоциты и моноциты - это различные типы клеток, которые имеют важную роль в иммунной системе человека и могут участвовать в тромбовоспалительных процессах. Эозинофилы - это один из видов лейкоцитов, или белых кровяных клеток, которые играют важную роль в аллергических реакциях и защите от паразитов. В случае аллергии или инфекции эозинофилы могут активироваться и выделять вещества, которые участвуют в развитии воспаления и повреждении тканей. Эозинофилы также могут взаимодействовать с тромбоцитами и участвовать в образовании тромбов. Тромбоциты, или тромбоцитарные клетки, являются элементами крови, которые отвечают за свертываемость крови и предотвращение кровотечений. В случае повреждения сосудов, тромбоциты активируются и начинают образовывать тромбы, чтобы остановить кровотечение. Однако, избыточное образование тромбов может привести к тромбозу, что является частью тромбовоспалительного процесса. Лимфоциты - это другой вид лейкоцитов, которые играют важную роль в иммунной системе человека. Они отвечают за опознание и уничтожение инфицированных клеток и других патогенов. Лимфоциты также могут выделять вещества, которые участвуют в развитии воспаления и повреждении тканей. Моноциты - это еще один вид лейкоцитов, который может участвовать в тромбовоспалительных процессах. Моноциты могут проникать в поврежденные ткани и превращаться в макрофаги, которые могут фагоцитировать, или поглощать, инфицированные клетки и другие патогены.[1] Макрофаги также могут выделять вещества, которые участвуют в развитии воспаления и повреждении тканей. Нейтрофилы и тромбоциты считаются ключевыми игроками в этом процессе. Нейтрофилы - это белые кровяные клетки, которые играют важную роль в борьбе с инфекцией и имеют способность перемещаться по сосудам и проникать в ткани. Тромбоциты, или свертывающие факторы, в свою очередь,

играют ключевую роль в процессе свертывания крови. Когда в организме возникает воспаление, нейтрофилы и тромбоциты активируются и начинают взаимодействовать друг с другом. Нейтрофилы могут выделять различные вещества, которые привлекают тромбоциты к месту воспаления. В свою очередь, активированные тромбоциты могут выделять множество биоактивных молекул, которые усиливают воспалительный ответ[2].
В результате взаимодействия нейтрофилов и тромбоцитов может происходить образование тромбов и окклюзия кровеносных сосудов, что может привести к различным патологиям, таким как тромбозы и инфаркты. Поэтому важно понимать механизмы, которые лежат в основе тромбовоспаления, и разрабатывать методы его профилактики и лечения.

Микроскопические данные, связанные с хемотаксисом лейкоцитов и ростом тромбов, часто обрабатываются и интерпретируются с помощью методов машинного обучения, микрофотографии клеток используются для диагностики заболеваний и изучения биологических процессов. Основные методы анализа микрофотографий включают классификацию клеток, где изображение разделяется на отдельные клетки для их индивидуального анализа. Это реализуется с помощью алгоритмов пороговой обработки, обнаружения контуров и сегментации на основе машинного обучения. После сегментации, из каждой клетки извлекаются признаки, такие как форма, текстура и интенсивность цвета, которые используются для классификации клеток с помощью моделей машинного обучения. Для этих целей существуют специализированные программные инструменты, которые используют алгоритмы компьютерного зрения для автоматического обнаружения и отслеживания клеток. Эти методы и инструменты помогают автоматизировать процесс анализа микрофотографий и упрощают обработку больших объемов данных.

Хотя машинное обучение и использование cell tracker-ов предоставляют ценные инструменты для анализа микрофотографий клеток, существующие модели не лишены недостатков. Несовершенность моделей может проявляться в проблемах точности сегментации клеток, например, в случае несоответствия исследуемых изображений изображениям, используемых при обучении модели или при наличии шума. Это может привести к ошибочной классификации клеток или потере важной информации.[3]

Для преодоления этих ограничений, активно исследуются новые методы, включая классификацию изображений на основе сверточных нейронных сетей.

Сверточные нейронные сети (Convolutional Neural Networks, CNN) способны автоматически извлекать сложные пространственные признаки из изображений и обладают высокой гибкостью в адаптации к различным типам клеток и условиям съемки. Реализация метода классификации на основе сверточной нейросети может повысить точность и надежность сегментации клеток, что является важным шагом в достижении более точных и полных результатов анализа микрофотографий.[4]
Разработка и применение метода классификации изображений на основе сверточной нейросети имеет большую значимость для улучшения качества анализа микрофотографий клеток, поскольку позволяет преодолеть ограничения существующих моделей и достичь более точной и надежной сегментации клеток.

2.2 Методы машинного и глубокого обучения, используемые в работе.

Одним из наиболее часто используемых классов алгоритмов для решения задач машинного зрения являются нейронные сети.
Искусственная нейронная сеть – это параллельно распределенная структура обработки информации, которая состоит из нейронов, связанных между собой. Модель нейронной сети в программировании является машинной интерпретацией головного мозга. Центральная нервная система человека представлена спинным и головным мозгом, функциональность которых осуществляется при помощи огромного количества нейронов, которые связаны между собой синаптическими связями и могут передавать информацию посредством импульсов.
Нейронные сети обладают способностью к самообучению. Иначе говоря, они могут выдавать результат на основе полученного опыта, обобщая имеющиеся прецеденты на новые случаи. Для обучения нейронной сети чаще всего используют методы обучения с учителем, например, метод обратного распространения ошибки и его модификации. Нередко используют и обучение без учителя. Тип обучения главным образом зависит от решаемой задачи. Обучение с учителем происходит при наличии полного набора размеченных данных. В обучающем наборе каждому примеру соответствует решение, которое сеть должна получить. Разность между правильным решением и полученным представляет собой ошибку, которую необходимо устранять с помощью новой настройки параметров. Обучение без учителя проводится без контроля разработчика над процессом. На вход подается набор данных, и

нейронная сеть пытается самостоятельно найти взаимосвязи. При обучении без учителя у разработчика отсутствуют правильные решения, которые сеть должна получить. Одним из достоинств нейронных сетей является решение задач в условиях неопределенности. Способность к самообучению позволяет сетям искать решение задач с неизвестными закономерностями и зависимостями между входными и выходными данными.

