Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУ «Информатика и системы управления»

КАФЕДРА ИУ-7 «Программное обеспечение эвм и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

НА ТЕМУ:

«Классификация методов анализа медицинских изображений на основе машинного обучения»

Студент **ИУ7И-77Б** _____ **Тэмуужин Я.**

Руководитель _____ **Строганов Ю. В.**

2023 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ

Заведующий кафедрой ИУ-7
(Индекс)
_____ И. В. Рудаков
(И.О.Фамилия)
«16» сентября 2022 г.

ЗАДАНИЕ
на выполнение научно-исследовательской работы

по теме

«Классификация методов анализа медицинских изображений на основе машинного обучения»

Студент группы **ИУ7И-77Б**

Тэмуужин Янжинлхам

Направленность НИР

учебная

Источник тематики

НИР кафедры

График выполнения НИР: 25% к 6 нед., 50% к 9 нед., 75% к 12 нед., 100% к 15 нед.

Техническое задание

Провести обзор существующих методов анализа медицинских изображений на основе машинного обучения. Сформулировать критерии сравнения методов анализа медицинских изображений на основе машинного обучения. Классифицировать существующие методы анализа медицинских изображений на основе машинного обучения.

Оформление научно-исследовательской работы:

Расчетно-пояснительная записка на **12-20** листах формата А4.
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т. п.)
Презентация на **6-10** слайдах.

Дата выдачи задания «16» сентября 2022 г.

Руководитель НИР _____ **Строганов Ю. В.**
(Подпись, дата) (И.О.Фамилия)

Студент _____ **Тэмуужин Я.**
(Подпись, дата) (И.О.Фамилия)

Оглавление

Введение

Медицинская визуализация играет решающую роль в диагностике и лечении различных заболеваний. Машинное обучение стало мощным инструментом анализа медицинских изображений, помогающие врачам-специалистам принимать более точные и объективные диагностические решения. В последние годы было разработано множество методов, основанных на глубоком обучении, и они достигли самых современных результатов в различных задачах анализа медицинских изображений.

Цель данной работы – провести обзор методов анализа медицинских изображений на основе машинного обучения.

Для достижения поставленной цели необходимо решить следующие задачи:

- изучить предметную область;
- рассмотреть базовые понятия, используемые при проектировании методов машинного обучения;
- провести анализ основных методов анализа медицинских изображений;
- сравнить методы.

1 Анализ предметной области

1.1 Задачи анализа медицинских изображений

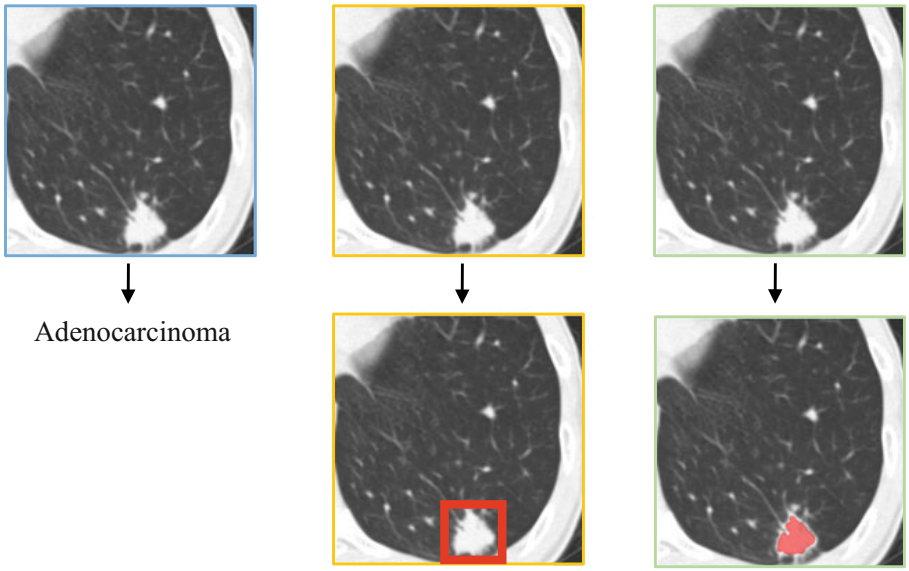

Рисунок 1.1 – Основные задачи анализа медицинских изображений: классификация, обнаружения и сегментация [1].

Существуют различные задачи, относящиеся к анализу медицинских изображений, основными являются следующие.

– **Классификация** была одной из первых областей анализа медицинских изображений, в которых машинное обучение внесло большой вклад [2]. Модели-классификаторы направлены на то, чтобы отличить злокачественные образования от доброкачественных, либо на идентификацию (наличие или отсутствие) определенных заболеваний по входным изображениям. Традиционно в моделях классификации применили обычные алгоритмы машинного обучения, как Support Vector Machine, Random Forest, Decision Tree, использующие признаки, созданные вручную экспертами предметной области. В последние годы, как и в других задачах компьютерного зрения, методы глубокого обучения успешно применяются для классификации, наиболее распространенными являются методы на основе CNN [3].

– **Сегментация** медицинских изображений – это процесс идентификации набора пикселей или вокселей поражений, органов и других субструктур из фоновых областей [2]. Сегментацию также можно рассматривать как задачу классификации всех пикселей изображения на подгруппы [4]. Сегментация позволяет извлекать из изображений детализированную информацию, которую используют для планирования лечения, мониторинга прогрессирования заболевания и улучшения диагностики. Сегментация является одной из самых сложных и трудоемких задач для клиницистов, поэтому наблюдается самое большое разнообразие методологии основанных на машинном обучении. С момента своего представления в 2015 году U-Net [5] стала, самой известной архитектурой для сегментации медицинских изображений. За последние два года сочетание U-Net и Трансформера, как TransUNet и Swin UNETR, способствовало достижению самых современных результатов. Среди всех распространенных задач анализа изображений, таких как классификация и обнаружение, для сегментации требуется наибольшее количество высококачественных аннотаций [6].

– **Обнаружение** (detection) заключается в классификации категории объекта и определении его положения с использованием ограничивающей рамки на изображении [1]. Обнаружение считается менее сложным, чем сегментация изображения, поскольку выполняется только нахождение присутствия объектов, а не отделения и идентификации конкретных структур или областей интереса. Большинство опубликованных систем обнаружения объектов методом машинного обучения по-прежнему используют CNN. В частности широко используется R-CNN (Region Based Convolutional Neural Networks) и её модификации как Faster R-CNN, Mask R-CNN.

– **Регистрация** изображения, процесс совмещения двух или более изображений в одну систему координат с согласованным содержимым, также является важной задачей анализа медицинских изображений [7]. Регистрация часто выполняется для обнаружения изменений или мониторинга поражений на данных одного и того же пациента, снятых в разные моменты времени. Также используется для устранения деформации объекта изображения (из-за дыхания, анатомических изменений и т. д.).

– **Синтез** изображений – процесс искусственного синтеза желаемых изображений по заданному изображению или текстовой информацией, может рассматриваться как процесс, противоположный классификации изображений [4]. Чтобы увеличить количество обучающих примеров для многих других задач анализа медицинских изображений, часто выполняется синтез обучающей выборки, применяя генеративные модели, как GAN [8].

– **Генерация** клинических отчетов для данного изображения с использованием глубокого обучения стало быстрорастущей областью анализа медицинских изображений [9]. Часто для генерации текстового отчета, на результат модели анализа изображения применяется модели NLP, как LSTM, Трансформер и его модификации.

– **Реконструкция** изображения направлена на улучшение качества изображения или преобразование его в форму, более конструктивную и полезную для наблюдателя.

1.2 Медицинские изображения

Медицинская визуализация – это процесс создания визуальных представлений внутренних структур тела для клинического анализа. Существует различные виды методов медицинской визуализации, широко используемыми в анализе машинным обучением являются следующие.

– **Рентгенография (X-ray)** – это технология медицинской визуализации, которая использует рентгеновское излучение для создания изображений внутренних структур тела. Это один из самых распространенных методов диагностики, который позволяет врачу определить местонахождение и размер объектов внутри тела, а также определить их патологические изменения. Рентгенография грудной клетки широко используется в диагностике для выявления патологий сердца и заболеваний легких, таких как туберкулез, ателектаз, консолидация, плевральный выпот, пневмоторакс и гиперкардиальная инфляция. Рентгеновские изображения доступны, недороги и менее дозоэффективные (используется минимальная доза радиации) по сравнению с другими методами визуализации [10].

– **Компьютерной томография (КТ)** – это рентгеновская процедура, которая создает изображения поперечного сечения с помощью компьютерной обработки. КТ-изображения более детализированы, чем обычные рентгеновские изображения, и могут показывать кости, а также мягкие ткани и органы. В обычных рентгеновских снимках рентгеновские лучи посылаются только в одном направлении, а КТ использует моторизованный источник рентгеновских лучей, который испускает узкие пучки рентгеновских лучей при вращении вокруг пациента. Когда рентгеновские лучи проходят через пациента, они улавливаются специальными цифровыми детекторами, расположенные напротив источника, и передаются на компьютер. Срезы изображения могут обрабатываться по отдельности в двумерной форме или складываться вместе для создания трехмерного изображения, которое может выявить аномальные структуры [11]. Области применения включают обнаружение опухолей или поражений в брюшной полости, а также локализацию травм головы, опухолей и тромбов. Они также используются для диагностики сложных переломов костей и опухолей костей [12].

– **Магнитно-резонансная томография (МРТ, MRI)** В МРТ создается сильное магнитное поле, заставляющее протоны в организме выравниваться с этим полем. Когда через пациента пропускается радиочастотный ток, протоны стимулируются и выходят из равновесия, борясь с притяжением магнитного поля. При выключения радиочастотного поля датчики МРТ обнаружат энергию, высвобождаемую при выравнивании протонов с магнитным полем. Время, необходимое протонам для выравнивания с магнитным полем, а также количество высвобождаемой энергии, меняется в зависимости от окружающей среды и химической природы молекул. Врачи могут определить разницу между различными типами тканей на основе этих магнитных свойств [13]. МРТ обычно используются для визуализации некостных или мягких тканей тела. Сравнительные исследования показали, что МРТ лучше показывает головной и спинной мозг, нервы и мышцы, чем КТ [12].

– **Патологические изображения (Pathology)** Патологическая анатомия – научно-прикладная дисциплина, изучающая патологические про-

цессы и болезни с помощью научного, главным образом микроскопического, исследования изменений, возникающих в клетках и тканях организма, органах и системах органов. В области анализа патологических изображений большое внимание уделено анализу *цитологических (cytology)* изображений, которые используются для изучения клеток организмов, поскольку они являются результатом более простых биопсии. *Гистопатология (histopathology, ГП)* – раздел патологии, изучающий процессы в тканях (совокупности клеток и межклеточного вещества) организмов. ГП изображения обеспечивают более подробное представление о заболевании и его влиянии на ткани. В частности, некоторые характеристики заболевания, например лимфоцитарная инфильтрация рака, могут быть выведены только из ГП изображения. ГП диагностика является обязательной для обнаружения множества заболеваний, включая почти все типы рака [14].

1.3 Методы обучения модели

Методы машинного обучения можно разделить на следующие основные категории, в зависимости от характера исходных данных, на которых они обучаются.

1. **Обучение с учителем (Supervised learning)** – это тип машинного обучения, при котором модель обучается на размеченных данных, т.е. желаемый результат предоставляется для заданного ввода. Цель состоит в том, чтобы найти значения параметров модели, при которых значение функции потерь – разность между желаемым и фактическим выходами – является минимальной.

2. **Обучение без учителя (Unsupervised learning)**. При обучении без учителя модели обрабатывают данные без меток и самостоятельно находят закономерности и взаимосвязи во входных данных. Такой тип обучения используется для таких задач, как уменьшение размерности и кластеризация данных.

3. **Обучение с частичным привлечением учителя (Semi–**

supervised learning, SSL). В отличие от обучения без учителя, которое обрабатывает только непомеченные данные для изучения значимых представлений, SSL объединяет помеченные и непомеченные данные. SSL применяется, когда доступно больше непомеченных данных, чем помеченных.

4. **Самообучение (Self-supervised learning)** — это тип обучения без учителя, при котором метки для входных данных создаются из самих немаркированных данных без внешнего контроля. Самообучение может быть создан двумя способами: методами, основанными на *предтекстовых задач (pretext tasks)*, и методами, основанными на *контрастных потерях (contrastive loss)*. Также как SSL, самообучение применяется, когда доступно больше непомеченных данных, чем помеченных.

5. **Трансферное обучение (Transfer learning)** – это метод, при котором модели предварительно обучаются на большом количестве данных с метками, которые могут принадлежать другой области, например, ImageNet. Затем предварительно обученные модели переносятся в целевую область и настраиваются (fine-tuned) с использованием гораздо меньшего количества доступных обучающих примеров.

1.4 Нейронные сети

Нейронные сети являются подмножеством алгоритмов машинного обучения и лежат в основе алгоритмов глубокого обучения. Их название и структура основаны на концепциях, полученных в результате исследований процесса передачи сигналов между нейронами человеческого мозга [15].

Нейронные сети моделируются как наборы нейронов, соединенных в направленный ациклический граф. Вместо аморфного набора связанных нейронов модели сетей организуются в отдельные слои нейронов: входной слой, скрытые слои и выходной слой. Наиболее распространенным типом слоя является **полносвязный слой**, в котором нейроны между двумя соседними слоями полностью попарно связаны, но нейроны в одном слое не имеют общих связей [16]. На рисунке 1.2 приведен пример нейронной сети с полносвязными слоями.

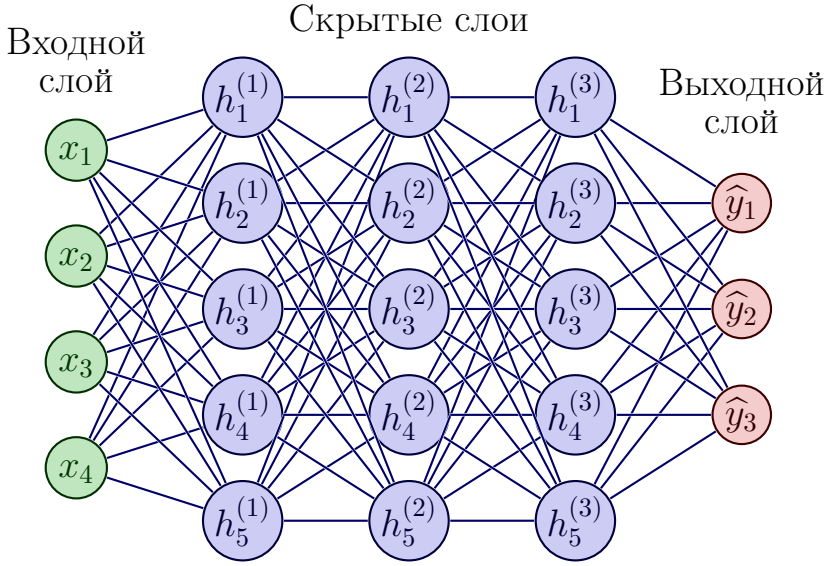

Рисунок 1.2 – Четырехслойная нейронная сеть.

Целью нейронной сети является аппроксимация некоторой функции $y = f(x)$, то есть определить отображение $y = \widehat{f}(x, \theta)$ и найти значение параметров θ, приводящие к наилучшему приближению истинной функции f.

Каждой связи между нейронами соседних слоев присваивается вес w_i, определяющий насколько важен значение нейрона одного слоя для нахождения значения в следущем слое. Также есть смещение или пороговое значение b, размерность которого равен числу нейронов предыдущего слоя или.

Для вычисления значения нейрона скрытого или выходного слоя выполняется скалярное произведение значение узлов предыдущего слоя x_i и их весов w_i, и добавляется смещение b.

$$f(x, W, b) = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b = w^T x + b. \tag{1.1}$$

Например, для однослойной сети, приведенной на рисунке 1.3, значение нейрона в выходном узле равен $\widehat{y} = w_1 x_1 + w_2 x_2 + w_3 x_3 + b = w^T x + b$ и параметрами сети являются $W = \{[w_1, w_2, w_3], b\}$.

