Пермский филиал федерального государственного автономного образовательного учреждения высшего образования

«Национальный исследовательский университет

«Высшая школа экономики»

Факультет экономики, менеджмента и бизнес-информатики

Образовательная программа бакалавриата «Программная инженерия»

**ОТЧЕТ**

**по проектной работе**

Выполнила студентка группы ПИ-20-1

Лобастова Марина Антоновна

(фамилия, имя, отчество)

(подпись)

Руководитель проекта

м. н. с. НИУ ВШЭ-Пермь

(должность)

Собянин Кирилл Валентинович

(Фамилия, Имя, Отчество)

\_\_\_\_\_\_\_\_\_\_\_

(оценка) (подпись)

(дата)

Пермь, 2021

Результаты проверки работы в системе “Антиплагиат”

Изображение выглядит как текст, снимок экрана, компьютер

Автоматически созданное описание

Аннотация

Лобастова М. А.

“Разработка загрузчика данных для автоматизированной системы локализации очага ишемического инсульта в остром периоде”, 2021 г. Курсовой проект, кафедра информационных технологий в бизнесе. 28 страниц, 1 рисунок.

В данной работе рассматривается решение задачи сегментации снимков МРТ головного мозга с целью оценки локализации и объемов кровоизлияния для диагностики ишемического инсульта в остром периоде. Спроектирована архитектура типа Unet, нейронная сеть обучена на 20–25 снимках каждого из 51 пациентов в используемом наборе данных, показатель точности работы нейросети по метрике dice - 53%. Результат получен после тестирования системы на данных 10 пациентов. Подведены итоги работы, рассмотрены перспективы использования полученных результатов будущих исследований.

Оглавление

[Введение 5](#_Toc75866996)

[1 Актуальность проблемы 5](#_Toc75866997)

[2 Цели и задачи 6](#_Toc75866998)

[Глава 1. Теоретическая часть 7](#_Toc75866999)

[1.1 Обзор научной литературы 7](#_Toc75867000)

[1.2 Теоретическое обоснование задачи 8](#_Toc75867001)

[1.3 Описание используемых технологий и инструментальных средств…………………………………………………………………………...9](#_Toc75867002)

[Глава 2. Проектирование 12](#_Toc75867003)

[2.1 Архитектура Unet 12](#_Toc75867004)

[2.2 Проектирование нейросети 13](#_Toc75867005)

[2.3 Метрики точности 14](#_Toc75867006)

[Глава 3. Разработка, тестирование, внедрение программы 17](#_Toc75867007)

[3.1 Оценка результатов обучения 17](#_Toc75867008)

[3.2 Стратегии повышения точности работы нейросети 17](#_Toc75867009)

[3.2.1 Расширение набора данных для обучения 17](#_Toc75867010)

[3.2.2 Предобученный энкодер 17](#_Toc75867011)

[3.2.3 Перспективы использования 18](#_Toc75867012)

[Заключение 19](#_Toc75867013)

[Библиографический список 20](#_Toc75867014)

ПРИЛОЖЕНИЕ А [Код программы нейронной сети 22](#_Toc75867016)

Введение

1 Актуальность проблемы

Около 15,7% всех смертей в мире происходят в результате инсульта, который, таким образом, занимает второе место в мире среди всех причин смертности. По данным официальной статистики, в РФ ежегодно происходит более 400 тыс. случаев инсульта, причем лишь 8–10% из них оказываются относительно легкими и заканчиваются восстановлением нарушенных функций в первые 3 недели заболевания. В течение первого года умирают более 150 тыс. человек. В стране проживают более 1 млн человек, перенесших инсульт, что составляет 0,7% всего населения [1].

Частично данные статистики связаны со спецификой заболеваний: многие из них развиваются незаметно для человека, пока в какой-то момент не наступает острое кризисное состояние, при котором нужно действовать чрезвычайно оперативно.

Главным методом диагностики как при выявлении инсульта, так и при его непосредственном лечении, является анализ МРТ головного мозга. При таком подходе оценка локализации и объемов кровоизлияния является приоритетной задачей, потому что позволяет только оценить ущерб, нанесенный организму, а, значит, прогнозировать реабилитационный потенциал и неврологический дефицит после выписки пациента. Также оценка локализации и объемов кровоизлияния важна при определении типа инсульта, от которого зависит дальнейшая стратегия лечения и реабилитации.

Автоматизированная система диагностики очагов кровоизлияния позволила бы не только многократно ускорить диагностику кровоизлияний, но и, при высокой точности работы системы, устранить влияние человеческого фактора на диагноз пациента - точность диагнозов системы не зависит от количества пациентов в день или концентрации в момент времени. Таким образом, автоматизированная диагностика МРТ головного мозга могла бы стать крайне востребованной и актуальной технологией, которая бы приблизила человечество к решению проблемы крайне высокой смертности от болезней кровеносной системы.

2 Цели и задачи

*Гипотеза* нашего исследования состоит в следующем: если будет разработаем автоматизированную систему для оценки локализации и объема кровоизлияния по снимкам МРТ, такая система позволит сократить время постановки диагноза и повысит качество оказываемой медицинской помощи.

*Целью* работы является создание программы, сегментирующей снимок МРТ на область, относящуюся к очагу кровоизлияния, и остальную часть мозга.

