

Литературный обзор на тему "глубокое обучение нейронных сетей в задачах компьютерного зрения"

Научно-исследовательская работы (НИР) по теме «Исследования литературы тему оптимизации архитектур, обучения и работы современных нейронных сетей в задачах геологии, решаемых методами компьютерного зрения» в рамках соглашения 22-21-20051 Российской научного фонда в рамках темы «Исследование методов автоматической визуальной оценки содержания асбестового волокна в горной породе на асbestовых карьерах Свердловской области (на примере карьер ПАО «УралАсбест»)».

Михаил Владимирович Ронкин,

г. ЕКАТЕРИНБУРГ, 2022 год.

Содержание

1 Введение	3
2 Анализ проблемной области	6
3 Основы глубоких сверточных нейронных сетей (1980 - 2012)	12
3.1 Неокогнитрон и предпосылки к сверточным сетям	12
3.2 Архитектура ConvNet	16
3.3 Функция активации softmax	19
3.4 Архитектура LeNet 5	21
3.5 Сети глубокого доверия, автокодирующие сети и метод жадного предобучения	28
3.6 Термин "Глубокое Обучение"	30
3.7 Функция активации ReLU	31
3.8 Графические ускорители как основной инструмент обучения нейронных сетей	32
3.9 ImageNet, GPU, Internet - катализаторы прогресса	34
4 Регуляризация обучения нейронных сетей	36
4.0.1 Классические подходы регуляризации нейронных сетей	36
4.0.2 Аугментация данных	37
4.0.3 Инициализация Весовых параметров	40
4.0.4 Регуляризация методом Дропаут(DropOut)	41
4.0.5 Нормализация	43
5 Развитие глубоких нейронных сетей в 2012-2016 годах	48
5.1 Глубокие сверточные нейронные сети AlexNet и ZFNet	48
5.2 Осмысление концепции сверточного слоя в 2012-2014 годах	49
5.3 Идея остаточного обучения (Residual layer)	55
5.3.1 Слой остаточного обучения (Residual layer)	55
5.3.2 Архитектуры на основе ResNet	59
6 Современное состояние глубоко обучения в задачах компьютерного зрения	69
6.1 Мобильные сверточные архитектуры 2016-2018	69
6.2 Попытки использования идеи "слой внимания"	75
6.3 Автоматический поиск архитектур	82
6.4 Архитектуры типа трансформер в задачах компьютерного зрения	92
7 Задача семантической сегментации	100
7.1 Базовые вопросы семантической сегментации.	100
7.2 Подход расширенной головной части	107
7.3 Подход энкодер-декодер	118
7.4 Гибридные подходы	127
7.5 Быстрые подходы к решению задач семантической сегментации	129

8 Задачи обнаружения объектов и экземплярной сегментации	132
8.1 Особенности класса задач	132
8.2 Многоступенчатые архитектуры	135
8.3 Одноступенчатые архитектуры	147
8.4 Быстрые одноступенчатые архитектуры	151
8.5 Многоэтапная экземплярная сегментация	156
8.6 Одноступенчатые архитектуры экземплярной сегментации	163
8.7 Быстрые архитектуры экземплярной сегментации	163
9 Оптимизация нейронных сетей в параллельных устройствах	166
9.1 Центральные процессоры.	166
9.2 Использование графических ускорителей GPU.	167
9.3 Работа с памятью	169
9.4 Особенности ускорителей типа FPGA	170
10 Заключение	172
СПИСОК ЛИТЕРАТУРЫ	174

РЕФЕРАТ

Проведены научно-исследовательские работы (НИР) по теме «Исследования литературы тему оптимизации архитектур, обучения и работы современных нейронных сетей в задачах геологии, решаемых методами компьютерного зрения» в рамках соглашения 22-21-20051 между Российским научным фондом, руководителем проекта и организацией о предоставлении гранта на проведение фундаментальных научных исследований, заключенного между УрФУ и Российским научным по приоритетному направлению деятельности «Проведение фундаментальных научных исследований и поисковых научных исследований малыми отдельными научными группами» за счет средств областного бюджета на реализацию проекта № 22-21-20051 «Исследование методов автоматической визуальной оценки содержания асбестового волокна в горной породе на асbestosных карьерах Свердловской области (на примере карьер ПАО «УралАсбест»)». А также научно-исследовательские работы (НИР) по теме «Исследования литературы тему оптимизации архитектур, обучения и работы современных нейронных сетей в задачах геологии, решаемых методами компьютерного зрения» в рамках договора № 1-22-РНФ (внутренний номер Н764.22CP.30312) от 18.04.2022, заключенного между УрФУ и некоммерческим партнерством «Региональный научно-технический центр» по теме «О предоставлении денежных средств победителю конкурса по приоритетному направлению деятельности Российского научного фонда «Проведение фундаментальных научных исследований и поисковых научных исследований малыми отдельными научными группами» за счет средств областного бюджета на реализацию проекта № 22-21-20051 «Исследование методов автоматической визуальной оценки содержания асбестового волокна в горной породе на асbestosных карьерах Свердловской области (на примере карьер ПАО «УралАсбест»)».

Цель работы: проведение литературного обзора и первичных исследований по теме «Исследование методов автоматической визуальной оценки содержания асбестового волокна в горной породе на асbestosных карьерах Свердловской области (на примере карьер ПАО «УралАсбест»)».

1 Введение

Цифровизация горнодобывающей промышленности стремительно растет, что часто называют 4-ой промышленной революцией или индустрия 4. В настоящее время многие приложения для горной добычи автоматизированы с использованием подходов, основанных на глубоком обучении нейронных сетей (Deep-Learning Neural Network, DL). Примерами приложения DL в обсуждаемой области являются:

- бурение,
- взрывные работы,
- транспортировка материала,
- горнообогатительные работы,

- обработка материала
- и т.д..

Несмотря на широкий перечень приложений DL в горнодобывающей промышленности тут этот подход применялся относительно редко по сравнению с другими видами отраслей. Более того, сегодня не предполагается проведение полной автоматизации этой отрасли, человеческий контроль по-прежнему важен для большинства этапов добычи [1].

Основываясь на предварительном анализе использования DL в горнодобывающей промышленности, можно заключить, что в большинстве случаев применимы системах компьютерного зрения с использованием сверточных нейронных сетей (СНС) [1]. Среди примеров можно привести следующие:

- классификация типов руд [2];
- классификация отдельных типов горных пород [3];
- классификация типов минералов [4];
- распределение оценок размеров частиц [5];
- анализ результатов бурения [6, 7];
- анализ спутниковых снимков участка карьера [8, 9];
- разведка карьера [10];
- распознавание типов местности [11];
- вождение автономных транспортных средств [12];
- распознавание зерен минералов [13];
- оценка карьерного взрыва [14],
- и многие другие.

Среди других приложений, например авторами проведены исследования в области компьютерного зрения для оценки содержания асбеста в каменных кусках. Результаты демонстрируют первоначальные перспективы по этим темам для случаев:

- оценки производительности в заводских условиях [15];
- условиях карьера [16]
- и оценке качества взрывных работ [17].

Использование методов компьютерного зрения в данных приложениях имеет следующие особенности. Типичные системы использования компьютерным зрением с DL, предполагают извлечение большого числа признаков, которые не могут быть формально описаны как обычно. Глубокое обучение позволяет автоматически извлекать сложные признаки и их зависимости даже из необработанных наборов данных. С другой стороны, эта область требует достаточно больших объемов баз данных со всеми вытекающими отсюда проблемами.

Среди таких проблем больших данных основными являются производительность вычислений и объем пространства, который должен иметь хранилище данных. Другой особой проблемой является большое разнообразие распределений важных (информационных) признаков, которые будет использовать алгоритм принятия решений. С этими проблемами может столкнуться практически любая современная нейронная сеть [18].

Общий анализ рассматриваемой области показывает, что в большинстве случаев тут используются только базовые сверточные архитектуры без каких-либо оптимизаций. Другие проблемы: ускорить время вывода и снизить требования к реализации СИС, особенно для низко эффективных устройств. Здесь мы имеем в виду пространство для хранения и оптимизацию вычислений.

Итак, целью настоящего обзора это рассмотрение современных тенденций оптимизации использование нейронной сети глубокого обучения в горнодобывающей промышленности. Больше всего усилий в этом обзор направлен на задачи компьютерного зрения и сверточные нейронные сети.



Рисунок 2.1 – Фото забоя открытого асбестового карьера снимок сделан на Баженовском асбестовом месторождении, г. Асбест, Свердловская область, Россия.

2 Анализ проблемной области

Анализ и оценка эффективности и продуктивности добычи ценных горных ресурсов, в частности, открытым способом (в открытых карьерах), является одной из приоритетных задач в горнодобывающей отрасли [19, 20, 21, 22, 14, 15]. Добычу в карьере можно рассматривать как сбор и переработку некоторых кусков породы, полученных по результатам взрывных работ на большом количестве относительно небольших рабочих мест в карьере (т.н. забоях). Рис. 2.1 демонстрирует пример фото такого рабочего места (забоя). Снимок сделан на Баженовском асбестовом месторождении, г. Асбест, Свердловская область, Россия. Полученные куски породы (см. Рис. 2.1) перемещаются на перерабатывающую фабрику для дальнейшей обработки. В конечном счете, выход продукции для всего открытого карьера получается как среднее для всех используемых забоев. Поэтому в целом предприятие производит общую оценку производительности на карьере. Такая оценка аналогично складывается из оценок для каждого забоя. Поэтому качественная оценка и контроль текущей производительности на каждом из забоев необходим для успешного управления карьером и обогатительным заводом.

Оценка продуктивности на карьере, как правило, проводится либо визуально, либо в лаборатории. Специалисты геологической службы проводят визуальные осмотры практически в режиме реального времени, но с относительно низкой точностью. Стационарный лабораторный контроль обеспечивает достаточно высокую точность, но может потребовать много времени [19]. Однако на практике такой контроль дает лишь

среднюю оценку по всем рабочие места. Дополнительное исследование другого аспекта лабораторного оценка содержания асбеста описана в [23]. С другой стороны, оценки геологической службы (путем визуального анализа) могут быть очень субъективными. Как правило, специалисты геологической службы не могут формально описать свои алгоритмы или критерии для оценки производительности карьера, по которым они принимают решения. В связи с этим такие методы оценки трудно описать научно (формально) и трудно воспроизвести. Кроме того, это требует много времени и довольно дорого обучения соответствующих специалистов. Результаты экспертов часто отличаются от результатов лабораторий. Более того, оценки, сделанные разными экспертами, могут существенно различаться [23, 16]. В литературе исследователи обсуждают методы автоматизации для решения некоторых конкретных задач горнодобывающей промышленности с использованием различных систем компьютерного зрения.

Анализ компьютерного зрения (применительно к задачам горнодобывающей отрасли) показал, что современные системы позволяют автоматизировать некоторые процессы. Большинство работ в этой области посвящено оценке размеров обломков горных пород (*t.e.* фрагментации). Эта задача может быть решена как задача сегментации контура с использованием подхода семантической сегментации (semantic segmentation) см., например, [21, 20, 24] или как задача экземплярной сегментации (объектной сегментации, instance segmentation), [14].

Задача семантической сегментации предполагает решение так называемой попиксельной классификации (в смысле DL). Тут имеется в виду, что выход нейронной сети должен иметь те же пространственные размеры, что и вход, а количество каналов равно количеству классов. Затем каждый пиксель для всех каналов подвергается мягкому повторному взвешиванию (с помощью softmax в виде вектора), чтобы определить его класс.

Среди всех архитектур семантической сегментации в данной области наиболее популярным выбором является U-Net [25] несмотря на то, что она была представлена в 2015 году, эта архитектура остается одной из самых популярных и сегодня. Рисунок 2.2 иллюстрирует архитектуру U-Net и операцию попиксельный softmax. Архитектура U-Net относится к типу архитектур сегментации энкодер-декодер. Основная идея U-Net заключается в использовании карт признаков из кодировщика в декодере. С 2015 года было предложено множество модификаций архитектуры. Большинство из них были предложены для медицинских приложений [26, 27] и некоторые из них для приложений анализа изображений микроскопа [28]. Это связано с тем, что подход U-Net позволяет достаточно точно работать с небольшими деталями изображений в силу сохранения информации в переносимых из энкодера картах признаков.

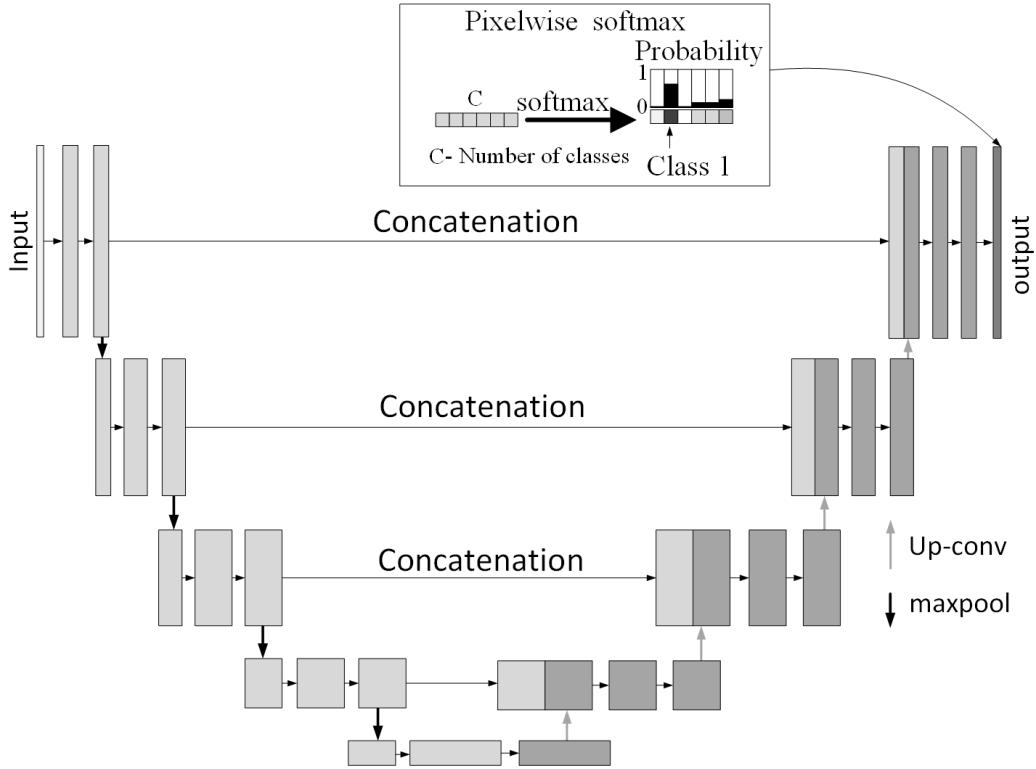


Рисунок 2.2 – Иллюстрация архитектуры U-Net и операцию попиксельный softmax

Задача сегментации экземпляров (объектной сегментации, instance segmentation) в DL обычно рассматривается как задача обнаружения объектов с семантической сегментацией экземпляров в каждой ограничивающей рамке (bounding box) [29]. Кроме этого подхода, может быть несколько других подходов, например, основанных на идее группировки результатов семантической сегментации в объекты [30]. Наиболее распространенной архитектурой для экземплярной сегментации является Mask-R-CNN [31] архитектура которого имеет многоступенчатую структуру. Архитектура Mask-R-CNN состоит из следующих операций.

- Кодировщик признаков выделяет карты признаков для всего изображения (например, ResNet [32] или пирамidalная сеть feature pyramidal network, FPN [33]).
- Сеть предложений регионов для извлечения предварительного набора классов независимых регионов потенциального интереса (regions of interest (ROI), так называемые анхоры (anchors)),
 - анхоры выделяются небольшой сверточной сетью (region proposal network);
 - для анхоров производится оценка объектности и регрессия размеров;
 - для полученных анхоров производится предварительный отбор при помощи процедуры не максимального сжатия (non-maximum suppression NMS).
- Головная часть, включает:
 - часть классификации
 - часть регрессии рамок регионов кандидатов,

- а также часть семантической сегментации для каждого анхора.
- После работы сети алгоритм немаксимального сжатия (NMS) применяется к окончательным предложенными объектам.

Иллюстрация архитектуры Mask-R-CNN показана на рис. 2.3 [31].

Несмотря на то, что Mask-R-CNN был предложен в 2017 году, в настоящее время он по-прежнему остается одним из самых популярных решений для сегментации экземпляров [34].

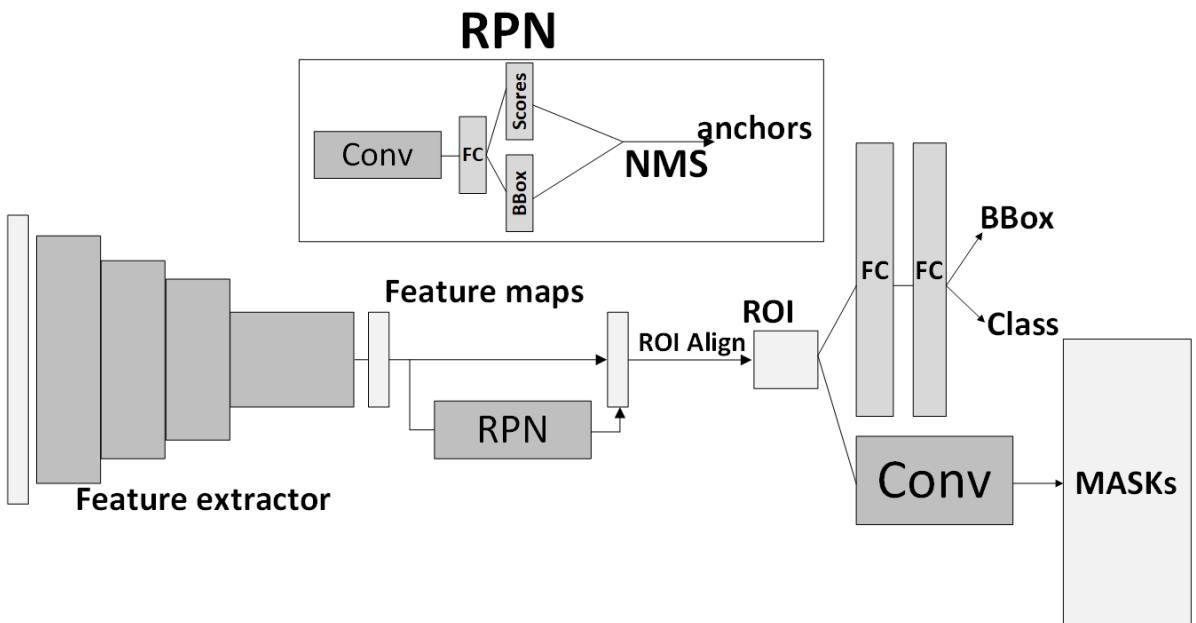


Рисунок 2.3 – Иллюстрация архитектуры Mask-R-CNN с использованием FPN части

Среди работ в рассмотренной области можно выделить следующие основные выводы.

- В работе [21] предложена классификация угля и пустой породы на конвейерной ленте с использованием самдельной СНС. Авторы собрали набор из 300 изображений только с углем, смесью и горными породами. Этого было достаточно, чтобы достичь примерно 90% точности как с предопределенной, так и со случайной аугментациями.
- Авторы [19] предлагают использовать архитектуру U-Net на смешанных изображениях пустой и полезной пород для сегментации пустой породы для дальнейшей сортировки.
- В статье [20] рассмотрена задача определения размера руды как задача контурной сегментации (как часть задачи семантической сегментации). Авторы предлагают слегка модифицированную архитектуру Res-U-Net для определения крупности руды как в условиях конвейерной ленты, так и в условиях карьера. Достигнута точность около 90% в обоих случаях.
- Кроме того, авторы [20] протестировали классический алгоритм водораздела [35], который показывает сравнимую точность для крупномасштабных изображений на конвейерной ленте, но гораздо менее точен для карьера.

- В некоторых других работах были предприняты попытки применить классические методы к условиям карьера. Исследователи из [24] объединили сегментацию U-Net с алгоритмом водораздела [35], чтобы сгруппировать результаты в объекты.
- Кроме того, в работе [36] предложено использовать модифицированный алгоритм только водораздела для сегментации карьера.
- Способ использования Mask-R-CNN был предложен в [14], где авторы достигли точность около 90% для фрагментации в условиях карьерных работ. Кроме того, в работе заявлено, что алгоритмы глубокого обучения лучше работают с традиционными подходами компьютерного зрения для крупномасштабных изображений забоев.
- В работах [14, 37] отмечается связь точности классических алгоритмов с условиями съемки. Такие алгоритмы дают хорошие результаты в условиях: дневное время, достаточная освещенность, сухая погода и другие внешние условия.
- Многие авторы, например [37, 38, 14] констатируют, что традиционные подходы имеют дисперсию результатов, в зависимости от разброса размеров кусков породы, ситуаций перекрывающихся экземпляров и в зависимости от качества изображения для крупномасштабных изображений.
- В статьях [37, 38, 14] сравниваются традиционные подходы и подходы глубокого обучения к крупномасштабным изображениям забоях карьера. Авторы заявили о преимуществах Mask-R-CNN для крупномасштабных изображений в общем случае (в произвольных условиях съемки).

В рамках работ нашего коллектива мы предлагаем идентифицировать фрагменты горных пород из рабочих пространств карьера (забоев) и прожилки полезного продукта (асбеста) внутри выбранных кусков породы. В целом такой подход предполагает следующие этапы.

1. получение изображений забоя открытого карьера.
2. Обнаружение кусков асбестосодержащей породы на изображениях забоя.
3. Получение изображений каждого выбранного куска породы.
4. Сегментация асbestовых жил для каждого изображения породы.
5. Средняя оценка производительности (содержания) асбеста на забое карьера как отношение площадей кусков породы к площади прожилок внутри них.

Среди проделанных в настоящее время работ получены следующие результаты, соответствующие описанной последовательности действий.

- В статье [15] используется архитектуру U-Net с модифицированными блоками для сегментации размеров кусков горных пород и асbestовых прожилок внутри них на конвейерной ленте завода.

- В работе [16] предложено использовать архитектуры Mask-R-CNN для определения кусков породы на изображениях забоя и архитектуры U-Net с модифицированными блоками для сегментации асбестовых прожилок внутри выбранных кусков породы.
- В работе [17] проанализирована возможность решения задачи оценки размеров и определения позиций асбестосодержащей породы на изображениях забоев как задачи обнаружения объектов. Рассмотрена архитектура YOLOv5 [39]. Показано, что подход позволяет восстанавливать распределения размеров пород. Благодаря использованию архитектуры YOLOv5 проблемы с фрагментацией изображения карьера могут быть решены в 10 раз быстрее без значительной потери точности.

Результаты литературного анализа приложений компьютерного зрения в горнодобывающей промышленности показывают, что большинство применяемых методов глубокого обучения основаны на хорошо зарекомендовавших себя методах, таких как U-Net и Mask-R-CNN, с использованием оригинальных экстракторов признаков, предложенных авторами этих работ. Более того, авторы не уделяют достаточного внимания оптимизации предлагаемых ими алгоритмов (моделей). Хорошо известно, что обозначенные подходы являются ресурсоемкими и неэффективными для низкопроизводительных конечных устройств и в задачах реального масштаба времени [40]. Однако, стоит отметить, что исходные архитектуры U-Net и Mask-R-CNN в целом могут применяться в качестве базовых решений (т.н. baseline).

Современный анализ задач сегментации показывает, что архитектуры, решающие задачи сегментации, включают части кодировщика признаков (блоки в U-Net), а также специфические т.н. части шеи, головы. Часть извлечения признаков оказывает наибольшее влияние на точность и вычислительную сложность рассматриваемых архитектур. Тем не менее, в рассмотренных работах не уделяется внимание проблеме выбора кодировщика признаков и оптимизации его работы. Из проводимого обзора можно заключить, что основное решение соответствующих задач состоит в следующем: [41, 42, 43, 44, 45, 46]:

- использование современных специально разработанных для работы в реальном времени архитектур кодировщиков признаков;
- выбор подходящих общих решений для соответствующих задач (семантической сегментации или экземплярной (объектоной) сегментации).
- метод оптимизации, регуляризации и обучения архитектур.

В рамках данного обзора будут рассмотрены эти и другие аспекты использования сверточных нейронных сетей в рамках современных систем компьютерного зрения.

3 Основы глубоких сверточных нейронных сетей (1980 - 2012)

3.1 Неокогнитрон и предпосылки к сверточным сетям

В 1975 г К. Фукушимой была предложена архитектура оригинальной искусственной нейронной сети - когнитрон [47]. Данная архитектура базировалась на работах по изучению принципов функционирования глаза. Одной из наиболее известных работ по этой теме является работа D. Hubel [48]. Также стоит отметить, что в работах D. Hubel 1959 [49] и L.G.Roberts [50] была впервые предложена идея создания систем искусственного зрения как таковых.

Когнитрон стал прообразом использования искусственных нейронных сетей в задачах компьютерного зрения. Когнитрон представлял собой многослойную сеть, каждый слой которой состоял из т.н. ускоряющих и замедляющих нейронов (excitatory, inhibitory). Каждый возбуждающий нейрон следующего слоя был связан с близлежащими нейронами предыдущего слоя. А каждый замедляющий нейрон имел входы с близлежащих нейронов своего слоя и выходные связи с возбуждающими нейронами следующего слоя. Вход каждого возбуждающего нейрона определялся отношением суммы воздействий возбуждающих к тормозящим нейронам предыдущего слоя. Иллюстрация связей двух слоев когнитрона приведена на рисунке 3.4.

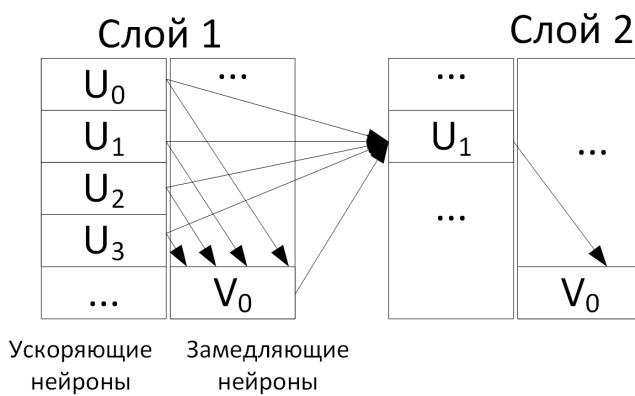


Рисунок 3.4 – иллюстрации работы двух слоев когнитрона.

Отметим, что согласно оригинальной идеи когнитрон имел 2 слоя. Обучение проходило по принципу самоорганизации - таким образом, что выход второго слоя должен был описывать входное изображение.

В 1980 году и С. Мийяке (Fukushima, Miyake) была предложена модифицированная архитектура нейронной сети - неокогнитрон [51]. Основными отличиями новой архитектуры от когнитрона, описанной выше были:

- использование многослойной архитектуры;
- организация нейронов каждом слое в виде двух-мерных структур;

- использование в каждом скрытом слое нескольких т.н. планов;
- использование в каждом слое простых (С-клетка) и сложных (S-клетка) нейронов;
- С-клетки выполняют функции клеток возбуждения в когнетроне;
- S-клетки выполняют функцию выделения максимальной информации (остальная информация локальной области обнуляется) - то есть выполняют функцию нелинейности (функцию активации);
- снижение размеров планов с увеличением глубины слоя таким образом, что последний слой представлял собой одно число на каждый план.

Использование нескольких планов в каждом слое нейронной сети позволяет по разному выучивать сеть одни и те же признаки во входных данных (особенности входного изображения). Использование S-клеток позволяет гарантировать что выделение максимальной информации о каждом признаке в независимости от его расположения. Таким образом целью С клеток является увеличение воздействия на каждый следующий нейрон, а целью S клеток является снижение такого воздействия при минимизации выходной информации нейрона. Архитектура С и S клеток в неокогнитроне называется латентным торможением [52]. Иллюстрация архитектуры неокогнитрона приведена на рисунке 3.5. Понятие "план" в архитектуре неокогнитрона близки по понятию **канал или карта признаков** в современных сверточных нейронных сетях. Понятие С-клетки близко к понятию **свертка**, а понятие S-клетки близко к понятию **субдескстретизация (pooling)** в современных сверточных нейронных сетях. Кроме того использование S-клетки в качестве функций активации близко к понятию т.н. maxout функций активации. Частным случаем maxout является наиболее популярная в настоящее время функция активации ReLU (полулинейная функция активации).

В основе представления данных в архитектурах когнитрон и неокогнитрон, по существу, был положены:

- принцип образования локальных связей;
- принцип конкуренции локальных связей;
- принцип образования **рецептивного поля**.

Описанные принципы заложены в основе современных глубоких сверточных нейронных сетей. Иллюстрации локальной связанности, конкуренции локальных связей и образования рецептивного поля приведены на рисунках 3.6 А), Б) и В) соответственно.

Локальная связанность состоит в том, что каждый нейрон следующего слоя имеет связи только с близлежащими нейронами предыдущих слоев. Таким образом, каждый нейрон отвечает за информацию содержащуюся в определенной пространственно-локализованной области входного изображения (упорядоченного двух-мерного массива данных) (рис. 3.6 (А)). **Конкуренция локальных связей** заключается в том, что каждый участок входных данных может быть по разному учтен в разных нейронах следующего слоя (с разным весовым коэффициентом) (рис. 3.6 (Б)). Принцип локальной

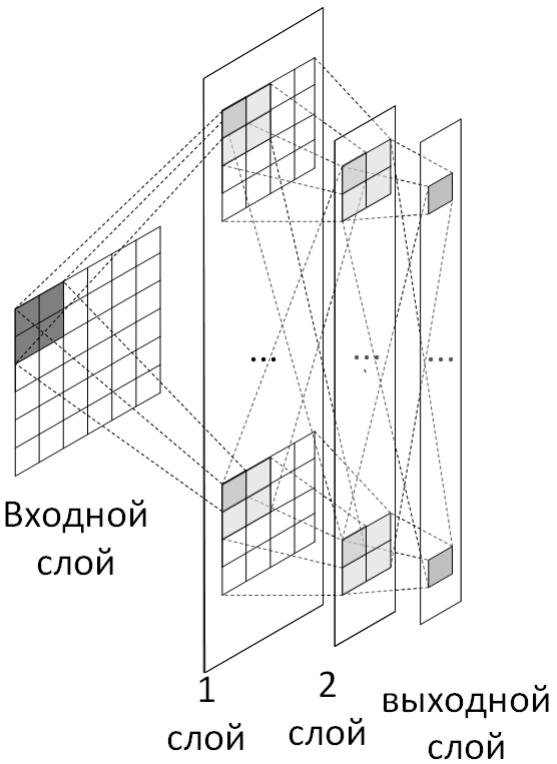


Рисунок 3.5 – иллюстрации архитектуры неокогнитрона. Разделение слоя на C- и S- клетки опущено.

связанности позволяет сжать информацию с предыдущих слоев взвешенным суммированием. Конкуренция локальных связей позволяет избежать потери важной информации с предыдущего слоя за счет ее дублирования в разных нейронах.

Принцип образования рецептивного поля состоит в том, что в многослойной структуре нейроны каждого следующего слоя включают информацию сразу об локальной области нейронов предыдущего слоя. Таким образом, с увеличением количества слоев, каждый нейрон последующего слоя содержит в себе информацию о все большей области входного изображения (рис. 3.6 (B)). В силу локальности связей нейроны начальных слоев содержат в себе сравнительно простые составные части входного изображения, называемые **низкоуровневые признаки**. Каждый низкоуровневый признак сам по себе несет сравнительно мало информации - то есть "почти наверное" не позволяет сделать однозначный вывод о содержании входных данных. Однако, с увеличением числа слоев в нейронной сети возрастает и сложность признаков - каждый признак несет в себе все больше информации о входных данных - то есть повышается вероятность однозначного сопоставления входных данных целевому результату. Описанные признаки называются **высокоуровневые признаки**. Иллюстрация возможности выделения низкоуровневых и высокоуровневых признаков при помощи рецептивного поля многослойной нейронной сети с локальной связью приведена на рисунке 3.7. Отметим, что принцип рецептивного поля был предложен в 1962 г в работе D. Hubel [53] в ходе анализа зрительной системы животных. Также следует отметить работу [54], в которой исследована иерархическая структура глаза и многослойный принцип образования в нем рецептивного поля.

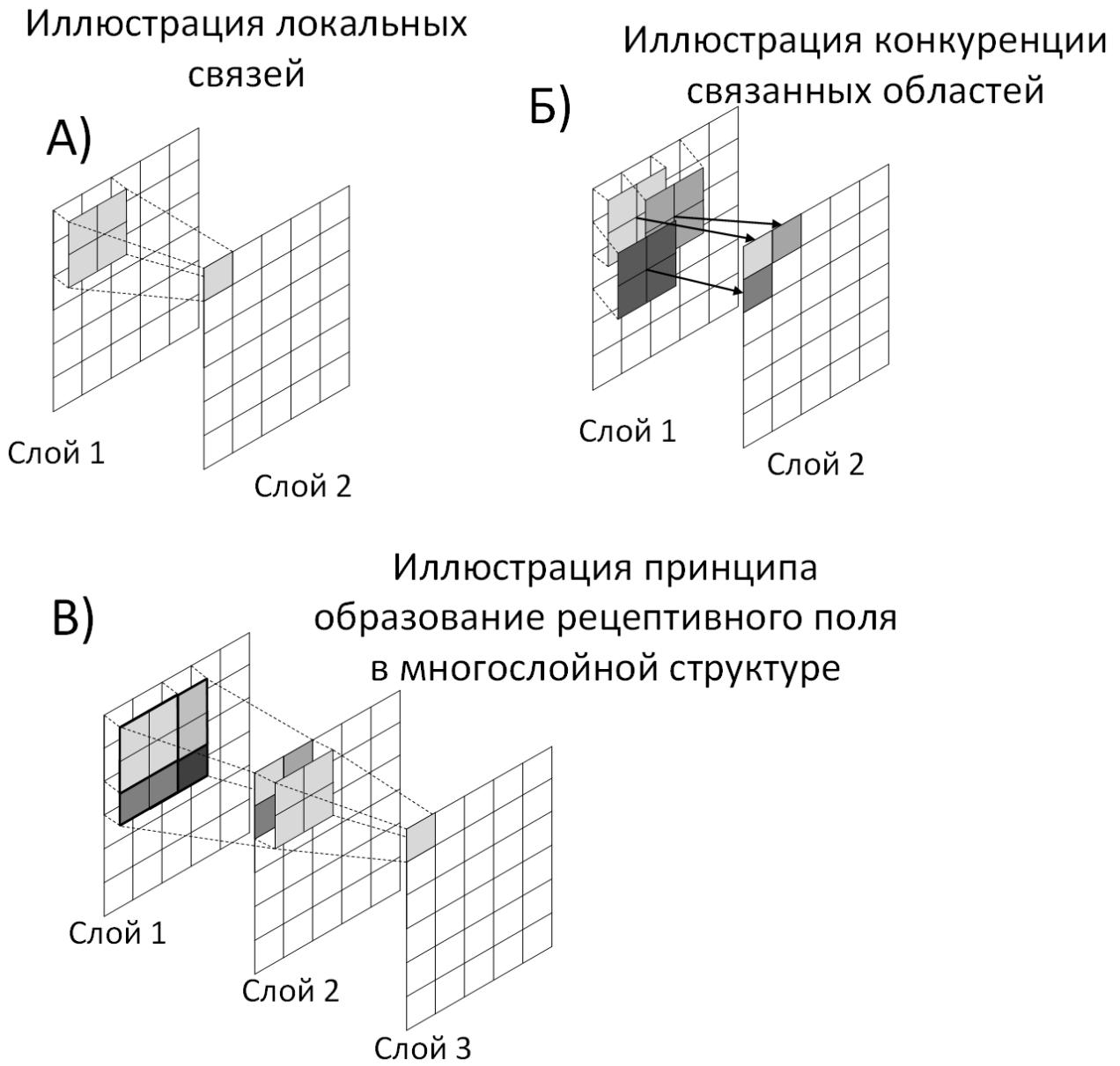


Рисунок 3.6 – иллюстрации локальной области связанности (А), конкуренции областей связанности (Б) и образования рецептивного поля (В).

В 1988 была предложена инвариантная к сдвигу искусственная нейронная сеть (SIANN, shift-invariant artificial neural network) - фактически ставшая прообразом сверточных нейронных сетей [55]. Сеть содержала два скрытых слоя, каждый из которых был организован подобно С-клеткам нейрокогнитрона + функция активации. В качестве функций активации использовались т.н. двойные сигмоиды ($f(z) = 2/(1+\exp(-z))-1$). Сеть обучалась методом обратного распространения ошибки. Варианты модификаций данной архитектуры активно изучались в 1990-х [56]. Однако сегодня данная архитектура может быть рассмотрена только как подкласс сверточных нейронных сетей [57]. Также в 1989 году была описана архитектура одномерной сверточной нейронной сети для решения задач распознавания звуковых фонем [58].

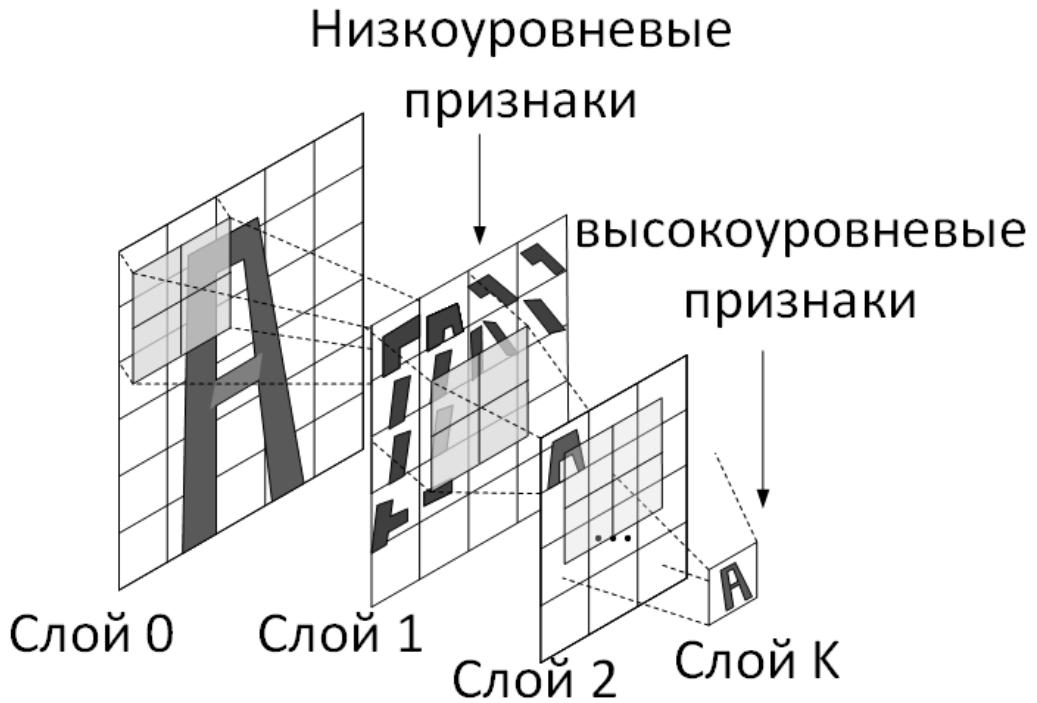


Рисунок 3.7 – Иллюстрация возможности выделения низкоуровневых и высокоуровневых признаков при помощи рецептивного поля многослойной нейронной сети с локальной связью.

3.2 Архитектура ConvNet

В 1989 г. Яном Лекуном была предложена первая сверточная нейронная сеть ConvNet [59]. Нейронная сеть была для решения задачи распознавания рукописных цифр по их фотографиям [60]. Архитектура ConvNet была первой сверточной нейронной сетью (convolutional neural network ,CNN) в современном понимании данного типа архитектур. В частности, в работе [60] термин сверточные нейронные сети (convolutional neural networ, CNN) был впервые популяризован [57]. Нейронная сеть ConvNet была успешно применена для распознавания рукописных цифр на почтовых индексах [59]. Иллюстрация архитектуры ConvNet, описанная в работе [59] приведена на рисунке 3.8.

Архитектура ConvNet обучалась методом обратного распространения ошибки, в качестве функции потерь использовался средний квадрат ошибки. В качестве функций активации - логистические функции. Нейронная сеть имела 3 скрытых слоя, два из которых сверточные и один полно связанный. Выходной слой также полно связанный и имеет 10 выходов по одному для каждой цифры. В основе архитектуры ConvNet (рис. 3.8) лежит понятие двухмерной свертки. Данное понятие является основополагающим для сверточных нейронных сетей. Основная идея использования свертки в нейронных сетях заключается в замене полно связного слоя на одно или несколько сравнительно небольших сверточных ядер (не больших по общему числу параметров).

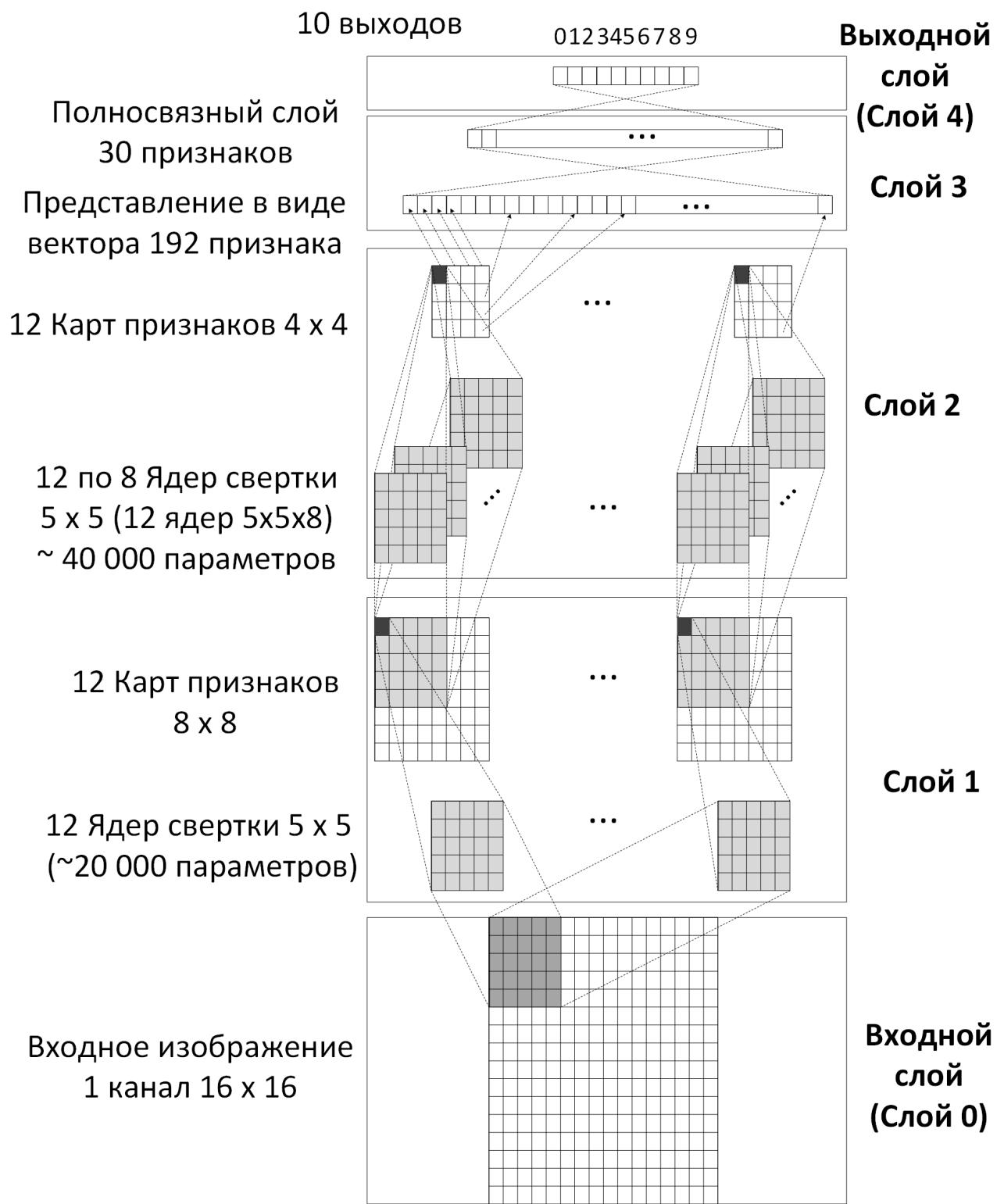


Рисунок 3.8 – Иллюстрация архитектуры сверточной нейронной сети ConvNet 1989 г.

В случае одного двух-мерного ядра и одного канала изображения запишем операцию свертки в следующем виде:

$$r[i, j] = (k * x)[i, j] = \sum_{a=0}^{h_k-1} \sum_{b=0}^{w_k-1} k[a, b] x[i + a, j + b], \quad (3.1)$$

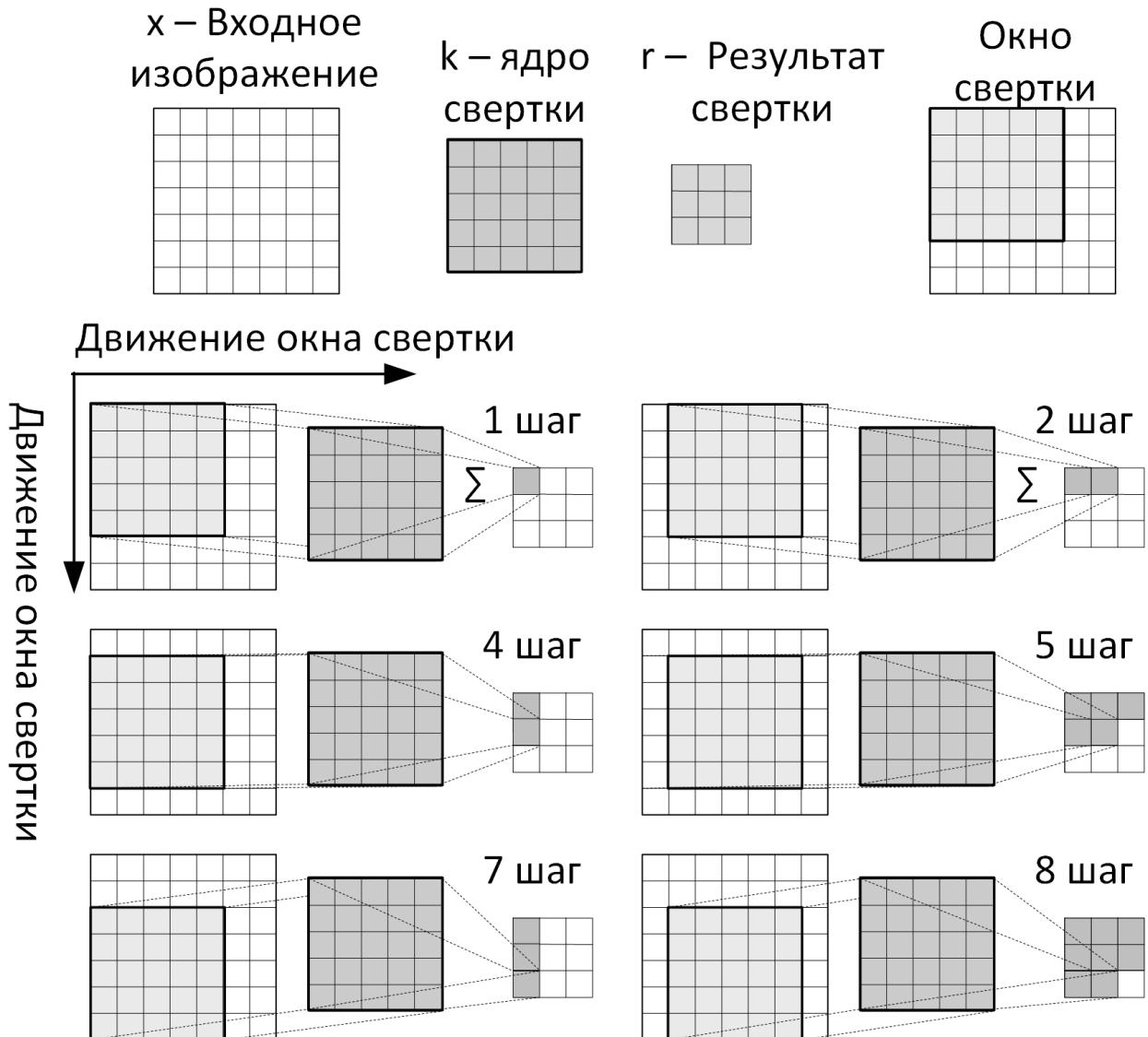


Рисунок 3.9 – Иллюстрация двухмерной свертки 5×5

где:

- $k[a, b]$ - значение ядра свертки с координатами $[a, b]$ (k имеет размерность $h_k \times w_k$);
- $x[i, j]$ - значение входного двухмерного массива с координатами $[i, j]$ (x имеет размерность $h_x \times w_x$);
- $r[i, j]$ - значение результата свертки с координатами $[i, j]$ (r имеет размерность $h_x - h_k + 1 \times w_x - w_k + 1$);
- $*$ - операция свертки.

3.9.

Каждый сверточный слой нейронной сети ConvNet (рис. 3.8) образован несколькими наборами двухмерных сверток. Каждое двухмерное ядро свертки имеет свои значения весовых параметров. Результатом воздействия каждого сверточного ядра является карта признаков. Каждая карта выделяет свои признаки (но признаки одно уровня для одного слоя).

Каждый набор ядер свертки может быть описан многомерной структурой, которая в широком смысле может быть названа **тензор**. Например, слой 1 (рис. 3.8) можно описать при помощи трехмерных тензоров сверточных ядер и карт признаков на выходе слоя. В случае слоя 2 (рис. 3.8) имеется набор трехмерных тензоров сверточных ядер (четырех мерный тензор) и трехмерная карта признаков на выходе. При этом каждый элемент каждой карты образован суммой результатов соответствующих ей двухмерных сверточных ядер со смещением (параметр смещения **-bias** - дополнительный параметр каждого ядра).

Таким образом для выхода слоя 2:

$$r[c_{out}, i, j] = (k * x)[c_{out}, i, j] = \sum_{c=0}^{C-1} \sum_{a=0}^{h_k-1} \sum_{b=0}^{w_k-1} k[c_{out}, c, a, b] x[c, i + a, j + b] + \theta[c_{out}], \quad (3.2)$$

где

- C - число каналов свертки для входного массива;
- $x[c, i, j]$ - значение канала c входного массива с координатами $[i, j]$ (x имеет размерность $C \times h_x \times w_x$);
- $c_{out} = 0, \dots, C_{out} - 1$ - число выходных карт признаков;
- $\theta[c_{out}]$ - смещение результата свертки ядра $k[c_{out}, :, :, :]$ (каждого ядра);
- $r[c_{out}, i, j]$ - значение выходной карты признаков c_{out} ;
- $k[c_{out}, c, a, b]$ - 4-х мерный массив ядер свертки;
- $\theta[c_{out}]$ - смещение ядра свертки.

3.3 Функция активации softmax

В 1989-1990 годах в работах [61, 62] было предложено использование в выходном слое нейронных сетей функцию активации softmax вместо набора выходов, каждый из которых имел свою логистическую функцию активации. Так, например, в работе [60] архитектура ConvNet1989 использовалась для решения задача распознавания рукописных цифр. При этом сеть имела 10 выходных нейронов - каждый для одной цифры от 0 до 9 (см. рис. refch1:fig:convnet1989). Каждый нейрон имел свою логистическую функцию (??). Результатом работы сети выбирался ответ с максимальным значением (**принцип победитель получает все**). Однако такой подход затрудняет интерпретацию результатов, особенно если результатов будет не 10, а скажем 100. *Например, пусть для одной эпохи обучения мы получили для двух выходов сети значения 0,95 и 0,92, а для следующей эпохи 0,96 и 0,955 - как понять на какой эпохе обучение было оптимальным?* Функция softmax позволяет устранить описанную проблему. В случае с функцией softmax результаты всех выходов нейронной сети нормируются так, чтобы в сумме всегда была 1. Тогда каждый результат можно интерпретировать как вероятность того, что, один экземпляр входного набора данных соответствует одному значению выходных данных (например, одному номеру метки класса). Функция softmax и ее производная softmax' может быть записана следующим

образом:

$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_{c=0}^{C-1} \exp(z_c)}, \quad \text{softmax}'(z_i) = \text{softmax}(z_i)(1 - \text{softmax}(z_i)) \quad (3.3)$$

где C - число классов; z_i - входное значение функции - т.н. **логит (logit)**, (например для многоклассовой логистической регрессии $z_i = \sum_{j=0}^{N-1} w_{ij}x_j$, где w_{ij} матрица весов размером $C \times N$, x_j - набор входных значений).

Для реализации функции softmax нейронная сеть должна иметь C выходных нейронов без функций активации для каждого. Вектор результирующих значений выходных нейронов должен быть пересчитан для каждого значения по формуле (3.3). Результатом работы нейронной сети считается номер позиции максимального значения вектора:

$$\hat{y} = \arg \max_i (\text{softmax}(z_i)), \quad (3.4)$$

где $\arg \max_i$ - функция номера позиции с максимальным значением; \hat{y} - результат работы алгоритма. Иллюстрация работы слоя, реализующего выражения (3.3) и (3.4) приведена на рисунке

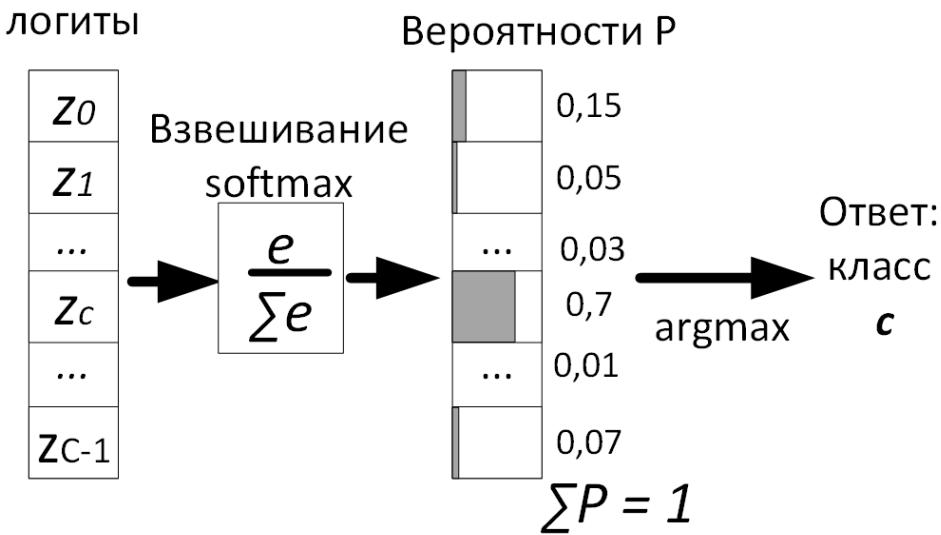


Рисунок 3.10 – Иллюстрация работы выходного слоя нейронной сети с функцией активации softmax

При обучении нейронной сети с функцией активации выходного слоя softmax схема работы будет несколько отличаться от проиллюстрированной на рис. 3.10. Так как во время обучения номер целевого класса известен вместо выражения (3.4) вероятность класса будет оцениваться и максимизироваться именно для целевых классов.

В работе [63] 1997 года преимущества использования функции softmax были показаны для задачи распознавания лиц по фотографиям с использованием сверточной нейронной сети и ряда других алгоритмов.

Отметим, что в современных нейронных сетях задачи классификации можно разделить на три вида:

- **задачи бинарной классификации** - необходимо оценить только наличие или отсутствие одного класса (например, есть или нет кошка на изображении); один выходной нейрон, выходная функция активации - логистическая функция;
- **задачи много-классовой классификации** - необходимо оценить наличие одного из нескольких классов (например, оценить это слон, собака или кошка); выходных нейронов, выходная функция активации softmax;
- задачи много-меточкой классификации - необходимо оценить вероятность наличия по каждому классу одновременно (например одновременно оценить есть ли на изображении и кошки и собаки и слоны) - выходных нейронов,, для каждого выходная функция активации - логистическая функция.

Также следует отметить, что **если решается задача регрессии** (например определить позицию или длину кошки), то используется число нейронов, равное числу оцениваемых параметров, каждый выходной нейрон не имеет функции активации.

3.4 Архитектура LeNet 5

В 1998 Яном Лекуном была предложена модифицированная относительно ConvNet (рис. 3.8) нейронная сеть LeNet5 [64]. Данная нейронная сеть была применена для распознавания рукописных цифр без предварительной обработки изображений. Также в работе [64] был впервые использован набор данных - рукописных MNIST (Modified NIST), который в настоящее время является одним из наиболее популярных в задачах компьютерного зрения (набор доступен по ссылке [65]. Иллюстрация архитектуры нейронной сети LeNet5 приведена на рисунке 3.11. По существу, большинство современных сверточных нейронных сетей (на момент написания книги) являются наследниками архитектуры LeNet5. Более того LeNet5 часто является базовой архитектурой, с которой проводятся сравнения более совершенных архитектур. Также архитектура LeNet стала первой, доказавшей, что системы распознавания, основанные на автоматическом обучение (нейронные сети) могут работать лучше чем системы, основанные на вручную описанных эвристических правилах (классические методы машинного обучения) [64]. Отметим, что помимо архитектуры LeNet5 в работе [64] были рассмотрены и другие варианты систем компьютерного зрения, не показавшие высокой точности в рассмотренной задаче. В частности графовые нейронные сети, сети пространственного смещения, классификаторы на основе метода ближайших соседей и метод опорных векторов, а также некоторые другие.

Архитектура СНС LeNet5 включает в себя 5 слоев (2 сверточных слоя, 2 полносвязных слоя и 1 выходной слой).

В качестве функций активации использовались функции масштабированного гиперболического тангенса:

$$f(x) = \text{Atanh}(Sx), \quad (3.5)$$

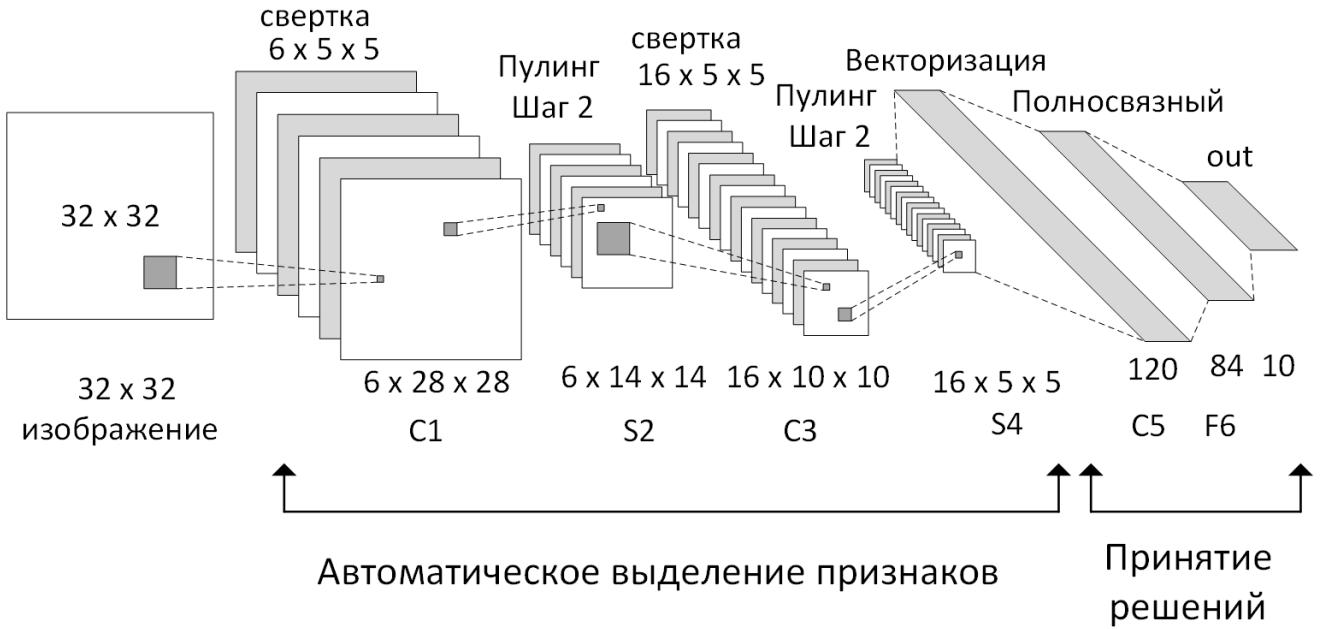


Рисунок 3.11 – Иллюстрация архитектуры сверточной нейронной сети LeNet5

где $\tanh(x)$ - гиперболический тангенс (насыщения $-1 \leq x < 0$ и $x \geq 0$); A и S - регулируемые коэффициенты - т.н. гиперпараметры. Последний слой использовал радиально базисные функции, которые реализованы так, что для каждого из 10 выходов $y_j = \sum_{i=0}^{83} (x_i - w_{ij})^2$. В качестве функции потерь использовалась мульти-целевая биномиальная энтропия + средний квадрат ошибки (multilabel).

В отличии от ConvNet 1989 г в архитектуре LeNet5 каждый сверточный слой дополнен операцией **субдискретизация (пулинг, pooling)** [64]. Идея слоя пулинга уже упоминалась ранее касательно неокогнитрона. Однако, в LeNet она переосмыслена. Слой субдискретизации позволяет снизить размерность карт признаков. Предполагается, что при этом не произойдет потери информации. Для проведения субдискретизации по карте признаков устанавливается окно (например размером 2×2 скользящее по карте с шагом 2 (возможны и другие варианты)). В каждом окне формируется одно значение (в LeNet5 среднее значение - **average pooling**):

$$r \left[\frac{i}{s}, \frac{j}{s} \right] = \sum_{a=i}^{i+w_p} \sum_{b=j}^{j+h_p} x[a, b], \quad (3.6)$$

где:

- $r \left[\frac{i}{s}, \frac{j}{s} \right]$ - результат операции пулинг (размерность $w_r \times h_r$);
- $x[a, b]$ - входной двухмерный массив (размерность $w_x \times h_x$);
- $w_p \times h_p$ - размеры окна пулинга;
- s - шаг перемещения окна пулинга;
- $w_r = \frac{w_x - w_p}{s} + 1; h_r = \frac{h_x - h_p}{s} + 1$.

Иллюстрация работы пулинга приведена на рисунке 3.12. Следует отметить, что использование субдискретизации потенциально может привести к потери информации. По этому каждый слой, содержащий субдискретизацию должен иметь увеличенное число карт признаков по сравнению с предыдущим слоем (как правило удвоенное) - для компенсации возможных потерь. Автором LeNet также постулируется общее положение: снижение пространственного разрешение в нейронной сети должно сопровождаться пропорциональным увеличением числа карт признаков.

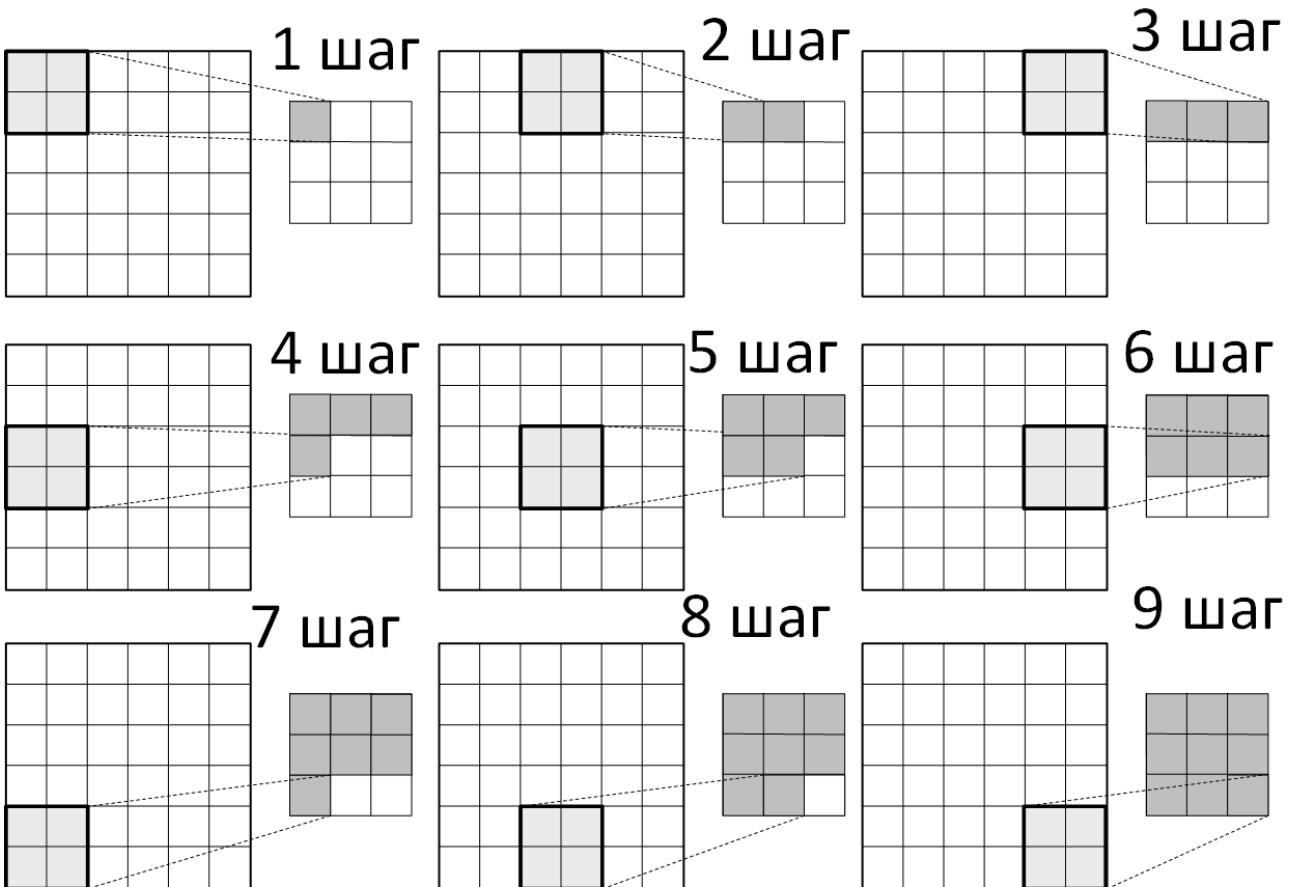


Рисунок 3.12 – Иллюстрация принципа работы пулинга

Также следует отметить, что в настоящее время операция субдискретизации, в виде, описанном выше (3.6)- локальный усредняющий пулинг используется редко. Это связано с неустойчивостью усреднения к шумам и выбросам в данных. Наиболее популярный на сегодняшний день вид субдискретизации - максимальный пулинг (**макспулинг, maxpooling**). Результатом работы макспулинга является число с наибольшей интенсивностью среди всех выделенным окном субдискретизации - то есть с наибольшей информацией.

$$r \left[\frac{i}{s}, \frac{j}{s} \right] = \max_{a,b} x[a,b], \quad a = i, \dots, i + w_p, b = j, \dots, j + h_p. \quad (3.7)$$

Идея макспулинга уже упоминалась в приложении к неокогнитрону, где она называлась S-клетка. Однако, окончательно данная идея была сформулирована в работе [66] в 1992 году [67].

По существу, нейронная сеть LeNet и все архитектуры на ее основе состоит из двух основных частей:

- 1 часть: **энкодер признаков - система автоматического выделения признаков (feature extractor)** - каскад сверточных слоев.
- 2 часть: **головная часть - решения задачи - в данном случае многослойный персептрон.** Данная часть архитектуры может быть выбрана различным способом в зависимости от задачи.

В классических методах машинного обучения предполагается наличие лишь части 2 из описанных выше. При этом признаки - входные данные части 2 в классических методах выделяются вручную (например основываясь на некоторых математических моделях решаемой задачи или эвристически - на основе наблюдений). В архитектуре LeNet не требуется решение проблем подбора или выделения признаков из данных, их описания и т.д. эта задача решается в ходе обучения нейронной сети. Такой подход особенно полезен в системах компьютерного зрения - где, как уже было сказано выше, формальное описание задачи вручную крайне сложно, если вообще возможно.

Также следует отметить, что часть 1 описанного выше разделения архитектуры НС, выделяет признаки аналогично концепции рецептивного поля, описанной выше (см. неокогнитрон). Таким образом, чем глубже сеть (чем больше слоев в энкодере признаков) тем большая область входного изображения может быть обработана. То есть чем больше слоев - тем более высокоуровневый признак и признак большего размера может быть выделен во входном изображении.

Однако, подход LeNet имеет ряд ограничений по сравнению с классическими методами машинного обучения. Во-первых, подход требует большего времени обучения и большего размеры выборки данных - так как помимо головной части системы необходимо обучить энкодер признаков. Во-вторых, НС требует больше ресурсов для работы (память, вычислительные ресурсы) - так как НС имеет больше параметров, чем классические системы. В-третьих, НС не могут обучаться он-лайн - в ходе своей работы. То есть можно обучить НС, использовать ее, и копить новую информацию и данные, затем, если нужно, то необходимо полностью переобучать НС и использовать уже новый результат на практике. Отметим, что на практике данные ограничения не снижают применимости НС в задачах компьютерного зрения и многих других.

По задумке автора архитектура нейронной сети LeNet обладала следующими полезными свойствами инвариантности. Эти предположения полезно указать, так как они являются частью общих достоинств сверточных нейронных сетей перед другими алгоритмами обучения [64, 68].

- **Инвариантность к масштабу** целевой сцены (или объекта) на входном изображении (в некоторых пределах). Инвариантность достигается за счет реализации концепции локального рецептивного поля по средствам регулирования глубины энкодера признаков и размерах ядер свертки.
- **Инвариантность к положению** целевой сцены (или объекта) на входном изображении. Инвариантность достигается за счет переиспользования весовых параметров (использование свертки) в примененная к разным частям изображения. В частности в работе [64] отмечается, что если объект на входном изображении сдвинуть, то и объекты на каждой карте признаков сдвинутся, но не изменится - сам признак останется выделанным. Однако, если на изображении есть несколько сцен и их взаимные позиции изменились, то инвариантность к этой ситуации может не сохраниться [64].
- **Инвариантность к искажению** целевой сцены (или объекта) на входном изображении (в некоторых пределах) - то есть расширение обобщающей способности. Инвариантность достигается за счет использования слоев субдискретизации. Таким образом, из каждой области каждой карты признаков будет выделена лишь наиболее информативная составляющая. При этом предполагается, что небольшие искажения тестовых объектов по отношению к тренировочным не дадут высокой интенсивности - так как под них НС не обучена. Другими словами ядра свертки не коррелируют с искажениями на которых они не обучены и дадут что то около нуля.
- Переиспользование весовых параметров (использование свертки, **пространственная инвариантность весовых параметров**). При этом при использовании сверток, в отличии от нейронов неокогнитрона, в СНС число параметров меньше чем в полносвязных НС. Так для LeNet5 число тренируемых параметров ~ 60000 , а аналогичная полносвязная НС имела бы ~ 340000 .

Отметим также деталь архитектуры LeNet5, полезную с точки зрения дальнейшего изложеия материала: В слои 3 фактически использовалась групповая свертка (4 группы) - то есть каждая выходная карта признаков образовывалась только частью входных карт признаков[64]. Это во-первых позволяет экономить вычислительные ресурсы, а во-вторых позволяет избежать т.н. симметрии слоя - ситуации, когда все карты признаков выделяют один и тот-же признак.

В 1998 году Лекуном также была опубликована работа, посвященная анализу различных приемов улучшения точности LeNet [69]. В частности в работе предложены и описаны достоинства следующих приемов.

- Предлагалось использовать в качестве функции активации **гиперболический тангенс** вместо сигмодиа $f(z) = \tanh(z)$ или его модификацию $f(z) = Atanh(Sz) + Bz$, где A, S, B - константы.