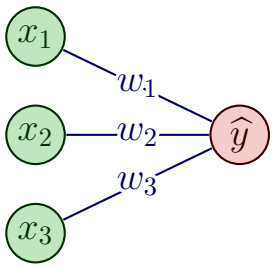

Рисунок 1.3 – Однослойная нейронная сеть.

1.4.1 Функция активации

При вычислении значений скрытого слоя для введения нелинейности в модель используется функция активации или нелинейность (activation function, non-linearity). Значение этих функции часто называют активацией. Широко распространёнными являются следующие.

1. **Логистическая сигмоида (σ)** — монотонно возрастающая нелинейная функция логистическая функция. Большинство значений функции σ будут близки к асимптотам $y = 0$ и $y = 1$, следовательно их градиент почти будет равен 0, что затрудняет обучение на основе градиента. По этой причине в настоящее время в сетях с прямой связью использование σ в качестве функция активации в скрытых слоях не рекомендуется.

$$g(x) = \sigma(x) = \frac{1}{1 + e^{-x}}. \tag{1.2}$$

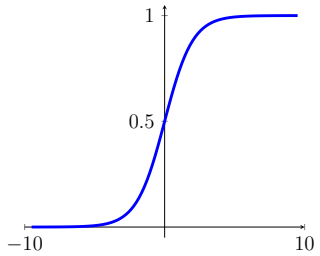

Рисунок 1.4 – Сигмоида.

2. **Гиперболический тангенс** (tanh). Как и у σ, его активация достигает насыщения. Но когда необходимо использовать сигмоидальную функцию активации, tanh обычно работает лучше, чем σ. Поскольку tanh подобен функции тождества вблизи 0, обучение сети $\hat{y} = w^T \tanh(U^T \tanh(V^T x))$ похоже на обучение линейной модели $\hat{y} = w^T U^T V^T x$ при небольших значений, что упрощает обучение сети.

$$g(x) = \tanh(x) = 2\sigma(2x) - 1 = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{1.3}$$

3. **Выпрямленные линейные единицы (Rectified Linear Units, ReLU)** стали использоваться почти во всех сетях глубокого обучения. ReLU зна-

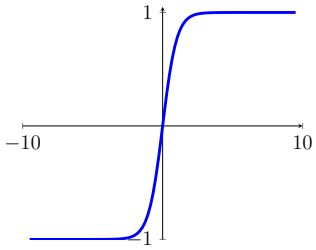

Рисунок 1.5 – Гиперболический тангенс.

чительно ускоряет (например, в 6 раз в AlexNet [17]) сходимость стохастического градиентного спуска по сравнению с сигмовидными функциями σ и tanh. Утверждается, что это связано с её линейной, ненасыщающей формой и простой реализацией. Недостатком данной функции является то, что она может привести к такому обновлению весов, что нейрон никогда больше не активируется. То есть ReLU не может обучаться с помощью методов, основанных на градиенте, при входных данных с активацией 0. Эта проблема может быть решена с помощью правильной настройки скорости обучения (learning rate).

$$g(x) = \max(0, x). \tag{1.4}$$

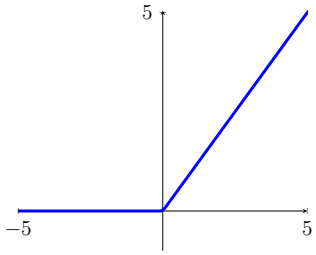

Рисунок 1.6 – ReLU.

4. **ReLU с «утечкой» (Leaky ReLU)** основана на использование ненулевого наклона (угловой коэффициента) α при отрицательных x, что решает описанную выше проблему ReLU. Функция фиксирует α до небольшого значения, например 0,01 или 0,1.

$$g(x, \alpha) = \max(0, x) + \alpha \min(0, x). \tag{1.5}$$

5. **Линейная единица ошибок Гаусса (Gaussian Error Linear Unit, GeLU)**. ReLU детерминистически умножает ввод на ноль или единицу,

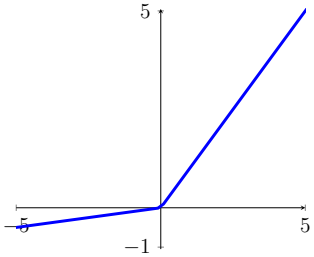

Рисунок 1.7 – ReLU с «утечкой», $\alpha = 0.1$.

a dropout стохастически умножает на ноль. GeLU объединяет эти функциональности ReLU и dropout, умножая ввод на ноль или единицу, но значения этой маски «ноль-единица» определяются стохастически, сохраняя зависимость от входного значения. В частности, x умножается на $\Phi(x)$ – значение функции распределения стандартного нормального распределения при x, то есть x масштабируется на то, насколько он больше, чем другие входные данные.

$$g(x) = x\Phi(x), \tag{1.6}$$

где $\Phi(x)$ вычисляется, используя функцию ошибок (интеграл вероятности) $\operatorname{erf}(x)$, по формуле:

$$\begin{aligned} \Phi(x) &= P(X \leq x), X \sim \mathcal{N}(0,1) \\ &= \frac{1}{2}\left(1 + \operatorname{erf}(x/\sqrt{2})\right). \end{aligned} \tag{1.7}$$

Как показано в [18], результаты моделей с GeLU соответствует или превосходит результаты моделей с ReLU или ELU в различных задачах компьютерного зрения, обработки естественного языка и автоматического распознавания речи.

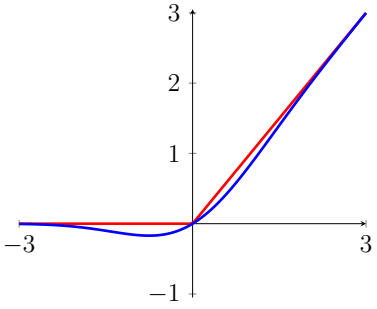

Рисунок 1.8 – GeLU (синий), ReLU (красный).

Значения скрытого слоя модели, представленной на рисунке 1.9, вычисляется по формуле $h = g(W^1 x + b^1)$, где g — функция активации. Выходное значение находится по формуле $\hat{y} = W^2 h + b^2$.

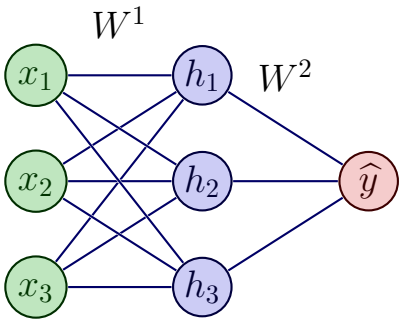

Рисунок 1.9 – Двухслойная нейронная сеть.

1.4.2 Функция потерь

Для оценки параметров моделей и соответсвия между ожидаемым и получившимся данным используется функция потерь (loss, cost function). Потери будут высокими, если прогнозируемые метки сильно отличаются от фактических, и низкими, если значения близки. Для задач регрессии наиболее распространенной функцией потерь является **квадратичная функция потерь (MSE, mean square error)**. Потеря MSE для одного примера x_i при параметров W и потеря по набору входных данных $i = \{1, \cdots, M\}$ находятся по следующим формулам:

$$L_i(W) = (\widehat{y_i} - y_i)^2, \tag{1.8}$$

$$L(W) = \frac{1}{M} \sum_i L_i(W) = \frac{1}{M} \sum_i (\widehat{y_i} - y_i)^2, \tag{1.9}$$

где y — ожидаемый, $\widehat{y}$ — получившийся результат.

Для задачи классификации широко используется **функция потерь кросс-энтропии (Cross-entropy loss, CE)**. Потеря CE для одного примера x_i при параметров W по N классам и потеря по набору входных данных $i = \{1, \cdots, M\}$ находятся по следующим формулам:

$$L_i(W) = -\sum_{j=1}^{N} y_j^{(i)} \log(P_j^{(i)}), \tag{1.10}$$

$$L_i(W) = \sum_{i=1}^{M} L_i(W) = -\sum_{i=1}^{M}\sum_{j=1}^{N} y_j^{(i)} \log(P_j^{(i)}), \tag{1.11}$$

где $P_j^{(i)}$ — вероятность того, что x_i принадлежит классу j; $y_j^{(i)}$ равно 1 если x_i принадлежит классу j, иначе 0.

Для моделей классификаторов изображений широко используется функция потерь **SVM (Multi-class Support Vector Machine loss, SVM loss)**. Функция потерь SVM настроен так, чтобы при параметрах W в выходном слое правильный класс для каждого изображения имел оценку выше, чем неправильные классы, на некоторую фиксированную константу Δ. Для i-го примера даны пиксели изображения x_i и метка y_i, задающая индекс правильного класса. Таким образом, потеря SVM для i-го примера формализуется следующим образом:

$$L_i = \sum_{j \neq y_i} \max(0, f_j - f_{y_i} + \Delta), \tag{1.12}$$

где $f = f(x_i, W)$ вектор оценок класса, $f_{y_i} = f(x_i, W)_{y_i} = w_{y_i}^T x_i$ — оценка правильного класса ($w_{y_i}^T$ — y_i-ая строка матрицы параметров W, трансформированная в столбец), f_j — оценка j-го неправильного класса. Чтобы избежать переобучения (overfitting), функцию потерь можно расширить регуляризационным штрафом $R(W)$. Наиболее распространенным штрафом является квадрат нормы $L2$, который суммирует все квадраты элементов W:

$$R(W) = \sum_k \sum_l W_{k,l}^2. \tag{1.13}$$

Таким образом, потеря SVM по набору входных данных $i = \{1, \cdots, M\}$ формализуется следующим образом:

$$L = \frac{1}{M} \sum_i L_i(W) + \lambda R(W), \tag{1.14}$$

где λ — сила регуляризации (regularization strength).

1.4.3 Градиентный спуск

Для оптимизации модели, т. е. для нахождения значений параметров модели, которые максимально снижают значение функция потерь, используется метод **градиентного спуска (gradient descent, GD)**. Идея метода заключается в итеративном изменении значений параметров w и b пока L не достигнет своего минимума (или значения, достаточно близкого к минимуму). Поскольку градиент — это вектор, указывающий в направлении наибольшего увеличения функции, значение функции можно минимизировать, итеративно двигаясь по шагу в направлении отрицательного градиента, то есть в направление скорейшего убывания функции:

$$w := w - \alpha \frac{dL(w,b)}{dw}, \tag{1.15}$$

$$b := b - \alpha \frac{dL(w,b)}{db}. \tag{1.16}$$

В формулах 1.15 и 1.16 α — коэффициент скорости обучения или размер шага спуска. На рисунке 1.10 изображен процесс минимизации функции потерь градиентным спуском.

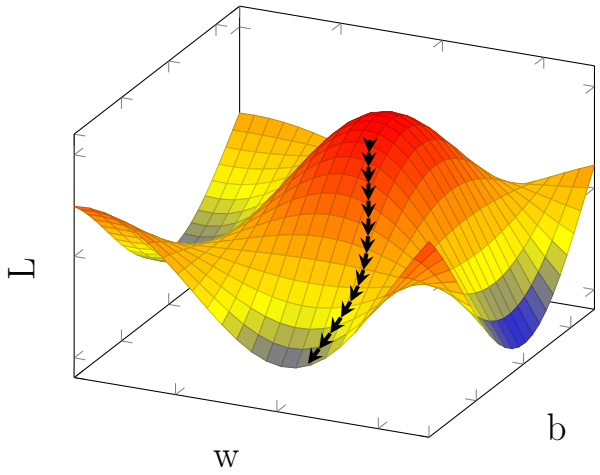

Рисунок 1.10 – Градиентный спуск.

Для вычисления градиента наиболее часто используется **метод обратного распространения (backpropagation)**, который заключается в рекурсивном применении цепного правила дифференцирования. На схеме 1.11

визуализировано применение этого метода на модель с входными данными $x1, x2$ и параметрами $w1, w2, b$. Сумма скалярных произведений и смещения передается в сигмоидную функцию активации $f(w, x) = \frac{1}{1+e^{-(w1x1+w2x2+b)}}$. Прямой проход вычисляет значения от входных данных к выходным (показаны зеленым цветом), обратный проход выполняет обратное распространение, которое начинается с конца и рекурсивно применяет цепное правило для вычисления градиентов (показаны красным) до входных данных. Например, на предпоследнем узле схемы локальный градиент от $1/x$ равный $\frac{-1}{x^2}$ по $x = 1.37$ умножаем на текущий градиент 1.00 и получаем значение следующего градиента -0.53. После вычисления градиентов, в методе градиентного спуска параметры будут обновлятся следующим образом: $w1 = 2 - (-0.2)\alpha$, $w2 = -3 - (-0.4)\alpha$, $b = -3 - (0.2)\alpha$.

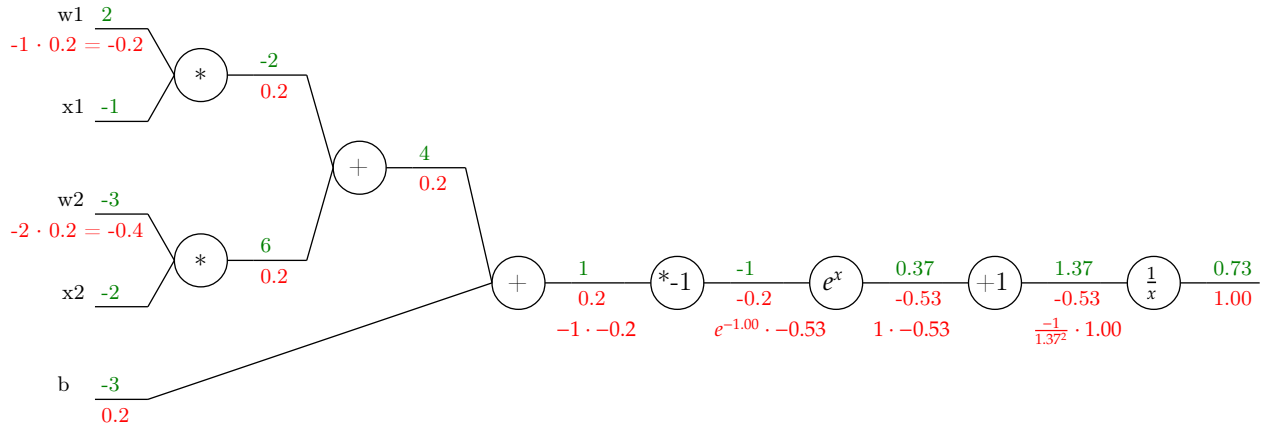

Рисунок 1.11 – Схема метода обратного расстояния.

На практике часто используются модификации метода градиентного спуска, как SGD (стохастический градиентный спуск), Adam, AdamW, позволяющие быстрее оптимизировать функцию потерь.

1.4.4 Проектирование нейронных сетей

Ключевым моментом при проектировании нейронных сетей является определение архитектуры, то есть определение глубины сети (количество слоев), ширины каждого слоя и вида связи между нейронами. Также инициализация начальных значений параметров модели, выбор методов регуляризации и предобработки обучаемых данных играют важную роль в производительно-

сти модели. Идеальная сетевая архитектура и настройка параметров должны быть найдены при валидации (validation), т.е. с помощью экспериментов, основанных на отслеживании ошибки набора проверки (validation set error) [16].

1.4.5 Глубокое обучение

Нейронные сети с прямой связью (FFN, feedforward neural networks), также называемые многослойными персептронами (MLP, Multi Layer Perceptron), являются основой многих важных архитектур алгоритмов глубокого обучения, как сверточные сети и сети с прямой связью [16]. Именно количество слоев или глубина сети отличает алгоритм глубокого обучения от простой нейронной сети (сети с более чем 3 слоями относятся алгоритмам глубокого обучения). В сетях с прямой связью выходные данные одного слоя используются в качестве входных данных только для следующего слоя. Это означает, что в сети нет петель – информация всегда передается вперед и никогда не возвращается обратно. Нейросети, в которых предыдущие выходные данные используются в качестве входных, называются *сетями с обратной связью (recurrent neural networks, RNN)* [19].