Задачи работы:

1. провести литературный обзор, изучить схожие исследования и современные методы, и стратегии решения проблемы;
2. определить тип и архитектуру системы, оптимальные для задачи сегментации снимков МРТ;
3. определить оптимальные метрики для оценки точности работы системы, которые бы в достаточной степени отражали качество результатов системы;
4. обучить систему на тренировочном наборе данных, оценить качество ее работы;
5. оценить результаты проделанной работы, изучить возможности интеграции разработанной системы в более комплексные алгоритмы оценки здоровья пациентов.

*Объектом исследования* курсового проекта является ишемический инсульт в остром периоде. *Предмет исследования* – это автоматизация системы сегментации снимков МРТ для диагностики ишемических инсультов.

*Методы работы* – анализ, моделирование и проектирование.

На данный момент автоматизированные системы диагностики, получающие данные из медицинских изображений, активно изучаются учеными. Подробный анализ исследований, проведенных на эту тему, представлен в главе 1 курсового проекта. Наибольшую точность работы показывают специально организованные системы машинного обучения - нейросети, построенные в соответствии с архитектурой типа Unet. Создание именно такой системы представляется нам наиболее эффективным решением проблемы исследования, поэтому разрабатываемая система проектируется согласно описанным выше принципам.

Глава 1. Теоретическая часть

1. Обзор научной литературы

Извлечение значимой информации из медицинских изображений, например снимков КТ и МРТ, с помощью автоматизированных систем анализа изображений на сегодняшний день является критически важной задачей для всей мировой науки. Предполагается, что автоматизация диагностики заболеваний на основе медицинских изображений позволит исключить человеческий фактор и сопряженные с ним ошибки при постановке диагнозов. Также в случаях, когда от времени, затраченного на постановку диагноза, зависит объем поражения ткани или степень запущенности заболевания, автоматизированные системы диагностики позволяют многократно сократить время между сканированием снимка и началом реанимационных мероприятий.

Традиционным методом сегментации изображений является сведение задачи сегментации к задачам оптимизации графов, например, на основе алгоритмов муравьиных колоний. Именно такой подход применен в исследовании [2], однако, как замечают авторы в заключении, “проблемы надежности и ответственности остаются главными препятствиями на пути широкомасштабного использования этих алгоритмов”.

Более эффективным и современным решением для задач кластеризации и сегментации является применение нейронных сетей, в особенности сверточных, которые нацелены на распознавание образов. Например, в исследовании [3] авторы рассматривают сегментацию гистологических изображений при помощи модели обучаемого активного контура. В данной работе приводится гибридный метод объектной сегментации желёз, который сочетает в себе как современные сверточные нейронные сети, так и классические методы математической обработки изображений.

Из множества архитектур сверточных нейросетей на медицинских изображениях одной из самых эффективных является архитектура типа Unet, впервые предложенная исследователями из Университета Фрайбурга [4]. Например, в исследовании [5] такая нейросеть используется для сегментирования снимков МРТ головного мозга для отслеживания изменений мозговых структур, что может быть полезно для раннего диагностирования болезни Альцгеймера. Другой пример использования нейросети с архитектурой типа Unet – исследование [6], цель которого заключается в автоматизации диагностики гематом головного мозга на снимках КТ. Похожая задача стояла перед авторами статьи [7], которые занимались сегментированием следов черепно-мозговых травм на КТ снимках.

В обзоре [8] рассматриваются современные решения задачи обнаружения мозгового инсульта, основанные на глубоком обучении и нейровизуализации. Данный обзор приводит качественные и количественные сравнения результатов исследований, которые могут быть полезны на этапе оценки полученных результатов точности работы нашей системы.

Во всех приведенных выше, а также многих других исследованиях нейронные сети с архитектурой Unet показывают высокую точность, результаты работы систем сопоставимы с человеческими, а иногда и превосходят их, что делает Unet перспективной областью исследований в рамках анализа и сегментирования медицинских изображений. Однако на данный момент не было обнаружено исследования, описывающего систему для сегментации снимков МРТ для оценки локализации и объемов мозговых кровоизлияний, что подтверждает актуальность нашей работы.

1. Теоретическое обоснование задачи

Часть программной системы, рассматриваемая в данном отчёте, должна выполнять следующие задачи:

* интеграция системы загрузки данных с системой обработки и сегментации изображений;
* обучение модели бинарной сегментации очагов кровоизлияния на снимках МРТ при помощи предварительно созданного набора данных;
* оценка точности работы полученной модели;
* сохранение предобученной модели в формате, позволяющем интегрировать систему в более комплексные.

На этапе разработки программной системы подготовлен модуль нейросети, задачей которой является бинарная сегментация поступающего на вход изображения на кластеры очагов кровоизлияния и остальное. Требуется определить принадлежность конкретного пикселя изображения к очагу, что является задачей бинарной классификации, и произвести это действие для каждого пикселя входного изображения, таким образом создав маску для изображения, соответствующую расположению очага на снимке.

1. Описание используемых технологий и инструментальных средств

Для решения задачи сегментации МРТ снимков оптимальным является использование сверточных нейросетей с архитектурой Unet. Однако для реализация такой модели системы возможна почти на любом общедоступном объектно-ориентированном языке программирования, который поддерживает работу с математической библиотекой для моделирования системы нейронов.

Одно из возможных решений - использование языка С++, главным преимуществом которого является быстродействие, что очень важно для процессов машинного обучения, которые могут длиться часами. Также в С++ имеется несколько библиотек, доступных для машинного обучения и нейронных сетей, которые способствуют более быстрому выполнению сложных алгоритмов.