Понятие искусственной нейронной сети было предложено ещё в 1943 году У. Маккалоком и У. Питтсом в статье [5]. В частности, ими была предложена модель искусственного нейрона. Чтобы отразить суть биологических нейронных систем, искусственный нейрон строится следующим образом. Он получает входные сигналы (исходные данные либо выходные сигналы других нейронов нейронной сети) через несколько входных каналов. Каждый входной сигнал проходит через соединение, имеющее определенный вес. С каждым нейроном связано определенное пороговое значение. Вычисляется взвешенная сумма входов, из нее вычитается пороговое значение и в результате получается величина активации нейрона. Сигнал активации преобразуется с помощью функции активации и в результате получается выходной сигнал нейрона.
На Рис.1 приведен пример искусственного нейрона.

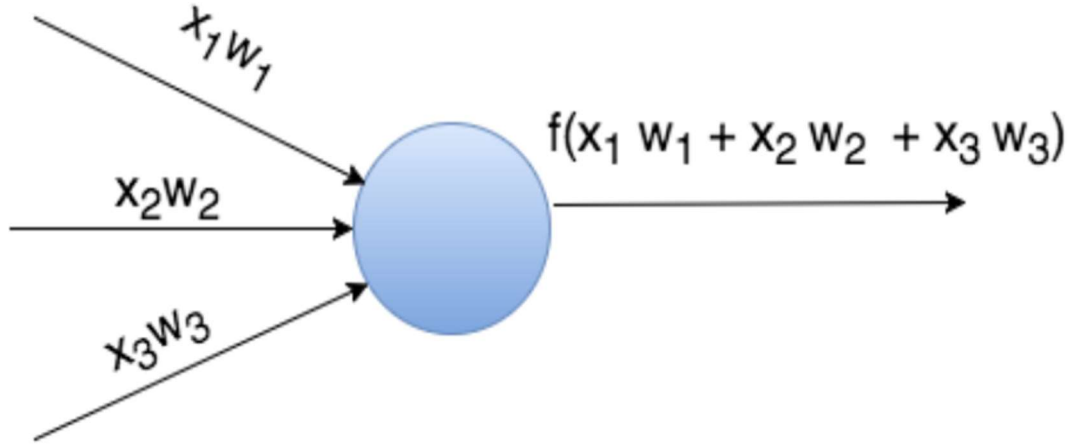

Рис. 1: Искусственный нейрон

x_i– входной сигнал w_i– вес входного сигнала f – функция активации

Функции активации

- Положительная линейная: $f(s) = \max(0, s)$
- Линейная: $f(s) = s$

- Сигмоидная: $f(s) = \frac{1}{1+e^{-s}}$
- Софтмакс (Softmax): $f(s)_j = \frac{e^{s_j}}{\sum_{k=1}^{K} e^{s_k}}$, для $j = 1, \ldots, K$

Введем обозначения: X - множество описаний объектов, Y - множество допустимых ответов. Предполагается, что существует неизвестная целевая зависимость - отображение $y^*: X \to Y$, значения которой известны только на объектах конечной обучающей выборки $X_m = \{(x_1, y_1), \ldots, (x_m, y_m)\}$
Вводится функция потерь $L(y, y_0)$, характеризующая величину отклонения ответа y от правильного ответа $y_0 = y^*(x)$
на произвольном объекте x ∈ X. Тогда эмпирический риск [6] - функционал качества, характеризующий среднюю ошибку на
обучающей выборке:

$$Q(a, X^m) = \frac{1}{m}\sum_{i=1}^{m} L(y_i, y^*(x_i))$$

В процессе обучения нейронная сеть настраивает веса W, минимизируя эмпирический риск.
При решении задачи многоклассовой классификации на выходе нейронной сети
необходимо получить вероятность принадлежности объекта каждому из классов. В этом случае в качестве функции потерь обычно используется кросс-энтропия

$$L\big(y_i, y^*(x_i)\big) = -\sum_{j=1}^{K} y_{ij}^* \log\,(y_{ij})$$

K — количество меток классов в задаче.
В данной работе для классификации клеток с помощью нейронных сетей используется описанная функция потерь.

С появлением больших объемов данных и больших вычислительных возможностей стали активно использоваться нейронные сети. Особую популярность получили сверточные нейронные сети, архитектура которых была предложена Яном Лекуном [7] и нацелена на эффективное распознавание изображений. Свое название архитектура сети получила из-за наличия

операции свёртки, суть которой в том, что каждый фрагмент изображения умножается на матрицу (ядро) свёртки поэлементно,
а результат суммируется и записывается в аналогичную позицию выходного изображения. В архитектуру сети заложены априорные знания из предметной области компьютерного зрения: пиксель изображения сильнее связан с соседним (локальная корреляция) и объект на изображении может встретиться в любой части изображения. Особое внимание светрочные нейронные сети получили после конкурса ImageNet, который состоялся в октябре 2012 года и был посвящен классификации объектов на фотографиях. В конкурсе требовалось распознавание образов в 1000 категорий. Победитель данного конкурса — Алекс Крижевский, используя сверточную нейронную
сеть, значительно превзошел остальных участников [8]. Успех применения светрочных нейронных сетей к классификации изображений
привел к множеству попыток использовать данный метод к другим задачам. В последнее время их стали активно использоваться для задачи классификации текстов.