- Предлагалось проводить **нормализацию входных изображений** таким образом, чтобы математическое ожидание (Среднее по всем экземплярам) было равно 0 а дисперсия 1. Это необходимо для понижения вероятности переобучения слоев с сигмоидом, а также стандартизации условий работы сети (например, все весовые параметры можно инициализировать значениями от 0 до 1).
- Предлагалось проводить **инициализацию весовых коэффициентов** каждого слоя нейронной сети случайными величинами с нормальным распределением и дисперсией обратно-пропорциональной числу параметров в слое - **инициализация Лекуна**.

$$W_i \sim N\left(0, \frac{1}{\sqrt{n_i}}\right), \quad (3.8)$$

где n_i - число параметров слоя с номером i , W_i - набор весовых коэффициентов слоя с номером i ; $N(0, 1/\sqrt{n_i})$ - нормальное распределение с 0 математическим ожиданием и дисперсией $1/n_i$. При использовании инициализации Лекуна дисперсия весовых коэффициентов в каждом слое равна 1, что является оптимальной ситуацией с точки зрения снижения вероятности вымывания градиента.

- Предлагалось использовать **стохастический пакетный градиентный спуск (mini-batch stochastic gradient descent, SGD)**. В каждую эпоху обучения предлагалось случайным образом **разделять набор тренировочных данных на пакеты (батчи, batch)** фиксированного размера. Для каждого пакета (или некоторой выборки пакетов) предлагалось проводить процедуру прямого прохождения и обратного распространения ошибки.
- Предлагалось использовать обновления весовых значений методом SGD с т.н. моментом (экспоненциальным сглаживанием, **SGD with moment**). Использование момента позволяет снизить влияние случайности в стохастическом пакетном градиентном спуске. При этом выражение (??) может быть переписано в виде:

$$W^{\{t+1\}} - W^{\{t\}} = -\eta \nabla L(\hat{y}^{\{t+1\}}, y^{\{t+1\}}) + \mu (W^{\{t\}} - W^{\{t-1\}}), \quad (3.9)$$

где μ коэффициент сглаживания. Как правило $\mu \approx 0.9$.

- Предлагалось использовать **адаптивные стратегии изменения скорости обучения**. Подход позволяет снизить влияние неверного выбора скорости обучения не результат, и ускорить обучение (сначала высокая скорость, потом низкая). Позже в книге данные методы будут рассмотрены более подробно.
- Предлагалось использовать **процедуру перекрестной проверки** (кросс валидация, **cross-validation**) в качестве процедуры остановки обучения (т.н. **ранняя остановка, early stop**). Для проведения данной процедуры тренировочная выборка данных должна была быть предварительно поделена на две части - тренировочную (порядка 70%) и валидационную (порядка 30%). Обновление значений весовых параметров происходит только по результатам средней ошибки,

вычисленной по тренировочной выборке. Затем результат работы сети (ошибка работы) вычисляется для валидационной выборки. Если ошибки на тренировочной выборке падает, а на валидационное нет (в насыщении или растет), то такая ситуация называется **переобучение (overfitting)**. Если ошибка на валидационной выборке близка и снижается вместе с ошибкой на тренировочной выборке, то такая ситуация называется **недообучение (underfitting)**. В момент, когда ошибка на тренировочной и валидационной выборке равны друг другу (зависимости пересекаются) обучение считается оптимальным. Таким образом, валидационная выборка позволяет определить эпоху обучения НС в которую значения весовых коэффициентов имеет значения, позволяющие получить максимальную обобщающую способность. Иллюстрация процесса обучения (график зависимости ошибки обучения $L(\hat{y}, y)$ от эпохи (ep) приведен на рисунке 3.13. Также на рисунке отмечены ситуации недообучения, переобучения и эпохи ранней остановки ep^* - оптимального обучения. Отметим, что при обучении нейронной сети весовые параметры учатся выделять признаки из входных данных. Ситуация переобучения нейронной сети соответствует тому, что сеть начинает воспринимать искажения или другие помехи/шумы в тренировочном наборе данных как признаки. При этом сеть старается подстроиться под них, что позволяет добиться точности на тренировочном наборе вплоть до 100%. Однако, НС запоминает на наиболее общие признаки, а частные искажения данных тренировочной выборки, то на валидационных данных ошибка не будет снижаться. Ситуация недообучения соответствует тому, что НС не научилась выделять все необходимые признаки, необходимые для правильного принятия решений.

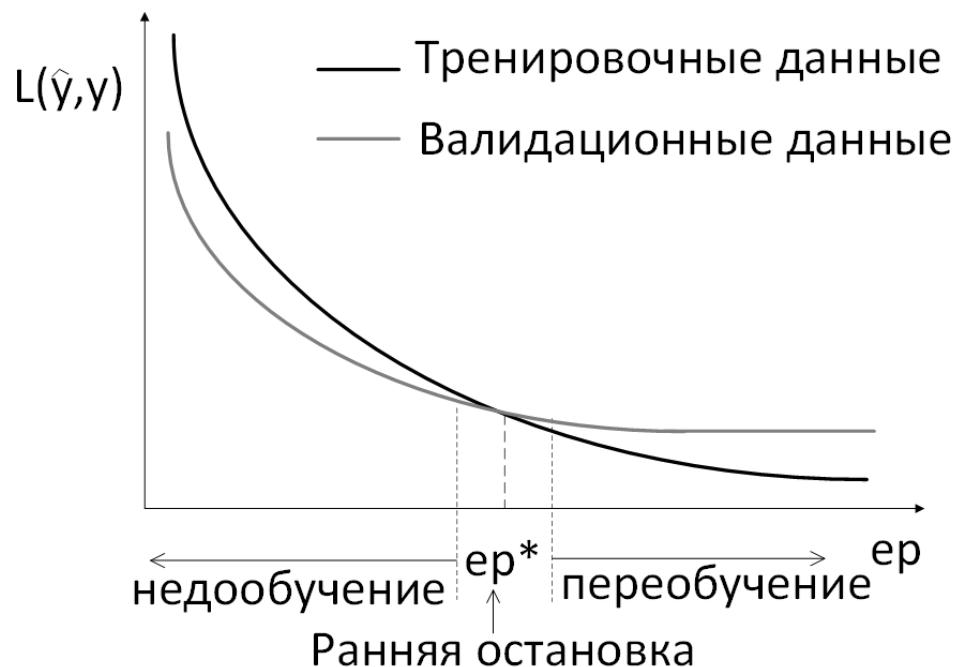


Рисунок 3.13 – Иллюстрация зависимости ошибки обучения от эпохи для тренировочной и валидационной выборок, ep^* - эпоха ранней остановки

Работа [69] стало обобщением опыта по обучению нейронных сетей на момент ее написания. Многие из приемов, описанных в [69] используются в НС компьютерного зрения по настоящее время.

3.5 Сети глубокого доверия, автокодирующие сети и метод жадного предобучения

В середине 90-х и до 2005 (порядка 10 лет, не считая работ некоторых авторов) интерес к изучению нейронных сетей упал в силу технических ограничений вычислительной техники и проблемы вымывания градиента. Однако, например, по оценкам Лекуна [68] в конце 90-х около 10% чеков в банках США обрабатывались с использованием сверточных нейронных сетей.

В 2006 год Джоном Хинтоном была предложена техника предобучения нейронных сетей при помощи жадного послойного обучения [70, 71]. Данный подход частично позволял избежать проблем выявленных ранее [72]. В основе данной идеи лежала архитектура нейронной сети - ограниченная машина Больцмана [73, 74], - архитектура предложена в 1986 г, полная машина Больцмана предложена в 1985 г. на основе т.н. сети Хопфилда (1982 г) ([75]) - подробное изложение данных архитектур выходит за рамки данной книги. В 2006 году ограниченная машина Больцмана была применена Хинтоном для решения задачи кодирования изображений (сжатие изображений) [76]. Данная архитектура нейронной сети в настоящее время известна как **автоэнкодер**. Отметим, что технически идея автокодирования была описана в 1987 в работе [77] где была названа автоассоциативной сетью. Идея автокодирования в данной архитектуре основывалась на гипотезе о том, что многослойные нейронные сети позволяют выделять внутренние (закодированные) представления о регулярностях в данных (признаках) в окружающей среде (в обучающем наборе данных). Отметим, что также в работе [77] была указана гипотеза о преимуществах иерархического (многослойного) кодирования входных данных [77].

Типичная архитектура автоэнкодера представлена на рисунке 3.14. Сеть стоит из двух частей:

- часть 1: **энкодер** - сжатие входных данных - автоматическое выделение из них признаков. Принцип работы данной части аналогичен энкодеру признаков LeNet 5. Выход данного слоя часто называется **скрытым или латентным пространством (latent space)**.
- часть 2: **декодер** - автоматическое восстановление данных из входных признаков (скрытого пространства).

Сеть автоэнкодер обучается без учителя таким образом, что выходное изображение должно повторить входное. Ошибка воспроизведения минимизируется в ходе итерационного обучения. Выход обученного энкодера используется как кодирующая сеть. В случае необходимости закодированное изображение может быть восстановлено при помощи декодера.

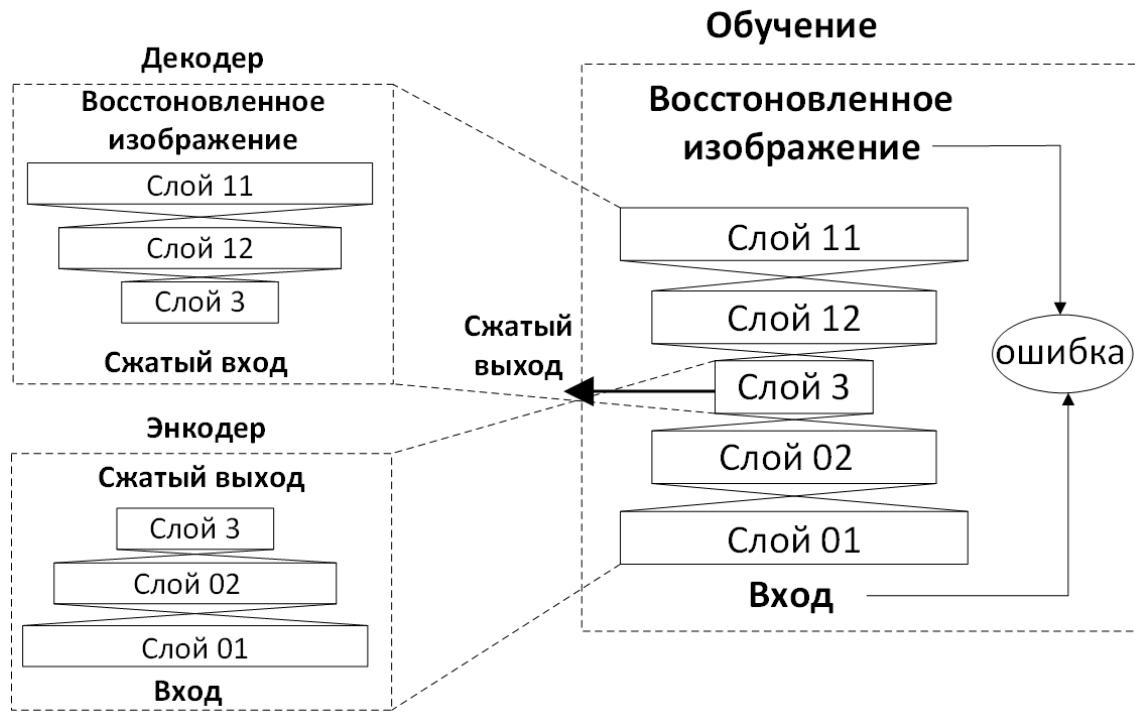


Рисунок 3.14 – Иллюстрация архитектуры автоэнкодера

Техника жадного послойного обучения представляет собой использование автоэнкодера для каждого слоя нейронной сети. Иллюстрация данного подхода приведена на рисунке 3.15. Для первого слоя входными данными являются целевые входные данные НС. Результат предобучения первого слоя используется в качестве входных данных для предобучения второго слоя и т.д. Нейронные сети, предобученные методом жадного послойного предобучения часто называются **нейронные сети глубокого доверия**. [72].

После предобучения (**pretrained**) НС должна быть дообучена (**fine tuning**) для решения целевой задачи. Например, в методом обратного распространения ошибки - в случае задачи обучения с учителем.

Идея предобучения нейронных сетей заключается в ожидании того, что весовые параметры обученных сетей будут настроены таким образом чтобы максимально эффективно выделять характерные признаки из входных данных. При этом ожидается, что приблизительно те же признаки должна научиться выделять НС в ходе обучения так как признаками должны быть все наиболее регулярные особенности тренировочной выборки. Другими словами весовые параметры предобученной НС почти наверное позволяют использовать НС для решения целевой задачи. В работе 2010 года [78] на основании большого числа экспериментов авторами было показано, что использование предобучения нейронных сетей является методом регуляризации обучения нейронных сетей для целевой задачи. Также было показано, что использование предобучения НС повышает обобщающую способность и снижает дисперсию ошибок для тестовых данных. Другими словами снижается как ошибка на тренировочных данных, так и ее разность с ошибкой на тестовых данных, так и дисперсия ошибки на тестовых данных [78].

Послойное предобучение

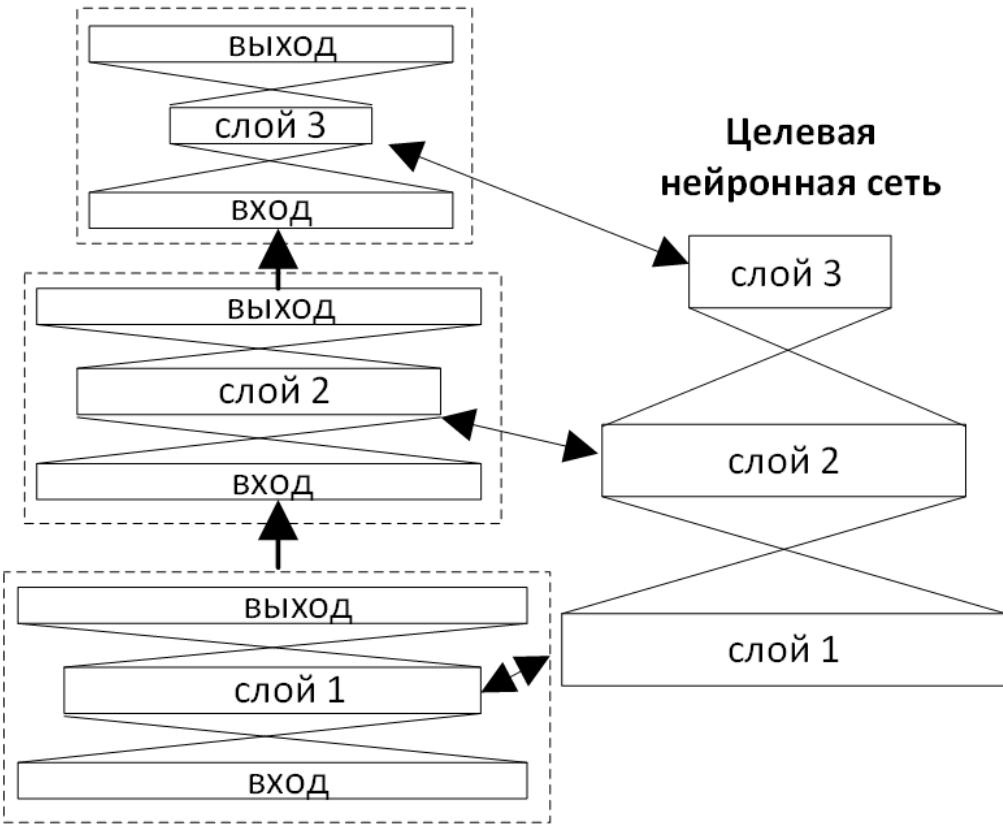


Рисунок 3.15 – Иллюстрация принципа жадного послойного предобучения (сети глубокого доверия)

В 2007 году Й. Бенджо обобщил результаты Д. Хинтона и показал, что использование предобучения нейронных сетей позволяют тренировать нейронные сети гораздо большей глубины [79]. Также отметим, что Д. Хинтона, Й. Бенджио и Я. Лекуна на 2006 год работали совместно в рамках программы CIFAR NCAP[72]. В 2004-2006 годах в работах [80, 81, 82] были показаны преимущества использования графических ускорителей (Graphic Process Unit, GPU) при обучении сверточных нейронных сетей. Фактически с описанных работ начинается **эпоха глубокого обучения нейронных сетей**.

3.6 Термин "Глубокое Обучение"

Идея **глубокого обучения (Deep Learning) нейронных сетей** была высказана в работе Бенджо и Лекуном в работе [83]. В данной работе авторы призвали сообщество исследователей методов машинного обучения к разработке алгоритмов, позволяющих максимально автоматизировать процесс выбора признаков, необходимых для решения задач. Также авторы показали, что необходимым условием выделения наиболее абстрактных признаков (высоко-уровневых признаков) является увеличение глубины нейронных сетей. Для глубоких нейронных сетей было показано, что с увеличением глубины нейронных сетей возрастает их обобщающая способность. При этом обобщающая способность обусловлена нелинейностями в НС [83]. Также отмечается, что для сверточных нейронных

сетей все признаки являются пространственно локализованными. Признаки различных уровней образуют иерархию, в которой локальные высокоуровневые признаки формируются нелинейными комбинациями более низкоуровневых локальных признаков. То есть такая сеть должна реализовывать так называемый принцип рецептивного поля [68].

Отметим, что авторы [83] в противоположность глубокому обучению авторы понятие мелкого обучения (**shallow learning**) - нейронные сети с одним или небольшим количеством слоев. При этом, обещающая способность мелкой нейронной сети можно назвать локальной. Преимущества глубокой нейронной сети в этом случае можно обосновать обобщением большого количества локальных областей обобщения. Также авторами [83] была отмечена одна из основных проблем глубокого обучения - т.н. "**проклятие размерности**" как основного фактора, ограничивающего углубление нейронных сетей. Проклятие размерности заключается в экспоненциальном росте числа параметров слоя НС при линейном увеличении числа входов. Решение данной проблемы является одним из трендов развития НС по настоящее время. При этом авторами [83] была выдвинута гипотеза, что переход от однослойной архитектуры к многослойной позволяет сократить число параметров, необходимое для решения задачи.

Другими словами, главное преимущество глубокого обучения нейронных сетей состоит в возможности полностью автоматического выделения, преобразования и выбор признаков, релевантные поставленной задачи. Процедура обучения требует достаточно большого набора данных для правильной обработки таких признаков. Сами выделенные признаки как правило слабо интерпретируемы экспертами и часто не могут быть объяснены.

3.7 Функция активации ReLU

В 2009-10х годах в работах [84, 85] было предложено использование ректифицированных функций активации вместо логистической функции активации и других вариантов функций с насыщением (ректифицированных - исправляющих, например операция модуль в качестве функции активации). Среди различных вариантов ректификационных функций наилучших результатов удалось добиться для функции **ректификационный линейный модуль (ReLU - rectified linear unit)**. График функции ReLU и ее производной ReLU'(z) представлены на рисунке 3.16. Функция ReLU(z) и ее производная ReLU'(z) имеют следующий вид:

$$\text{ReLU}(x) = \max(0, x), \quad \text{ReLU}'(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases} \quad (3.10)$$

Использование функции ReLU позволяет частично избежать проблем, связанных с вымыванием градиента. Так как функция не имеет насыщения в области $z > 0$. При этом функция имеет низкую вычислительную сложность и позволяет обнулить часть весовых коэффициентов, что также может исключить часть вычислений (если один из параметров 0, то умножение на него можно не проводить). Функция ReLU является одной из наиболее популярных в настоящее время. Как правило, нейронные сети с использованием ReLU

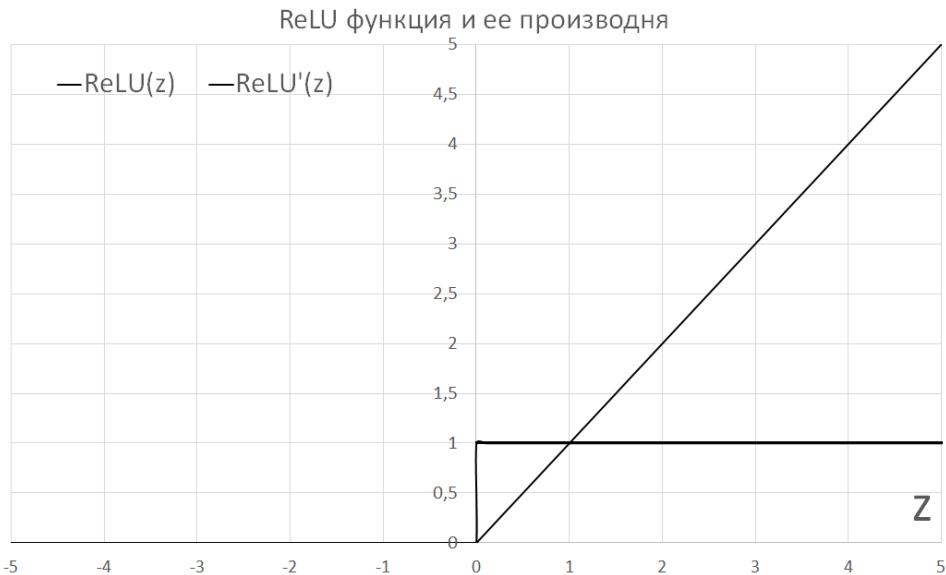


Рисунок 3.16 – график функции ReLU и ее производной

показывают наилучшие результат среди других функций активации. Однако, потенциально у функции ReLU есть ряд недостатков, о которых будет сказано в свое время. Также следует отметить, что в настоящее время по мимо описанной реализации ReLU в литературе предложено семейство линейных и нелинейных модификаций этой функции активации, которые могут быть использованы если стандартной реализации недостаточно [86].

3.8 Графические ускорители как основной инструмент обучения нейронных сетей

Как уже было отмечено выше впервые преимущества использования графических ускорителей при обучении нейронных сетей были показаны в работах [80, 81, 82]. Так в работе [80] было заявлено 12-кратное ускорение обучения нейронной сети с использованием GPU по сравнению с обучением нейронной сети на центральном процессоре (ЦПУ) компьютера. Преимущества GPU заключаются в их возможности максимально быстро выполнять перемножения векторов и матриц - то есть выполнять операции умножения с накоплением над одномерными и двухмерными массивами числе с плавающей запятой с высокой степенью параллельности. Отметим, что операция умножения с накоплением для чисел с плавающей запятой часто называется flop (float-point operation) большинство линейных операций (например $W^T X = \sum wx$) сводятся к операциями типа flop.

Обучение на GPU в 2004-2006 годах (в работах [80, 81, 82]) представляло высокую сложность с точки зрения программного обеспечения. В силу технических ограничений существовавших на тот момент GPU разработка методов прямого прохождения через нейронную сеть и обратного распространения ошибки приходилось выполнять при помощи инструкций, предназначенных для расчета графических операций. В 2007 году компанией NVIDIA была представлена среда разработки под GPU CUDA (фреймворк CUDA), а также семейство графических ускорителей общего назначения (general purpose gpu, GPGPU).

Использование CUDA и GPGPU позволило значительно упростить процесс разработки программного обеспечения для ускорения обучения нейронных сетей. Фреймворк CUDA имеет Си-подобный синтаксис и позволяет вести достаточно гибкую и быструю разработку программного обеспечения для графических ускорителей компании NVIDIA. Тенденция использования CUDA и GPU NVIDIA для обучения нейронных сетей сохраняется по настоящее время. Однако, сегодня разработчику не нужно осваивать непосредственно работу с CUDA - так как есть достаточно большое количество высокоуровневых фреймворков позволяющих компилировать код программного обеспечения как для CUDA (под GPU NVIDIA), так и для центральных процессоров (CPU). Большинство современных высокоуровневых фреймворков обучения глубоких нейронных сетей, в том числе для задач компьютерного зрения разработаны для языка программирования Python [87].

В 2010 и 2011 годах Киерсаном и авторами работ [88, 89] были описаны реализации глубоких полносвязной [88], и сверточной [89] нейронных сетей, обученных на графических ускорителях (GPU) компании NVIDIA с использованием на тот момент, недавно представленного фреймворка разработки CUDA. Разработанные архитектуры нейронных сетей [88, 89] показали рекордные, на тот момент, точности для задач распознавания рукописных цифр (MNIST) и для набора цветных изображений 10 классов фотографий животных и техники (CIFAR-10 [90]). Важно отметить, что в работах [88, 89] нейронные сети обучались с использованием аугментации и без предобучения. Сверточная нейронная сеть основывалась на архитектуре LeNet 5, однако была значительно модифицирована, в частности, в ней слои усредняющей субдискретизации (average pooling) были заменены на слои maxpooling [89].

Результаты Киерсана стали отправной точкой в разработке современных фреймворков глубокого обучения с использованием GPU. Одним из первых успешных фреймворков разработки глубоких нейронных сетей с использованием GPU NVIDIA стало Tehano, представленное для языка программирования Python в 2010 году [91]. Фреймворки типа Tehano значительно ускорили процесс развития глубокого обучения нейронных сетей. Во-первых такие фреймворки позволяют конечному пользователю не задумываться о тонкостях реализации низкоуровневых операций в нейронных сетях (например, свертка, варианты градиентного спуска или метод обратного распространения ошибки) - эти операции уже реализованы в фреймворке и запускаются "из коробки". Во-вторых конечному пользователю не нужно заботиться об особенностях запуска своего кода для конкретных аппаратных конфигураций, как правило фреймворки имеют свои реализации рассчитанные как для любых GPU компаний NVIDIA, так и для произвольных CPU. В современных фреймворках также допускается использование графических ускорителей компании AMD - но в качестве эксперимента. В-третьих, современные фреймворки предоставляют API (Application Programming Interface - интерфейс программного приложения) для языка Python [92]. Данный язык программирования является одним из наиболее простых и высокоуровневых языков программирования, что позволяет освоить разработку нейронных сетей на Python практически любому, начиная со школьной скамьи.

Следует отметить, что набор CIFAR-10, а также его расширенная версия CIFAR-100 являются классическими наборами для тестирования и сравнения нейронных сетей (т.н. бентчмарков - benchmark) наборы разработаны Алексом Крижевски и представлены в работе Крижевски и Хинтона в 2010 году [93].

В 2011 году Киерсан и его коллектив соавторов показали наилучшие результаты в соревновании по распознаванию автомобильных знаков [94] при помощи вышеописанной архитектуры вида [89]. В том числе, авторы [94] показали превосходство своей нейронной сети по сравнению с экспертами, также прежде не видевшими тестовый набор данных. При этом точность работы нейронной сети составила 1,02%, эксперты смогли достичь точности в 1,19% [67].

В 2012 году Киерсаном и его коллективом было показано, что ансамбль нейронных сетей позволяет достичь точности эквивалентной человеку для рукописного набора цифр MNIST (погрешность 0,2%) [95]. Отметим, что понятие **ансамбль нейронных сетей** означает, что используется структура из нескольких параллельно работающих нейронных сетей, результаты работы которых объединяются некоторым алгоритмом принятия конечного решения. Входные данные для такого алгоритма называются метаданными, а сам алгоритм мета-алгоритм. В качестве такого алгоритма может быть простое усреднение, выбор по большинству (голосование) или еще одна нейронная сеть - мета-сеть. Иллюстрация архитектуры ансамбля нейронных сетей приведена на рисунке 3.17. в работе [95] в качестве алгоритма использовалось простое усреднение, а каждая пара нейронных сетей имела свою пред обработку. Отметим, что ансамблевый подход к решению задач требующих прецизионной точности характерен и в настоящее время. Как правило, соревнования, подобные описанным выигрываются именно с использованием ансамблей нейронных сетей.

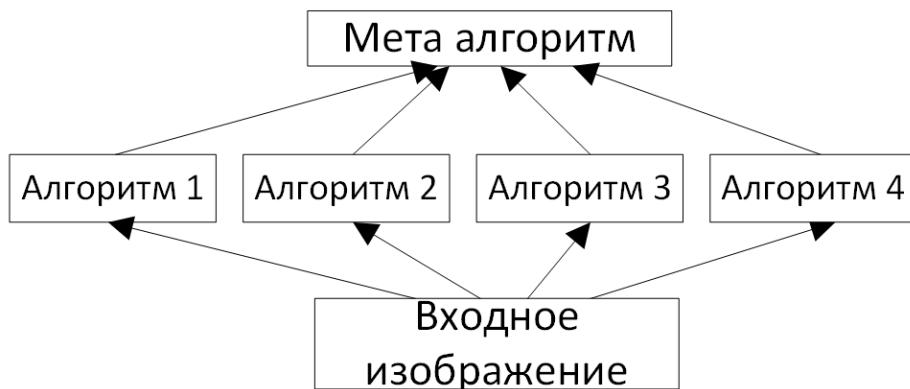


Рисунок 3.17 – Иллюстрация архитектуры ансамбля нейронных сетей

3.9 ImageNet, GPU, Internet - катализаторы прогресса

В начале 21 века (в начале эпохи глубокого обучения) ключевыми тенденциями в глубоком обучении были [72]:

- возможность использования обучения без учителя и самообучения нейронных сетей,

- а также ключевая проблема статистического оценивания – высокое качество обобщения на новые данные после обучения на небольшом количестве примеров (обобщающая способность).

Данные тенденции были смещены на второй план до 2010х годов. В это время в результате сильных прорывов в цифровизации технологий стали позволять проводить обучения НС на достаточно больших объемах данных. По крайне мере проводить предобучение на достаточно больших объемах данных. Этому способствовали как успехи в развитии GPU - вычислительной базы и развитии инструментов работы с GPU, так и успехи в развитии широкополосного доступа к сети интернет - то есть возможности сбор больших выборок данных для обучения. Тенденции 2010х привели к возможности углубления нейронных сетей и простоте работы с ними. Это в свою очередь привлекло внимание как научного, так и инженерного сообществ и привело к гонке экстенсивного развития нейронных сетей.

Одним из наиболее важных катализаторов интереса к развитию нейронных сетей стали соревнования алгоритмов компьютерного зрения ImageNet Large Scale Visual Recognition Challenge (ImageNet LSVRC, ILSVRC), запущенные в 2010 году. Соревнования ILSVRC проводятся регулярно с 2010 года и основаны на тестовой части набора данных **ImageNet** [96, 97]. Набор данных ImageNet включал в 2012 году порядка 10 миллионов изображений в высоком разрешении, размеченных для задач классификации на 10 тысячи категорий. Для соревнований ILSVRC по классификации изображений использовалась тестовая выборки из набора ImageNet, включающие порядка 150000 изображений для 1000 категорий [98, 97]. Отметим, что ILSVRC проводятся по некоторым номинациям, которые от года к году могут меняться [97]. В 2012 году авторам работы [99] удалось достичь точности классификации изображений ILSVRC 84%, тогда как в 2011 году точность была 75% [97]. В 2020 для набора данных ImageNet достигается точность 1.2% [100]. Важно отметить, что описанные выше цифры точности соответствуют т.н. **"top-5 accuracy"** - это метод оценивания при котором результат классификации считается правильным если он соответствует любому из 5 выходов нейронной сети, которые имеют наиболее высокую вероятность. Оценка точности в классическом понимании (ответ с максимальным значением вероятности - правильный) соответствует термину **"top-1 accuracy"**. В 2012 году точность по top-1 была 64%, в 2011 % 51 %, в 2020 91% [100].

Следует отметить, что ImageNet LSVRC являются наиболее престижными соревнованиями между алгоритмами компьютерного зрения по настоящее время. Хотя, набор ImageNet и не является уже наиболее крупным набором изображений [101].

4 Регуляризация обучения нейронных сетей

4.0.1 Классические подходы регуляризации нейронных сетей

В случае глубокого обучения в приложениях компьютерного зрения регуляризация — это достаточно широкая тема. В общем смысле термин можно определить так: Регуляризация - модификация модели, ее метода обучения или ее входных данных таким образом, чтобы повысить точность на тестовой выборке при сохранении почти неизменной точности при обучении. То есть цель регуляризации - повысить обобщающую способность. В классическом смысле термин регуляризация относится к введению ограничений на значения обновлений весовых параметров (например, дропаут, верхний порог градиента, L2, L1). Однако, термин также употребим в отношении таких операций, как

- нормализации слоев (батч-нормализация, нормализация весов и т.д.);
- аугментация данных, расширение выборки;
- методы дропаута;
- организация ансамблей алгоритмов;

К другим методам повышения качества работы архитектур можно отнести:

- оптимизация архитектур (например, остаточные связи и др. приемы).
- регуляризация процесса обучения (Кросс-валидация, адаптивные методы градиентного спуска, learning rate schedule, инициализация весов или предобучение).

В работах 1989-1990 годов Погио и Гироси было показано, что точность работы нейронных сетей может быть повышена регуляризацией функции потерь при их обучении методом обратного распространения ошибки [102, 103]. Типичные техники регуляризации были предложены задолго до работы данных авторов. В том числе, наиболее популярная техника - гребневая регуляризация была предложена советским математиком А.Н. Тихоновым [104, ?]. Данный тип регуляризации в наиболее простом виде часто называется **L2 регуляризация или регуляризация Тихонова**. По существу авторы работ [102, 103] L2 регуляризацию к методу обратного распространения ошибки. По существу, задача обучения с регуляризацией соответствует введению дополнительного ограничения для оптимизируемых параметров (весовых коэффициентов):

$$\begin{cases} L(y, \hat{y}(W, X)) \rightarrow \min \\ \sum_j w_j^2 \leq \text{const} \end{cases} \Rightarrow L(y, \hat{y}(W, X)) + \frac{\lambda}{2} \sum_j w_j^2 \rightarrow \min, \quad (4.11)$$

где $L(y, \hat{y}(W, X))$ - функция потерь нейронной сети; $W = \{w_j\}$ - набор обучаемых весовых коэффициентов нейронной сети; λ функция регуляризации. Использование регуляризации запрещает переобучаться нейронной сети за счет введения т.н. штрафа λ в функцию потерь за слишком большую норму значений весовых параметров ($\|W\|_2^2 = \sum_j w_j^2$).

При использовании такого подхода к регуляризации обновление весовых параметров соля может записано как

$$W_j = W_j - \eta \nabla_w L(j+1) - \lambda_j p_j \|W_j\|_{p_j}^{p_j-1}$$

Например

$$L_2 : W_j - \eta \nabla_w L_{j+1} - \lambda_j 2 \|W_j\|_2^{\lambda_j} = (\lambda'_j)/m;$$

$$L_1 : W_j - \eta \nabla_w L_{j+1} - \lambda_j.$$

Другими словами задача обучения из формулировки "функция потерь должна быть минимальной" превращается в формулировку "должен быть найден баланс между минимум функции потерь и максимум нормы значений весовых параметров". Заметим, что интуитивно понятно (и доказывается математически), что чем меньше значения весовых параметров тем меньше дисперсия (разброс) результатов - то есть выше обобщающая способность нейронной сети. Однако, можно также заметить, что введение регуляризации вводит смещение в функцию потерь, то есть выражение (4.11) ни когда не достигнет нулевых значений кроме как в тривиальном случае (когда все веса = 0, но этот случай нас не интересует). Таким образом, введение регуляризации вносит дополнительную ошибку в результаты обучения нейронной сети. Другими словами - принцип регуляризации - это компромисс между минимизацией дисперсии и смещения ошибки обучения нейронной сети. На практике значения λ выбираются достаточно небольшими (порядка 10^{-5} – 10^{-6}), чего однако, часто хватает для необходимого повышения обобщающей способности без существенной потери точности работы нейронной сети. Отметим, что помимо L2 регуляризации в ряде случаев в нейронных сетях используется L1 регуляризация (LASSO) [105] и комбинация L1 и L2 операций регуляризации [106]. Однако, именно L2 регуляризация является наиболее популярным среди аналогов [107]. В целом сеть может иметь различную регуляризацию для разных слоев, однако это используется редко. Сегодня подход L2 регуляризации часто реализует в виде параметра оптимизатора weight decay [108, 109]. Также отметим, что регуляризация L1 может быть использована для так называемого упрощения (или разряжения) нейронной сети – то есть сведенияя к нулю части его параметров с целью исключения части вычислительных операций.

4.0.2 Аугментация данных

Существенно улучить результаты Лекуна удалось авторам работы [110] 2003 года. Для этого авторами был предложен ряд техник расширения тренировочного набора данных путем искажений оригинальных экземпляров изображений - данная техника в настоящее время называется **Аугментация (augmentation - дополнение)**. Техники аугментации изображений были предложены в работе [111] 1992 года, однако популярность обрели только после работы [110]. Аугментация как правило выполняется путем аффинных (обратимых) искажений входных изображений или добавления к ним шумов или помех (например блики) [112]. Примеры результатов аугментации для изображения рукописной цифры приведены на рисунке 4.18. Отметим, что операцию аугментации следует выполнять с осторожностью.

Мы рекомендуем начинать с обучения нейронной сети без аугментации, а затем постепенно увеличивая долю аугментированных данных следить за ошибкой на тестовых данных. Дело в том, что слишком интенсивной аугментацией можно изменить частоту встречи тех или иных признаков в данных или создать новые признаки. В современных нейронных сетях аугментация проводится в процессе обучения нейронной сети случайным образом, что позволяет частично снизить вероятность появления описанных выше проблем. Отметим, что аугментация может быть рассмотрена как техника регуляризации нейронных сетей.

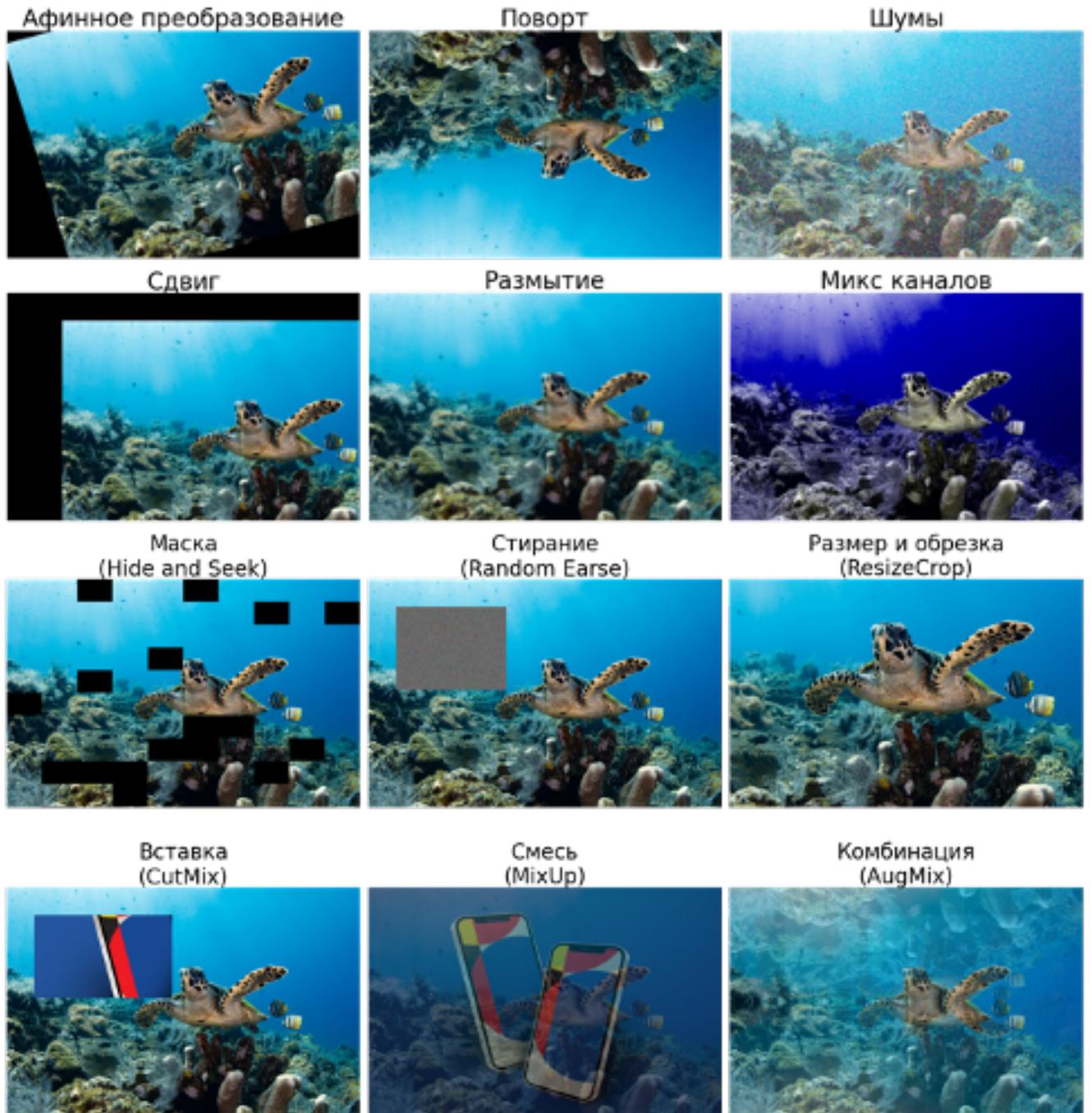


Рисунок 4.18 – Иллюстрация примеров аугментации изображений для расширения входной выборки.

Среди методов аугментации данных можно выделить такие, как использование [?, 113]:

- афинные преобразования - то есть преобразований типа обратимая трансформация изображения, например поворот изображения или его зеркальное отображение;
- создание рамок вокруг изображения;
- добавление к изображению шумов различной природы (гауссовые, спекл-шумы, шумы вида соль и перец, шумы с распределением Пойзона и другие);
- размытие изображений или их обработка другими типами фильтрации;
- обрезка и масштабирование изображений (crop + resize);
- комбинирование изображений в виде сочленения их участков (MOSAIC, CUTMIX)
- комбинирование изображений наложением (MIXUP)
- использование смешанных техник, состоящих из описанных выше (AUGMIX)

В ряде случаев для аугментации также используются следующие подходы:

- генеративные нейронные сети для порождения новых примеров [114]
- адаптивные обучаемые подходы [115, 116]
- в ряде работ комбинацию подходов также предлагается выбирать как выборку случного подмножества простых видов аугментации из расширенного множества и воздействующих с заданной интенсивностью [117].
- при обучении на больших батчах и нескольких GPU возможно использование процедуры т.н. повторяемой аугментации (Repeated Augmentation). При этом подходе для каждой GPU используются одни и те же исходные семплы, но с разными результатами аугментации [118].

В ряде современных источников литературы также предлагается использовать не только комбинирование самих изображений, но и комбинирование их меток. Такой подход как правило требует наличия меток в виде one-hot векторов. Подход носит название кросс-дата аугментация [119]. Подход может быть выражен как

$$\begin{aligned} I'_i &= f(\lambda, I_i, I_j), \\ l'_i &= \lambda l_i + (1 - \lambda) l_j \end{aligned} \tag{4.12}$$

где I'_i , l'_i - это новое изображение и его метка в формате one-hot; f – функция аугментации, I_i, I_j – входные изображения, l_i, l_j – их метки; λ – коэффициент интенсивности. Среди методов комбинирования данных и меток следует выделить работы 2017 и 2019 годов, в которых исследованы методы аугментации MixUp - наложение изображений и сложение меток в соответствующих прозрачности пропорциях [120] и метод CutMix - вставка частей изображений и сложение меток в соответствующих площади пропорциях. Также различными авторами предлагаются модификации данных и других схожих методов [112].

Следует также отметить, что иногда метки могут аугментироваться без изображений, то есть просто путем размытия one-hot вектора на вектор, отображающий некоторое распределение значений. Предполагается, что в этом случае последний слой (например, softmax) будет менее склонен к переобучению [121, 122]. Такое размытие может быть детерминированным [123], стохастическим или получена по результатам работы дополнительной нейронной сети [123].

В наиболее простом случае корс аугментация сводится к сглаживанию меток, то есть замена one-hot кодирования вида:

$$p(y|y_i) = \begin{cases} 1 & \text{if } y = y_i \\ 0 & \text{otherwise} \end{cases} \quad (4.13)$$

на

$$p(y|y_i) = (1 - \varepsilon) p(y|y_i) + \varepsilon u(y|y_i) = \begin{cases} 1 - \varepsilon + \varepsilon u(y|y_i) & \text{if } y = y_i \\ \varepsilon u(y|y_i) & \text{otherwise} \end{cases} \quad (4.14)$$

Другим вариантом сглаживания будет использование т.н. температуры, при этом подходе используется следующая функция активации

$$\text{softmax}_\tau(x_i) = \frac{\exp(x_i\tau)}{\sum \exp(x_j\tau)}, \quad (4.15)$$

где $\tau \in (0, 1]$ – коэффициент температуры.

В сущности аугментация решает проблему недостаточности разнообразных данных. То есть таким образом мы пытаемся повысить точность и обобщающую способность показав сети признак во всем его многообразии. От сюда же следует и основная опасность аугментации данных – аугментация не должна создавать дисбаланс, новые признаки или искажать распределение существующих.

4.0.3 Инициализация Весовых параметров

В 2009 году Хавьери Глори и Джеки Хинтоном было предложено в качестве альтернативы предобучению нейронных сетей проводить инициализацию их весовых параметров следующим способом, называемым **метод инициализации Хавьера (иногда называют инициализация Глори)** [124]:

$$W_i \sim U \left[-\frac{\sqrt{6}}{\sqrt{n_i + n_{i+1}}}, \frac{\sqrt{6}}{\sqrt{n_i + n_{i+1}}} \right] \quad (4.16)$$

где $U[a, b]$ - равномерное распределение со значениями от a до b ; w_i is the weights of i -th layer, n_i , n_{i+1} число входных параметров слоя с номером i (n_i) и число выходных параметров (n_{i+1}).

Инициализация Хавьера основана на схеме инициализации Лекуна (3.8). Однако, авторы [124] утверждали, что в оригинальной идеи Лекуна был проанализирован случай только прямого распространения ошибки. При получении выражения (4.16) было также учтено и обратное распространение ошибки. Авторами [124] утверждалось, что при инициализации весовых параметров вида (4.16) вероятность переобучения достаточно сильно снижается. При этом авторами было обнаружено, что нейронные сети с логистическими функциями активации могут выходить из области насыщения в ходе обучения. Эксперименты авторов показали, что НС, с инициализацией весов (4.16) имеет точность на тестовых данных выше, чем без инициализации, но ниже чем с пред обучением.

В 2015 году Каймингом Хе и соавторами из компании Microsoft был предложен метод инициализации функции ReLU, позволявший обходить результаты, получаемые при помощи инициализации Хавьера Глори [125]. В основе предложенного метода инициализации была положена идея учета несимметричного характера поведения функции ReLU. **Метод инициализации Хе (иногда называют инициализация Кайминга):**

$$W_i \sim w_i \sim N \left[0, \frac{2}{n_i} \right] \quad (4.17)$$

В настоящее время существует несколько вариантов выражений для инициализации Хе и Глори, а также некоторых других, рекомендуемых для различных вариантов функций активации [126]. Использование инициализации весов при отсутствии возможности или необходимости предобучения нейронных сетей позволяет повысить обобщающую способность нейронных сетей, ускоряет процесс обучения и стабилизирует результаты их обучения (при повторениях процедуры).

4.0.4 Регуляризация методом Дропаут(DropOut)

В 2012 году Джеки Хинтоном и соавторами, в том числе Алексом Крижевски, была опубликована работа, в которой авторы указали на одну из возможных интерпретаций переобучения и предложили решение данной проблемы - т.н. **метод дропаутов (dropping out, dropout)**[127]. Авторами замечено, что в переобученной нейронной сети имеет место **проблема содаоптации (co-adaptation** - это состояние обучения нейронной сети, когда нейроны каждого следующего слоя обучаются корректировать результаты работы нейронов предыдущего слоя. В Нормально обученной нейронной сети (не переобученной) предполагается, что каждый следующий слой нейронной сети работает независимо от предыдущего. То есть обучается выделять регулярные признаки из входных для него данных. В переобученной нейронной нейрон перестает выделять регулярности и начинает подстраивать ответ под конкретные экземпляры входных данных, включая все их случайные и регулярные особенности [127]. Другими словами нейронная сеть перестает работать как система с гибким статистическим выводом и превращается в аналог конечного автомата. В работе [127] было предположено, что решениями проблемы со-адаптации могут быть использование нескольких нейронных сетей, каждая из которых обучена на части входных данных или реализация такой структуры внутри одной сети. Последний вариант

предполагает, что в каждой эпохе обучения нейронная сеть не учитывает ряд случайных признаков (выходов нейронов для каждого слоя) в результатах работы слоя - этот метод называется - метод Дропаутов. Число случайно выключаемых признаков как правило задается т.н. вероятностью дропаута p , который является гиперпараметром при обучении нейронной сети. Стандартный дропаут полносвязной сети как

$$y = f(W^T X) \odot m, \text{ где } m \sim Bernoulli(p) = 1 \text{ с вероятностью } p_0 \text{ с вероятностью } 1 - p \text{ этап обучения} = \\ (4.18)$$

То есть метод дропаутов используется только при обучении нейронной сети. При ее работе все нейроны включены, а результат работы каждого нейрона должен быть домножен на $1 - p$.

Другими словами дропаутов используется только при обучении нейронной сети. При ее работе все нейроны включены, а результат работы каждого нейрона должен быть домножен на $1 - p$. Иллюстрация работы метода дропаутов во время двух разных эпох обучения для $p = 0,33$ приведена на рисунке 4.19.

Отметим, что использование метода дропаутов требует увеличения общего числа параметров нейронной сети. Также метод увеличивает дисперсию результатов, поэтому как правило рекомендуется снизить скорость обучения и использовать варианты метода обратного распространения ошибки с моментом. Также метод дропаутов увеличивает общее время обучения нейронной сети в силу описанных особенностей. Следует отметить, что метод дропаутов используется после активационных функций. Несмотря на свои недостатки метод дропаутов в своих различных вариантах реализации остается одним из наиболее популярных методов регуляризации в нейронных сетях [107, 128]. В 2013 году была опубликована работа [129], посвященная математическому обоснованию работоспособности метода дропаутов. В работе [129] было показано, что метод дропаутов увеличивает обобщающую способность нейронной сети (регуляризирует обучение) и выполняет роль нормализации весовых параметров - то есть приводит их распределение к нормальному закону. Также в работе [130] было экспериментально показано, что метод дропаутов может быть использован совместно с L2 и некоторыми другими типами классической регуляризации.



Рисунок 4.19 – Иллюстрация работы метода дропаутов во время двух разных эпох обучения для $p = 0,33$.

Отмечается, что в некоторых случаях вместо использована распределения Бернулли в данном методе могут быть использованы и другие распределения, например, гауссово распределение [128]. Также отмечается, что правильная интерпретация метода дропаута предполагает не «выключение» функции активации как таковую, то приведение ее значение к нулю, а приведение ее значение к результату для значения входа $-\infty$. То есть для функции активации, которая имеет не нулевое значение, например \tanh дропаут должен приводить выставлению значений функций активации в -1 . Такая реализация называется α -дропаут. Эта реализация была предложена в рамках работы по поиску саморегуляризующих сетей, где также была предложена оригинальная функция активации с ненулевым значением для входа $-\infty$ [131].

В сверточных нейронных сетях метод дропаутов не подходит в силу высокой скоррелированности соседних пикселей. По этой причине должны быть использованы специализированные методы дропаута, например канальный дропаут [132], блочный дропаут [133] или дропаут пулинг [134]. Канальный дропаут может быть описан как

$$\begin{aligned} & \vdots \\ y &= \text{channel} \{f(WT \cdot X)\}_{W \times H} \odot m \\ & \vdots \\ y &= \text{channel} \{f(WTX)\}_{W \times H} \odot (1 - p). \end{aligned} \tag{4.19}$$

Иллюстрация методов блочного и поканального дропаута для приведены на рисунках 4.20 а и б. Также на рисунке в приведен пример работы дропаута маскпулинга.

Следует отметить, что к данной методики также относятся такие приемы как стохастическая глубина [135]. Это метод обучения нейронной сети в котором для различных прохождений градиента в обучения используются различные подмножества, полученные из изначальной архитектуры путем включения-выключения отдельных ее блоков. Иллюстрация метода стохастической глубины приведена на рисунке 4.21.

4.0.5 Нормализация

В 2015 году Сергеем Иоффе и Кристианом Седжеди была предложена техника пакетной нормализации - батчнормализации (batch normalization, batchnorm) [136].

Идея метода батчнормализации заключалась в следующем. Как правило обучение нейронной сети проводится методом стохастического пакетного градиентного спуска, то есть не по всей тренировочной выборке а по случайным подвыборкам - мини пакетам. Различия статистических характеристик между такими мини пакетами замедляет обучение нейронной сети и повышает требования к точности подбора гиперпараметров и требования к инициализации весовых параметров - то есть повышает вероятность переобучения и снижает обобщающую способность. Под статистическими характеристиками авторы понимают в первую очередь среднее и дисперсию по каждому признаку для каждой подвыборки (мини-пакета) входных данных. Описанный эффект авторы назвали

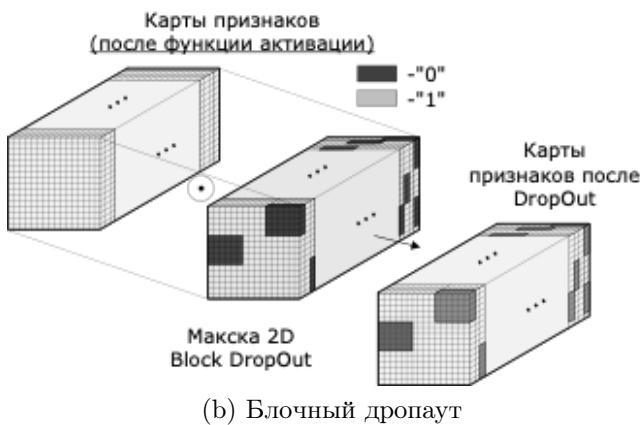
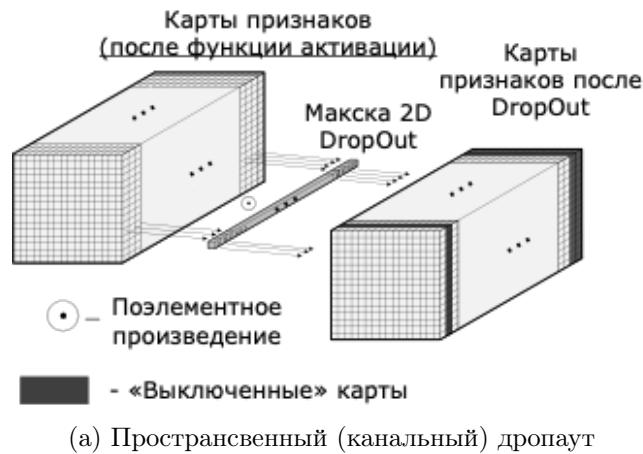


Рисунок 4.20 – Иллюстрации блочного и поканального дропаута, а также операции пулинг-дропаут

внутренний ковариационный сдвиг, internal covariate shift. Для решения проблемы ковариационного сдвига в работе [136] было предложено использование нормализации результатов работы каждого слоя нейронной сети. При этом статистические характеристики должны усредняться по всем мини-пакетам и использоваться при работе обученной нейронной сети [136]. Математически батч-нормализацию можно выразить следующим



Рисунок 4.21 – Иллюстрация работы метода стохастической глубины.

образом:

$$\begin{aligned}
 \hat{y}_i &= f(\tilde{z}_i); \\
 \tilde{z}_i &= \gamma \frac{z_i - \mu}{\sqrt{\epsilon + \sigma^2}} + \beta, \\
 \mu_{\text{TEST}} &= \alpha \mu_{\text{TEST}} + (1 - \alpha) \mu \\
 \sigma_{\text{TEST}} &= \alpha \sigma_{\text{TEST}} + (1 - \alpha) \sigma
 \end{aligned} \tag{4.20}$$

где

- \hat{y}_i - результат работы нейрона;
- $f(\cdot)$ - функция активации;
- \tilde{z}_i - нормализованный линейный выход;
- z_i - линейный выход (например $z_i = W^T X_i$);
- γ, β - коэффициенты масштабирования (обучаются во время тренировки нейронной сети);
- $\mu = \frac{1}{N} \sum_{i=0}^{N-1} z_i$ - среднее по минипакету,
 N - размер минипакета (используется при обучении);
- $\sigma = \frac{1}{N} \sqrt{\sum_{i=0}^{N-1} (z_i^2 - \mu)^2}$ - среднеквадратическое отклонение по минипакету (используется при обучении);
- ϵ - константа для предотвращения деления на 0;
- $\mu_{\text{TEST}}, \sigma_{\text{TEST}}$ - значения среднего и среднеквадратического отклонения, которые будут использоваться вне обучения нейронной сети;
- α - коэффициент сглаживания экспоненциальным средним при обновлении $\mu_{\text{TEST}}, \sigma_{\text{TEST}}$.

Из выражения (4.20) и пояснений к нему могут быть сделаны следующие заключения касательно батчнормализации.

- Операция батчнормализации по разному работает при обучении и в основном режиме работы нейронных сетей.
- При обучении нейронных сетей в выражении (4.20) используются значения μ, σ , вычисленные для текущего минипакета.

- Значения коэффициентов γ и β обучаются во время тренировки и фиксируются в рабочем режиме нейронной сети.
- В рабочем режиме нейронной сети в выражении (4.20) используются значения μ_{TEST} и σ_{TEST} , вычисленные входе обучения.
- Операция батчнормализация должна быть использована перед активационной функцией.
- Операция батчнормализации сводится к тому, что каждый пакет приводится по каждому признаку: сначала к среднему 0 (μ) и дисперсии 1 (σ^2), а затем масштабируется (γ) и смещается (β). По задумке авторов смещение и масштаб должны быть таковыми, чтобы результат нормализации (\tilde{z}_i) всегда оказывался в "правильном" диапазоне значений для активационной функции. Например, для логистической функции желателен диапазон $\tilde{z}_i \sim 0,6 - 0,8$, для функции ReLU $\tilde{z}_i > 0$ и т.д. При этом так как параметры γ и β обучаются, сеть должна сама выбрать нужный диапазон значений \tilde{z}_i .
- Поскольку набор параметров $\mu, \sigma, \gamma, \beta$ одинаков для всех минибачей, следует ожидать одинакового поведения нейронной для каждого из них. Это снижает вероятность изменения ошибки работы нейронной сети в силу того, что установленные гиперпараметры больше подходят для одного минипакета и меньше подходят для дорого. Данное обстоятельство можно считать регуляризацией работы нейронной сети.

По задумке авторов смещение и масштаб должны быть таковыми, чтобы результат нормализации всегда оказывался в "правильном" диапазоне значений для активационной функции. Например, для логистической функции желателен диапазон значений $\sim 0.6 - 0.8$ на входе, для функции ReLU значения > 0 и т.д. При этом так как параметры γ и β обучаются, а сеть должна сама выбрать нужный диапазон значений выхода нормализации. Поскольку набор параметров $\mu, \sigma, \gamma, \beta$ одинаков для всех минибачей, следует ожидать одинакового поведения нейронной для каждого из них. Это снижает вероятность изменения ошибки работы нейронной сети в силу того, что установленные гиперпараметры больше подходят для одного минипакета и меньше подходят для дорого. Данное обстоятельство можно считать регуляризацией работы нейронной сети. Благодаря описанным выше особенностям метод батч нормализации стал одним из наиболее популярных методов регуляризации обучения нейронных сетей. После выхода работы [136] многими авторами были предложены различные вариации данного метода, а также ряд альтернативных вариантов нормализации (например нормализация весовых параметров) [107, 137, 138]. **Недостатками классической версии** (4.20) являются: чувствительность метода к изменению размера пакета; снижение регуляризационных свойств при небольшом размере пакета. Для корректной работы батчнормализации рекомендуется использовать пакеты от 50 – 100 экземпляров данных [139]. Иллюстрация для разных вариантов нормализации приведены на рисунке 4.22

Отметим, что в оригинальной работе [136] были представлены лишь интуитивные пояснения касательно того почему метод батчнормализации работает. Позже рядом авторов были предприняты попытки формального математического обоснования метода. Однако, общего результата нет по настоящее время [140, 141]. Также следует отметить, что батчнормализация на практике редко используется совместно с методом дропаутов. Дискуссия касательно обобщающих свойств обоих методов и рекомендаций к их использованию также остается открытой [142, 139].

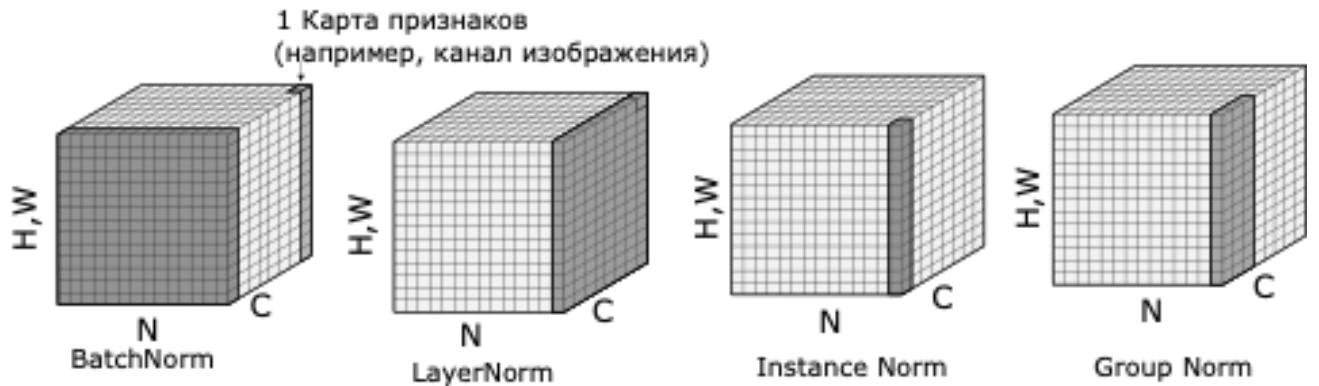


Рисунок 4.22 – Иллюстрация типовых альтернатив батч-нормализации

В настоящее время используются такие подходы к нормализации, как нормализация слоя [143], и по группе изображений [144]. В некоторых случаях, например в случае нормализации слоя выражения вида (4.20) используется как для обучения, так и во время работы нейронной сети. Однако, во втором случае коэффициенты γ, β не обучаются. В ряде источников литературы также предлагается использование нормализации или проведения других операций для слоя весовых параметров. Кроме нормализации весовых параметров, которая может быть выполнена аналогично выражениям (4.20) операциями могут быть: стандартизация весовых параметров; нормализация весовых параметров на максимальное собственное значение; ограничение нормы весовых параметров, в том числе их предельных значений [145]. В отличие от нормализации, стандартизация выполняется как

$$W = \gamma \frac{W - \min(W)}{\max(W) - \min(W)} + \beta. \quad (4.21)$$

5 Развитие глубоких нейронных сетей в 2012-2016 годах

5.1 Глубокие сверточные нейронные сети AlexNet и ZFNet

В 2012 году Алексом Крижевски и коллективом со-авторов был выигран ILSVRC 2010 в категории классификация изображений. [99]. Архитектура предложенной нейронной сети авторами работы [99] принято называть AlexNet (2012 год). Архитектура состоит из 7 скрытых слоев (плюс один выходной), 5 из которых сверточные и два полносвязных скрытых слоя и один полносвязный выходной слой. Иллюстрация архитектуры AlexNet приведена на рисунке 5.23.

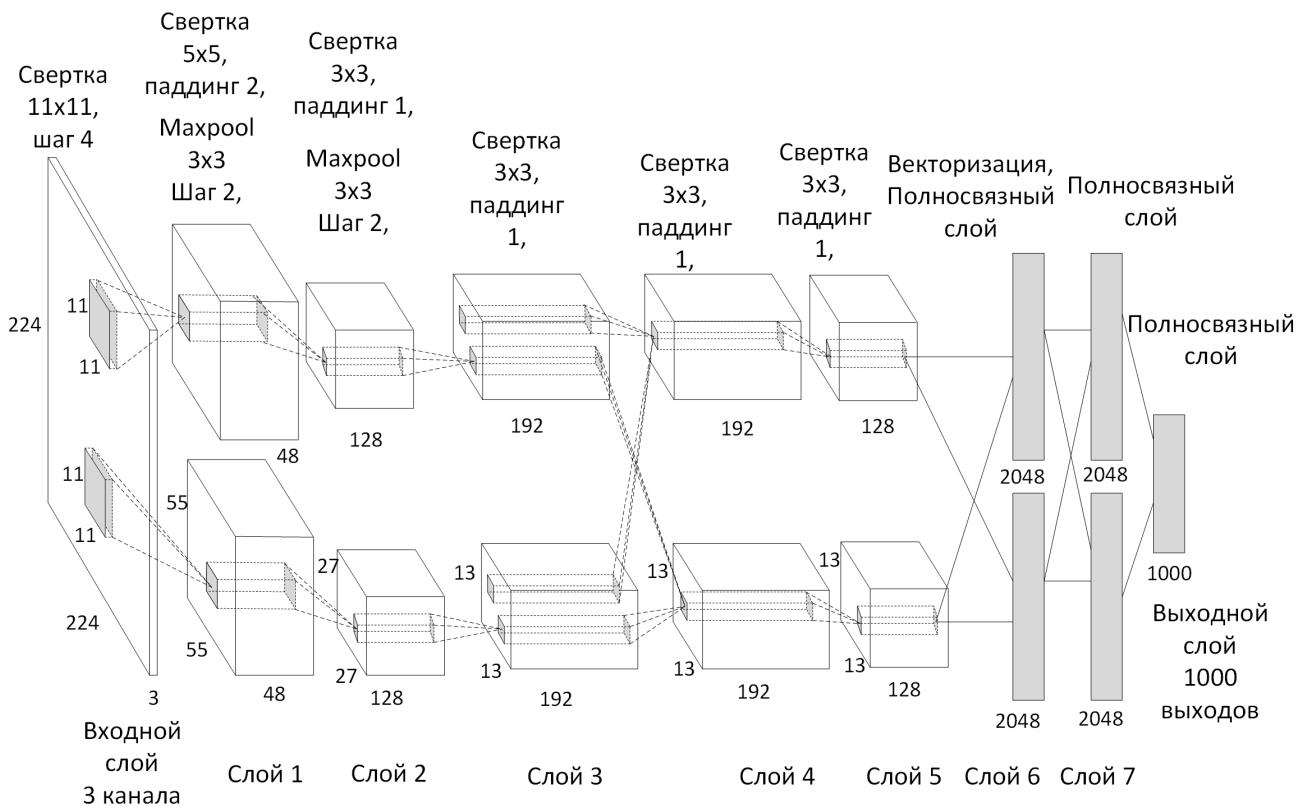


Рисунок 5.23 – Иллюстрация архитектуры глубокой сверточной нейронной сети AlexNet

Особенности архитектуры AlexNet, позволившие перейти к 7 слоям (от 5 в LeNet) были [99]:

- использование функций активации ReLU в скрытых слоях;
- использование аугментации данных, в том числе все изображения сжимались до размеров 256×256 из которых затем случайным образом выбирались патчи размером 224×224 - т.н. случайный кроп (random crop) - один из наиболее популярных видов аугментации по настоящему времени; а также проводилось изменение интенсивности цветовых каналов входных изображений (входные изображения были в формате RGB - красный, зеленый и синий каналы).
- использование обучения стохастическим пакетным градиентным спуском с моментом и L2 регуляризацией;

- регуляризация методом дропаутов с вероятностью 50%;
- обучение на нескольких GPU при помощи параллельной обработкой групп каналов свертки - т.н. групповая свертка, когда все каналы C (см. рис (3.2)) разделяются на группы и затем объединяются.
- использование операции **локальной нормализации отклика фильтров** (local response normalization, LNR для предотвращения взрывного роста весов;
- использование операции субдискретизации с перекрытием.

Большинство из описанных приемов получили развитие и имеют популярность в современных архитектурах.

Архитектура AlexNet стала одним из наиболее значимых прорывов в области развития глубоких нейронных сетей. Архитектура AlexNet была первой одержавшей победу на соревнованиях ImageNet, более того, архитектура показала погрешность почти в 2 раза ниже, чем классические алгоритмы. Начиная с архитектуры AlexNet у научного и инженерного сообщества отпали сомнения в том, каким должен быть подход к решению задач компьютерного зрения. Начиная с 2012 года и по настоящее время подавляющее большинство задач компьютерного зрения решаются только методами глубокого обучения нейронных сетей [146]. Также работа [99] задала направления развития глубокого обучения сверточных нейронных сетей.

В 2013 победу на соревнованиях ILSVRC одержала модифицированная архитектура AlexNet - т.н. **ZFNet** [147]. Авторы работы [147] увеличили долю аугментации данных; провели предобучение на нескольких наборах данных, схожих с ImageNet; а также предложили метод визуализации работы слоев нейронной сети. Благодаря визуализации авторам работы [147] удалось выбрать гиперпараметры архитектуры улучшающие точность работы нейронной сети. **Также метод LNR было предложено заменить на метод локальной нормализации контраста** (local contrast normalization, LCR) [147].

5.2 Осмысление концепции сверточного слоя в 2012-2014 годах

Архитектура VGGnet. До 2014 года концепция сверточного слоя глубокой нейронной сети оставалась почти без изменений с работ Лекуна 1989 года. Хотя за этот период были изменения функция активации, к слою были добавлены слой пулинг, были предложены некоторые методы регуляризации.

В 2014 авторами коллектива Visual Geometry Group (VGG) был предложен метод сокращения числа параметров сверточного слоя - **каскадная свертка**. Идея каскадной свертки, заключается в замене одной свертки со сравнительно большим размером ядра (например 7×7) на последовательное соединение нескольких небольших сверточных ядер, описывающих тот же размер рецептивного поля (например 3 свертки 3×3) [148]. Таким образом может быть задано общее выражение для замены свертки с размером ядра $K \times K$ на набор ядер, например размером 3×3 (наиболее популярный случай) $(K - 3)/2 + 1$. При этом число параметров с $K \cdot K$ снижается до $((K - 3)/2 + 1) \cdot 9$. Например, $5 \times 5 = 2 \cdot 3 \times 3$;

$7 \times 7 = 3 \cdot 3 \times 3$ $11 \times 11 = 5 \cdot 3 \times 3$ Общее выражение для расчета числа сверток может быть задано как

$$(K - k)/2 + 1 \quad (5.22)$$

Общее выражение для расчета числа параметров квадратных сверток без учета смещения может быть задано как

$$\frac{P_K}{P_k} = ((K - 3)/2 + 1) \cdot k^2 \quad (5.23)$$

где K - размер общего ядра; k - размер уменьшенного ядра; P_K, P_k числа параметров квадратных сверток без учета смещения в соответствующих случаях. Иллюстрация работы каскадной свертки показана на рисунке 5.24.

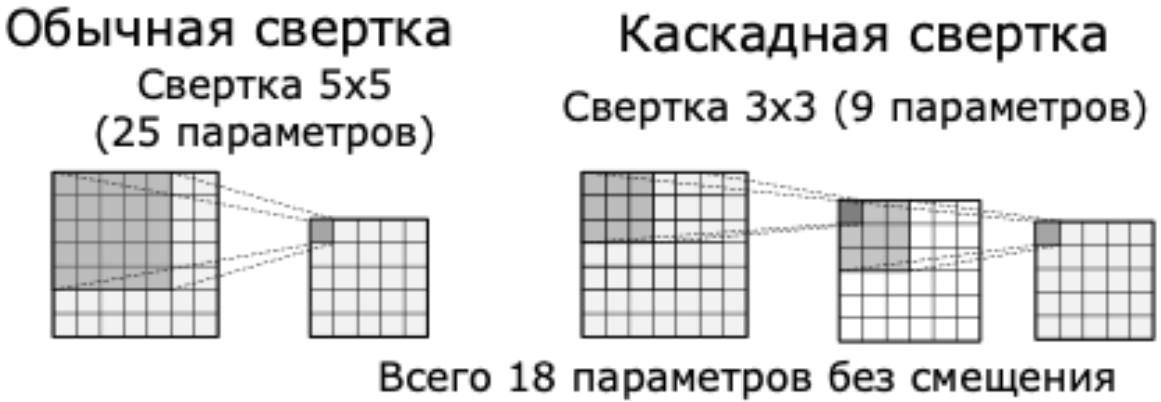


Рисунок 5.24 – Иллюстрация архитектуры глубокой сверточной нейронной сети VGGNet.