Другим популярным решением для задач глубокого обучения является язык Java. При выборе его открывается возможность использования виртуальной машины Java Virtual Machine, которая позволяет легко реализовать модели машинного обучения на различных платформах. Также Java предлагает специальные инструменты для визуализации процессов, протекающих во время обучения и тестирования нейросети. Java – язык с открытым исходным кодом, поэтому он доступен для любых модификаций, которые разработчики сочтут нужными. Java поддерживается многими библиотеками машинного обучения, в том числе Java Machine Learning Library.

Однако на сегодня небезосновательно общепринятым стандартом является использование языка программирования Python. Этот язык простой, гибкий, масштабируемый; благодаря повсеместному использованию он постоянно развивается и улучшается. Python не зависит от платформы и прост для интеграции с другими языками программирования. Появляются новые библиотеки машинного обучения, получают развитие старые, упрощается синтаксис самого языка, повышается его эффективность. Большое количество готовых к использованию библиотек это ощутимое преимущество, которое может помочь ускорить разработку и сэкономить ресурсы, поскольку позволяет применять готовые решения вместо их создания с нуля. Принимая во внимание все вышеперечисленные преимущества, команда проекта остановила свой выбор на языке программирования Python.

Для подготовки данных можно использовать большое количество библиотек, как связанных с нейронными сетями, так и просто предназначенных для обработки данных. Одной из крупнейших таких библиотек является Pandas. Она предназначена для анализа и обработки данных и представляет собой быстрый, гибкий и мощный инструмент, достаточно широко распространённый в задачах работы с данными.

Другим немаловажным инструментом обработки данных и приведения их в пригодный для нейросети вид является библиотека NumPy, нацеленная на математические вычисления и в том числе поддерживающая многомерные массивы, или тензоры, которые часто используются при работе с нейронными сетями.

При создании нейронных сетей выбор библиотеки машинного обучения является не менее значимым, чем выбор самого языка программирования. Для Python существует множество подходящих модулей, что и было одной из основных причин его выбора. Один из них - [TensorFlow](https://www.tensorflow.org/), библиотека сквозного машинного обучения. [TensorFlow](https://www.tensorflow.org/) позволяет строить глубокие нейронные сети, которые успешно справляются с задачами распознавания образов и рукописного текста, а также обработкой естественных языков. Эта библиотека имеет отличную архитектурную поддержку, позволяющую с легкостью производить вычисления на самых разных платформах, в том числе на десктопах, серверах и мобильных устройствах.

Отдельной частью библиотеки TensorFlow, также способной помочь при решении задачи, является Keras. Этот расширяющий интерфейс, увеличивающий модульность других библиотек, может быть интегрирован с большинством библиотек помимо TensorFlow, к примеру с Deeplearning4j, MXNet, Microsoft Cognitive Toolkit, Theano. В этой библиотеке реализованы практически все автономные модули нейронной сети, включая оптимизаторы, нейронные слои, функции активации слоев, схемы инициализации, функции затрат и модели регуляризации. Это позволяет строить новые модули нейросети, просто добавляя функции или классы.

Однако выбор команды проекта остановился на [PyTorch](https://pytorch.org/), библиотеке машинного обучения Python. Главное ее преимущество – адаптированность к графическому процессору, которая обеспечивает оптимизацию и масштабирование распределенных задач обучения как в области исследований, так и в области создания ПО. Также PyTorch является очень удобным с практической точки зрения инструментом для построения глубоких нейронных сетей. В библиотеке есть компилятор машинного обучения под названием Glow, который кратно повышает производительность фреймворков глубокого обучения. Более того, эта библиотека позволяет полноценно подготовить данные, что гарантирует совместимость её методов и отсутствие необходимости дополнительный конвертаций данных.

Таким образом, для реализации нейронной сети был выбран язык программирования Python и библиотека машинного обучения PyTorch.

Глава 2. Проектирование

1. Архитектура Unet

Анализ литературы показал, что наиболее эффективным решением в задачах сегментации медицинских изображений являются сверточные нейронные сети архитектуры Unet. Впервые такой способ организации сверточных слоев был предложен учеными из Университета Фрайбурга в 2015 [4]. В оригинальной статье тривиальная полносвязная сверточная нейронная сеть была разделена на две части: сверточную и разверточную. В первой части для каждого изображения последовательно удваивается количество каналов признаков при одновременном сжатии размеров изображения также в 2 раза. Достигается это благодаря особому построению слоев, которые идут блоками. Каждый блок содержит два последовательно идущих сверхточных слоя (ядро свертки 3x3), после которых идет слой с функцией активации ReLU и подвыборкой (пулингом) с функцией максимума 2×2 с шагом 2, которая отвечает за увеличение карты признаков.

Разверточная часть также построена из блоков, выполняющих противоположные блокам сверточной части действия: слой, обратный подвыборке, выполняет транспонирование сверточного слоя, таким образом имитируя операцию деконволюции, что расширяет карту признаков, после следует свертка 2×2, которая уменьшает количество каналов признаков.

После идет конкатенация с соответствующим образом обрезанной картой признаков из сжимающего пути и две свертки 3×3, после каждой из которой идет ReLU. Обрезка нужна из-за того, что мы теряем пограничные пиксели в каждой свертке. На последнем слое свертка 1×1 используется для приведения каждого 64-компонентного вектора признаков до требуемого количества классов. Сеть состоит из 23 сверточных слоев. Схематическое представление архитектуры представлено на рисунке 1.

