**Министерство Просвещения Российской Федерации**

**Муниципальное автономное общеобразовательное учреждение "Лицей №14 имени Заслуженного учителя Российской Федерации А.М. Кузьмина"**

**ИНДИВИДУАЛЬНЫЙ ПРОЕКТ**

**Мобильное приложение выявления пневмонии на рентгеновских снимках легких**

**Выполнила:**

Учащаяся 10 класса «К»

Дудкина София Сергеевна

**Руководитель:**

Слезин Кирилл Анатольевич

Учитель информатики

**Тамбов, 2022**

Оглавление

[**Введение** 3](#_Toc100391266)

[**Изучение методов создания нейронных сетей** 4](#_Toc100391267)

[**Сбор данных для обучения** 6](#_Toc100391268)

[**Реализация нейронной сети** 7](#_Toc100391269)

[**Создание мобильного приложения** 8](#_Toc100391270)

[**Интеграция нейронной сети в мобильное приложение** 10](#_Toc100391271)

[**Заключение** 11](#_Toc100391272)

[**Список литературы** 12](#_Toc100391273)

**Введение**

Человеческий организм ежедневно подвергается воздействию опасных вирусов и бактерий, из-за чего человек зачастую заболевает и обращается к врачу. Бывают случаи, когда одних знаний врачебного персонала недостаточно, допускаются ошибки, которые могут стоить жизни человека. Для постановки верного диагноза иногда необходимо провести дополнительные исследования организма, например рентгенографию. В частности, рентгенография легких является наиболее эффективным методом выявления пневмонии. В связи с пандемией COVID-19 без того нелегкая задача осложнилась увеличением объемов работы, нехваткой времени и специалистов. Без решения данная проблема несет угрозу для жизней многих людей.

Сейчас сложно представить современный мир без технологий. Каждый день люди пользуются различными программами и интернет-ресурсами. Сфера IT стремительно развивается, внедряясь в жизнь человека. В том числе, нейронные сети, которые сегодня можно встретить повсеместно.

Технологии машинного обучения не стоят на месте. Все чаще и чаще нейронные сети используются на современных предприятиях и в повседневной жизни. Технология давно используется во многих приложениях для быстрого анализа аудио и изображений. Нейронным сетям можно найти применение повсеместно, в том числе в оптимизации медицинских задач. Точнее, можно использовать распознавание изображений для анализа рентгенограмм. Это позволит ускорить процесс постановки диагноза, а при совместной работе нейросети и врача может возрасти и точность постановки диагноза.

Появляется идея использования прогресса в развитии нейронных сетей для решения проблем в медицинской сфере. А именно, применить навыки классификации изображений при помощи нейросетей для выявления отклонений на рентгеновских снимках легких.

Исходя из вышесказанного, была выбрана цель: разработка мобильного приложения выявления пневмонии на рентгеновских снимках лёгких. Для достижения поставленной цели необходимо выполнить задачи: изучить существующие методы создания нейронных сетей, произвести сбор данных для обучения, реализовать нейронную сеть, создать приложение, интегрировать нейронную сеть в приложение.

**Изучение методов создания нейронных сетей**

Нейронные сети способны решать множество различных задач, для каждой из которых существует свой вид нейронных сетей. Свёрточные нейронные сети (*convolutional neural networks, CNN*) применяются для обработки изображений и аудио. Типичным применением CNN является классификация изображений.

Нейронная сеть зачастую состоит из нескольких слоев нейронов, связанных между собой. То есть каждый нейрон следующего слоя связан со всеми нейронами предыдущего слоя, и у каждой связи есть свой весовой коэффициент. Коэффициент будет зависеть от важности определяемого параметра. После все веса суммируются, исходя из чего сеть дает результат.

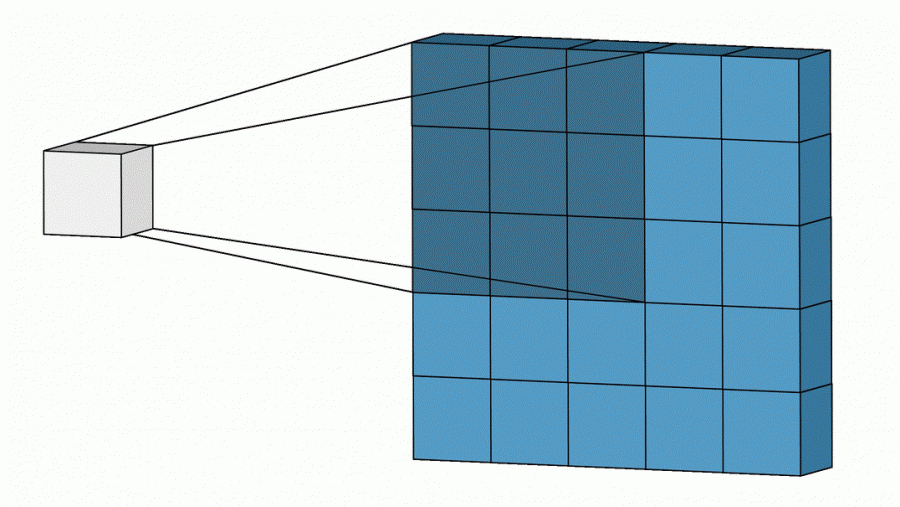
Свёрточные нейронные сети не обрабатывают целое изображение сразу, а постепенно небольшими частями. Если нейронная сеть получает на вход изображение, она не обрабатывает сразу все пиксели. Сначала нейросеть обрабатывает небольшой участок изображения, например квадрат n × nпикселей из верхнего левого угла, затем сдвигает этот квадрат на 1 пиксель и обрабатывает следующий участок (рис. 1).