Предложенный прием каскадная свертка позволил авторами работы [148] предложить ряд значительно более глубоких архитектур сверточных нейронных сетей, имеющих от 11 до 19 слоев (VGG-11, VGG-13, VGG-16, VGG-19). Архитектуры VGG-19 и VGG-16 показали точности 92% и 91,4% соответственно по "top-5 accuracy" на ILSVRC-14. Архитектура VGG-16 используется по настоящее время в силу простоты. Однако по соотношению числа параметров к точности она является одной из наименее эффективных среди современных архитектур [100]. Иллюстрация архитектуры VGG-16 приведена на рисунке 5.25.

Архитектура Сеть в Сети. В 2013 M. Lin и соавторы опубликовали работу [149], в которой работе были предложены следующие приемы оптимизации сверточных нейронных сетей, ставшие впоследствии классическими.

- **Концепция расширения понятия сверточный слой до понятия "сеть в сети" (Network In Network, NIN).** Концепция сеть в сети - это архитектура слоя, представленного как последовательное соединение подслоя свертки и нескольких полно связанных подслоев. Цель полносвязного подслоя - изменение числа каналов на выходе слоя за счет полно связанных подслоев. А также увеличение числа нелинейностей (активационных функций) в каждом слое - соответственно увеличении возможностей

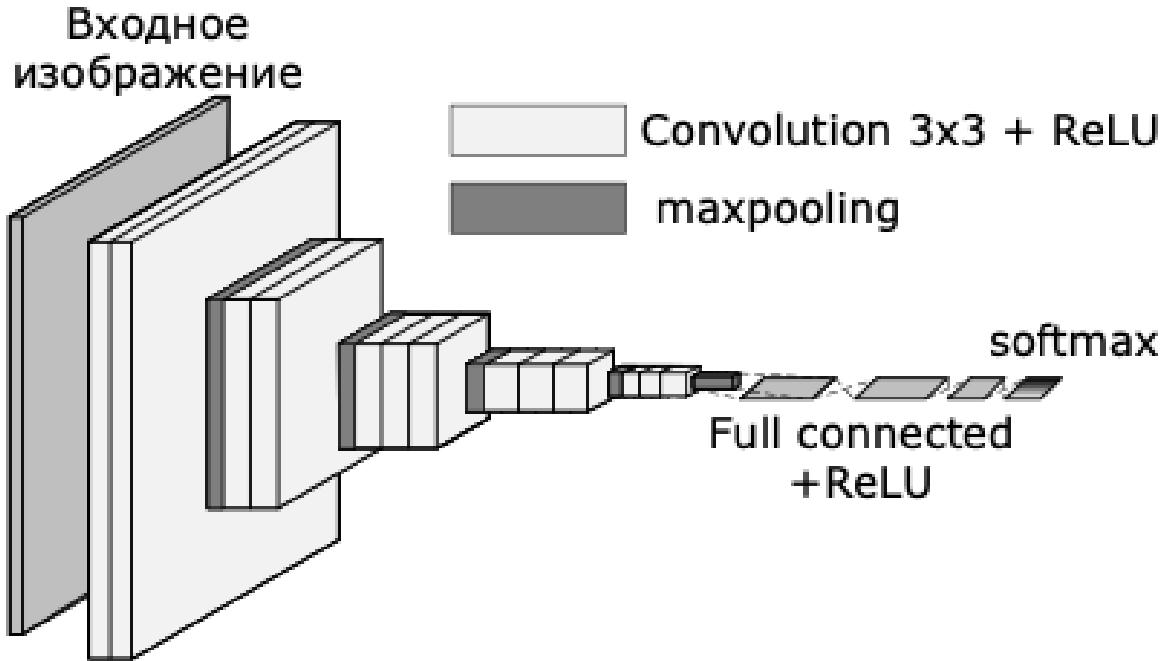


Рисунок 5.25 – Иллюстрация архитектуры глубокой сверточной нейронной сети VGGNet.

для выделения нелинейных признаков каждым слоем нейронной сети. Кроме того предложенная архитектура слоя позволяла выделить меж-канальные зависимости (признаки) внутри слоя. Сами авторы работы предлагали интерпретацию своей концепции как микро-сеть, скользящая по входному изображению. Иллюстрация работы блока NIN с полносвязной и точечной свертками, а также блока NIN как такового показаны на рисунках 5.26 а-в.

- **Использование глобального усредняющего пулинга (global average pooling, GAP)** вместо слоев векторизации набора полносвязных слоев в головной части нейронной сети. Основная идея глобального пулинга заключается в устраниении набора полносвязных слоев, необходимых в случае векторизации, соответственно число параметров сети значительно снижается без существенных потерь в точности. Это делает обучение нейронных сетей более простым и снижает вероятность переобучения. На данный момент существует несколько реализаций слоя глобального пулинга, однако усредняющий вариант остается наиболее популярным. В некоторых операциях также популярен и глобальный максимальный пулинг, но в общем он не используется. Иллюстрация работы глобального пулинга и ее сравнение с подходом векторизации слоя приведены на рисунке 5.27.
- **Использование точечной свертки ($1 \times 1 \times C$ - pointwise convolution) для выделения межканальных признаков.** Точечная свертка одна из наиболее часто используемых операций. Она позволяет сжимать или увеличивать число каналов в каждом сверточном слое. Это позволяет варьировать число каналов и число выделяемых ими признаков. Данная концепция была предложена как замена полносвязным подслоям с целью уменьшения числа параметров. Иллюстрация работы

точечной свертки и современной реализации блока типа узкое горлышко на его основе приведены на рисунке 5.28.

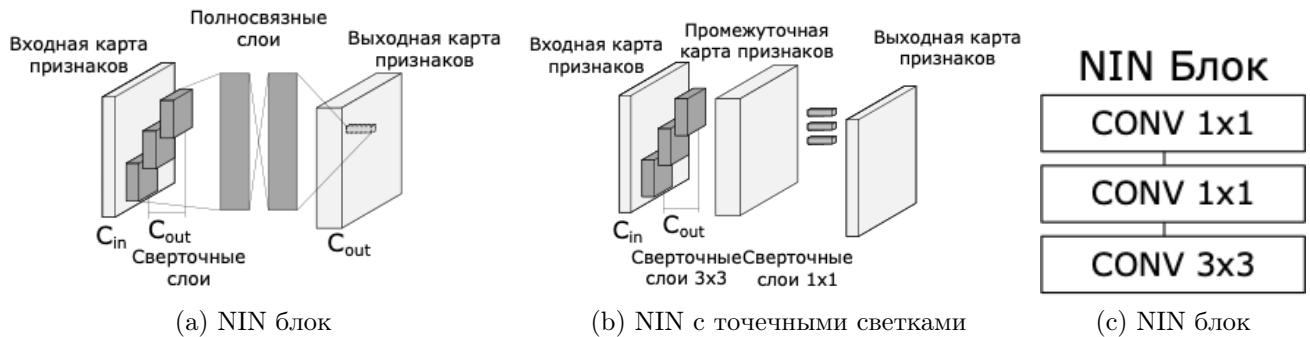


Рисунок 5.26 – Иллюстрация блока Network in Network.

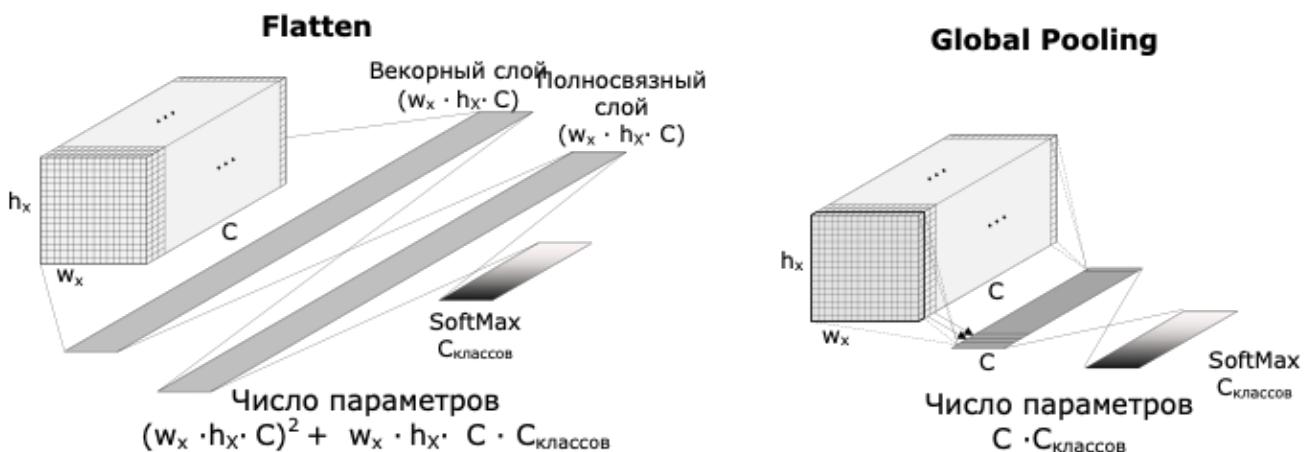
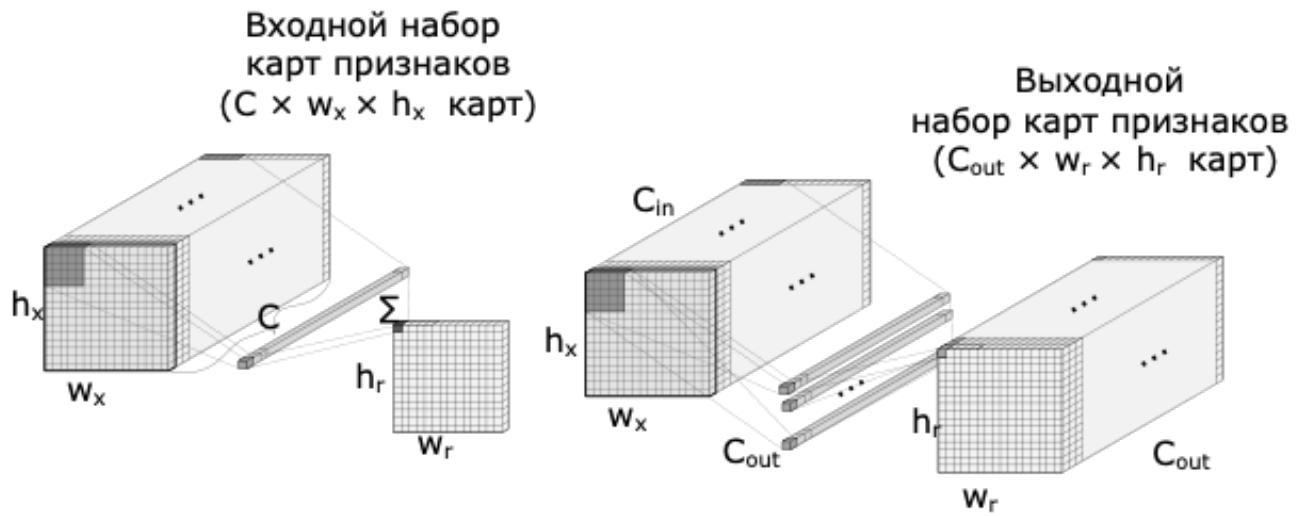


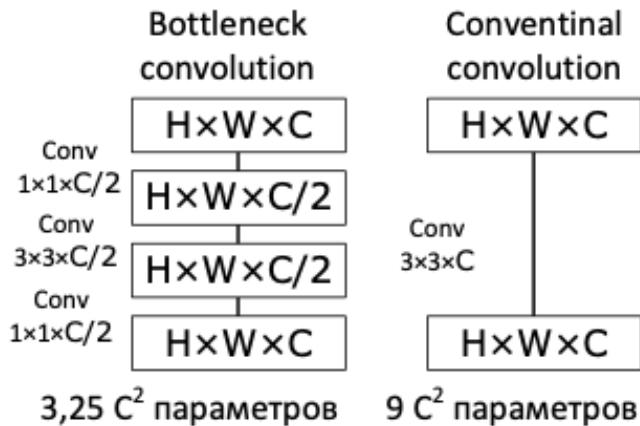
Рисунок 5.27 – Иллюстрация сравнения работы глобального пулиинга (GAP) и слоя векторизации flatten

Концепция сеть в сети по существу стала основой перехода от абстракции "слой" в рассуждениях об архитектурах сверточных сетей к абстракции "блок" - то есть набор слоев, рассматриваемых как единица проектирования новых архитектур. Сегодня каждый блок нейронной сети может иметь разное число путей прохождения градиента внутри, свои методы регуляризации, и другие особенности. С точки зрения стыковки блоков остается важным только число и размер карт признаков на входе и выходе. Структура блока может быть подвергнута любым изменениям без оказания влияния на остальные блоки нейронной сети.

Архитектуры семейства Inception. Сам по себе подход Network In Network, предложенный авторами работы [149] не получил широкой популярности. Однако идея использования "сети в сети" легла в основу архитектуры **GoogleInception V1 (GoogLeNet V1)**, предложенной компанией Google (коллектив авторов во главе с Кристианом Седжеди) в 2014 году [121].



(a) Иллюстрация работы слоя точечной свертки



(b) Иллюстрация блока типа узкое горло с точечной сверткой

Рисунок 5.28 – Иллюстрация работы точечной свертки

Авторы работы [121](GoogLeNet V1) переработали концепцию NIN, расширили каждый слой до нескольких параллельных сверточных подслоев и использовали для каждой свертки дополнительно свертку 1×1 для регулирования общего числа параметров сети. Цель такого подхода по мнению авторов решить проблему, сформулированную т.н. принципом Хебба "связанные нейроны активируются вместе" - то есть способствуют выделению одного и того же признака. Таким образом, использование нескольких параллельных подслоев должно повысить вероятность выделения полезного и регулярного признака для каждого слоя нейронной сети. Также авторы предложили использовать подслои с разным размером ядра свертки - то есть с разным размером эффективного рецептивного поля, а также слои пулинга. Иллюстрация типичной архитектуры слоя сети GoogLeNet V1 приведена на рисунке 5.29. Отметим, что в работе [121] были также предложены использование промежуточных слоев принятия решений и метод аугментации для предобучения нейронных сетей наложением разных экземпляров данных и их меток. Архитектура GoogLeNet V1 включала 22 слоя и показала точность 93,3 % на ILSVRC 2014. В последующие несколько

лет Кристианом Седжеди с соавторами были предложены несколько модифицированных вариантов архитектуры Inception [122, 150].

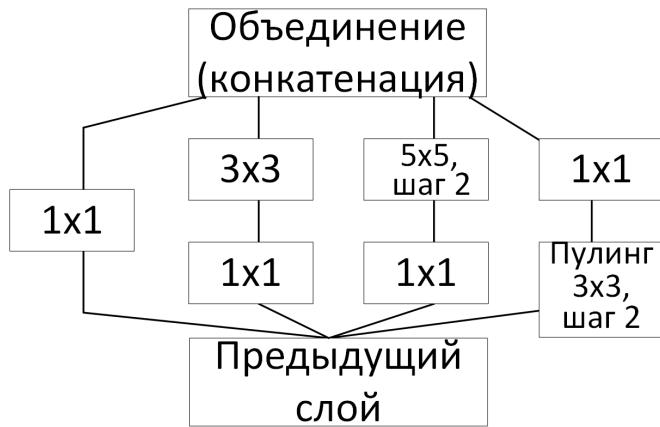


Рисунок 5.29 – Иллюстрация типичной архитектуры слоя сети GoogLeNet V1

Помимо блока в архитектуре было введено еще несколько приемов улучшения точности, в частности промежуточные классификаторы. Результаты их работы складывались с итоговым в заданных пропорциях. Однако, этот прием не получил дальнейшего распространения. Иллюстрация архитектуры Inception с дополнительными выходами показана на рисунке 5.30. Отметим, что позже дополнительные выходы были заменены на другие методы реугляризации, например батч-нормализацию.

Также авторами архитектуры было продолжено улучшения в архитектурах Inception V2, V3 [122]. Также были предложены еще несколько версий, в том числе 4 и Inception-ResNet V1, V2 [150]. Однако их значимость весьма ограничена. Основной особенностью версии 3 стало использование факторизации свертки. То есть одна квадратная свертка разделась на насколько прямоугольных. Каждая прямоугольная свертка может выделять свои признаки. Вертикальные свертки могут выделить горизонтальные линии, а горизонтальные свертки, наоборот. Предполагается, что обе эти свертки будут использованы последовательно, такой блок называется пространственно-разделенная свертка. Благодаря этому приему удается сократить число карт признаков в 1,5 раза при увеличении числа функций активации (фактически операция умножения \cdot заменяется тут на сумму $+$). Иллюстрация пространственно-разделенной свертки показана на рисунках 5.31. Иллюстрация блока Inception V3 показана на рисунке 5.32. Также отметим, что в архитектуре Inception V3 была использована техника сглаженных меток, о которой речь уже шла выше см. (4.14).

Отметим, что расширение концепции сверточного слоя не ограничивается отмеченными выше работами. Скорее они являются лишь началом на этом пути [151]. В последующих разделах будет показано, как к данной концепции будут добавлены пакетная нормализация, тождественные связи, механизм внимания и другие приемы, позволяющие значительно повысить обобщающую способность и точность нейронных сетей.

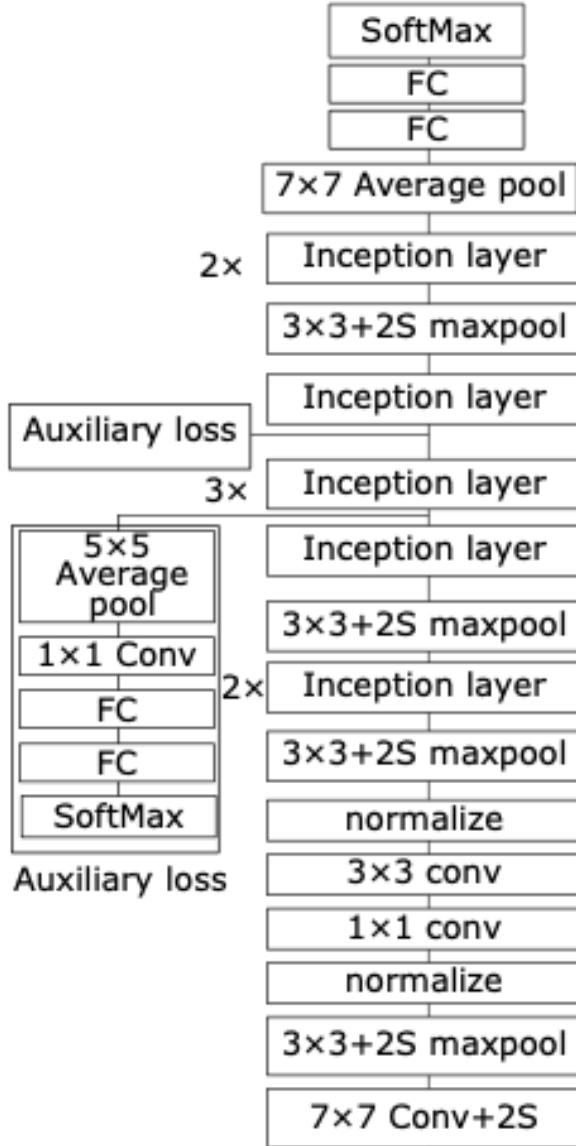
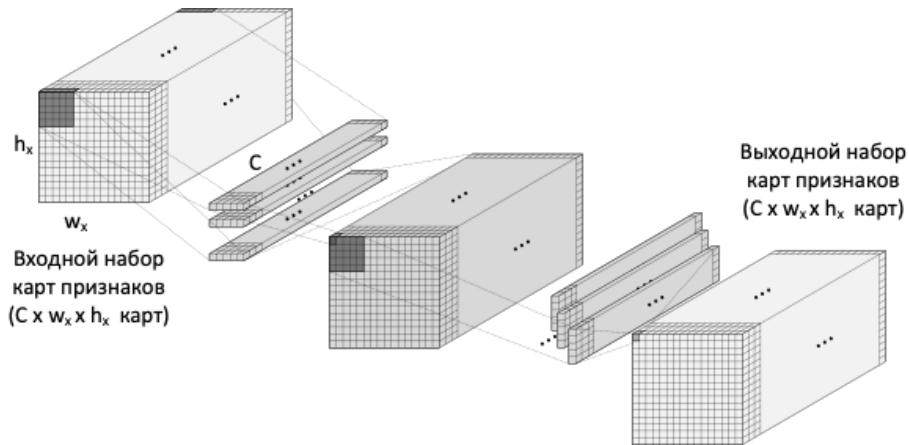


Рисунок 5.30 – Иллюстрация типичной архитектуры сети GoogLeNet V1

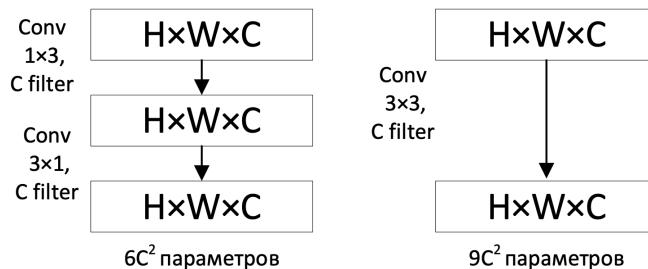
5.3 Идея остаточного обучения (Residual layer)

5.3.1 Слой остаточного обучения (Residual layer)

В 2015 году Каймингом Хе и соавторами (компания Microsoft) было предложено использовать **принцип идентичных связей** для обучения нейронных сетей увеличенной глубины. По задумке автора идентичные связи должны быть параллельны основному слою нейронной сети [32]. Слой (или бок слоев) нейронной сети с идентичными связями принято называть **skip-connection**, **identity-connection**, **residual-connection** или **residual-layer**, **остаточный слой**, **тождественный слой**. Иллюстрации остаточного блока нейронной сети без остаточного слоя, аналогичного блока с остаточным слоем [32] и блока с модифицированным остаточным слоем [152] (2016 г., Каймингом Хе и соавторы) приведены на рисунках 5.33 А), Б) и В) соответственно.



(a) Иллюстрация пространственно-разделенной свертки



Число параметров без смещения!

(b) Блок пространственно-разделенной свертки

Рисунок 5.31 – Иллюстрация пространственно-разделенной свертки.

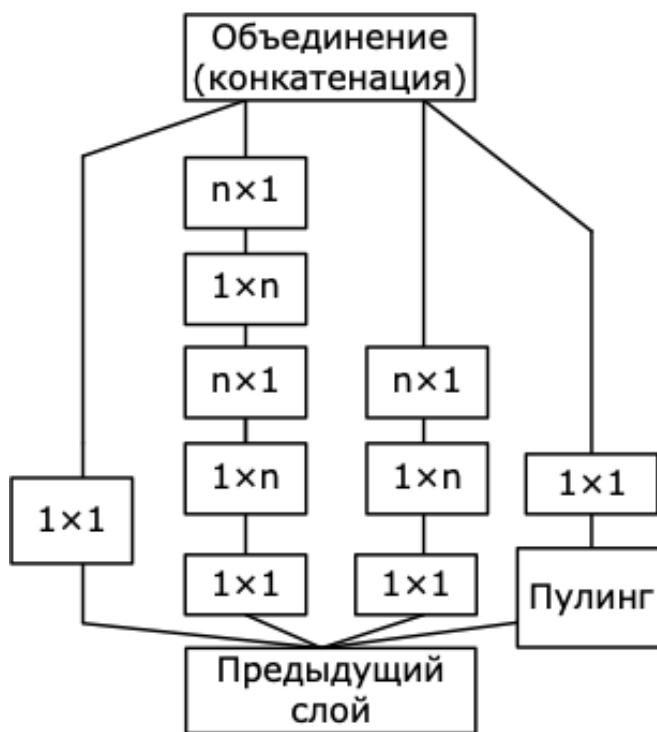


Рисунок 5.32 – Иллюстрация блока Inception V3.

Среди вариантов на рисунках 7 вариант с узким горлом (bottleneck block или bottleneck layer (Δ) – это наиболее популярный вариант остаточного слоя. Такой блок экономит общее число параметров за счет сужения числа карт признаков перед основной сверткой. Блок работает следующим образом. После основной свертки происходит расширение числа карт признаков. Таким образом остаточная связь работает как умная (обучаемая) обратная

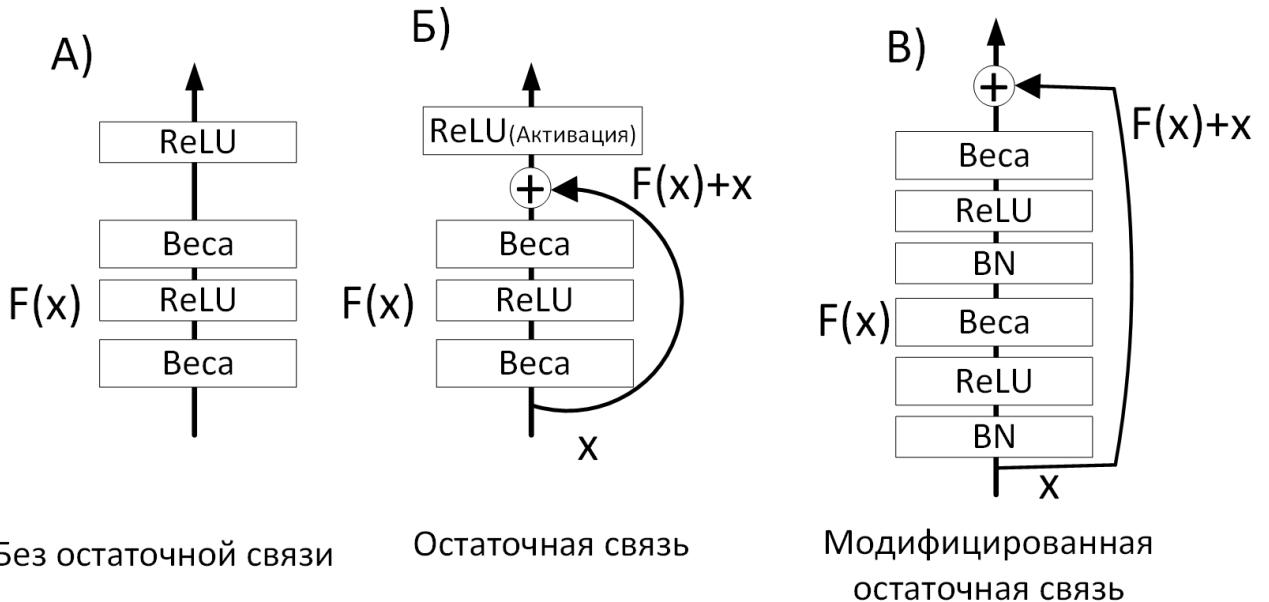


Рисунок 5.33 – Иллюстрация слоя с остаточной связью: А) блок слоев без остаточной связи; Б) блок с остаточной связью; В) блок с модифицированной остаточной связью

отрицательная связь – то есть позволяет регулировать выход. Если в основной части будут произойдет проблема типа взрыв или вымывание градиента, то остаточная связь компенсирует это. Если слой не нужен сеть просто сведет его к тождественной связи (справа). Если слой нужен частично –то результат будет комбинация остаточной связи и основного блока. Если слой нужен полностью, то результат будет таким, что остаточная связь окажется не нужной.

Благодаря использованию остаточной связи (см. рис 5.33 Б) авторами работы [32] удалось увеличить глубину нейронных сетей до 152 слоев. Архитектуры сверточных нейронных сетей с остаточными связями, разработанных авторами [32] принято называть ResNet-18, ResNet-34, ResNet-50, ResNet-101, ResNet-152, где цифра обозначает число слоев. В соревнованиях ILSVRC 2015 модель ResNet-152 достигла погрешности 4,6% для одной сети и 3,6% для ансамбля сетей [32]. При этом значения ошибки ниже, чем для эксперта (у человека ошибка для тех же условий порядка 5 % [153]).

В основе работы остаточных связей лежат следующие идеи. Если во время тренировки нейронной сети в основном слое или блоке слоев возникает вымывание или взрыв градиента, то остаточная связь действует как регуляризация, компенсируя проблему. Компенсация происходит за счет суммирования информации (входных данных) перед блоком и результата работы самого блока. В случае, если блок не нужен нейронной сети (например, слишком глубокий слой), то по задумке авторов сеть должна обучиться так, чтобы влияние остаточной связи преобладало в блоке. Таким образом остаточная связь является приемом регуляризации обучения нейронных сетей. Благодаря остаточным связям стало возможным увеличивать глубину нейронных сетей практически до бесконечности.

В 2016 году Каймингом Хе и соавторами была опубликована работа [152] в которой авторы смогли увеличить глубину сети ResNet с 152 до 1001 слоя, благодаря разработке модифицированной версии блока с остаточными связями (рис. 5.33 В). При этом оказалось, что точность на наборе данных CIFAR-10 увеличилась лишь с 5,46 % до 4,62 % [152]. Данный результат можно объяснить тем, что в результате обучения сети большинство слоев выше 100-200 полностью или почти полностью представляют собой остаточную связь - то есть не нужны. Также отметим, что в 2016 году в работе [135] была опубликована архитектура рекордно глубокой сети, имеющей 1202 слоя. Сеть была обучена используя прием дропаута слоев (т.н. стохастическая глубина), что, позволило несколько увеличить точность но оставило вопрос об эффективности такой архитектуры (в отношении точность/время/вычислительные ресурсы) открытым.

Эксперименты проведенные Каймингом Хе и соавторами в работах [32, 152] показали сообществу исследователей нейронных сетей, что увеличение возможностей нейронных сетей за счет их углубления не является бесконечным и заставило задуматься о других путях совершенствования нейронных сетей. В 2016 году сообщество исследователей нейронных сетей начало работать на увеличением ширины слоя нейронных сетей при фиксированном числе слоев, а затем перешло к оптимизации каждого слоя. Так, в соревнованиях ILSVRC 2016 второе место с отрывом в 0,04% одержала архитектура ResNeXt - вариант архитектуры ResNet с оптимизированной структурой блока [154]. При этом первое место занял ансамбль из набора известных архитектур, не внесший существенного вклада в развитие методов глубокого обучения[155].

Среди достоинств блоков с остаточными связями можно привести следующие.

- Тождественный слой существенно снижает вероятность вымывания градиента. Можно обучать более глубокие сети. Понижаются требования к выбору параметров обучения. Во время процедуры обратного распространения ошибки тождественные связи работают как отрицательная обратная связь, позволяя обойти слишком резкие изменения градиента в слое.
- Остаточная связь позволяет сохранить входные признаки после работы блока, если в нем произошло переобучение.
- Использование остаточных связей – увеличение путей обработки признаков. Чем больше путей обработки, тем больше нелинейных признаков может быть выделено.
- Остаточная связь (как путь) не меняет рецептивного поля. Комбинация тождественного и основного результатов работы блока позволяет варьировать рецептивное поле. оптимизация

5.3.2 Архитектуры на основе ResNet

Архитектура Wide ResNet Одной из первых удачных попыток оптимизации ResNet стала архитектура Wide ResNet, которая была предложена в 2016 году [Wide ResNet]. Основная идея этого подхода была в том, чтобы сделать сеть менее глубокой, но с большей шириной слоя. Авторами [Wide ResNet] было замечено, что широкие слои обучаются быстрее и легче чем глубокие. Таким образом было предложено сделать широкую сеть сохранив число параметров относительно глубокой. Так как сеть широкая в ней имелась возможность ввести дропаут. В классическом подходе ResNet использовалась только батч-нормализация. Авторами были предложены варианты с числом слоев от 16 до 40.

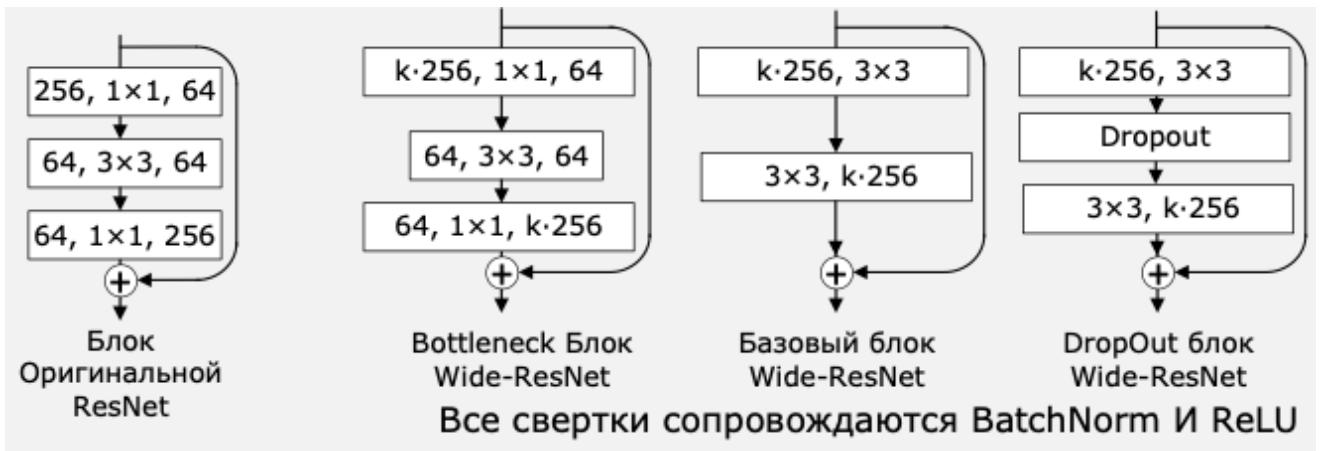


Рисунок 5.34 – Иллюстрация вариантов слоя WideResNet

Архитектура ResNet 38 В работе [156] авторы отметили еще одну особенность блоков ResNet. Технически можно описать архитектуру ResNet как некоторый ансамбль блоков. Однако комбинация из двух последующих блоков не эквивалентна ансамблю в полном смысле этого слова. В архитектуре DenseNet эта особенность не учтена. Другими словами

$$y_2 = y_1 + f_2(y_1, w_2) = y_0 + f_1(y_0, w_1) + f_2(y_0 + f_1(y_0, w_1), w_2) \neq y_0 + f_1(y_0, w_1) + f_2(y_0, w_2) + f_2(f_1(y_0, w_1), w_2). \quad (5.24)$$

Для компенсации этого эффекта авторы предложили свою архитектуру. Иллюстрация архитектуры и пояснения к ней приведены на рисунке 5.35. Таким образом данная архитектура реализует выражение $y_0 + f_1(y_0, w_1) + f_2(y_0, w_2) + f_2(f_1(y_0, w_1), w_2)$. Данное выражение по существу реализует принцип полной факторизации двух блоков ResNet. По задумке авторов, при стыковках таких блоков в архитектуру образуется экспоненциальное увеличение числа виртуальных простых архитектур.

Для построения архитектуры авторы также использовали подход и наработки предложенные в рамках архитектуры WideResNet. Также была предложенная специальная версия данной архитектуры для решения задач семантической сегментации.

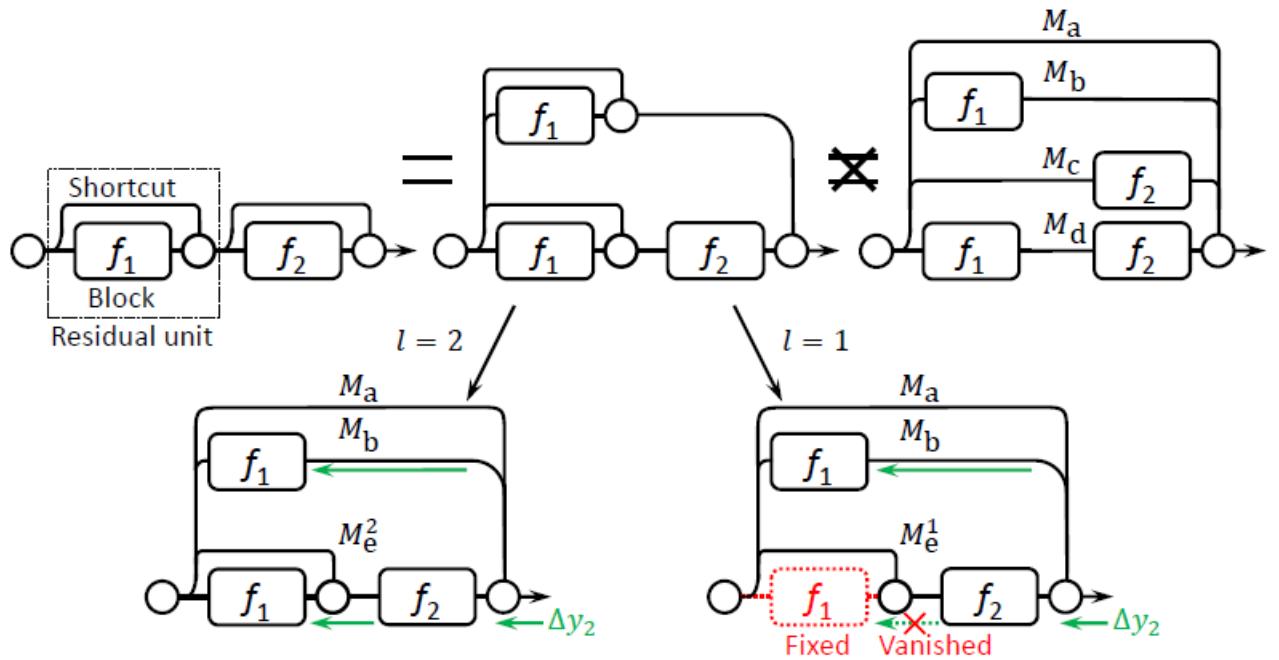


Рисунок 5.35 – Иллюстрация архитектуры ResNet38 и соответствующих ей пояснений

Архитектура ResNeXt Другим удачным подходом, предложенным в 2016 году, стала архитектура ResNeXt. Или расширенный ResNet [ResNeXt]. Основная идея этого подхода заключается в разделение одного пути прохождения градиента внутри остаточного блока на несколько идентичных. При этом предполагается, что каждый такой проход будет давать разные признаки за счет разной перегруппировки карт признаков точечными свертками. То есть сети оказывается помочь в вопросе выделения большего числа признаков в ходе обучения – аналогично блоку inception. Архитектура ResNeXt стала де-факто победителем соревнований imagenet 2016 года.

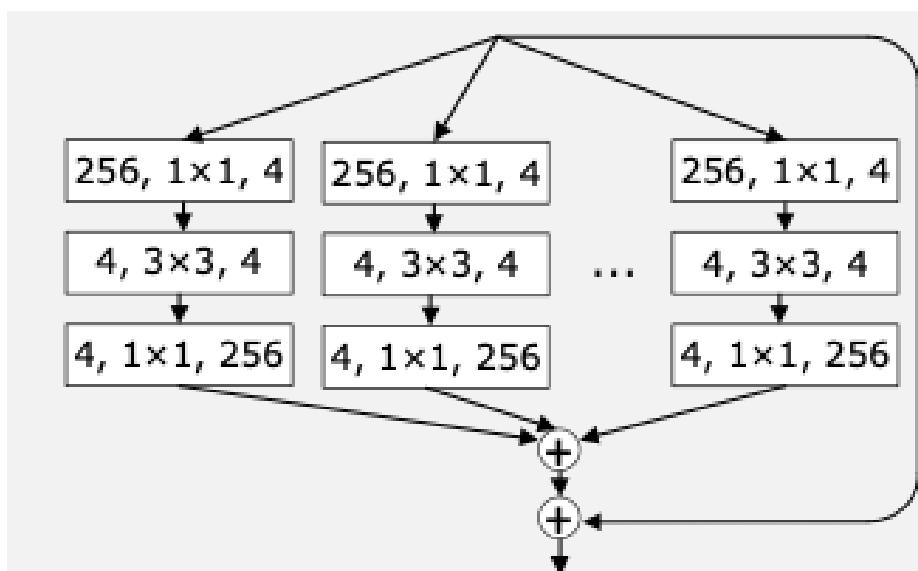


Рисунок 5.36 – Иллюстрация слоя ResNeXt

Блок Xception Модернизацией архитектуры ResNeXt стала архитектура Xception [157]. Идея авторов сводится к замене нескольких путей прохождения градиента в блоке ResNeXt к тому, чтобы сделать каждую карту признаков отдельным элементом сврточного блока. Фактически это свелось к варианту групповой свертки с числом групп, равным числу блоков. Такой свртке предшествует точечная свртка. Цель точечной свртки выставить нужное число карт признаков и перегруппировать их информацию.

Свртку в основе архитектуры Xception было предложено назвать глубокой сврткой или DepthWise Convolution. В классической свртке каждое ядро было трехмерным тензором, где третья размерность соответствовала числу входных карт признаков. В глубокой свртке каждое ядро фактически двухмерное и действует только для каждой отдельной карты признаков. В более продвинутых вариантах можно ввести, например, по два ядра на каждую карту признаков и тем самым увеличить число выходов свртки по отношению к числу ее входов. Если к глубокой свртке добавить точечную свртку – то такую архитектуру называют глубокой разделенной сврткой - Depth Wise Separable Convolution (DWConv). Описанная операция свртки позволяет существенно сэкономить число параметров по сравнению с классическим подходом – это стала одним из главных преимуществ архитекторы Xception и всех последующих архитектур, использующих этот подход. Также такая свртка увеличивает число нелинейных связей по сравнению с классическим подходом. Иллюстрации работы блоков ResNeXt и Xception приведены на рисунках 5.37

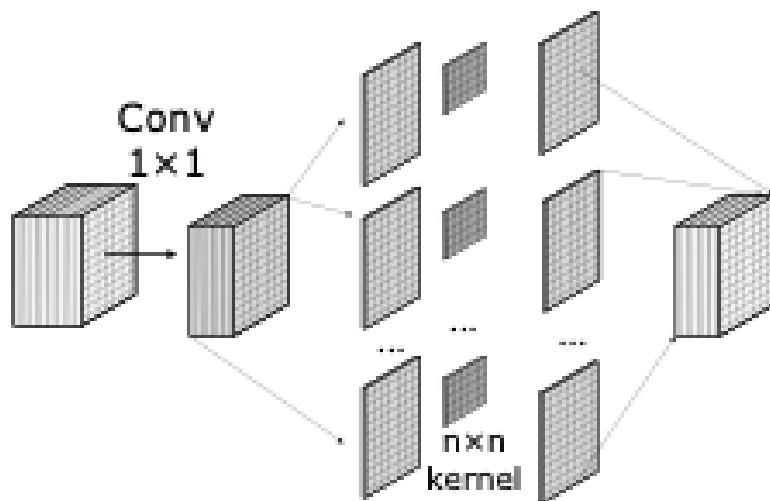


Рисунок 5.37 – Иллюстрация слоя Xception

Операция depthwise separable convolution В настоящее время depthwise separable convolution приобрела особую популярность. Однако, в более современных подходах операции точечной и глубокой свртки расположены наоборот [158]. Также в некоторых источниках литературы предлагается использовать пространственно-разделенную глубокую свртку deepwise spatial separable conv. Подход позволяет сократить число параметров еще в 1.5 раза . Ниже приведем пример соответствующего расчета для свртки с размером ядра 3×3

с учетом параметра смещения.

$$\frac{P_{layer_{con}}}{P_{layer_{DW}}} = \frac{C_{out} \times P_{kernel_{con}}}{C_{in} \times P_{kernel_{DW}} + C_{out} \times P_{kernel_{point}}} \frac{32 \times (16 \times 3 \times 3 + 1)}{16 \times (3 \times 3 + 1) + 32 \times 17} = \frac{4640}{704} = 6.5 \quad (5.25)$$

где $P_{layer_{con}}, P_{layer_{DW}}$ - число параметров обычного и глубокого разделённого слоев соответственно; $P_{kernel_{con}}, P_{kernel_{DW}}, P_{kernel_{point}}$ - число параметров ядра свертки для обычной светки; глубокой части свертки и точечной свертки; C_{out}, C_{in} - число входных и выходных карт признаков. Иллюстрация работы глубокой разделенной свертки приведена на рисунке 5.38.

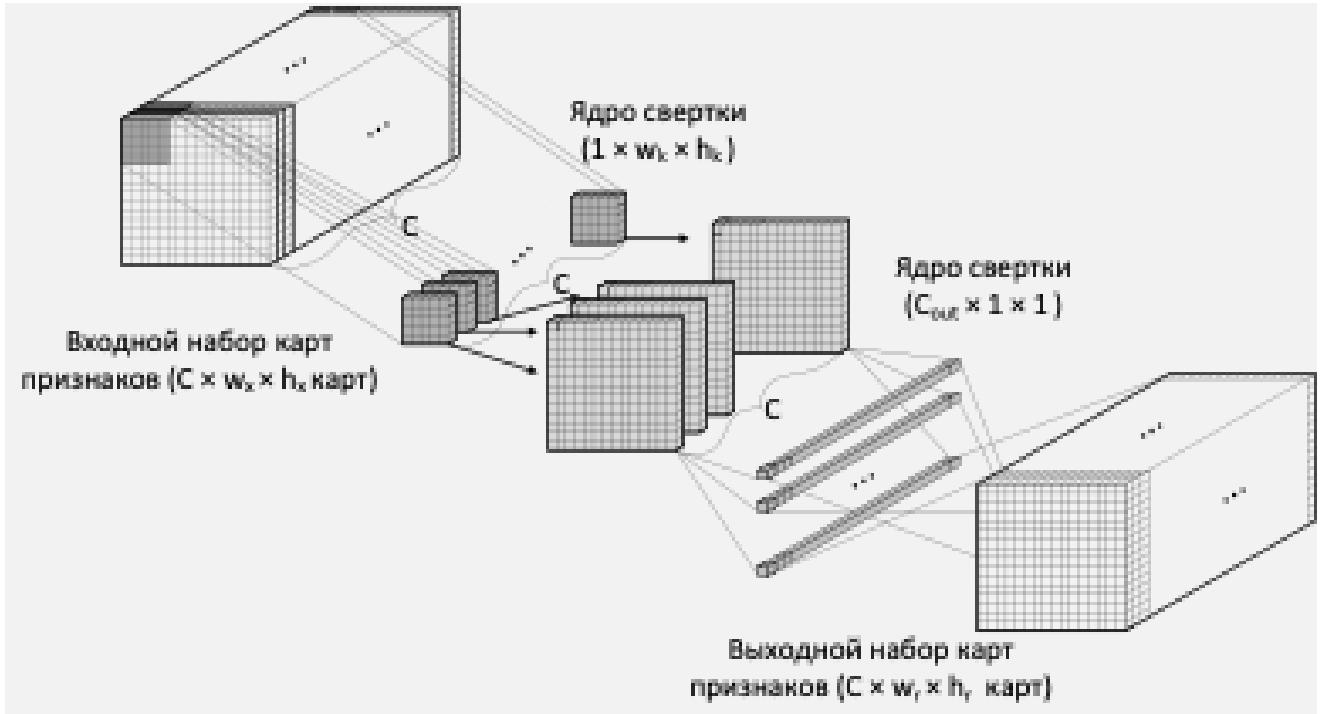


Рисунок 5.38 – Иллюстрация работы глубокой свертки depthwise separable convolution

Архитектура DenseNet Среди различных вариантов реализации идеи остаточного слоя в данном разделе хочется отметить работу 2017 года [159]. В данной работе было предложено расширить понятие блока с остаточными связями до вида, который принято называть **DenseNet**. Иллюстрация блока DenseNet приведена на рисунке 5.40. По существу, архитектура DenseNet может быть классифицирована как отдельный класс сверточных нейронных сетей. Идея блока DenseNet заключается в организации набора остаточных связей таким образом, что информация с каждого слоя блока DenseNet добавляется к результату последующего блока. Блок DenseNet можно описать как

$$x \rightarrow [x, f_1(x), f_2([x, f_1(x)]), f_3([x, f_1(x), f_2([x, f_1(x)])]), \dots].$$

Добавление информации описанное в выражении (5.3.2) происходит без соответствующего увеличения числа параметров, что позволяет обойти т.н. проблему проклятия размерности о которой говорилось выше. Таким образом, не смотря на большое число признаков в каждом из слоев, общее число параметров в блоке DenseNet остается небольшим, что дополнитель но снижает вероятность переобучения нейронной сети и ускоряет ее обучение.



Рисунок 5.39 – Иллюстрация блока DenseNet с остаточными связями

Каждый слой в блоке имеет доступ к входному слою – что ускоряет обучение. Для варьирования общим числом параметров модели между слоями DenseNet используют промежуточные простые слои.

Благодаря использованию блока DenseNet авторам работы [159] удалось создать архитектуру с числом слоев от 121 до 264. При этом данные архитектуры имели точности сопоставимые с классическими моделями ResNet с тем же числом слоев, однако при этом в архитектурах DenseNet число параметров было в 3-10 раза ниже [159]. Иллюстрация архитектуры DenseNet приведена на рисунке 5.40. Регулирование размера карт признаков между блоками осуществляется при помощи промежуточных операций свертки и пулинга.

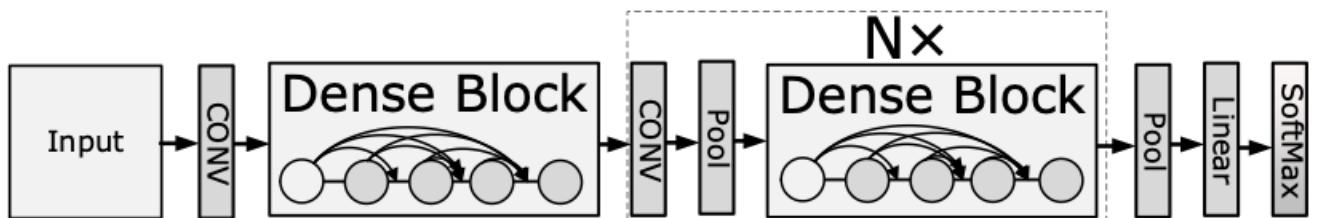


Рисунок 5.40 – Иллюстрация архитектуры DenseNet

Идея использования остаточной связи стала революционной в сверточных нейронных сетях, практически все последующие работы, предлагающие новые архитектуры сверточных и полносвязных нейронных сетей содержали остаточные связи в различных вариантах в рамках кодировщика признаков (feature encoder) [151].

В работе 2020 года CSPNet авторами было замечено, что недостатком архитектуры DenseNet является избыточное повторное использование значений градиента во время обучения. Таким образом слои Dense блока повторно обучаются копиями уже использованной информации. Для решения этой проблемы авторы CSPNet предложили разделить карты признаков на две части перед использованием в блоке. Одна часть используется в основной

ветви DenseBlock и проходит через переходной слой, а другая комбинируется с ней после этих операций. Таким образом с одной стороны градиенты разделяются, а с другой стороны карты признаков разделяются, образуя еще больше путей при меньшей вычислительной сложности. Иллюстрация работы CSP блока показана на рисунке 5.41.

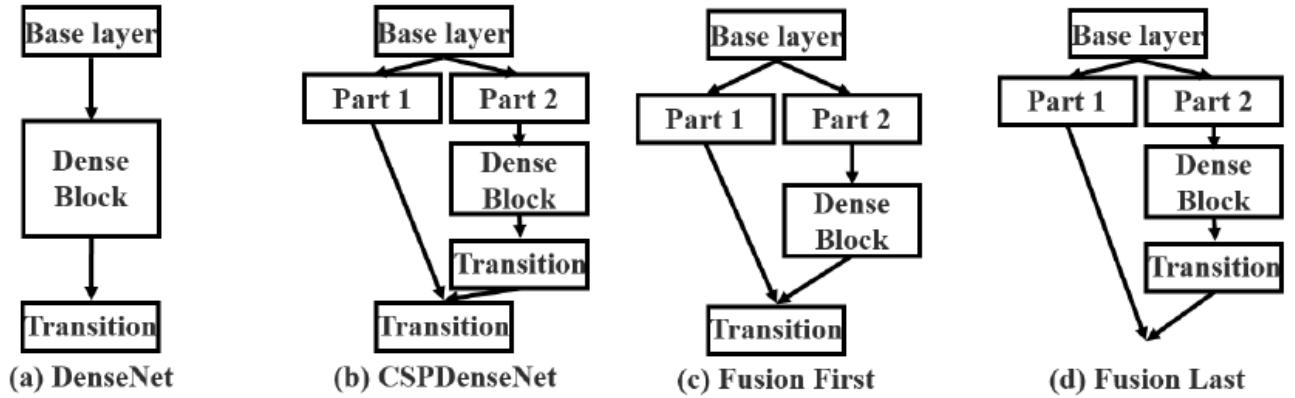


Рисунок 5.41 – Иллюстрация архитектуры CSP блока

Таким образом цель CSP блока состоит в максимизации различия комбинаций градиентов. Авторами рассмотрены два варианта блока. Вариант CSP (Fusion First) предполагает объединение карт признаков перед переходным блоком. Такой подход приводит к переиспользованию большей части градиентов. Второй вариант CSP (Fusion last) предполагает объединение карт признаков после переходного блока. В этом случае переиспользование градиента ограничено так как поток градиента ограничен. Подход CSP был использован также для ResNet и ResNeXt. При этом авторы отметили, что для ResNeXt блока не нужно делать узкого горла так как в основной ветви используется только половина карт признаков. С CSPNet can be also easily applied to ResNet and ResNeXt. Since only half of the feature channels are going through Res(X)Blocks, there is no need to introduce the bottleneck layer anymore.

Архитектура Big Transfer Architecture (BiT) Рассмотренные варианты архитектур на основе ResNet представляют лишь часть тех, которые предложены в литературе. До сих пор появляются идеи развития этого подхода, в том числе вполне успешные. Одними из современных модификаций ResNet также являются такие архитектуры, как Big Transfer Architecture (BiT) [160, 161]. Модель основана на стандарте ResNet (50, 101 и 152) с увеличением глубины и ширины (в масштабе от 1 до 4 раз). Архитектура ориентирована на маштабное предобучение модели в соответствии с современными возможностями. Масштаб определяется глубиной архитектуры и размером набора данных для предобучения. В силу такой ориентации в архитектуре операция BatchNorm (BN) заменена на комбинацию Group-Norm и Weight Standardization. При обучении больших моделей небольшими партиями для каждого устройства BN работает плохо или влечет за собой затраты на синхронизацию между устройствами. Было показано, что GN в сочетании с WS позволяет повысить производительность при обучении для небольших батчей для наборов данных ImageNet и

СОСО. Также они полезны для обучения с большими батчами и оказывают значительное влияние на снижение временных затрат при переносе обучения. В оригинальной работе предложено три варианта BiT:

- BiT-L (большой размер): предварительно обучен на наборе данных JFT-300M (содержит от 10 до 1 миллиарда наборов данных изображений);
- BiT-M (средний размер): предварительно обучен на наборе данных Imagenet-21K (содержит 14 миллионов наборов данных изображений);
- BiT-S (малый размер): обучение на наборе данных ILSVRC-2012 (1,3 миллиона наборов изображений).

Также авторы предложили набор эвристических правил для тонкой настройки BiT-HyperRule, где [160]:

- Фиксируются все гиперпараметры, кроме: продолжительность и расписание изменений значений скорости обучения во время тренировки, и разрешения входных изображений.
- Если размер изображения меньше 96×96 , производится перемасштабироване изображения до размера 160×160 и обрезка до размера 128×128 . Если размер больше 96×96 , перемасштабироване изображения до размера 448×448 и обрезка до размера 384×384 .
- Если размер набора данных превышает 20 000 изображений, используется аугментация методом MixUp.
- Если возможно, размер батча устанавливается 512.

Блок ResNet-D В 2019 году также был описан успешный набор эвристических техник для повышения качества обучения архитектур ResNet. Набор был назван Bag of Tricks, для набора также была представлена модификация блока ResNet, названная ResNet-D [162]. Иллюстрация предложенного блока и еще двух модификаций, проанализированных авторами приведены на рисунке 5.42.

Предложенный набор техник включал следующие техники аугментации и регуляризации обучения.

- Случайная обрезка (crop) путем обрезки изображений, с соотношением сторон выбранным случайным образом $3/4$ или $4/3$ и область случайным образом выбрана в диапазоне от 8%, до 100% площади изображения. Изображение затем изменяется в размере до 224×224 .
- Отразить по горизонтали с вероятностью 0,5 (flip).
- Масштабирование значений в шкале HSV (оттенок, насыщенность и яркость) с коэффициентами в диапазоне $[0,6, 1,4]$, выбранными при помощи равномерного распределения.

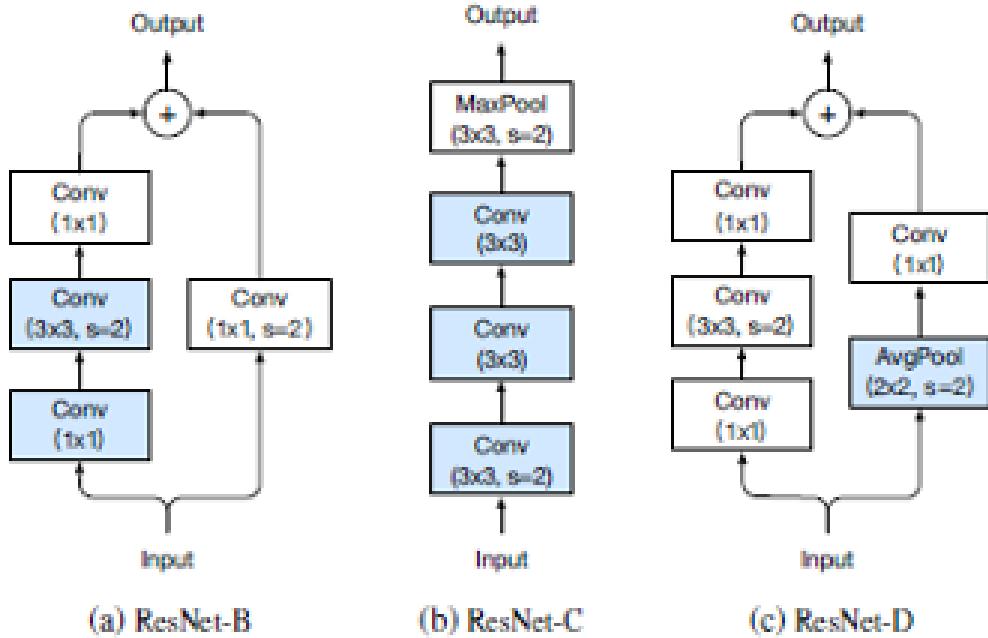


Рисунок 5.42 – Иллюстрация блока DenseNet с остаточными связями

- Добавление т.н. PCA шумов с интенсивностью, выбранной из нормального распределения $N(0, 0,1)$. Под термином «PCA шумы» понимается идея разложения изображения методом главных компонент, добавление шумовых составляющих к каждой компоненте и восстановление изображения [163].
- Использование кросс-аугментации методом MixUp (для меток и изображений). В случае включения MixUp вероятность его использования должна быть 100% - то есть все обучающие примеры.
- Перед подачей на вход сети изображение нормализуется для формата RGB, путем вычитая значений $[123,68; 116,779; 103,939]$ и разделив на значения $[58,393; 57,12; 57,375]$ для красного, зеленого и синего каналов, соответственно. Использование сглаживания меток по выражению

$$q_i = \begin{cases} 1 - \varepsilon, & i = y \\ \varepsilon/K - 1, & i \neq y \end{cases}, \quad (5.26)$$

где K – число классов, а $\varepsilon = 0, 1$. Использование дисциллированной функции потерь вида $L(y_i, softmax(z)) + T^2 L(softmax(r/T), softmax(z/T))$, где T – гиперпараметр, в оригинальной работе $T = 20$, для процесса дисцилляции использовались модель учитель ResNet-152D, модель ученик ResNet-50D Для обучения используется ускоренный градиентный спуск по Нестерова (NAG) [164]. Градиентный спуск используется с параметром weight decay для значений только весовых параметров и только для сверточных слоев и полносвязных слоев. Остальные параметры, включая смещение и параметры слоев батч-нормализации остаются нереугляризируемыми. При инициализации для слоев батч-нормализации в конце каждого ResNet блока коэффициент масштаба γ устанавливается в 0. Таким образом, все остаточные блоки

просто возвращают свои входные данные, имитируя сеть, которая имеет меньшее количество слоев и легче обучается на начальном этапе.

При обучении предложено как можно больше увеличивать размер батча. Это объясняется тем, что увеличение размера батча не меняет сходимости стохастического градиента, но уменьшает его дисперсию. Поэтому скорость обучения может быть увеличена, чтобы добиться большего прогресса. Предложено выбирать начальную скорость обучения 0,1 и размер батча 256. В оригинальной статье каждая модель обучается в течение 120 эпох на 8 графических процессорах. При переходе на больший размер пакета b начальная скорость обучения предложено увеличивать до $0,1 \times b/256$. Однако также отмечается, что использование слишком большой скорости обучения может привести к численной нестабильности последующей тренировки сети. Авторами также предложено использовать в начале быстрое снижение скорости обучения со скачкообразным возвратом к изначальному значению, примерно на 5 эпохе (warm-up). Также авторы отметили, что размер бата может быть увеличен при переходе к разрядности вычислений обучения FP16 (16 бит). В экспериментах получено, что при размере бата 1024 и разрядности FP16 точность при обучении вырастает на 0,5% - 1%. Также авторами предложено использование косинус-закона спада скорости обучения (без подогрева). Таким образом выражение для скорости может быть описано следующим образом [162]:

$$\eta_t = \begin{cases} \frac{\eta}{2}, & t \leq T_w \\ \frac{\eta}{2} \left(1 + \cos\left(\frac{(t-T_w)\pi}{T-T_w}\right)\right), & t > T_w \end{cases} \quad (5.27)$$

Модификация Resnet strikes back В работе [165] 2021 года были предложены дополнительные шаги по улучшению архитектуры. В данной работе были предложены следующие шаги по улучшению точности архитектуры.

- Использована бинарной кросс-энтропии (много меточной классификации вместо много классовой) – было показано что такой подход повышает точность.
- Использование аугментации, состоящей из:
 - вырезания с масштабированием (Random Resized Crop, RRC));
 - горизонтального отражения (horizontal flip);
 - смешивание изображений и меток MixUp и CutMix,
 - а также процедура случайной аугментации RandAugment, состоящая в выборе случайного подмножества простых видов аугментации из расширенного множества и воздействующих с заданной интенсивностью;
 - при возможности предлагается использование процедуры Repeated Augmentation [118].
- Использование стохастического дропаута (Stochastic Depth) [135].
- Использование cos-schedule для скорости обучения.

- Использование оптимизатора методом Lamb [166].

Модификация пересмотренный Resnet В работе того же 2021 года [167] были также предложены такие следующие приемы обучения.

- Снижение интенсивности значений weight decay.
- Чем меньше модель, тем ниже должно быть разрешение модели.
- Для длительного обучения лучше выбирать более глубокую модель, чем более широкую.
- К блоку ResNet целесообразно добавить SE слой, который описан в соответствующем разделе [168].

6 Современное состояние глубоко обучения в задачах компьютерного зрения

6.1 Мобильные сверточные архитектуры 2016-2018

Как уже было отмечено выше с 1989 до 2016г основным направление развития сверточных нейронных сетей было поиск путей их углубления. В том числе, эта тенденция была связана с ростом возможностей серверной вычислительной техники которая подразумевалась в основных приложениях компьютерного зрения. Однако, к 2016 году успех глубокого обучения сверточных нейронных сетей все с большей интенсивностью открывали вопрос об их использовании в низкопроизводительный (в т.ч. мобильных) вычислительных устройствах. К этому моменту уже были предложены ряд техник по сжатию (компрессии) обученных нейронных сетей для их реализации в конечных устройствах [169]. Были предложены и специализированные архитектуры аппаратных ускорителей для нейронных сетей [170]. Однако, процесс компрессии и ускорения не позволял изначально обучить модель с требуемым размером. То есть для экспериментов с обучением нейронных сетей все равно требовались высокопроизводительные серные вычислительные устройства и высокие временные затраты.

Архитектура SqueezeNet. В 2016 году для решения проблемы мобильных архитектур авторами работы [171] была предложена архитектура нейронной сети, обладающей производительностью на уровне AlexNet (см. выше) при числе параметров в 50 раз ниже (5 МБайт против 240 у AlexNet). Архитектура сети была названа **SqueezeNet**. Основные идеи SqueezeNet заключались в следующем.

- Замена части традиционных сверток на свертки 1×1 (точечные свертки) - что снижает число параметров, например в случае свертки 3×3 снижает в 9 раз.
- Использования слоя сокращения числа карт признаков перед сверткой (сжимающий слой, squeeze layer). Предполагается, что не все карты признаков необходимы для конечного результата. В ходе обучения нейронная сеть должна научиться оставлять только не обходимые карты. Интуитивно данный принцип следует из предположения, что в правильно обученной нейронной сети каждый слой работает независимо от последующих слоев.
- Реализация субдискретизации (пулинга) только в окончательных слоях нейронной сети. По мнению авторов и результатам работы Кайменга Хе [172] отказ от пулинга в начальных слоях должен способствовать увеличению обобщающей способности при классификации. Такой подход предложено называть **задержанная субдискретизация (delayed downsampling)**.

- Основной блок нейронной сети squeezeNet - т.н. fire block, представляющий собой сначала слой сокращение числа карт признаков (squeeze), затем слой расширение (expand). Таким образом общее число карт признаков и соответственно параметров нейронной сети остается сравнительно не большим при реализации прочих идей, использованных в архитектуре AlexNet. Также следует отметить, что все свертки слоя расширения делились на два блока - точечные свертки и свертки 3×3 , после проведения операции сворачивания и активации результаты объединялись (конкатенировали). Доля сверток каждого вида регулировалась при сохранении общего числа карт признаков.

Идеи, описанные в рамках реализации SqueezeNet положили стали отправной точкой в исследовании архитектур СНС для низкопроизводительных устройств [173, 57].

Архитектура ShuffleNet. Также в 2017 году была предложена следующая идея оптимизации архитектур путем использования групповой свертки с перемешиванием. Такая архитектура была названа ShuffleNet [174]. В основе данной архитектуры следующая идея. Групповая свертка делит входные карты объектов на две или более групп и выполняет свертку отдельно для каждой группы. При групповой свертке веса не распределяются между группами — каждая группа может выучить свои параметры — свои признаки. Однако, само по себе это может быть недостатком, если нет обмена информации между разными группами, потому что это может ограничить способность сети к выделению более высоконивневых признаков. Однако можно увеличить вариации информации путем перетасовки групп. После групповой свертки операция перетасовки каналов перестраивает карты выходных признаков, чтобы увеличить информацию для каждого уровня рецептивного поля. Таким образом блок ShuffleNetV1 состоит из нескольких этапов.

- 1 этапа - групповой свертки 1×1 с 3 группами. Слой может использоваться с сокращением числа каналов (уменьшает количество каналов в 4 раза).
- 2 этапа – перетасовка каналов.
- 3 этапа свертка с ядром 3×3 . Используется со слоем батчнорм, но без ReLU. Авторами [174] было обнаружено, что отсутствие использования ReLU дало лучшие результаты здесь.
- 4-этапная групповая свертка 1×1 для повторного увеличения количества каналов (чтобы соответствовать количеству каналов остаточного соединения).
- Остаточная связь добавляется к выходным данным. Если слой 3×3 имеет шаг 2, в ветвь остаточной связи добавляют слой пулинга с размером ядра 3×3 и шагом 3. В этом случае также результаты основной и остаточной ветвей конкатенируются вместо сложения.

Иллюстрация принципа работы ShuffleNet а также других вариантов, проанализированных авторами приведена на рисунке 6.43.

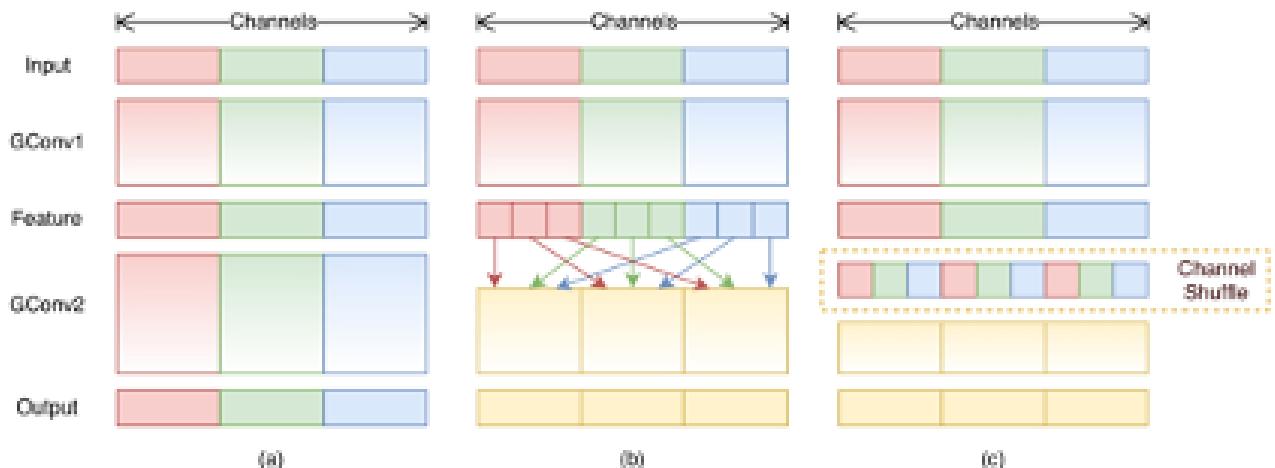


Рисунок 6.43 – Иллюстрация блока shufflenet (в) и других вариантов групповой свертки, проанализированных авторами

Следует обратить внимание, что групповая свертка аналогична глубинной свертке. Фактически, глубокая свертка иногда реализуется с использованием более общей групповой свертки. Позже была предложена модификация слоя ShuffleNet V2 [175]. В блоке этой архитектуры используется модифицированная блочная структура: 1 этап - операция разделения на две половины каналов (2 группы). На 2 этапе используется один из нескольких вариантов слоев, которые могут как иметь, так и не иметь групповые свертки. Иллюстрации нескольких вариантов блока ShuffleNet 2V в нескольких вариантах приведены на рисунке 6.44.

Архитектура MobileNet. Одним из основных этапов развития мобильных нейронных сетей в период 2016-2018 гг. стали работы Андрю Ховрада и команды соавторов из компании Google по разработки семейства архитектур **MobileNet V1/V2**[158] (MobileNets V1 2017г) [176] (MobileNet V2 2018г). Основные идеи архитектур MobileNet V1/V2 заключаются в следующем.

- Замена блока классической свертки 3×3 на т.н. **глубокую разделяемую свертку (deep wise convolution)**. Такая свертка, по существу, является комбинацией предельного случая групповой свертки - когда число групп равно числу карт признаков и последующей точечной свертки регулирующей число карт признаков. Данная идея была предложено в архитектуре Xception [157] - вариации семейств архитектур ResNet и Google Inception. Главным достоинством такого подхода является сокращение общего числа параметров свертки.