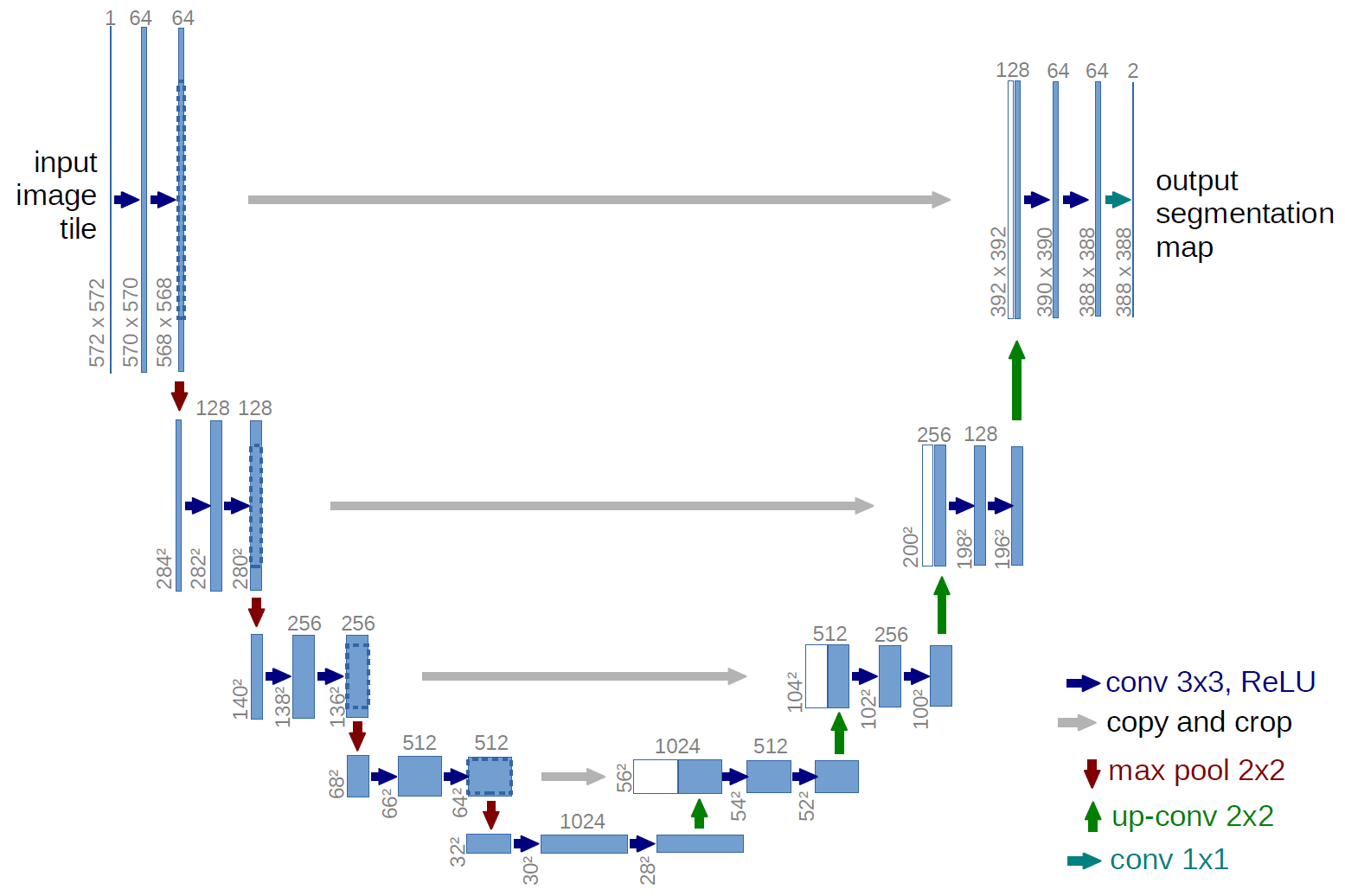


Рисунок 1 - Архитектура Unet

Такие модификации архитектуры обусловлены тем, что позволяют обучаться на меньшем количестве примеров и выполнять более точную сегментацию.

1. Проектирование нейросети

Оригинальная архитектура имеет ряд своих недостатков. В частности, размерность изображения - результата работы сети (бинарной маски для исходного изображения) составляет 388×388, в то время как входного (снимка, на основе которого делается предсказание) - 572×572. Такое сжатие происходит из-за использования ядра свертки 3×3, которое не делает предсказаний для крайних пикселей изображения, так как у тех нет окружающих их пикселей с одной из сторон (крайние пиксели никогда не оказываются в центре ядра, потому что тогда ядру не над чем будет скользить за краем). В нашей задачи принципиально важно, чтобы размерности снимка МРТ и полученной для него маски совпадали, потому что в противном случае пришлось бы интерполировать значения маски, поэтому границы очагов были бы определены менее точно, что повлияло бы на качество результатов нейросети. Для сохранения размерностей необходимо применить паддинг (нулевое дополнение). Паддинг добавляет к краям “поддельные” нулевые пиксели, таким образом крайние значения пикселей могут оказаться в центре ядра свертки. Таким образом, размерность изображения искусственно увеличивается на 2 в каждом из измерений, после выполнения операции свертки размерность результирующей матрицы уменьшается на 2 в каждом из измерений, таким образом исходная и итоговая матрицы имеют одинаковую размерность.