2 Классификация существующих методов

2.1 Методы на основе сверточной нейронной сети

Сверточная нейронная сеть является основным методом анализа изображений для различных типов задач компьютерного зрения. Большинство методов анализа медицинских изображений основано на сверточные сети.

2.1.1 Свёрточная нейронная сеть (Convolutional Neural Network, CNN)

Сверточные сети, также известные как *сверточные нейронные сети (Convolutional Neural Network, ConvNet)*, представляют собой специализированный тип нейронной сети для обработки данных, которые имеют сеточную топологию, такую как изображение [16]. В отличие от обычной нейронной сети, слои ConvNet имеют пространственную структуру, с измерениями: ширина, высота, глубина.

Входной слой ConvNet содержит изображение, где ширина и высота будут размерами изображения, а глубина будет равна числу цветовых каналов. В большинстве случаев глубина равна 3, так как большинство входных изображений представлены в модели RGB — имеют 3 цветовых канала. Остальные слои сети также будут иметь пространственную структуру.

ConvNet состоит из следующих типов слоев: сверточный слой (convolution layer, CONV), слой активации (activation function layer, AF), слой подвыборки (pooling layer, POOL) и полносвязный слой (fully connected layer, FC). На рисунке показана архитектура ConvNet.

Сверточный слой (CONV)

Параметры CONV состоят из набора обучаемых фильтров. Каждый фильтр представляет собой набор ядер, причем для каждого отдельного канала (среза) ввода существует одно ядро. Ядро представляет собой матрицу весов. Это ядро перемещается или «скользит» по соответствующим входным каналам, выполняя *свертку* или поэлементное умножение на тот часть входного канала, на которой оно находится в данный момент. Затем результаты умножения

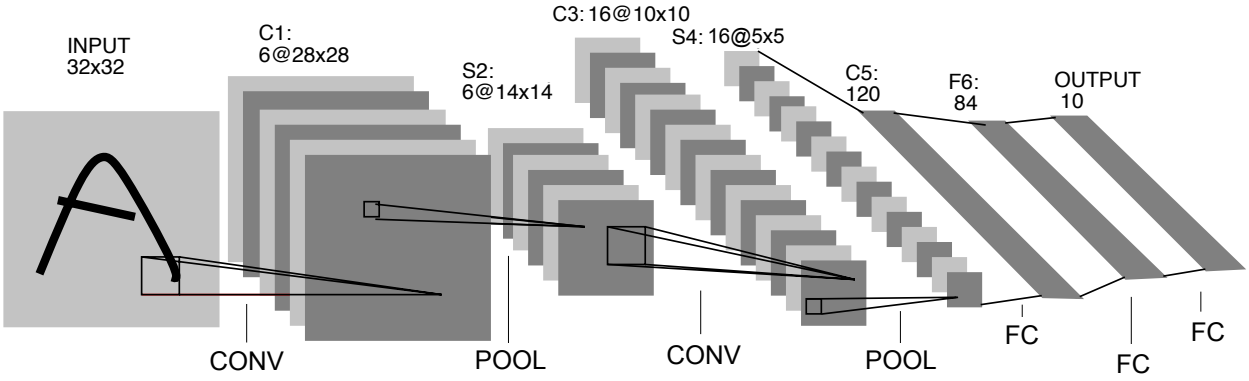

Рисунок 2.1 – Архитектура сверточной сети [20].

суммируются и сформируется результирующая матрица. На рисунке 2.2 показана пример операции свертки, где на ввод с размером 5 × 5 применяется 3 × 3 ядро. Свертке левой верхней части входного канала соответствует первый элемент результирующей матрицы со значением 12.

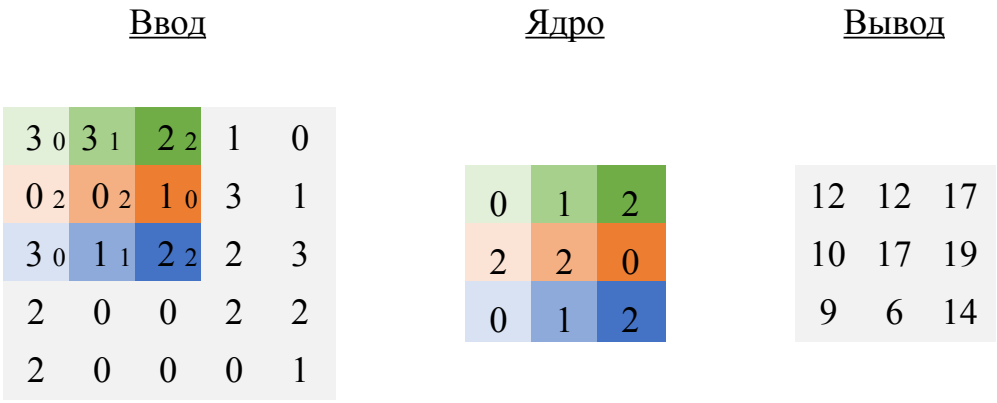

Рисунок 2.2 – Вычисление значений карты активации.

Затем каждая из версий (вычисленных матриц), обработанных для каждого канала, суммируется для формирования одного канала, к которому добавляется смещение, и создается окончательный выходной канал – карта активации. При этом, в результате CONV количество вычисленных карт активации равно количеству фильтров [21]. Процесс сформирования одной карты активации показан на рисунке 2.3, где входными данными является изображение с 3 каналом RGB.

Размером вывода CONV управляют следующие гиперпараметры.

1. *Количество использованных фильтров*, которое соответствует глубине вывода CONV.

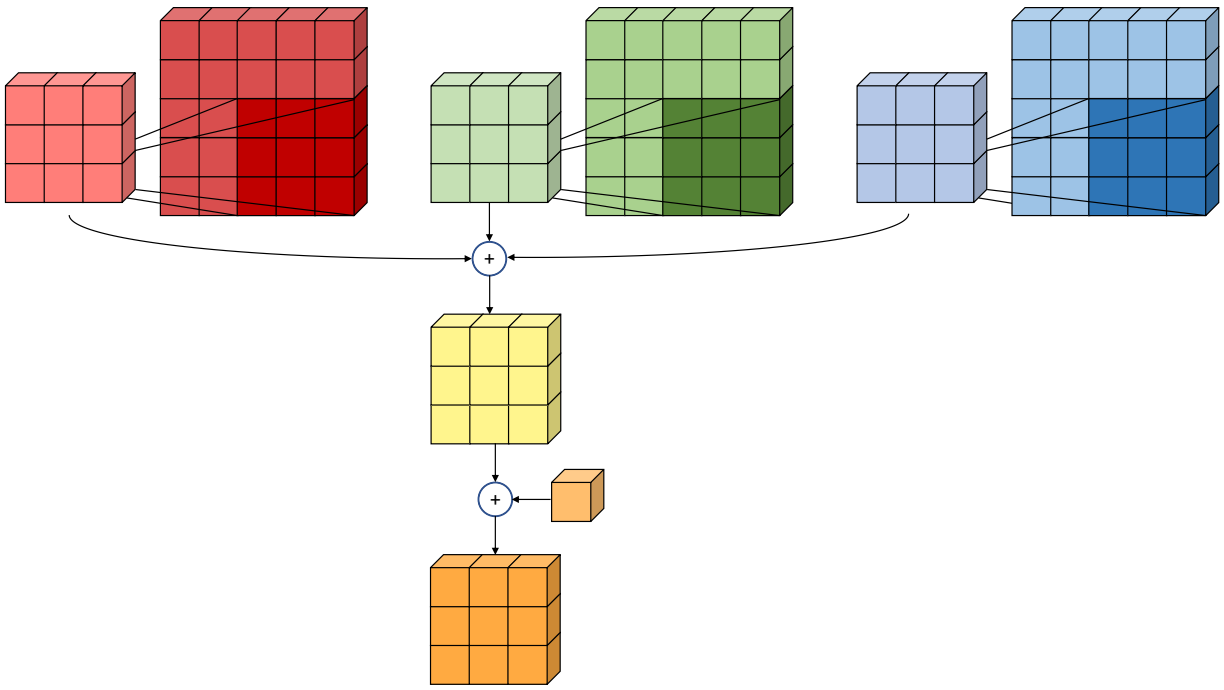

Рисунок 2.3 – Сформирование карты активации CONV.

2. *Stride (шаг)* – количество ячеек (пикселей), на которое перемещается ядро после каждой операции. Увеличение шага позволит производить выводы меньшего размера.

3. *Padding (заполнение)* – процесс добавления дополнительных ячеек (пикселей) к краям входных данных. Обычно добавляют ячейки нулевого значения, вследствие этого к ним применяется термин “нулевое дополнение” — “zero padding”. Чаще всего, это используется для точного сохранения пространственного размера входного объема.

Размер карты активации вычисляется по следущей формуле:

$$AM = \frac{(N + 2P - F)}{S} + 1, \tag{2.1}$$

где N – размер входных данных, P – количество добавленных ячеек (padding), F – размер ядра, S – шаг перемещения ядря (stride). Например, для ввода с длиной и шириной $N = 7$ и фильтра $F = 3$ с шагом $S = 1$ и $P = 0$ получим вывод $AM = \frac{(7+2\cdot1-3)}{1} + 1 = 5$. С шагом $S = 2$ мы получим вывод $AM = 3$. Чтобы сохранить размер ввода, то есть чтобы $AM = 7$, необходимо использовать нулевое дополнение $P = 1$.

В сети, показанной на рисунке 2.1, на CONV слое C1 применены 6 филь-

тров с размером $F = 5$ и с шагом $S = 1$ без заполнении, в итоге созданы 6 карт активации с размером $AM = \frac{(32+2\cdot 0-5)}{1} + 1 = 28$.

В процессе обучения сеть будет изучать фильтры, которые активируются, когда они видят какой-то тип визуальной особенности, такой как край или пятно определенного цвета на первом слое, или, на более высоких слоях целые узоры. Обратный проход для операции свертки, как для данных, так и для весов, также является сверткой, но с пространственно-перевернутыми фильтрами.

Слой активации (AF)

В AF к выходным данным CONV применяется поэлементная функция активации. Это оставляет размер данных без изменений. Чаще всего используется функция активации ReLU или ReLU с «утечкой».

Слой подвыборки (POOL)

На практике принято периодически вставлять слой *подвыборки (subsampling)* или *объединения (pooling)* между последовательными слоями свертки. Его функция состоит в том, чтобы постепенно уменьшать пространственный размер представления, что будет уменьшить количество параметров и вычислений в сети и, следовательно, контролировать переобучение. POOL работает независимо с каждым срезом входных данных и изменяет его размер в пространстве, используя операцию max или avg. Размер глубины ввода остается неизменным.

Чаще всего используется POOL с фильтрами размером 2×2, применяемыми с шагом 2, отбрасывая 75% ввода. Каждая операция max или avg в этом случае будет принимать максимум или среднюю из 4 чисел. В сети, показанной на рисунке 2.1, на POOL слое S2 применены 6 фильтры с таким же размером и шагом, в итоге созданы 6 карт активации с размером $AM = \frac{(28-2)}{2} + 1 = 14$.

Полносвязный слой (FC)

Выходные данные последнего слоя CONV или POOL выравниваются, то есть трехмерная матрица преобразуется в вектор, а затем передаётся в FC.

FC используется для нахождении оценок классов и работает как обычный слой простой нейронной сети. На практике обычно используется 1-3 FC в конце сверточной сети.

Проектирование свёрточных нейронных сетей

Сверточные сети содержат порядка 100 миллионов параметров и обычно состоят примерно из 10-20 слоев (отсюда и глубокое обучение). Это резко отличается от обычных нейронных сетей, где большее количество слоев (более чем 4) редко помогает повышать точность. Одним из аргументов в пользу этого наблюдения является то, что изображения содержат иерархическую структуру (например, лица состоят из глаз, которые состоят из краев и т.д.), поэтому нужно несколько уровней обработки.

2.1.2 FCN

Типичные сверточные сети, в том числе LeNet [20], AlexNet [17] и их более глубокие преемники, принимают входные данные фиксированного размера и производят непространственные выходные данные. Полносвязные слои этих сетей имеют фиксированные размеры и отбрасывают пространственные координаты. Однако эти полносвязные слои также можно рассматривать как CONV слои с ядрами, покрывающими всю их входную область. Это превращает сеть в *полностью сверточную сеть (Fully Convolutional Network, FCN)*, которая принимает входные данные любого размера и выводит карты классификации соответствующего размера. Вместо одной классификации для всего изображения как ConvNet, FCN классифицирует каждый пиксель изображения – выполняет сегментацию, увеличивая размерность вывода сверточных слоев. FCN [22] – первая модель глубокого обучения для сегментации изображений, на основе которой создано множество других моделей с высокой точностью, как U-Net, V-Net, Deconvolution Network.

Преобразование слоев FC в слои CONV

Чтобы преобразовать ConvNet в FCN, последний FC слой-классификатор отбрасывается, и все остальные FC слои преобразуются в CONV, в которых размер фильтра точно соответствует размеру входного объёма.

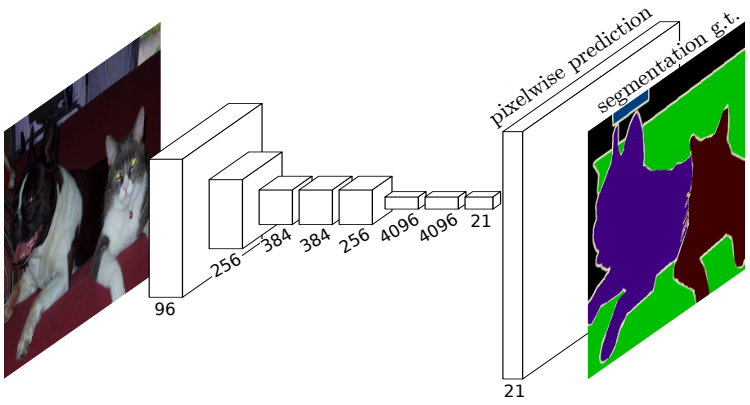

Рисунок 2.4 – Архитектура FCN [22].

Рассмотрим архитектуру AlexNet, которая принимает изображение размером 224 × 224 × 3, а затем использует ряд слоев CONV и POOL, чтобы уменьшить образ до размера 7 × 7 × 512. Далее используется два слоя FC размером 4096 и последний слои FC с 1000 нейронами, который вычисляет оценки класса. Каждый из этих трех слоев FC можно преобразовать в слои CONV, следующим образом:

– первый FC слой с 4096 нейронами, принимающий объем 7 × 7 × 512 — CONV слой с K = 4096 фильтрами размера F = 7, что дает выходной объем 1 × 1 × 4096;

– второй FC слой с 4096 нейронами — CONV слой с K = 4096 фильтрами размера F = 1, что дает выходной объем 1 × 1 × 4096;

– последний FC слой с 1000 нейронами — CONV слой с K = 1000 фильтрами размера F = 1, что дает окончательный результат 1 × 1 × 1000.

Upsampling

Как описано выше, FCN выполняет классификацию каждого пикселя изображения, увеличивая размерность вывода последнего слоя, который был преобразован из FC слоя в CONV. Такой процесс увеличения размерности называется *upsampling*, результат процесса называют *upsampled* выводом. Существуют различные методы upsampling, в реализации FCN [23] используется *биленейная интерполяция*, в которой в качестве интерполированного значения используется взвешенное среднее четырёх соседних пикселей.

Skip соединения

Чтобы увеличить уровень детализации upsampled вывода, в сети добавляются skip соединения между upsampled выводом и выводом предыдущих CONV слоев соответствующего размера. На рисунке 2.5 показан процесс добавления skip соединений для сформирования выводов FCN-16s и FCN-8s, где выводы слоев POOL и upsampled выводы показаны в виде квадратных сеток, а промежуточные слои показаны в виде вертикальных линий.

– FCN-32s сформируется в результате 32× upsampling – увеличения размерности вывода последнего слоя `conv7` в 32 раз.