Архитектура сверточной нейронной сети

Сверточная нейронная сеть обычно представляет собой чередование сверточных слоев (convolution layers), субдискретизирующих слоев (subsampling layers) и при наличии полносвязных слоев (fully-connected layer) на выходе. Все три вида слоев могут чередоваться в произвольном порядке [7]. В сверточном слое нейроны, которые используют одни и те же веса, объединяются в карты признаков (feature maps), а каждый нейрон карты признаков связан с частью нейронов предыдущего слоя. При вычислении сети получается, что каждый нейрон выполняет свертку некоторой области предыдущего слоя (определяемой множеством нейронов, связанных с данным нейроном).
Пример архитектуры сверточной нейронной сети представлен на Рис. 2.

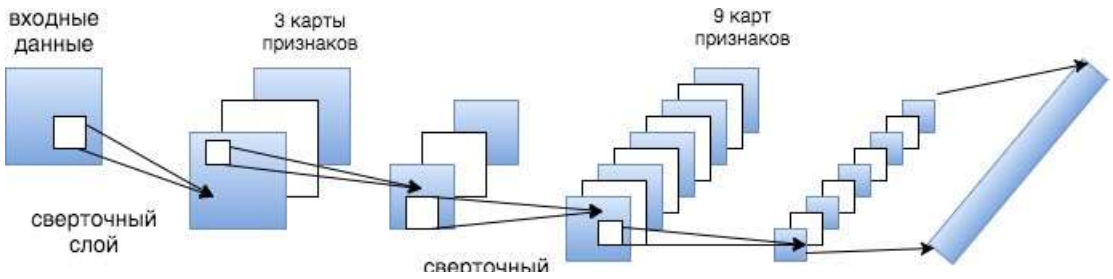

Рис. 2: Архитектура сверточной нейронной сети

Полносвязный слой

Слой, в котором каждый нейрон соединен со всеми нейронами на предыдущем уровне, причем каждая связь имеет свой весовой коэффициент. На Рис.3 показан пример полносвязного слоя.

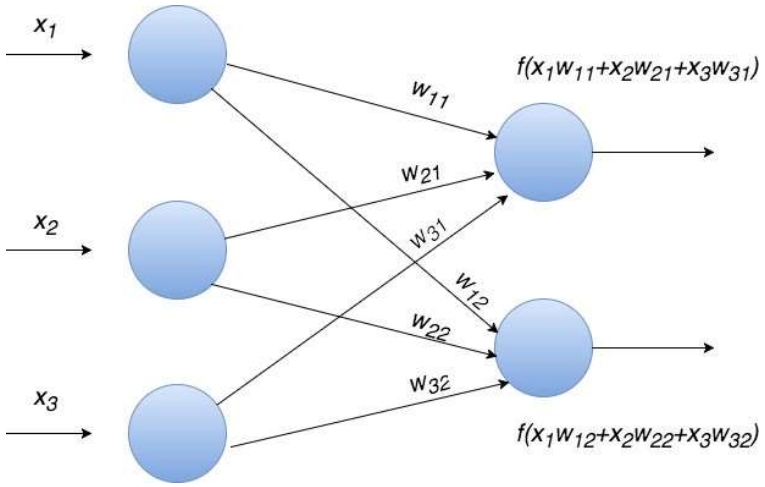

Рис. 3: Полносвязный слой

w_{ij}– вес входного сигнала, f – функция активации.

Сверточный слой

В отличие от полносвязного, в сверточном слое нейрон соединен лишь с ограниченным количеством нейронов предыдущего уровня, т. е. сверточный слой аналогичен применению операции свертки, где используется лишь матрица весов небольшого размера (ядро свертки), которую «двигают» по всему обрабатываемому слою. Еще одна особенность сверточного слоя в том, что он немного уменьшает изображение за счет краевых эффектов.

На Рис. 4 показан пример сверточного слоя с ядром свертки размера 3 × 3.

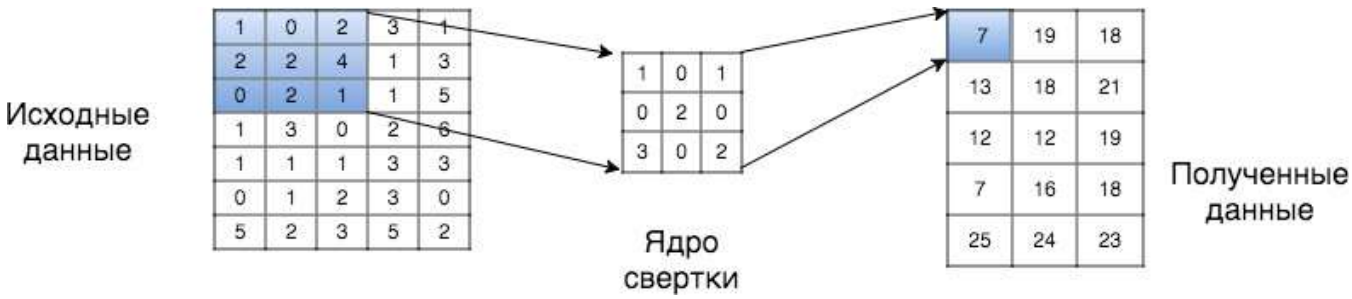

Рис. 4: Сверточный слой

Субдискретизирующий слой

Субдискретизирующий слой – «операция подвыборки» или «операция объединения». Слои этого типа выполняют уменьшение размерности (обычно в несколько раз). Это можно делать разными способами, но зачастую используется метод выбора максимального элемента (max-pooling) — вся карта признаков разделяется на ячейки, из которых выбираются максимальные по значению.