Другим существенным отличием оригинальной архитектуры является разница в количестве каналов свойств. На вход классического Unet поступает изображение с 1 каналом свойств, на выходе получается двухканальное изображение. В нашей задаче входным изображением является черно-белый снимок МРТ (таким образов, количество каналов свойств составляет 1), на выходе необходимо сегментирующая бинарная маска (количество каналов свойств также 1).

Таким образом, сверточная часть нашей нейросети преобразовывает исходное изображение размером 256×256 с 1 каналом свойств в изображение 16×16 с 1024 каналами свойств, на каждом из 5 шагов сжимая размер изображения в 2 раза (256-256-128-64-32-16) и увеличивая количество каналов свойств также в 2 раза на всех шагах, кроме первого (1-64-128-256-512-1024). Развертывающая часть получает на вход результат работы предыдущего модуля - изображение 16×16 с 1024 каналами свойств и преобразовывает его к изображению 256×256 с 1 каналом свойств, на каждом из 4 шагов увеличивая размер изображения в 2 раза (16-32-64-128-256) и уменьшая количество каналов свойств также в 2 раза на всех шагах, кроме последнего (1024-256-128-64-1).

1. Метрики точности

Для оценки качества модели машинного обучения необходимо выбрать единую метрику, которая позволит получить представления о качестве обученной модели. Самой очевидной, но далеко не самой точной является субпиксельная точность. Она равна проценту правильно классифицированных пикселей от общего количества пикселей на изображении. Такая метрика кажется достаточно точной, однако в задачах сегментации субпиксельная точность очень неустойчива к проблеме классового дисбаланса. Так называют ситуацию, когда классы чрезвычайно несбалансированы, то есть один или несколько классов доминируют по занимаемой площади, а другие классы занимают лишь незначительную часть изображения. Например, большинство снимков МРТ, анализируемых на предмет наличия очагов инсульта, вообще не имеют следов кровоизлияния, и даже на снимках МРТ с крупными очагами большая часть изображения также относится к не травмированной мозговой ткани. Таким образом, если нейросеть будет на любой снимок отвечать полностью нулевой матрицей (то есть делать вывод, что очага кровоизлияния на МРТ нет), точность ее работы, согласно данной метрике, составит более 90%. Однако критически важными являются именно снимки с поражениями мозговой ткани, а цена ошибки в случае ложноотрицательного результата во много раз превосходит цену ошибки для ложноположительного результата. Поэтому использование метрики субпиксельной точности является контрпродуктивным в задачах сегментации, особенно медицинских изображений.

Более продвинутой метрикой с точки зрения объективности получаемых результатов считается мера Жаккара (коэффициент флористической общности, Intersection-Over-Union, Jaccard Index), вычисляемая по формуле

где A - множество сегментированных системой пикселей, B - множество пикселей для выделения на изображении. Таким образом, мера Жаккара характеризует отношение области перекрытия между прогнозируемой сегментацией и истиной, разделенная на область объединения между прогнозируемой сегментацией и истиной. Этот показатель находится в диапазоне от 0 до 1 (0–100%), где 0 означает отсутствие перекрытия, а 1 означает полностью перекрывающую сегментацию. Такой подход эффективно решает проблему классового дисбаланса, а потому является простым, но чрезвычайно эффективным показателем в задачах сегментации изображений.

Другой показатель точности работы нейросети, во многом схожий с мерой Жаккара, это мера Сёренсена (мера игральной кости, Dice Coefficient). Данная метрика вычисляется по формуле

где A - множество сегментированных системой пикселей, B - множество пикселей для выделения на изображении. Мера Сёренсена очень похожа на меру Жаккара: они положительно коррелированы, то есть, если один говорит, что модель A лучше, чем модель B при сегментации изображения, то другой скажет то же самое. Как и мера Жаккара, мера Сёренсена находится в диапазоне от 0 до 1, где 1 означает наибольшее сходство между предсказанным и истинным.

Меры Жаккара и Сёренсена на сегодняшний день наиболее популярны в задачах семантической сегментации, при том, что не имеют критических различий и показывают сопоставимые результаты. Для создания же нашей системы была выбрана мера Сёренсена.

Глава 3. Разработка, тестирование, внедрение программы

1. Оценка результатов обучения

По результатам обучения и настройки гиперпараметров нейросети, точность работы наибольшей из них составила 53% (по метрике Dice Coefficient). Такие показатели были достигнуты при сочетании следующих гиперпараметров: функция активации ReLu, функция потери BinaryCrossEntropyWithLogitsLoss, оптимизатор Adam, размер батча 5, количество эпох 50. Обучение сети заняло 2 часа 1 минуту.

Нейросеть – бинарный сегментатор с такими показателями точности работы можно считать достаточно эффективной. Например, сопоставимые результаты были получены в исследовании [12] (Dice Coefficient: 63,5%). Авторы статьи использовали Unet для сегментации глиомы и рассеянного склероза по снимкам МРТ. Нейросеть была обучена на основе 93 наборах изображений, что объясняет более высокие показатели точности.

Также возможно сравнение результатов нашей модели с исследованием [13], в котором авторы изучали возможность сегментации КТ снимков полносвязной нейросетью с использованием 103 наборов изображений. Согласно метрике игральной кости, точность ее работы составила 44%.