– FCN-16s: чтобы отобразить в результате более мелкие детали, сохраняя при этом семантическую информацию высокого уровня, добавляются skip соединения между выводами слоев `conv7` и `pool4`. При этом размерность вывода `conv7` увеличивается в 2 раза, и поверх `pool4` добавляется слой CONV с фильтрами размера F = 1, чтобы получить дополнительные предсказания класса. Далее эти выводы суммируются и увеличиваются 16 раз для сформирования вывода, который совпадает по размеру с входным изображением.

– FCN-8s: для дальнейших улучшений, вывод, полученный при skip соединении слоев `conv7` и `pool4`, увеличивается 2 раза и поверх `pool3` добавляется слой CONV с фильтрами размера F = 1 для классификации. Далее эти выводы суммируются и увеличиваются в 8 раз и сформирует вывод FCN-8s.

Вышеописанные три выводы рассматриваются как разные версии FCN, и в экспериментах [22], проведенных на различные наборы данных, FCN-8s последовательно достигает высочайшей достоверности. Также FCN-8s дает относительное улучшение на 30% по сравнению с предыдущими лучшими результатами теста PASCAL VOC 11/12, достигнутый сетью R-CNN, с более быстрым выводом и обучением.

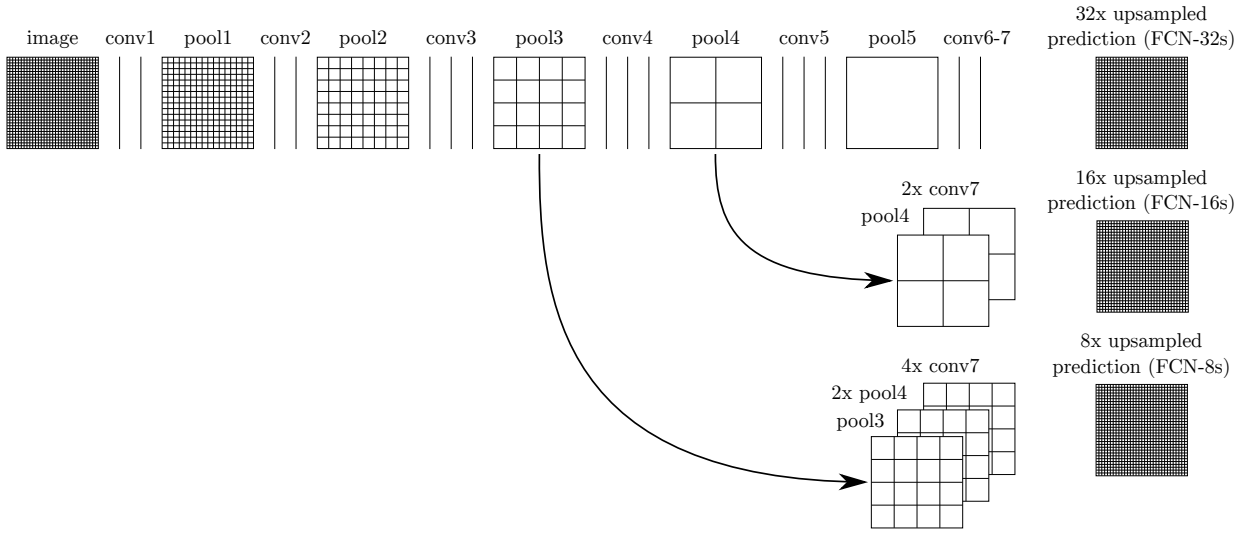

Рисунок 2.5 – Сформирование вывода сети FCN skip соединениями [22].

2.1.3 U-Net

U-Net является одной из самых известных моделей для задачи сегментации изображении. U-Net, предложенная в [5] для сегментации изображений биологической микроскопии, модифицирует и расширяет архитектуру FCN, для работы с низким количеством обучающих изображений, обеспечивая более точную сегментацию. Модель состоит из «сужающей» пути, или кодировщика, и «расширяющей» пути, который является декодером.

Кодировщик соответствует типичной архитектуре ConvNet, состоит из повторного применения двух CONV слоев с фильтрами размера 3×3 без нулевого дополнения, за каждой из которых следует ReLU и операция максимального объединения – max POOL с фильтром 2×2 с шагом $S = 2$ для понижения размерности ввода. После каждого слоя max POOL количество каналов, или количество фильтров следующего CONV слоя, удваивается.

Каждый шаг расширяющей пути, или декодера, состоит из операции upsampling, удваивающий размернось ввода. За upsampling следует CONV с фильтрами размера 2×2, который вдвое уменьшает количество каналов признаков. Затем этот вывод с удвоенной размерностью, глубина которого уменьшилась в 2 раза, объединяется с соответствующим обрезанным выводом кодировщика и передается в 2 CONV слои с 3×3 фильтрами, за каждой из которых следует ReLU. Обрезка вывода кодировщика необходима из-за потери краевых пикселей при каждом слое CONV. Добавляя нулевое дополнение в CONV можно сохранить размерность ввода, следовательно, выводы коди-

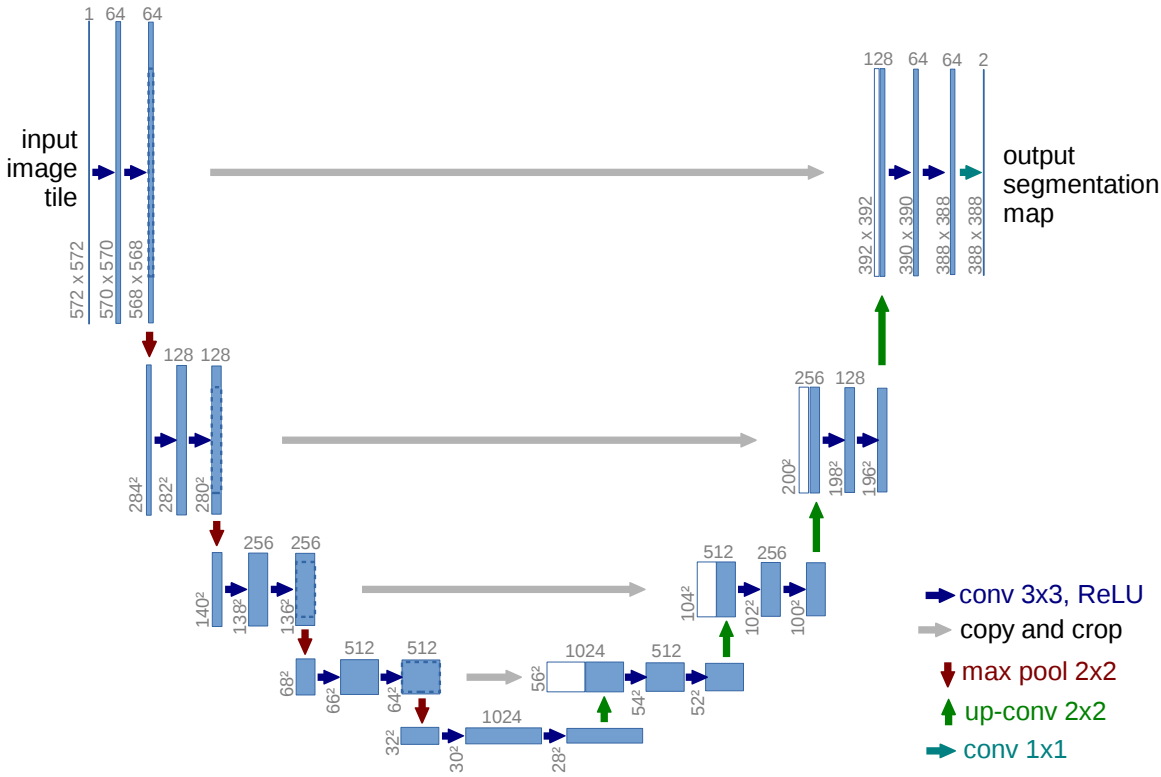

Рисунок 2.6 – Архитектура U-Net [5].

ровщика можно будет объединить без обрезки. За последним 3 × 3 CONV слоем добавляется 1 × 1 CONV для классификации каждого пикселя по желаемым количеством классов.

Важная модификация, внесенная в FCN для создания U-Net, заключается в том, что в декодере через skip соединения upsampled выводы объединяются с выводами более высоких CONV слоев кодировщика, также добавлено большое количество каналов (фильтров в CONV слоях), которые позволяют сети распространять контекстную информацию на уровни с более высоким разрешением. Как следствие, кодировщик симметричен декодеру и дает U-образную архитектуру.

Транспонированная свертка

Другое отличие U-Net и FCN состоит в том, что для upsampling вместо билинейной интерполяции U-Net использует *транспонированную свертку (transposed convolution)*, также известный как *свертки с дробным шагом (fractionally strided convolution)*. Транспонированную свертку можно рассматривать как операцию обращения обычной свертки, результат которого имеет тот же размер, что и ввод обычной свертки. Например, в результате свертки

ядра 3 × 3 на входе 6 × 6 с шагом S = 1 и отсутствием нулевого заполнения P = 0 получается вывод размера 4 × 4. Транспонирование этой свертки будет иметь вывод с размером 6 × 6 при вводе 4 × 4 (см. рисунок 2.7).

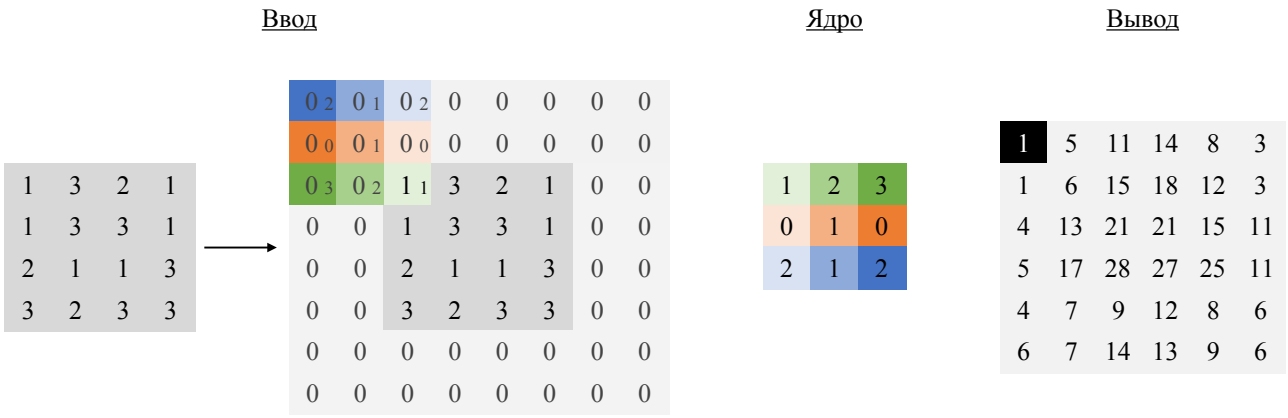

Рисунок 2.7 – Транспонированная свертка.

Существуют разные методы реализации транспонированной свертки, один из них это ее вычисление через эквивалентную обычную свертку: для ввода, полученный путем свертки с ядром размера F, шагом S и нулевым дополнением P, эквивалентной обычной сверткой будет свертка с ядром размера F, шагом S' = 1 и нулевым дополнением P' = F − P − 1. При этом ядро транспонируется относительно главной и побочной диагонали. Для вышеприведенного примера (см. рисунок 2.7) транспонированная свертка будет эквивалентна обычной свертке, в которой на ввод применяется нулевое дополнение P' = 3 − 0 − 1 = 2.

В случае транспонирования свертки, которая применена с шагом S > 1, между элементами входной матрицы добавляются Z = S − 1 ячеек со значением 0. На примере, представленном на рисунке 2.8, между элементами 2 × 2 ввода, полученный в результате свертки данных размера 5 × 5 с шагом S = 2 и P = 0, добавляются Z = 2 − 1 = 1 ячейки со значением 0.

Аугментация данных

Как и другие модели анализа медицинских изображений, U-Net использует аугментацию данных для эффективного обучения на малом количестве аннотированных изображений. В случае медицинских изображений в первую очередь требуется инвариантность к сдвигу и вращению, также устойчивость к деформациям и вариации значения серого оттенка (gray value variations). В

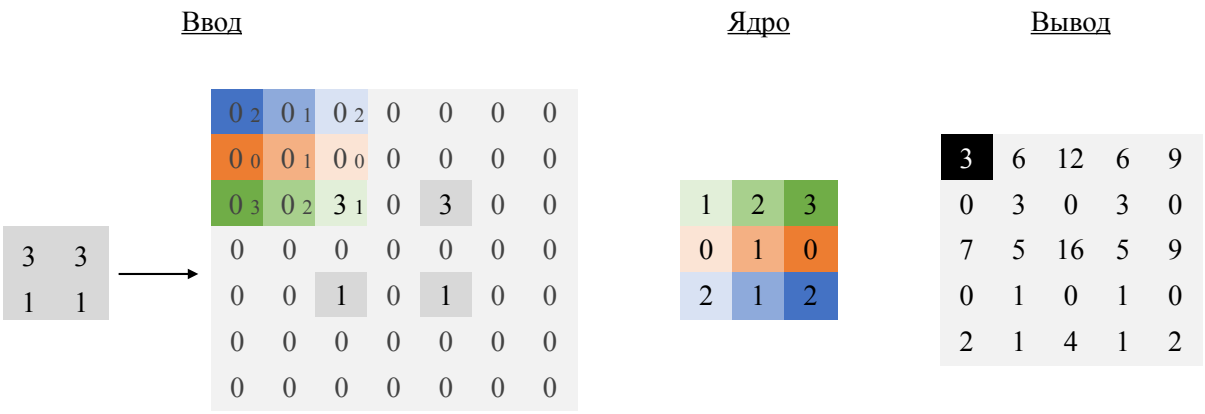

Рисунок 2.8 – Транспонированная свертка при S = 2.

частности, случайные эластичные деформации (elastic deformations) обучающих выборок являются ключевой концепцией для обучения сети сегментации с малым количеством аннотированных изображений.

В U-Net для аугментации данных генерируются гладкие эластичные деформации, используя случайные векторы смещения на сетке 3×3. Смещения взяты из распределения Гаусса со стандартным отклонением 10 пикселей, затем смещения на пиксель вычисляются с использованием бикубической интерполяции.

nnU-Net

nnU-Net [24] – метод сегментации медицинских изображений, созданный на основе U-Net, – настраивается автоматически для произвольных новых наборов данных. В частности, все основные топологические параметры модели, включая предварительную обработку, структуру сети, обучение и последующую обработку, настраиваются для любой новой задачи в области биомедицины, без необходимости ручных вмешательств.

Чтобы обеспечить быструю адаптацию к заданному набору данных, в методе сформулированы эвристические правила, основанные на зависимости между свойствами набора данных и сетевой архитектурой. После выполнения эвристических правил nnU-Net создает следующие три конфигурации U-Net:

– 2D U-Net (с двумерной сверткой);

– 3D U-Net (с трехмерной сверткой), работающая с полным разрешением изображения;

– последовательность 3D U-Net, в котором первая U-Net работает с изображениями с пониженным разрешением, а вторая обучается для улучшения разрешения карт сегментации, созданных первой.

После оценки созданных конфигураци кросс-валидацией (cross-validation), nnU-Net эмпирическим путем выбирает наиболее эффективную конфигурацию. Результатом процесса автоматической настройки и обучения nnU-Net является полностью обученная модель, которую можно использовать для сегментации заданного набора изображений.

Все архитектуры U-Net, сконфигурированные nnU-Net, основаны на одном и том же шаблоне, который почти полностью совпадает с оригинальным U-Net и его трехмерным аналогом [25]. В [24] предлагают гипотезу о том, что правильно сконфигурированную обычную U-Net по-прежнему трудно превзойти. В частности ни одна из конфигураций U-Net, созданный nnU-Net, не использовала недавно предложенные архитектурные модификации, такие как остаточные (residual) соединения, плотные (dense) соединения, механизмы внимания или расширенные (dilated) свертки.