На Рис. 5 показан пример субдискретизирующего слоя с методом выбора максимального элемента.

Рис. 5: Субдискретизирующий слой

Batch Normalization (батч-нормализация) это метод нормализации входных данных, применяемый в глубоких нейронных сетях. Он был предложен в статье "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift"[9].

Батч - подмножество данных, которое обрабатывается моделью за один проход. Идея метода заключается в том, что для каждого батча входных данных, используемых для обучения нейронной сети, вычисляются среднее значение и стандартное отклонение. Затем каждый элемент батча нормируется по этим значениям. Таким образом, входные данные нормализуются и центрируются на каждом этапе обучения, что позволяет ускорить сходимость и увеличить производительность. Применение батч-нормализации в нейронных сетях имеет несколько преимуществ:

Уменьшает внутреннюю ковариацию (internal covariate shift) - изменение распределения входных данных на каждом слое нейронной сети. Это уменьшает возможность затухания градиента и ускоряет сходимость нейронной сети.

Увеличивает стабильность процесса обучения, так как при нормализации входных данных стандартное отклонение близко к единице, а среднее значение близко к нулю.

Кроме того, батч-нормализация является дополнительным шагом нормализации входных данных, которые передаются между слоями нейронной сети, что уменьшает переобучение.

Батч-нормализация может применяться к любому типу слоев нейронной сети, но наибольшую пользу она приносит в сверточных и полносвязных слоях.

Обучение сверточной нейросети.

Обучение сверточной нейросети происходит поэтапно:
1. Инициализация весов. На первом этапе веса нейросети инициализируются случайными значениями, например, из равномерного или нормального распределения.
2. Прямой проход (forward pass). На этом этапе входные данные подаются на вход сверточной нейросети, которая последовательно выполняет свертки, активации и пулинги. Результатом является выходной тензор, который содержит предсказания нейросети для каждого класса.
3. Вычисление функции потерь. После прямого прохода вычисляется значение функции потерь, которая показывает, насколько хорошо нейросеть справляется с задачей классификации. Для классификации с использованием функции потерь кросс-энтропии вычисляется кросс-энтропия между предсказаниями нейросети и правильными метками классов.
4. Обратное распространение ошибки (backpropagation). На этом этапе производится расчет градиентов функции потерь по весам нейросети. Градиенты вычисляются с помощью алгоритма обратного распространения ошибки, который позволяет определить, как изменение каждого веса повлияет на значение функции потерь.
5. Обновление весов. С помощью оптимизатора Adam[10] производится обновление весов нейросети в соответствии с вычисленными градиентами. Оптимизатор Adam использует комбинацию градиентов первого и второго порядков, чтобы быстрее и эффективнее находить оптимальные значения весов.
6. Повторение. Процессы прямого и обратного проходов повторяются до тех пор, пока значение функции потерь не достигнет минимума или не будет достигнуто максимальное количество эпох обучения.

Таким образом, обучение сверточной нейросети с оптимизатором Adam заключается в последовательном выполнении прямого и обратного проходов с последующим обновлением весов нейросети.

Backpropagation, или обратное распространение ошибки, является алгоритмом обучения нейронных сетей, который позволяет эффективно распространять ошибку от выходов нейронной сети к ее входам.

Идея заключается в том, что при обучении сети известен правильный ответ на каждом шаге, поэтому можно рассчитать ошибку на выходе сети. Зная эту ошибку, можно использовать ее для корректировки весов на каждом слое сети, начиная с последнего слоя и двигаясь назад до первого слоя. Для каждого нейрона в сети вычисляется ошибка, которая является производной функции ошибки по входу нейрона. Эта ошибка затем передается обратно через сеть, корректируя веса на каждом слое в соответствии с ошибкой[8]. Алгоритм backpropagation может быть использован вместе с любым методом оптимизации весов, таким как градиентный спуск или оптимизаторы, такие как Adam или RMSprop. Backpropagation позволяет нейронным сетям обучаться на больших объемах данных, и является основным алгоритмом обучения глубоких нейронных сетей. Он является одним из ключевых инструментов для достижения высокой точности в задачах машинного обучения, таких как распознавание образов, классификация изображений, распознавание речи и др.

ResNet-50

ResNet-50 - это глубокая сверточная нейронная сеть, состоящая из 50 слоев, которая была представлена в 2015 году компанией Microsoft Research. ResNet-50 стал очень популярным алгоритмом в задачах компьютерного зрения и глубокого обучения благодаря своей высокой точности и относительно небольшому количеству параметров (около 25 миллионов).

Основная идея ResNet заключается в использовании блоков остаточных соединений (Residual blocks), которые позволяют решить проблему затухания градиента в глубоких нейронных сетях. Затухание градиента возникает, когда градиенты возвращаются на более ранние слои нейронной сети и становятся настолько малыми, что не способны обновить веса.

Каждый блок остаточных соединений в ResNet-50 состоит из нескольких сверточных слоев, а также соединения, называемого "shortcut connection", который позволяет обновлять веса с использованием информации из более

ранних слоев нейронной сети. Формально, каждый блок остаточных соединений может быть выражен формулой:

$$y_i = f(x_i, w_i) + x_i$$

где x_i- входной вектор, w_i - вектор параметров, f - операция свертки с функцией активации. Это соединение называется "shortcut connection", поскольку позволяет обновлять веса с использованием информации из более ранних слоев нейронной сети.