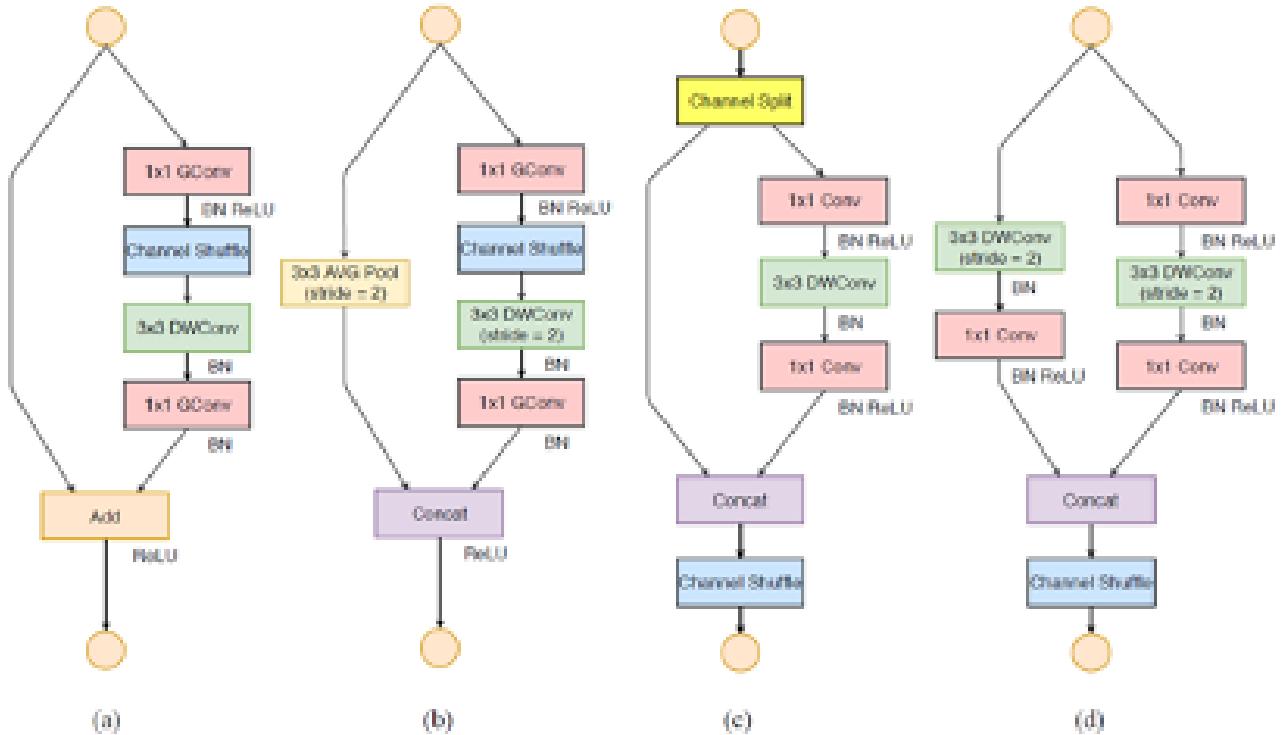


Рисунок 6.44 – Иллюстрация блока shufflenet v2 в нескольких вариантах

- Реализация субдискретизации (пулинга) путем использования сверток с увеличенным шагом. То есть сверточное ядро движется по входной карте признаков не с шагом в 1 пиксель (в одну позицию), а с увеличенным шагом. Отметим, что в работе [177] было показано, что такая замена не снижает точности по сравнению с макспулингом, однако снижает общее число вычислительных операций в нейронной сети.
- основной блок в архитектуре сети - MobileNet V1/V2 блок состоит из слоя расширения числа карт признаков (expantion); слоя глубокой разделяемой свертки и слоя сокращения чисел карт признаков (слой проекции - projection или bottleneck layer). Также блок MobileNet V2 содержит остаточную связь. Основная идея блока MobileNet заключается в том, чтобы увеличить число карт признаков, провести над ними операцию фильтрации (глубокой разделяемой свертки) и затем оставить только полезные признаки (слой проекции).
- Для оптимизации архитектуры нейронной сети выбиралась общая структура сети - базовая сеть (baseline - определяет число слоев их взаимосвязи); при этом варьировались два параметра: параметр ширины (width multiplier) и параметр входного разрешения (resolution multiplier). **Параметр ширины** - это какое количество карт признаков будет использоваться в блоке (слой expantion), при этом число карт признаков после сокращения фиксировано (слой projection). **Параметр разрешение** - это то какой размер входного изображения ожидается - чем меньше размер, тем меньше вероятность выделения признаков небольших размеров, но в квадрат раз ниже вычислительная сложность.

Итоговая архитектура MobileNet v1 состоит из 13 блоков. Заметим в архитектуре не используется локальный пулинг. Вместо этого некоторые из слоев глубокой свертки имеют шаг 2 [158]. Иллюстрация блока MobileNet V1 приведена на рисунке 10 также на рисунке 10 также проиллюстрирована глубокая разделённая свертка. Отметим, что в каждый сверточный слой блока содержал помимо свертки также батч-нормализацию и функцию активации ReLU6 - вариант функции ReLU с насыщением. Такая функция может быть выражена как

$$ReLU6(x) = \min(\max(0, x), 6). \quad (6.28)$$

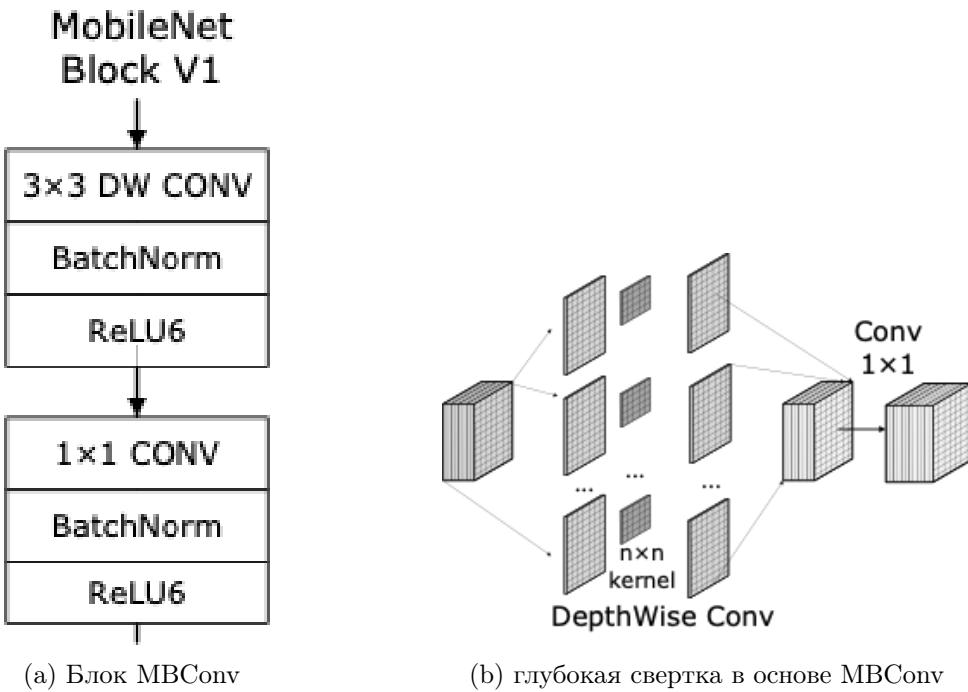


Рисунок 6.45 – Иллюстрация архитектуры блока MBConv

В блоке MBNet V2 последний слой – слой сжатия не имеет функции активации. Авторы [176] предположили, что так, как этот слой выдает небольшое число карт признаков, использование нелинейностей может привести к потери полезной информации. Достоинство блока обратного остаточного блока – это возможность управлять числом параметров в сети. То есть даже если глубокая свертка должны быть выполнена для весьма большого числа карт признаков – общее число параметров на входе и выходе останется небольшим. При этом, степень расширения влияет на то, сколько признаков можно извлечь из изображения, а размер входного изображения влияет на то каким должно быть оптимальное receptive field для выделения признаков. Напомним, что чем выше значение receptive field, тем больше возможность учета контекста признака, но ниже детализация работы с ним. Иллюстрация блока MobileNet V2 приведена на рисунке 6.46. На рисунке символ α – параметр ширины; символ ρ – параметр разрешения.

Иллюстрация блока MobileNet V2 приведена на рисунке 6.46. Отметим, что в каждый сверточный слой блока содержал помимо свертки также батч-нормализацию и функцию активации ReLU6 - вариант функции ReLU с насыщением. На рисунке символ α - параметр ширины; символ ρ - параметр разрешения.

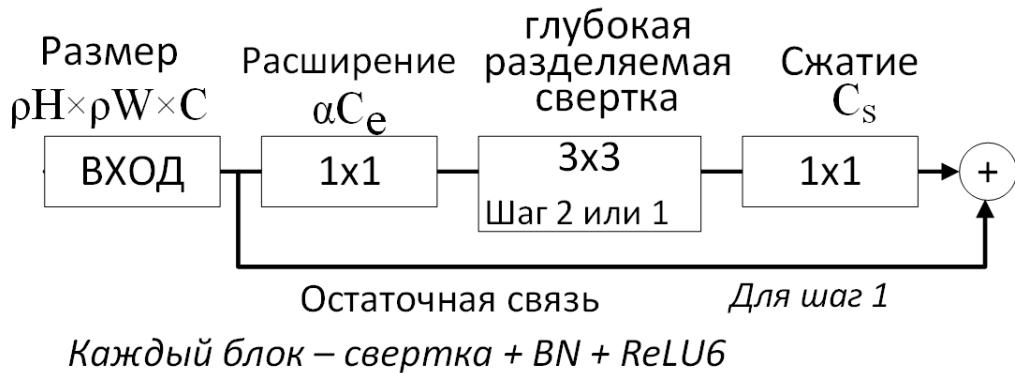


Рисунок 6.46 – Иллюстрация блока MobileNet V2

Итоговая архитектура MobileNet v2 состоит из 17 блоков. В головной части архитектуры вместо полносвязного слоя используется точечная свертка как аналог той же операции. Таким образом блок MobileNet V2 имеет меньше параметров, чем для версии V1 и лучшую точность. Блок архитектуры MobileNet V2 стала эталоном оптимизированного блока и часто используется других архитектурах сверточных нейронных сетей.

Архитектура RepVGG. В работе 2021 года [178] авторы предложили использование приема, названный структурной репараметризацией обучения и работы нейронной сети. Цель такого приема провести обучение на избыточной архитектуре. При этом работа сети выполняется упрощенной ее версии. В качестве эталона рабочей архитектуры авторы предложили рассматривать VGGNet, тогда как во время обучения наиболее стабильные результаты по мнению авторов могут быть достигнуты для ResNet.

В основе своих рассуждений в работе [178] выдвинута гипотеза, что стандартный слой свертки 3×3 имеет наилучшую "вычислительную плотность". Например, архитектура VGG-16 имеет в 8,4 раза больше FLOP, чем современная EfficientNet-B3, но работает в 1,8 раза быстрее. При этом авторы отмечают, что с точки зрения обучения известно, что архитектура с несколькими ветвями выгодней. Однако такая топология не эффективна с точки зрения занимаемой памяти (необходимо держать в памяти все результаты до конца работы блока). По исследованиям авторов в среднем перерасход составляет порядка 2 раз.

Рисунок 6.47 – Иллюстрация принципа работы архитектуры RepVGG

В качестве решения описанного противоречия авторы предложили использовать топологию модернизированного ResNet в процессе обучения нейронной сети. В процессе работы отбрасывать дополнительные ветви. Такой подход позволил сделать обучение эффективным, сохраняя при этом эффективное расходование памяти и низкую вычислительную сложность во время работы. Иллюстрация принципа работы ReprVGG приведена на рисунке 6.47. В качестве обоснования работы авторы предлагают интерпретировать работу ResNet как ансамбля из 2^n "плоских" моделей, где n число блоков так как каждый блок имеет "поток информации" для двух путей. Такая топология полезна во время обучения. В работе [178] авторы предлагают модель, которая интерпретируется как 3^n "плоских" моделей во время обучения. При этом одна из остаточных ветвей имеет обучаемую свертку 1×1 . Каждый путь сопровождается слоем батч-нормализации перед сложением.

Для предложенной конфигурации предложенный прием репараметризации заключается в следующем:

- Во время работы используется представление слоя батч-нормализации в виде операции свертки со смещением:

$$M' = bn(M * W) = (M * W - \mu) \frac{\gamma}{\sigma} + \beta = M * W \frac{\gamma}{\sigma} + \beta - \frac{\mu\gamma}{\sigma} = M * W' + b' \quad (6.29)$$

где M' результат работы слоя $bn(M * W)$; M - входные данные; W - ядро слоя; $\mu, \sigma, \gamma, \beta$ - набор параметров батч-нормализации; W', b' - параметры слитой нормализации.

- Авторы предложили рассматривать соли как: один имеющий свертку 3×3 и два имеющих свертки 1×1 (одна из сверток с фиксированными параметрами), а также отдельно 3 вектора смещения.
- Во время работы смещения слоев складываются, а свертки 1×1 дополняются нулями до размера 3×3 и складываются.

Авторы предложили семейство архитектур RepVGG, имеющих ResNet подобную структуру, в которой, однако свертки с большим размером ядра заменены на каскадные свертки. Вместо операций максимального пулинга используются свертки с шагом 2. Среди вариантов архитектур имеют место решения как для классификации, так и для других задач компьютерного зрения.

6.2 Попытки использования идеи "слой внимания"

Общая идея внимания в изображениях. Идея "подсвечивания" для нейронной сети наиболее информативных участков входных данных (информационных с точки зрения задачи) присутствовала в сообществе исследователей нейронных сетей начиная с 1990-х [179]. В 2014 Бахданау (Bahdanau) предложил использование данной идеи в задачах машинного перевода, предложенная архитектура была названа **механизм внимания (attention)** [180]. Идея механизма внимания получила широкое развитие и распространение в различных

приложениях обработки естественного языка и других схожих задачах [181]. В 2016-2017 годах в работах [182, 183, 184] была развита идея **механизма само-внимания (self-attention)** для задач обработки естественного языка. Также в работе [184] были предложены т.н. **много-головое внимание (multi-head attention)** и **блок трансформер (transformer)**. Начиная с 2017 года идеи механизма внимания активно исследуются в приложениях к сверточным нейронным сетям в различных задачах компьютерного зрения [185, 186]. Отметим, что в противовес к само-вниманию, классический механизм внимания называют **кросс-внимание (cross-attention)**. Основной идеей механизма внимания является обучение слоя (блока) выделять наиболее важные участки входных данных при помощи нормализации функцией softmax (3.3) [185]. В случае само-внимания слой обучается только на основе входных данных. Если речь идет о кросс-внимании то слой обучается на входных и уже предсказанных выходных данных. Иллюстрация механизма само-внимания для сверточных сетей показана на рисунке 6.48.

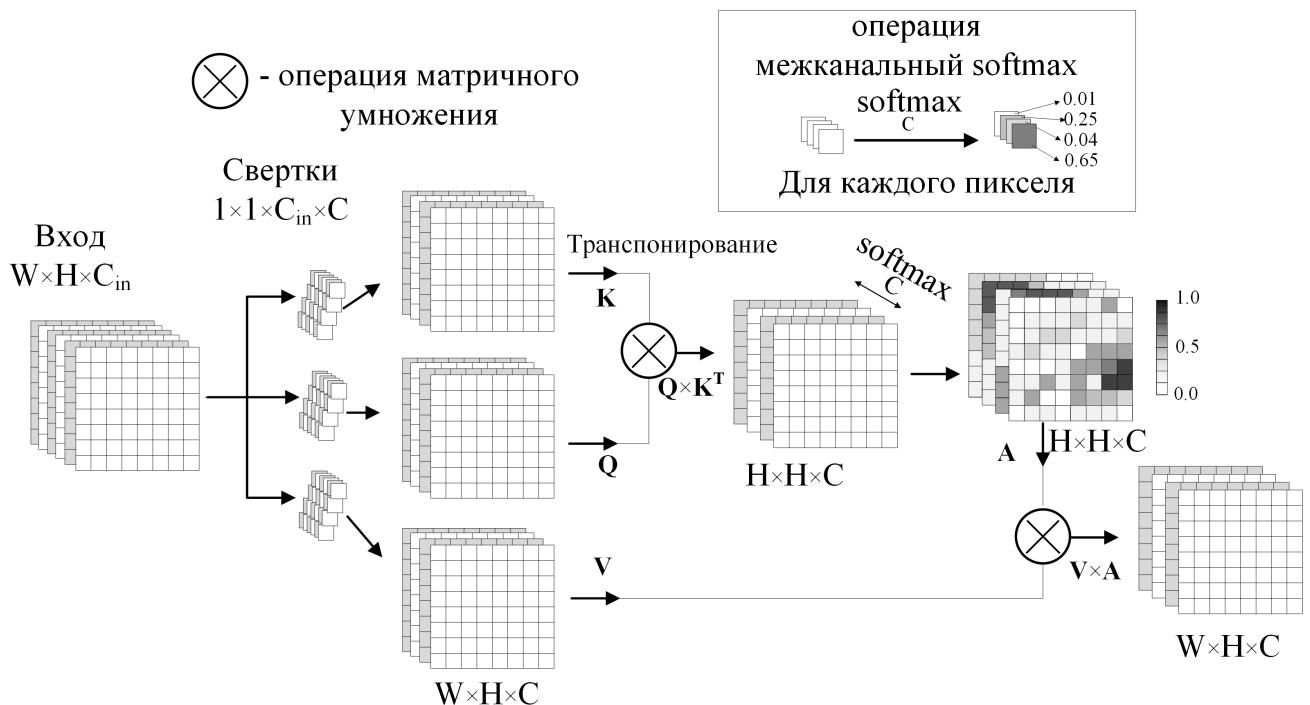


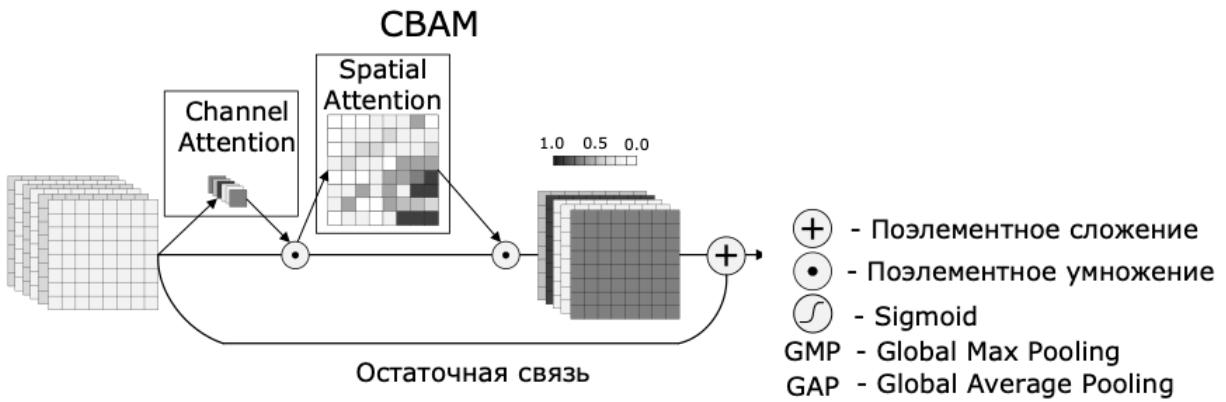
Рисунок 6.48 – Иллюстрация слоя само-внимания для сверточных сетей

Иллюстрация на рисунке 6.48 соответствует типу меж-канальное самовнимания, так как взвешивание функцией softmax проводится по каждой позиции пикселя, но для всех каналов. Отметим, что иллюстрация 6.48 это упрощенный пример авторов и носит только иллюстративный характер. В основе показной идеи лежит попытка перегруппировать пиксели карт признаков и взвесить их таким образом, чтобы наиболее важные участки имели большее значение. В данном случае участки определяются для каждой позиции пикселя между разными картами признаков. Один и тот же набор карт признаков трижды претерпевает нелинейные изменения при помощи точечной свертки с функций активации. Три измененных копии входных данных обрабатываются по-разному. Первые две копии перемножаются, образуя новый набор карт признаков. На этом наборе каналов при помощи

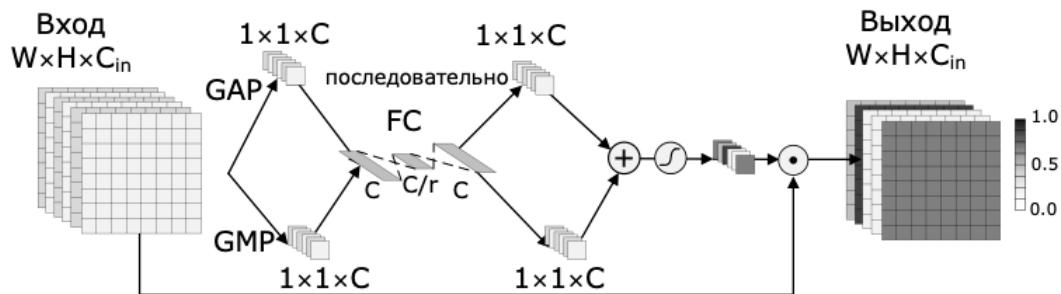
межканального softmax взвешивания выделяются наиболее важные участки. Затем участки проецируются на третью копию входных данных позволяя выделить на ней на те самые важные участки. Так как весовые параметры всех сверток обучаются вместе с сетью, сеть сама должна определить, где у нее важные участки по отношению к каждой задаче.

В работе [187] показано, что слой самовнимания позволяет провести перегруппировку пикселей входного слоя - то есть делает результат работ нейронной сети независимым от нерегулярных особенностей каждого экземпляра входных данных. Другими словами слой внимания дополняет одно из наиболее важных свойств сверки - инвариантность к положению объекта (как по координатам, так и по наклону, повороту и т.д.). В работе [187] 2019 года авторы привели математические доказательства, что много-голове само внимание эквивалентно свертке. Однако, отметили, что вероятно, предпочтительнее использовать оба типа слоев вместе.

Блок пространственного и поканального внимания СВАМ. В литературе было предложен несколько реализаций слоя внимания для сверточных сетей. Одна из наиболее популярных идей — это блок канального и пространственного внимания. Такой подход призван выделить как признаки между каналами, так и для каждого из них. Если блок не нужен, то используется остаточная связь. Такой блок получил название Convolutional Block Attention Module (СВАМ) [188]. Иллюстрация работы блока СВАМ приведена на рисунке .



Канальное внимание (Channel Attention)



Пространственное внимание (Spatial Attention)

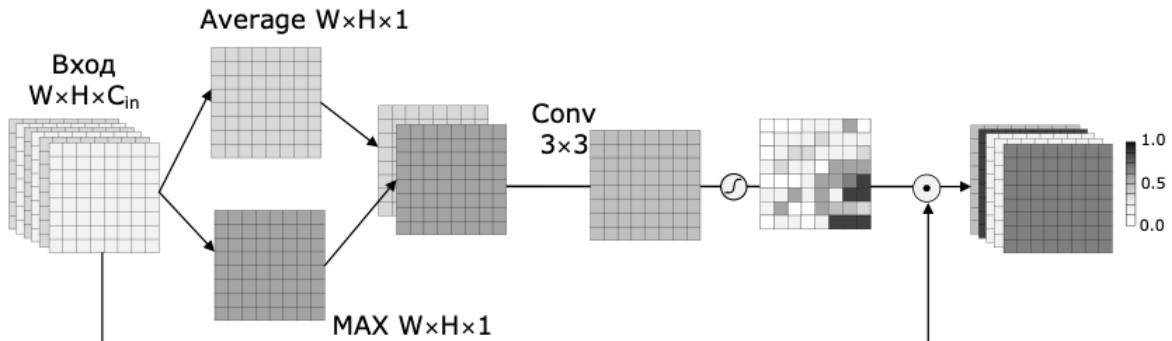
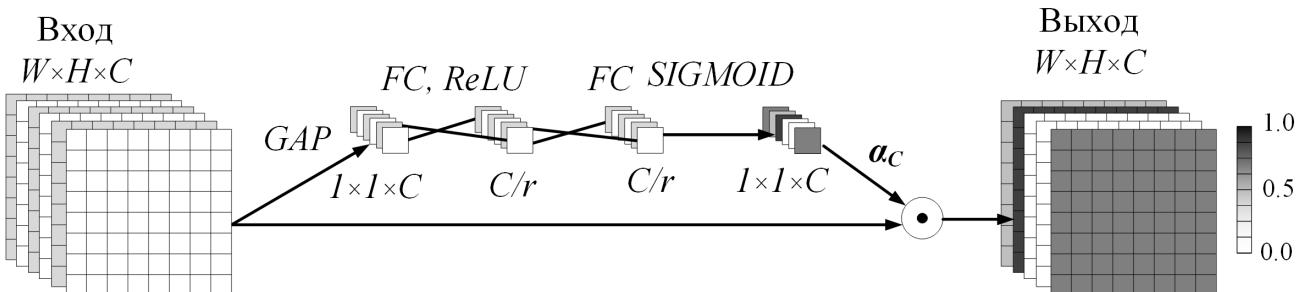


Рисунок 6.49 – Иллюстрации механизма пространственного и поканального внимания СВАМ (сверху), а также его канальной части (посередине) и его пространственной части (снизу)

Канальное внимание реализуется за счет глобального среднего и максимального пулинга (это GAP и GMP соответственно). Такие пулинги формируют векторы, с числом параметров, равным числу каналов входных данных. Затем векторы преобразуются при помощи полносвязных слоев и взвешиваются при помощи функции активации, так для каждой карты признаков определяется весовой параметр ее значимости.

Для пространственного внимания карт признаков усредняется подканально, также подканально выбираются максимальные значения пикселей – происходит такой межканальный пулинг. Из полученных карт признаков формируется одна при помощи свертки. Для полученной карты вычисляются пространственные веса, которые считаются одинаковыми для всех кар признаков исходного изображения. Эта маска весовых параметров перемножается с каждой из входных карт признаков. Таким образом удается реализовать раздельные канальное и пространственное подходы к вниманию в сверточных сетях [188].

Блок сжатия и возбуждения SE. В 2017 году в работе [168] концепция внимания для сверточных сетей был значительно переработана. В результате **Jie Hu и коллективом соавторов** был предложено **слой сжатия-возбуждения (Squeeze-and-Excitation layer, SE layer)**. Структура блока сжатия-возбуждения приведена на рисунке 6.50. В основе работы блока лежит идея выделения наиболее важных карт признаков (каналов) через их автокодирование (верхняя часть рисунка) и затем через функцию активации сигмоид (логистическая функция для каждого элемента). В результате такой операции формируются коэффициенты α_C , характеризующие "важность" каждой карты признаков. Таким образом, SE Layer призван помочь нейронной сети принимать во внимание только наиболее важные (регулярные, релевантные задаче) признаки. Отметим, что строго говоря слой SE Layer не является блоком внимания в классическом смысле так как использует логистическую функцию активации для каждого выхода вместо единой нормализации функцией softmax. Однако, часто слой SE Layer рассматривают в качестве варианта межканального внимания.



GAP - Глобальная усредняющая субдискретизация

FC - Полносвязный слой

○ - Умножение матриц ($W \times H$) на коэффициенты α_C

C/r - сжатие размерности

Рисунок 6.50 – Иллюстрация слоя Сжатия-возбуждения (SELayer)

Таким образом SE Layer пытается провести перекалибровку каналов для усиления значимости более информативных из них. То есть цель блока повысить информативность карт признаков – то есть чувствительность к канальным позициям полезных признаков. В некотором смысле SE блок альтернативен батч-нормализации. Также можно сказать, что такой слой — это вариация слоя внимания, он позволяет использовать глобальную

информацию для усиления конкретного признака. Каждый SE блок состоит из:

- Операция сжатия (Squeeze) информации из карт (при помощи global average pooling, GAP) .
- Операция возбуждения (Excitation) при помощи двух полно связных слоев для сжатой информации. Цель операции взвешивание информации из разных каналов.
- Операция перекалибровки (пере-взвешивания) с помощью набора функций sigmoid.

В оригинальной идеи пере-калибровка в начальных слоях позволяет возбудить информативные признаки не зависимо от классов (class-agnostic manner) - то есть слой «укрепляет» (регуляризирует) низкоуровневые представления признаков. В окончательных слоях SE блок позволяет выделить признаки, наиболее характерные для каждого класса [168]. Отличие SE от классического самовнимания состоит в замене функции активации softmax (то есть много кассовой функции – выделение одного класса) на набор сигмоидов (то есть много меточная классификация – выделение всех возможных классов). В некотором смысле это аналогично вспомогательным выходам в inception layer. Важное достоинство SE – низкая вычислительная сложность. На основе SE слоя были предложены различные варианты архитектур. Некоторые из них, связанные с блоком ResNet показаны на рисунке 6.51.

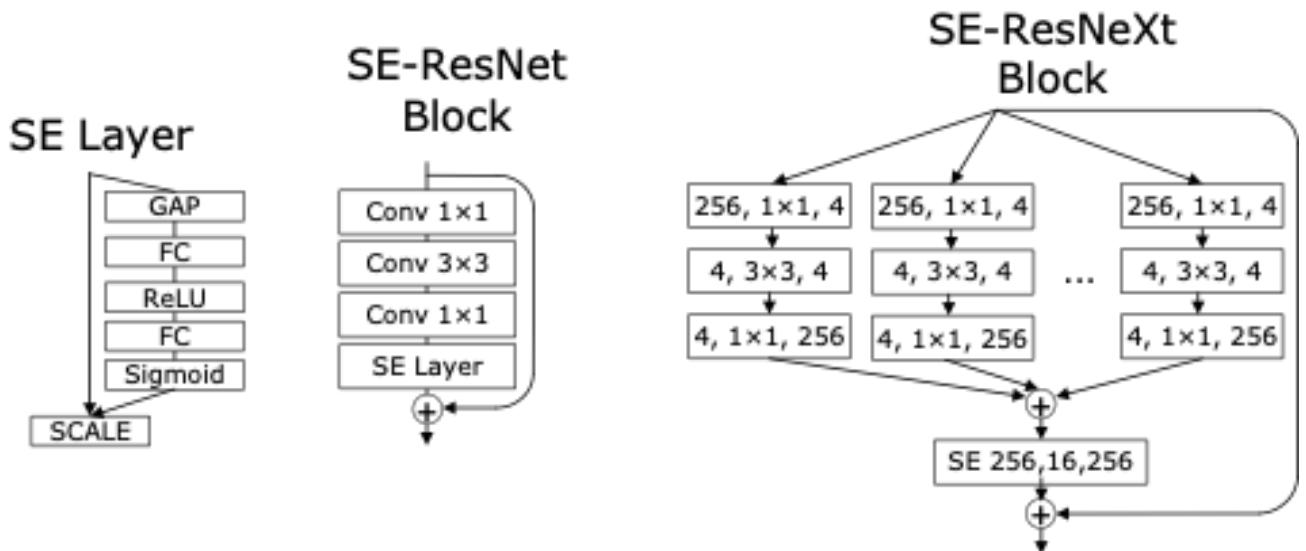


Рисунок 6.51 – Иллюстрация некоторых примеров блоков со слоем Сжатия-возбуждения (SELayer)

В 2017 году команда авторов работы [168] одержали победу в соревнования по классификации ILSVRC 2017 с ошибкой 2,25 (4,5)% по методике top-5-accuray [168]. Слой SE позже был использован во многих успешных архитектурах сверточных нейронных сетей. В частности, с использованием слоя SE были синтезированы блоки MobleNet V3 в 2019 году [189].

Не локальный блок. В работах [190, 191] были предложены модификации архитектур классификации, обнаружения объектов и для решения других задач компьютерного зрения с использованием блока глобального внимания. По существу блоки представляют собой оригинальное переосмысление идей SE block. По существу авторы работы [190] предложили оригинальную архитектуру слоя, выполняющего операцию вида $y_i = \sum_j f(x_i, x_j)g(x_i)/C(x)$, где $f(x_i, x_j)$ - некоторая функция оценки сходства, а $g(x_i)$ т.н. унарный функционал, а $C(x)$ - нормирующий множитель. Например $C(x) = \sum_j f(x_i, x_j)$. Такая архитектура в некотором смысле расширяет идею слоя внимания. Однако, в данном случае могут быть проанализированных различные не линейные функционалы f и g . Идея NL блока заключается в попытке учета "долговременного" или пространственно разнесенного контекста. Также авторами было предложено выполнять операции блока с применением пулигна слагаемого x_j . Такую модификацию было предложено назвать pooling trick.

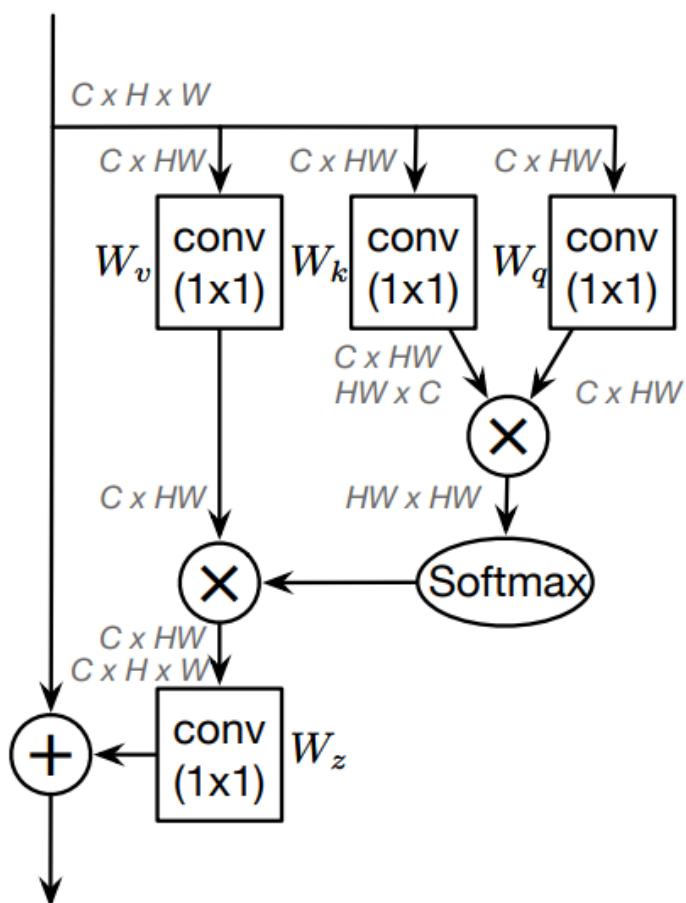


Рисунок 6.52 – Иллюстрация принципа работы Non-Local-block (NLNet)

На основании идей NL блока авторы работы [191] предложили модификацию блока внимания. Модификации включали упрощенную версию NL блка. Также в работе была показана связь данной идеи с SE слоем[191]. Общую концепцию, которая лежит в основе данных блоков было предложено назвать блоком глобального контекста (Global context). Иллюстрация данной модификации приведена на рисунке 6.52. Предложенная авторами архитектуру было предложено назвать GCNet. Данная архитектура имеет вычислительную

сложность ниже сравниваемых вариантов, при этом не снижая точности.

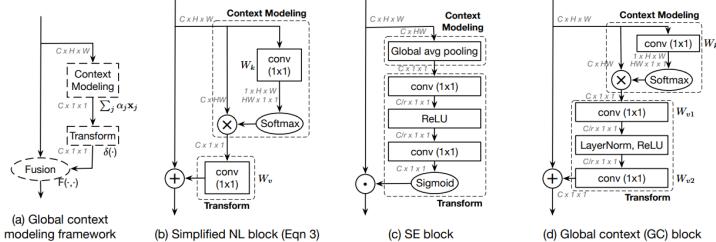


Figure 4: **Architecture of the main blocks.** The feature maps are shown as feature dimensions, e.g. $C \times H \times W$ denotes a feature map with channel number C , height H and width W . \otimes denotes matrix multiplication, \oplus denotes broadcast element-wise addition, and \odot denotes broadcast element-wise multiplication.

Рисунок 6.53 – Иллюстрация принципа работы Non-Local-block (NLNet) и global-context block (GCNet) в сравнении с SE block

6.3 Автоматический поиск архитектур

Идея поиск нейронных архитектур NAS. Идея автоматизации поиска архитектур нейронных сетей развивалась в сообществе исследователей с периода возникновения самой идеи глубокого обучения, по крайней мере с 2002 года [192]. Однако, существенных успехов удалось достичь только в 2017 году, когда был предложен метод, известный как **Neural Architecture Search (NAS)** [193]. Также отметим, что начиная с 2015 года проводятся соревнования по автоматическому синтезу алгоритмов машинного обучения, в том числе, но не только, по автоматическому поиску архитектур глубокого обучения (AutoML Challenge) [194]. Также отметим, что термин AutoML - автоматический поиск архитектур, включает, но не ограничен методами NAS [195, 196].

По существу, **методы NAS подразумевают три составляющие.**

- Пространство поиска (Search Space) - по каким блокам, по каким видам сверток, функций активации, методом регуляризации искать.
- Стратегия поиска (Search Strategy) – то как искать, как отбросить ненужное.
- Оценка качества модели (Performance Estimation Strategy or Candidate Evaluation Method) – например, точность и вычислительная сложность и напр. на неполных данных
- **Пространство поиска (Search Space).** Набор блоков, слоев или архитектур которые могут быть использованы. Выбор таких архитектур исходит из опыта исследователя. Отметим, что данный факт является наиболее фундаментальным ограничением методов NAS. В некоторых современных системах общая структура блоков и их максимальное число может быть фиксированы, но параметры блоков (наличие пулинга, число карт признаков, размер входного массива, число остаточных связей и т.д.) могут меняться. Такой подход носит название **Micro Search Space** или **Cell-based Search Space** [197]. Например, NAS может позволить сказать нужен ли пулинга, нужны ли остаточные связи и другие элементы структуры сверточной сети.

- **Стратегия поиска (Search Strategy).** Набор правил для выбора тех или иных блоков. От выбора стратегии поиска может зависеть число возможны комбинаций в пространстве поиска, как правило число комбинаций должно быть максимально ограничено.

В рамках методов NAS рассматриваются такие стратегии как [193, 198]:

- генетические эволюционные алгоритмы (начало 2000-ных);
- методы Байесовской оптимизации (2010-е);
- обучение с подкреплением (2017);
- рекуррентное обучение с кодированием всех вариантов в категориальные вектора (2018).

- **Оценка качества модели (Performance Estimation Strategy, Candidate Evaluation Method).** Данный вопрос также является не тривиальным. Так как к архитектуре могут быть предъявлены совместно требования высокой точности, оптимальной производительности для конкретной конфигурации оборудования или времени работы, требования к занимаемому месту в памяти и другие требования [197]. Также к данному пункту относится выбор метода снижения вычислительной нагрузки при подборе архитектуры. К таким методом можно отнести следующие [193]:

- сокращение тренировочной выборки (Lower fidelity estimates);
- испытания всех моделей с пониженным числом параметров (например можно снизить число карт в 2 для каждой исследуемой модели) (Lower fidelity estimates);
- ранняя остановка обучения моделей с экстраполяцией результатов (Learning Curve Extrapolation);
- инициализация весовых параметров новых вариантов архитектур весовыми параметрами предыдущих вариантов (сокращает число эпох тренировки сети) (Network Morphisms);
- обучение единой модели, включающей все варианты и использование комбинаций ее составляющих в качестве новых предобученных моделей (One-Shot Architecture Search).

Отметим, что все описанные подходы к упрощения процесса выбора архитектуры приводят к смещению конечного результата, однако могут быть использованы для ранжирования вариантов. После выбора итоговой архитектуры - она должна быть дообучена [193].

Архитектуры NASNet и PNASNet. В 2018 были опубликованы результаты оказавшие наибольшее влияние на развитие NAS: Зопф Б. и соавторы (команда Google Brain) предложили архитектуру **NASNet** [198], а Лью Ч. в соавторстве с коллективом, включающим Зопф Б. предложили **метод прогрессивного поиска архитектур (Progressive NAS, PNAS)**, а также архитектура **PNASNet** [199]. рекуррентный подход,

предложенный Зопфом в работе [200] и метод регуляризации путем выключения частей блоков при обучении каждого варианта архитектуры.

Для синтеза NasNet [198] использовался подход Cell-based Search Space. Общая архитектура состояла из двух типов ячеек (блоков, cells), составленных в заданном порядке. Авторами работы были подобраны два типа архитектур для набора данных CIFAR-10 и для набора данных ImageNet. Для поиска архитектур авторы использовали модификацию подходов обучения с подкреплением и рекуррентного обучению, предложенную Б.Зопфом в работе [200] и метод регуляризации путем выключения частей блоков при обучении каждого варианта архитектуры, названный Scheduled DropPath. Для ускорения поиска был предложен подход с сокращением тренировочной выборки.

В работе [199] авторы предложили стратегию последовательно усложнения структуры блоков с прореживанием пространства поиска (PNAS). Прореживание выполнялось предварительной оценкой точности для каждого из кандидатов-блоков. При этом общая структура сети фиксировалась перед проведением экспериментов.

Архитектуры PNASNet и NASNet показали практически эквивалентные результаты для набора данных ImageNet, однако метод PNASNet потребовал в 8 раз меньше вычислительных ресурсов и при этом архитектура PNASNet требовала несколько меньшего объема памяти. Архитектура PNASNet стала победителем на ILSVRC 2018 и достигла точности 96,2 % (на 1,2% выше чем вручную оптимизированные архитектуры)[199].

Примеры иллюстрации блока PNASNet приведены на рисунке 6.54.

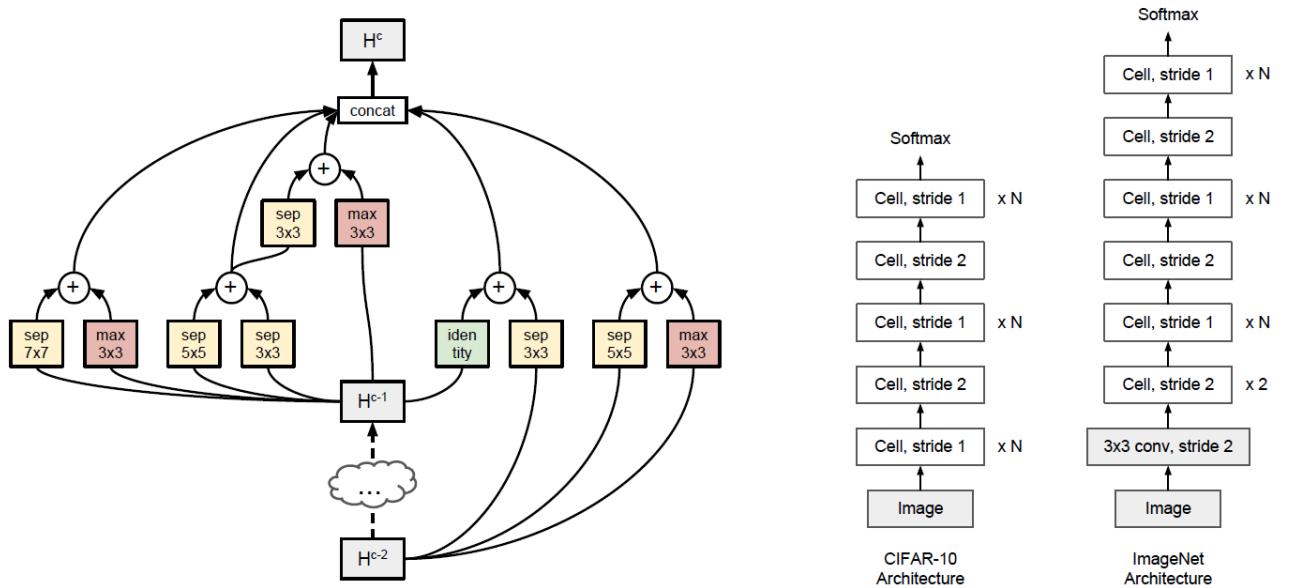


Рисунок 6.54 – Иллюстрация блока PNASNet

Архитектура MobileNet V3. Одним из наиболее популярных примеров работы NAS стала архитектура MobileNet V3. Эта архитектура стала приемником MobileNet V2 которая рассматривалась ранее. Особенностью архитектуры MobileNet V3 стала архитектура блока. Так же как и для второй версии блока – блок MobileNet V3 содержит этапы расширения глубокой свертки и сжатия, однако тут еще добавлен блок сжатия и расширения – SE блок.

То есть блок MobileNet V3 использует идею внимания. Также в данном блоке заменены функции активации. Вместо используемых ранее функций $ReLU_6$ тут авторы используют функцию, представляющую нелинейное ReLU - swish и ее модификацию, полученную линейной аппроксимацией т.н. hard-swish. Так же авторы модифицировали слой SE, использовав там точечные свертки вместо полносвязных слоев и заменив операцию сигмоид на $hard - \sigma$. Такие замены могут быть полезны с точки зрения отсутствия операций взятия экспоненты. Операция экспоненты может быть ресурсо-затратной для некоторых типов вычислителей.

$$h_{swish}(x) = x \cdot ReLU_6(x + 3)/6; \\ hard - \sigma = ReLU_6(x + 3)/6 \quad (6.30)$$

Графики функций swish и hard-swish приведены на рисунке 6.55.

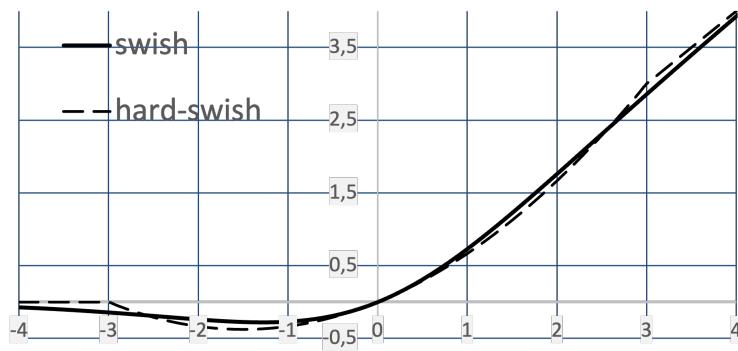


Рисунок 6.55 – Иллюстрация графиков swish и hard-swish

Иллюстрация блока MBconv3 приведена на рисунке 6.56.

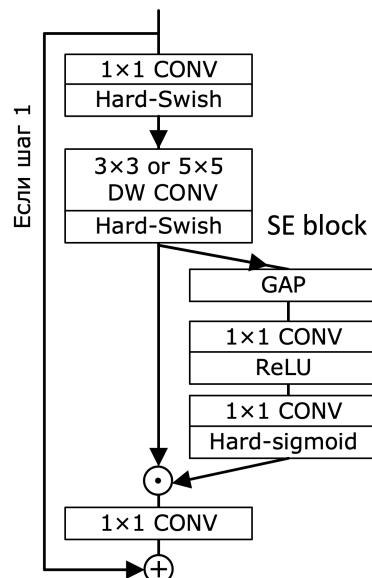


Рисунок 6.56 – Иллюстрация блока MBConv

Авторами было предложено несколько вариантов архитектур с разным числом параметров.

Параметры архитектуры MobileNet V3 были выбраны автотоиском NAS как оптимальные для некоторого семейства устройств и для набора данных ImageNet. Однако гарантии что на других устройства сеть будет работать также хорошо нет.

Архитектура EfficientNet. В 2019-2020 годах исследователи из Google дополнили метод NAS и предложили семейства архитектур **EfficientNet V1** (2019) [201] и **EfficientNet V2** (2020) [202]. В основе предложенного подхода лежала идея поиска оптимальной базовой архитектуры методом PNAS (baseline) и ее масштабирование по трем параметрам - т.н. compound scaling method. Общая структура сети определялась методом PNAS, масштабированию подлежали входное разрешение, число слоев каждого блока сети (глубина) и число карт признаков в каждом блоке (ширина). При этом масштаб изменялся с одинаковым коэффициентом для всех трех параметров [201]. Семейства архитектур EfficientNet V1/V2 стали одними из основных архитектур для решения практических задач компьютерного зрения в 2020-2021 годах. Помимо официальных версий EfficientNet в литературе предложены варианты архитектур для специальных задач, например для использования в мобильных телефонах [173]. Архитектура EfficientNet стала практически универсальной. Параметры EfficientNet также выбирались методами автотоиска архитектур. При этом общая структура фиксировалась, а базовый блок имел слегка модифицированный вид блока MobileNet V3 – так, как это показано на рисунке 6.57. Такой блок было предложено назвать MBConv6.

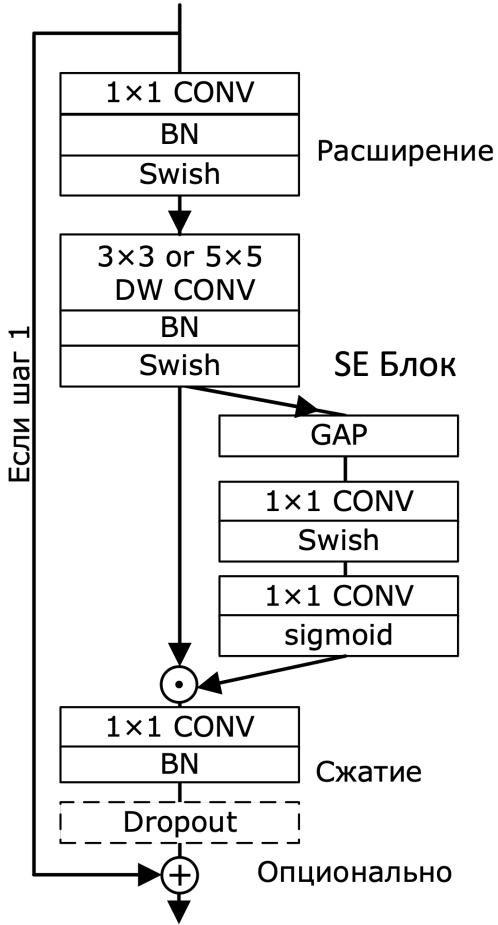


Рисунок 6.57 – Иллюстрация блока MBConv6 (блок EfficientNet)

Для разработки семейства архитектур EfficientNet авторы провели поиск в два этапа. На первом этапе фиксировалась общая структура сети и выбирались ее параметры:

- параметр ширины (Network width), $C \sim \alpha$ - число каналов в свертках;
- параметр глубины (Network depth, $L \sim \rho$) - число слоев в блоке.
- параметр разрешение (Input resolution, $\gamma \sim H \times W$) – размер входного изображения. Обычно, размер 224×224 , но чем больше параметров, Тем больше разрешение и тем выше может быть точность так как больше будет слоев и receptive field.

Таким образом число блоков остается прежним, но каждый блок может быть разной глубины, ширины и иметь разный входной размер.

Благодаря подбору методом NAS была предложена архитектура EfficientNet B0. Структура данной сети представала на данном слайде. Следующие архитектуры семейства EfficientNet выбирались методом compound scaling. Суть этого метода сводится к фиксированию базовых значений гиперпараметров α , ρ и γ , а затем их совместному масштабированию при заданных условиях. Цель метода compound scaling заключалась в поиске баланса точность - число операций (FLOPS). При этом использовалось два критерия:

- Первый критерий: каждый раз FLOPS меняется как 2^ϕ - то есть каждый раз вычислительная сложность повышается в двое.

- Второй критерий: $\alpha \cdot \rho^2 \cdot \gamma^2 \approx 2$ - общие изменения фиксируются "сверху". При этом операция возвведения в квадрат, соответствует тому, что параметр ширины слоя меняет вычислительную сложность в корень раз по сравнению с глубиной и разрешением.

Таким образом было предложено целое семейство архитектур EfficientNet включающих варианты от B0 до B7. Особенности этих архитектур приведены в таблице 6.1.

Table 6.1 – Семейство моделей

Модель	Ширина α	Глубина ρ	Разрешение	DropOut rate
B0	1.0	1.0	224	0.2
B1	1.0	1.1	240	0.2
B2	1.1	1.2	260	0.3
B3	1.2	1.4	300	0.3
B4	1.4	1.8	380	0.4
B5	1.6	2.2	456	0.4
B6	1.8	2.6	528	0.5
B7	2	3.1	600	0.5

Отметим, что мы выделяем такое подробное описание EfficientNet, потому что эта архитектура стала одной из наиболее популярных для решения задач компьютерного зрения в настоящее время. При этом EfficientNet одна из немногих архитектур, которая вышла за рамки академического сообщества и используется в независимых разработках на уровне с mobile net v2, resnet и некоторыми другими. Кроме официальных вариантов EfficientNet, которые приведены на данном слайде в литературе предложены и некоторые другие варианты, как с меньшим числом параметров, так и с большим.

Позже авторами EfficientNet была представлена архитектура EfficientNet V2 [202]. Основной особенностью которой стало обнаружение эффективности замены в начальных слоях глубокой разделенной свертки на аналог обычной свертки 3. В ходе NAS оказалось, что такой подход лучше работает с ускорителями в силу технических причин. Подход было предложено называть fused convolution. Также авторы предложили подход progressive learning который заключается в том, что в ходе предобучения увеличивается размер входных изображений и ряд параметров архитектуры. Предполагается, что для на ранних эпохах обучения сети проще выделить признаки на изображениях меньшего размера. А на поздних эпохах обучения для сети будет полезным более пристальное внимание к деталям - поэтому размеры изображения увеличиваются. Также были и предложены и другие приемы для обучения [202]:

- с ростом эпохи увеличивается размер входного изображения;
- увеличивается вероятность dropout;
- число и виды аугментации;
- интенсивность mixup – (наложение изображений и меток);

- с размером изображения меняется размер батча;
- использован stochastic depth.

Предположено, что большая регуляризация на небольших изображениях вызывает недообучение и наоборот, недостаточно большая регуляризация на больших изображениях вызывает переобучение. Описанные приемы позволили увеличить эффективность работы сети. Таким образом предложены варианты EfficientNet V2: S,M,L,XL. Иллюстрации блоков архитектур EfficientNet и EfficientNet V2 приведены на рисунках 6.58

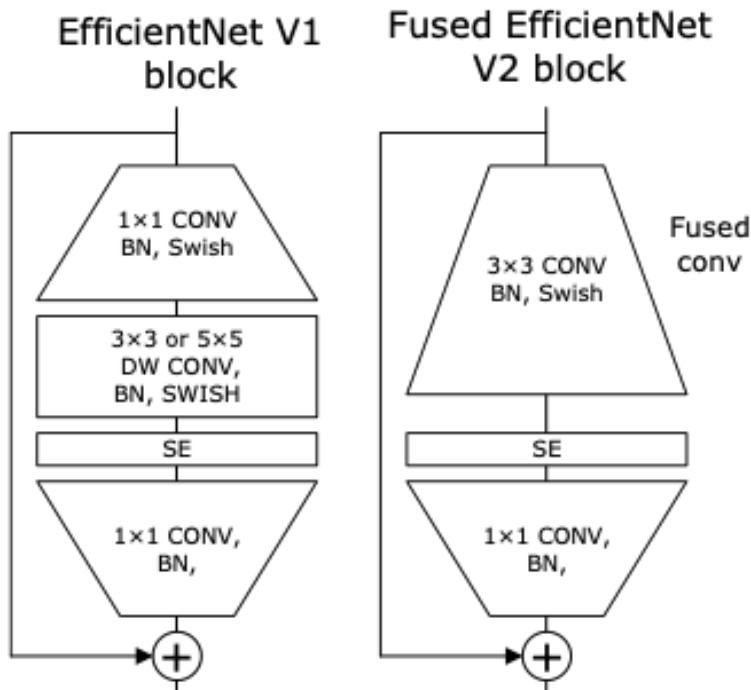


Рисунок 6.58 – Иллюстрации блоков архитектур EfficientNet и EfficientNet V2

На сегодняшний день подход EfficientNet наверно является апогеем метода NAS в применении к сверточным нейронным сетям. Авторам удалось предложить более не менее универсальную архитектуру с высокой степенью оптимальности работы.

Поиск архитектур RegNet. В работе [203] авторы предложили подход побора наилучшего пространства поиска для решения широкого круга задач. Подход является пересмотрением как идей NAS, так и классического ручного поиска архитектур. При этом преимуществом второго подхода авторы называют возможность интерпретации архитектуры и большую универсальность. Тогда как Nas дает более точные результаты в заданных условиях. В работе [203] авторы предложили искать осуществлять поиск наилучшего "пространства поиска" (design space) архитектур. Исходя из общего пространства, авторы предложили ряд специфичных архитектур для решения отдельных задач. Такие архитектуры было предложено назвать RegNet. Поиск пространства (design space) осуществляется итеративно путем создания некоторых популяций. В каждой

такой популяции концентрация эффективных архитектур должна увеличиваться. Отбор архитектур должен производиться на основании вручную заданных правил.

Подход начинается в относительно широкого пространства поиска, которое называется AnyNet. В таком подходе предложено выделить 3 стадии: поток (stream); тело (body) и головную часть (head). Поток это сверточный слой с ядром размера 3×3 и шагом 2, выдающий 32 канала на выходе. Головная часть это GAP и полносвязный слой. Часть тело состоит из т.н. ступеней (stages), которые прогрессивно снижают разрешение. Каждая ступень это набор блоков, где первый блок имеет также шаг 2, а остальные блоки сохраняют размер каждой карты и полностью идентичны. Таким образом В качестве базового блока предложено использовать блок с остаточной связью. Иллюстрация архитектур таких блоков приведена на рисунке 6.59. Такие блоки предложено называть ResNetX. Отметим, что технически блок может соответствовать как блоку ResNet, так и его модификациям ResNeXt и Xception. Помимо блока ResNetX авторы также предложили блок ResNetY, включающий также SE слой.

Для каждого блока подразумевается, что ширина блоков, их число (глубина) и число групп в блоке не ограничены. Однако авторы также предложили выбирать значения этих параметров из некоторого априори заданной квантованной линейного функционала. общее число вариантов для каждого параметра называется степенью свободы. В каждой итерации число степеней сокращается.

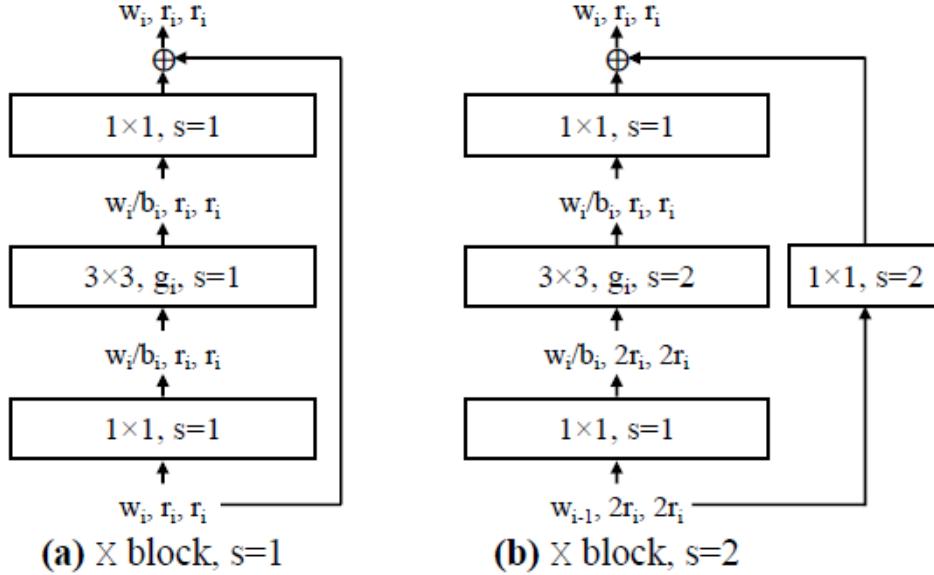


Figure 4. The **X block** is based on the standard residual bottleneck block with group convolution [31]. (a) Each X block consists of a 1×1 conv, a 3×3 group conv, and a final 1×1 conv, where the 1×1 convs alter the channel width. BatchNorm [12] and ReLU follow each conv. The block has 3 parameters: the width w_i , bottleneck ratio b_i , and group width g_i . (b) The stride-two ($s = 2$) version.

Рисунок 6.59 – Иллюстрация блоков resnetx

Из общего пространства AnyNet методом т.н. "human-in-the-loop" получается сужение до пространства которое предположительно содержит только простые (регулярные) архитектуры. Такое пространство предложено назвать RegNet. Предполагается, что архитектуры RegNet должны быть интерпретируемыми и в основном эффективные.

Пространство RegNet получается итеративные путем т.н. семплирования моделей и проверки их распределения ошибок. Для этого пространство моделей обучается на наборе данных ImageNet с ограниченным числом эпох. Важной особенностью тут является то, что отбираются не отдельные модели, а подпространства, заданные своими функционалами. То есть например, авторы приходят к выводу, что для ширины блоков подходит выражение $w_j = w_0 w_m^{\lfloor s_j \rfloor}$ где w_0 начальная ширина, w_m , коэффициент, например 2, $\lfloor s_j \rfloor$ шаг свертки. На основании выбранного пространства RegNet авторы предложили несколько модификаций: мобильные архитектуры REGNETX-600MF, REGNETY-600MF, эффективные архитектуры REGNETX-12GF (в сравнении с ResNeXt) и высокоточные архитектуры в сравнении с EfficientNet) REGNETY-400MF - REGNETY-8.0GF.

6.4 Архитектуры типа трансформер в задачах компьютерного зрения

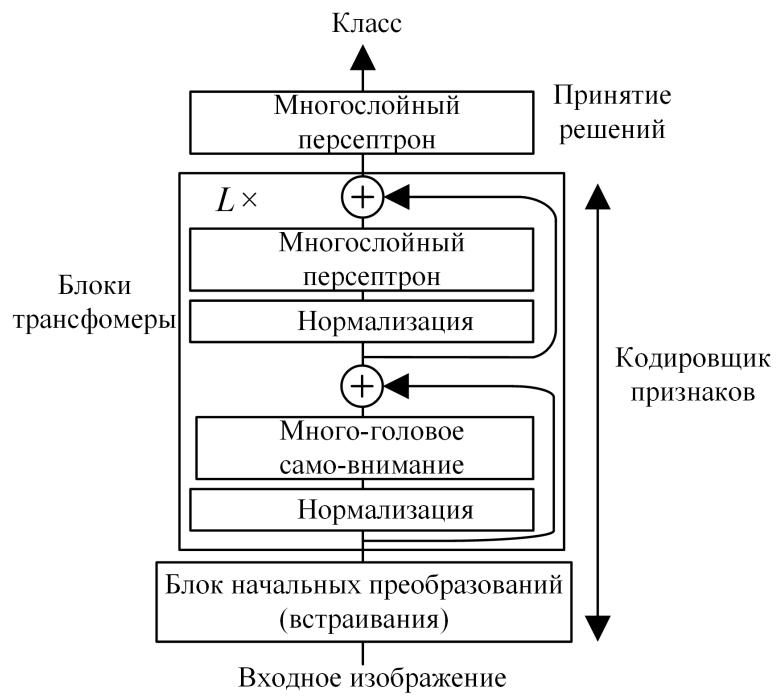
Архитектуры ViT. Как уже было сказано ранее в 2017 году различными исследователями было предложено использование идеи слоев внимания в задачах компьютерного зрения. Также в 2017 году А. Васвани был предложен блок, названный трансформер (transformer) для задач обработки естественного языка [184]. В составе данного блока использовались только слои внимания и полно связные слои. В 2018 году было предложено использовать блока трансформер в задачах генерации изображений и обработки видео. В 2020 году были предложены первые успешные архитектуры для задачи классификации изображений [204]. Многими авторами предлагалось замена части слоев сверточных сетей на блоки трансформеры (полносвязные блоки с вниманием (такие же, как Васвани)) [205].

В 2020 году была предложена архитектура visual transformer (ViT), в которой авторы использовали только блоки трансформеры и смогли достичь точности на наборе ImageNet сопоставимой со сверточными сетями. По существу, архитектура ViT являлась переосмыслением архитектуры BERT для обработки естественных языков (2018 г. [206]). Иллюстрация архитектуры ViT показана на рисунке 6.60.

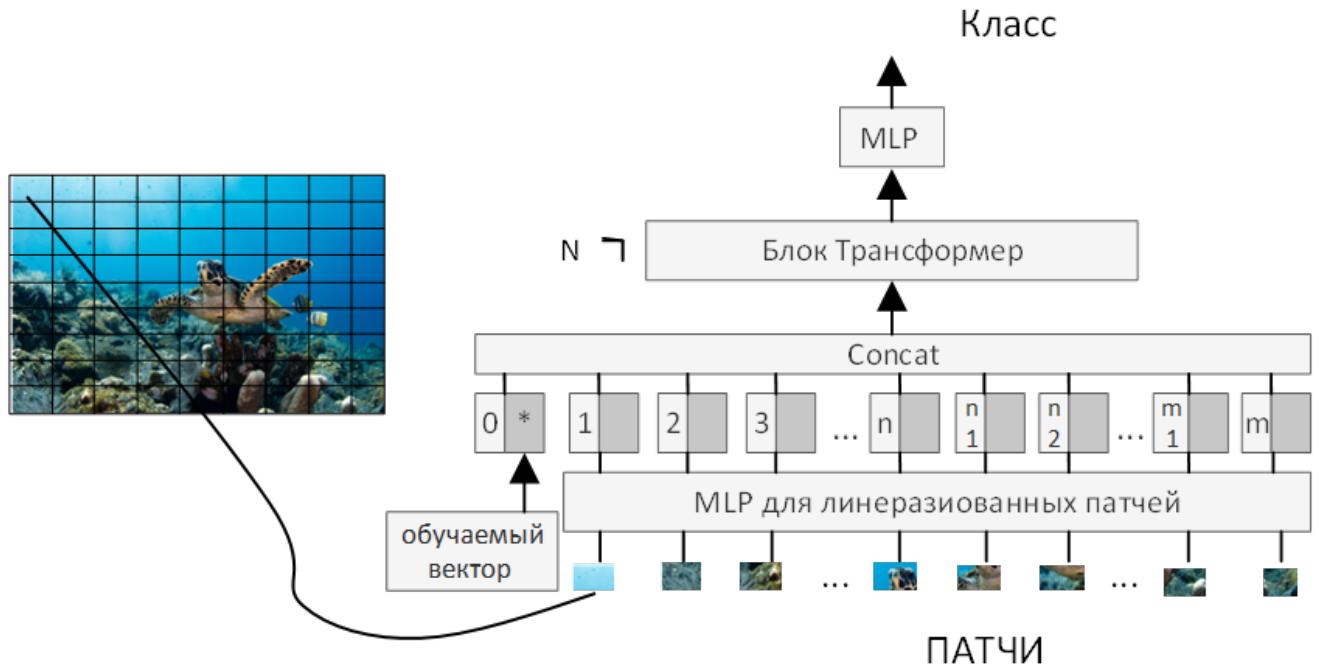
Архитектура ViT состоит из блока начальных преобразований; набора слоев трансформеров и блок принятия решений на основе многослойного персептрона. Вход для преобразователя изображений, делится на набор т.н. патчей - участков изображения. Базовый блок трансформатора состоит из слоя многоголовочного слоя самовнимания [184] с использованием операции нормализация слоя [143] и набора полно связных слоев также с нормализацией слоя. Таким образом архитектура ViT не имеет сверточных слоев. Более подробная иллюстрация ViT приведена на рисунке 6.61. Иллюстрация показана для архитектуры ViT-L с 16 каналами многоголового само-внимания.

Архитектура ViT стала первой доказавшей, что решение задач компьютерного зрения возможно по средствам полно связных нейронных сетей. Данная модель показала точность 88,5% по методу top-1 на наборе данных ImageNet. Однако, для достижения высокой точности модель ViT была предобучена на наборе данных JFT-300M (набор с 300 млн. изображений [207]) [208]. Отмечается, что в случае предобучения на небольших наборах данных (менее 14 млн изображений) точность модели ViT значительно снижается (примерно до 75%). Также число параметров ViT в 10 раз выше чем для сверточной сети (600 тыс. параметров). Архитектура ViT стала одним из основных этапов в принятии трансформеров сообществом исследователей систем компьютерного зрения [209]. Вариант совершенствования архитектур ViT появляются и по настоящее время. Так, например в июне 2021 г была представлена модель ViT-G, показавшая точность 90,5% по методу top-1 на наборе данных ImageNet и имеющая порядка 2-х миллиардов параметров [210].

Другие архитектуры трансформеры. Главными достоинствами архитектур трансформеров являются следующие [209, 204].



(a) принцип работы трансформеров



(b) Иллюстрация работы блока трансформера

Рисунок 6.60 – Иллюстрация архитектуры ViT

- Возможность построения сложных зависимостей между признаками локальными участками изображения независимо от их взаимного расположения. Более того, было замечено, что блоки-трансформеры могут легко запоминать глобальные особенности изображений и трудно запоминать локальные и относительно небольшие особенности.

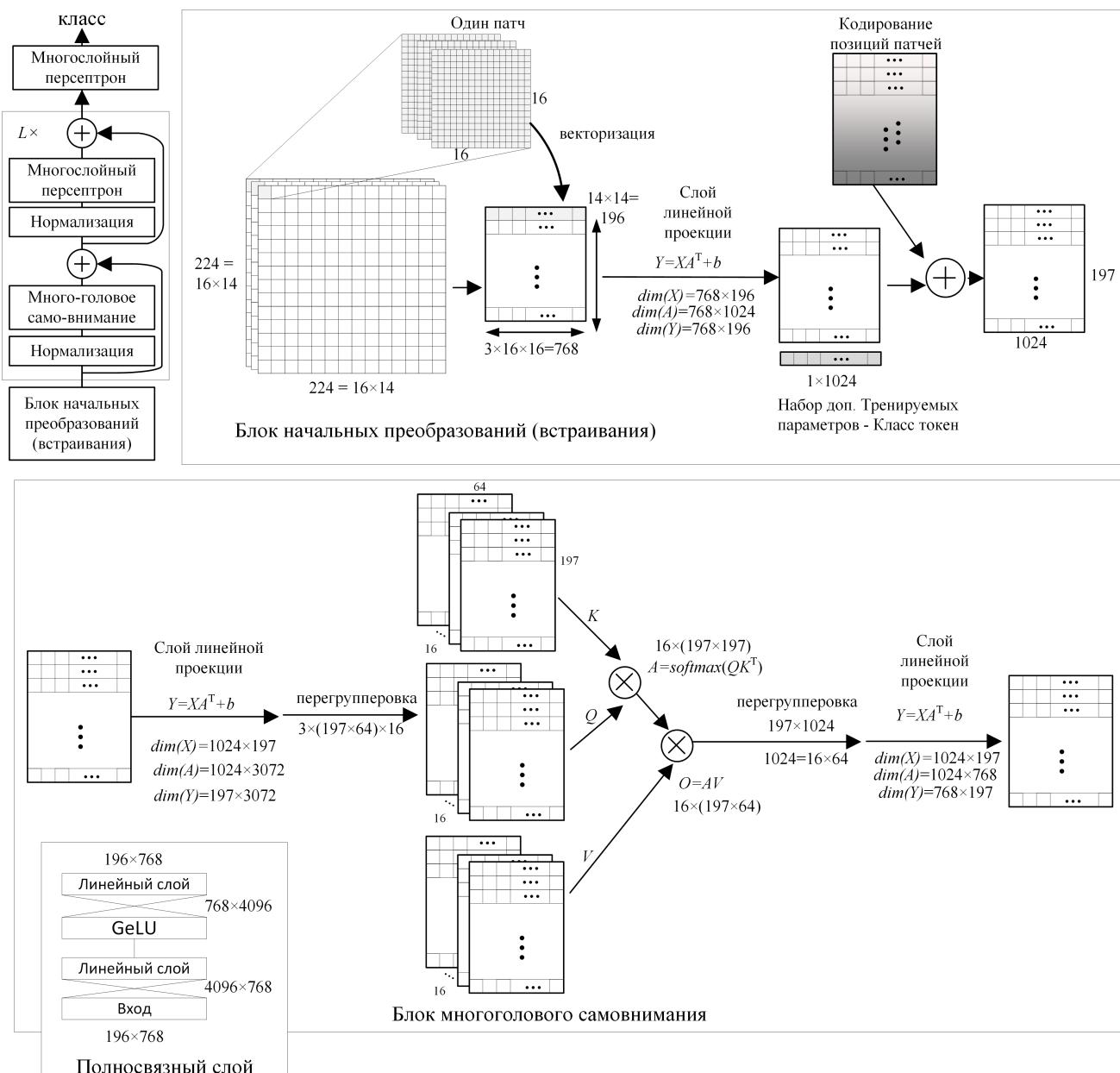


Рисунок 6.61 – Подробная иллюстрация архитектуры ViT-L

- Трансформеры позволяют обучать модели со значительно увеличенным числом параметров. Что также дает преимущества при предобучении на больших наборах данных, таких как JFT-300M. В предыдущие эпохи (до 2015 гг) данное обстоятельство скорее было недостатком. В настоящее время в силу развития вычислительной техники и возможностей сети Интернет имеется возможность сравнительно быстрой работы с большими объемами данных. При этом путь повышения точности и обобщающей способности за счет предобучения является предпочтительным.

- Возможность для некоторых архитектур предобучения на не размеченных данных. Ряд архитектур трансформеров имеют возможность предобучаться подобно жадному предобучению о котором говорилось выше. Такая возможность особенно важна в силу сказанного в предыдущем пункте. Также отмечается [211, 212], что предобучения баз разметки потенциально способствует повышению выразительной силы (сложности признаков) и обобщающей способности нейронной сети. Метод предобучения на не размеченных данных принято называть **self-supervised learning** (самообучение).

Итак, недостатками архитектур трансформеров типа ViT являются:

- часто необходимость обучения на очень больших наборах данных, в том числе необходимость предобучения,
- большое число параметров архитектуры,
- необходимость долгого обучения для выделения локальных признаков.