Другим интересным для нас исследованием является статья [14]. Авторы исследования занимались проблематикой сегментации очагов инсульта на мультиспектральных МРТ снимках и разработали сверточную нейросеть с точностью работы 64,5% по метрике Dice Coefficient.

Таким образом, результаты обучения нашей нейросети сопоставимы с приведенными в аналогичных исследованиях, поэтому мы можем считать поставленную перед проектом цель выполненной.

1. Стратегии повышения точности работы нейросети
2. Расширение набора данных для обучения

Говоря о путях усовершенствования нейросети, невозможно не отметить вот какой факт: часто приходится слышать о сегментирующих нейросетях с точностью выше 95%, например, для автопилотируемых автомобилей. Такие нейросети обучаются на десятках тысяч изображений, что обуславливает высокую точность сегментации даже при не самых эффективных архитектурах. Однако для медицинских изображений собрать датасет даже из тысячи изображений зачастую представляется невозможным из-за недостатка источников изображений, а также сложностями с конфиденциальностью используемых данных. Недостаточное количество примеров для обучения может также порождать переобучение модели, что также негативно сказывается на точности работы нейронной сети.

Тем не менее, существует способ увеличения количества изображений в наборе путем создания дополнительных данных из уже имеющихся - так называемая аугментация. Новые данные могут быть получены в результате отражения по горизонтали, случайного кадрирования, изменения цвета, замены фона, размытия, добавление дополнительных линий на изображения, сжатий и растяжений вдоль осей, дефокусировки и многих других манипуляций. Также возможно применение различных комбинации, например, одновременно выполнение поворота и случайного масштабирования.

1. Предобученный энкодер

Одним из неочевидных способов повышения эффективности нейронной сети типа Unet может быть улучшение показателей сверточной части (энкодера) путем использования уже предобученной на большом наборе данных независимой нейросети. Такая модель не только производит вычисления быстрее, чем обучающаяся в процессе, но и показывает более высокие показатели точности сегментации, что подтверждается исследованиями. Например, в статье [10] авторы сравнивают три схемы инициализации весов: LeCun, энкодер с весами из VGG11 и полную сеть, обученную на наборе данных Carvana.

1. Перспективы использования

В результате проделанной работы мы получили модель нейронной сети, способной бинарно сегментировать изображения. Она продемонстрировала достаточно высокие показатели точности после обучения на снимках МРТ головного мозга, где были выделены очаги кровоизлияния. Но при тренировке на другом наборе изображений, например, с выделенными следами церебральной болезни мелких сосудов, мы можем ожидать также достаточно высокие результаты. Архитектура сети не зависит от того, какие именно снимки поступаю на вход системы; модель может быть обучена бинарной классификации любых объектов, поэтому разработанная модель сети может быть применения при решения любых других задач бинарной сегментации медицинских изображений.

Полученная модель успешно справляется с задачей сегментации снимков МРТ на очаги кровоизлияния и остальное. Как и предполагалось, это позволяет многократно сократить время постановки диагноза, однако, что более значимо, эта нейросеть может быть использована для дальнейших исследований в области диагностики инсультов. Например, выходные результаты этой нейронной сети могут стать основой для другой нейронной сети, определяющей на основе локализации и объемов кровоизлияния тип инсульта на снимке. Задача установления вида инсульта является ключевой при выборе стратегии лечения и дальнейшей реабилитации, поэтому такая автоматизированная система будет очень востребована как для специалистов - практикующих врачей, так и для ученых - исследователей.

Заключение

В результате проведенной работы был разработан обработчик и загрузчик данных для программы, сегментирующая снимок МРТ на область, относящуюся к очагу кровоизлияния, и остальную часть мозга. Для этого был проведен анализ литературы, изучены исследования со схожей проблематикой и современные методы и стратегии решения проблемы. На массиве изображений по 20–25 снимков для каждого из 51 пациентов была обучена и протестирована нейронная сеть архитектуры Unet, показатель точности работы – 53%, что сопоставимо с результатами исследований со схожей проблематикой.

Гипотезу исследования о том, что, если разработать автоматизированную систему для оценки локализации и объема кровоизлияния по снимкам МРТ, то такая система позволит сократить время постановки диагноза и повысит качество оказываемой медицинской помощи, можно считать подтвержденной: действительно, система работает гораздо быстрее человека, показывая при этом сопоставимую точность сегментации изображения.

Полученный загрузчик данных доступен для использования также и в других задачах, а использованные методики могут быть полезны и при отсутствии доступа к исходному коду. Таким образом, загрузчик доступен для интеграции в любую нейронную сеть для дальнейшего использования и может быть использовать для оценки здоровья пациента на основе медицинских изображений, например, установлении типа инсульта по снимку МРТ,