ResNet-50 состоит из пяти блоков остаточных соединений с разным количеством сверточных слоев. Внутри каждого блока сверточные слои объединены с операцией батч-нормализации и функцией активации ReLU. После последнего блока остаточных соединений на выходе получается вектор признаков, который подается на полносвязный слой с последующей классификацией. Схема нейросети представлена на рис. 7.

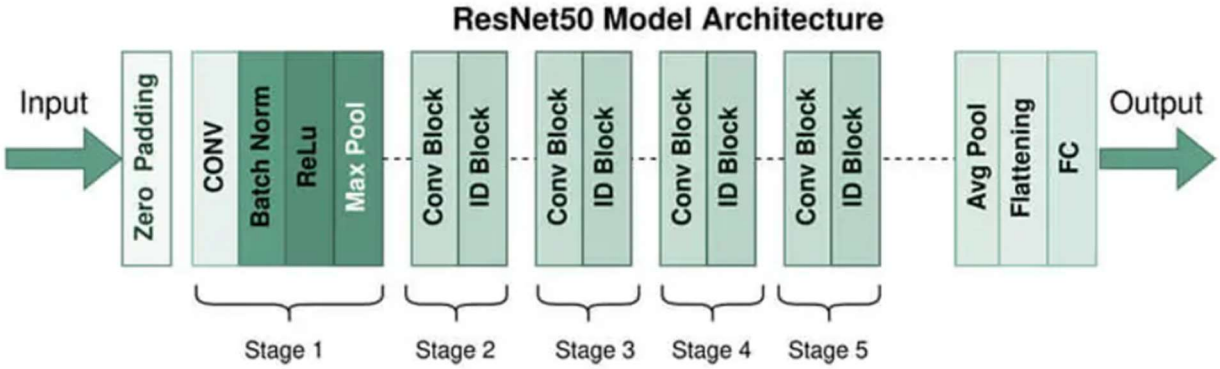

Рис. 7 – схема ResNet-50

3. Практическое выполнение. Результаты и их обсуждение.

Данные для анализа представляют собой датасет с медицинскими изображениями[14].
Этот набор данных содержит 12 500 увеличенных изображений клеток крови (Файлы формата JPEG) с сопровождающими их метками типов клеток (Файл формата CSV). Имеется около 3000 изображений для каждого из 4 различных типов клеток. К типам клеток относятся эозинофилы, лимфоциты, моноциты и нейтрофилы. Пример изображения на рис.6

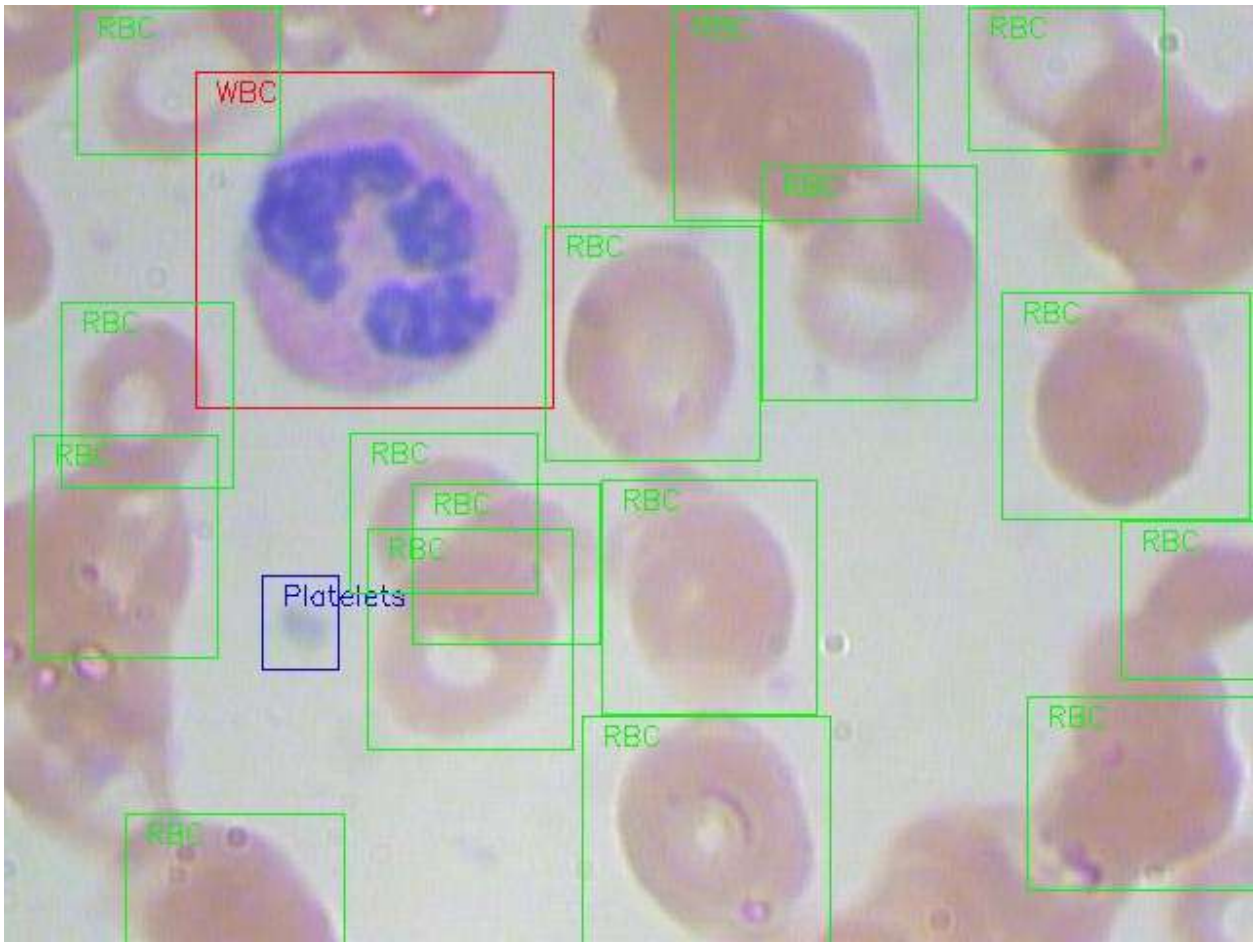

Рис. 6 – пример изображения из набора данных.