В 2020 и по настоящее время предпринимаются попытки модификации архитектур трансформеров, которые призваны решить эти проблемы. Основными направлениями решения соответствующих проблем являются

- поиск оптимальных архитектур слоев трансформеров,
- поиск оптимальных комбинаций сверточных слоев и слоев трансформеров,
- исследование необходимости операций внимания в полносвязных слоях.

В современной литературе часть предложенных архитектур- трансформеров полностью полносвязные, часть имеют и сверточные и полносвязные слои. Например в августе 2021 года была представлена архитектура BotNet. Авторы BotNet предложили заменить часть слоев архитектуры ResNet на слои многоголового само-внимания и при этом получили небольшой прирост точности (0,5%) по сравнению с оригинальной архитектурой, однако, они увеличили число параметров[213]. Таким образом вопрос о том, какой подход (или комбинация подходов) - лучше остается открытым. В работе [214] посвященной сверточных сетей и трансформеров приходят к выводу, что трансформеры быстрее и более точно выучивают крупно-размерные признаки, чем сверточные сети. Однако, архитектурам-трансформерам требуется больше примеров для мелко-размерных признаков.

Итак современные решения архитектур трансформеров включают следующее.

- Использование метаобучения путем дистилляции знаний для конкретных задач (DeiT) [215]. В архитектуре DeiT, число параметров было снижено до 80 тыс. и при этом модель показала точность 83,5% по методу top-1 на наборе данных ImageNet с использованием только данных ImageNet (без предобучения на JFT-300M).
- В конце июня 2021 года была также представлена архитектура Vision Outlouker (VOLO) показавшая точность 85,5% в условиях аналогичных DeiT, однако имеющая большие параметров [216].

- Вычисление информации о внимании только для некоторых окон, а не для всех путей; окно может перемещаться между слоями (SWIN) [46].
- Последовательное совмещение сверточных и трансформирующих слоев (CoAtNet) [217].
- Совмещение в каждом блоке сверточных и преобразовательных операций (MaxViT) [218].
- Поиск наиболее эффективных с вычислительной точки зрения модификаций трансформаторного слоя (MobileViT) [219]. По задумке авторов каждый MobileViT блок должен выделять как локальные, так и глобальные признаки. При этом цель авторов была снизить необходимое число параметров. Для выделения локальной информации блок использует стандартную квадратную свертку заданной размерности, для выделения глобальной информации результат локальной свертки увеличивает число каналов при помощи точечной свертки, для этого результата используется блок трансформер. полученные карты признаков разделяют на патчи и векторизуют, затем проходят через слой многоголового внимания и полно связанный слой и собираются обратного в набор двухмерных карт признаков. Число карт признаков сокращается обратно при помощи одномерной свертки.
- Стратегии метаобучения, *t.e.* стратегии обучения сетей глубокого обучения. Такие стратегии могут включать перегонку знаний из крупных сетей в малые; повышение производительности целевых сетей путем добавления меток к дополнительным сетям [220]; с помощью cross data augmentation (аугментации меток и данных) [221].

Архитектура ConvNeXt В 2020 году авторами работы [222] был проведен анализ блока ResNet в рамках текущих возможностей по обучению нейронных сетей. В том числе возможностей для предобучения. Авторы предложили модификацию блока, иллюстрация которой показана на рисунке 6.62.

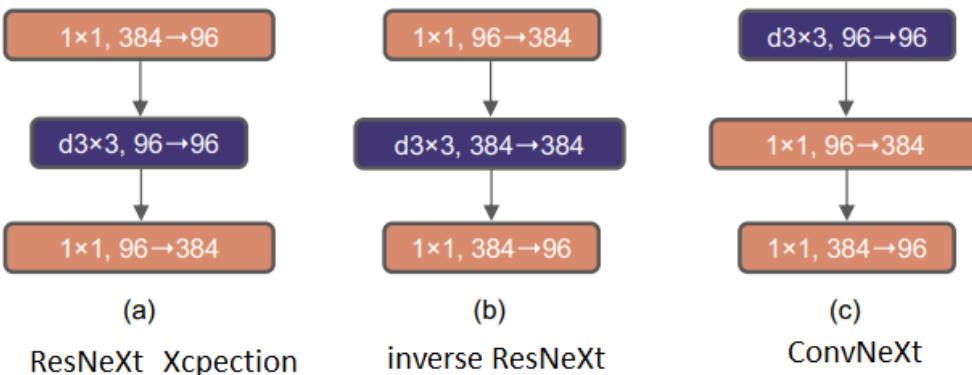


Рисунок 6.62 – Иллюстрация блока ConvNext (c) и его аналогов (a,b)

В работе [222] авторы использовали ряд техник, вдохновлённых работами DeiT и Swin и ViT. По существу работа [222] сводится к тщательному исследованию возможностей использования приемов соответствующих архитектурам трансформерами в сверточных архитектурах. В результате авторы предлагают новую сверточную архитектуру, превосходящую по точности архитектуры трансформеры на стандартных тестах. Среди проанализированных техник выделим следующие.

- Обучение методом AdamW (с параметром weight decay) на более чем 300 эпохах.
- Аугментация при помощи техник mixup, CutMix, RandAugment, Random Erasing.
- использование техник обучения со стохастической глубиной (Stochastic Depth) и сглаживанием меток (Label Smoothing).
- Построение макроархитектуры на принципах предложенных в работе SWIN. Вместо структуры этапов архитектуры ResNet-50 (3, 4, 6, 3) блоков использована более экономная структура (3, 3, 9, 3), характерная для архитектуры Swin-T.
- Замена начального блока (stem) ResNet (свертка 7×7 , шаг 2 + max pooling (шаг 2)) на блок подобный трансформам ViT. Такой блок назван “patchify stem”. Входное изображение разделяется на непересекающиеся ячейки (патчи) размером 4×4 . Для этого можно использовать свертку размера патча (ядро размером 4) с эквивалентным шагом 4.
- Использование принципа ResNeXt, сформулированного авторами как увеличение ширины слоя за счет использования групп. Однако авторы также предлагают использовать принцип MobileNet (обратный слой узкого горла) и глубокую разделенную свертку как MobileNet и Xception архитектурах. При этом отмечается, что глубокая разделенная свертка эквивалентна операции self-attention блока типа взвешенное суммирование. Использование свертки 1×1 эквивалентно пространственному разделению и перемешиванию каналов. Комбинация этих операций по мнению авторов аналогична тому, какой эффект производят блоки архитектур трансформеров. Аналогично работе ResNeXt авторы также предложили использовать 96 (вместо 64) каналов в слоях.
- Использования подхода обратного (инвертированного) узкого горла (inverted bottleneck) позволило авторам понизить вычислительную сложность при небольшом увеличении точности.
- Экспериментально авторы установили, что в блоках оптимально использовать свертки с размером 7×7 . Отмечается, что этот путь вдохновлен тем, что Swin использует оконное внимание аналогичных размеров.

- Также авторы предложили поменять операции глубокой части свертки и точечной свертки. Отмечается, что эта идея также вдохновлена архитектурами трансформерами, где блок внимания идет прежде полно связного блока. В ходе экспериментов авторы установили, что такой прием понижает вычислительную сложность модели, но слегка снижает точность.
- Сверточные слои ConvNeXt используются с функцией активации GELU вместо ReLU аналогично тому, как это делается в архитектурах трансформеров. Однако, в экспериментах авторы отметили, что точность от этого приема самое по себе не меняется.
- При этом авторы предложили использовать функцию активации только между двумя точечными свертками следуя стилю блоков трансформеров. Этот прием слегка улучшил точность работы модели.
- Блоки батч-нормализации предложено оставлять перед точечными свертками (удалены между и после них). Это также немногого увеличивает точность архитектуры.
- Также авторами показано, что в их блоке можно заменять слои батч-нормализации на нормализацию слоя без потери точности. Однако авторы отмечают что в обычных архитектурах ResNet это может привести к потере точности.
- Для понижения размерности авторы следует стратегии архитектуры Swin. Вместо операций макс-пулинга используются свертки с ядром 2×2 и шагом 2. Однако, отмечается, что в прямую этот подход приводит к нестабильности результатов обучения. Для компенсации нестабильности перед слоем понижения размерности авторы добавляю слой нормализации слоя. Также в силу этого приема авторы добавляют слой нормализации после начального блока (stem) и после глобального усредняющего пулинга в конце. Такой прием повышает точность и стабилизирует результаты обучения модели.

Предложенный в работе [222] набор техник позволил существенно повысить точность сверточных архитектур и возобновил дискуссию о необходимости использования блоков трансформеров в целом. Авторами были предложены несколько вариантов архитектуры ConvNeXt, показавшие лучшие результаты как в задачах классификации изображений, так и в других задачах компьютерного зрения в качестве кодировщиков признаков.

Архитектуры-Миксеры. В 2021 году был представлен ряд трансформеров-подобных полно связных архитектур для решения задач компьютерного зрения. В таких архитектурах соли многоголового само-внимания был заменены на наборы операций с полно связными слоями [223]. Наиболее известная из таких архитектур MLP-Mixer, предложенная командой Google Brain. Архитектура показала точность (87,9% по методу top-1 на наборе данных ImageNet) с предобучением на наборе данных JFT-300M, показав при этом производительность в 2,5 раза выше, чем архитектура ViT [224]. На данный момент полностью полно связные архитектуры не дают высокой точности, сопоставимой с

трансформерами, однако потенциально способных их заменить. Основным преимуществом таких архитектур является пониженное число параметров (из за отсутствия многоголового само-внимания) и высокая скорость работы (операции с полно связанными слоями - самые быстрые). Иллюстрация архитектуры Миксера приведена на рисунке 6.63.

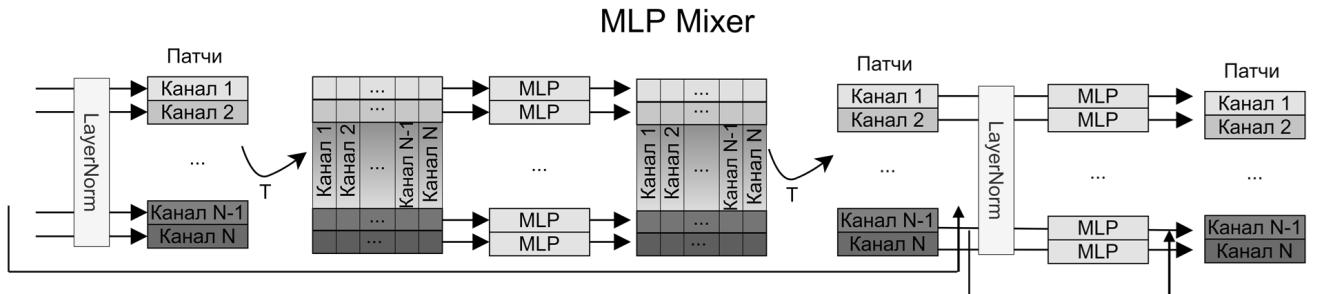


Рисунок 6.63 – Иллюстрация блоков архитектуры MLP-Mixer

Среди архитектур миксеров также можно выделить следующие работы.

- В работе [46] показано, что замена слоев внимания на блоки MLP-Mixer в архитектуре SWIN снижает точность лишь на 2% при понижении вычислительной сложности на 25 %. Предложенный вариант авторы называли SWIN-Mixer.
- В работе [225] авторы модифицировали блок MLP-Mixer, предложили замену слоя нормализации на оригинальный вариант аффинного преобразования, а также набор техник, позволяющих достичь точности архитектур семейства EfficientNet. Полученный блок было предложено назвать ResMLP.
- В работе [226] авторы предположили, что не так важен тип нелинейной операции в блоке типа трансформера, как важен факт такой операции. Поэтому было предложено использовать операцию пулинга. Полученный блок оказался несколько точнее блока ResMLP при чуть большей скорости работы. Полученный блок было предложено назвать PoolFormer.
- В работе [227] авторы предложили использование приема репараметризации [203] для упрощения архитектуры MLP. Были рассмотрены несколько вариантов архитектур. В том числе варианты со свертками. Во всех случаях авторы смогли достичь точности сопоставимой с архитектурой ResNet при сравнимом числе параметров во время работы.
- В работе [228] предложено использование т.н. глобальных фильтров в частотной области вместо слоя трансформера.
- В работе [229] предложено использовать вместо внимания фильтрацию в областях пространственных частот и межканальных частот.

Исследования в данном направлении только набирают обороты [223, 230, 231].

7 Задача семантической сегментации

7.1 Базовые вопросы семантической сегментации.

Особенности задачи семантической сегментации. Как уже обсуждалось предмет компьютерное зрение подразумевает ряд задач. Среди всех этих задач классификация изображений имеет наиболее фундаментальное значение для предмета глубокое обучение в компьютерном зрении. С задачи классификации началась архитектура LeNet. Однако, в целом компьютерное зрение включает также такие задачи как семантическая сегментация, обнаружение объектов и некоторые другие. Иллюстрация, позволяющая сравнить этих задачи приведена на рисунке 7.64.

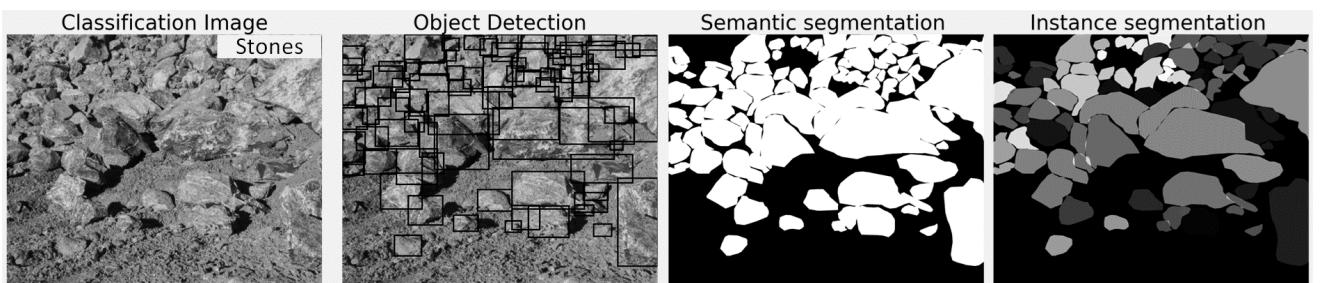


Рисунок 7.64 – Наиболее распространенные задачи компьютерного зрения

В этом разделе будет рассмотрена задача семантической сегментации. Постановка проблемы в этой технике: поиск областей изображений соответствующих тому или иному классу. Задача семантической сегментации известна в приложениях компьютерного зрения достаточно давно [232, 233]. Однако, первые успешные попытки ее решения при помощи нейронных сетей были опубликованы в 2012 году в работе C.Farabet и соавторов [234]. В частности, в данной работе был предложен пиксельной классификации - то есть присвоение класса каждому пикселю входного изображения. То есть результатом работы нейронной сети должен стать набор карт признаков - по одной для каждого класса. При этом каждая карта признаков должна выделить всю площадь (или контур) объектов данного класса. Для повышения надежности сегментации объектов разных масштабов в работе предлагалось использовать несколько сверточных энкодеров признаков и затем объединить полученные карты. Однако, архитектура, предложенная в [234] не получила широкого распространения. Таким образом, в рамках этой и последующих архитектур глубокого обучения нейронных сетей задача семантической классификации может быть рассмотрена как расширение идеи классификации, но классификации по каждому пикселю выходной карты признаков. Иллюстрация соответствующих примеров приведена на рисунке 7.65.

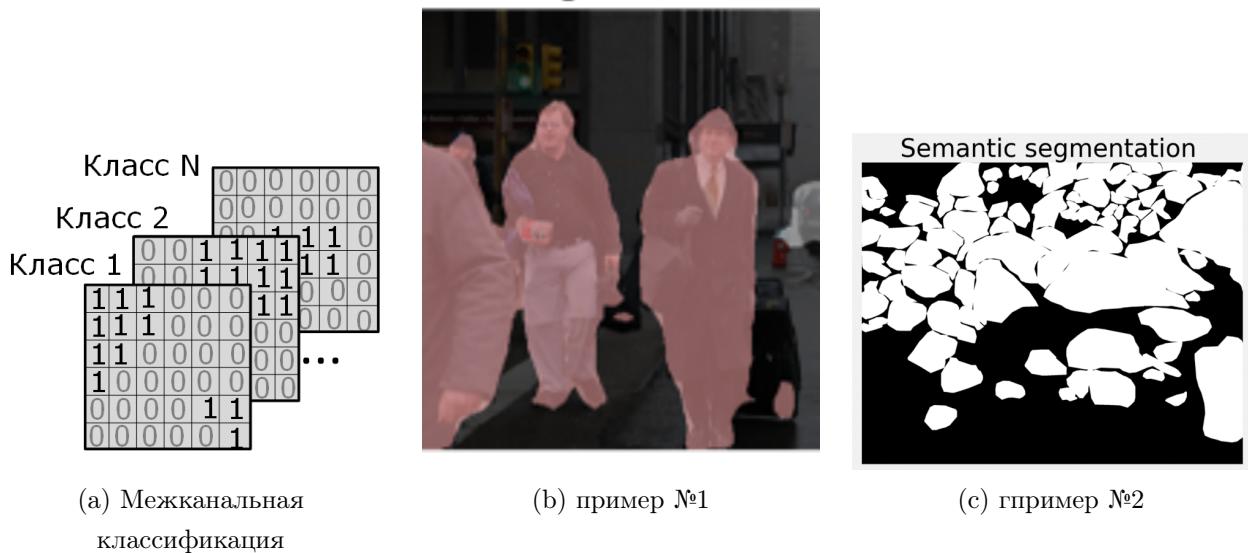


Рисунок 7.65 – Иллюстрация принципа семантической сегментации.

Фактически семантическая сегментация отличается от классификации только главной частью. Однако тут особенностью является выбор функции потерь. Так как задача предполагает что каждый класс может занимать лишь небольшую часть изображения мы почти всегда имеем дело с несбалансированной классификацией. В отношении рис. 7.65 (а) отметим, что маска для семантической сегментации имеет смысл аналогичный двухмерному one-hot кодированию.

Итак, особенностями задачи семантической сегментации являются:

- Архитектура модели семантической сегментации представляет собой комбинацию кодировщика признаков (или его блоков), который может быть взят из задачи классификации и новой части принятия решений.
- Могут быть выделены несколько подходов к построению общей архитектуры семантической сегментации:
 - на основе замены только головной части сети классификации или головной части и последних слоев кодировщика признаков;
 - на основе подхода энокодер-декодер, то есть часть принятия решений будет иметь структуру аналогичную кодировщику признаков.
- Часть принятия решений подразумевает увеличение размеров карт признаков. Такое увеличение может быть скачкообразным при использовании подхода замены головной части. В случае подхода энокодер-декодер увеличение будет более плавным. Скачкообразное увеличение размеров может производиться в несколько этапов.
- Потенциальная несбалансированность классов, например за счет использования объектов разных масштабов. Также фон может занимать достаточно большую площадь изображения.

- Функция потерь может иметь как интерпретацию в области классификации, так и в геометрический смысл.
- В зависимости от постановки задачи могут быть выбраны разные функции классификации.

Особенности функции потерь. При решении задачи семантической сегментации могут быть рассмотрены три типа постановки на уровне функций потерь [235].

- Межканальная кросс энтропия. Функция кросс-энтропии, тут используется между каналами для каждой позиции. пикселей:

$$L = \sum_{w=0, h=0}^{W-1, H-1} \sum_{c=0}^{C-1} y_{cwh} \log \hat{y}_{cwh} \quad (7.31)$$

где L_i - значение функции потерь, W, H, C - значения числа пикселей по горизонтали, вертикали и числа каналов. Также функция может быть записана как

$$L_i = \sum_c \begin{cases} \sum_{w,h} \log \hat{y}_{wh}, & \text{if } y == c \\ 0, & \text{otherwise.} \end{cases} \quad (7.32)$$

Для бинарного случая функция может быть переписана как

$$L = \sum_{w=0, h=0}^{W-1, H-1} y_{wh} \log \hat{y}_{wh} + (1 - y_{wh}) \log (1 - \hat{y}_{wh}) \quad (7.33)$$

- Геометрическая мера сходства. Среди таких мер чаще всего используют меру аналогичную метрики пересечения площадей – intersection over union (IOU). Также часто используется такая мера, как Дайс (DICE) или еще ее называют коэффициент Серенса-Дайса. В некоторых современных системах также используются переосмыслиенные функционалы вида IoU. В некоторых случаях кросс энтропия и мера геометрического сходства используются совместно. Выражения для этих метрик следующие:

$$IoU = \frac{|A \cap B|}{|A \cup B|} DICE = 2 \frac{|A \cap B|}{|A| + |B|}. \quad (7.34)$$

где $|A \cup B|$ - объединение; $|A \cap B|$ - пересечение; $|A| + |B| = |A \cap B| + |A \cup B|$. Соответствующая функция потерь записывается как

$$L = 1 - IoU \text{ or } L = 1 - DICE. \quad (7.35)$$

Иногда IoU записывается как

$$IoU = 2 \frac{|A \cap B| + 1}{|A \cup B| + 1}, \quad (7.36)$$

где слагаемое 1 необходимо для предотвращения ситуации, когда $|A \cup B| = 0$. Важно отметить, что в задачах семантической сегментации функции данного семейства используются после выполнения операций взвешивания (сигмоид, softmax).

- Мера оценки сегментации контуров классов. Такая постановка задачи отличается от первых двух. В этом случае могут быть использованы меры оценок некоторых расстояний, например расстояние Хаусдорфа [236]. Выражение для расстояния может быть следующим:

$$d(X, Y) = \max_{x \in X} (\min_{y \in Y} \|x - y\|^2) \quad (7.37)$$

На практике иногда также используются и другие меры. Однако, как правило, они таки или иначе сводятся к аналогам уже приведенных. В отношении первого из приведенных подходов следует отметить такие его модификации как [235, 237, 238]:

- Сбалансированная кросс-энтропия:

$$L_i = \sum w_y y_i \log \hat{y}_i + (1 - w_y)(1 - y) \log (1 - \hat{y}), \quad (7.38)$$

где $w_y = 1 - \frac{y}{H \cdot W}$. Для бинарного случая:

$$L_i = \sum w_y y_i \log \hat{y} + (1 - w_y)(1 - y) \log (1 - \hat{y}), \quad (7.39)$$

- Взвешенная кросс-энтропия:

$$L_i = \sum w y_i \log \hat{y} + (1 - w)(1 - y) \log (1 - \hat{y}), \quad (7.40)$$

где w - весовой параметр. Параметр может быть выбран так, что, если $w > 1$, то снижается число false negative результатов, если $w < 1$ то снижается число false positive результатов.

- Фокальная функция потерь:

$$L_i = \sum \alpha(1 - \hat{y})^\gamma L_{CE}, \quad (7.41)$$

где L_{CE} - функция потерь кросс энтропии, вычисленная например, как 7.31.

Также отметим, что все приведенные и другие функции потерь могут комбинироваться в той или иной степени для достижения определённых результатов сегментации. Так, часто кросс энтропия комбинируется с геометрическими мерами. Такой подход позволяет с одной стороны достигнуть высокой точности определения принадлежности некоторой области изображения некоторому классу, а с другой стороны более точно восстановить границы и другие небольшие характерные особенности изображений [235]. В некоторых случаях к данным функционалам потерь добавляют и меры оценки качества восстановления изображения в целом, например средний квадрат ошибки или связанные с ним функции структурной схожести и отношения сигнал-шум [239, 240, 241]. Также в ряде случаев

комбинируют меры оценки границ областей и сегментации самих областей. Такой подход может преподавать использование двух головных частей для решения обеих задач. Подход позволяет повысить точность восстановления границ объектов при их сегментации [242].

Методы повышения размерности карт признаков. Выход архитектуры семантической сегментации, как правило, должен иметь такие же размеры, как и входное изображение. Для решения задачи повышения разрешения используются различные подходы. Условно такие подходы можно разделить на 3 группы:

- методы интерполяции изображений.
- методы на основе обратной свертки.
- методы на основе свертки сверх-высокого разрешения.

Также могут быть и другие подходы к повышению разрешения, например при помощи генерации новых изображений [243].

Примеры методов интерполяции приведены на рисунке 7.66. Самый простой метод интерполяции – это во круг каждого пикселя построить рамку из нулей или из значений как у исходного пикселя. Однако, такие подходы не дают хорошего разрешения (box interpolation, bad of nails). Среди методов интерполяции наиболее популярен метод билинейной интерполяции. Это метод аппроксимации новых значений из нескольких имеющихся. Могут также быть более сложные методы аппроксимации, например сплайнами, или низкочастотной фильтрацией. Однако, их вычислительная сложность будет значительно выше. Важно отметить, что в общем случае интерполяция может быть не только повышающей, но и понижающей разрешение.

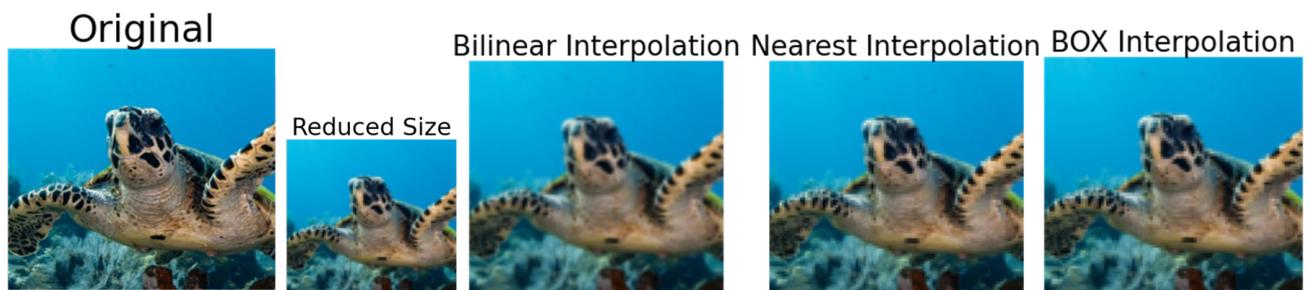
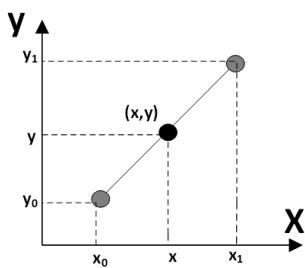


Рисунок 7.66 – Сравнение типов интерполяции

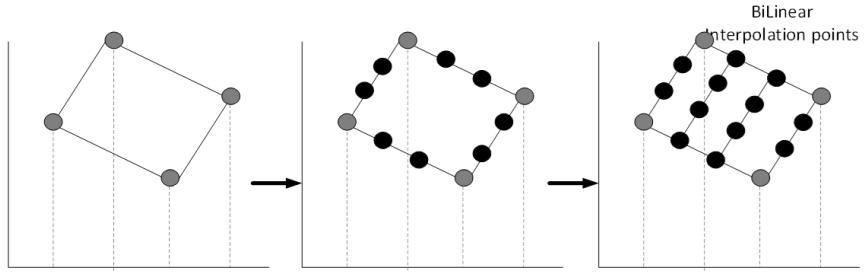
Идея метод билинейной интерполяции приведена на рисунках 7.67. Идея аналогична линейной интерполяции одномерных функций. Напомним, что в методе линейной интерполяции новое значение между двумя имеющимися оценивается как среднее значение на линии, построенной между ними. Для двумерной плоскости можно аналогично взять 4 точки и провести линии между ними. Затем на этих линиях отложить новые точки. Для этих новых точек провести промежуточные линии внутри образовавшегося четырехугольника. Таким образом можно провести интерполяцию в несколько раз. На практике, как правило, считается, что старые значения пикселей будут в углах интерполируемого квадрата [244].

Линейная интерполяция



● Интерполированные точки

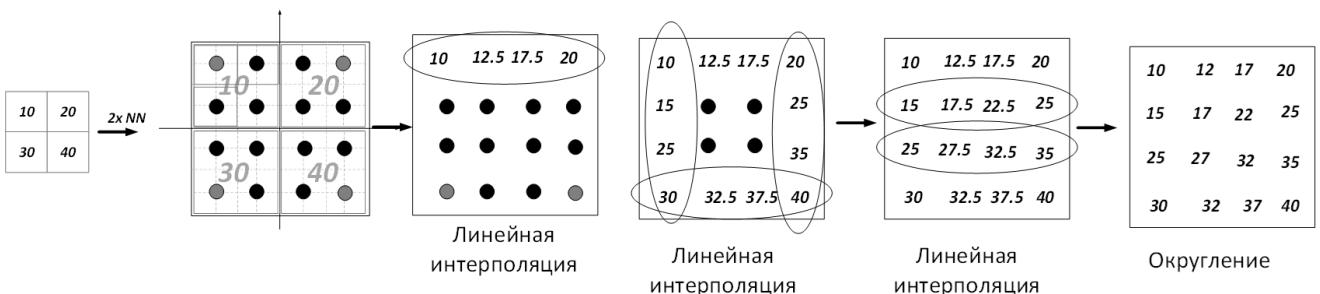
Принцип Билинейной интерполяции



● Точки по которым проводится интерполяция

(a) принцип перехода от линейной к билинейной интерполяции

Принцип Билинейной интерполяции



(b) принцип работы билинейной интерполяции

Рисунок 7.67 – Иллюстрации работы билинейной интерполяции

Другим методом повышения разрешение является использование транспонированной свертки. Также эта операция может быть названа сверткой с дробным шагом [245]. Идея этой свертки была предложена в работе [246]. По существу, свертка, в отличии, от традиционной распространяет каждый элемент воздействия на промежуточную карту признаков. То есть ядро свертки скользит по каждому элементу. Промежуточные карты признаков затем складываются, образуя выходную карту признаков. Также в этой операции паддинг применяется не ко входу, а к выходу. Шаг в транспонированной свертке реализуется как сдвиг результатов распространения входных значений. То есть тоже применяется к выходу. Если провести транспонированную свертку с шагом больше двух, то она даст соответствующее повышение разрешения. Плюс такой свертки – она имеет обучаемые параметры. Отмечается, что транспонированная свертка может быть проинициализирована путем построения ее весовых параметров соответствующих билинейной интерполяции [139]. Иллюстрации работы транспонированной свертки с шагом 1 и с шагом 2 приведены на рисунках 7.68.

Третья группа методов – это метод так называемой свертки супер-разрешения. Принцип такой свертки показан на рисунке 7.69. В своей основе операция использует глубокую свертку. Однако, после операции глубокой свертки результат каждой карты признаков комбинируется с другими по заданному правилу. Такое правило называется перетасовкой пикселей.

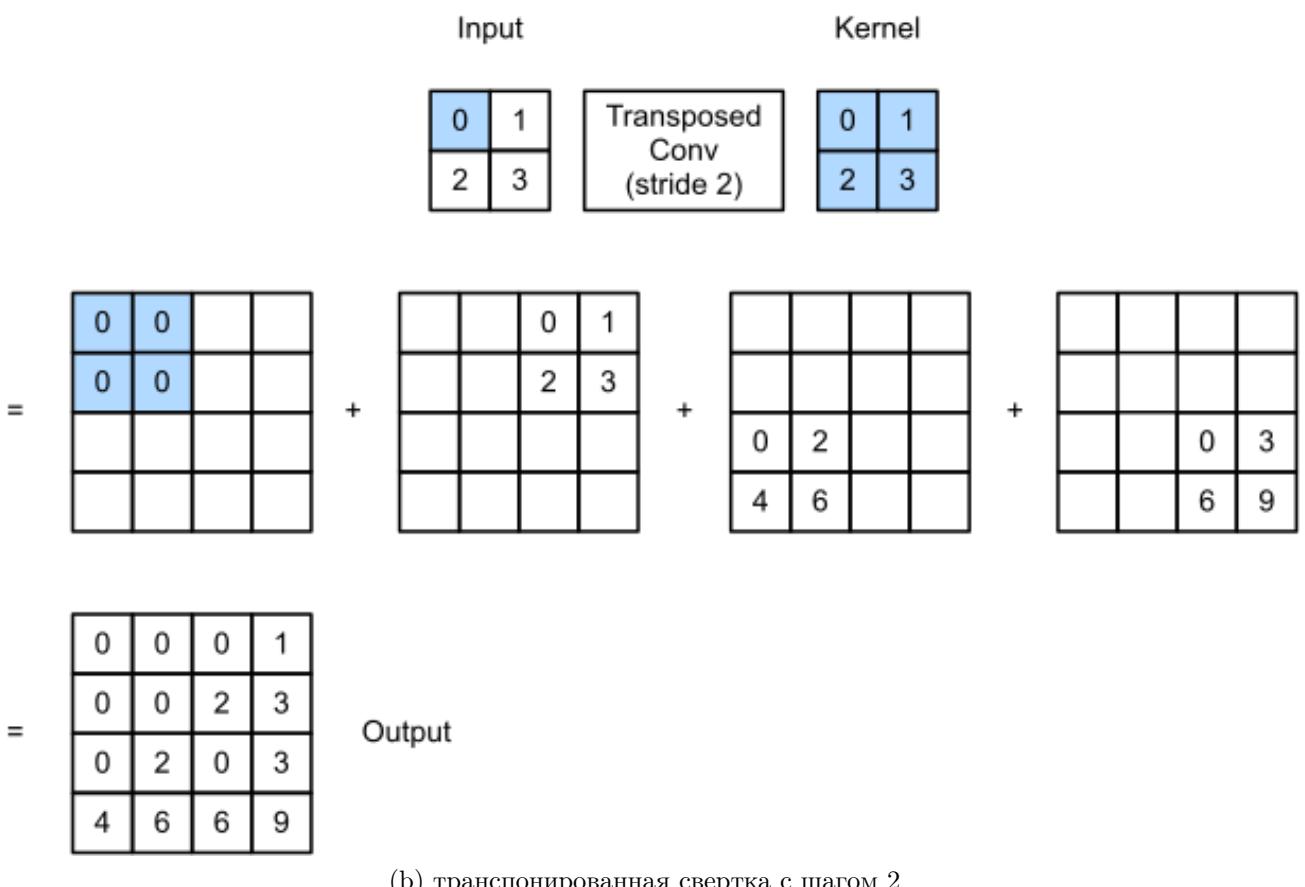
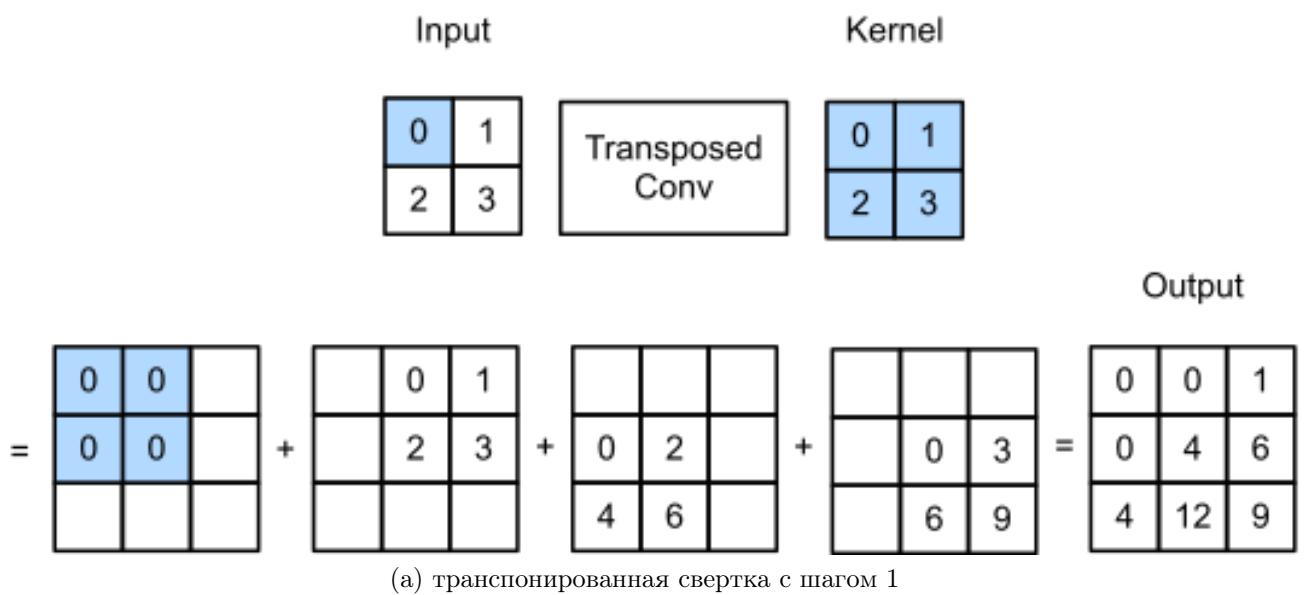


Рисунок 7.68 – Иллюстрации работы операции транспонированная свертка

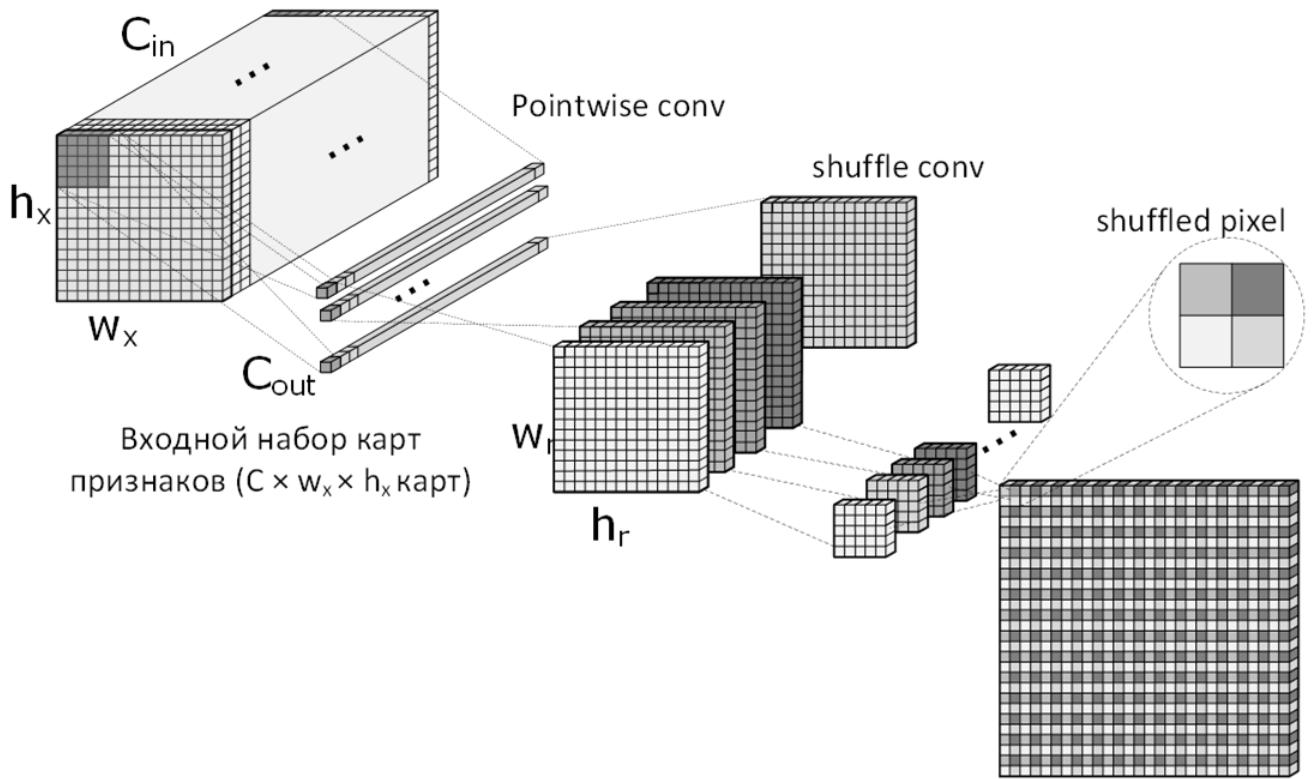


Рисунок 7.69 – Сравнение типов интерполяции

7.2 Подход расширенной головной части

Архитектура FCN. В 2014 году в работе J. Long и соавторов [246] была предложена полностью сверточная архитектура (fully convolutional network (FCN)) сети семантической сегментации. В данной архитектуре авторы модифицировали архитектуры AlexNet и VGGNet. Было предложено заменить последние слои на сверточные слои и после них поставить слой увеличения размерности при помощи операции транспонированной свертки. Отметим, что данная архитектура является полностью сверточной. В том числе в ней для получения итоговых карт признаков по числу выходных классов используется точечная свертка вместо полносвязного слоя. Иллюстрация архитектуры приведена на рисунке 7.70.

Отметим также, что в работе [246] авторы предложили использовать несколько путей для расширения размеров карты признаков (upsampling). Основная идея тут заключалась в том, что объекты разного размера могут иметь разное качество сегментации (то есть качество восстановления границ объектов) в зависимости от глубины кодирования. Под термином качество сегментации подразумевается качество восстановления границ объектов. Другими словами авторы [246] предложили комбинировать результаты кодирования с нескольких слоев кодировщика для получения результатов с разным рецептивным полем. Иллюстрация этого принципа показана на рисунке 7.71. В отношении этого принципа отметим, что для объектов большего размера требуется большее рецептивное поле, чем для небольших объектов. Однако чем меньше рецептивное поле, тем больше внимания уделяется деталям на изображении. При этом слишком мелкое рецептивное поле может привести к потерям точности в общем случае. Также это может привести к появлению нежелательных разрывов

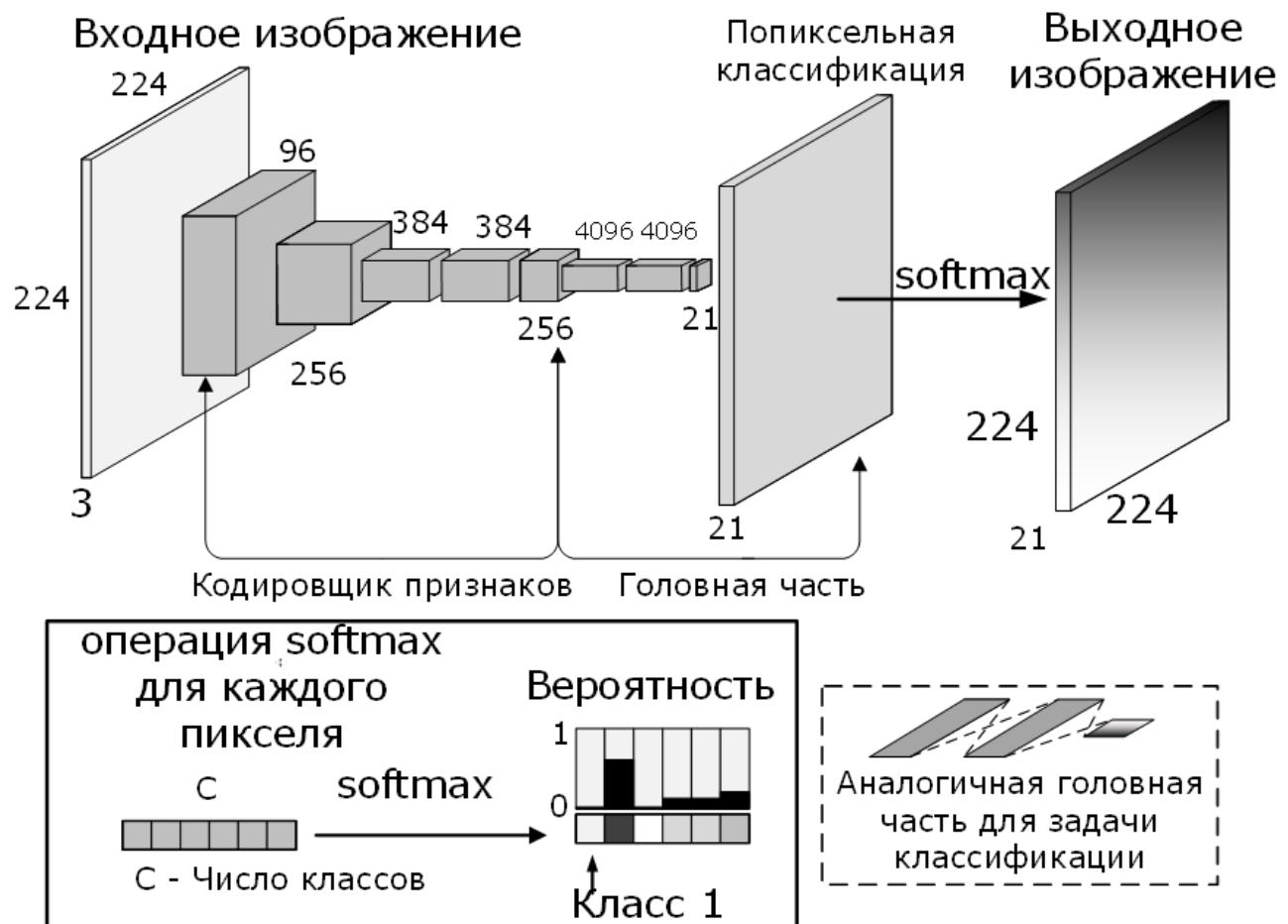


Рисунок 7.70 – Иллюстрация архитектуры FCN

семантических областей на итоговых изображениях. Таким образом, для объектов разных размеров должен быть найден оптимальный размер кодировщика.

Отметим, что архитектура FCN относится к классу методов на основе замены только головной части сети классификации. Архитектуры на основе FCN используются по настоящее время, так одна из последних модификаций, вдохновленных FCN под названием FCN-Transformer [247] лидирует на конец 2022 года в бенчмарке сегментации медицинских изображений [248].

Особенности использования расширенной свертки. Одна из проблем архитектур семантической сегментации – это использование большого числа слоев понижения размерности – то есть использование операций пулинга в кодировщике признаков. Пулинг приводит к потере пространственного разрешения – например утрате мелких деталей на изображении. Концепция расширенной свертки призвана решить проблему понижения размерности. Идея такой свертки в том, чтобы провесит паддинг не изображения, а ядра. Разрежённость ядра называется степенью расширения или dilation rate. Идея такой свертки была предложена в работе [249]. Расширенная свертка может быть описана следующим

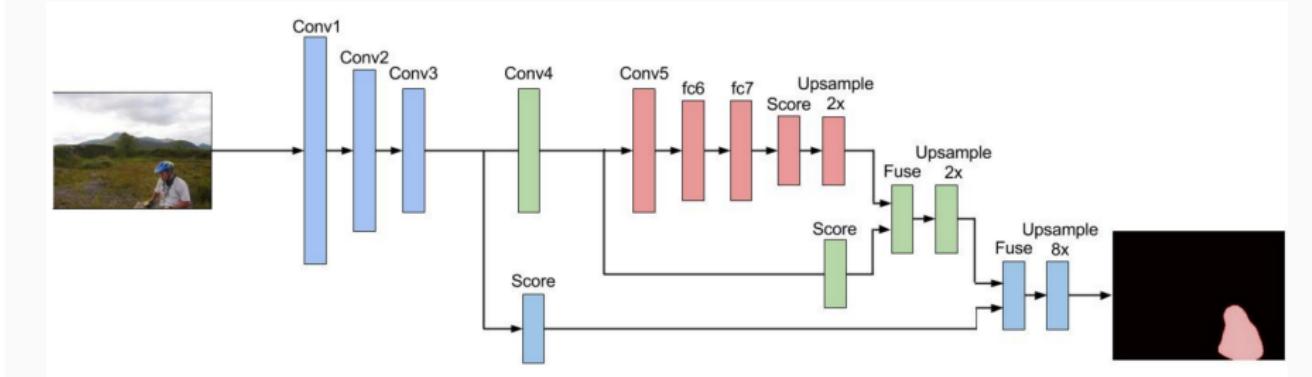
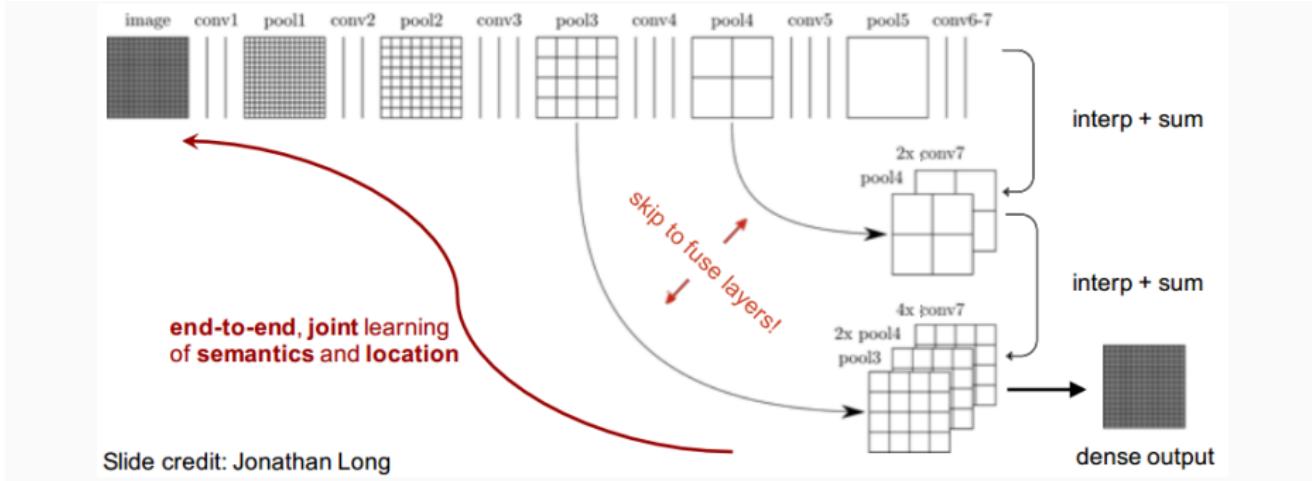


Рисунок 7.71 – Иллюстрация принципа восстановления изображений с разным рецептивным полем в архитектуре FCN

выражением:

$$r(i, j) = (k * x)(i, j) = \frac{1}{W \cdot H} \sum_{w=0}^{W_G-1} \sum_{h=0}^{H_G-1} k(h, w)x(i + l_h, j + l_w), \quad (7.42)$$

где l т.е. l_w, l_h - это степень расширения. Иллюстрация работы такой свертки со степенью расширения 2 приведена на рисунке 7.72. Отметим, что такая свертка редко используется с шагом больше 1.

Расширенная свертка позволяет увеличить receptive поле без потери пространственного разрешения и при экономии общего числа параметров. Другими словами, расширенная свертка реализует эффект увеличенного размера ядра. Так, например для фильтрации того какой эффект можно получить при помощи расширенной свертки с ядром размера 7, позволяет достичь свертка с размером ядра 13. На рисунке 7.73 показаны примеры работы расширенной свертки и обычной с разными размерами ядра. Из рисунка виден описанный эффект.

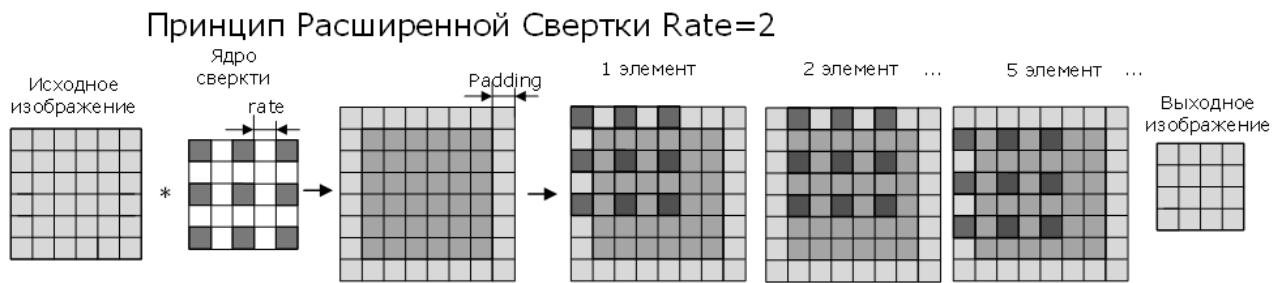
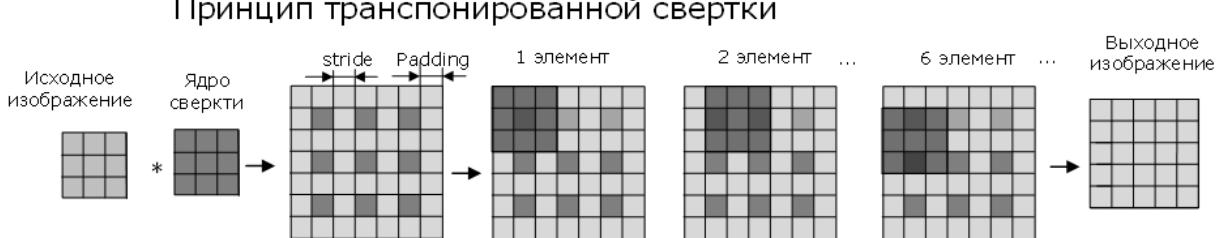
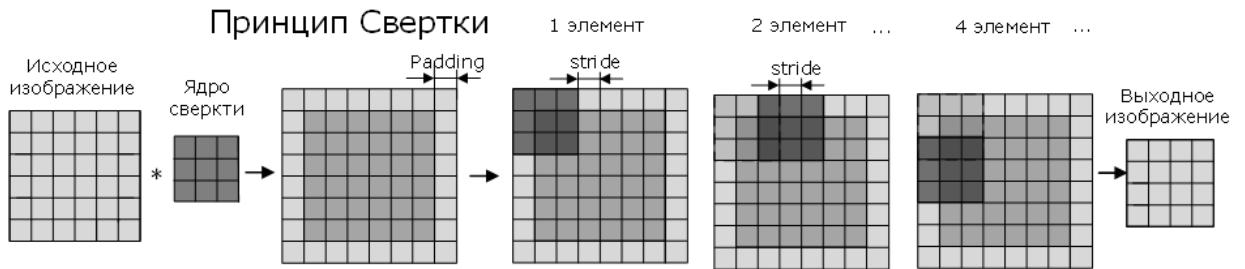


Рисунок 7.72 – Иллюстрации работы расширенной свертки

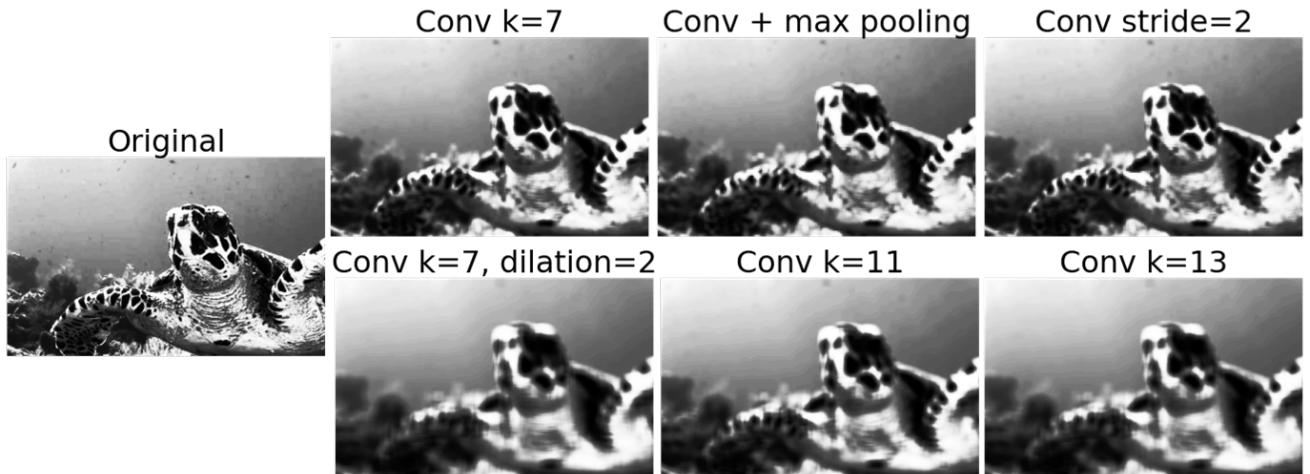


Рисунок 7.73 – Иллюстрации работы расширенной свертки

Часто операция расширенной свертки используется с постепенным или параллельным увеличением степени расширения. Этот подход реализует концепцию увеличенное поле зрения или large Filed of View. Если каскад расширенных сверток реализуется последовательно, то как правило степень расширения увеличивается по нарастанию глубины. На рисунках 7.74 проиллюстрирована работа каскада последовательных расширенных случаев для одномерной и двумерной операций свертки. Данные примеры иллюстрируют возможность получения увеличенного рецептивного поля по сравнению с классической

сверткой размером 3×3 .

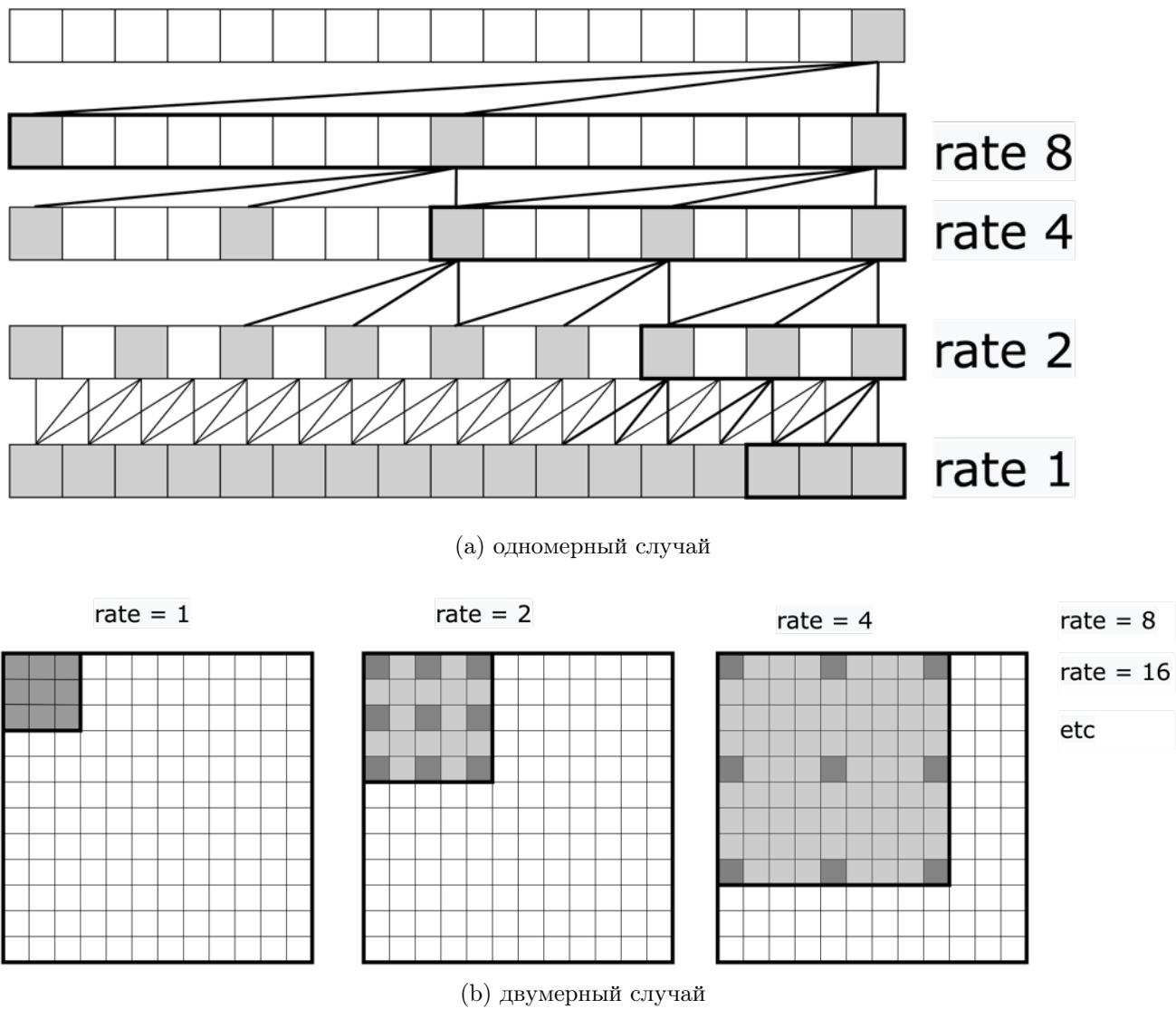


Рисунок 7.74 – Иллюстрации работы каскада последовательных расширенных сверток

Отметим, что в компьютерном зрении в настоящее время чаще используется подход параллельного использования расширенных сверток. Объяснение этого подхода будет дано ниже.

Архитектура DeepLab. Одним из первых успешных примеров использования концепции расширенной свертки стала архитектура DeepLab [250]. Основанная идея архитектуры DeepLab замена верхней части кодировщика признаков сети классификации VGG на набор последовательных слоев расширенной свертки. Такой блок расширенных сверток предложено называть atrous convolution или LargeFOV. Однако, отметим, что сегодня под этим термином в первую очередь понимается параллельное использование расширенных сверток. Даная архитектура используется скачкообразный принцип получения головного слоя. Для этого применяется билинейная интерполяция. Иллюстрация архитектуры DeepLab приведена на рисунке 7.75.

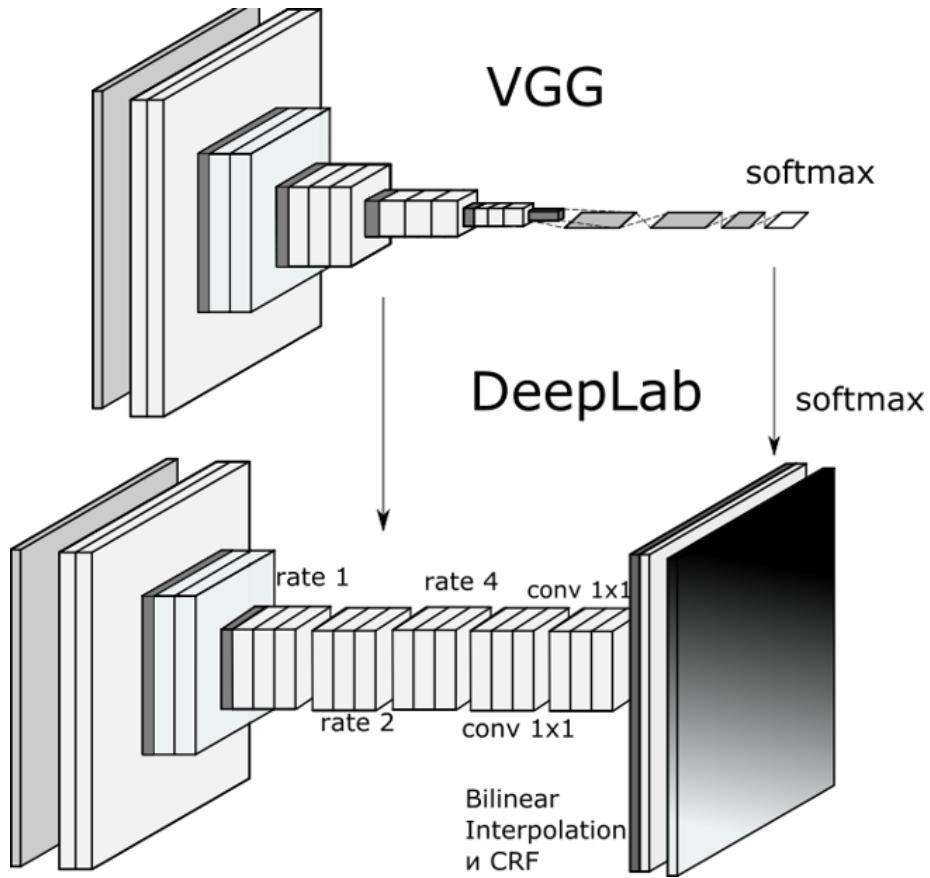


Рисунок 7.75 – Иллюстрации архитектуры DeepLab

Суть подхода DeepLab заключается в отказе от части слоев пулинга в кодировщике признаков при достижении достаточного рецептивного поля. В конце сеть использует билинейную интерполяцию для получения изображения итогового размера. В оригинальной работе авторы так же использовали дополнительную операцию после получения результатов. Операция называется Fully connected Conditinal Random Field (CRF). Операция по существу, похоже на билатерального фильтра, но работает итерационно. Целью такой операции было уточнение границ классов. Данная операция относится к классу методов классического компьютерного зрения. Этот прием дал небольшой прирост точности, однако существенно замедлил работу архитектуры. Примеры использования свертки пулинга и транспонированной свертки, расширенной свертки, свертки с пулингом и интерполяции, расширенной свертки и интерполяции, а также расширенной свертки и интерполяции плюс билатерального фильтра показаны на рисунке 7.76 для сравнения результатов. Отметим, что архитектура DeepLab была не первой, в которой использовалась данная техника. Так первые попытки использования этой техники были применены в архитектуре FCN [246]. Однако в архитектуре FCN этот подход не стал каноном.



Рисунок 7.76 – Иллюстрации сравнения использования интерполяции с билатеральным фильтром и без него (аналог CRF)

Архитектура PSPNet. Другим вариантом развития подхода сегментации как расширения идей сверточных сетей стала архитектура PSPNet или Pyramid scene parsing network [251]. В основе этой архитектуры лежит предложение использования т.н. пространственного пирамидального пулинга (spatial pyramid pooling, SPP). Основная идея пирамидального пулинга это идея использования глобального среднего или максимального пулингов, но отдельных областей изображения. При чем параллельно используются несколько версий операции с разным размером выхода. Идея SPP слоя показана на рисунке 7.77. Отметим, что сама идея SPP была впервые предложена для решения задач обнаружения объектов в работе [252]. Каждый уровень пулинга позволяет по-разному учитывать контекст изображения. Однако, отметим, что в SPP карты признаков объединяются (конкатенируются) вместо суммирования. Таким образом информация с разным рецептивным полем учитывается более точно.

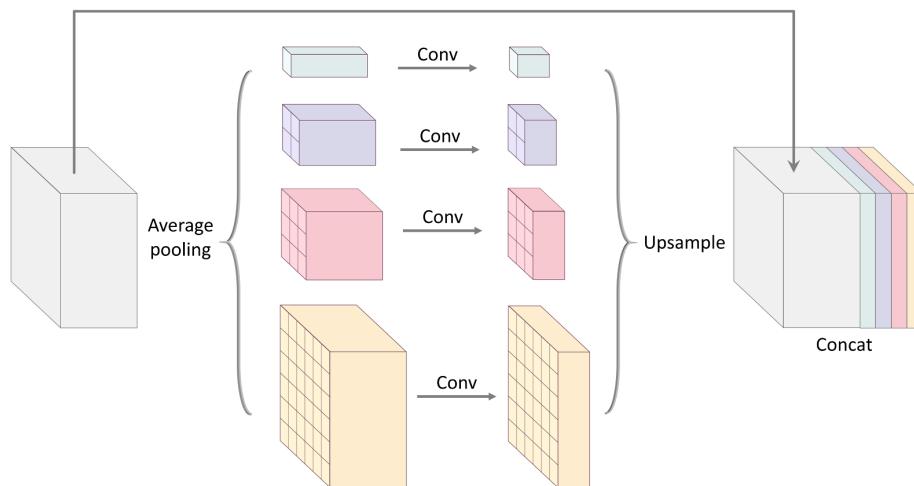


Рисунок 7.77 – Иллюстрация слоя пирамидального пулинга SPP

Идея SPP была использована в архитектуре PSPNet. Иллюстрация архитектуры приведена на рисунке 7.78. В архитектуре PSPNet использованы кодировщики признаков на основе ResNet архитектур. Каждый блок SPP сопровождается слоем батч нормализации. Процедура расширения размеров карт признаков выполняется при помощи билинейной интерполяции. Также в SPP блоке используется остаточная связь. В работе [178] была предложена модифицированная версия данной архитектуры с RepVGG кодировщиком признаков для низкопроизводительных устройств.

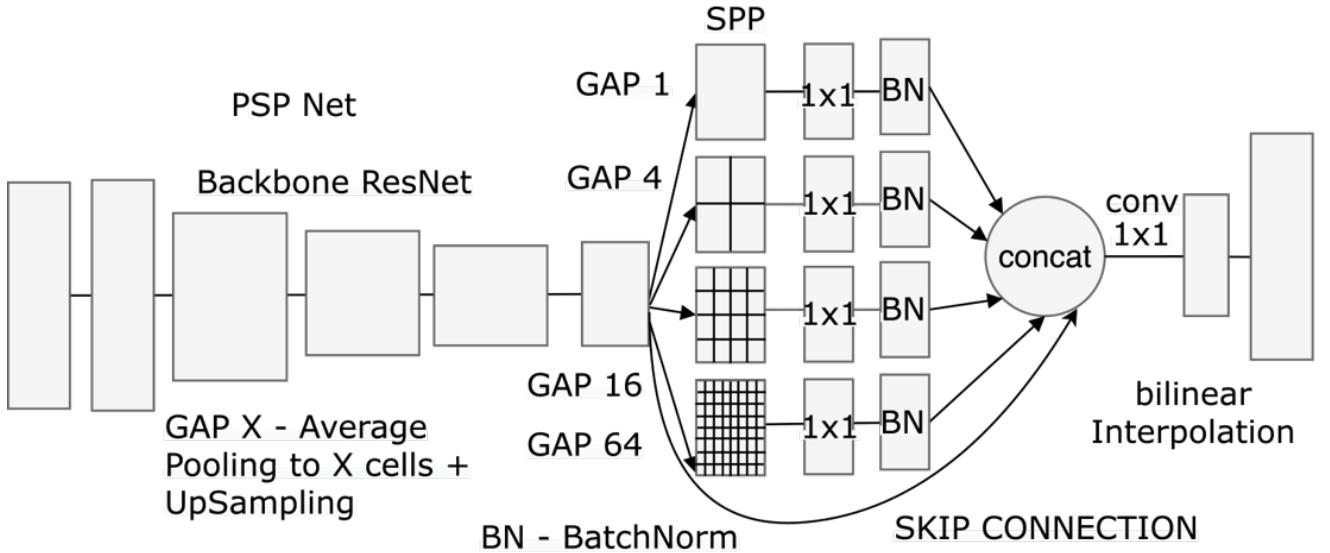


Рисунок 7.78 – Иллюстрация архитектуры PSPNet

Деформированная Свертка. Одни из развитий идеи расширенной свертки стала идея свертки с обучаемым расширением или деформированной свертки. Эта идея была предложена в работе [253]. Основной предпосылкой для разработки этого подхода стала идея, что расширенная свертка будет лучше работать для объекта если ядро будет соответствовать положению объекта на изображении. Так, например, если объекта будет расложен по горизонтали то он будет более качественно выделен соответствующим фильтром, чем аналогичный объект, расположенный по вертикали. Поэтому в этом случае, вероятно будет правильней фильтр повернуть. Отметим, что в некотором смысле эта идея аналогична предложению проводить обучаемые аффинные преобразования изображений в модели [254]. Однако тут вычислительная сложность будет ниже. По существу деформированная свертка сводится к тому, что для каждого значения на поле свертки обучается пространственное смещение. Для этого в дополнение к основному ядру свертки добавляется дополнительное, генерирующее $2N$ карт признаков, где каждая карта задает смещение по направлениям вертикали и горизонтали для каждого элемента свертки. Иллюстрация деформированной свертки показана на рисунке 7.79. Выражение описывающее деформированную свертку может быть задано как

$$y(p_0) = \sum_k^K w \cdot x(p_0 + k + \Delta p_k), \quad (7.43)$$

где $y(p_0)$ - результат свертки для набора пространственных координат p ; Δp_k - вводимые деформации.