Рис. 1

Таким образом сеть обрабатывает все изображение. То есть в процессе обработки происходит переход от конкретных особенностей к более абстрактным деталям, выделяя наиболее существенные признаки.

Изначально основным вариантом реализации было написание нейронной сети с нуля. В этом случает необходимо вручную прописывать каждый слой нейронной сети, что является довольно объемным и трудным решением. Также обучение нейронной сети с нуля требует огромных мощностей, много времени и данных.

Изучив больше информации, выяснилось, что наиболее эффективным и простым способом обучения нейросети для классификации изображений является Transfer Learning. Transfer Learning - метод обучения нейронной сети, когда за основу берется готовая модель сети, обученная на большом наборе данных, и навыки этой модели применяются к другой проблеме. Например, можно использовать знания, полученный при решении задачи классификации животных, можно использовать для классификации различных видов транспорта.

Это работает за счет того, что основные слои нейросети фокусируются на обобщенных деталях, а последние слои уже на задачах классификации. Как выяснилось, содержание базовых слоев хорошо применимо для других задач.

Чаще всего, глубокое обучение требует большое количество данных, когда для трансферного обучения достаточно значительно меньших объемов. Также на дообучение нейронной сети под собственные задачи уходит значительно меньше времени и ресурсов.

Таким образом, не приходится проделывать лишнюю работу и изобретать то, что уже существует. Нет необходимости в огромном наборе данных и трате времени.

Для трансферного обучения необходимо использовать уже обученную готовую модель. Одним из самых популярных вариантов является остаточная сеть ResNet50, которая содержит 50 слоев нейронов. Данная сеть обучена на наборе ImageNet, который включает в себя 14 197 122 изображений, разбитых на 21 841 класс (по данным 2017 года). Именно эта сеть и была взята за основу.

**Сбор данных для обучения**

Для обучения был необходим набор данных с большим количеством рентгенограмм легких. Сборка данные вручную могла бы занять много времени, а также найденных ресурсов могло бы не хватить. В связи с этим было необходимо найти сборку фотографий рентгенов легких.

В итоге было найдено довольно популярное готовое решение – набор данных Chest X-Ray Images (Pneumonia) от Kaggle. Данный набор разбит на 3 категории: обучение, тестирование и валидация. Каждая из категорий содержит набор фотографий со здоровыми и зараженными легкими. Суммарно датасет содержит 5856 фотографий, чего должно хватить для обучения.

**Реализация нейронной сети**

Для достижения цели был выбран язык программирования Python, так как является относительно простым и удобным. В последнее время появилось множество удобных фреймворков, позволяющих упростить процесс написание кода, в связи с чем рассматривались библиотеки TensorFlow и Pytorch. Было несколько попыток реализации сети при помощи TensorFlow. Однако данная библиотека оказалась достаточно низкоуровневой, требовала очень много знаний и шаблонного кода. После было принято решение перейти на Pytorch. Pytorch оказался намного проще в освоении и написании кода.

Первым делом было необходимо импортировать нужные библиотеки: torch, copy, torchvision, numpy, time, matplotlib. Далее для построения графика потерь и точности был инициализирован словарь, который хранит данные о потерях и точности за каждую эпоху обучения. Реализована функция train, в которую передается количество эпох для обучения, функция потерь, оптимизатор и модель остаточной сети (ResNet50).

Следующий шаг – разбиение датасета на несколько наборов.

Оптимизатор – алгоритм, который при необходимости корректирует веса и скорость обучения для правильной и быстрой работы модели. Один из наиболее популярных оптимизаторов – это оптимизатор Adam. Он является одним из наиболее эффективных и не требовательных к памяти. Также Adam даёт хорошую точность по сравнению с другими оптимизаторами. Из-за этого выбор сделан в пользу данного оптимизатора.

Для вычисления потерь между прогнозируемым и фактическим значениями применяется функция CrossEntropyLoss. В функцию передаются фактические и прогнозируемые значения, а на выход функция возвращает потери при обучении.