Идея заключается в том, чтобы обучить модель предсказывать к какому классу относится клетка. За основу взята архитектура сверточной нейронной сети ResNet-50[7], разработанная для классификации изображений.
Программная реализация выполнена с использованием языка программирования Python и фреймворка глубокого обучения

Keras(TensorFlow)[12]. Код размещен с доступом по ссылке в репозитории Github[13] и в приложении к курсовой работе. В результате обучения нейросети и предсказания на тестовом датасете были достигнуты следующие результаты:

	precision	recall	f1-score	support
EOSINOPHIL	0.93	0.83	0.88	623
LYMPHOCYTE	1.00	1.00	1.00	620
MONOCYTE	0.99	0.75	0.85	622
NEUTROPHIL	0.70	0.93	0.80	624
accuracy			0.88	2489
macro avg	0.90	0.88	0.88	2489
weighted avg	0.90	0.88	0.88	2489

таблица 1.

Precision (точность) - доля верно классифицированных объектов данного класса относительно всех объектов, которые модель отнесла к этому классу.

Recall (полнота) - доля верно классифицированных объектов данного класса относительно всех объектов данного класса в исходном датасете.

F1-score (F-мера) - среднее гармоническое между precision и recall. Это метрика, которая учитывает обе метрики и позволяет оценить качество модели в целом.

Support - количество объектов данного класса в исходном датасете.

Accuracy (точность) - доля верно классифицированных объектов относительно всех объектов в тестовой выборке. Общая точность модели составила 88%.

Macro avg - среднее арифметическое между precision, recall и F1-score для всех классов.

Weighted avg - взвешенное среднее между precision, recall и F1-score для всех классов, где каждый класс учитывается в соответствии с его размером в исходном датасете.

Нейросеть обучалась на протяжении 50 эпох, с функцией досрочного прекращения обучения при отсутствии улучшения качества.

График функции потерь (loss) отображает, как изменяется значение функции потерь во время обучения сети(Рис. 8). График точности (accuracy) на рис. показал, как изменяется точность распознавания данных в зависимости от числа эпох обучения(Рис 9.)

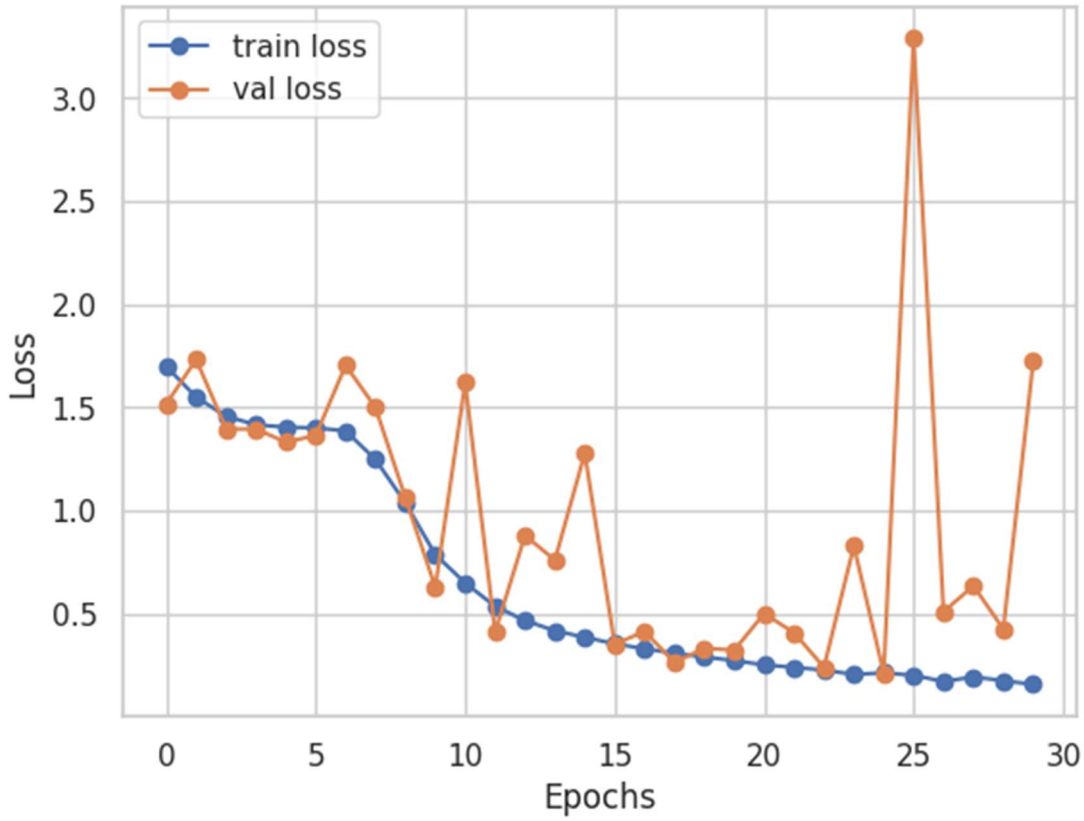

Рис. 8 График зависимости значения функции потерь на тестовом и на валидационном датасете от номера эпохи