Библиографический список

1. **Пирадов М. А., Максимова М. Ю., Танашян М. М.** Инсульт: пошаговая инструкция. Руководство для врачей 2-е изд., переработанное и дополненное. **М.: ГЭОТАР-Медиа Москва, 2020. 288 с.**
2. Депутат Е. В., Скобцов Ю. А. Сегментация медицинских изображений с помощью алгоритма муравьиных колоний // Информационные управляющие системы и компьютерный мониторинг 2010. Сборник материалов к I Всеукраинской научно-технической конференции студентов, аспирантов и молодых ученых. Донецк, ДонНТУ-2010. С.12-16.
3. Модель обучаемого активного контура для сегментации гистологических изображений / ХВОСТИКОВ А.В. [и др.] // [Научная Визуализация](https://www.elibrary.ru/contents.asp?id=41857424). 2019. № 3. С. 64-75.
4. Ronneberger, Olaf; Fischer, Philipp; Brox, Thomas. U-Net: Convolutional Networks for Biomedical Image Segmentation // Medical Image Computing and Computer-Assisted Intervention (MICCAI), Springer, LNCS, 2015. Vol.9351. PP 234—241.
5. Bumshik Lee, Nagaraj Yamanakkanavar, Jae Young Choi. Automatic segmentation of brain MRI using a novel patch-wise U-net deep architecture // The PLOS ONE Staff (2021) Correction: Automatic segmentation of brain MRI using a novel patch-wise U-net deep architecture. PLOS ONE 16(1), August 2020. PP. 236 – 241.
6. Automated hematoma segmentation and outcome prediction for patients with traumatic brain injury / [HemingYao](https://www.sciencedirect.com/science/article/pii/S093336571931317X" \l "!) [и др.] // [Artificial Intelligence in Medicine](https://www.sciencedirect.com/science/journal/09333657), July 2020. [Vol. 107](https://www.sciencedirect.com/science/journal/09333657/107/supp/C).
7. CT Image Segmentation in Traumatic Brain Injury / S.M.R. Soroushmehr [и др.] // 2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), 2015. PP. 2973-2976.
8. Neuroimaging and deep learning for brain stroke detection - A review of recent advancements and future prospects / [R Karthik](https://pubmed.ncbi.nlm.nih.gov/?term=Karthik+R&cauthor_id=32882591) [и др.] // Computer Methods and Programs in Biomedicine, December 2020. Vol.197.
9. Джимми У. Ки. - Искусственные нейронные сети управления технологическими процессами. Часть 2 //  [Сontrol Engineering Россия, Август 2016](https://controlengrussia.com/magazine/control-engineering-rossiya-avgust-2016/). № 4. С. 106-110.
10. [Vladimir Iglovikov](https://arxiv.org/search/cs?searchtype=author&query=Iglovikov%2C+V), [Alexey Shvets](https://arxiv.org/search/cs?searchtype=author&query=Shvets%2C+A) - TernausNet: U-Net with VGG11 Encoder Pre-Trained on ImageNet for Image Segmentation // [Cornell University website](https://www.cornell.edu/)

[https://arxiv.org/abs/1801.05746 Просмотрено 20.06.2021](https://arxiv.org/abs/1801.05746%20Просмотрено%2020.06.2021)

1. Karthik, R., Menaka, R., Johnson, A., & Anand, S. Neuroimaging and Deep Learning for Brain Stroke Detection. A Review of Recent Advancements and Future Prospects. // Computer Methods and Programs in Biomedicine, LNCS, 2020. Vol.9351. PP. 234—241.
2. J. Dolz, I.B. Ayed, C. Desrosiers, Dense multi-path U-Net for ischemic stroke lesion segmentation in multiple image modalities, brainlesion: glioma, multiple sclerosis. // Stroke Trauma. Brain Inj. Lect. Notes Comput. Sci. (2019) PP. 271–282.
3. S.M. Abulnaga, J. Rubin, Ischemic stroke lesion segmentation in CT perfusion scans using pyramid pooling and focal loss. // Stroke Trauma. Brain Inj. Lect. Notes Comput. Sci. (2019) PP. 352–363.
4. Z. Liu, C. Cao, S. Ding, Z. Liu, T. Han, S. Liu. Towards clinical diagnosis: automated stroke lesion segmentation on multi-spectral MR image using convolutional neural network. // IEEE Access. 6 (2018) PP. 570–589.

ПРИЛОЖЕНИЕ A

Код программы нейронной сети

import torch

import torch.nn as nn

import torch.optim as optim

import torch.utils.data.dataset as dset

from torch.utils.data.sampler import SubsetRandomSampler, Sampler

import torchvision

import matplotlib.pyplot as plt

import numpy as np

class Block(nn.Module):

def \_\_init\_\_(self, in\_ch, out\_ch):

super().\_\_init\_\_()

self.conv1 = nn.Conv2d(in\_ch, out\_ch, 3, padding=1)

self.relu = nn.ReLU()

self.conv2 = nn.Conv2d(out\_ch, out\_ch, 3, padding=1)

def forward(self, x):

tmp = self.conv1(x)

tmp = self.relu(tmp)

tmp = self.conv2(tmp)

return self.relu(tmp)

class Encoder(nn.Module):

def \_\_init\_\_(self, chs=(1, 64, 128, 256, 512, 1024)):

super().\_\_init\_\_()

self.enc\_blocks = nn.ModuleList([Block(chs[i], chs[i + 1]) for i in range(len(chs) - 1)])

self.pool = nn.MaxPool2d(2)

def forward(self, x):

ftrs = []

for block in self.enc\_blocks:

x = block(x)

ftrs.append(x)

x = self.pool(x)

return ftrs

class Decoder(nn.Module):

def \_\_init\_\_(self, chs=(1024, 512, 256, 128, 64)):

super().\_\_init\_\_()

self.chs = chs

self.upconvs = nn.ModuleList([nn.ConvTranspose2d(chs[i], chs[i + 1], 2, 2) for i in range(len(chs) - 1)])