**Создание мобильного приложения**

На данный момент для разработки мобильных приложений используют в основном два языка: Java и Kotlin. Они оба обладают статической типизацией и являются объектно-ориентированными языками. Kotlin является относительно новым языком программирования и более компактным. Код написанный на Kotlin может быть короче на треть. Cогласно исследованиям компании JetBrains (Kotlin Census 2020), наиболее популярным языком программирования для написания мобильных приложений стал Kotlin.

А также согласно статистике Google (I/O 2021), большинство лучших приложений на Android написаны на Kotlin, и более 60% разработчиков используют его. Именно это и стало причиной выбора Kotlin.

Далее было необходимо выбрать среду разработки. Изучив множество вариантов выбор остановился на AndroidStudio и Intellij IDEA. Intellij IDEA является мультиязычной средой, поддерживает работу с GitHub и позволяет настроить нужное окружение. AndroidStudio основано на Intellij IDEA, но AndroidStudio адаптировано именно под Android и имеет больее понятный и удобный интерфейс. Поэтому выбор был сделан в пользу AndroidStudio.

Изначально было необходимо изучить документации по выбранному языку программирования и уже после приступить к созданию приложения.

Разработка мобильного приложения началась с написания интерфейса. В интерфейс основного окна были добавлены классы: TextView для вывода полученного результата, ImageView для отображения фотографии рентгена, ImageButton, класс кнопок, при нажатии на которые будет выполняться открытие галереи или камеры для выбора изображения, а также получение результата. После разработки дизайна окна было необходимо каждому элементу присвоить свой id для последующей привязки к ним функций.

Далее в классе MainActivity была написана функция onCreate, которая отвечает за генерацию окна. В ней объявлены кнопки и привязаны к ним функции. При запуске программы вызывается функция onCreate, соответственно появляется основное меню, в котором отображаются кнопки.

Для открытия галереи программа должна проверить есть ли у нее права на доступ к галерее. Если есть доступ к галерее, то при нажатии на кнопку откроется окно выбора фотографии. Если у программы нет прав на открытие галереи, то она должна запросить их у пользователя при помощи встроенной функции requestPermission. После получения положительного ответа от пользователя можно будет выбрать изображение, в противном случае программа продолжит запрашивать права для доступа к галерее. За открытие галереи для выбора фотографий отвечает функция chooseImageGallery, которая передает код запроса в startActivityForResult, где при выборе фотографии, она отображается в окне ImageView и сохраняется в переменную imageBitMap. Аналогично и с открытием камеры.

После того, как пользователь выбрал фотографию, при нажатии кнопки getResult происходит вызов функции outputGeteration, где происходит открытие нейронной сети. Далее необходимо конвертировать изображение в тензор. Это делается при помощи функции fromBitmap из модуля tensorflow, класса TensorImage. После того как было получен тензор, его можно пропустить через нейронную сеть. Это делает функция process, в которую передается сам тензор, и на выходе получается список, содержащий категории, которые распознала нейронная сеть. Далее этот список сортируется по вероятности вычисления, и из этого списка необходимо взять первое значение, то есть наиболее вероятное, и вывести категорию и вероятность в процентах в resultView. После этого нейронную сеть необходимо закрыть.

В итоге получается функционирующее приложение, соответствующее необходимым требованиям. Полный код приложения находится в приложении 1.

**Интеграция нейронной сети в мобильное приложение**

Как выяснилось в ходе разработки, развертка на Android требует особый формат нейросети - Tensorflow lite. Так как обучение производилось при помощи библиотеки Pytorch необходимо было конвертировать нейросеть в другой формат.

В результате получена нейросеть нужного формата. Необходимо импортировать модель в проект Android приложения. Для получения результата была создана функция outputGenerator, которая принимает анализируемое изображение в формате Bitmap, конвертирует его в тензорное изображение и пропускает через нейронную сеть. В результате мы получаем список категорий и уверенности в полученном результате, который сортируется по уверенности и выбирается наиболее вероятный вариант. Полученный вердикт выводится в TextView.

**Заключение**

В процессе разработки были получены навыки трансферного обучения, основные знания о работе сверточных нейронных сетей, изучен язык программирования для разработки мобильных приложений, также развертка нейросети на Android.

Итак, была разработана нейронная сеть, а также мобильное приложение. Была проведена работа с основными библиотеками для создания нейронных сетей, изучен язык программирования для создания мобильных приложений. Были получены основные знания о работе нейронных сетей. Получен опыт мобильной разработки и машинного обучения. В результате работы было получено мобильное приложения для выявления пневмонии на рентгеновских снимках. Точность выявления около 86%. Это недостаточно для полноценной работы приложения, однако при последующей доработке можно увеличить набор данных, тем самым повысив точность нейронной сети.

**Список литературы**

<https://www.jetbrains.com/research/devecosystem-2018/kotlin/>

https://developer.android.com/kotlin/