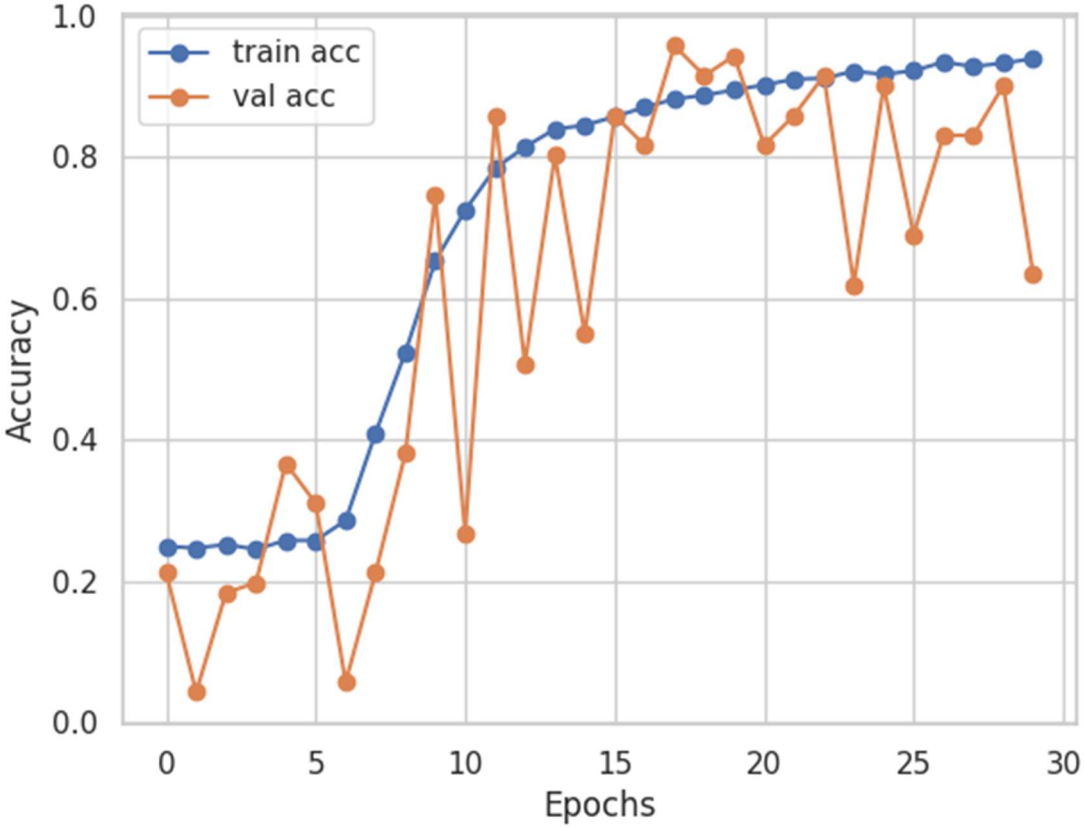

Рис. 9 - График зависимости точности предсказания на тестовом и на валидационном датасете от номера эпохи

Анализируя результаты, можно сделать вывод о том, что нейросеть достигла высокой точности в распознавании типов кровяных клеток. Средняя взвешенная точность предсказания достигла 0.88 – доля верно предсказанных классов. Можно заметить, что наибольшие трудности возникают с классификацией нейтрофилов, где точность составляет только 0.70. Результаты оказываются полезными и могут в перспективе быть использованы для улучшения средств диагностики заболеваний, связанных с кровью.

Вывод

В ходе выполнения курсовой работы была реализована модель классификации кровяных клеток с использованием языка программирования Python и библиотеки Keras. Была достигнута точность в 90% на валидационном наборе данных, что говорит о высокой эффективности разработанной модели. В процессе работы были изучены основы программирования на языке Python, а также основы глубокого обучения с использованием библиотеки Keras. Были рассмотрены различные архитектуры нейронных сетей, а также применены методы обучения и оптимизации модели.

Реализованная модель может быть использована для облегчения диагностики заболеваний, связанных с кровью, быть полезна для дальнейшего изучения процесса тромбовоспаления и разработки программного обеспечения для анализа медицинской информации.

В целом, выполнение курсовой работы позволило расширить знания в области машинного обучения, глубокого обучения и разработки программного обеспечения.

Список используемой литературы

1. Yoo S. K., Starnes T. W., Deng Q., Huttenlocher A. Lyn is a redox sensor that mediates leukocyte wound attraction in vivo. // Nature. — 2011
2. С.П.Свиридова, О.В.Сомонова, Ш.Р.Кашия и др. Роль тромбоцитов в воспалении и иммунитете. Исследования и практика в медицине 2018, т.5, №3
3. Nicolas Chenouard, Ihor Smal, Fabrice de Chaumont и др. Objective comparison of particle tracking methods // Nature. — 2014
4. В.К. Беляков, Е.П. Сухенко, А.В. Захаров и др. О методике классификации клеток крови и ее программной реализации. Программные продукты и системы — 2014
5. McCulloch, W. S. A logical calculus of the ideas immanent in nervous activity / Warren S. McCulloch, Walter Pitts // Springer New York. — 1943
6. Воронцов, К. В. Курс лекций по машиному обучению / К. В. Воронцов. — 2015.
7. Yann LeCun Leon Bottou, Y. B. Gradient-based learning applied to document recognition / Yoshua Bengio Yann LeCun, Leon Bottou, Patrick Haffner // IEEE. — 1998.
8. John Duchi Elad Hazan, Y. S. Adaptive subgradient methods for online learning and
stochastic optimization / Yoram Singer John Duchi, Elad Hazan // JMLR. — 2011.
9. Sergey Ioffe, Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.
10. Werbos P. J., Beyond regression: New tools for prediction and analysis in the behavioral sciences. Ph.D. thesis, Harvard University, Cambridge, MA, 1974.
11. Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Deep Residual Learning for Image Recognition
12. Документация библиотеки Keras [Электронный ресурс]. – Режим доступа: http://keras.io/ (дата обращения: 05.05.2023).
13. Репозиторий Github [Электронный ресурс]. – Режим доступа: https://github.com/idomrachev/CourseWork (дата обращения: 05.05.2023).
14. Репозиторий Github [Электронный ресурс]. – Режим доступа: https://github.com/cosmicad/dataset (дата обращения: 05.05.2023).