2.1.4 ResNet

ResNet (Residual Network) [26] – это тип ConvNet, позволяющий обучать более глубокие сети. В частности, с увеличением глубины сети возникает проблема исчезновения градиентов (vanishing gradients), что затрудняет их обучение. Также возникает проблема деградации: точность сначала достигает пика и затем быстро ухудшается. Для устранения этих проблем, в ResNet вводится *остаточное (residual)* соединение, позволяющее сети иметь гораздо глубокие уровни, чем традиционные нейронные сети, но при этом иметь возможность эффективно обучаться.

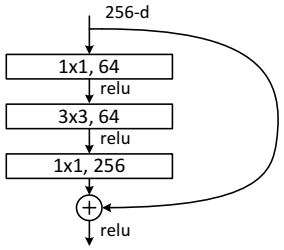

Рисунок 2.9 – Остаточное соединение ResNet [26].

На рисунке 2.9 приведен пример блока слоев с остаточным соединением, где ввод с глубиной 256 передается на слои свертки. Вывод последней свертки с глубиной 256 и исходный ввод суммируются и передается следующему слою ReLU.

Остаточное соединение примененная к некоторому блоку слоев формализуется следующим образом:

$$y = \mathcal{F}(x, \{W_i\}) + x, \tag{2.2}$$

где x и y – входной и выходной векторы рассматриваемых слоев, $\mathcal{F}(x, \{W_i\})$ – функция, которую необходимо изучить. Для примера, представленной на рисунке 2.9, $\mathcal{F} = \mathrm{CONV}_{1\times1,256}(\mathrm{CONV}_{3\times3,64}(\mathrm{CONV}_{1\times1,64}(x)))$.

В уравнении 2.2 размерности x и $\mathcal{F}$ должны быть равны. Если это не так, выполняется линейная проекция:

$$y = \mathcal{F}(x, \{W_i\}) + W_s x. \tag{2.3}$$

С момента своего появления ResNet получила широкое распространение и использовалась в качестве базовой архитектуры в различных приложениях. На таблице 2.1 представлена архитектура модели ResNet50 и ее модификации BoTNet50.

2.1.5 Inception

Inception V1 (GoogLeNet), V2, V3 – это модели ConvNet, использующие нескольких параллельных CONV слоев с фильтрами разного размера, предназначенные для изучения как локальных, так и глобальных признаков изображения.

Чтобы сделать вычисление свертки более эффективным, перед каждым CONV добавляется дополнительный 1×1 CONV, который уменьшает глубину входных каналов. Далее, CONV слои с большими фильтрами заменяются несколькими CONV слоями с фильтрами меньшего размера, например, 5×5 CONV заменяется двумя 3×3 CONV. При этом CONV размера фильтра $n \times n$ заменяются комбинацией параллельных сверток $1 \times n$ и $n \times 1$. Например, 3×3 CONV эквивалентна двумя параллельным 1×3 CONV и 3×1

CONV.

Также как ResNet, Inception широко используется в качестве базовой архитектуры для многих других задач компьютерного зрения.

2.2 Методы на основе Трансформера

Трансформер – тип архитектуры нейронной сети, представленный в [27] для задачи машинного перевода. Трансформер достигнул самых высоких результатов и стал стандартной моделью в задачах обработки естественного языка (NLP), заменив модели RNN, такие как LSTM (Long short term memory).

Трансформеры также стали широко применяться для задач компьютерного зрения. Модели ViT (Vision Трансформеrs, Трансформеры зрения), представленные в [28], достигли самых современных результатов во многочисленных задачах анализа изображений, включая сегментацию, классификацию изображений, обнаружение объектов, реконструкция, генерация отчета из изображений.

Основное отличие Трансформера от других моделей как ConvNet или RNN заключается в том, что модель использует *механизм внимания (attention mechanism)* для обработки последовательностей данных, что позволяет эффективно найти корреляцию между элементами данных и оценивать их важность.

2.2.1 ViT

Для применения Трансформера, который работает с последовательными данными как текст или аудио, в ViT [28] изображение $x \in \mathbb{R}^{H \times W \times C}$ преобразуется в последовательность сплющенных (сглаженных) фрагментов (patch) $x_p \in \mathbb{R}^{N \times (P^2 \cdot C)}$, где $H \times W$ – разрешение исходного изображения, C – число каналов, $P \times P$ – разрешение каждого фрагмента изображения, $N = HW/P^2$ – результирующее число фрагментов.

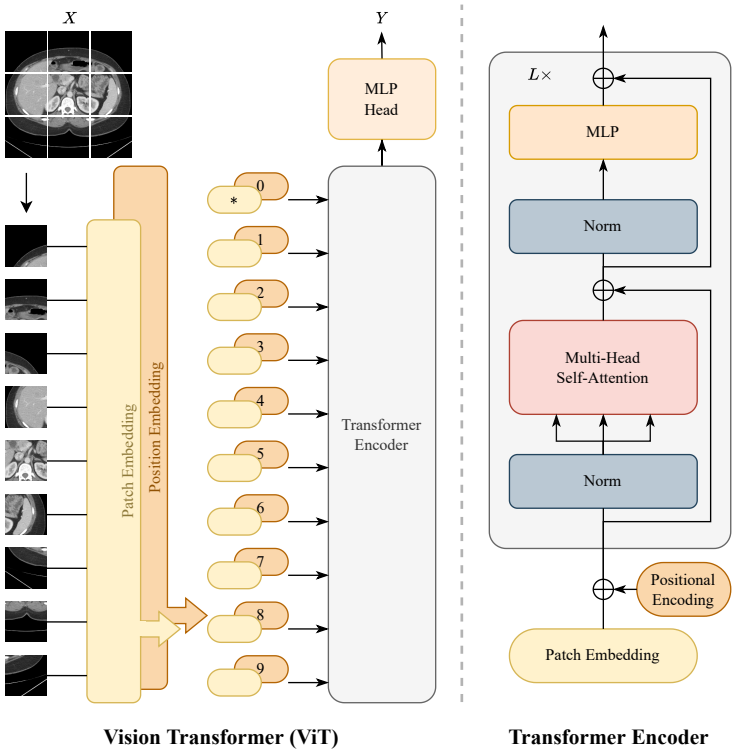

Рисунок 2.10 – ViT [29].

Линейная проекция сглаженных фрагментов

В Трансформере все подслои производят выходные данные размерности D, поэтому каждый фрагмент $x_p^i \in \mathbb{R}^{1\times(P^2\cdot C)}$ линейно проецируется в вектор $x_p^i \in \mathbb{R}^{1\times D}$ с использованием обучаемой матрицы embedding-а $E \in \mathbb{R}^{(P^2\cdot C)\times D}$ по формуле 2.5. Выходные данные этой проекции называются patch embedding-ами.

$$z_0 = [x_{class}; x_p^1 E; x_p^2 E; \ldots; x_p^N E], \quad E \in \mathbb{R}^{(P^2\cdot C)\times D}. \tag{2.4}$$

Position embeddings

Для сохранения информации о позициях фрагментов в изображении к patch embedding-ам добавляются *position embedding*-и $E_{pos} \in \mathbb{R}^{(N+1)\times D}$. В ViT используется стандартные обучаемые embedding-и 1D-позиций, поскольку использование более продвинутых 2D embedding-ов не дало значительного прироста производительности. Полученная последовательность векторов передается кодировщику.

$$z_0 = [x_{class}; x_p^1 E; x_p^2 E; \ldots; x_p^N E] + E_{pos}, \quad E_{pos} \in \mathbb{R}^{(N+1)\times D}. \tag{2.5}$$

Кодировщик-трансформер

Кодировщик в ViT состоит из L чередующихся блоков множественного внутреннего внимания (multi-head self-attention, MSA) и простой сети с прямой связью (fully connected feed-forward network, FFN или MLP). Перед каждым блоком к данным применяется нормализация (Layer-normalization, LN), и остаточные соединения (residual connections) – после каждого блока. MLP содержит два слоя с функцией активации GeLU.

Структура кодировщика в ViT отличается от структуры кодировщика обычных Трансформеров NLP только тем, что в ViT используется pre-norm, то есть нормализация выполняется до слоев MSA и MLP.

$$z'_l = MSA(LN(z_{l-1})) + z_{l-1}, \qquad l = 1 \dots L \tag{2.6}$$

$$z_l = MLP(LN(z'_l)) + z'_l, \qquad l = 1 \dots L \tag{2.7}$$

$$y = LN(z_L^0). \tag{2.8}$$

Self-attention (SA)

Attention mechanism (механизм внимания) в нейронных сетях имитирует физиологический процесс обработки сигналов [30]. Типичная функция внимания определяет важность различных элементов входных данных и «концентрируется» на наиболее важных частях. Self-attention (само-внимание, SA) или intra-attention (внутреннее внимание) представляет собой вариант механизма внимания, который позволяет обрабатывать все элементы ввода одновременно и моделировать отношения и зависимости между ними. SA в Трансформерах называют вниманием масштабированного скалярного произведения (scaled dot-product attention).

До вычисления SA из каждого входного вектора выводят следующие векторы:

– q – вектор запроса (query), представляющий текущий элемент-фрагмент, для которой вычисляется SA;

– k – вектор ключа (key), который используется для представления всех элементов входной последовательности, то есть информации, с которой модель сравнивает вектор q;

– v – вектор значения (value), который используется для извлечения «содержания» элементов ввода.

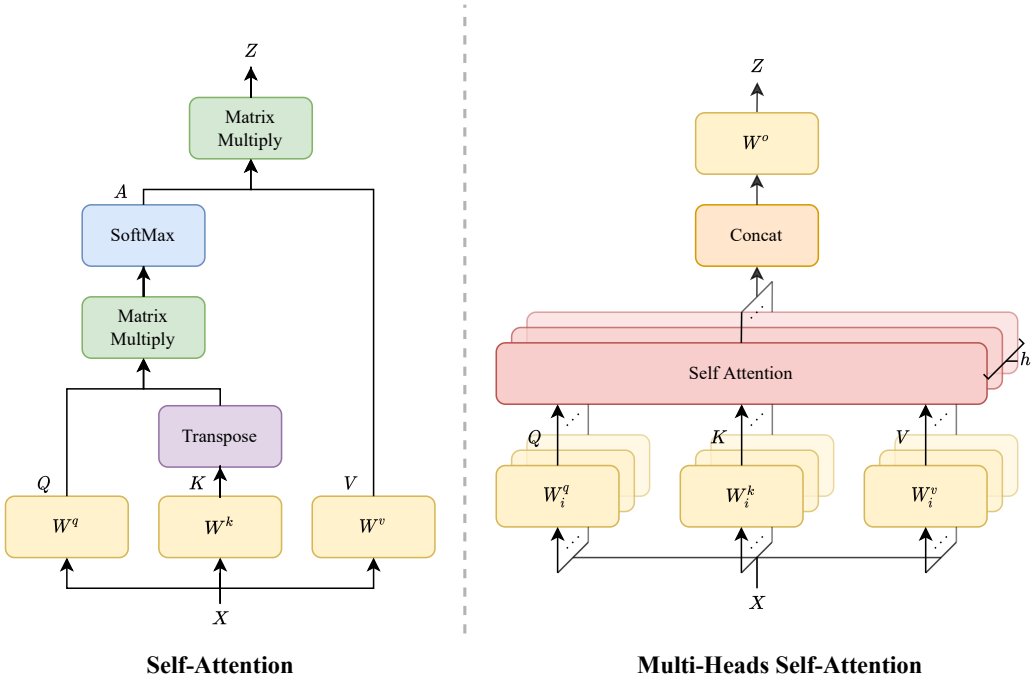

Рисунок 2.11 – SA и MSA [29].

SA для j-ого входного вектора вычисляется по формуле 2.9. Скалярное произведение векторов q_j и k_i позволяет модели определить, какие векторы k_i наиболее релевантны вектору q_j. После его нормализации скалярное произведение умножается на вектор v_i, что позволяет модели использовать информацию или содержание более релевантных к q_j элементов ввода для прогнозирования или других следующих вычислений.

$$SA(q_j, k, v) = \sum_i \text{softmax}\left(\frac{k_i^\top q_j}{\sqrt{D}}\right) v_i. \tag{2.9}$$

На практике SA вычисляют одновременно для всех векторов запроса q всех N входных векторов, объединенных в матрицу Q. Следовательно, для ввода $z \in \mathbb{R}^{N\times D}$ выводятся матрицы $Q \in \mathbb{R}^{N\times D}$, $K \in \mathbb{R}^{N\times D}$ и $V \in \mathbb{R}^{N\times D}$, используя три обучаемых параметра W^q, W^k и W^v, соответственно:

$$Q = z \times W^q, \quad W^q \in \mathbb{R}^{D\times D}, \tag{2.10}$$
$$K = z \times W^k, \quad W^k \in \mathbb{R}^{D\times D}, \tag{2.11}$$
$$V = z \times W^v, \quad W^v \in \mathbb{R}^{D\times D}. \tag{2.12}$$

Таким образом, SA для всех входных векторов вычисляться по формуле:

$$SA(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{D}}\right)V. \tag{2.13}$$

Multi-head Self-attention (MSA)

В [27] было обнаружено, что вместо вычисления одной функции SA с матрицами Q, K, V размерности $N \times D$, полезно проецировать матрицы Q, K, V h раз с различными обучаемыми линейными проекциями $W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{D \times d_h}$, где d_h обычно равно D/h. Затем для каждой из этих спроецированных версий Q, K, V параллельно вычисляется SA, получая d_h-мерные выходные значения. Они объединяются (соединяются) до размера $h \cdot d_h = D$ и снова проецируются обучаемым параметром $W^O \in \mathbb{R}^{D \times D}$, в результате чего получаются окончательное значение размера $N \times D$.

$$Z_i = SA(QW_i^Q, KW_i^K, VW_i^V), \quad Z_i \in \mathbb{R}^{N \times d_h}. \tag{2.14}$$

$$MSA(Q, K, V) = [Z_1, \ldots, Z_h]W^O, \quad W^O \in \mathbb{R}^{D \times D}. \tag{2.15}$$

Классификатор

Чтобы выполнить классификацию, используется стандартный подход: в последовательность z_0 (см. формулу 2.5) добавляется дополнительный обучаемый «токен классификации» $z_0^0 = x_{class}$, состояние которого на выходе кодировщика (z_L^0) служит представлением изображения y (формула 2.8). Классификатор реализован в виде MLP:

– при предварительной тренировке (pre-training) – с одним скрытым слоем, на который применяется tanh как функция активации;

– при настройке (fine-tuning) – с одним линейным слоем.

На вход MLP принимает нормализованный $z_L^0 \in \mathbb{R}^{1 \times D}$, то есть y, и выводит оценку класса.

2.2.2 Swin Трансформер

Такие задачи, как семантическая сегментация или обнаружение, требуют обработку или классификацию всех пикселей изображения. Применение ViT в этих задачах не подходит для изображений с высоким разрешением, поскольку вычислительная сложность SA квадратична по отношению к размеру изображения. Чтобы преодолеть эту проблему, в [31] предлагается модель Swin Трансформер общего назначения, которая имеет линейную вычислительную сложность в зависимости от размера изображения.