self.dec\_blocks = nn.ModuleList([Block(chs[i], chs[i + 1]) for i in range(len(chs) - 1)])

def forward(self, x, encoder\_features):

for i in range(len(self.chs) - 1):

x = self.upconvs[i](x)

x = torch.cat([x, encoder\_features[i]], dim=1)

x = self.dec\_blocks[i](x)

return x

class UNet(nn.Module):

def \_\_init\_\_(self, enc\_chs=(1, 64, 128, 256, 512, 1024), dec\_chs=(1024, 512, 256, 128, 64), num\_class=1):

super().\_\_init\_\_()

self.encoder = Encoder(enc\_chs)

self.decoder = Decoder(dec\_chs)

self.head = nn.Conv2d(dec\_chs[-1], num\_class, 1)

def forward(self, x):

enc\_ftrs = self.encoder(x)

out = self.decoder(enc\_ftrs[::-1][0], enc\_ftrs[::-1][1:])

out = self.head(out)

return out

def get\_balanced\_subset(dset):

with\_lesion\_ind = dset.y.sum(axis=0).sum(axis=0) != 0

lesion\_indices = np.arange(len(with\_lesion\_ind))[with\_lesion\_ind]

return torch.utils.data.Subset(dset, lesion\_indices)

def compute\_accuracy(model, loader):

model.eval()

accuracy = 0

i = 0

for (x, y) in loader:

x = x[:, None, ...]

y = y[:, None, ...]

prediction = model(x)

accuracy += dice\_loss(prediction, y)

i += 1;

return accuracy / i

def dice\_loss(pred, answ):

accuracy = 0

intersection = (pred \* answ).sum(-1).sum(-1)

a\_sum = pred.sum(-1).sum(-1)

b\_sum = answ.sum(-1).sum(-1)

accuracy += (2. \* intersection) / (a\_sum + b\_sum)

return torch.Tensor.item(accuracy.sum(0))

def train\_model(model, train\_loader, val\_loader, loss, optimizer, scheduler, num\_epochs):

loss\_history = []

train\_history = []

val\_history = []

for epoch in range(num\_epochs):

model.train()

loss\_accum = 0

for i\_step, (x, y) in enumerate(train\_loader):

x = x[:, None, ...]

y = y[:, None, ...]

prediction = model(x)

loss\_value = loss(prediction, y)

optimizer.zero\_grad()

loss\_value.backward()

optimizer.step()

loss\_accum += loss\_value

ave\_loss = loss\_accum / (i\_step + 1)

train\_accuracy = dice\_loss(prediction, y)

val\_accuracy = compute\_accuracy(model, val\_loader)

loss\_history.append(float(ave\_loss))

#train\_history.append(train\_accuracy)

#val\_history.append(val\_accuracy)

print("Average loss: %f, LR: %f, Train accuracy: %f, Val accuracy: %f" % (ave\_loss, step\_lr.get\_last\_lr()[0], train\_accuracy, val\_accuracy))

return loss\_history, train\_history, val\_history

def save\_chekpoint(model, optimizer, filename):

state = {

"state\_dict": model.state\_dict(),

"optimizer":optimizer.state\_dict(),

}

torch.save(state, filename)

the\_x = torch.load('/content/drive/MyDrive/Colab Notebooks/data.pt')

the\_y = torch.load('/content/drive/MyDrive/Colab Notebooks/label.pt')

data\_size = the\_x.data.shape[0]

test\_split = .2

split = int(np.floor(test\_split \* data\_size))

indices = list(range(data\_size))

np.random.shuffle(indices)

train\_indices, test\_indices = indices[split:], indices[:split]

the\_train\_x = the\_x[train\_indices]

the\_train\_y = the\_y[train\_indices]

full\_train\_dataset = NiftiDataset(the\_train\_x, the\_train\_y)

train\_dataset = get\_balanced\_subset(full\_train\_dataset)

the\_test\_x = the\_x[test\_indices]

the\_test\_y = the\_y[test\_indices]

test\_dataset = NiftiDataset(the\_test\_x, the\_test\_y)

batch\_size = 5

data\_size = len(train\_dataset)

validation\_split = .2

split = int(np.floor(validation\_split \* data\_size))

indices = list(range(data\_size))

np.random.shuffle(indices)

train\_indices, val\_indices = indices[split:], indices[:split]

train\_sampler = SubsetRandomSampler(train\_indices)

val\_sampler = SubsetRandomSampler(val\_indices)

train\_loader = torch.utils.data.DataLoader(train\_dataset, batch\_size=batch\_size,

sampler=train\_sampler)

val\_loader = torch.utils.data.DataLoader(train\_dataset, batch\_size=batch\_size,

sampler=val\_sampler)

sample, label = train\_dataset[0]

nn\_model = nn.Sequential(UNet())

optimizer = optim.Adam(nn\_model.parameters(), lr=1e-4)

loss = nn.BCEWithLogitsLoss()

step\_lr = optim.lr\_scheduler.StepLR(optimizer, step\_size=2, gamma=0.5)

one\_batch = next(iter(train\_loader))

other\_batch = next(iter(train\_loader))

very\_other\_batch = next(iter(train\_loader))

loss\_history, train\_history, val\_history = train\_model(nn\_model, [next(iter(train\_loader)) for i in range(7)], val\_loader, loss, optimizer, step\_lr, 50)

save\_chekpoint(nn\_model, optimizer, 'model1.pth.tar')