Приложение. Программная реализация алгоритма.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator,
load_img, img_to_array
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Dropout, Add, DepthwiseConv2D
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense,
Input, InputLayer, GlobalAveragePooling2D ,Activation, BatchNormalization
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping

from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score

import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)

plt.rcParams['font.size'] = 14
from google.colab import drive
drive.mount('/content/drive')
!cp -r "/content/drive/MyDrive/blood_cells" "/content/"
img_size = 224
batch_size = 16


train_datagen = ImageDataGenerator(rescale =
1/255.,brightness_range=[0.5,1.5], zoom_range=0.2,
                                   width_shift_range =0.15,
height_shift_range =0.15, horizontal_flip=True )
val_datagen = ImageDataGenerator(rescale = 1/255.)
test_datagen = ImageDataGenerator(rescale = 1/255.)


train_generator =
train_datagen.flow_from_directory('/content/blood_cells/dataset2-
master/dataset2-master/images/TRAIN',
                                                  target_size =
(img_size, img_size),
                                                  batch_size =
batch_size,
                                                  shuffle=True,
                                                  class_mode='sparse')
```

```
val_generator =
val_datagen.flow_from_directory('/content/blood_cells/dataset2-
master/dataset2-master/images/TEST_SIMPLE',
                                                target_size =
(img_size, img_size),
                                                batch_size =
batch_size,
                                                shuffle=False,
                                                class_mode='sparse')

test_generator =
test_datagen.flow_from_directory('/content/blood_cells/dataset2-
master/dataset2-master/images/TEST',
                                                target_size =
(img_size, img_size),
                                                batch_size =
batch_size,
                                                shuffle=False,
                                                class_mode = "sparse")
def residual_block(x, kernel_size, filters, s=2):
    f1,f2,f3 = filters
    x_shortcut = x

    x_shortcut = Conv2D(f3, kernel_size=(1,1), strides=(s,s),
padding='valid')(x_shortcut)
    x_shortcut = BatchNormalization()(x_shortcut)

    x = Conv2D(f1, kernel_size=(1,1), strides=(s,s), padding='valid')(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)

    x = Conv2D(f2, kernel_size=kernel_size, strides=(1,1),
padding='same')(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)


    x = Conv2D(f3, kernel_size=(1,1), strides=(1,1), padding='valid')(x)
    x = BatchNormalization()(x)

    x = Add()([x,x_shortcut])
    x = Activation('relu')(x)

    return x
input_layer = Input(shape=(224,224,3))
x = Conv2D(64, kernel_size=(7,7), strides=(2,2))(input_layer)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = MaxPooling2D((3,3), strides=(2,2))(x)
```

```
x = residual_block(x,3,[64,64,128],s=1)
x = residual_block(x,3,[64,64,128],s=1)
x = residual_block(x,3,[64,64,128],s=1)

x = residual_block(x,3,[128,128,512])
x = residual_block(x,3,[128,128,512])
x = residual_block(x,3,[128,128,512])

x = residual_block(x,3,[256,256,1024])
x = residual_block(x,3,[256,256,1024])
x = residual_block(x,3,[256,256,1024])

x = GlobalAveragePooling2D()(x)
x = Flatten()(x)
output_layer = Dense(4, activation='softmax')(x)

resnet50 = Model(inputs=input_layer,outputs=output_layer)
checkpoint_filepath = '/tmp/checkpoint'
checkpoint = ModelCheckpoint(filepath=checkpoint_filepath,
monitor="val_loss",
                             mode="min",
                             save_best_only = True,
                             verbose=1)

earlystopping = EarlyStopping(monitor='val_loss',min_delta = 0, patience =
5, verbose = 1, restore_best_weights=True)
opt = tf.keras.optimizers.Adam(learning_rate=0.0001)

resnet50.compile(loss='sparse_categorical_crossentropy', optimizer=opt,
metrics=['accuracy'])
history = resnet50.fit(train_generator,
                       epochs=50,
                       validation_data=val_generator,
                       callbacks=[checkpoint,earlystopping])
y_pred = resnet50.predict(test_generator).round()
y_test = test_generator.classes
class_labels = list(test_generator.class_indices.keys())
y_pred = np.argmax(y_pred,axis=1)

print(classification_report(y_test,y_pred,target_names = class_labels))
print(accuracy_score(y_test, y_pred))
plt.plot(history.history['accuracy'], '-o',label='train acc')
plt.plot(history.history['val_accuracy'],'-o', label='val acc')
plt.legend()
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.show()
plt.plot(history.history['loss'], '-o', label='train loss')
plt.plot(history.history['val_loss'], '-o', label='val loss')
plt.legend()
```

```
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.show()
```