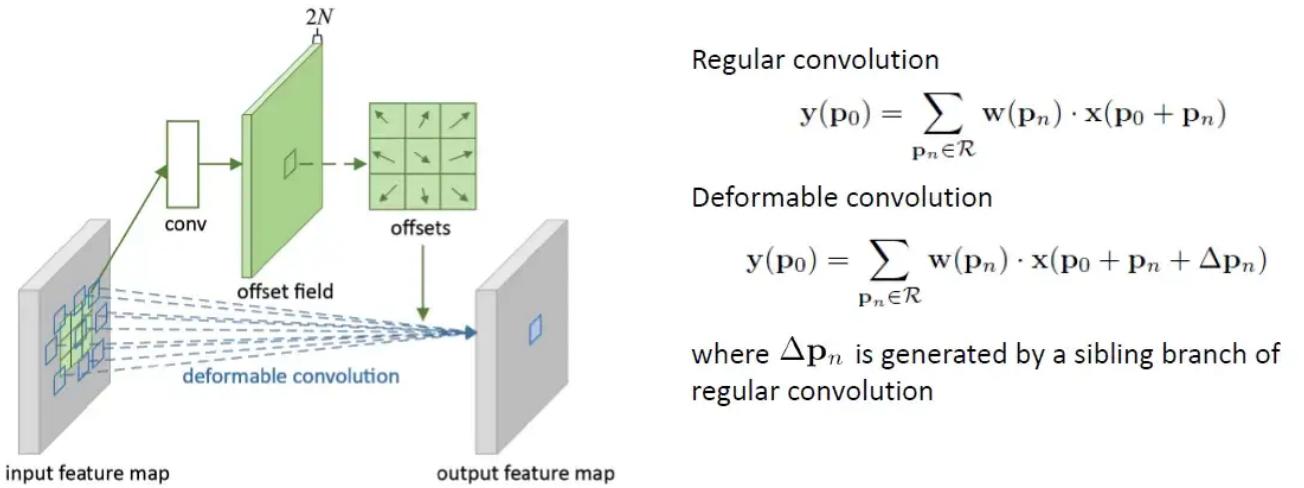


Рисунок 7.79 – Иллюстрация принципа работы деформированной свертки

При анализе деформированных сверток авторы пришли к выводам, что размеры рецептивного поля деформируемых сверток коррелируют с размерами сегментируемых объектов, что указывает на эффективность применения деформаций. При этом размеры фильтров в фоновой области (не сегментируемой области) находятся между размерами средних и крупных объектов, что указывает на то, что для распознавания фоновых областей необходимо относительно большое рецептивное поле [253].

Отметим, что в работе [253] также было предложено использование т.н. Aligned-кодировщиков признаков. То есть вариантов архитектур, в которых операции максимального пулинга заменялись на операции свертки с увеличенным шагом. Также кодировщики дополнялись промежуточными слоями. Были предложены Aligned- версии таких архитектур, как Inception, ResNet и Xception.

Также авторы предложили вторую версию деформированной свертки в работе [255]. Данная версия свертка имела не только смещения в пространстве, но и то, что авторы назвали амплитудной модуляцией свертки.

$$y(p_0) = \sum_k^K w \cdot x(p_0 + k + \Delta p_k) \Delta m, \quad (7.44)$$

где Δm - дополнительный коэффициент модуляции в диапазоне от 0 до 1, вводимый для каждой позиции ядра свертки. В некотором смысле авторы приблизили идею деформированной свертки к т.н. динамической свертке и идеи внимания. То есть итоговые коэффициенты свертки формируются динамически для каждого изображения. Также отметим, что предложения по модификациям расширенной и деформированной сверток появляются и вне рассмотренных работ. Например, в работе [256] расширенная свертка дополнена идеей внимания. В работе [257] предложен модуль внимания с использованием идеи деформированной свертки. Архитектура на момент конца 2022 года имеет лучшие

показатели на популярном наборе данных ADE20K [258].

Архитектура DeepLab V2. Развитием подхода DeepLab стала идея использование параллельно нескольких сверток с разной степенью расширения. Такой подход было предложено назвать atrous Spatial Pyramidal Pooling (ASPP). Подход был предложен в рамках архитектуры DeepLab V2 [259]. Идея SPP слоя близка идее ASSP, однако в данной версии производится сложение результирующих карт признаков вместо их объединения.

Цель слоя ASSP можно объяснить следующим. Часто области изображений одного класса имеют разные масштабы, и разный уровень необходимого контекста их окружения для их корректной классификации. То есть области могут быть разного размера и разной сложности своих границ. Другими словами для таких областей может быть необходим разный размер рецептивного поля. Слой ASSP позволяют учесть потребности в наличие нескольких уровней рецептивного поля одновременно. Таким образом повышается вероятность выделения как совсем небольших, так и достаточно больших объектов по сравнению с размером изображения. Идея блока ASSP второй версии DeepLab показана на рисунке 7.80. Отметим, что структура блока ASPP не фиксирована однозначно. В некоторых последующих архитектурах она будет изменена.

Atorus Spatial Pyramid pooling

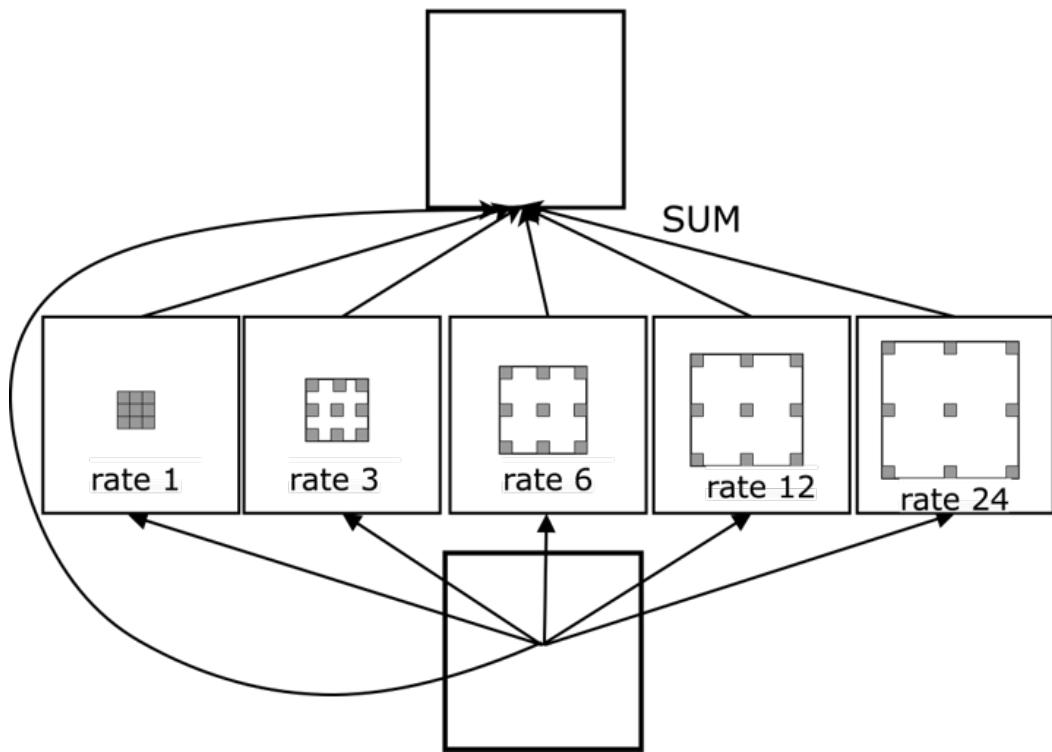


Рисунок 7.80 – Иллюстрация блока ASPP

Иллюстрация архитектуры DeepLab V2 приведена на рисунке 7.81. Данная архитектура близка в своих базовых принципах к своей первой версии. Тут ASPP используется с дополнительными свертками размера 1×1 . Такой подход позволяет более точно выделить признаки для каждой ветви ASPP и уменьшить общее число карт признаков. Предполагается, что сеть выучит в ходе тренировки какие карты нужно оставить. Авторы также предложили заменить кодировщик VGG на ResNet. Подход DeepLab V2 включает несколько моделей, для небольших используются степени расширения 2, 4, 8, 12 для большой модели 6, 12, 18, 24 как показано на рисунке 7.81. Также в этой модели предполагается использование CRF.

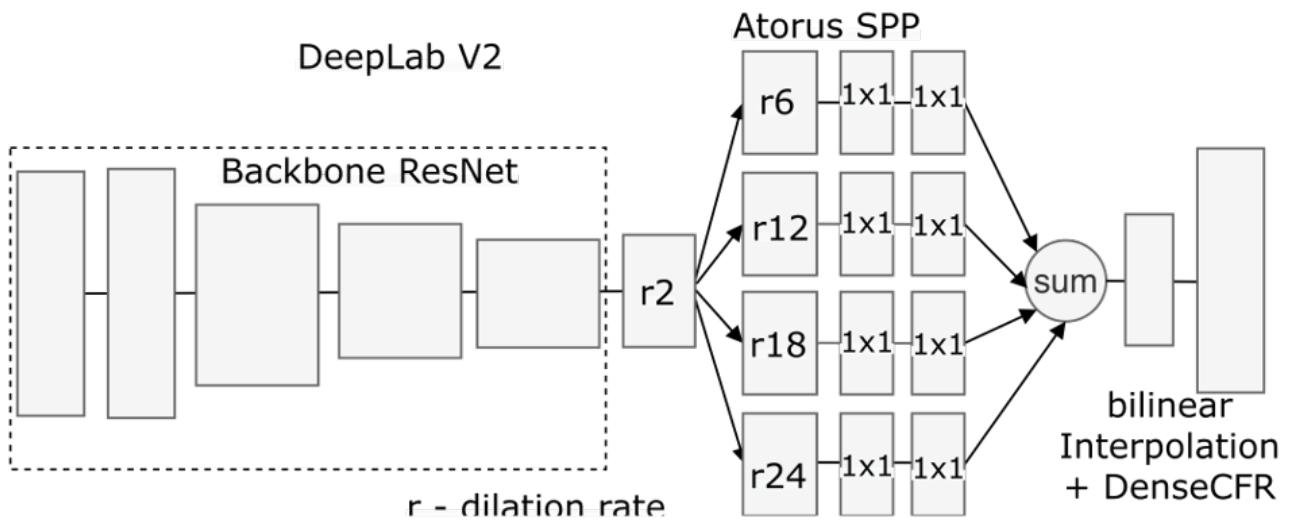


Рисунок 7.81 – Иллюстрация архитектуры DeepLab v2

Также отметим, что позже на основе подходов DeepLab и DeepLab V2 и ResNet38 была предложена модификация архитектура ResNet38 для решения задач семантической сегментации. Авторы работы упростили архитектуру DeepLab и заменили кодировщик на их версию. Это дало прирост точности и сделало ResNet38 одним из классических этапов в семантической сегментации [156].

Архитектура DeepLab V3. Идеи второй версии DeepLab и PSPNet были объединены в архитектуре DeepLab V3 [260]. Иллюстрация этой архитектуры приведена на рисунке 7.82.

Основная идея DeepLab 3 заключается в попытке достижения баланса между использованием расширенной и классической сверток в составе ASPP блока. Дело в том, что повышение рецептивного поля (field of view) – это увеличение контекста. Однако для точной классификации часто нужно небольшое рецептивное поле, то есть нужна более прицельная классификация. Для объединения этих двух подходов авторы DeepLab V3 объединили в блоке ASPP классическую и расширенные свертки, а также так называемый пулинг изображений (image pooling) – то есть глобальный пулинг и расширение до размера изображения. Также каждый уровень пулинга включает батч нормализацию. Также в этой версии архитектуры DeepLab предложено заменить операцию суммирования в ASPP блоке на конкатенацию. Также в этой архитектуре авторы предложили использовать

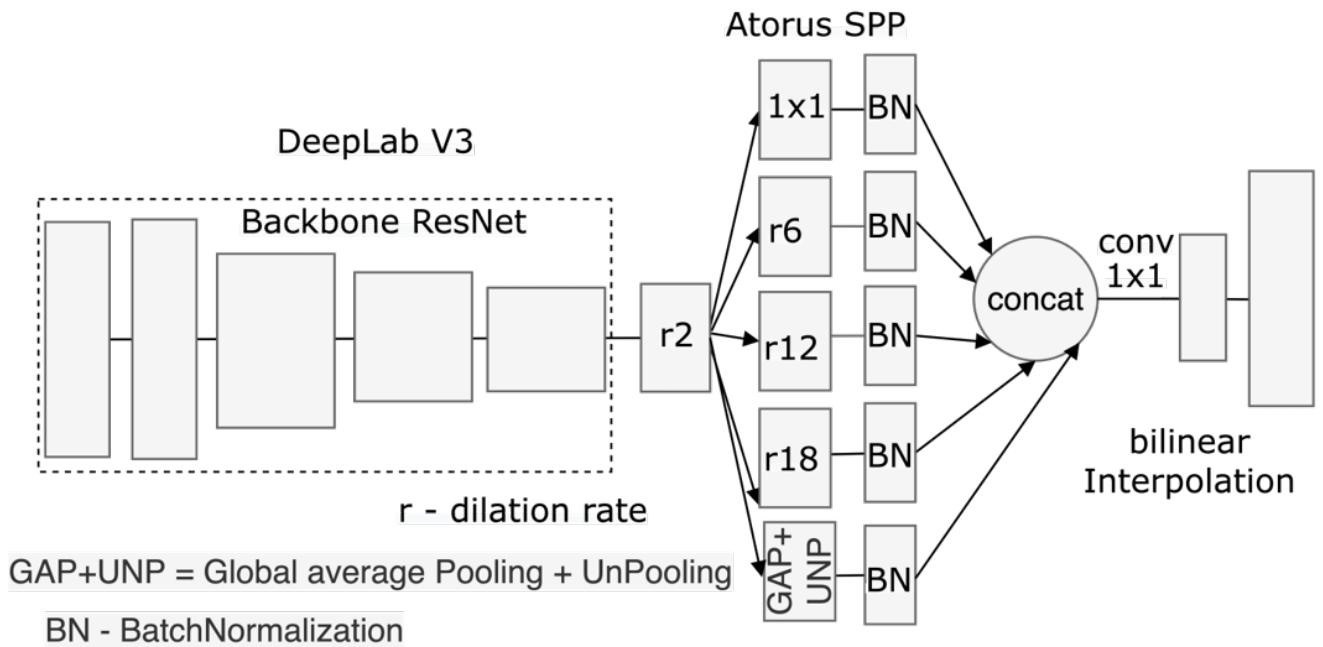


Рисунок 7.82 – Иллюстрация архитектуры DeepLab v3

дообучения с особой техникой аугментации, названной bootstrapping on hard images или бутстреп на сложных изображениях. Другими словами изображения, содержащие классы наиболее сложные для сегментирования дублировались на этом этапе. В целом архитектура DeepLab 3 без предобучения дала небольшой прирост точности относительно PSPNet. Также был проанализирован вариант DeepLab 3 с предобучением кодировщика на наборе данных JFT300 M. Этот вариант дал значительный прирост точности на стандартных тестах. В данной версии архитектуры авторы также отказались от использования CRF для результатов работы модели.

7.3 Подход энкодер-декодер

Архитектура DeconvNet. В 2015 году в работе [261] была предложена архитектура сверточной нейронной сети с использованием следующих идей.

- Глубокой головной части (декодера) в противовес скачкообразному изменению размеров выходных карт признаков в архитектуре FCN.
- Использование слоев т.н. обратного пулинга (повышающей передискретизации или unpooling) в декодере.
- Построение декодера с использование транспонированной свертки аналогично архитектуре FCN.

В основе архитектуры использовался кодировщик признаков VGG 16. Операция обратного пулинга реализовывалась как противоположная операции max pooling. При этом в ходе кодирования позиции пикселей выбранные в ходе операции max pooling фиксировались. Операция unpooling затем использовала эти позиции для восстановления размеров карт признаков на каждом этапе декодера. Архитектуру было предложено назвать DeconvNet. Иллюстрация архитектуры DeconvNet приведена на рисунке 7.83.

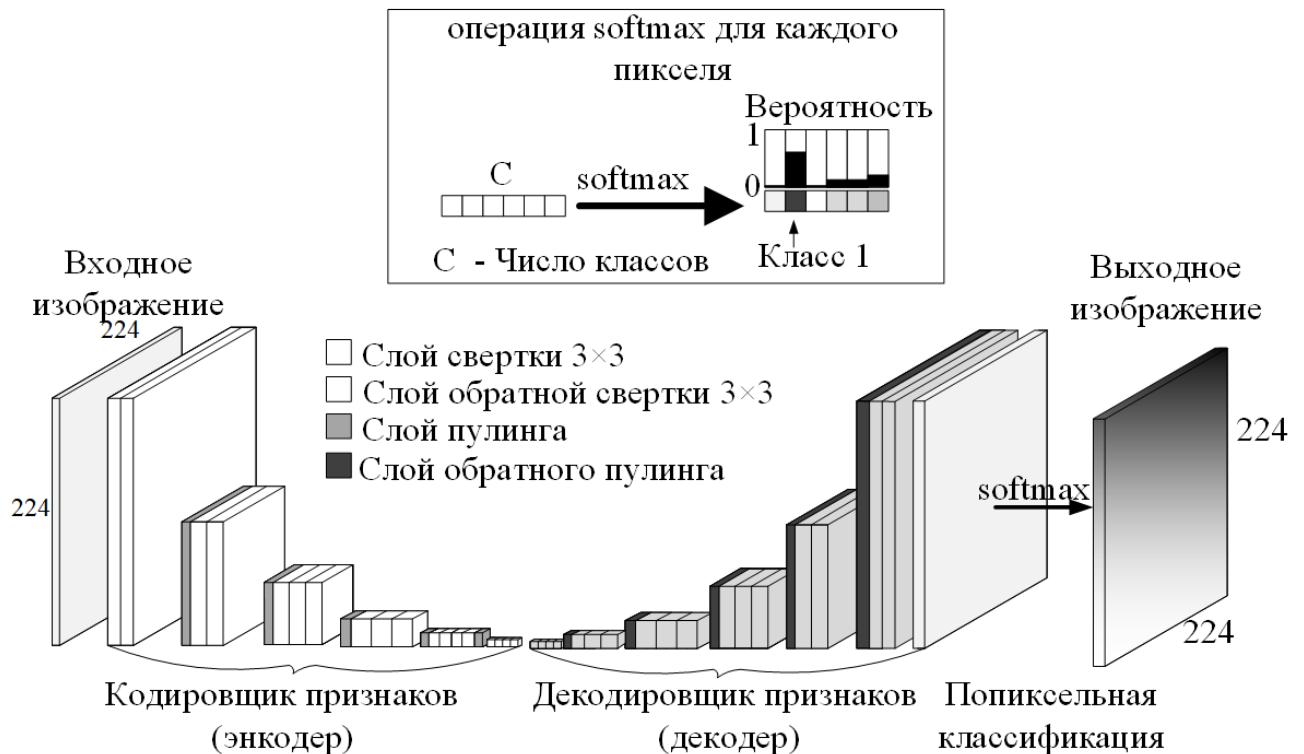


Рисунок 7.83 – Иллюстрация архитектуры DeconvNet

По существу, архитектура DeconvNet состоит из двух частей: кодировщик признаков (энкодер) и декодировщик признаков (декодер), а также слоя попиксельной классификации. Цель использования энкодера заключается в выделение релевантных признаков во входных данных. Цель использования декодера заключается в восстановлении из выделенных признаков т.н. маски целевого класса (силуэт или контур соответствующего объекта или сцены). Число выходных карт признаков (выходных каналов) сети должно соответствовать числу классов на которые должно быть поделено (сегментировано) изображение. Решение о том, к кому классу принимается путем использования операции softmax между всеми каналами по каждому пикслю. Таким образом, каждый выходной канал содержит силуэт соответствующий одному целевому классу.

Архитектура SegNet. В 2016 году идеи архитектур DeconvNet и FCN была оптимизирована и опубликована в работе [262] под названием **SegNet**. Основными особенностями SegNet стали использование сверточных слоев как в декодере и усечение глубины кодировщиков признаков в обоих частях архитектуры. То есть декодирование признаков осуществляется при помощи unpooling и таких же сверток как энкодере.

Результаты SegNet оказались не хуже, чем в DeconvNet. Иллюстрация архитектуры приведена на рисунке 7.84.

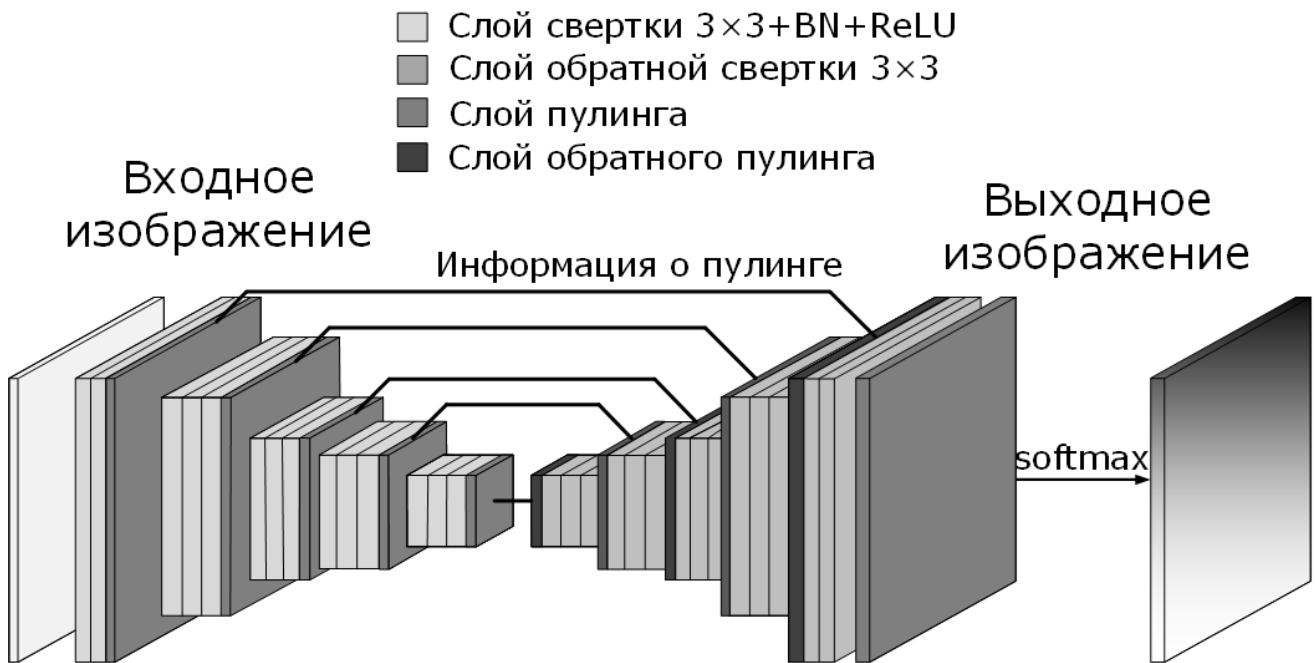


Рисунок 7.84 – Иллюстрация архитектуры SegNet

В отношении архитектуры также следует отметить, что одной из главных целей при ее разработке было экономия ресурсов. Так подход усечения кодировщика признаков позволяет сократить число параметров сети. А операция копирования только позиций пулинга не требует много ресурсов для хранения соответствующих индексов. При этом комбинация обеих операций позволяет восстанавливать в декодере границы классов достаточно точно.

Архитектура U-Net. В 2015 году О. Роненбергом и соавторами была предложена архитектура U-Net [25]. Особенностью данной архитектуры было использование в декодирующй части архитектуры совмещенных (конкатенация) карт признаков энкодера и декодера. Такой подход позволил существенно увеличить точность особенно в случае малоразмерных деталей изображения. Изначально архитектура U-Net была предложена для решения медицинских задач [263]. В настоящее время архитектура является одной из наиболее популярных среди любых задач, сводящихся к семантической сегментации (по данным портала paperswithcode.com) [264]. Однако, на сегодня U-Net не является наиболее точной архитектурой, например для классического набора данных PASCAL VOC 2012 [265] по данным портала paperswithcode.com [266]. Иллюстрация архитектуры U-Net приведена на рисунке 7.85.

Отметим некоторые особенности оригинальной архитектуры U-Net.

- Архитектура состоит из блоков.
 - Каждый блок энкодера представляет собой две свертки с ядром 3×3 и слой макс-пулинг.

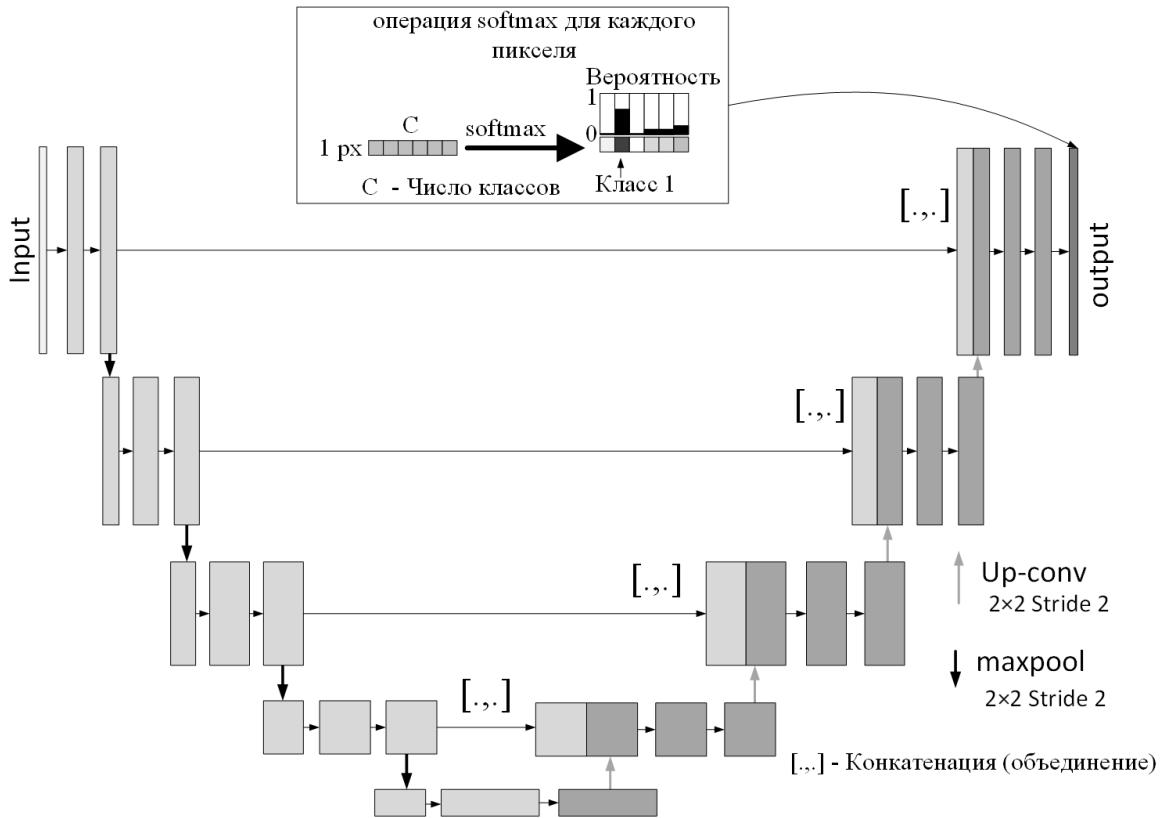


Рисунок 7.85 – Иллюстрация архитектуры U-Net

- В декодирующей части каждый блок состоит из транспонированной свертки с ядром 2×2 и также двух сверток с ядром 3×3 . Таким образом повышение разрешения осуществляется с обучаемыми параметрами.
- Информация из блоков энкодера "сливается" с декодированной восстановленной информацией в первом сверточном слое каждого блока декодера.
- Перед слоем принятия решений используется свертка 1×1 для приведения общего числа карт признаков к необходимому на выходе.
- В оригинальной работе все свертки используются без т.н. паддинга. То есть с уменьшением разрешения карт признаков.
- Вместо паддинга исходное изображение дополняется рамкой, зеркально отражающей граничные области входного изображения (mirror padding).
- Также в работе предложено проводить аугментацию данных деформированием входных изображений и выходных масок.
- Так как архитектура разработана для задач где предполагается большое число близкорасположенных объектов (например клетки), то предлагается взвешивать участки изображений, отображающие границы объектов. То есть сеть будет штрафоваться за их неверную сегментацию выше, чем в остальных областях.

Архитектуры на основе U-Net. После 2015 года различными авторами были предложены большое количество архитектур на основе U-Net и подхода энкодер-декодер в целом [267]. В том числе, наиболее успешные архитектуры, которые, однако переосмыслены с использованием расширенных понятий свертки и с использованием метода NAS или другими подходами. Отметим, что, в описанных архитектурах семантической сегментации предполагается подход обучения с учителем. Однако, в литературе также обсуждаются архитектуры семантической сегментации с т.н. слабым обучением (weak supervised) - то есть с использованием только классов объектов на изображении или частичной разметки и меток классов [268].

Одним из таких подходов, например является архитектура U-Net++ [269]. Иллюстрация архитектуры U-Net++ приведена на рисунке 7.86. В основе архитектуры лежат следующие идеи.

- Введение слоев свертки на путях объединения признаков энкодера и декодера. По задумке авторов это согласует (устраняет семантические различия) между картами признаков кодировщика и декодера.
- Введение блоков DenseNet, что улучшает число вариантов рецептивного поля и путей прохождения градиента в архитектуре.
- Введение контроля за необходимостью использования элементов архитектур. Это позволяет исключать ненужные элементы. В худшем случае архитектура должна достигать производительности сравнимой с использованием U-Net (основного пути архитектуры). То есть, на рисунке 7.86 функция потерь может быть рассчитана как по всем возможным выходам, так и по части из них. Таким образом, есть возможность выбрать лучший случай.

Эта архитектура была доработана в работе где было предложено название новой модификации UNet3+. Иллюстрации сравнения архитектур U-Net, U-Net++ и U-Net3+ показаны на рисунке 7.87. В данной архитектуре каждый блок декодера включает все карты признаков меньшего или того же размера из энкодера и все карты признаков большего размера (предыдущие блоки) из декодера. Это позволяет учитывать одновременно как мелкие детали, так и крупногабаритные семантические особенности во всех своих масштабах. Также в данной архитектуре предполагается расчет функции потерь на каждом этапе декодера. Для этого выход каждого блока подается на сверточный слой с ядром 3×3 и проходит операции билинейной интерполяции.

Среди других модификаций архитектур на основе U-Net, отметим некоторые примеры, такие, как:

- Архитектура LinkNet [270]. Основная идея: упростить архитектуру (аналогично segnet и deconvnet). Для этого пересмотрена структура блока декодера U-Net, в виде комбинации точечной свертки, транспонированной свертки с перекрытием и точечной свертки без остаточных связей. Также в архитектуре оптимизированы входной и выходной блоки и сокращено число промежуточных блоков. Иллюстрация архитектуры LinkNet приведна на рисунке 7.88.

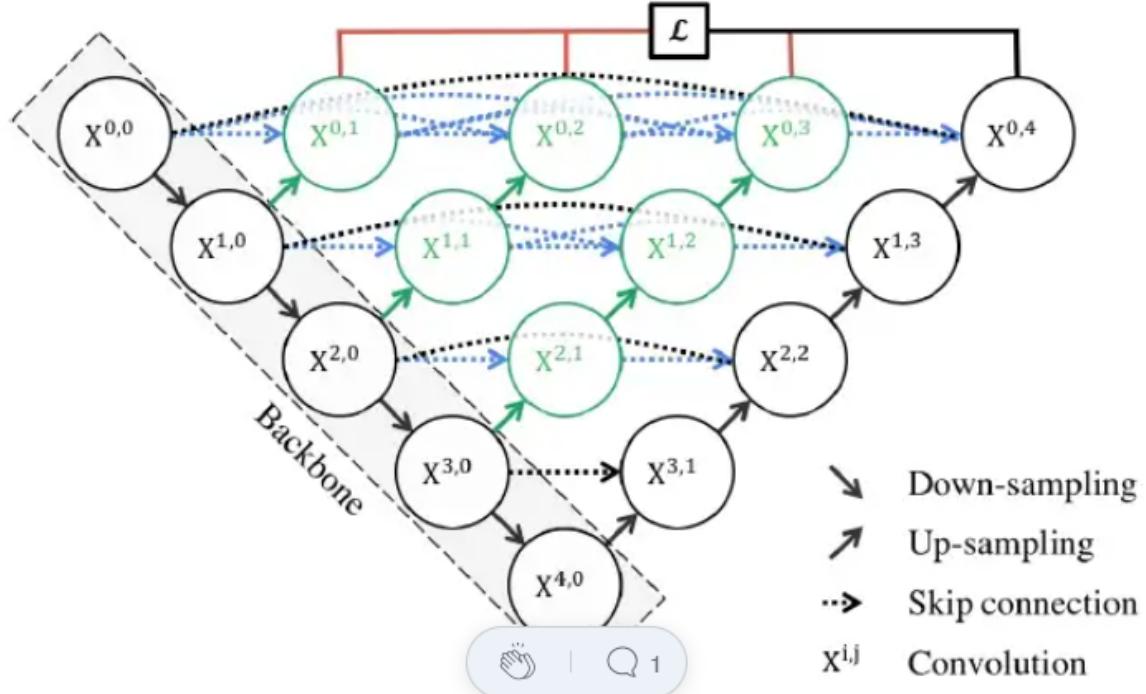


Рисунок 7.86 – Иллюстрация архитектуры U-Net++

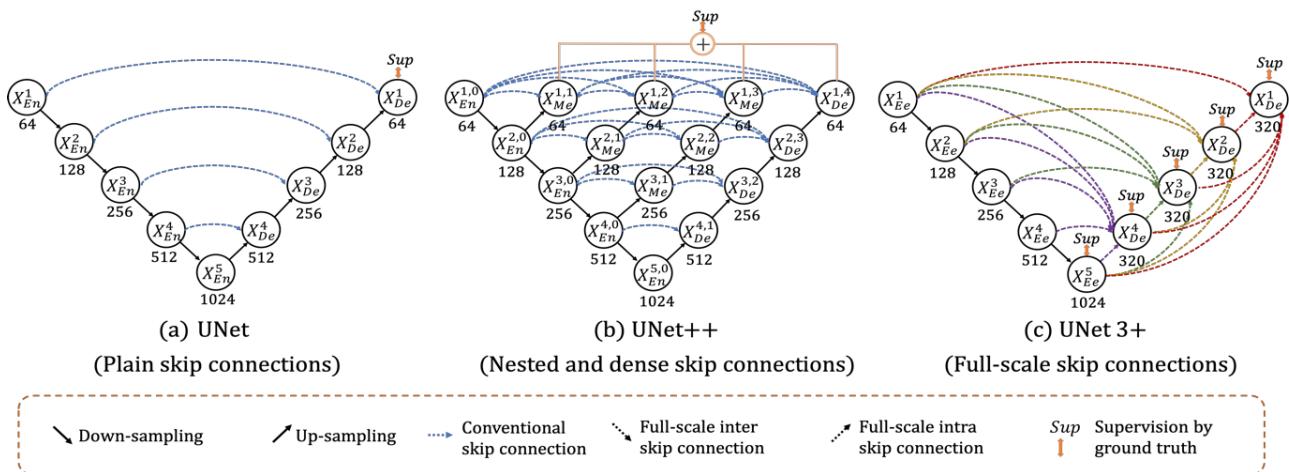


Рисунок 7.87 – Иллюстрации сравнения архитектур U-Net, U-Net++ и U-Net3+

- Архитектура Res-U-Net [271]. основная идея: использования идеи остаточной связи между слоями модулей в блоках U-Net. Также эта архитектура была модифицирована, новая версия получила название Res-U-Net++ [272]. Иллюстрация архитектуры Res-U-Net приведна на рисунке 7.89.
- Архитектура FC-DenseNet [273]. Основная идея: пересмотр структуры блока U-Net, использование DenseNet блоков. Иллюстрация архитектуры приведна на рисунке 7.90.
- Архитектура MDU-U-net [274]. Основная идея: организация всех допустимых комбинаций связей между картами признаков слоев энкодера и декодера без дополнительных сверточных слоев в связях.

- Архитектура Attention-U-Net [275]. Основная идея: использование слоя внимания на этапе конкатенации.
- Архитектура U2-Net [?]. Основная идея: представления каждого блока как энкодера так и декодера в виде U-Net структуры.

Эти и многие другие примеры подчеркивают важность идей, предложенных в архитектуре U-Net. Однако как правило архитектуры данного семейства используются в медицинских приложениях.

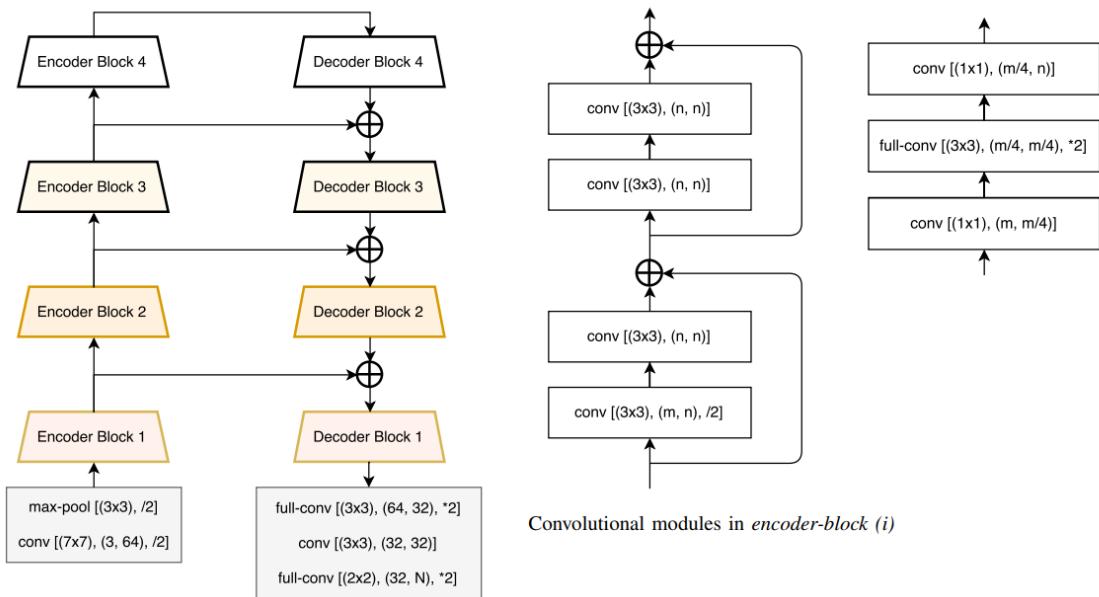


Рисунок 7.88 – Иллюстрации архитектуры LinkNet

Generator: Res-Unit

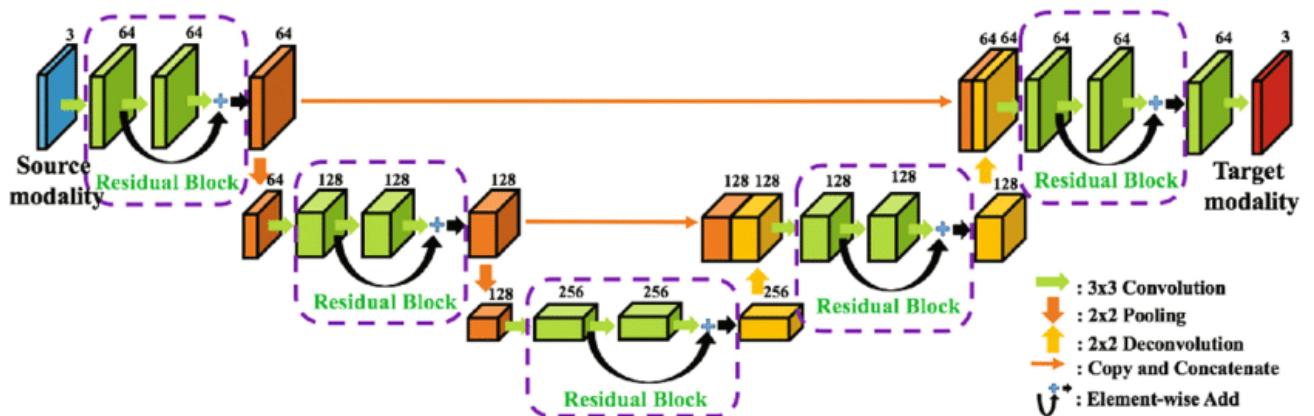


Рисунок 7.89 – Иллюстрации архитектуры Res-U-Net

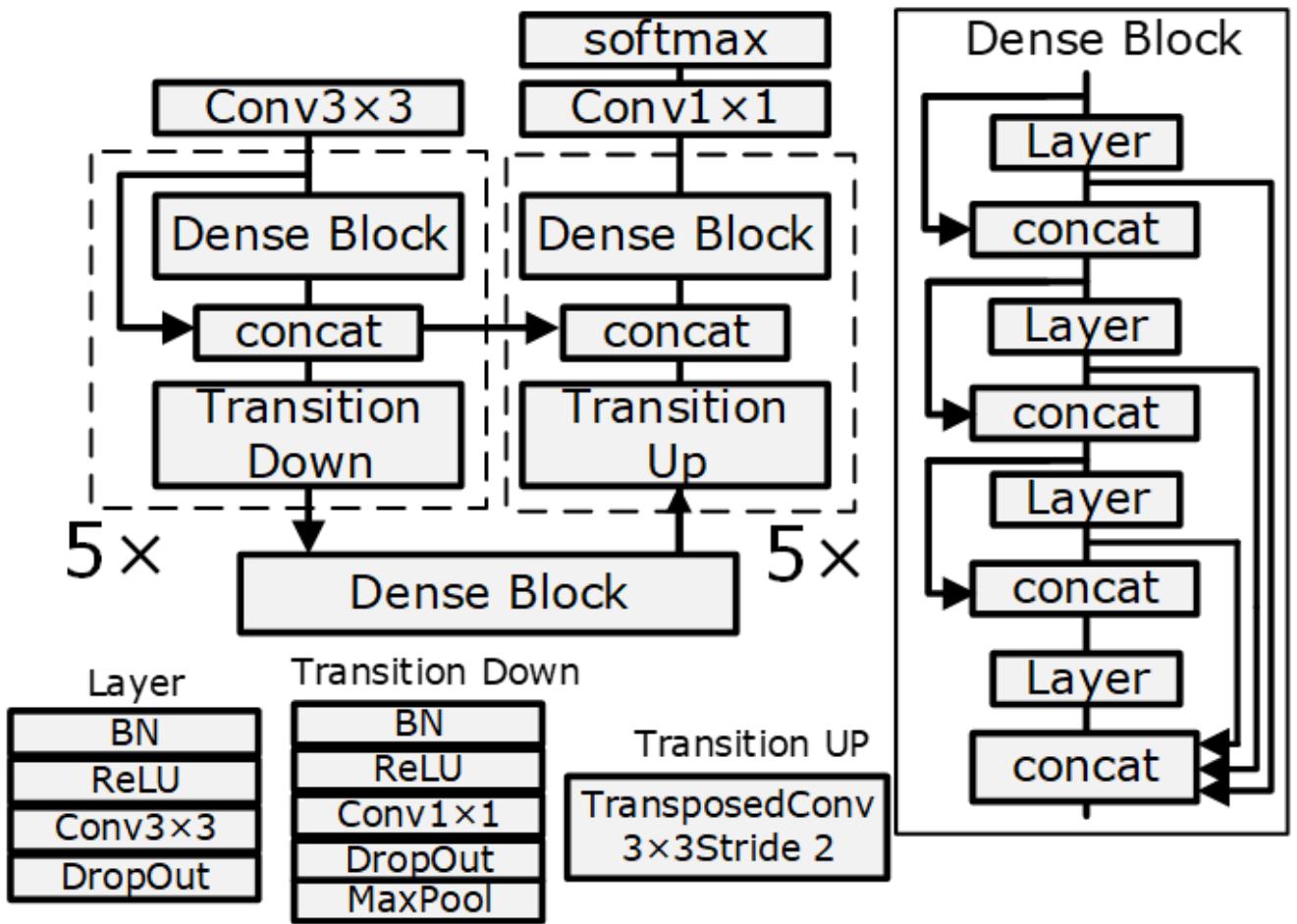


Рисунок 7.90 – Иллюстрации архитектуры FC-DenseNet

Архитектура MA-Net. Архитектура MA-Net предложенная в работе [276]. В основе архитектуры идеи использования самовнимания типа слой сжатия и возбуждения и идеи внимания энкодер-декодер в структуре соответствующей модели семантической сегментации. Архитектура предлагает два типа блоков внимания. Блок точечного внимания (Position-wise Attention Block, PAB) предназначен для моделирования внимания к пространственно-независимым признакам. Блок PAB используется на уровне латентного пространства между энкодером и декодером. Предполагается что на этом уровне выделяются пространственно-независимые признаки входного изображения. Локализующая информация о выделенных признаках восстановливается в декодере за счет использования соответствующих карт признаков из кодировщика. Объединение признаков происходит при помощи блока слияния много масштабного внимания (Multi-scale Fusion Attention Block, MFAB). Блок предназначен для выделения наиболее важных карт признаков как декодера, так и энкодера. Для этого в обоих наборах карт признаков используются структуры, аналогичные SE блоку. Также авторы предложили использовать ReNet блоки с заменой батч-нормализации на групповую нормализацию.

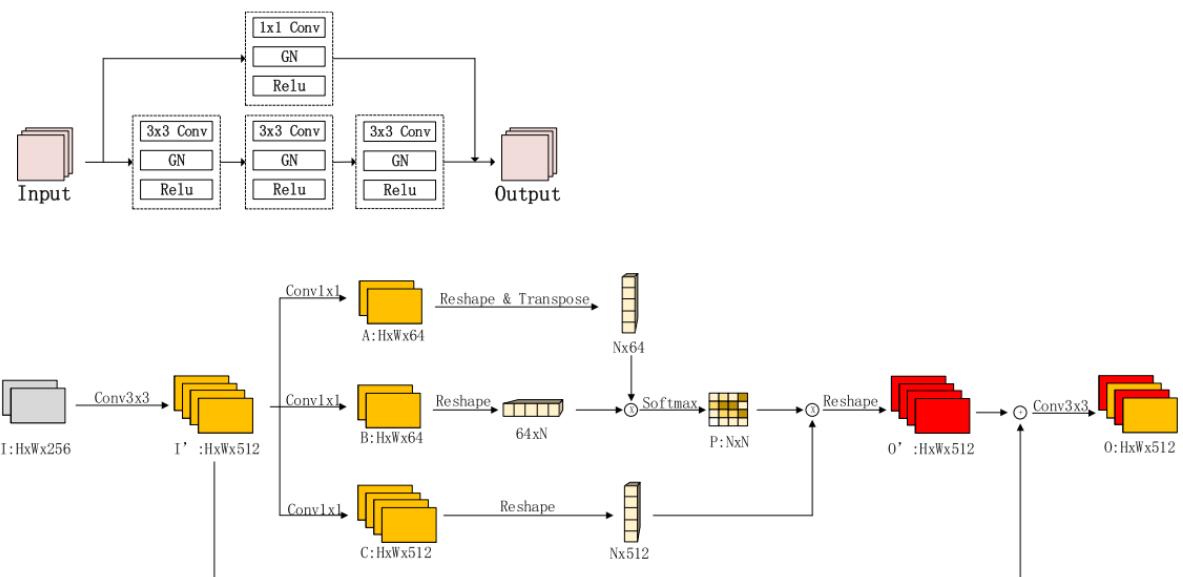
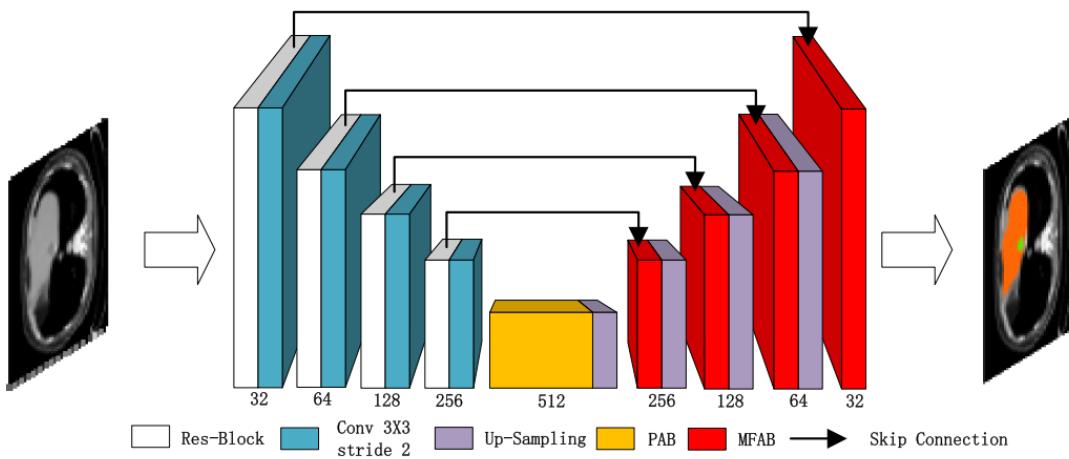


FIGURE 3. The Position-wise Attention Block (PAB). The input image is $H \times W \times 256$ and output is $H \times W \times 512$. The attention feature map is obtained by Softmax function.

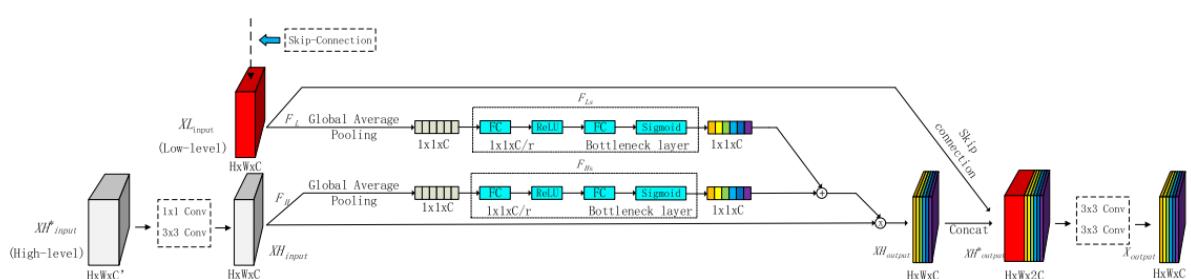


FIGURE 4. The Multi-scale Fusion Attention Block (MFAB). We use two SE-Blocks to capture Low-level and High-level feature map respectively. The final channel attention feature map is obtained via a Concat connection.

Рисунок 7.91 – Иллюстрации сравнения архитектуры MANet, блоков точечного внимания и многомасштабного внимания

7.4 Гибридные подходы

Архитектура DeepLab V3+ Недостатком рассмотренных архитектур являлось резкое расширение размера изображение в головном слое. Эта операция так или иначе может привести к потере информации, особенно если речь идет о небольших объектах. Кроме того, архитектуры типа энкодер-декодер лучше обрабатывают локальные признаки. Особенно, если информация о них копируется из кодирующей части в декодирующую. Однако Архитектуры рассмотренного класса с ASPP блоком более устойчивы к разным масштабам и разному контексту информации признаков для одних и тех же изображения. Для объединения достоинств подходов энкодер-декодер и расширенной свертки авторами был предложен подход DeepLab V3+. Иллюстрация этого подхода приведена на рисунке 7.92 [277].

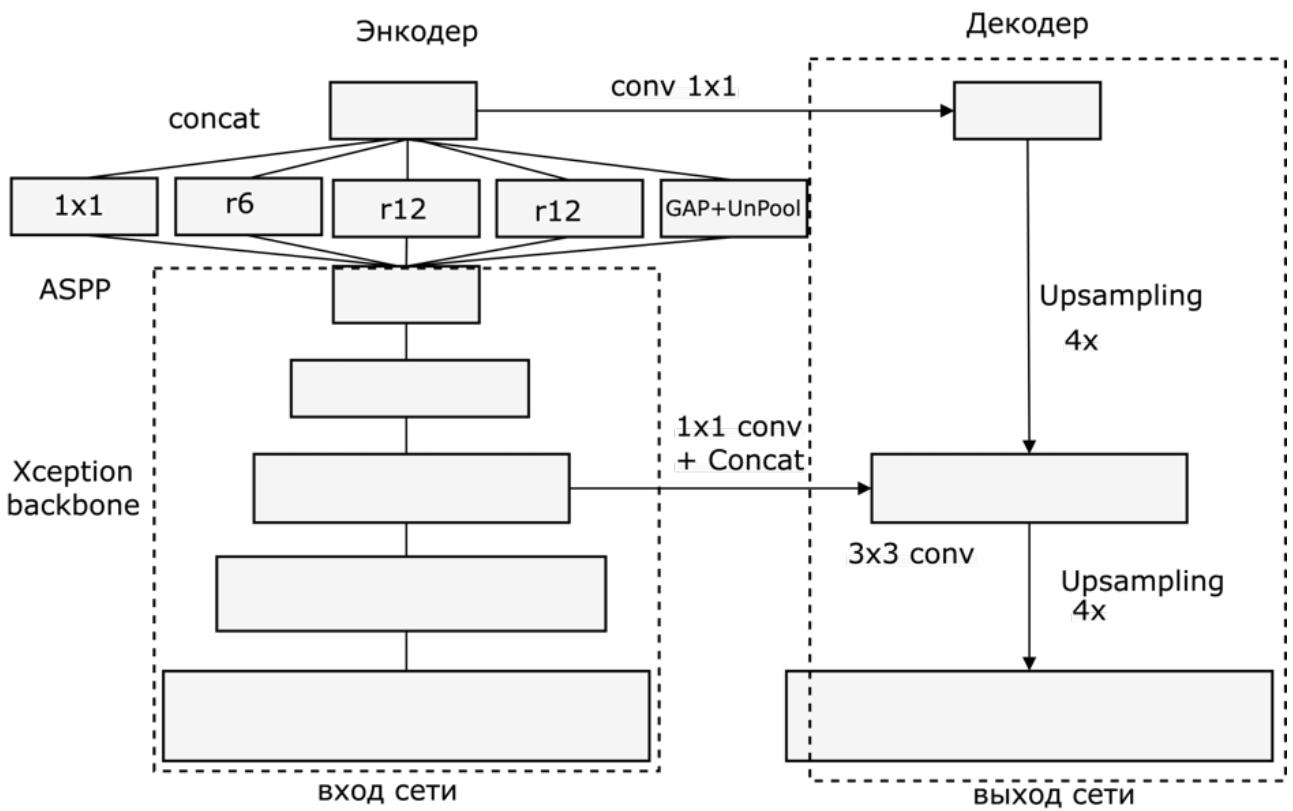


Рисунок 7.92 – Иллюстрация архитектуры DeepLab v3+

Результат объединения подходов энкодер-декодер и ASPP блока в архитектуре DeepLab v3+ дает эффект повышение резкости границ классов. Это особенно важно в случае небольших объектов. Кроме того особенностями DeepLab V3+ являются использование глубокой разделенной свертки как в кодировщике признаков, так и в ASPP блоке. То есть авторы предложили глубокий разделенный ASPP. Такой вариант расширенной свертки было предложено назвать Atrous Separable Convolution. В качестве кодировщика признаков в архитектуре DeepLab V3+ был использован модифицированный кодировщик модели Xception. Такая модификация была названа Aligned Xception ее суть сводится к добавлению блоков в архитектуру и замене операций пулинг на увеличенный шаг в глубоких свертках. В декодирующей части операция расширения размерности выполняется при помощи

билинейной интерполяции. Данная операция сопровождается сверткой 1×1 перед операцией и сверткой 3×3 после. Первая свертка нужна для согласования числа карт признаков. Тогда как вторая свертка позволяет уточнить карты признаков, полученные после билинейной интерполяции. Архитектура DeepLab V3+ с остается популярной и сегодня.

Архитектуры на основе DeepLab 3+ Архитектура DeepLab 3+ стала вдохновением для большого числа исследователей. Так, например, на основе подхода DeepLab 3+ также были предложены, например версия полученная путем автоматического поиска методом NAS. Архитектура имеет название AutoDeepLab [278]. В этой архитектуре был слегка модифицирован кодировщик признаков и упрощена структура ASPP блока. Также предложена версия с использованием блоков внимания в кодировщике [279]. В работе [280] предложено модифицировать ASPP блок используя принцип DenseNet. При этом авторы объединяют особенности работ последовательного и параллельного включения блоков расширенной свертки. Иллюстрация предложенного авторами блока DenseAspp приведна на рисунке .

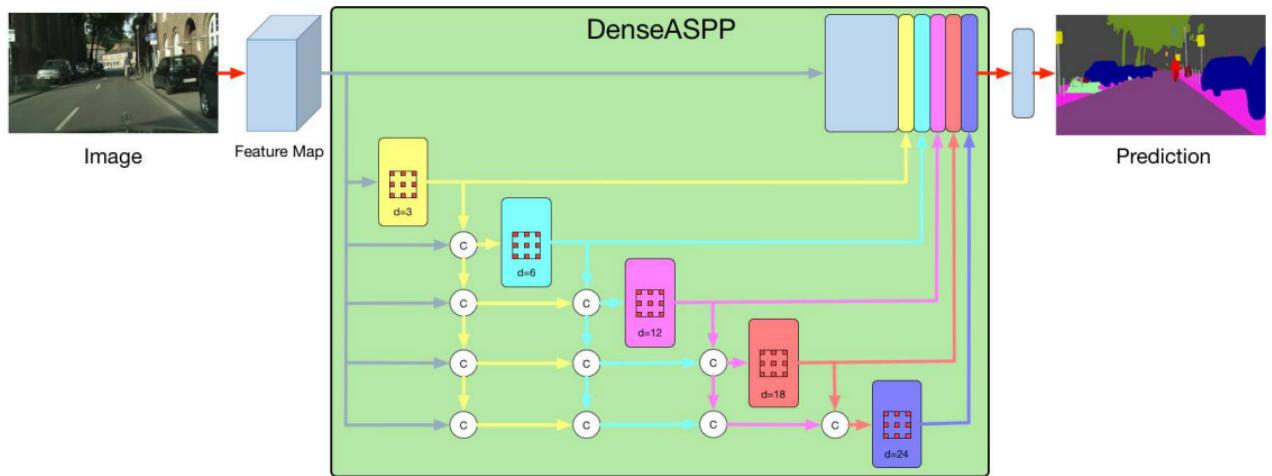


Рисунок 7.93 – Иллюстрация блока DenseAspp

Архитектура Pyramid Attention Network. В работе [281] авторы предложили модифицированную версию DeepLab, где вместо ASPP используется модуль на основе внимания для пирамиды признаков (Feature Pyramid Attention (FPA) Module). Архитектура получила название Pyramid Attention Network (PAN). Модуль пирамиды признаков включает несколько уровней, каждый из которых имеет ядро свертки уменьшенного размера. По задумке авторов это позволяет аналогично ASPP блоку выделять информацию о контексте классов на изображении с разным масштабом. Также блок включает ветвь Global average pooling branch, представляющую, по задумке авторов, функцию аналогичную SE слою. Авторы объясняют функцию модуля следующим образом. Модуль сливает вместе информацию о признаках с различным уровнем контекста и выполнять эффект внимания на уровне пикселей для высокоуровневых карт признаков. Также в работе предложен модуль

расширения изображения с глобальным вниманием (Global Attention Upsample). Идея модуля заключается в том, чтобы объединять признаки разных уровней при использовании некоторого аналога SE слоя. Так признаки высокого уровня сворачиваются при помощи GAP в вектор коэффициентов, который домножает признаки низкого уровня перед объединением. Иллюстрация архитектуры приведена на рисунке 7.94.

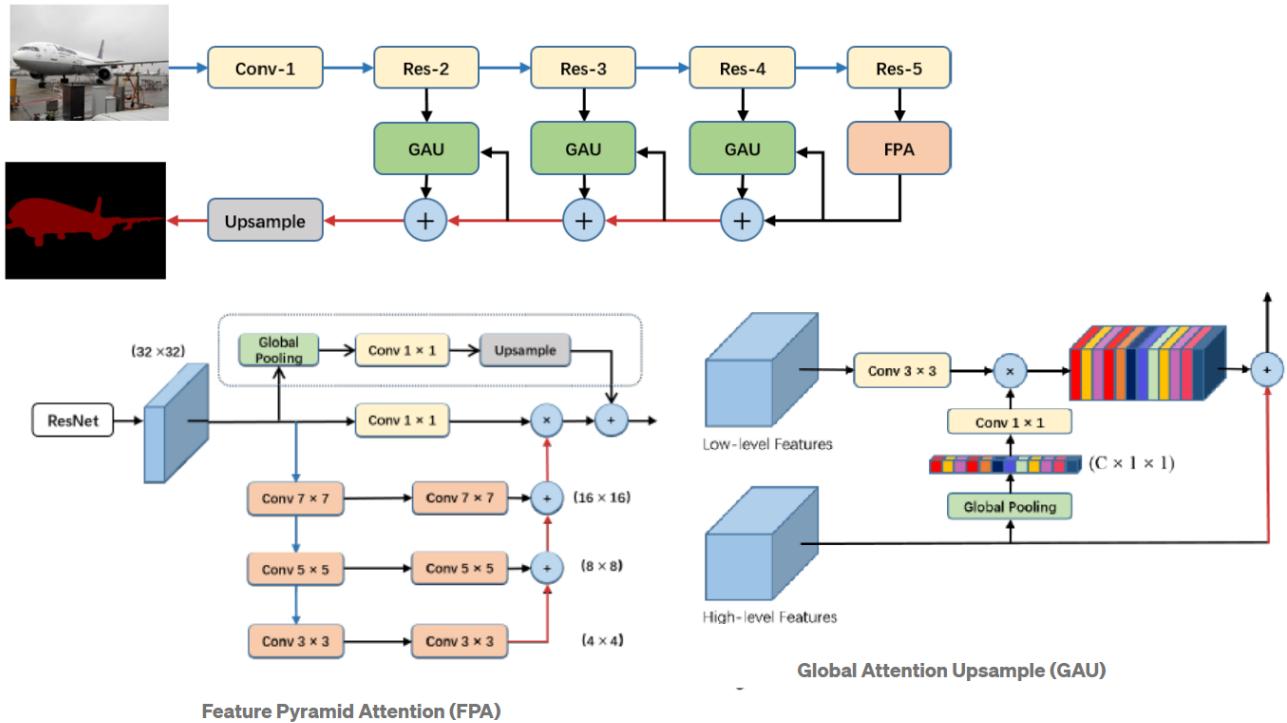


Рисунок 7.94 – Иллюстрация Архитектуры Pyramid Attention Network

7.5 Быстрые подходы к решению задач семантической сегментации

Как уже говорилось выше проблема семантической сегментации имеет два основных подхода к ее решению. Первый подход предполагает построение выходных изображений с помощью замены головной части архитектуры сети классификации. Второй подход основан на структурах кодер-декодер. Оба подхода объединены, например, в архитектуре DeepLabv3+ [277]. Эта архитектура включает экстрактор признаков Xception [157], часть декодера, подобную U-net, использующую расширенные свертки в скрытом пространстве. Многие современные быстрые архитектуры для семантической сегментации основаны на этих решениях [282]. Ниже приведены некоторые из наиболее важных решений. Архитектура RegSeg использует модифицированный блок SE-ResNeXt [283] в архитектуре, подобной DeepLabv3+. Авторы модифицировали подход RegSeg для решения задач быстрой сегментации. При этом была предложена новая модификация RegNet-Y блока и новый вариант кодировщика признаков. Также новый тип блока использован в декодирующей части. Иллюстрация особенностей архитектуры RegSeg показана на рисунке 7.95.

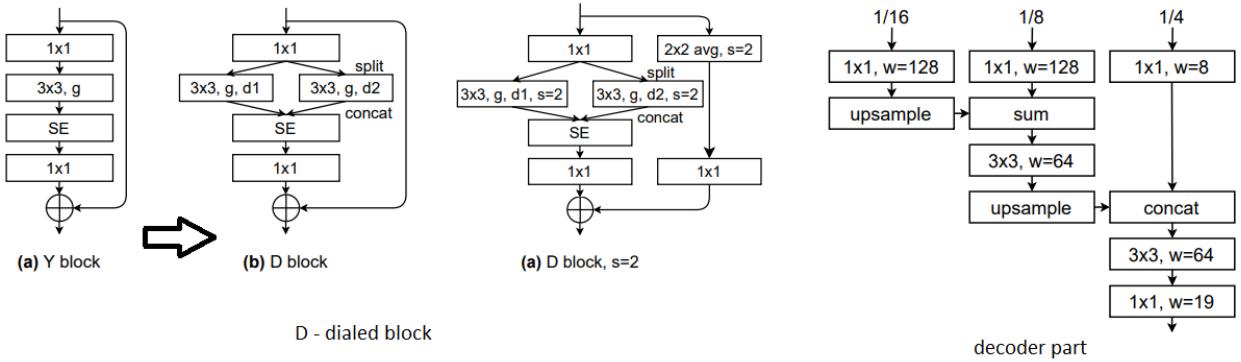


Рисунок 7.95 – Иллюстрации архитектуры RegSeg

Архитектура PP-LiteSeg [284] использует ряд приемов для модификации (облегчения) DeepLab3+ архитектуры. В архитектуре ASSP блок заменен на упрощенный SPP. В том числе, снижено число карт признаков, убрана остаточная связь, объединение заменено на суммирование. Для энкодера использована ResNet подобная архитектура, обученная методом дистилляции знаний. Также в архитектуре объединение признаков кодирующей и декодирующей частей происходит с использованием блока внимания. Авторы предполагают возможность выбора между пространственным и канальным вниманием. Однако в экспериментах авторы выбирают пространственный вариант. Иллюстрация архитектуры показана на рисунке 7.96.

Архитектура SFNet [285] предлагает использовать DeepLabv3+ подобную архитектуру в которой ASPP блок также заменен на SPP версию, а к блоку декодирования добавлена операция Flow Alignment Module. Авторы отмечают, что одна из проблем, с которой сталкиваются архитектуры типа энокодер-деркодер - это потеря информации о локализации восстанавливаемых областей. Таким образом для достижения высокой точности должен быть найден вычислительной простой механизм сохранения этой информации. Блок Flow Alignment Module предложен в качестве решения обозначенной проблемы. Основная идея такой операции заключается в объединение семантической информации (поток) между высокоуровневыми признаками декодера и низкоуровневыми картами признаков кодировщика, чтобы решить проблему несоответствия локальной информации внутри архитектуры DeepLabv3+. Иллюстрация архитектуры SFNet приведена на рисунке 7.97.

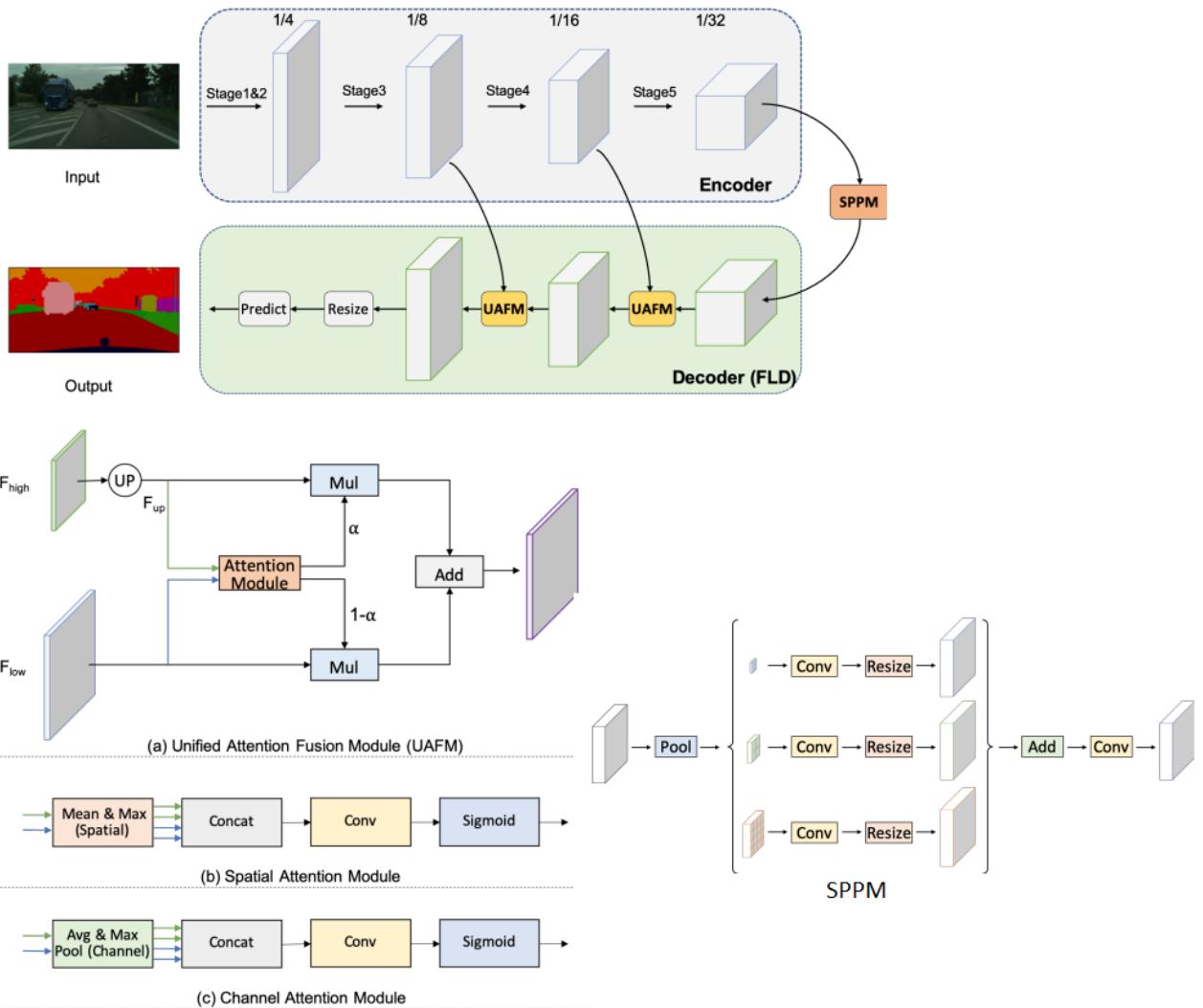


Рисунок 7.96 – Иллюстрации архитектуры PP-LiteSeg

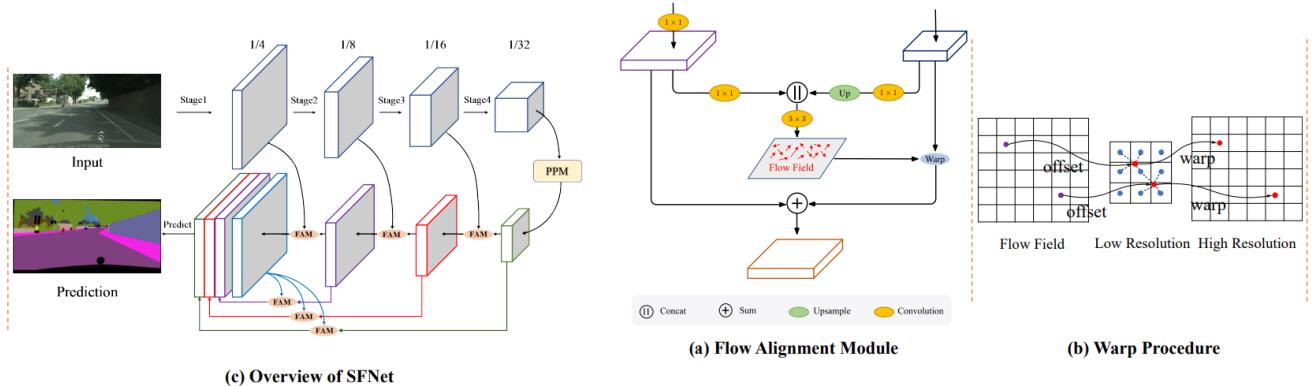


Рисунок 7.97 – Иллюстрации архитектуры SFNet

8 Задачи обнаружения объектов и экземплярной сегментации

8.1 Особенности класса задач

Общие замечания касательно задачи обнаружения объектов. С задачей обнаружение объектов связаны сразу несколько типа задач компьютерного зрения. К таким задачам относятся как само обнаружение объектов, так и схожие задачи. Например, могут быть следующие задачи.

- Задача обнаружение объектов (object detection) - поиск неизвестного числа объектов на изображении. При этом определяются их габаритные геометрические размеры и позиция, например, центральная позиция. То есть каждый объект вписывается в рамку.
- Задача классификация + локализация (classification + localization) – поиск одного или известного небольшого числа объектов на изображении. Задача значительно упрощается по отношению к первой.
- Задача экземплярной сегментации (объектная сегментация, instance segmentation) – поиск и сегментация неизвестного числа объектов на изображении. Производится как поиск позиций объектов, так и их сегментация. В отличие от задачи сематической сегментации тут каждый объект представляется отдельной маской.
- Задача обнаружения ключевых точек объекта (keypoint detection) - поиск особых точек, характерных для объектов. Например, может быть осуществлен поиск частей лица по особым точкам на нем или поиск конечностей человека для определения его позы или вида активности.
- Задача паноптической сегментации (panoptic segmentation) - поиск и сегментация объектов на изображении, а также семантическая сегментация всего остального изображения. Например, при распознавании движения автомобиля важным будет обнаружение других участников движения как объектов, а полотна дороги семантически.