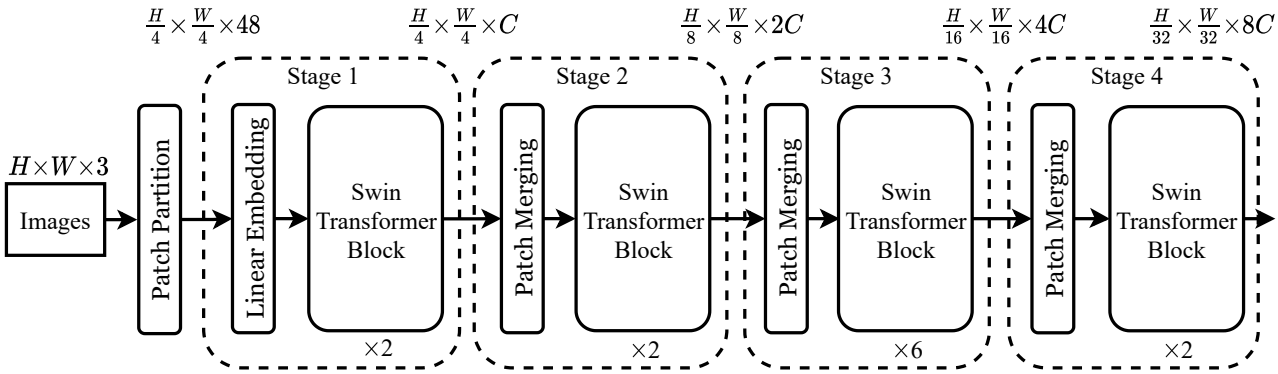

Рисунок 2.12 – Swin Трансформер [31]

На вход изображение RGB разбивается на неперекрывающиеся фрагменты, и каждый фрагмент обрабатывается как «токен», и его признак (feature) задается как конкатенация необработанных значений RGB пикселей. В реализации [31] используется размер фрагмента 4×4, и, таким образом, размер признака каждого фрагмента равно $4 \times 4 \times 3 = 48$. К этому признаку применяется слой линейного embedding-а, чтобы спроецировать его на некоторое измерение C. Несколько блоков Трансформера с модифицированным вычислением SA (блоки Swin Трансформер) применяются к этим токенам фрагментов. Блоки Swin Трансформер не изменяет количество токенов ($\frac{H}{4} \times \frac{W}{4}$) и вместе с слоем линейного embedding-а называются «Этапом 1» (см. рисунок 2.12). Затем применяется слой, объединяющий фрагменты (patch merging, PM), который уменьшает количество токенов до $\frac{H}{8} \times \frac{W}{8}$ и увеличивает размерность по глубине до $2C$. Этот PM слой вместе с повторяющими блоками Swin Трансформер считается как «Этап 2» (см. рис. 2.12). Процедура повторяется дважды, как «Этап 3» и «Этап 4», с разрешениями на выходе $\frac{H}{16} \times \frac{W}{16} \times 4C$ и $\frac{H}{32} \times \frac{W}{32} \times 8C$ соответственно.

Эти этапы совместно создают иерархическое представление изображения с тем же разрешением карты признаков, что и у типичных сетей ConvNet, например, VGG [32] и ResNet [26]. В результате предлагаемая архитектура может заменить «backbone» сети в существующих методах для различных задач компьютерного зрения.

Patch merging (PM)

Чтобы создать иерархическое представление, количество токенов уменьшается с помощью *слоев объединяющих фрагменты (patch merging, PM)*. PM слой группирует каждые 2×2 соседних фрагментов и объединяет их по глубине, что преобразует входные данные из размерности $H \times W \times C$ в $\frac{H}{2} \times \frac{W}{2} \times 2^2C$. Процесс объединения в PM слоях показана на рисунке 2.13.

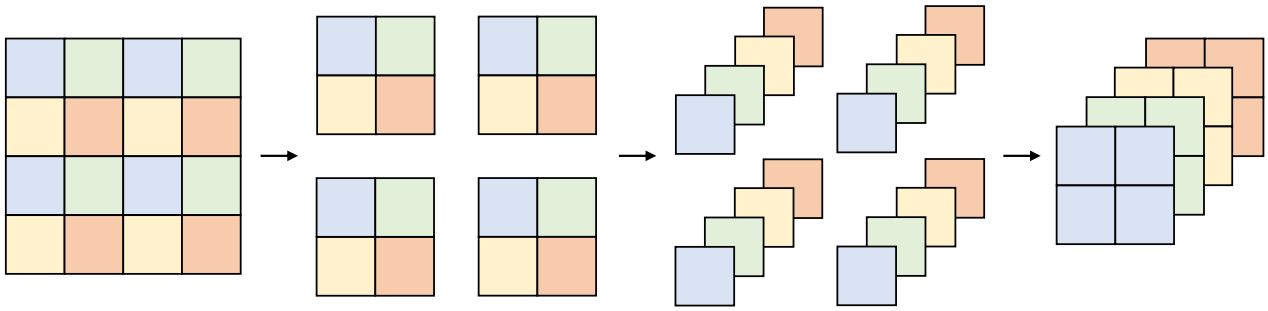

Рисунок 2.13 – Процесс объединения 2×2 соседних фрагментов в слое PM.

Затем на вывод слоя PM применяется линейный слой, который уменьшает размерность по глубине до $\frac{H}{2} \times \frac{W}{2} \times 2C$. В результате, количество токенов уменьшается в $2 \times 2 = 4$ (уменьшение разрешения в 2 раза), а измерение по глубине увеличивается в 2 раза.

Блоки Swin Трансформер

Стандартный MSA, используемый в ViT, вычисляет глобальное SA – взаимосвязь между каждым фрагментом вычисляется относительно всех других фрагментов. Это приводит к квадратичной сложности по отношению к количеству фрагментов, что не подходит для изображений с высоким разрешением.

Чтобы решить эту проблему, Swin Трансформер использует *оконный* MSA *(Window based MSA, W-MSA)*. Окном называют набор фрагментов изображения, и MSA вычисляется только в пределах каждого окна. Например, на

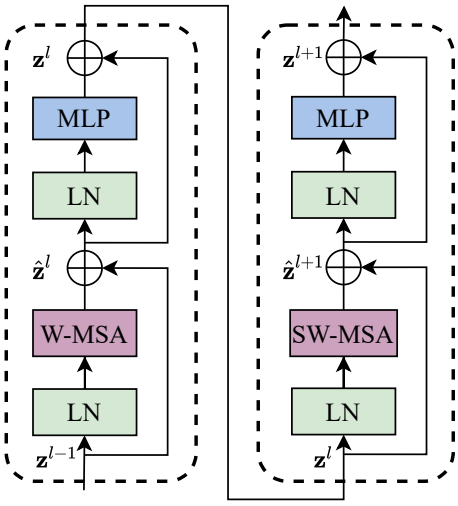

Рисунок 2.14 – Блок Swin Трансформер [31]

рисунке 2.15 используется окно размером 4 × 4 фрагментов, и для каждого окна будет вычисляться отдельный MSA. Поскольку размер окна фиксирован во всей сети, сложность W-MSA линейна по отношению к количеству фрагментов, что является большим улучшением по сравнению с квадратичной сложностью стандартной MSA.

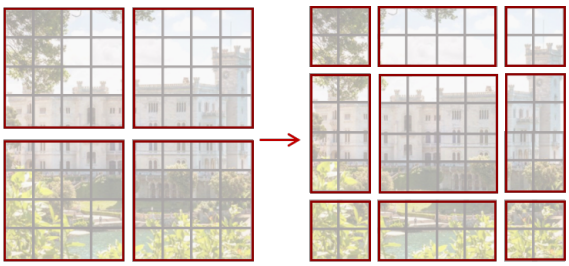

Рисунок 2.15 – W-MSA и SW-MSA [31].

Однако вычисления MSA в пределах окна ограничивает возможности найти корреляцию между всеми фрагментами. Чтобы решить эту проблему, Swin Трансформер использует MSA *смещенного окна (Shifted Window based MSA, SW-MSA)* после W-MSA. После блока Swin Трансформер с W-MSA в блоке с SW-MSA окна размерности $M \times M$ смещаются в правый нижний угол с шагом $\lfloor \frac{M}{2} \rfloor$. На рисунке 2.15 приведен результат смещения 4 × 4 окон в блоке с SW-MSA с шагом 2.

Однако это смещение приводит к появлению фрагментов, не принадлежащих ни к одному окну, а также к полупустым окнам. Swin Трансформер применяет метод «циклического сдвига», который перемещает фрагментов вне окон в неполные окна. После этого сдвига окно может состоять из подокон,

которые не являются соседними в исходных данных, поэтому применяется маска, чтобы ограничить вычисление MSA внутри каждого подокна. Такой подход смещения окон вводит важные перекрестные связи между окнами и улучшает производительность модели.

Таким образом, два последовательных блока Swin Трансформер вычисляются по формуле:

$$\hat{z}^l = W\text{–}MSA(LN(z^{l-1})) + z^{l-1}, \tag{2.16}$$

$$z^l = MLP(LN(\hat{z}^l)) + \hat{z}^l, \tag{2.17}$$

$$\hat{z}^{l+1} = SW\text{–}MSA(LN(z^l)) + z^l, \tag{2.18}$$

$$z^{l+1} = MLP(LN(\hat{z}^{l+1})) + \hat{z}^{l+1}, \tag{2.19}$$

где $\hat{z}^l$ и z^l обозначают выводы модуля (S)W-MSA и модуля MLP для блока l соответственно.

2.3 Гибридные методы

2.3.1 BoTNet

BoTNet является модификацией модели ResNet, точность которого в задачах сегментации и классификации значительно превосходит результаты других методов, которые также используют ResNet в качестве основы. В частности, 3×3 CONV слои в последнем этапе ResNet заменены блоками MSA Трансформера (см. таб. 2.1). При этом в блоках MSA используется SA с представлениями относительного положения [33], в котором учитывается 2D позиция каждого элемента. На таблице 2.1 представлена архитектура ResNet50 и BoTNet50.

2.3.2 ViT$_C$

Модели ViT чувствительны к выбору алгоритма оптимизации (SGD или AdamW), выбору гиперпараметров, специфичных для набора данных, к глубине сети и т. д. По сравнению с ViT современные сети ConvNet исключи-

этап	вывод	ResNet-50	BoTNet-50
c1	512 × 512	7×7, 64, шаг 2	7×7, 64, шаг 2
c2	256 × 256	3×3 max pool, шаг 2	3×3 max pool, шаг 2
		$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times3$
c3	128 × 128	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times4$
c4	64 × 64	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times6$
c5	32 × 32	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times3$	$\begin{bmatrix} 1\times1, 512 \\ \texttt{MSA}, 512 \\ 1\times1, 2048 \end{bmatrix} \times3$
# парам.		**25.5**×10^6	**20.8**×10^6
TPU-v3		**786.5** ms	**1032.66** ms

Таблица 2.1 Архитектура BoTNet-50 и ResNet-50

тельно легко оптимизируются, используя простые методы, как SGD (стохастический градиентный спуск), аугментация данных и стандартные значения гиперпараметров [26].

В [34] утверждается, что эта проблема связана с основой моделей ViT, где изображение делится на фрагменты одинакового размера p с помощью свертки шага $S = p$ с фильтрами размера $p \times p$ (в [28] $p = 16$). Эта свертка с большими фильтрами и большим шагом противоречит типичному выбору конструкции CONV слоев в сетях ConvNet, что и вызывает вышеописанную проблему.

Далее в [34] представляется модель ViT$_C$, в котором основная свертка ViT с большими фильтрами и большим шагом заменяется небольшим количеством сверток шага $S = 2$ с фильтрами размера 3×3, что является типичным для различных ConvNet. Блок из 5 этих CONV слоев имеет примерно такую же сложность, как один блок Трансформер, поэтому, в ViT$_C$ уменьшают количество блоков Трансформеров на один, чтобы сохранить четность по FLOP-ам (floating point operations – единица, используемая для измерения производительности вычисления, показывающая число выполненных опера-

ций с плавающей запятой в секунду) и времени выполнения.

При обучении модели ViT_C было обнаружено заметное улучшение в поведении с точки зрения чувствительности к настройкам оптимизации, а также в конечной точности модели (повышение на 1-2% точности в ImageNet-1k) по сравнению с ViT, сохраняя при этом число FLOP-ов и время выполнения.

2.3.3 U-Net-образные методы

После успеха ViT, для сегментации медицинских изображений было предложено большое количество гибридных моделей, объединяющие архитектуру U-Net и блоки Трансформера или Swin Трансформера. Наиболее успешными являются следующие.

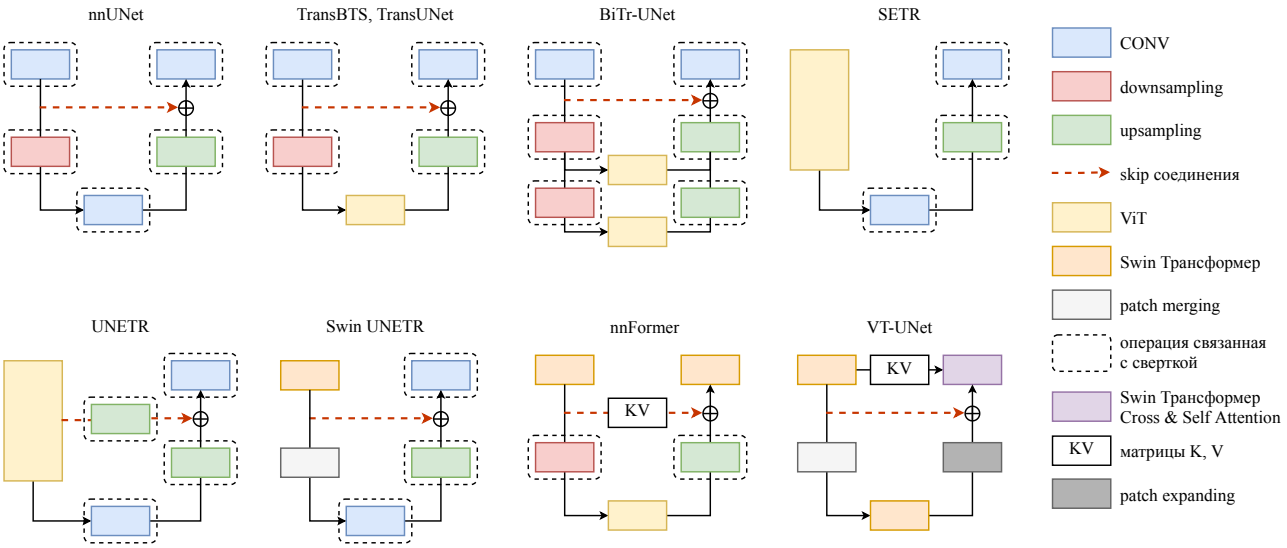

Рисунок 2.16 – Архитектура U-Net-образных методов.

– **TransUNet**. В качестве кодировщика используется 2D ConvNet. Результат кодировщика сглаживаются и передаются блоку Трансформер. Результат блока Трансформер передается ConvNet декодеру, который состоит из повторяющихся 2× upsampling слоев (2D биленейная интерполяция): результаты upsampling слоев объединяются с соответствующими выводами кодировщика и передаются в следующему слою [35].

– **TransBTS**: структура аналогична TransUNet; вместо 2D ConvNet используются 3D ConvNet и в upsampling слоях используется транспонированная свертка [36].

– **BiTr-UTet** основана на TransBTS; вместо 1 блока Трансформера используются 2 [37].

– **UNETR**: в кодировщике изображение обрабатывается как в ViT; размерности промежуточных выводов различных блоков Трансформера увеличиваются слоями транспонированных сверток и объединяются с выводами декодера, который реализован upsampling слоями транспонированных сверток [38].

– **SETR**: в кодировщике изображение обрабатывается как в ViT; декодер реализован 3 разными способами: SETR–NUP – upsampling выполняется билинейной интерполяцией; SETR–PUP – upsampling выполняется с помощью последовательных слоев транспонированной свертки; SETR–MLA – upsampling выполняется с помощью билинейной интерполяцией и CONV слоев. По сравнению с другими методами в SETR не используется skip соединения.

– **Swin UNETR**. Кодировщик основан на Swin Трансформер, состоит из повторяющихся блоков Swin Трансформера и слоев PM, выполняющие downsampling. W-MSA и SW-MSA вычисляются в трехмерных объемах. Промежуточные выводы кодировщика и upsampled (с помощью транспонированных сверток) выводы декодера перед объединением передаются блокам CONV слоев с остаточным соединением [39].

– **nnFormer**. Кодировщик и декодер основаны на блоки Swin Трансформера: W-MSA и SW-MSA вычисляются в трехмерных локальных объемах и называются как LV-MSA (Local Volume-based MSA) и SLV-MSA (Shifted Local Volume-based MSA). В кодировщике PM слои Swin Трансформера заменены на обычные downsampling CONV слои. Результат кодировщика передается блоку обычного Трансформера, где вычисляются обычные MSA в трехмерных глобальных объемах, называются GV-MSA (Global Volume-based MSA). В декодере upsampling реализован транспонированными CONV слоями. Обычное объединение выводов кодировщика и декодера с помощью skip соединения заменено на вычисление SA и это называется как *skip attention*. В частности, с вывода кодировщика образуют матрицы K и V с помощью линейной проекции, а upsampled вывод

декодера считается как Q, и с помощью них вычисляется обычная SA [40].

– **VT-UNet** не имеет CONV слоев и основана исключительно на блоках Swin Трансформера. Как в Swin UNETR, кодировщик состоит из повторяющихся блоков Swin Трансформера и PM слоев. Вывод последнего слоя PM кодировщика передается блоку Swin Трансформер и затем в *patch expanding (PE)* слой, в котором выполняются обратные к PM операции, т.е. размернось по глубине уменьшается, а 2D размернось увеличивается. Декодер состоит из блока Swin Трансформера, в котором кроме обычного SA вычисляется Cross-Attenion (CA). На рисунке 2.17 представлены блоки кодировщика и декодера VT-UNet. Каждый блок декодера получает

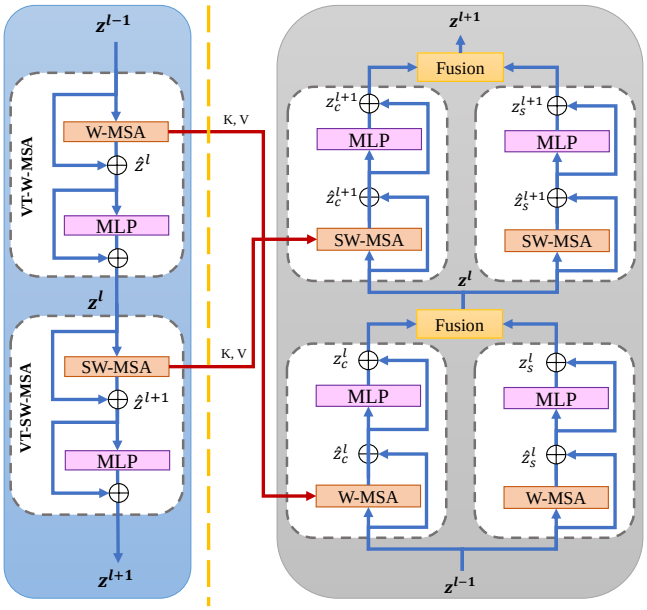

Рисунок 2.17 – Кодировщик и декодер VT-UNet [41].

вывод предыдущего блок декодера, а также матрицы K и V от соответствующего блока кодировщика, находящегося на том же этапе. Правая часть блока декодера (z_s, SA_r) вычисляет SA используя Q_D, K_D, V_D, сгенерированные из предыдущего блока декодера. А левая часть (z_c, CA_l) вместо K_D, V_D предыдущего блока использует K_E, V_E соответствующего блока кодировщика. Выводы отдельных частей z_s и z_c объединяются по

формуле 2.22:

$$SA_r = SA(Q_D, K_D, V_D), \tag{2.20}$$

$$CA_l = SA(Q_D, K_E, V_E), \tag{2.21}$$

$$z^l = \alpha z_c^l + (1 - \alpha) z_s^l + \mathcal{F}(z_s^l), \tag{2.22}$$

где $\mathcal{F}$ – Fourier Positional Encoding, α – коэффициент (в [41] равно 0.5). Вывод блока декодера передается слою PE, а затем объединяется с соответствующим выводом кодировщика skip соединением, и передается следующему блоку декодера [41].

3 Сравнение методов

3.1 Методы сегментации

В [41] представлены результаты сегментации опухоли головного мозга из МРТ изображений с помощью различных методов машинного обучения. Набор данных был получен из 19 различных учреждений, использовался в соревновании MSD (Medical Segmentation Decathlon), также содержит подмножество данных, использованных в соревнование BraTS (brain tumor segmentation) 2016 и 2017 годов.

Набор состоит из 484 многопараметрических МРТ изображений пациентов с диагнозом глиобластома (glioblastoma) или глиома более низкой степени злокачественности (lower-grade glioma). Соответствующими целевыми областями интереса (ROI, region of interest) были следующие подобласти опухоли:

– отек (edema, ED);

– увеличивающаяся часть(enhancing, ET);

– не увеличивающаяся часть (non-enhancing, NET);

– некротическое ядро опухоли (necrotic, NCR).

Методы оцениваются с помощью 95% расстояния Хаусдорфа (Hausdorff Distance, HD95) и оценки Dice (Dice Similarity Coefficient, DSC). При этом оценочные показатели вычисляются отдельно для 3 классов:

– ET;

– TC (tumor core) – ядро опухоли, является объединением ET, NET и NCR;

– WT (whole tumor) – вся опухоль, является объединением ED и TC.

Набор данных из 484 изображений разделен на 80%, 15% и 5% для обучения, валидации и тестирования соответственно. Для реализации использовалось PyTorch с одним GPU Nvidia A40. Для инициализации модели используются веса Swin Трансформера, который предварительно обучен на ImageNet-22K. Для обучения используется оптимизатор AdamW.

В таблице 3.1 приведены показатели HD95 и DSC «state-of-the-art» методов сегментации по подклассам WT, ET, TC, также их среднее (Avg.), где жирным обозначен самый высокий DSC (или низкий HD95), подчеркивается последующий.

Таблица 3.1 Результаты сегментации изображений из MSD BraTS [42] [41].

Метод	Avg.		WT		ET		TC	
	HD95 ↓	DSC ↑	HD95 ↓	DSC ↑	HD95 ↓	DSC ↑	HD95 ↓	DSC ↑
SETR–NUP	13.78	63.7	14.419	69.7	11.72	54.4	15.19	66.9
SETR–PUP	14.01	63.8	15.245	69.6	11.76	54.9	15.023	67.0
SETR–MLA	13.49	63.9	15.503	69.8	10.24	55.4	14.72	66.5
TransUNet	12.98	64.4	14.03	70.6	10.42	54.2	14.5	68.4
UNet	10.19	66.4	9.21	76.6	11.12	56.1	10.24	66.5
TransBTS	9.65	69.6	10.03	77.9	9.97	57.4	8.95	73.5
UNETR	8.82	71.1	8.27	78.9	9.35	58.5	8.85	76.1
nnUNet	4.60	81.9	3.64	**91.9**	4.06	80.97	4.91	85.35
VT-UNet-S	3.84	85.9	4.01	90.8	2.91	81.8	4.60	85.0
nnFormer	4.05	86.4	3.80	91.3	3.87	81.8	4.49	86.0
VT-UNet-B	**3.43**	**87.1**	**3.51**	**91.9**	**2.68**	**82.2**	**4.10**	**87.2**

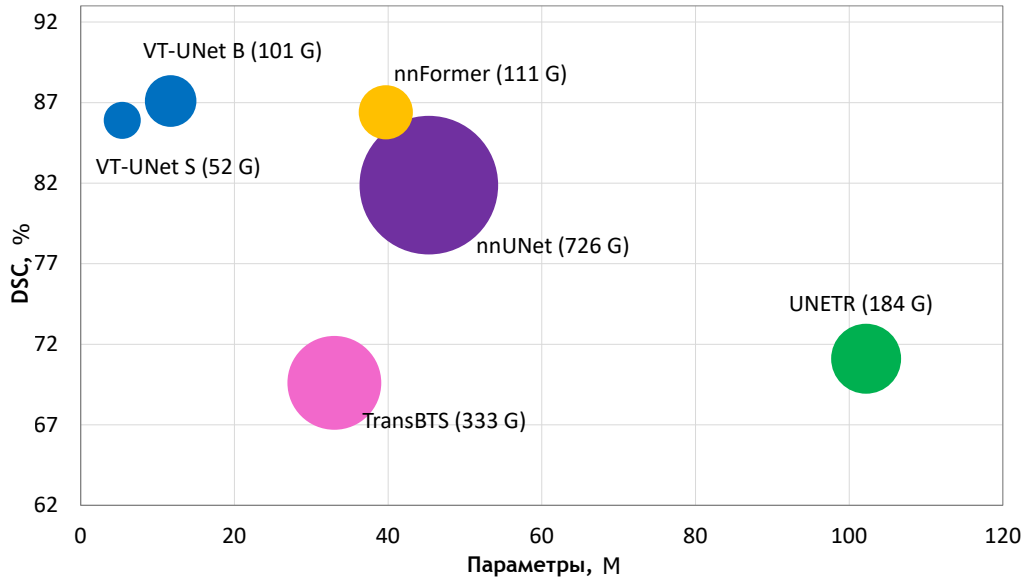

Рисунок 3.1 – Сравнение методов сегментации по DSC (%), количествам параметров (M, миллион) и FLOP-ов (G, gigaFLOPs = 10^9) [41].

Все три версии SETR отстают от других методов, при этом их показатели мало отличаются друг от друга. Это может объясняться отсутствием skip соединений между кодировщиком и декодером, так как UNETR, который также использует блок ViT в качестве кодировщика, но имеет skip соединения, достигает лучших результатов.

TransUNet, состоящий из 2D CONV слоев, работает хуже, чем аналогичная модель TransBTS с 3D CONV слоями. При этом TransUNet также работает хуже чем обычный 2D U-Net.

nnUNet, который автоматически реализует 3D U-Net для данного набора данных, работает лучше чем обычный 2D U-Net, а также методов SETR, UNETR, TransUNet, TransBTS, использующих блока Трансформера. Можно делать вывод, что выполняя подходящую к данным конфигурацию, а также используя подходящий вид свертки, с U-Net можно достигать высоких точностей, не используя блок Трансформера в кодировщике (как SETR, UNETR) или между кодировщиком и декодера (как TransUNet, TransBTS). Но по количеству FLOP-ов (см. рис. 3.1) nnUNet является наименее эффективным среди других методов с высоким DSC.

Приведены результаты двух версии метода VT-UNet: в VT-UNet-S фрагменты изображения спроецируются в вектор с длиной $C = 48$, а в VT-UNet-B $C = 72$. VT-UNet-B достиг наивысшего DSC и наименьшего HD95, сохраняя при этом малое количество параметров и низкую вычислительную сложность. Это подтверждает гипотезу о том, что используя только операции, основанные на Трансформера, можно достигать высоких результатов без использования операции на основе свертки.

nnFormer также достигает высоких DSC и низких HD95 примерно при таком же количестве FLOP-ов как и VT-UNet-B, но имеет почти в 4 раза больше параметров. Такое большое различие в количестве параметров может объясняться тем, что в nnFormer вывод кодировщика обрабатывается блоками ViT с вычислением глобальных MSA. Это также может быть связано с использованием CONV слоев для downsampling и upsampling вместо PM и PE, как в VT-UNet.

Из [9] в таблице 3.2 приведены показатели других методов, основанных на Трансформере, для сегментации изображений набора BraTs 2021 [43]. В отличии от предыдущего эксперимента, тут для VT-UNet-T $C = 48$, для VT-UNet-S $C = 72$, для VT-UNet-B $C = 96$.

Swin UNETR имеет наивысший DSC, но он является наименее эффективным (по числу FLOP-ов) и имеет большое количества параметров.

Как и в прошлом эксперименте, модели VT-UNet имеют высокие DSC, сохраняя при этом малое количество параметров и низкую вычислительную

Таблица 3.2 Результаты сегментации изображений набора BraTS 2021 [43] методами, основанных на Трансформере: количество параметров (M, миллион) и FLOP-ов (G, gigaFLOPs = 10^9), DSC (%). [9].

Метод	**#парам.**	**FLOP**	**DSC**
UNETR	102.5	193.5	84.51
TransBTS	33	333	84.99
BiTr-UNet	–	–	86.20
nnFormer	39.7	110.7	86.56
VT-UNet-T	**5.4**	**52**	86.82
VT-UNet-S	11.8	100.8	87.00
VT-UNet-B	20.8	165	88.07
Swin UNETR	61.98	394.84	**88.97**

сложность.

DSC метода BiTr-UNet, имеющего 2 блока обычного Трансформера между кодировщиком и декодером, выше чем DSC аналогичного метода TransBTS, который имеет 1 блок Трансформера.

Можно делать вывод, что в задачах сегментации использование блоков основанных на Swin Трансформере, а также эффективное использование признаков из кодировщика для формирования вывода в декодере приводят к лучшим результатам. Также можно видеть, что использование многих CONV слоев приводит к высоким FLOP-ам.

3.2 Методы классификации

В статье [44] сравниваются точности методов классификации в 5 общедоступных наборах данных, содержащих целые слайдовые гистопатологические изображения клеток рака молочной железы, желудка и колоректального рака. Наборы данных описаны в таблице 3.3, где в столбцах Train., Valid., Test. приведены количества изображений использованных при обучении, валидации и тестировании методов соответсвенно.

В наборах данных PCam, IDC, GasHis, MHIST изображения классифицированы в классы «доброкачественный» и «злокачественный». Только в BreaKHis классифицируются по другим подтипам. Все наборы данных были маркированы экспертами-патологами.

Таблица 3.3 Наборы данных, используемые в эксперименте [44]

Набор данных	**Вид рака**	**Классы**	**Train.**	**Valid.**	**Test.**	**Всего**
PCam	молочной железы (метастаз)	2	262,144	32,768	32,768	327,680
IDC	молочной железы	2	26,734	10,009	16,410	53,153
GasHisSDB	желудка	2	13,313	6,657	13,314	33,284
MHIST	колоректальный	2	2,175	–	977	3,152
BreaKHis	молочной железы	8	1,005	504	504	2,013

Из тестированных методов, ResNet50 (RN50) и Inception V3 (IV3) являются чистыми сетями ConvNet, ViT$_C$ и BoTNet50 (BN50) – гибридами ConvNet и Трансформера, ViT – чистым Трансформером. Также тестируются IV3 гибриды – методы, объединяющие вышеописанные одинарные методы с Inception V3. Результат гибрида IV3 и некоторого метода M образуется следующим образом: предпоследние слои моделей IV3 и M, содержащие векторы признаков, объединяются и передаются в полностью связанный FC слой-классификатор. Все модели методов были обучены с нуля (без trasnsfer обучения) для каждого из набора данных в течение одинакового количества эпох с использованием Adam в качестве метода оптимизатора. Для устранения неопределенности из-за случайности процесса обучения, выполнено 5 тренировочных прогонов для каждого метода.

В таблицах 3.4 и 3.5 приведены эмпирическое среднее значение и стандартная ошибка достоверности или точности (Accuracy, Acc.) и AUC (значение площади под ROC-кривой, идеальный классификатор имеет AUC, равный 1, худший – 0.5) методов, где жирным обозначен самый высокий показатель в подгруппе «одинарный» или «IV3 гибрид»; подчеркивается самый высокий показатель среди всех методов; помечается знаком $*$ если показатель не сильно отличается от наивысшего показателя подгруппы; помечается знаком $\diamond$ если показатель не сильно отличается от наивысшего среди всех методов показателя.

Таблица 3.4 Результаты эксперимента [44] на наборах данных PCam, IDC, GasHisSDB.