Иллюстрация особенностей указанных выше задач показана на рисунке 8.98.

Отметим некоторые особенности задачи обнаружения объектов.

- Задача предполагает решения двух проблем: классификации объектов и регрессии координат и размеров объектов. Функция потерь должна учитывать обе эти проблемы. Если задача обнаружения объектов дополняется другими задачами, например экземплярной сегментацией, то функция потерь должна учитывать и решение этой задачи.
- По существу, задача предполагает, что выход модели содержит указания на классифицированные регионы входного изображения, где содержится объект. Такие регионы соответствуют оконтуривающей рамки (или просто рамки), вписывающей объект в свои габариты. Такие рамки также называют bounding box.

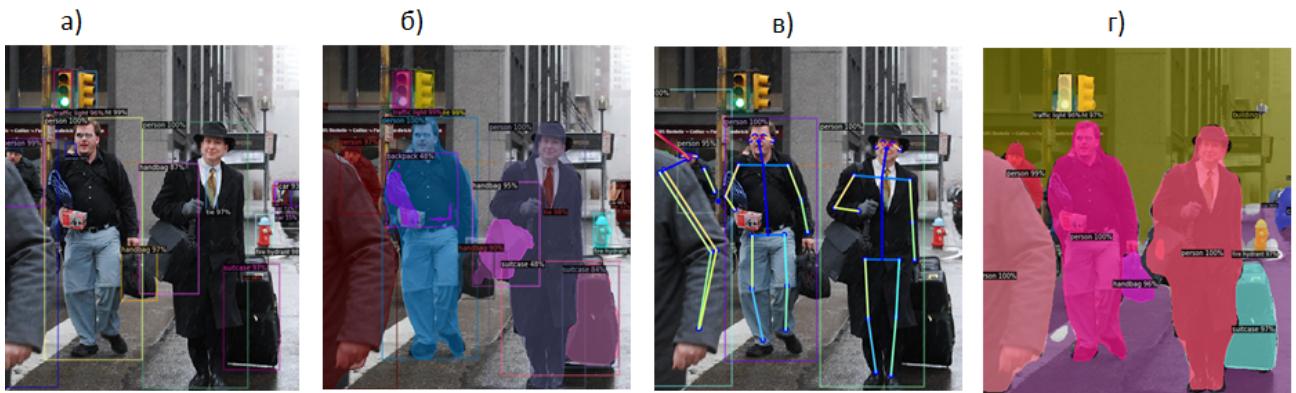


Рисунок 8.98 – Иллюстрации особенностей задач связанных с принципами обнаружения объектов: а) обнаружение объектов (также аналогична задача классификация + локализация); б) задача экземплярной сегментации; в) задача обнаружения ключевых точек объекта; г) задача паноптической сегментации.

- Регионы в которых потенциально может находиться искомый объект будем называть регионы потенциального интереса или region of interest, ROI.
- Во время работы модели для каждой позиции, где потенциально может находиться объект могут быть даны несколько оценок габаритных размеров оконтуривающей рамки такие предположения называются анхорами (anchor, anchor boxes).
- Для каждого объекта может быть установлен класс объекта, а также объектность. Объектность — это есть определение того, что в регионе находится именно объект, а не фон. Такая объектность также может называться уверенностью или score.
- Потенциальное число объектов на изображении, как правило, не известно. Поэтому выход модели указывает на избыточное число предполагаемых регионов, где может быть объект.
- В силу вышесказанного задача обнаружения объектов потенциально имеет дисбаланс. Для компенсации дисбаланса могут быть рассмотрены несколько методов. Среди таких методов можно выделить методы:
 - использование функций потерь специального вида;
 - использование предварительного отбора регионов кандидатов;
 - учет в расчетах функции потерь только регионов, для которых, например, объектность (уверенность в наличии объекта, предсказанная моделью) выше заданного порога.
- В зависимости от особенностей выбора метода отбора регионов кандидатов архитектуры обнаружения объектов могут быть разделены на два класса.

- Многоэтапные подходы. В этих подходах в том или ином виде архитектура подразумевает предварительный отбор регионов кандидатов. Как правило, такие архитектуры имеют низкое быстродействие, но достаточно высокую точность, в том числе при работе с объектами небольшого размера.
- Одноэтапные подходы. В таких подходах архитектура не подразумевает промежуточных этапов отбора регионов кандидатов. Такие подходы, как правило, работают быстрее.
- Также следует отметить, что по существу, задача обнаружение объектов необязательно предполагает обрамление объектов в прямоугольник. Так, ряд работ предлагают обрамление и в более сложные выпуклые геометрические фигуры. Например, работа [286] использует 6-ти угольник. Этот путь в некотором смысле сближает подходы экземплярной сегментации, поиска ключевых точек и обнаружения объектов.

Алгоритм не максимального сжатия. Как уже сказано выше, модели обнаружения объектов могут иметь число выходов превышающее число объектов на изображении. То есть результаты будут дублироваться. Однако при этом, как правило, разные регионы могут иметь разный уровень уверенности классификации – то есть разный выход вероятности нахождения объекта в регионе (разный score). Для компенсации этого эффекта часто используется алгоритм не максимального сжатия. Суть этого алгоритма сводится к следующим операциям.

- Вход:
 - Список предложенных регионов,
 - значений score (объектность или классификация)
 - минимальное пороговое значение score чтобы считать регион не пустым
 - пороговое значение пересечения площадей чтобы считать два региона дублирующими (IoU, Intersection over Union)
- Выход: Список регионов с объектами.
- Алгоритм:
 - Выбор из исходного списка рамок со значениями score выше порога.
 - Вычисление для каждой пары оставшихся полученных рамок IoU попарно.
 - Из всех рамок с IoU больше порога остается та, для которой score максимальен
 - Процедура может повторяться итеративно пока не прекратится удаление рамок

Иллюстрация работы алгоритма приведена на рисунке 8.99. Отметим, что описанный алгоритм соответствует наиболее простой реализации NMS. В литературе предложены и более продвинутые версии этого алгоритма [287].

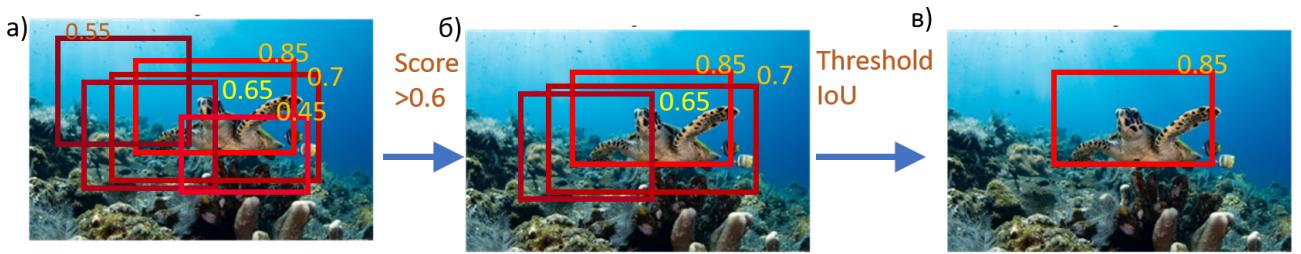


Рисунок 8.99 – Иллюстрации работы алгоритма NMS: а) входное изображение; б) изображение с рамками оставшимися после порога уверенности; в) итоговое изображение

8.2 Многоступенчатые архитектуры

Архитектура OverFeat Одной из первых успешных архитектур глубокого обучения сверточных нейронных сетей, решающих проблему обнаружение объектов при помощи только нейронных сетей была архитектура overfeat [288]. Иллюстрация работы данной архитектуры приведена на рисунке 8.100.

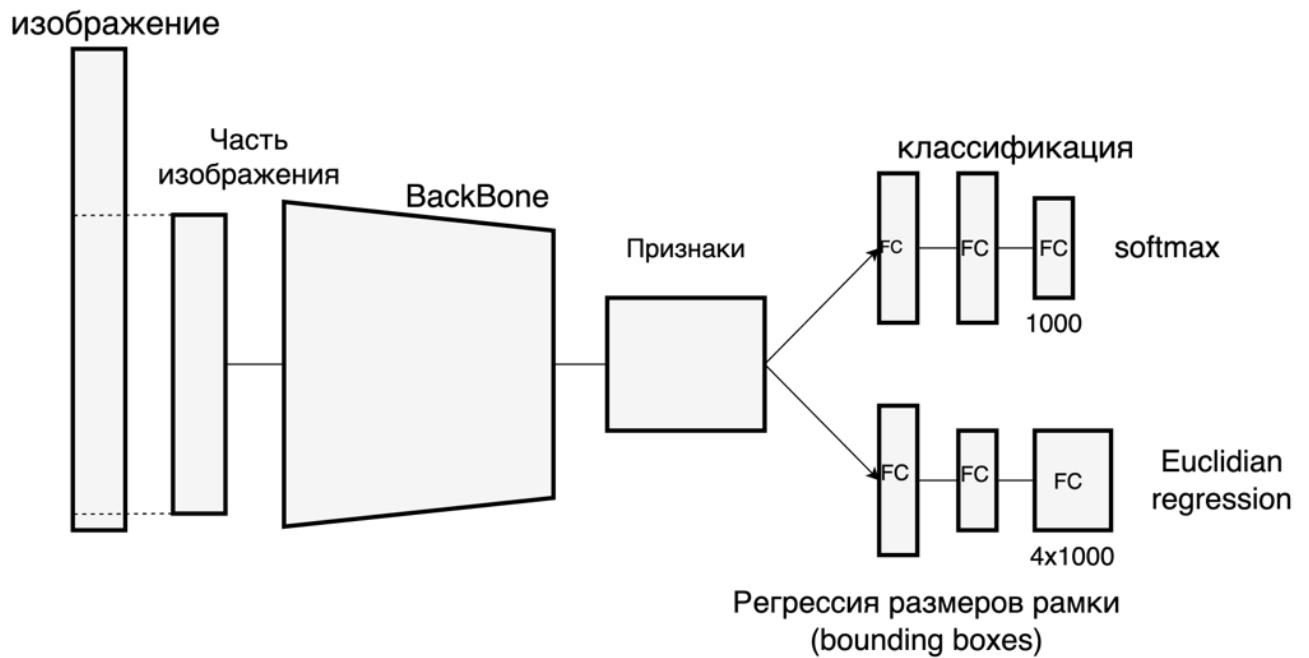


Рисунок 8.100 – Иллюстрации работы архитектуры overfeat

Метод Overfeat основан на идеи продвижение окна по изображению. У окна могут меняться размер и положение. В каждом состоянии окна из изображения извлекается соответствующая ему часть. Для извлечённой части происходит классификация объекта и предсказание его размеров. Таким образом, в рамках данной архитектуры обнаружение объектов — это одновременно и задача классификации, и задача регрессии. Отметим, что технически Overfeat — решает задачу локализации в каждом окне. Однако как можно догадаться перебор всех положений окна на изображении был бы слишком сложной задачей. Поэтому изначально сеть учится классифицировать объекты на изображении, затем учится предсказывать размеры и положение объекта для любого имеющего положения окна. Если

в это окно вообще попадает какая-то часть объекта. Состояние окна изображения, для которого найден объект называется **регион кандидат**.

Архитектура Region CNN (R-CNN). Одной из проблем архитектуры Overfeat была проблема перебора большого числа состояний окна а также сложность обучения: сначала надо было обучить сеть для классификации объектов, затем включить вторую головную часть – для регрессии и обучить ее. Частично решить эту проблему удалось в архитектуре регионарной сверточной сети или R-CNN [289], которая была предложена в 2014 году авторами Girshick R. и соавторов из университета Беркли.

Иллюстрация принципа работы архитектуры R-CNN приведена на рисунке 8.101. Сеть R-CNN осуществляет предсказание области нахождения объекта и его класса в несколько этапов - поэтому такой подход называется **много-этапное обнаружение (multi-stage object detection)** [289].

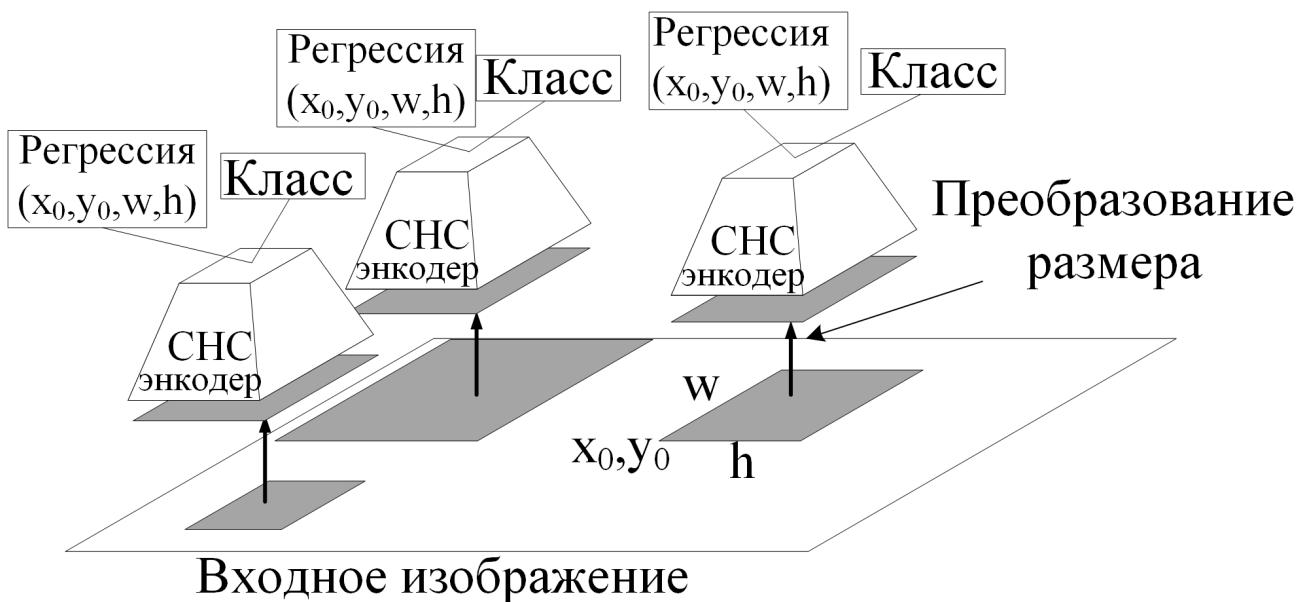


Рисунок 8.101 – Иллюстрация принципа работы архитектуры R-CNN

Архитектура R-CNN объединила в себе наиболее продуктивные для своего времени практики в области обнаружения и локализации объектов. Сеть R-CNN осуществляет предсказание области нахождения объекта и его класса в несколько этапов поэтому такой подход называется многоэтапное обнаружение. Основное отличие R-CNN от overfeat это подготовка перед началом обучения. На этом этапе производится выбор порядка 2000 регионов кандидатов, в которых предположительно есть объекты (Regions of Interest, ROI). Такой выбор регионов кандидатов быть выполнен любым алгоритмом, в том числе одним из алгоритмов классического компьютерного зрения. В оригинальной статье выбор регионов кандидатов было предложено проводить при помощи метода селективный поиск (selective search) – при помощи сегментации так называемыми суперпикселями, так как это показано на слайде. Этап подбора кандидатов – это 0 этап, он используется только как подготовка изображения и не включен в работу нейронной сети. Иллюстрация работы принципа selective

search показана на рисунке 8.102.

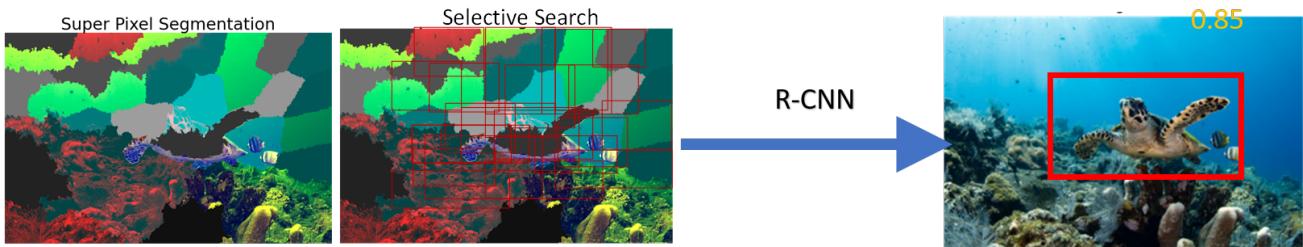


Рисунок 8.102 – Иллюстрация принципа работы selective search

Архитектура R-CNN работает следующим образом [289].

- 0 Этап - подготовка перед началом обучения.
 - Производится выбор 2000 регионов изображения, в которых предположительно есть объекты (**Regeons of Interest, ROI**). Авторы отмечают, что поиск может быть выполнен любым алгоритмом.
 - В оригинальной статье выбор регионов проводится методом селективный поиск (selective search). Это метод в котором выделяются контуры на изображении и проводится поиск объектов в рамках контуров путем оценки схожести соответствующих участков изображения [290].
 - Все выделенные регионы могут иметь разный размер. Выделенные регионы могут как содержать целевой объект, так и нет.
 - Отметим, что данный этап используется только как подготовка изображения к обработке нейронной сетью, то есть этап не включен в работу нейронной сети.
- 1 Этап - Автоматическое выделение признаков.
 - Производится сжатие всех выделенных регионов до одного размера.
 - Производится автоматическое выделение признаков (вектор 4096 значений). Выделение признаков производится при помощи сверточной нейронной сети AlexNet, в которой удален слой классификации - то есть при помощи кодировщика (энкодера) признаков AlexNet (см. 5.23).
- 2 Этап - классификация регионов.
 - Много-меточная (multilabel) классификация всех регионов ROI. Авторы используют для этого метод опорных векторов (SVM) отдельно для каждого класса.
 - Удаление лишних регионов методом **не максимальное сжатие (non-maximum suppression, NMS)**. В наиболее простом виде метод NMS заключается в следующем:
 - * для каждого класса производится поиск регионов с относительной площадью пересечения (Intersection over Union, IoU) больше заданного порога;

- * среди найденных регионов остается регион с наибольшей вероятностью классификации (с наибольшим результатом выхода классифицирующего слоя).
- 3 Этап - предсказание области новых регионов.
 - область нахождения каждого объекта описывается **границым прямоугольником (bounding box)**. Такой прямоугольник задается начальными координатами x_0, y_0 , а также шириной и длиной (w, h).
 - Для предсказания новых значений x_0, y_0, w, h для каждого ROI используется регрессионный оконечный слой с 4 выходами (по одному на каждый параметр). Как и в случае классификации слой используется поверх вектора признаков, полученного на 1-м этапе.
- В режиме тренировки нейронной сети для новых значений x_0, y_0, w, h производится следующая эпоха обучение нейронной сети (начиная с этапа 1).
- В режиме работы нейронной сети выходами являются объекты, прошедшие процедуру NMS, для которых присвоена метка класса и координаты bounding box: x_0, y_0, w, h .

Отметим, что в оригинальной статье [289] кодировщик признаков был предобучен на наборе данных ImageNet. Сеть была протестирована на наборе данных для решения задач обнаружения объектов PASCAL VOC [265]. Также отметим, что в литературе рассматриваются и исследуются различные модификации таких операций, как NMS и других, использованных в головной части R-CNN [291].

Архитектура SPPNet. Один из недостатков архитектуры R-CNN – это слишком большое время работы из-за необходимости оценки отдельно каждого из почти 2000 регионов кандидатов. Этот недостаток был решен в архитектуре Spatial Pyramidal Pooling Net или SPPNet [252] предложенной Кайменга Хе в 2015 году. В данной архитектуре авторами предложено проводить кодирование признаков для всего изображения, а операцию предварительного поиска регионов кандидатов проводить перед головной частью. Для всех выбранных регионов кандидатов применяется операция SPP.

Ранее мы уже обсуждали идею Spatial Pyramidal Pooling в отношении задачи семантической сегментации. Однако изначально этот прием был предложен для решения задачи обнаружения объектов. И так, основная идея SPPNet заключается в выделении признаков для всего изображения, а не для его отдельных частей и использовании пирамидального пулинга для поиска регионов кандидатов. Регионы кандидаты выбираются на каждой карте признаков при помощи селективного поиска. Выбранные регионы кодируются при помощи пирамидального пулинга. В данном случае под термином пирамидальный пулинг подразумевается несколько параллельных операций max-пулинг, выполненных с разным размером ядра. Фактически SPP блок делит каждый регион кандидат на несколько частей и ищет в них объект. Возможно, для поиска объекта достаточно одного значения на весь регион – тогда достаточно глобального пулинга. Если

нужна только четверть изображения – это тоже возможно, также мы выделяем 1/8 и 1/16 и так далее. Другими словами, цель такого пулинга - это учет одного и того же объекта с разным контекстом. Отметим, что селективный поиск и пирамидальный пулинг проводятся для каждой карты признаков. Иллюстрации работы пирамидального пулинга в архитектуре SPPNet и самой архитектуры приведены на рисунках 8.103.

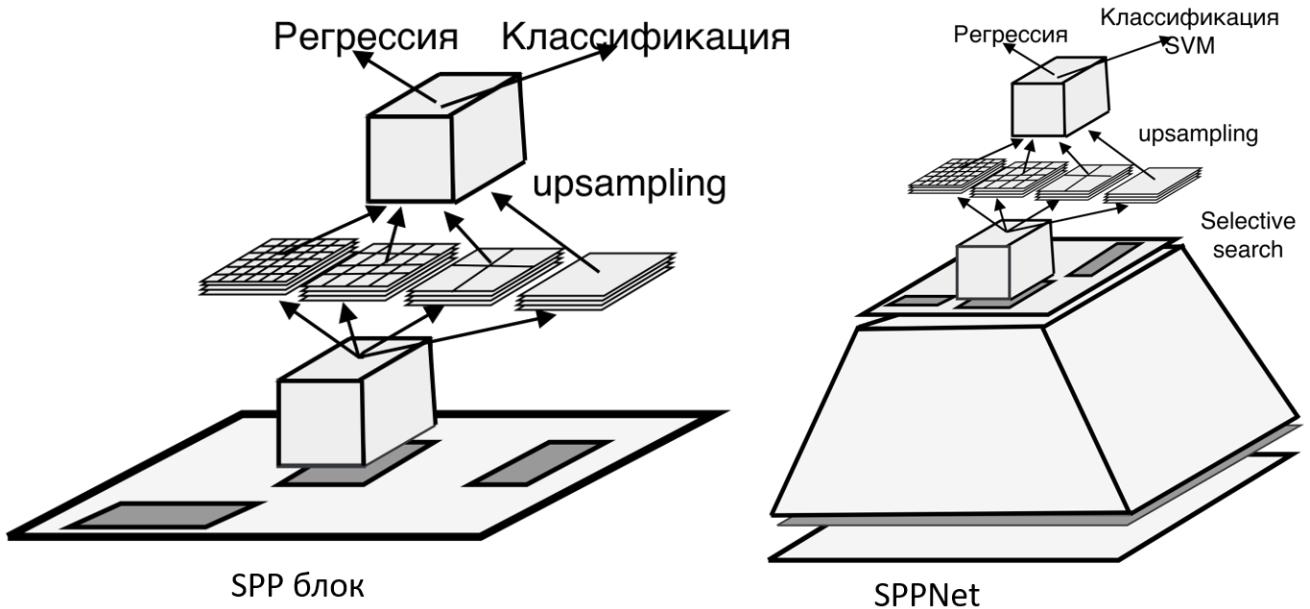


Рисунок 8.103 – Иллюстрации работы пирамидального пулинга в архитектуре SPPNet и самой архитектуры

Архитектура Fast R-CNN. В архитектуре SPPNet селективный поиск и пирамидальный пулинг проводятся для каждой карты признаков. Таким образом получается достаточно большой массив регионов кандидатов. В 2015 Girshick R. и соавторы предложили усовершенствованный вариант архитектуры SPPNet, называемый **Fast-RCNN** [292]. Так же как и SPPnet сеть использует операции selective search для результата работы кодировщика признаков. Однако вместо пирамидального пулинга сеть использует сжатие полученных регионов кандидатов до одного, заранее заданного размера. Эту операцию назвали ROI Pooling.

Принцип работы слоя ROI Pooling-а или пулинга регионов канатов показана на рисунке 8.104. Основная идея операции ROI Pooling заключается в том, что каждый регион кандидат трансформируется к заданному размеру при помощи сжатия методом макс-пулинга. Таким образом после операции ROI Pooling все регионы становятся одного размера, заданного заранее. Это позволяет использовать затем достаточно простую сеть для предсказания размеров и классификации. Отметим, что в Fast-RCNN вместо метода опорных векторов используется классический подход softmax классификации. Подход Fast-RCNN позволил сократить время работы сети в порядка 20 раз по сравнению с R-CNN. Иллюстрация архитектуры Fast-RCNN 8.105

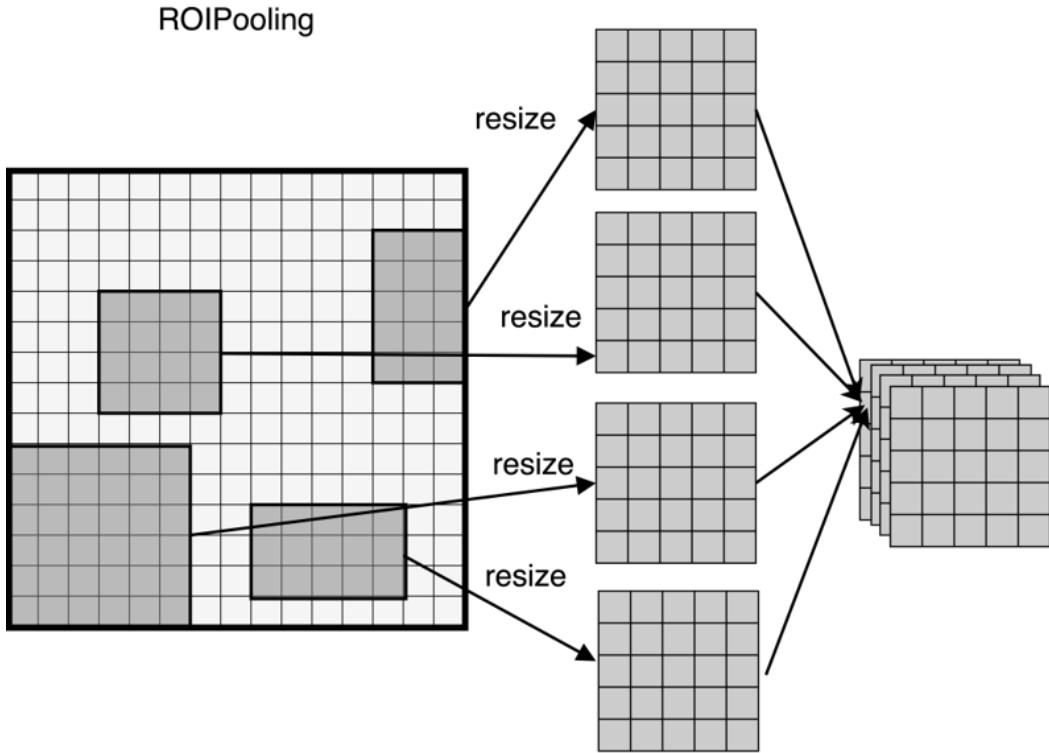


Рисунок 8.104 – Принцип работы слоя ROI Pooling

Архитектура Faster R-CNN. Позже в 2015 году в работе [293] коллективом авторов, включающим как Кайменга Xe, так и Girshick R. было предложено заменить селективный поиск на небольшую дополнительную нейронную сеть прогнозирующую регионы потенциального интереса (Region Proposals Network, RPN). Подход был назван **Faster-R-CNN**. Иллюстрация архитектуры Faster-R-CNN приведена на рисунке

Цель сети RPN – предварительное предсказания результатов по карте признаков. Результаты работы сети RPN – это значения координат и габаритов для регионов кандидатов. После работы RPN для каждого региона кандидата проводится ROI Pooling и дальнейшая обработка аналогично Fast-RCNN. В оригинальной работе RPN – это сверточный слой с размером ядра 3×3 . Сеть RPN действует по результатам набора карт признаков. Результат работы RPN – это так называемые анхоры для каждого предсказанного региона кандидата. Для каждого анхора результаты векторизуются и по ним определяются:

- утоненные координаты региона;
- проводится бинарная классификация на наличие объекта в регионе.

То есть RPN оценивает лишь объектность региона, а не класс объекта. Классификация сетью RPN – это промежуточный этап – цель такой классификации отобрать регионы, в которых в дальнейшем можно искать объекты. Для отсея лишних регионов кандидатов для результатов RPN проводится не максимальное сжатие (NMS). Иллюстрация работы сети RPN до операции NMS показана на рисунке 8.107.

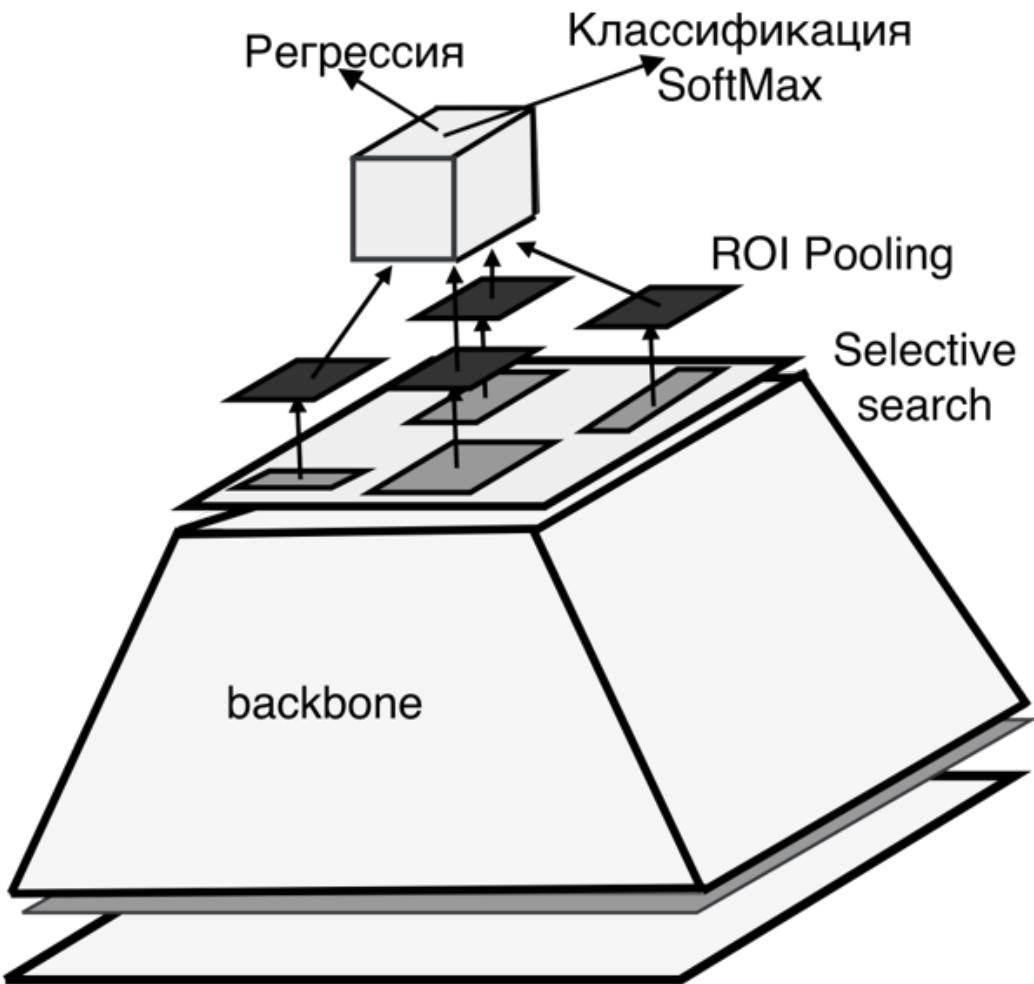


Рисунок 8.105 – Иллюстрация архитектуры Fast-R-CNN

В сущности, RPN решает задачу дисбаланса классов – то есть задачу того, что большинство регионов кандидатов оказываются пустыми. Для каждого положения окна фильтра RPN центральными координатами потенциального объекта считаются координаты центра фильтра. По результатам работы RPN генерируются по несколько анхоров с предсказанными параметрами. Размеры анхоров определяются коэффициентами масштаба и соотношений сторон, которые заданы заранее. В оригинальной работе для каждого предложения RPN предлагалось 9 анхоров отличающихся соотношением размеров сторон – так называемым aspect ratio и масштабом (scale). Масштабы задавались: 0,5 ; 1; 2. Соотношения сторон задавались 1:1; 1:2; 2:1. Подход Faster-R-CNN оказался в несколько раз быстрее предыдущей версии. Архитектура Faster-R-CNN до сих пор широко используется на практике. Как правило, архитектура используется с кодировщиком признаков на основе ResNet или его модификаций.

Архитектура Faster-R-CNN является одной из наиболее популярных архитектур сверточных нейронных сетей для решения задач обнаружения объектов. После опубликования статьи в 2015 было предложено много вариантов ее модификаций. Большая часть из этих модификаций связана с оптимизацией процессов ROI pooling и RPN частей. По существу некоторые из рассмотренных в данном обзоре архитектур, в частности R-

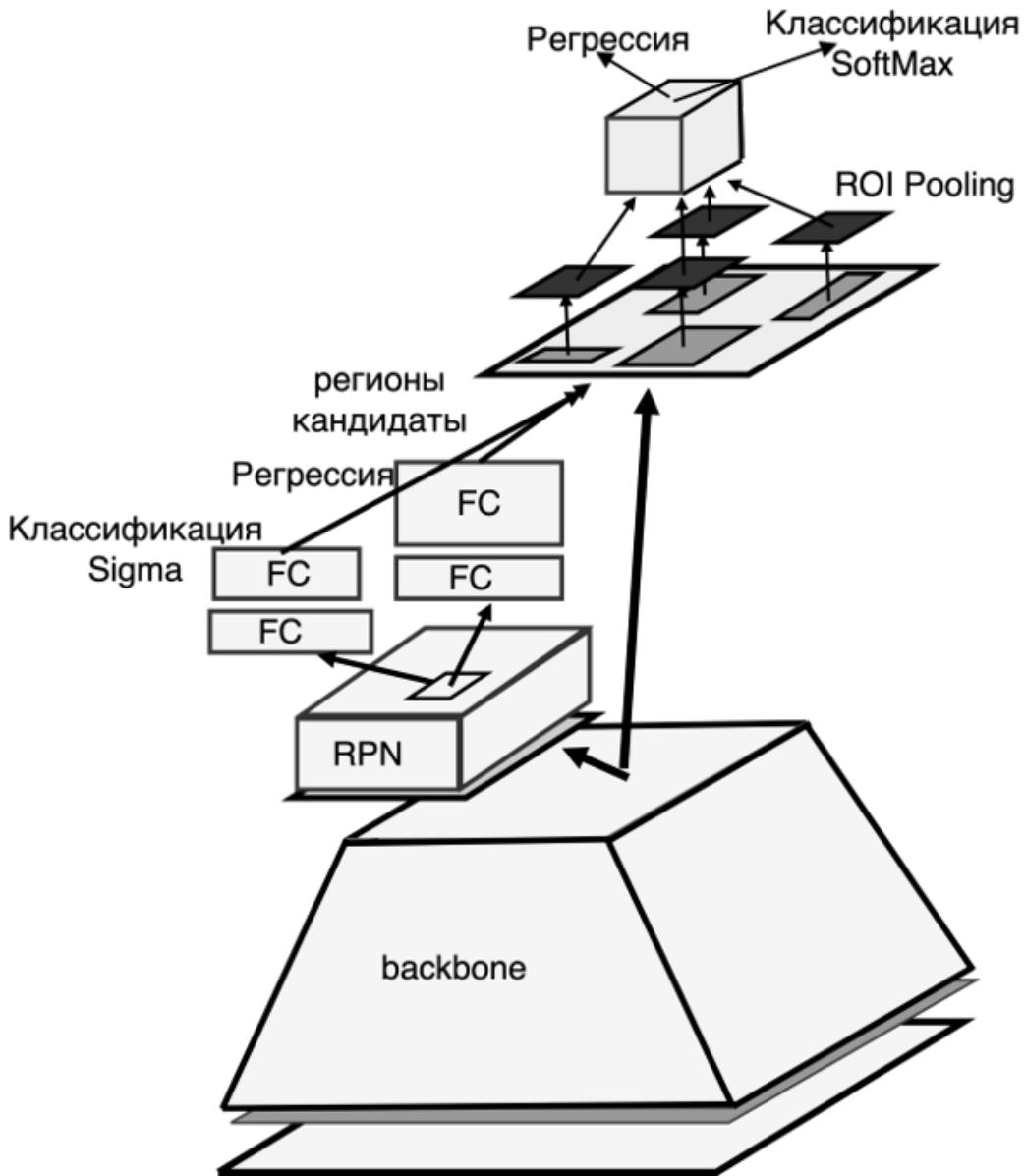


Рисунок 8.106 – Иллюстрация архитектуры Faster-R-CNN

FCN [294] и Mask-R-CNN [31] можно рассматривать как одни из таких модификаций. Так, например, в работе [295] архитектуры были упрощены. Кодировщик признаков был заменен Xception, число регионов потенциального интереса было снижено. Однако это не повлекло снижения точности на стандартных тестах. В работе [253] было предложено использование деформированной свертки на этапе ROI пулинга. Таким образом пулинг формировался из разряженных наборов пикселей вместо сгруппированных. Для этого помимо ROI формировался дополнительный набор карт признаков, предсказывающий соответствующие смещения. Иллюстрация работы архитектуры приведена на рисунке 8.108.

В работе [255] было предложено дальнейшее усовершенствование деформированного ROI пулинга. Авторы использовали вторую версию данной операции, которая включает амплитудную модуляцию. Также в данной работе авторы предложили архитектуру R-CNN Feature Mimicking. Основная идея этой архитектуры заключается в том, чтобы создать дополнительную суб архитектуру, которая бы обучалась на предложениях RPN

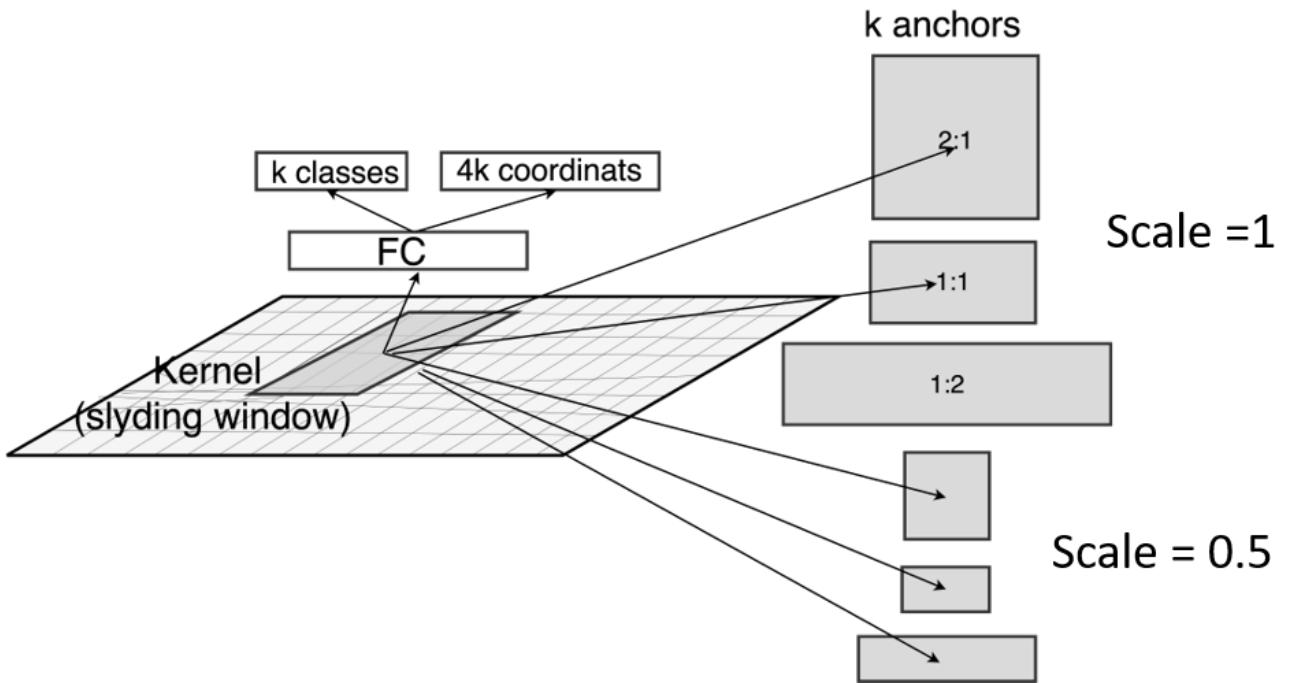


Рисунок 8.107 – Иллюстрация работы сети RPN до операции NMS

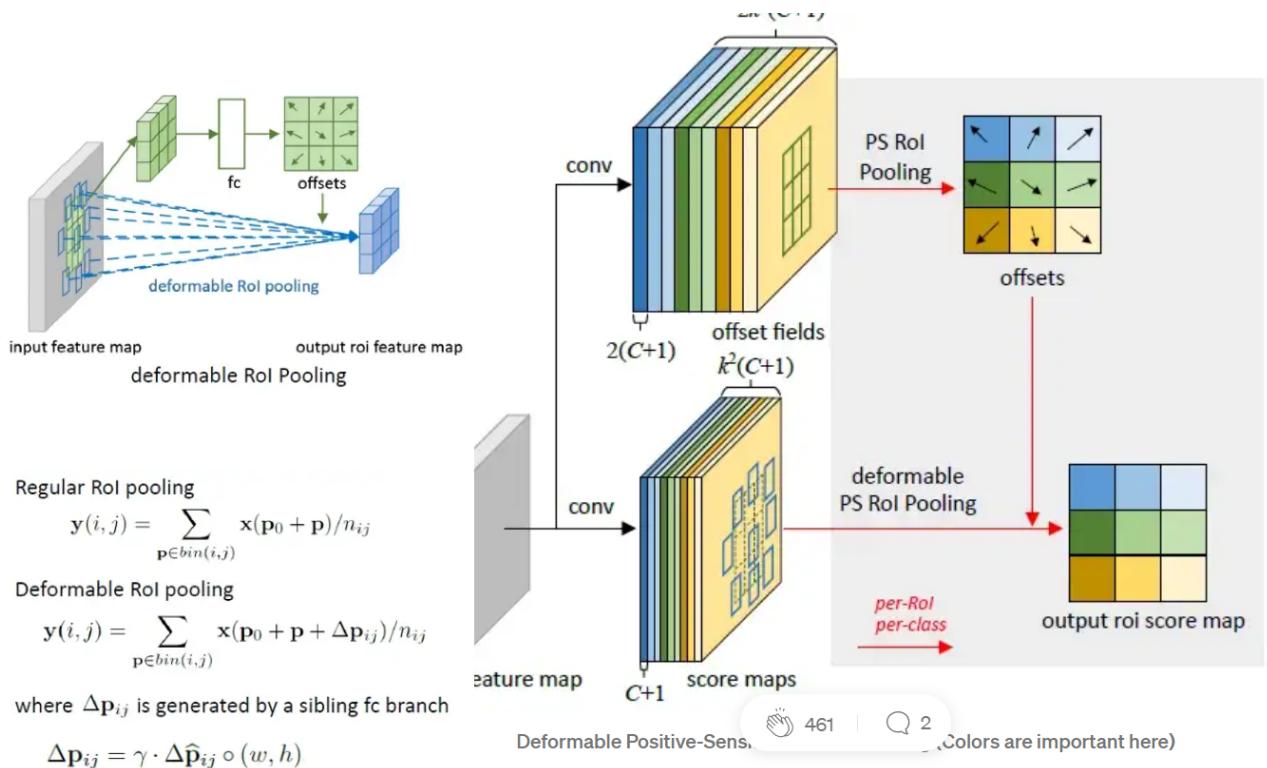


Рисунок 8.108 – Иллюстрация деформированного ROI пулинга.

сети и уточняла бы ее результаты. В предложенной архитектуре авторы использовали кодировщики признаков с деформированными свертками.

Архитектура пирамидальной сети признаков FPN. Авторами статьи [33] было предложено дополнение архитектуры кодировщика признаков в задачах обнаружения объектов пирамидальной шейной частью. По задумке авторов такая часть позволяет объединять признаки разных размеров. При этом следующая проблема рецептивного поля. Объекты небольших размеров лучше могут быть выделены кодировщиками небольшой глубины. Рецептивное поле при этом будет небольшим и особенностям таким объектам будет уделено больше внимания. При этом объекты больших размеров лучше будут выделены глубокими кодировщиками. При этом большое рецептивное поле позволяет учитывать как сами такие объекты, так и их окружение (контекст). Таким образом пирамидальная сеть предлагает способ объединения признаков разного уровня абстрактности (глубины). Иллюстрация архитектуры FPN приведена на рисунке 8.109.

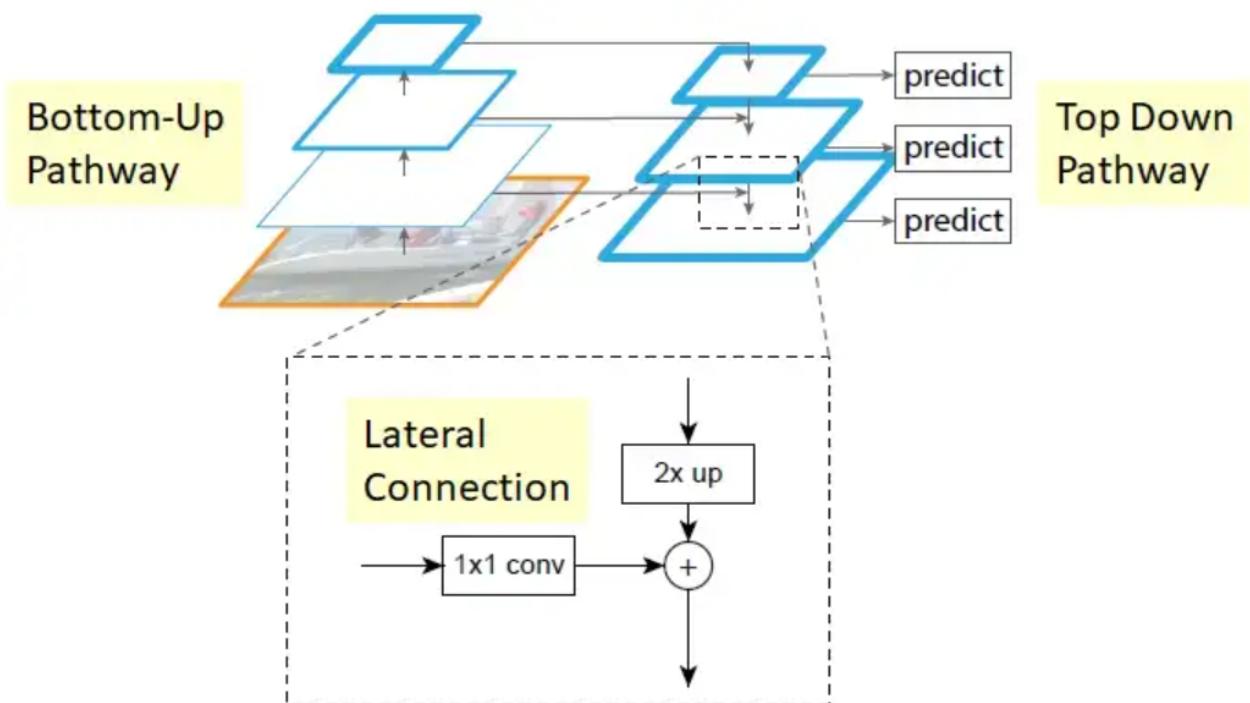


Рисунок 8.109 – Иллюстрация архитектуры FPN

Архитектура FPN объединяет признаки разного т.н. семантического уровня в шейной части по средствам повышения размерности и суммирования с более низким уровнем признака. Архитектура построена так, что каждый следующий выделяемый уровень должен иметь разрешение в 2 раза ниже предыдущего. Таким образом в низкоуровневых картах признака учитывается также и высокоуровневая (т.н. семантически слабая) информация. По задумке авторов это должно вводить баланс между способностью каждого уровня выделять признаки [33]. Также в шейной части могут быть использованы точечные свертки для управления размерностью выходных карт признаков. Также авторы предложили использование слоев 3×3 после процедуры повышения разрешения для компенсации эффекта алиасинга возникающего в этом случае. Отметим, что описанные решения были предложены не впервые, однако важным стало их объединение при решении задач класса

обнаружение объектов. Отметим также, что можно выделить три уровня карт признаков, которые следует обрабатывать в FPN части. Эти уровни могут соответствовать небольшим, средним и крупногабаритным объектам. Однако, иногда выделяют и больше карт признаков.

В оригинальной архитектуре авторы предложили использовать FPN в качестве меры модификации архитектуры Faster-R-CNN. Выделенные в шейной части карты признаков обрабатываются отдельно RPN сетью. По задумке авторов это позволяет отказаться от ручного введения коэффициента масштаба для анхоров. Анхоры разных масштабов образуются автоматически. Однако, это не снимает необходимость выделения анхоров с разным соотношением сторон [33].

Сегодня FPN используется широко для архитектур Mask-R-CNN/Faster-R-CNN и других подходов в решении задач обнаружения объектов и их сегментации. Иллюстрация архитектур Mask-R-CNN/Faster-R-CNN с FPN и RoiAlign показана на рисунке 8.110.

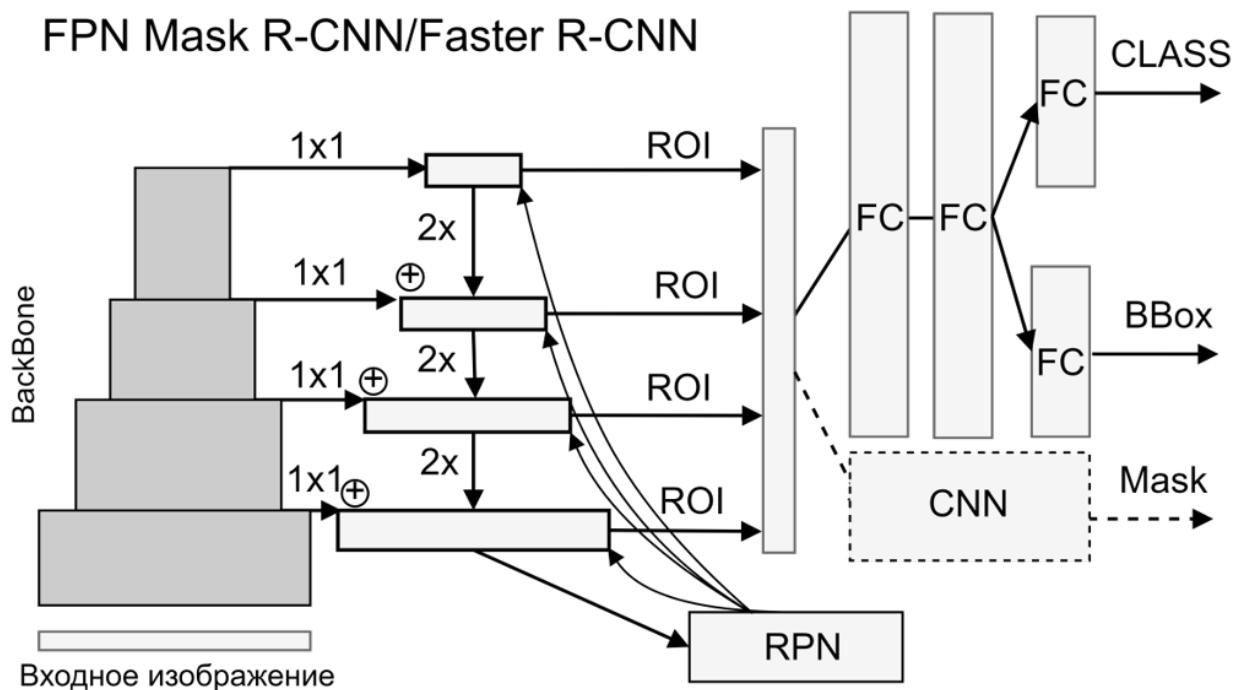


Рисунок 8.110 – Иллюстрация архитектур Mask-R-CNN/Faster-R-CNN с FPN и RoiAlign

Модификации пирамидальной сети. Подход FPN породил большое число работ, где авторы предлагали различные его модификации. Иллюстрация некоторых модификаций архитекторы FPN приведена на рисунке 8.111. Оригинальная FPN использует принцип "сверху вниз" (top-down pathway). Принцип был назван так в противовес естественному ходу кодировщика, названному bottom-up pathway. Различные модификации этого подхода включали идеи использования нескольких путей объединения признаков, приведем некоторые из этих модификаций:

- Подход "сверху вниз" (top-down pathway) (FPN) [33];
- Подход "сверху вниз" и снизу на верх (pannet) [296];

- Автоматизированный подход (поиск наилучшей конфигурации при помощи NAS (NAS FPN) [297];
- Двунаправленный подход (Bi FPN) [298];
- и многие другие.

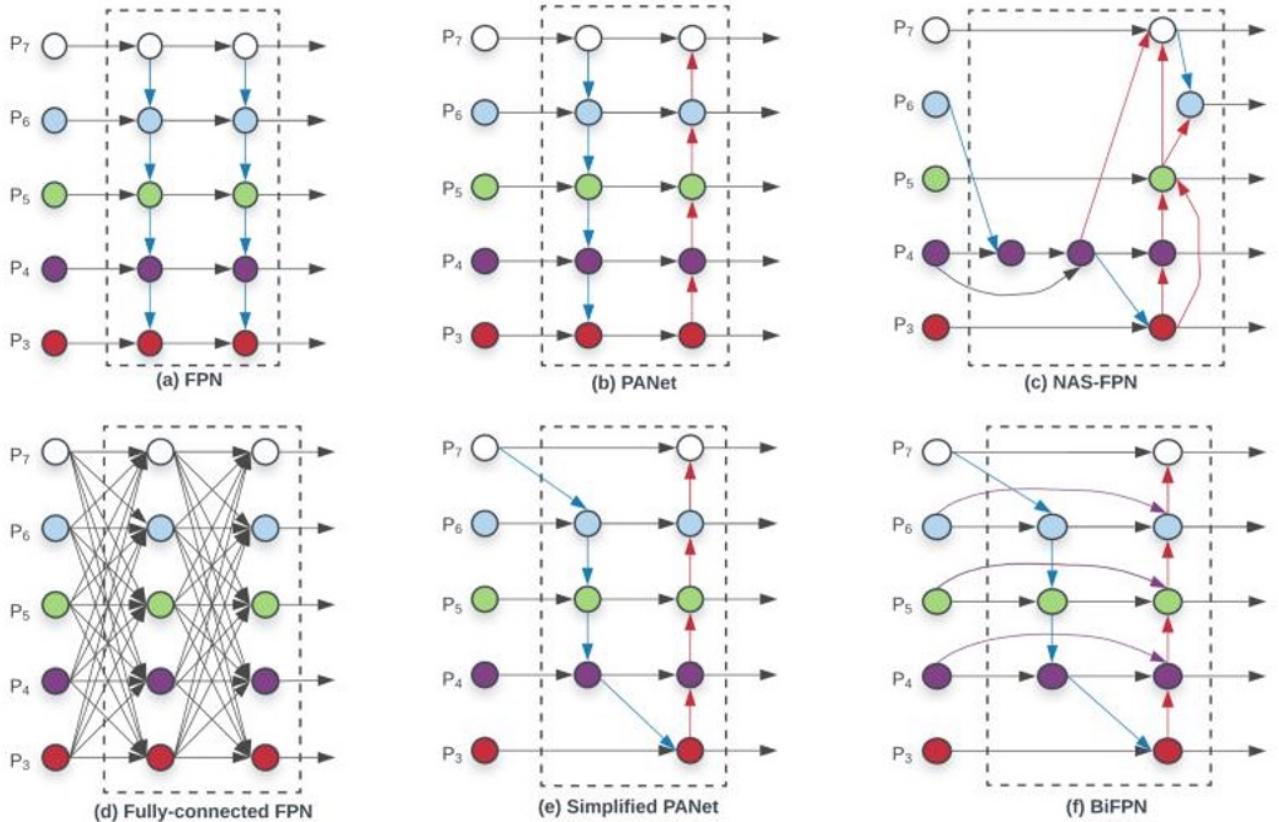


Figure 2: Feature network design – (a) FPN [16] introduces a top-down pathway to fuse multi-scale features from level 3 to 7 ($P_3 - P_7$); (b) PANet [19] adds an additional bottom-up pathway on top of FPN; (c) NAS-FPN [5] use neural architecture search to find an irregular feature network topology; (d)-(f) are three alternatives studied in this paper. (d) adds expensive connections from all input feature to output features; (e) simplifies PANet by removing nodes if they only have one input edge; (f) is our BiFPN with better accuracy and efficiency trade-offs.

Рисунок 8.111 – Иллюстрация архитектуры построения пирамиды признаков

Отметим, что в литературе предлагались и другие подходы построения FPN кроме освещенных выше. Как правило, FPN часть в подходах называется шейной частью.

Также возможна оригинальной представляется модификация подхода FPN, где используется несколько кодировщиков признаков таким образом, чтобы каждый последующий использовал как извлеченную в нем, так и в других кодировщиках. Идея этой архитектуры была предложена в работе [?]. Иллюстрация этого подхода приведена на рисунке 8.112. Последний из предложенных кодировщиков называется leader backbone. В этом кодировщике признаки формулируются в том виде, в котором будут использованы в головной части. Например, признаки могут быть объединены.

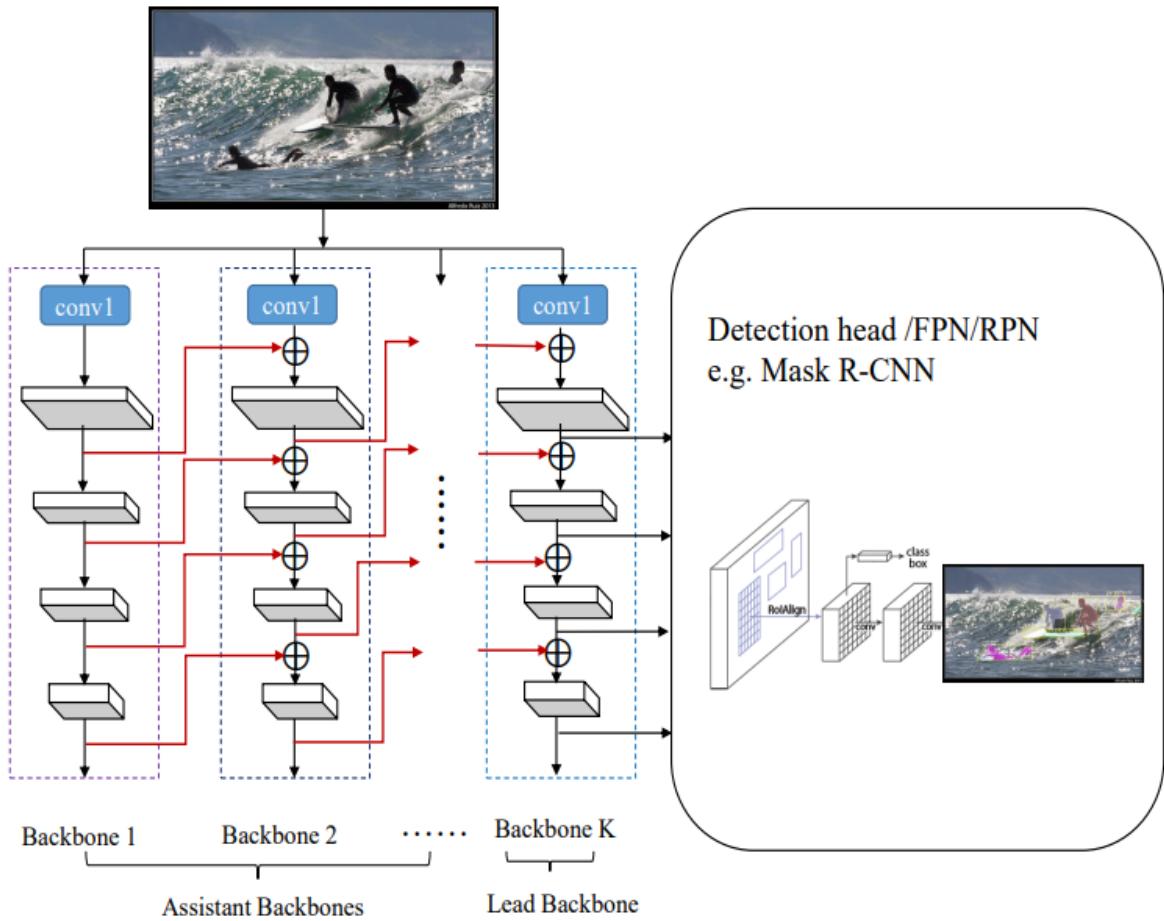


Рисунок 8.112 – Иллюстрация архитектуры CBNet

8.3 Одноступенчатые архитектуры

Архитектуры R-FCN. В работе [294] была предложена модификация R-CNN в которой были полностью исключены полно связные слои. Архитектура получила название R-FCN. Основная идея R-FCN это формирование результатов ROI пулинга в ходе обработки результатов работы кодировщика признаков. Подход получил название positive sense ROI или positive sense score map. Иллюстрация работы R-FCN архитектуры показана на рисунке 8.113 [294].

Рассмотрим принцип работы архитектуры.

- Кодировщик архитектуры обрабатывается сверточным слоем, который производит $k^2(C + 1)$ карт признаков. Где это число классов, а k это размер карты признаков, которая должна получиться после ROI пулинга. По сути этот этап аналогичен проведению RPN.

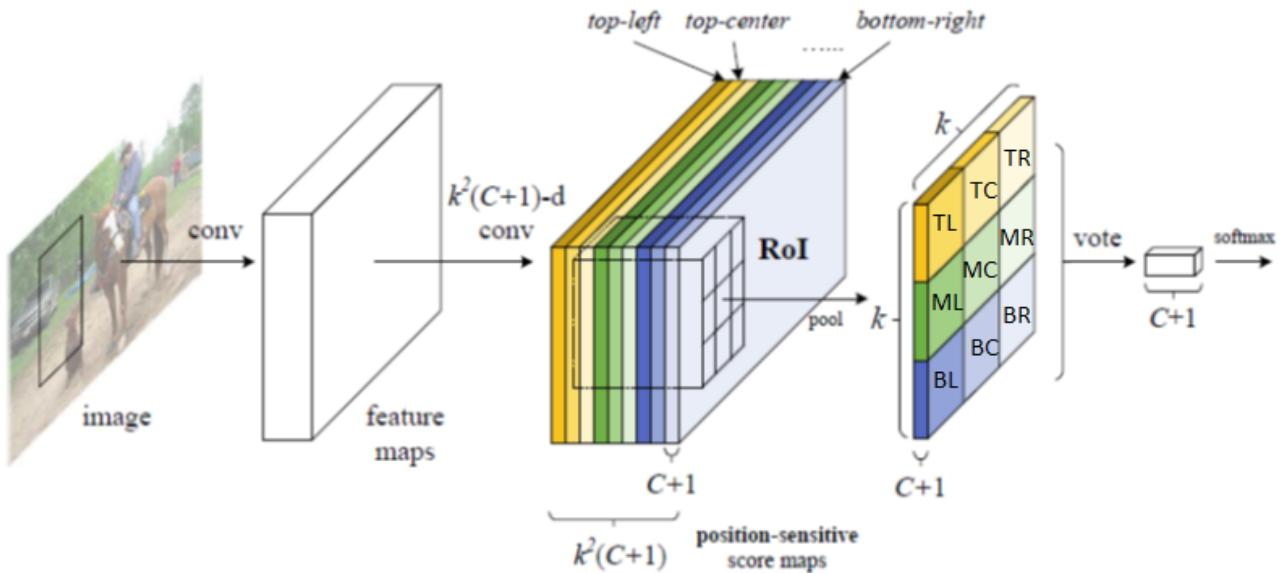


Рисунок 8.113 – Иллюстрация работы positive sense ROI блока и архитектуры R-FCN в целом

- Для сформированных $k^2(C + 1)$ карт признаков для каждого класса и фона получается по k^2 каналов (карт признаков). Эти карты признаков ассоциируются с пикселями потенциального ROI пулинга. Например, пусть результат ROI пулинга должен иметь размеры 3×3 . Тогда одна карта признаков создает верхний правый признак, другая средний и т.д.
- Полученные карты признаков перегруппированы согласно заданной интерпретации. Таким образом образуется $+ 1$ ROI результатов для каждого положения исходной карты признаков.
- Для каждого ROI считается средний score (выполняется GAP). Таким образом, получается вектор размером $+ 1$. Для вектора может быть применена операция softmax с целью получения интерпретируемого значения. Таким образом для каждого пикселя исходной карты признаков формируется оценка наличия объекта.
- Для проведения регрессии на этапе формирования $k^2(C + 1)$ карт признаков результаты кодировщика также обрабатываются сверточным слоем для получения $4k^2$ карт признаков. Эти карты интерпретируются как координаты и размеры для каждого объекта. Операция position-sensitive ROI pooling выполняется на этом наборе карт аналогично описанной выше процедуре. Таким образом для каждого положения окна формируется ограничивающая рамка. Иллюстрация архитектуры R-FCN с обеими головами показана на рисунке 8.114.

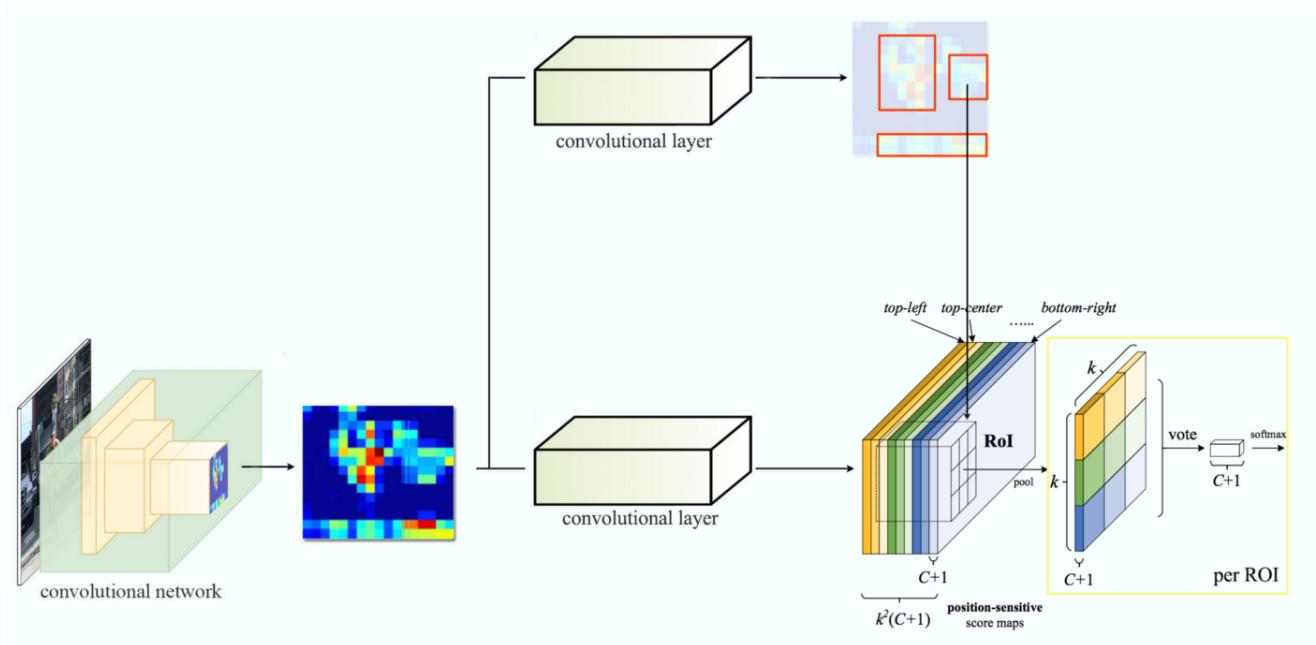


Рисунок 8.114 – Иллюстрация архитектуры R-FCN. Верхняя голова для формирования ограничивающих рамок.

Среди других особенностей данной архитектуры отметим, использование в ней онлайн отбора сложных примеров (Online Hard Example Mining, ОНЭМ) в ходе тренировки. Принцип ОНЭМ тут состоит в том, что из всех предложений регионов для расчета функции потерь оставляют только заданное количество ROI для которых score наибольший. На этапе работы архитектура использует NMS [294].

Архитектура RetinaNet В работе [299] авторами при участии Кайменга Хе была предложена архитектура RetinaNet. Эта архитектура представляет собой одноэтапную модификацию идею пирамедальной сети FPN. Архитектура RetinaNet использует FPN кодировщик признаков и два головных блока для каждого масштаба признаков. Одна голова решает задачу классификации, другая решает задачу регрессии. На каждом уровне кодировщика производится генерации нескольких анхоров для регионов кандидатов. Каждый анхор содержит предсказания размеров, положения, и класса объекта. Иллюстрация архитектуры показана на рисунке 8.115.

В архитектуре RetinaNet предлагает отказ от многоступенчатой структуры. Таким образом на выходе сети формируются все предложения объектов. Сеть не проводит промежуточных операций как, например, он-лайн отбор предложений ОНЭМ или не максимальное сжатие NMS. На выходе сети получается от 10 до 100 тысяч регионов кандидатов. Очевидно, что большинство регионов кандидатов пустые. Таким образом возникает задача с дисбалансом классов. В архитектуре RetinaNet эта задача решается особым способом [299].

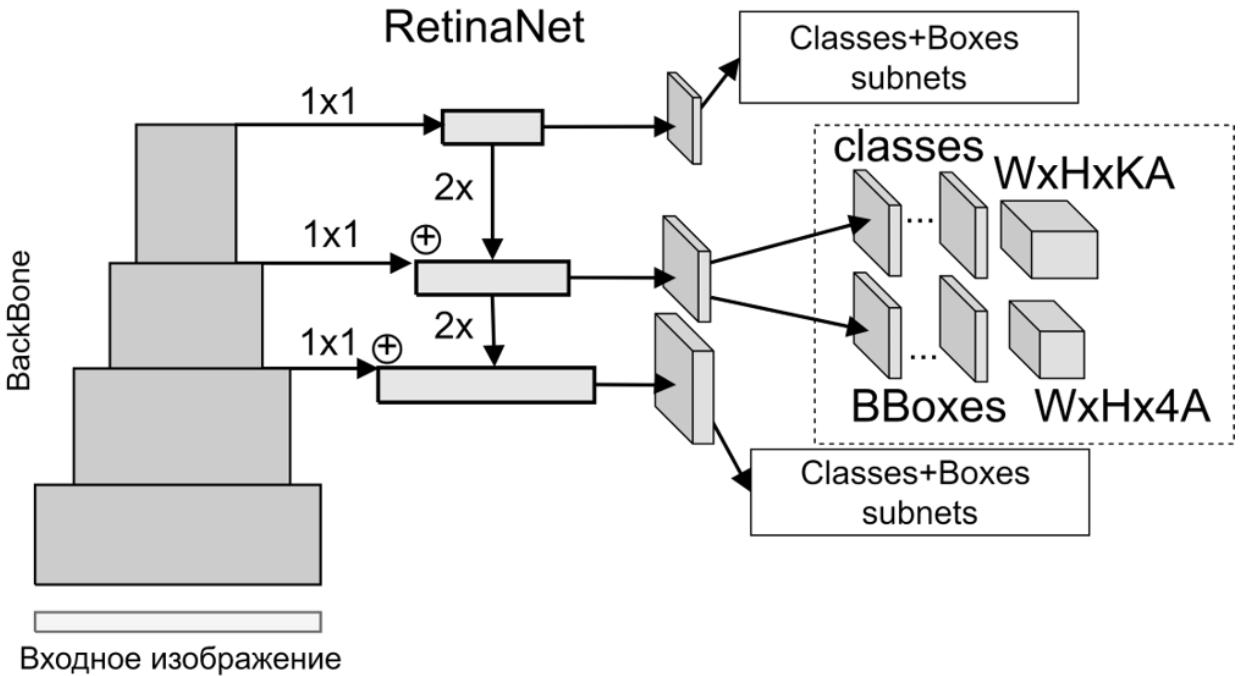


Рисунок 8.115 – Иллюстрация архитектур RetinaNet

Авторы RetinaNet предположили, что большую часть пустых анхоров сеть легко их классифицирует как фон. Такие регионы назвали простыми негативными примерами (easy negatives samples). Однако, если есть объекты, которые сложны для классификации, то сеть будет сложно выучить для них признаки. Такие объекты называют hard positive samples. Для балансирования между сложными объектами авторы RetinaNet предложили использовать функцию потерь focal loss. Эта функция уже рассматривалась выше. Фактически функция является некоторой оптимизацией кросс энтропии. Напомним, что функции потерь focal loss есть два гиперпараметра их значения выбираются обратно-пропорционально отношению числа сложно классифицируемых примеров к общему числу примеров для каждого класса. Общая функция потерь RetinaNet считается как линейная комбинация фокальной функции потерь для классификации и функции потерь для регрессии [299].