	Метод	Acc. [%]	AUC
	PCam		
одинарный	ResNet50	85.70 ± 0.51	0.9355 ± 0.0080
	Inception V3	**87.36 ± 0.57**	0.9431 ± 0.0046
	BoTNet50	83.35 ± 2.11	0.9094 ± 0.0196
	ViT	78.42 ± 0.42	0.8800 ± 0.0072
	ViT_C	86.40 ± 0.29	**0.9442 ± 0.0016**
IV3 гибрид	GasHis (IV3+BN50)	**87.86 ± 0.26**	0.9464 ± 0.0022
	IV3+RN50	87.80 ± 0.21 ◇ ∗	**0.9469 ± 0.0052**
	IV3+ViT	85.43 ± 0.60	0.9436 ± 0.0033
	IV3+ViT_C	87.43 ± 0.40	0.9427 ± 0.0058
	IDC		
одинарный	ResNet50	87.55 ± 0.11	0.9243 ± 0.0015
	Inception V3	**88.02 ± 0.16** ◇	**0.9318 ± 0.0031**
	BoTNet50	87.98 ± 0.28 ◇ ∗	0.9277 ± 0.0029
	ViT	83.49 ± 1.00	0.9189 ± 0.0006
	ViT_C	87.29 ± 0.29	0.9213 ± 0.0019
IV3 гибрид	GasHis (IV3+BN50)	88.01 ± 0.32 ◇ ∗	**0.9355 ± 0.0026**
	IV3+RN50	**88.05 ± 0.31**	0.9314 ± 0.0026
	IV3+ViT	87.52 ± 0.39	0.9267 ± 0.0013
	IV3+ViT_C	87.66 ± 0.26	0.9262 ± 0.0027
	GasHisSDB		
одинарный	ResNet50	98.09 ± 0.13	0.9980 ± 0.0001
	Inception V3	**98.83 ± 0.05**	**0.9990 ± 0.0001**
	BoTNet50	98.17 ± 0.06	0.9982 ± 0.0001
	ViT	94.24 ± 0.17	0.9860 ± 0.0008
	ViT_C	97.25 ± 0.20	0.9966 ± 0.0005
IV3 гибрид	GasHis (IV3+BN50)	98.79 ± 0.13 ◇ ∗	0.9988 ± 0.0001
	IV3+RN50	**98.80 ± 0.12** ◇	**0.9989 ± 0.0003** ◇
	IV3+ViT	97.76 ± 0.11	0.9974 ± 0.0001
	IV3+ViT_C	98.58 ± 0.04	0.9988 ± 0.0001 ◇ ∗

Среди одинарных методов Inception V3 на всех наборах данных, кроме MHIST, достигает высочайших результатов по обоим показателям. Ее две гибридные модели GasHis(IV3+BN50) и IV3+RN50 также оказались в целом наиболее эффективными. В большинстве случаев за Inception V3 следуют другие методы, основанные на ConvNet. При этом показатели BoTNet50, использующего блок внимания MSA, не сильно превышают показателей своего аналога ResNet50.

В большинстве наборов данных ViT почти всегда достигает самых низких результатов. При этом ViT_C часто превышает ViT. С точки зрения наборов

Таблица 3.5 Результаты эксперимента [44] на наборах данных MHIST и BreaKHis.

	Метод	Acc. [%]	AUC
MHIST			
одинарный	ResNet50	79.32 ± 0.94	0.8672 ± 0.0049
	Inception V3	78.14 ± 0.67	0.8598 ± 0.0086
	BoTNet50	79.90 ± 0.21 ∗	0.8716 ± 0.0044
	ViT	62.48 ± 0.81	0.6480 ± 0.0238
	ViT_C	**81.60 ± 0.97**	**0.8872 ± 0.0094** ⋄
IV3 гибрид	GasHis (IV3+BN50)	**83.01 ± 0.78**	**0.9031 ± 0.0054**
	IV3+RN50	81.58 ± 1.19 ⋄ ∗	0.8932 ± 0.0086 ⋄ ∗
	IV3+ViT	77.79 ± 1.12	0.8462 ± 0.0079
	IV3+ViT_C	80.66 ± 1.11	0.8842 ± 0.0128
BreaKHis			
одинарный	ResNet50	76.74 ± 0.78	0.9477 ± 0.0067
	Inception V3	**91.29 ± 0.66** ⋄	**0.9911 ± 0.0009**
	BoTNet50	78.81 ± 1.90	0.9639 ± 0.0056
	ViT	83.58 ± 0.62	0.9734 ± 0.0026
	ViT_C	75.07 ± 2.34	0.9440 ± 0.0091
IV3 гибрид	GasHis(IV3+BN50)	90.54 ± 0.48 ⋄ ∗	**0.9911 ± 0.0009**
	IV3+RN50	**91.37 ± 0.46**	**0.9911 ± 0.0011**
	IV3+ViT	83.70 ± 0.86	0.9726 ± 0.0021
	IV3+ViT_C	85.29 ± 1.44	0.9791 ± 0.0013

данных MHIST выделяется как единственный набор данных, в котором модель Inception V3 отстает от других ConvNet архитектур и где ViT_C работает лучше всего, в отличие от очень плохо работающего стандартного ViT. Результаты классификации набора данных еще меньшего размера BreaKHis показывают, что такое поведение не может быть полностью объяснено небольшим размером набора данных, и необходимы дальнейшие исследования.

Среди всех методов показатели IV3+RN50 и GasHis (IV3+BN50) оказались наивысшими на 4 из 5 наборов данных. Высокие показатели IV3+RN50 противоречат гипотезу о том, что блок MSA в BoTNet50 является ключевым компонентом успеха GasHis (IV3+BN50). Более вероятно, что высокие показатели IV3+RN50 и GasHis (IV3+BN50) связаны с одновременным использованием параллельных CONV слоев из IV3 и остаточных соединений из ResNet. Следовательно, эти гибридные методы также работают лучше чем их одинарные версии.

Заключение

В ходе выполнения данной работы были рассмотрены виды задач относящиеся к анализу медицинских изображений, описаны широко используемые виды медицинских изображений, изучен принцип работы нейронных сетей. Также были рассмотрены основные методы анализа медицинских изображений. Среди существующих методов машинного обучения было уделено внимание методам на основе глубокого обучения, а именно на ConvNet и Трансформеры.

На основе проделанных анализов и сравнений можно сделать вывод о том, что при решении задач классификации медицинских изображений методы основанные на ConvNet, в частности Inception V3 и ResNet50, или их комбинации являются наиболее эффективными по сравнению с моделями на основе Трансформера. А для задач сегментации медицинских изображений оптимальным являются U-Net образные методы, основанные на Swin Трансформер.

Литература

1. Gobert Lee, Hiroshi Fujita. Deep Learning in Medical Image Analysis: Challenges and Applications. Springer International Publishing, 2020.

2. A Survey on Deep Learning in Medical Image Analysis / Litjens Geert, Kooi Thijs, E. B. Babak [и др.] // CoRR. 2017. Т. abs/1702.05747. – Режим доступа: http://arxiv.org/abs/1702.05747.

3. Shen Dinggang, Wu Guorong, Suk Heung-Il. Deep Learning in Medical Image Analysis // Annual review of biomedical engineering. 2017. Т. 19, № 221–248. – Режим доступа: https://doi.org/10.1146/annurev-bioeng-071516-044442.

4. Liu Zhaoshan, Lv Qiujie, Lee Chau Hung [и др.]. Medical image analysis based on transformer: A Review. 2022. – Режим доступа: https://arxiv.org/abs/2208.06643.

5. Ronneberger Olaf, Fischer Philipp, Brox Thomas. U-Net: Convolutional Networks for Biomedical Image Segmentation. 2015. – Режим доступа: https://arxiv.org/abs/1505.04597.

6. Tajbakhsh Nima, Jeyaseelan Laura, Li Qian [и др.]. Embracing Imperfect Datasets: A Review of Deep Learning Solutions for Medical Image Segmentation. 2019. – Режим доступа: https://arxiv.org/abs/1908.10454.

7. Recent advances and clinical applications of deep learning in medical image analysis / Xuxin Chen, Ximin Wang, Ke Zhang [и др.] // CoRR. 2021. Т. abs/2105.13381. – Режим доступа: https://arxiv.org/abs/2105.13381.

8. Data augmentation using learned transforms for one-shot medical image segmentation / Amy Zhao, Guha Balakrishnan, Frédo Durand [и др.] // CoRR. 2019. Т. abs/1902.09383. – Режим доступа: http://arxiv.org/abs/1902.09383.

9. Shamshad Fahad, Khan Salman, Zamir Syed Waqas [и др.]. Transformers in Medical Imaging: A Survey. 2022. – Режим доступа: https://arxiv.org/abs/2201.09873.

10. Candemir Sema, Rajaraman Sivaramakrishnan, Antani Sameer [и др.]. Deep Learning for Grading Cardiomegaly Severity in Chest X-Rays: An Investigation. 2018. 11. – Режим доступа: https://lhncbc.nlm.nih.gov/LHC-publications/PDF/pub9872.pdf.

11. of Biomedical Imaging National Institute, Bioengineering. Computed Tomography (CT) // Engineering the Future of Health. 2022. – Режим доступа: https://www.nibib.nih.gov/science-education/science-topics/computed-tomography-ct.

12. Fernando Tharindu, Gammulle Harshala, Denman Simon [и др.]. Deep Learning for Medical Anomaly Detection – A Survey. 2020. – Режим доступа: https://arxiv.org/abs/2012.02364.

13. of Biomedical Imaging National Institute, Bioengineering. Magnetic Resonance Imaging (MRI) // Engineering the Future of Health. 2022. – Режим доступа: https://www.nibib.nih.gov/science-education/science-topics/magnetic-resonance-imaging-mri.

14. Rubin Raphael, Strayer David S., Rubin Emanuel [и др.]. Rubin's Pathology: Clinicopathologic Foundations of Medicine. 2008.

15. IBM Cloud Education. Neural Networks. 2020. – Режим доступа: https://www.ibm.com/cloud/learn/neural-networks.

16. Goodfellow Ian, Bengio Yoshua, Courville Aaron. Deep Learning. MIT Press, 2016. – Режим доступа: http://www.deeplearningbook.org.

17. Krizhevsky Alex, Sutskever Ilya, Hinton Geoffrey E. ImageNet Classification with Deep Convolutional Neural Networks // Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1. NIPS'12. Red Hook, NY, USA: Curran Associates Inc., 2012. с. 1097–1105. – Режим доступа: http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf.

18. Hendrycks Dan, Gimpel Kevin. Gaussian Error Linear Units (GELUs). 2016. – Режим доступа: https://arxiv.org/abs/1606.08415.

19. Nielsen Michael A. Neural Networks and Deep Learning. Determination Press, 2015.

20. Gradient-Based Learning Applied to Document Recognition / Yann Lecun, Leon Bottou, Y. Bengio [и др.] // Proceedings of the IEEE. 1998. 12. Т. 86. С. 2278 – 2324. – Режим доступа: http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf.

21. Shafkat Irhum. Intuitively Understanding Convolutions for Deep Learning. 2018. – Режим доступа: https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee

22. Long Jonathan, Shelhamer Evan, Darrell Trevor. Fully Convolutional Networks for Semantic Segmentation // CoRR. 2014. Т. abs/1411.4038. – Режим доступа: http://arxiv.org/abs/1411.4038.

23. Long Jonathan, Shelhamer Evan, Darrell Trevor. Implementation of Fully Convolutional Networks for Semantic Segmentation. 2014. – Режим доступа: https://github.com/shelhamer/fcn.berkeleyvision.org.

24. nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation / Fabian Isensee, Paul F. Jaeger, Simon A. A. Kohl [и др.] // Nature Methods. 2021. Т. 18, № 2. С. 203–211. – Режим доступа: https://doi.org/10.1038/s41592-020-01008-z.

25. Çiçek Özgün, Abdulkadir Ahmed, Lienkamp Soeren S. [и др.]. 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation. 2016. – Режим доступа: https://arxiv.org/abs/1606.06650.

26. He Kaiming, Zhang Xiangyu, Ren Shaoqing [и др.]. Deep Residual Learning for Image Recognition. 2015. – Режим доступа: https://arxiv.org/abs/1512.03385.

27. Vaswani Ashish, Shazeer Noam, Parmar Niki [и др.]. Attention Is All You Need. 2017. – Режим доступа: https://arxiv.org/abs/1706.03762.

28. Dosovitskiy Alexey, Beyer Lucas, Kolesnikov Alexander [и др.]. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. 2020. – Режим доступа: https://arxiv.org/abs/2010.11929.

29. Li Jun, Chen Junyu, Tang Yucheng [и др.]. Transforming medical imaging with Transformers? A comparative review of key properties, current progresses, and future perspectives. 2022. – Режим доступа: https://doi.org/10.48550/arxiv.2206.01136.

30. Bahdanau Dzmitry, Cho Kyunghyun, Bengio Yoshua. Neural Machine Translation by Jointly Learning to Align and Translate. 2014. – Режим доступа: https://arxiv.org/abs/1409.0473.

31. Liu Ze, Lin Yutong, Cao Yue [и др.]. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. 2021. – Режим доступа: https://arxiv.org/abs/2103.14030.

32. Simonyan Karen, Zisserman Andrew. Very Deep Convolutional Networks for Large-Scale Image Recognition. 2014. – Режим доступа: https://arxiv.org/abs/1409.1556.

33. Shaw Peter, Uszkoreit Jakob, Vaswani Ashish. Self-Attention with Relative Position Representations. 2018. – Режим доступа: https://doi.org/10.48550/arxiv.1803.02155.

34. Xiao Tete, Singh Mannat, Mintun Eric [и др.]. Early Convolutions Help Transformers See Better. 2021. – Режим доступа: https://doi.org/10.48550/arxiv.2106.14881.

35. Chen Jieneng, Lu Yongyi, Yu Qihang [и др.]. TransUNet: Transformers Make Strong Encoders for Medical Image Segmentation. 2021. – Режим доступа: https://doi.org/10.48550/arxiv.2102.04306.

36. Wang Wenxuan, Chen Chen, Ding Meng [и др.]. TransBTS: Multimodal Brain Tumor Segmentation Using Transformer. 2021. – Режим доступа: https://doi.org/10.48550/arxiv.2103.04430.

37. Jia Qiran, Shu Hai. BiTr-Unet: A CNN-Transformer Combined Network for MRI Brain Tumor Segmentation // Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries. Springer International Publishing, 2022. С. 3–14. – Режим доступа: https://doi.org/10.1007%2F978-3-031-09002-8_1.

38. Hatamizadeh Ali, Tang Yucheng, Nath Vishwesh [и др.]. UNETR: Transformers for 3D Medical Image Segmentation. 2021. – Режим доступа: https://doi.org/10.48550/arxiv.2103.10504.

39. Hatamizadeh Ali, Nath Vishwesh, Tang Yucheng [и др.]. Swin UNETR: Swin Transformers for Semantic Segmentation of Brain Tumors in MRI Images. 2022. – Режим доступа: https://doi.org/10.48550/arxiv.2201.01266.

40. Zhou Hong-Yu, Guo Jiansen, Zhang Yinghao [и др.]. nnFormer: Interleaved Transformer for Volumetric Segmentation. 2021. – Режим доступа: https://doi.org/10.48550/arxiv.2109.03201.

41. Peiris Himashi, Hayat Munawar, Chen Zhaolin [и др.]. A Robust Volumetric Transformer for Accurate 3D Tumor Segmentation. 2021. – Режим доступа: https://doi.org/10.48550/arxiv.2111.13300.

42. The Medical Segmentation Decathlon / Michela Antonelli, Annika Reinke, Spyridon Bakas [и др.] // Nature Communications. 2022. jul. Т. 13, № 1. – Режим доступа: https://doi.org/10.1038%2Fs41467-022-30695-9.

43. Baid Ujjwal, Ghodasara Satyam, Mohan Suyash [и др.]. The RSNA-ASNR-MICCAI BraTS 2021 Benchmark on Brain Tumor Segmentation and Radiogenomic Classification. 2021. – Режим доступа: https://doi.org/10.48550/arxiv.2107.02314.

44. Springenberg Maximilian, Frommholz Annika, Wenzel Markus [и др.]. From CNNs to Vision Transformers – A Comprehensive Evaluation of Deep Learning Models for Histopathology. 2022. – Режим доступа: https://doi.org/10.48550/arxiv.2204.05044.

45. Dumoulin Vincent, Visin Francesco. A guide to convolution arithmetic for deep learning. 2016. – Режим доступа: https://arxiv.org/abs/1603.07285.

46. Artificial Convolution Neural Network Techniques and Applications for Lung Nodule Detection / Shih-Chung Lo, S.-L.A. Lou, Jyh-Shyan Lin [и др.] // Medical Imaging, IEEE Transactions on. 1995. 12. Т. 14. С. 711

– 718. – Режим доступа: https://ieeexplore.ieee.org/document/476112?arnumber=476112.

47. GasHis-Transformer: A multi-scale visual transformer approach for gastric histopathological image detection / Haoyuan Chen, Chen Li, Ge Wang [и др.] // Pattern Recognition. 2022. oct. Т. 130. с. 108827. – Режим доступа: https://doi.org/10.1016%2Fj.patcog.2022.108827.