Также отметим, что одного focal loss для компенсации дисбаланса может быть недостаточно. По этой причине авторы RetinaNet также предложили исключать из результатов некоторые предложения. Например, предложения с пересечением площадей в некотором диапазоне, где сложно установить позитивные ли они или нет. Например, если есть предложения, которые имеют очень большое пересечение площадей между собой (выше заданного порога), то их стоит рассмотреть как потенциально позитивные для одного объекта. Если есть предложения, которые имеют пересечение площадей ниже заданного порога со всеми, то их можно рассмотреть как негативные. Остальные предложения можно отбросить. Для каждого потенциального предложения на каждом масштабе авторы RetinaNet используют по 9 анхоров с соотношениями сторон $\{1 : 2, 1 : 1, 2 : 1\}$ и масштабами $\{1, 2^{1/3}, 2^{2/3}\}$. Анхоры образуются путем обработки каждой карты признаков RPN сетью. Сеть RPN состоит из нескольких сверточных слоев 3×3 . В результате образуется KA карт,

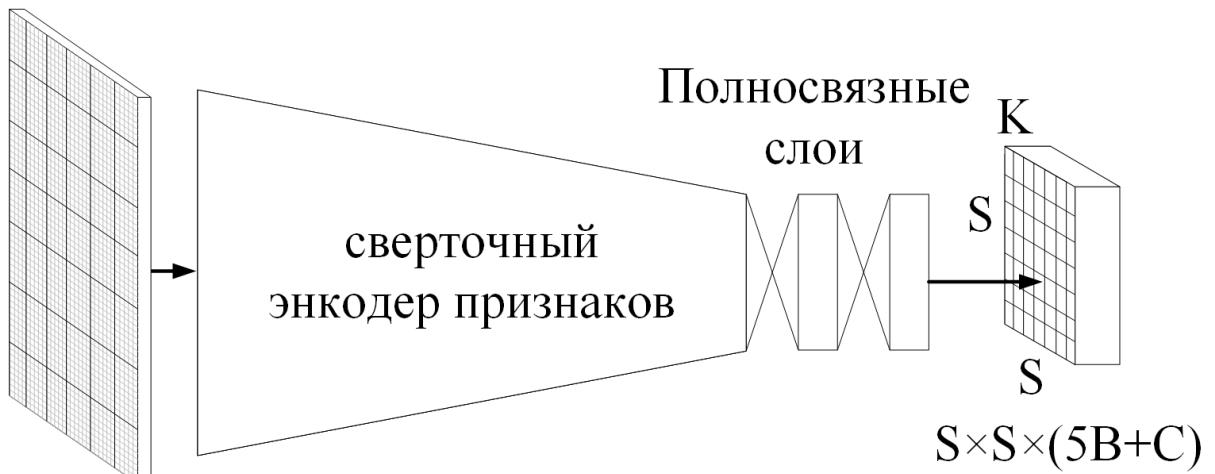
где K - число классов, а $A = 9$ анхоров. Аналогично происходит предсказание размеров. Тут RPN не использует полно связных слоев. То есть каждый пиксель выхода головной части ассоциируется с отдельным предложением [299].

На этапе работы сеть обрабатывает только порядка 1000 предложений с наибольшим score для каждого уровня FPN. При этом используется также и минимальный уровень уверенности. Для этих предложений используется алгоритм NMS [299].

Архитектура

8.4 Быстрые одноступенчатые архитектуры

Архитектура YOLO В 2016 году в работе [300] Редмондом Й. и соавторами была предложена "быстрая" - одноэтапная архитектура для решения задач обнаружения объектов. Архитектуру было предложено называть **YOLO** (you look only once). Архитектура YOLO состоит из энкодера признаков (модифицированного Inception GoogLeNet см. рисунок 5.29 [121]) и набора выходных слоев. Последний выходной слой содержит K каналов по $S \times S$ ячеек. По задумке авторов, каждая из $N = S \times S$ ячеек должна соответствовать определенной части входного изображения. Число выходных каналов определяется как $B \times 5 + C$, где B - число возможных боксов для каждой ячейки; C - число классов; цифра 5 - это значения предсказанных координат и размеров бокса (x, y, w, h) и уверенности в том, что в боксе есть объект. В оригинальной работе $S = 7, B = 2, C = 20$. Таким образом YOLO может предсказать до $N = B \cdot S \times S$ объектов, принадлежащих C классам. Иллюстрация архитектуры YOLO приведена на рисунке 8.116. Если один объект попадает на несколько ячеек, то для определения его координат используется метод не максимального сжатия по всем смежным ячейкам одного класса. Иллюстрация этого принципа приведена на рисунке 8.117. Сеть YOLO решает обе задачи обнаружения объектов (классификация и регрессия рамки) как регрессионную проблему. При этом все задачи: и обнаружение объекта и оценка его класса и координат решались одновременно при помощи сложной функции потерь. Также архитектура может предсказать лишь ограниченное число боксов для каждой ячейки. Таким образом, одним из основных недостатков подхода является невозможность разделения объектов с высокой степенью пересечения. Однако, скорость работы YOLO значительно выше, чем для Faster-RCNN [301]. В последствие были предложены различные модификации архитектуры YOLO [301], а также альтернативный вариант одноэтапных архитектур SSD (single short detection) [302]. На настоящее время архитектуры на основе YOLO являются наиболее точными для решения задач обнаружения объектов в реальном масштабе времени [303], а также одними из наиболее среди всех архитектур для набора данных COCO по данным портала paperswithcode.com [304].



Входное изображение

Рисунок 8.116 – Иллюстрация архитектуры YOLO

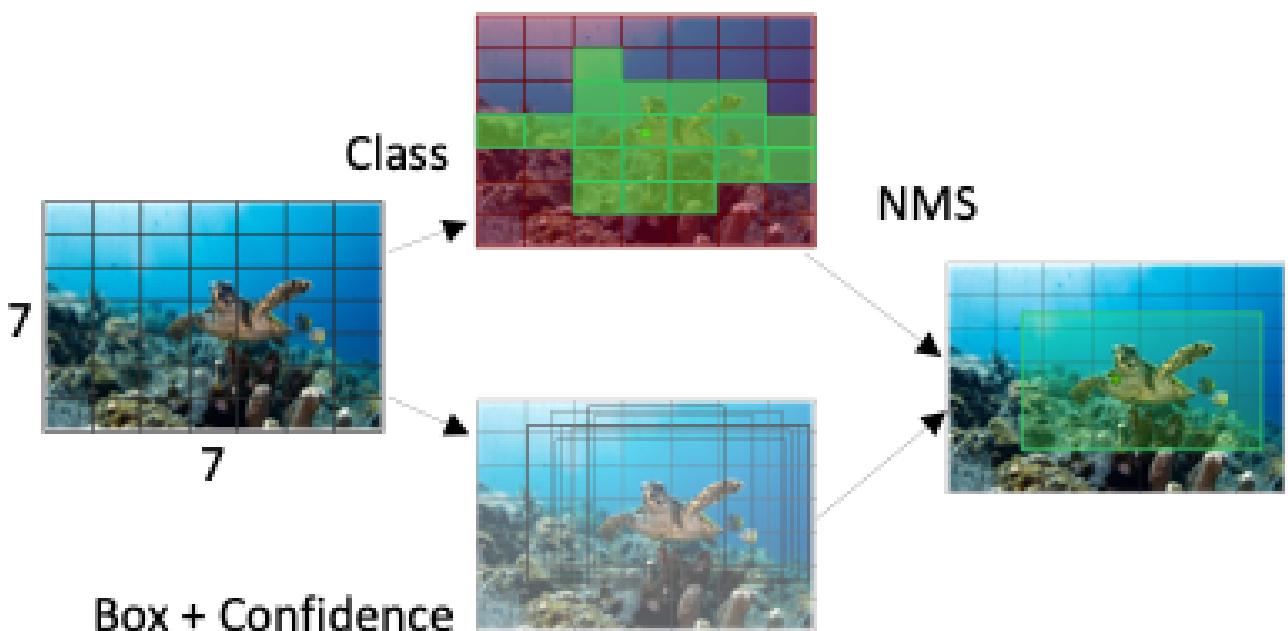


Рисунок 8.117 – Иллюстрация принципа работы архитектуры YOLO

Архитектура SSD В 2015 году была предложена архитектура много масштабного одноэтапного быстрого подхода. Эта архитектура была названа single short detector или SSD [302]. Основная идея подхода SSD – это одноэтапный поиск объектов на разных масштабах карт признаков или MultiScale object Detection. Эта идея аналогична пирамидальному подходу. Чем меньше размер карты признаков – тем более крупноразмерные объекты можно на ней выделить, но с потерей семантической информации. Чем больше карта признаков тем меньше размер потенциально выделяемого объекта. Иллюстрация подхода SSD показана на рисунке 8.118.

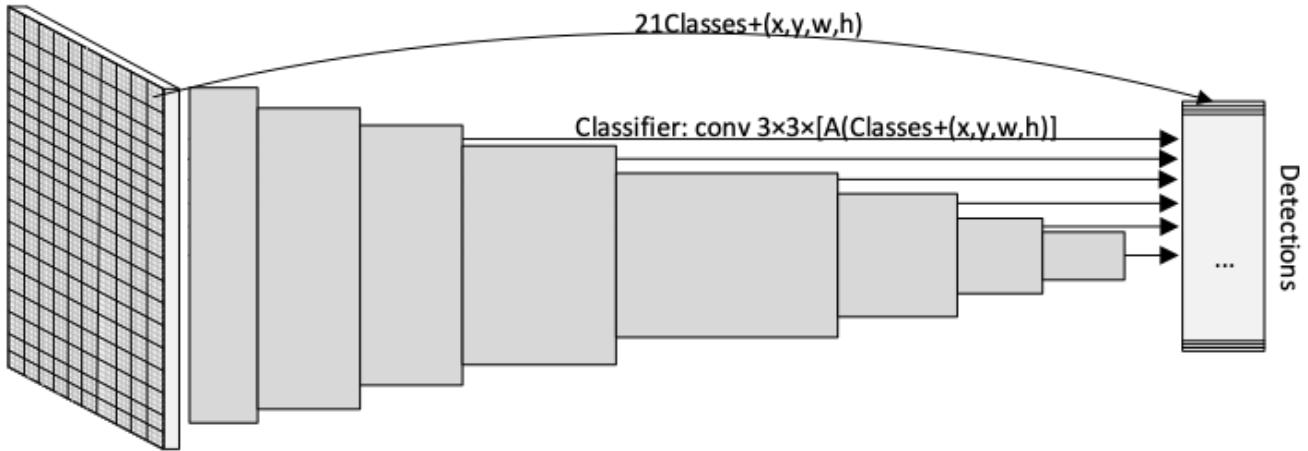


Рисунок 8.118 – Иллюстрация принципа работы архитектуры SSD

На каждом этапе работы архитектуры SSD для выделенных карт признаков используются свертки подобно RPN сети. То есть образуются анхоры. Каждый анхор включает - 21 класс (confidences) и 4 координаты рамки. Для разного масштаба карты признаков может быть разное количество регионов кандидатов . Параметры анхоров – масштаб и соотношения сторон выбираются, априори особым образом. Например, при помощи кластеризации тренировочного набора данных по размерам объектов. Всего сеть SSD выделяет порядка 8000 регионов кандидатов. Большинство регионов кандидатов пустые, (фон, negative samples) – то есть имеет место задача с дисбалансом. В архитектуре SSD предложили особый метод решения таких задач путем формирования батча специальным образом. Такой подход называн Hard Negative Examples Mining. Суть подхода заключается в том, что при расчете функции потерь отношение числа регионов с объектом (positive) к пустым регионом фиксировано, например, как 1:3. Среди пустых регионов выбираются те, для которых score максимальен. То есть точно пустые регионы. Основная идея этого подхода в том, что модели не нужно будет слишком стараться обучаться определять пустые регионы. За счет подхода виртуально создается ситуацию в которой пустые регионы определять очень легко. Тогда все обучение сети сконцентрировано на поиске регионов с объектами [302]. На основание идей SSD был предложен ряд архитектур, среди которых такие как DSSD [?] и ряд других. Также этот подход стал вдохновляющим для ряда других смежных исследований. Например, в работе [305] была предложена архитектура экземплярной сегментации MaskSSD на основе рассмотренного подхода. Однако, пожалуй в настоящее время этот подход уступил подходу YOLO.

Архитектура YOLO V2 Идея архитектуры YOLO второй версии приведена на рисунке 8.119 [306]. Основное отличие в архитектуре YOLO V2 в том, что полносвязные слои перво заменены на RPN слой. Это обычный сверточный слой, но он имеет смысл генерации анхоров. Также сеть содержит слой pass through, который сводится к учету признаков с двух уровней кодировщика. Также авторы увеличили размер входного изображения и число карт признаков. Таким образом итоговый тензор позволяет выделять порядка одной тысячи регионов кандидатов. Причем для каждого анхора теперь считаются как объектонсть

изображения, так и его класс. Как и в случае с SSD параметры анхоров задают путем кластеризации исходного набора данных. Также авторы предложили модификацию YOLO 2 позволяющую определять до 9000 классов связанных классов путем выбора квадратичной функции потерь для классификации и использования группировки классов по смыслу.

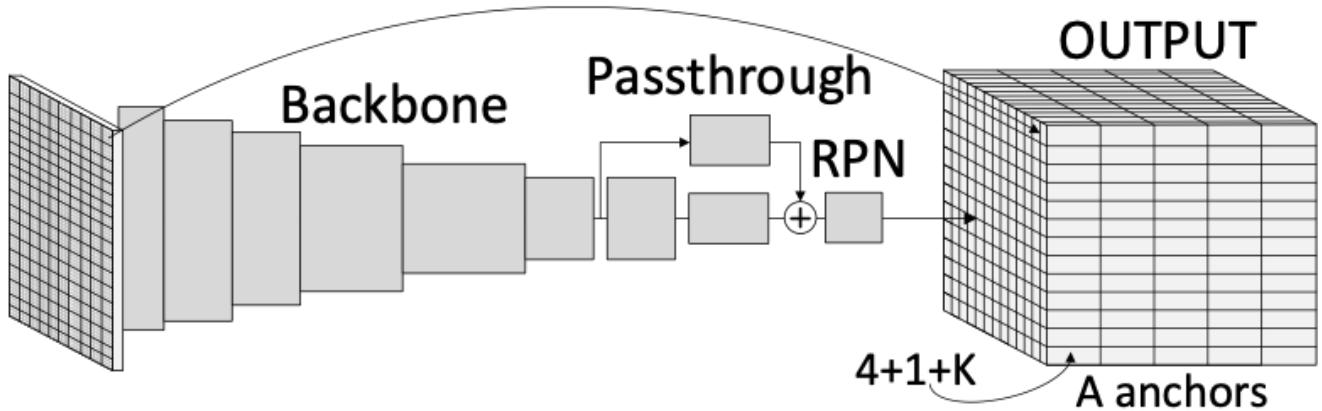


Рисунок 8.119 – Иллюстрация архитектуры yolo v2

Архитектура YOLO V3 В 3 версии архитектуры YOLO была полностью использована идея пирамидальной сети. Итоговая модель имеет три уровня признаков: для небольших объектов для средних объектов и последний для больших объектов. Как и в случае RetinaNet все признаки обрабатываются одновременно. Также в этой версии авторы отказались от подхода регрессии для решения задачи классификации и предложили прейти к многометочной классификации, то есть для каждой клетки изображения может быть обнаружено сразу несколько классов. Это может иметь значение, если, например объекты сильно пересекаются на изображении. Архитектура YOLO 3 стала последней предложенной лично автором первых версий. Иллюстрация архитектуры YOLO v3 приведена на рисунке 8.120 [307].

Архитектура YOLO V4,5 В 2019 году коллективом авторов под руководством Александра Бочковски [308] была предложена новая версия архитектуры YOLO. Архитектура YOLO v4 получила модификацию пирамидальной сети – шаг от другого подхода к обнаружению объектов Path Aggregation Network (PAN). По сути этот подход усложнённая версия пирамиды признаков. Также в архитектуру кодировщика были внесены наборы изменений. Помимо этого, авторы предложили набор приемов для обучения сети. Все это составило основу архитектуры YOLO v4, и семейства ее версий YOLO v4 scaled. Однако, эта архитектура имела ряд проблем, связанных с ее использованием. Эти проблемы были исправлены в пятой версии архитектуры - YOLO v5 это модификация 4 версии архитектуры, которая переписана для фреймворка pytorch с небольшими корректировками [309]. Иллюстрация архитектуры YOLO v4 приведена на рисунке 8.121.

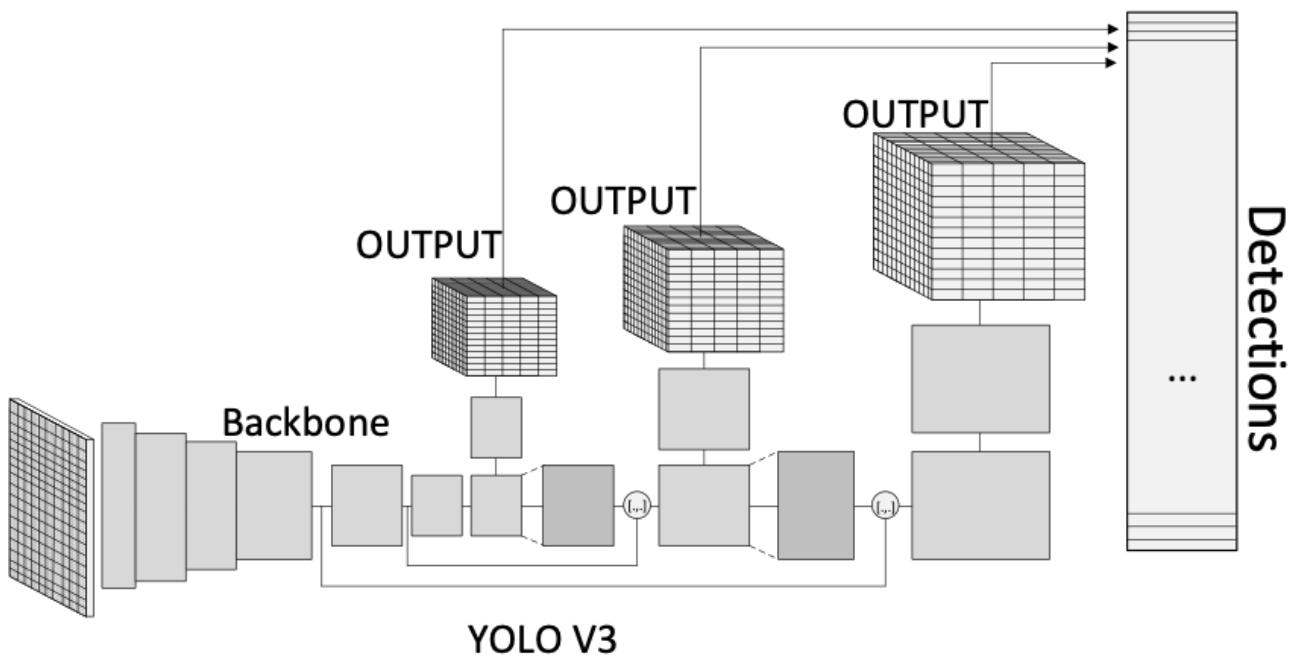


Рисунок 8.120 – Иллюстрация архитектуры yolo v3

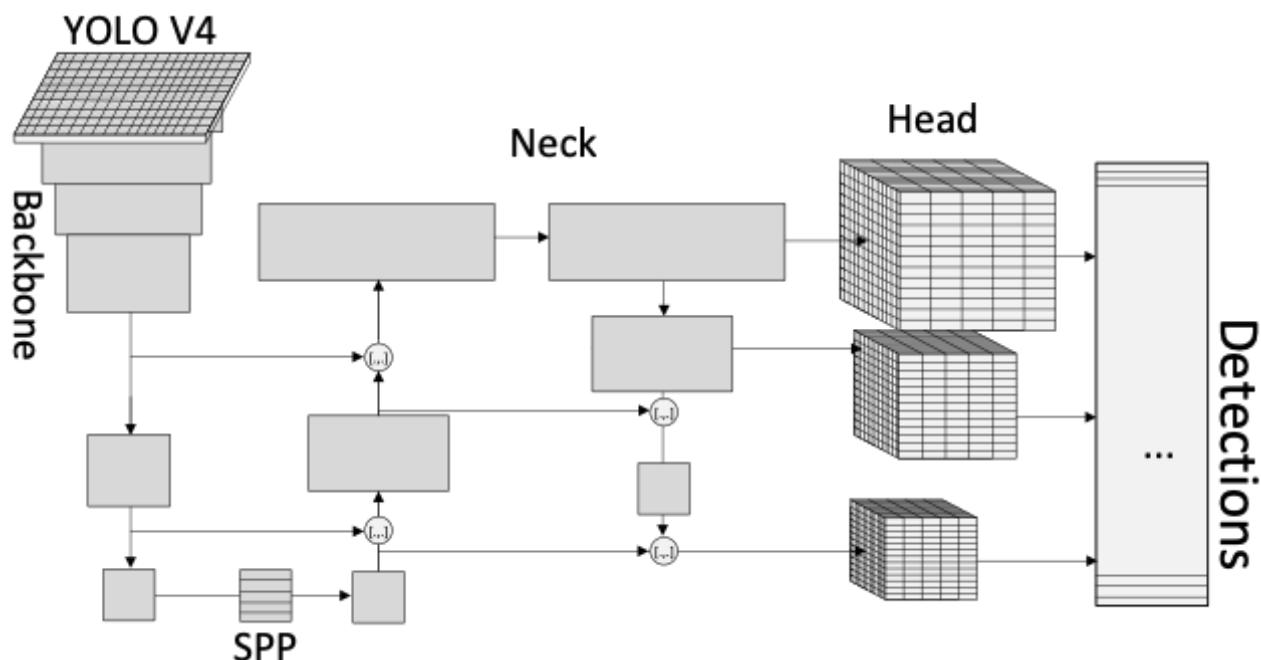


Рисунок 8.121 – Иллюстрация архитектуры yolo v4

Отметим, что архитектура YOLO v5 сегодня включает ряд модификаций с разными размерами от модели nano до L, также эта архитектура имеет ряд модификаций, в том числе, для решения задач классификации и экземплярной сегментации.

Архитектура YOLO 7 Архитектура YOLOv7 предложена автором 4й версии архитектур этого семейства [310]. В данной модификации предложен новый кодировщик признаков E-ELAN, а также оригинальный принцип масштабирования модели и обновленный набор приемов для обучения.

YOLOv7 surpasses all known object detectors in both speed and accuracy in the range from 5 FPS to 160 FPS and has the highest accuracy 56.8

From (a) to (b): It is observed that when depth scaling is performed on concatenation-based models, the output width of a computational block also increases. This phenomenon will cause the input width of the subsequent transmission layer to increase. (c) Proposed Model Scaling: When performing model scaling on concatenation-based models, only the depth in a computational block needs to be scaled, and the remaining of transmission layer is performed with corresponding width scaling. The proposed compound scaling method can maintain the properties that the model had at the initial design and maintains the optimal structure.

8.5 Многоэтапная экземплярная сегментация

Общие особенности решения задачи экземплярной сегментации. Напомним, что задача экземплярной сегментации (или объектной сегментации, instant segmentation) подразумевает обнаружение целевых объектов и сегментацию каждого из них по отдельности. Среди архитектур экземплярной сегментации наиболее популярной на сегодняшний день является архитектура Mask-R-CNN, которая была предложена в 2017 году Кайменгом Хе [31]. Однако эта архитектура была не первой, реализующей т.н. end-to-end подход к решению этой задачи на основе глубоких нейронных сетей. Более того подход, лежащий в основе архитектуры Mask-R-CNN не единственный путь решения рассматриваемой задачи. И так, ниже приведены некоторые особенности решений задачи экземплярной сегментации в настоящее время [311].

- Экземплярская сегментация может быть решена как задача поиска локальных масок или как задача поиска глобальных масок.
- Глобальный подход предполагает, что сначала будут получены маски для всего изображения, а затем на их основе будут сформированы ограничивающие рамки объектов. Достоинством этого подхода является баланс разрешений для объектов разных размеров. Однако, такие маски могут быть избыточными.
- Локальный подход предполагает, что сначала будут определены границы (ограничивающие рамки) объектов и затем для каждого из них праведна сегментация. Такой подход, как правило, менее избыточен, но может снижать разрешение модели.
- Экземплярская сегментация может быть решена несколькими путями, например следующими способами.

- Явный поиск обрамляющих масок по точкам. Такой подход предполагает поиск достаточного числа ключевых точек объектов, так, что по ним можно будет описать контур объекта. Популярной модификацией, реализующей этот подход является архитектура polar mask [312]. В этой архитектуре предложено искать центральные точки каждого объекта и плотность вероятности нахождения объекта в разных направлениях от каждого центра. Поиск осуществляется в полярных координатах.
- Поиск масок в каждой точке. Такой подход предполагает, что поиск масок объектов каждого класса будут осуществлен для каждой области входного изображения. Пример реализации такого подхода это архитектура TensorMask. Эта архитектура

Архитектура Mask-R-CNN. В 2017 году Кайменгом Хе и соавторами, в том числе Girshick R. было предложено улучшение архитектуры Faster-R-CNN, в том числе дополнительно к операциям классификации и определения координат объекта было предложено добавить небольшую сверточную сеть семантической сегментации для предсказанных регионов кандидатов. Архитектуру было предложено назвать **Mask-R-CNN** [31]. Архитектура Mask-R-CNN позволяет повысить точность решения задачи определения и локализации объектов, а также позволяет решить задачу объектной сегментации (Instance segmentaion). Иллюстрация архитектуры Mask-R-CNN приведена на рисунке 8.122.

Главными отличиями MASK-RCNN от Faster-RCNN стали:

- дополнительная головная часть сегментации объектов;
- замена операции ROI Pooling на более продвинутый вариант ROI Align.

Таким образом объектная сегментация свелась к тому, чтобы не просто оценить наличие объекта в регионе кандидата, но и сегментировать его. Суть операции ROI Align сводится к замене операции макс-пулинга на билинейную интерполяцию понижения размерности. Этот подход более точен, особенно для случая небольших объектов, однако требует больших временных затрат. В случае небольших объектов подход ROI Align может давать до 50% прирост точности. Для объектов больших размеров такой прирост может быть порядка 10 %. Иллюстрация работы операции ROI Align приведена на рисунке 8.123

Архитектура MASK-RCNN лидирует по полярности в приложениях экземплярной (объектной) сегментации по настоящее время [304, 313].

Архитектуры Faster-R-CNN и Mask-R-CNN одни из наиболее популярных для решения задач обнаружения и локализации объектов [291]. А также Mask-R-CNN остается одной из наиболее популярных архитектур для решения задачи объектной сегментации [30]. Однако, данные сети не являются наиболее точными для классических тестовых наборов данных, таких как COCO [314] по данным ресурса paperwithcode.com [304, 313]. В настоящее время данная архитектура, как и Faster-R-CNN часто используются с кодировщиком признаков на основе ResNet и его модификаций. В работах [190, 191] были предложены модификации

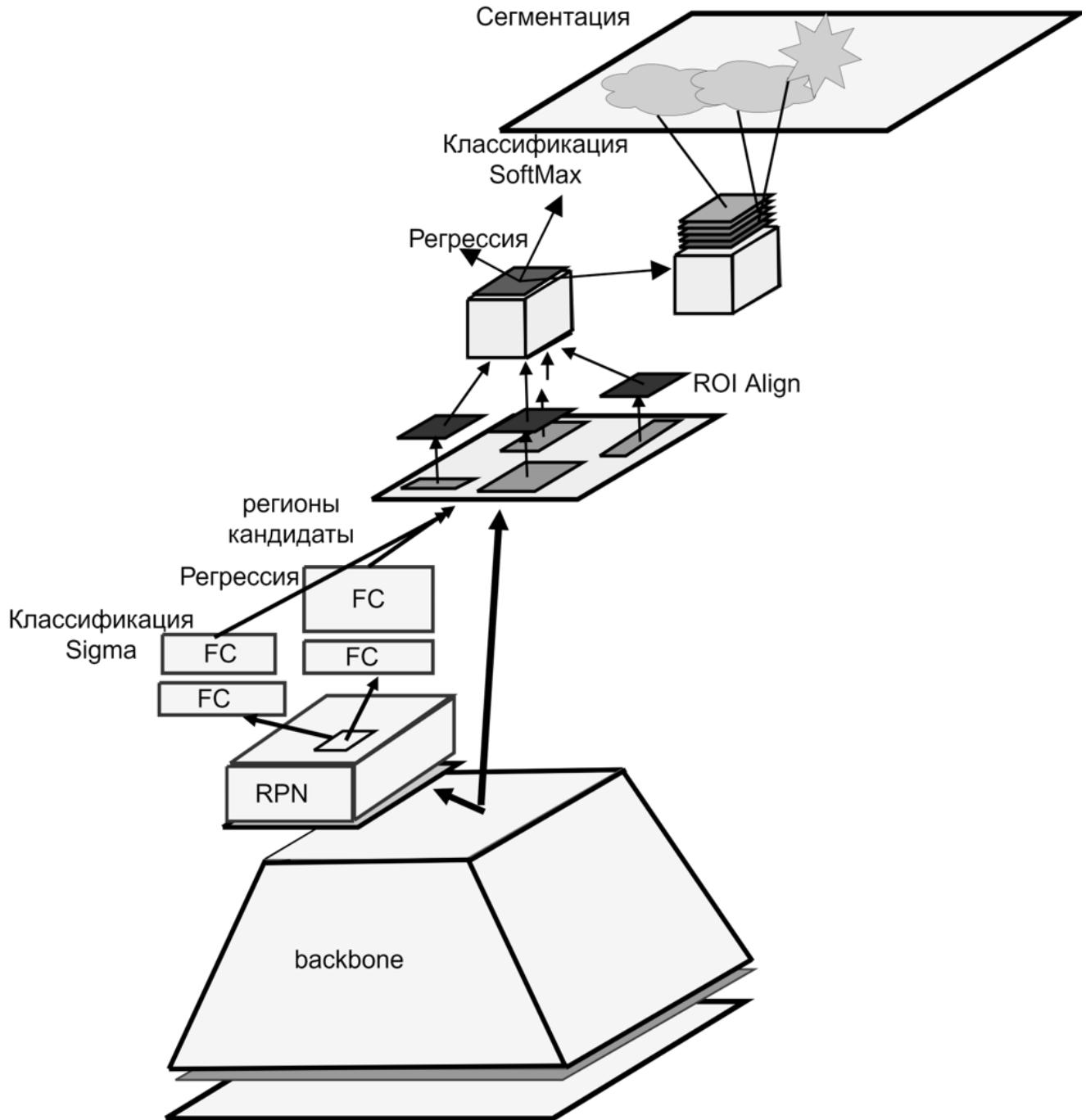


Рисунок 8.122 – Иллюстрация работы архитектуры Mask-R-CNN

данных архитектур с использованием модулей глобального контекста (внимания). В работе [33] было предложено использование пирамиды признаков в кодировщике архитектур.

Каскадные Архитектуры на основе Mask-R-CNN. Архитектура Mask-R-CNN породила большое число исследований, в которых были проведены попытки ее улучшения. Одним из таких подходов являлось использование пирамиды признаков FPN, о которой говорилось в своем разделе. Подход FPN является примером успешной модификации кодировщика признаков и шейной части. Однако, большая часть работ, посвященных модификации этого подхода направлены в сторону улучшения головной части. В том числе рядом авторов предпринимались попытки создания каскада головных частей. Основная идея

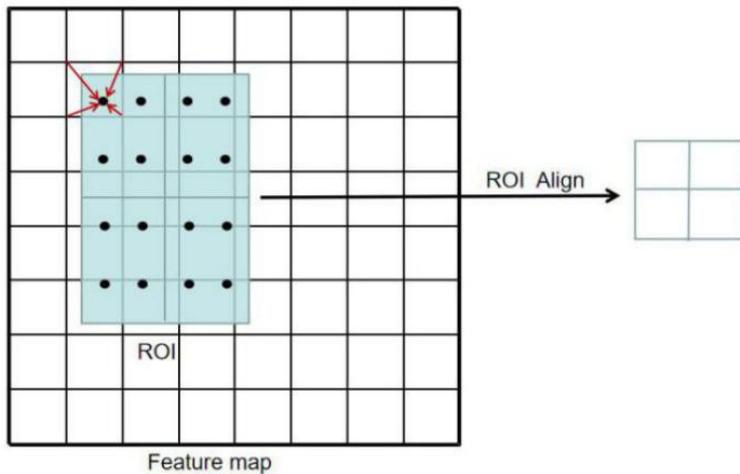


Рисунок 8.123 – Иллюстрация принципа работы подхода ROI Align

тут в том, что каждая головная часть в чем то отличается от предыдущей и дополняет ее в рамках общей цели повышения точности.

Архитектура Cascade Mask-R-CNN [315] предлагает подход с использованием нескольких связанных головных частей. Основная идея подхода Cascade Mask-R-CNN заключается в следующем. В стандартных архитектурах Faster-R-CNN/Mask-R-CNN возникает проблема выбора порога пересечения площадей (Intersection Over Union, IoU threshold). От величины порога зависит то будет ли объект позитивным или негативным (фон). Если порог слишком низкий, то результаты будут содержать слишком много ложных срабатываний. Однако повышение порога может приводить к деградации детектора (пропуску объектов). Этот высокого порога при обучении может привести к эффекту, который авторы назвали "экспоненциальное вымывание позитивных примеров" ("exponentially vanishing positive samples") и переобучению в следствие несогласованности значений IoU для которых модель обучается и для тех значений на которые нативно пресутствуют во входных данных. Также указывается, что проблема классического подхода в несоответствии порога IoU во время обучения и работы архитектуры. Указанная проблема по мнению авторов [315] может быть решения если архитектура будет иметь многоступенчатую головную часть. При этом каждая ступень следующая ступень обучается с повышающим значением порогом IoU. Это делает ступени более избирательными и защищёнными от ошибок типа ложное срабатывание. Головные части обучаются этап за этапом, используют тот факт, что нижняя ступень имеет достаточно корректное распределение срабатывания для тренировки следующей части. Иллюстрация архитектуры Cascade Mask-R-CNN приведена на рисунке 8.124. В отношении архитектуры отметим также, что подход предполагается быть применимым как архитектуре Mask-R-CNN, так и Faster-R-CNN.

На основе архитектуры Cascade-Mask-R-CNN была предложена ее модификация Hybrid task cascade for instance segmentation (HTC) [316]. Отличие подхода HTC от Cascade-R-CNN заключается в последовательном использовании результатов генерации масок экземплярной сегментации на каждом уровне каскада вместо их параллельного использования в Cascade-Mask-R-CNN. Также выделенные макси каждого этапа используются для генерации

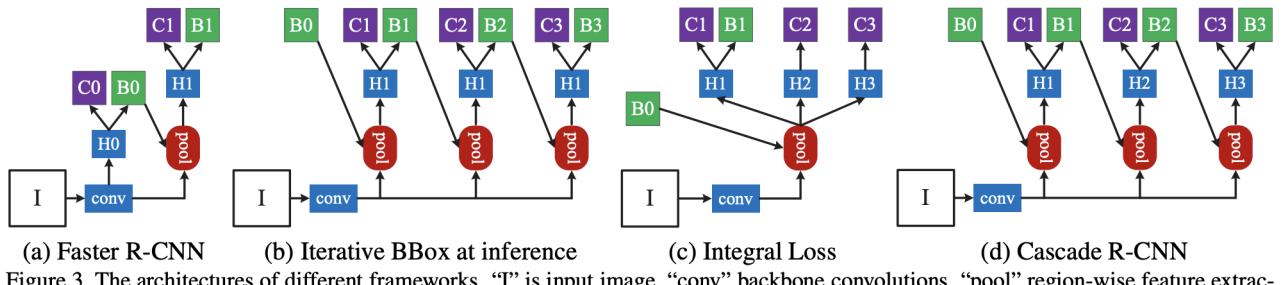


Figure 3. The architectures of different frameworks. “I” is input image, “conv” backbone convolutions, “pool” region-wise feature extraction, “H” network head, “B” bounding box, and “C” classification. “B0” is proposals in all architectures.

Рисунок 8.124 – Иллюстрация принципа работы подхода Cascade Faster-R-CNN/Mask-R-CNN

предложений следующего этапа. Кроме того в данной архитектуре используется дополнительная ветвь семантической сегментации (s), которая сливается с результатами выделения ограничивающих рамок и масок на каждом этапе головного каскада. Таким образом по мнению авторов архитектура максимально эффективно использует т.н. "поток информации" ("information flow") не только для разных ступеней каскада, но и для разных задач. Идея архитектуры НТС представлена на рисунке 8.125.

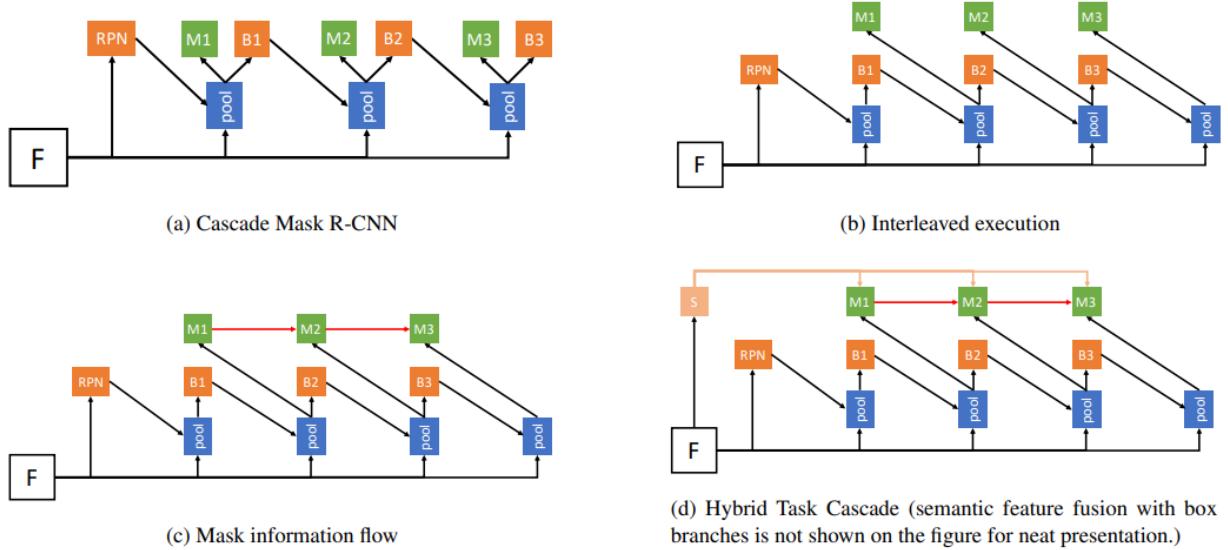


Рисунок 8.125 – Иллюстрация архитектуры НТС и ее прообразов

Архитектура НТС лежит в основе большинства наиболее точных современных методов решения задач: обнаружения объектов и их сегментации для стандартных фреймворков [304, 313].

Архитектура MaskLab. Другим примером модификации архитектуры Mask-R-CNN является подход MaskLab [?]. В этом подходе для каждого региона потенциального интереса (ROI) выполняется т.н. foreground/background segmentation. В данном случае под этим термином подразумевается комбинация проблем семантической сегментации и предсказания направлений. Проблема предсказания направления подразумевает оценку направления каждого пикселя к центру соответствующего объекта. По задумке авторов решение задачи семантической сегментации позволяет модели выполнить разделение между объектами различных классов, включая фон. Решение задачи предсказания направления к центру объектов позволяет разделять объекты одного класса. Иллюстрация архитектуры приведена на рисунке 8.126.

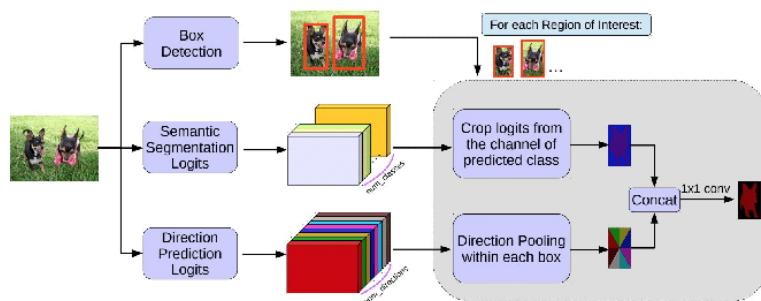


Рисунок 8.126 – Иллюстрация архитектуры MaskLab

В архитектуре MaskLab также была предложена модификация головной части, которая была названа ее уточнением. Этот принцип заключается в том, что т.н. грубые маски (coarse mask logits by only exploiting semantic and direction features) объединяются с соответствующими позициями карт признаков нижних слоев кодировщика признаков. Затем полученный набор каналов обрабатывается несколькими сверточными слоями для получения итоговых результатов [?].

Также в работе авторы предложили т.н. деформированный пулинг регионов (Deformable RoI pooling) [?]. По существу эта операция аналогична деформированному пулингу, предложенному в работе [254] (деформированная свертка). Однако в данной модификации есть небольшие изменения. Авторы предлагают сначала вырезать регионы интереса и затем изменять их размеры при помощи билинейной интерполяции. Затем регионы разделяются на несколько частей (sub-boxes). Для каждой части предсказывается ее смещение на изображении. Предсказание осуществляется при помощи дополнительной сети. Таким образом полученный регион должен по задумке авторов лучше выделять регионы одного класса и разных объектов. Иллюстрация этого принципа показана на рисунке 8.127.

Архитектура PANet. В работе [317] была предложена идея объединение различных карт признаков пирамидального кодировщика признаков для их последующей обработки в головной части. Идея получила название path aggregation network (PANet). Авторы архитектуры PANet предложили усилить базовую пирамидальную архитектуру FPN. Подход FPN предполагал объединение признаков "сверху вниз" - то есть от более высокоуровневых признаков к меньшим уровням. В архитектуре PANet был добавлен путь наоборот "снизу

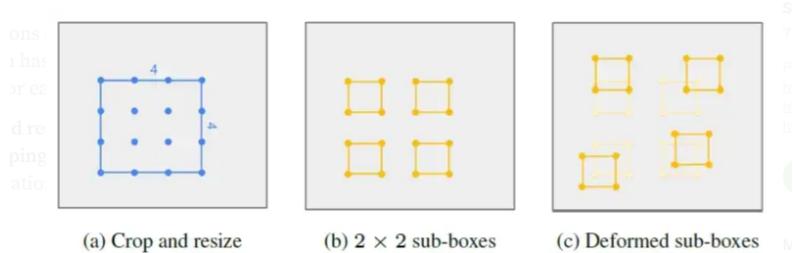


Рисунок 8.127 – Иллюстрация пулинга в MaskLab

в верх". Основная идея тут в том, чтобы повысить точность локализации признаков за счет учета низкоуровневых признаков на высоких уровнях. Иллюстрация архитектуры PANNet праведна на рисунке 8.128.

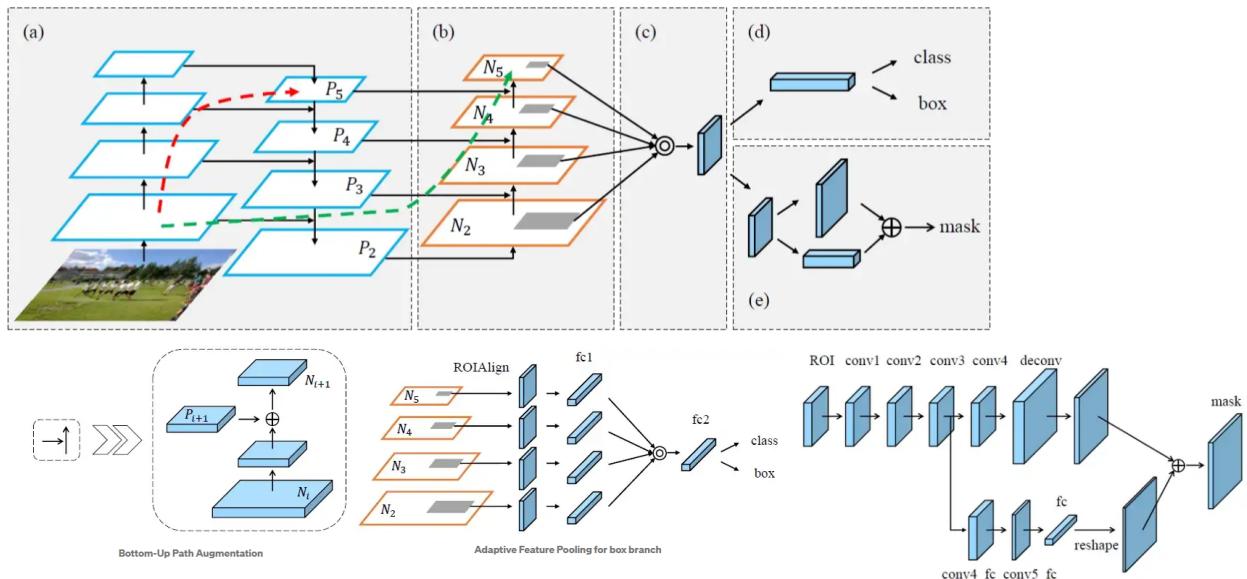


Рисунок 8.128 – Иллюстрация архитектуры PANet

Сочетание путей шейной части "сверху вниз" и "снизу вверх" по мнению авторов PANNet создает "короткий путь" между признаками верхней и нижней части. Таким образом повышается точность учета локализации признаков и их деталей (нижний уровень) и их абстрактных особенностей и контекст (верхний уровень). Напомним, что абстрактные особенности определяются величиной receptive поля, что, одна приводит к потере локальной информации о признаках. Также в архитектуре PANet было предложено использовать в головной части сбор (объединение) признаков разных уровней вместо их независимого использования. Такую операцию авторы назвали аддитивный пулинг признаков (Adaptive feature pooling) [317]. При этом предполагается, что одновременно будут учтены

Для формирования масок объектов авторами работы [317] было предложено использовать две головные части. Одна для непосредственно семантической сегментации предложений, а другая полносвязный слой. Финальные макси формируются путем объединения этих "путей зрения". По мнению авторов такой прием должен увеличить точность результатов работы модели. Полносвязный слой лучше позволяет учесть локально не связанный контекст, однако приводит к потере локальной информации. Для выделения признаков авторы используют операцию ROI Align аналогично подходу Mask-R-CNN.

8.6 одноступенчатые архитектуры экземплярной сегментации

Архитектура InstanceFCN Архитектура состоит из двух ветвей головной части. Первая ветвь генерирует k^2 карт признаков, которые соответствуют каналам выходной карты признаков. stance-sensitive score maps branch For the first branch (top path), we adopt a 1×1 512-d convolutional layer to transform the features, and then use a 3×3 convolutional layer to generate a set of k^2 instance-sensitive score maps, which is k^2 output channels. ($k=5$ finally.) An assembling module is used to generate object instances in a sliding window of a resolution $m \times m$. ($m=21$ here.) The idea is very similar to that of positive-sensitive score maps in R-FCN. But R-FCN uses positive-sensitive score maps for object detection while InstanceFCN uses instance-sensitive score maps for generating proposals. Objectness score map branch For the second branch of scoring instances (bottom path), we use a 3×3 512-d convolutional layer followed by a 1×1 convolutional layer. This 1×1 layer is a per-pixel logistic regression for classifying instance/not-instance of the sliding window centered at this pixel. Thus, it is a objectness score map. Loss function

8.7 Быстрые архитектуры экземплярной сегментации

Как уже отмечалось экземплярная сегментация, как правило, представлена достаточно громоздкими решениями. Напомним, что эта задача подразумевает одновременно как решение задачи обнаружение объектов, так и сегментации каждого из них. Ряд архитектур обнаружения объектов могут быть сравнительно быстро модифицированы для решения этой задачи. Так, одним из примеров быстрых подходов, позволяющих перейти к экземплярной сегментации является архитектура YOLO v5 и [309].

В литературе предложен ряд специфических архитектур быстрой экземплярной сегментации. В том числе архитектура YOLACT [318] и их модификации, такие как YOLACT++ [319], SOLOv2 [320] и SparseInst [321], которые обеспечивают самые современные результаты в сегментации экземпляров в реальном времени. тесты [322]. Основная идея этих методов заключается в использовании сегментации на основе центра в отличие от сегментации на основе региона в Mask-R-CNN. Здесь модель обучается различать пиксели, принадлежащие одному и тому же или разным объектам.

- Архитектура YOLACT [?] генерирует маски-прототипы по всему изображению, предсказывая набор коэффициентов для каждого экземпляра по двум параллельным головам. Окончательные маски строятся после немаксимального подавления прогнозируемых экземпляров. Иллюстрация работы архитектуры YOLACT приведена на рисунке
- Архитектура YOLACT++ [319] улучшает предыдущие результаты [318] за счет улучшения экстрактора признаков и добавления деформируемых сверток в головную часть.
- Архитектура SOLO [323] предполагала, что экземпляры можно разделить по центральному положению и размеру. Центральные позиции вычисляются путем разделения изображения на ячейки и поиска центральной позиции внутри каждой из них. Размеры экземпляров определяются экстрактором пирамидальных признаков.
- Архитектура SOLO v2 [320] улучшает предыдущие результаты, генерируя ядра для каждой маски. Эта концепция называется динамическая свертка. Иллюстрация архитектуры SOLO v2 приведена на рисунке 8.130.
- Архитектура SparseInst [321] создает две выходные ветви, аналогичные SOLOv2. Первая выходная ветвь предназначена для генерации маски с учетом классов, а вторая — для применения к классификации, оценкам объектов и ядрам масок. Ядра маски умножаются на предсказанные маски для создания масок сегментации. Иллюстрация архитектуры sparseinst приведена на рисунке .

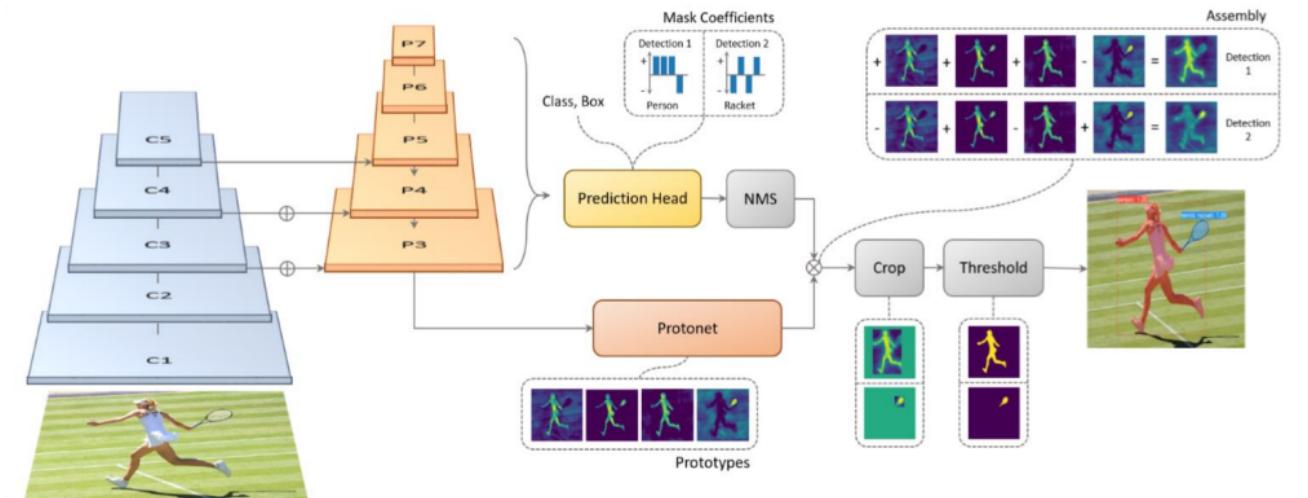


Рисунок 8.129 – Иллюстрация архитектуры yolact

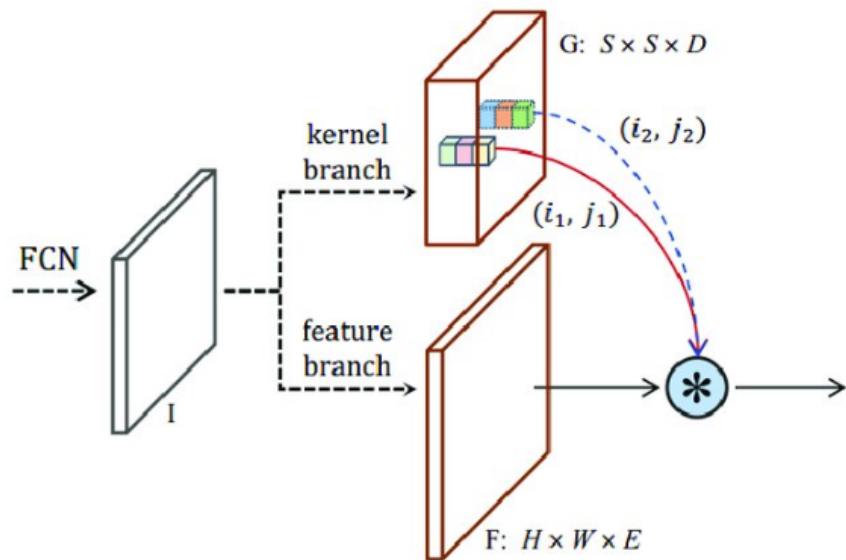


Рисунок 8.130 – Иллюстрация архитектуры solo v2

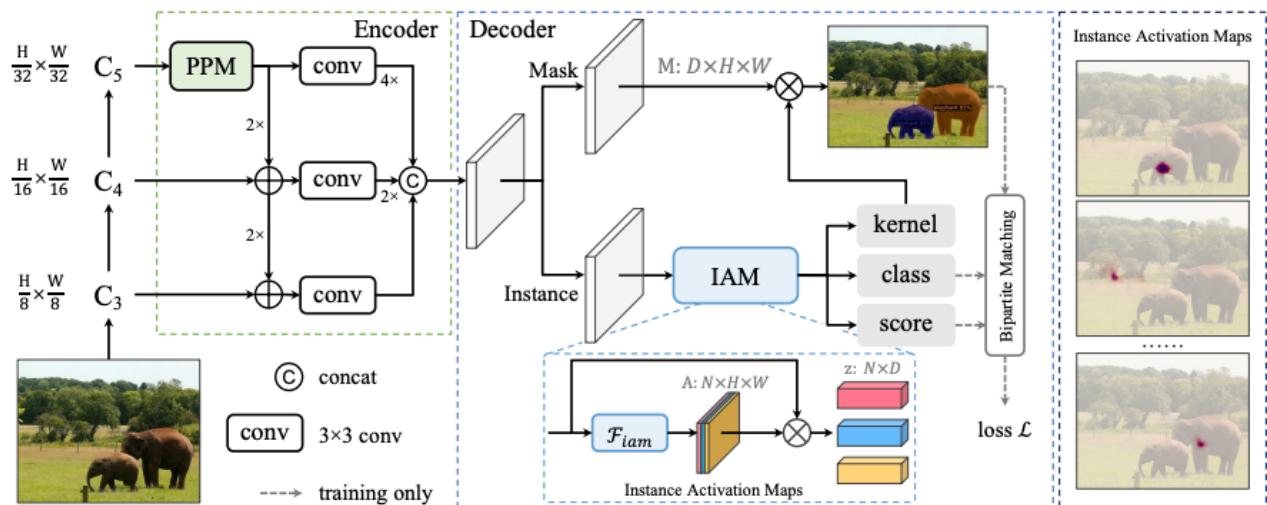


Рисунок 8.131 – Иллюстрация архитектуры sparseinst

9 Оптимизация нейронных сетей в параллельных устройствах

9.1 Центральные процессоры.

Современные процессоры можно рассматривать как передовые мультипроцессорные (многоядерные и многопоточные) архитектуры, оптимизированные для векторных и матричных операций. Даже некоторые настольные процессоры могут иметь до 64 ядер и 128 потоков (с точки зрения аппаратной многопоточности) [324] и до 72 ядер и 288 потоков в серверных процессорах [325]. В случаях ограниченного бюджета или некоторых требований к работе устройства возможность использования универсальных центральных процессоров (ЦП) в обучении и исполнении (вывод) нейронных сетей. Преимущества такого подхода в следующем [326].

- Нет необходимости в дополнительном оборудовании.
- Простая работа с любыми библиотеками, программным обеспечением в том числе так называемыми фреймворками (обычно современные фреймворки поддерживают архитектуры x86-64).
- Наличие дополнительных инструментов оптимизации производительности, таких как Open vino [327], NNPack [328], oneDNN [329], CcT [330] и др. [326], а также другие инструменты, например, Intel DLboost и BigDL [331] для оптимизации нейронных сетей для процессоров Xeon [332].
- Оптимизированные наборы инструкций SIMD, такие как AMX, AVX-512VNNI и другие, поддерживающие форматы BF16 и INT8.
- Высокая гибкость не только для машинного обучения, но и для вспомогательных задач.

Также одним из направлений развития процессоров являются гетерогенные архитектуры, включающие:

- центральные процессоры (CPU),
- графические ускорители (GPU),
- параллельные программируемые устройства типа FPGA
- другие типы ускорителей, использующие унифицированную кэш-память
- комбинации, такие как SoC.

Примером такого подхода является нейронное ускорение Intel на базе процессора Miryard. Ожидается, что этот подход позволит достичь максимальной эффективности в рамках ограниченного бюджета и энергопотребления. Среди прочих производители предлагают инструменты для оптимизации вывода нейронных сетей с использованием ограниченной точности, например, для форматов INT8 и BF16 [333].

9.2 Использование графических ускорителей GPU.

Графические процессоры (GPU) — тип процессорных устройств, построенных по принципу матричных архитектур с синхронным параллелизмом на уровне команд (так называемые Matrix Single Instruction Multidata, SIMD-архитектуры). Этот тип архитектуры ориентирован на обработку больших регулярных массивов данных (включая изображения). Изначально задачи, для которых разрабатывались такие процессоры, представляли собой специальные операции обработки изображений, которые выполнялись на аппаратном уровне. Большинство из этих операций включают умножение матриц и реализуются с помощью операций сложения и умножения с плавающей запятой (Fused Multiplication and Add, FMA). Поэтому ряд современных графических процессоров специализируются на этом типе операций. Такие ускорители называются графическими процессорами общего назначения (GPGPU) [334]. В настоящее время GPGPU представляют собой многопроцессорные системы, состоящие из набора потоковых мультипроцессоров (SIMT). Мультипроцессоры также можно объединять в группы, кластеры (или слайсы). Каждый мультипроцессор SIMT выполняет блок (или поток) инструкций SIMD, используя набор функциональных модулей.

Графические ускорители получили широкую популярность в задачах обучения и работы алгоритмов машинного обучения. В первую очередь это связано с тем, что алгоритмы машинного обучения (включая глубокие нейронные сети) хорошо поддаются распараллеливанию. Данные (параметры сети и входные данные) обычно включают в себя правильные векторы и матрицы чисел. Более того, операции могут выполняться последовательно, *т.е.* алгоритмы обучения и работы нейронных сетей включают набор последовательных инструкций, производимых одна за другой над одним набором данных. Большинство операций нейронной сети сводится к FMA. В таких задачах преимущества графических ускорителей заключаются в их широких возможностях параллелизма и высокой вычислительной мощности (десятки TFLOPS в формате FP32 и сотни TFLOPS в формате FP16 при использовании тензорных ядер), а также передовых технологиях построения GPU-серверов. Для процедуры обучения нейронных сетей графические ускорители обычно обеспечивают наилучшее соотношение цены и производительность. Кроме того, графические процессоры часто являются единственным аппаратным обеспечением, способным справиться и масштабироваться с вычислительными требованиями определенных алгоритмов и моделей глубокого обучения [335].

В настоящее время наиболее популярными в задачах машинного обучения среди производителей графических ускорителей являются графические ускорители NVIDIA [336]. Такая популярность обусловлена как реализацией технологий оптимизации самой NVIDIA, так и поддержкой CUDA для большинства фреймворков для обучения нейронных сетей. Для логического вывода производительность может быть увеличена до 10 раз, что связано с технологиями использования разреженных матриц и форматов параметров INT8/INT4. Реальная производительность графических ускорителей, как правило, ниже пиковой и зависит от оптимизации кода. Такая оптимизация в ручном режиме часто игнорируется, так как она очень трудоемка. Однако можно использовать оптимизацию с помощью таких утилит, как NVIDIA TensorRT [337].

Современные GPGPU позволяют выполнять операции FMA с несколькими вариантами разрядности [334]. В частности, поддерживаются операции смешанной точности, например, операции FMA вида $D = A \cdot B + C$, где операнды A и B могут иметь половинную точность [?]. Использование операндов пониженной точности уменьшает объем видеопамяти, занимаемой параметрами нейронной сети.

Для ускорения обработки вычислений в глубоких нейронных сетях в ряде архитектур графических ускорителей используются так называемые тензорные вычислительные модули. В зависимости от архитектуры GPU эти модули могут отличаться как по количеству, так и по поддерживаемому функционалу. Например, в процессорах архитектуры NVIDIA Ampere [338] каждый блок использует тензорное ядро, которое поддерживает операции смешанной точности FP16/FP32, FP16, BF16, TF32, FP64, INT8, INT4 и бинарные операции, а также операции над разреженными матрицами в соотношении 2 нуля на 4 значения. При этом каждый тензорный модуль представляет собой массив, например, $4 \times 4 \times 4$, предназначенный для выполнения в цикле операций FMA $D = A \cdot B + C$, где A , B , C и D матрицы 4×4 [339].

Современные фреймворки для нейронных сетей предоставляют инструменты для автоматического и адаптивного применения методов смешанной точности [340, 341, 342]. Они легко квантуют модель и уравновешивают ее производительность, эффективность и точность. На практике обученная нейронная сеть имеет большинство весовых коэффициентов, равных или близких к нулю. Такие веса (и соответствующие соединения) могут быть удалены или сокращены [343, 344].

В зависимости от номера слоя в модели может быть достигнута разреженность до 90% без значительных потерь в точности. В экспериментах [345] авторы получили 1% потери точности при коэффициенте разреженности 65% для архитектуры GoogLeNet. В TNN может быть автоматически удалено до 50% весов. Некоторые современные графические ускорители используют технологии оптимизации вычислений операций над разреженными матрицами [346]. Этот метод уменьшает время вывода нейронных сетей.

Для современных сверточных нейронных сетей весьма характерна достаточная степень разреженности. Это связано с частым использованием полулинейных функций ReLU. Также по назначению может быть достигнута разреженность параметров нейронных сетей. Такие матрицы можно встретить в некоторых задачах обучения нейронных сетей (включая некоторые веса, которые можно инициализировать нулями). Этот подход, например, на архитектуре NVIDIA Ampere [347], использует предположение, что операнды инструкций для разреженных матриц являются матрицами, в которых есть два ненулевых значения в каждом векторе с четырьмя входными значениями (разреженность строк 2:4). . Благодаря такой матричной структуре его можно сжимать, уменьшая требуемый объем памяти и пропускную способность почти вдвое (разреженное тензорное ядро). Отмечается, что использование разреженных вычислений может потребовать дополнительного обучения нейронной сети; в результате такого подхода время вывода можно сократить вдвое.

9.3 Работа с памятью

Среди факторов, ограничивающих производительность нейронных сетей, можно назвать скорость памяти и энергопотребление. Обычно для оценки производительности системы с точки зрения вычислительной производительности и пропускной способности памяти используется модель крыши [348]. Эта модель представляет собой предельные технические характеристики системы и может использоваться для оценки производительности реальной программы. Для достижения более высокой производительности следует одновременно применять несколько способов оптимизации. Например, повышение производительности может быть получено за счет оптимизации архитектуры вычислительных модулей, использования адаптированных форматов данных и оптимизации вычислительных алгоритмов. Кроме того, за счет оптимизации циклов может быть достигнуто значительное улучшение производительности. Можно использовать несколько методов, таких как переупорядочивание цикла, конвейерная обработка цикла, развертывание цикла и другие. Переупорядочивание цикла уменьшает количество обращений к памяти между итерациями цикла. Развертка цикла включает выполнение нескольких итераций одновременно параллельно. Конвейерная обработка цикла выполняет итерацию цикла с перекрытием таким образом, что следующая итерация начинается до окончания предыдущей. Оптимальное количество развернутых итераций может варьироваться в зависимости от программы. Этот параметр также влияет на количество обращений к памяти и должен зависеть от аппаратной архитектуры. Оптимизацию цикла можно выполнить, используя двойные буферы для хранения результатов [349]. В некоторых работах [350] предлагается развертывание ядра для CNN. Он представляет собой замену одного ядра свертки, например, 5×5 , несколькими ядрами размера 3×3 . Оптимизация циклов также может быть достигнута за счет использования регулярной структуры нейронных сетей [351].

В дополнение к стратегиям, описанным выше, для оптимизации циклов могут использоваться методы более высокого уровня. Эти методы включают мозаику петель и замену петель. При мозаичном расположении циклы разбиваются на более мелкие компоненты, плитки. Все входные данные для одного тайла хранятся в выделенном буфере или кэш-памяти. Цикл выполняется с каждой плиткой один за другим. Обмен циклами заключается в перестановке компонентов цикла таким образом, чтобы соседние компоненты использовали одни и те же данные. Таким образом, нам не нужно перезагружать данные из памяти для отдельных компонентов. Эти методы позволяют гибко контролировать изменение размеров и порядка плиток. Таким образом, мы можем оптимизировать циклы для различных слоев нейронной сети, например, CNN [351].

Принимая во внимание оптимизацию матричной свертки [352, 353], мы можем использовать несколько типов алгоритмов: пространственная свертка (CONV), векторизованная свертка (например, im2col), быстрая свертка Винограда и частотная свертка. Алгоритм Винограда является самым быстрым. Однако это требует отдельной реализации для различных размеров сверточных ядер, что не всегда возможно. Частотная свертка имеет умеренную скорость, но может быть реализована только для размеров ядра степени 2. Пространственная свертка является самой медленной, однако она предъявляет

самые низкие требования к кешу и пропускной способности памяти.

9.4 Особенности ускорителей типа FPGA

Во многих работах об экспериментальных и коммерческих проектах предлагается разработка специализированных аппаратных ускорителей для обучения и вывода нейронных сетей [354, 355, 350, 356]. Одной из основных причин разработки специализированных архитектур является стремление максимизировать параллелизм, присущий моделям машинного обучения и нейронным сетям, особенно для глубоких нейронных сетей. Как правило, процессоры общего назначения, а также графические процессоры и процессоры DSP не полностью обеспечивают оптимальную производительность. В то же время высокая универсальность таких процессоров приводит к высокому энергопотреблению и высокой цене. Даже если у них есть специальные подмодули для обработки нейронных сетей, они дополняют основную архитектуру [357].

В настоящее время предлагается достаточно большое количество различных специализированных архитектур ускорителей, как в литературе, так и в виде коммерческой реализации. Такие ускорители часто предназначены для использования в обучающих серверах или во встроенных устройствах логического вывода с низким энергопотреблением. В зависимости от задачи ускорители могут использовать различные архитектуры. Часто специализированные ускорители основаны на технологиях FPGA или ASIC. Преимуществом ПЛИС является возможность организации достаточно гибкой системы параллельных вычислений с заданной точностью. Современные ПЛИС могут включать в себя несколько вариантов расчета, таких как таблицы соответствий (LUT), цифровая обработка сигналов (DSP) и двоичная логика (триггер). Расчет может быть выполнен для любой из этих составляющих. Часто ПЛИС реализуются как гетерогенные системы с контроллером или процессором (технология System on Chip). Это позволяет разработчикам обеспечить максимально гибкий подход к аппаратной и программной реконфигурации исполнительного блока.

Программируемая пользователем вентильная матрица (FPGA) — популярная технология для реализации аппаратных ускорителей для обработки нейронных сетей. Они могут быть изготовлены на основе технологии специализированных интегральных схем (ASIC) [358]. Следует отметить, что обычно специализированные архитектуры FPGA основаны на принципах SIMD CPU или SIMT GPGPU. Сравнение производительности логического вывода сверточных нейронных сетей для различных CPU, GPU и FPGA представлено в работах [359]. Также отметим, что, помимо анализа аппаратных ускорителей, в некоторых работах рассматриваются программно-аппаратные архитектуры с возможностью реконфигурации [360]. К преимуществам специализированных ускорителей нейронных сетей, помимо энергоэффективности, цены и размера, относятся: оптимизированный набор инструкций (ISA); более высокая степень распараллеливания, повторного использования данных и буферизации (с использованием встроенных буферов FIFO); возможность обработки специализированных форматов данных, таких как нестандартная разрядность, а также разреженные вычисления. В некоторых реализациях вместо ISA вычисления

нейронной сети выполняются с использованием секвенсора конечного автомата [361].

Многие источники [345] отмечают, что ускоритель FPGA более предпочтителен для CNN, чем GPU и CPU из-за его возможности построения произвольных конфигураций вычислительных модулей. Эти модули обычно формируются в виде процессорного элемента (PE). Блок PE является элементом скалярных, SIMD или матричных вычислений. В литературе рассматриваются конфигурации PE в виде архитектуры SIMD (обычно для сопроцессоров CPU) или SIMT (для GPGPU). Эти конфигурации представлены в классе Темпоральных Архитектур [333]. Специализированные конфигурации PE для ускорителей DNN включают настраиваемые архитектуры VLIW и Decoupled Access/Execute, а также так называемые систолические массивы [362].

Архитектуры систолического массива реализуют принцип обработки потока данных и относятся к классу пространственных ускорителей [333]. Систолические массивы могут быть статическими, а также реконфигуруемыми. В ускорителях DNN наиболее распространены статические массивы, построенные по технологии ASIC [363]. В настоящее время наиболее распространенным типом систолического массива является блок тензорного процессора (TPU) [358].

Систолические массивы имеют преимущества высокой степени повторного использования данных, низких требований к памяти и низкого энергопотребления. Целью использования этих типов архитектуры является уменьшение эффекта задержки памяти для внешней памяти за счет оптимизации структуры процессора для архитектуры нейронных сетей [364]. Массивы систолических тензоров также могут быть оптимизированы для умножения разреженных матриц [363]. Многочисленные источники (подробнее см. в обзоре [333]) предполагают, что помимо TPU могут быть перспективны и несколько других типов пространственных архитектур ускорителей нейронных сетей. Эти системы предназначены для ускорения операций умножения матриц общего назначения (GEMM) и операций свертки с использованием множества регулярных исполнительных элементов.

10 Заключение

Проведен анализ литературы, по результатам которого составлен литературный обзор. Показано, что в настоящее время использование систем компьютерного зрения является перспективным направлением решения задач оценки производительности открытых горнодобывающих работ, в том числе в рамках разработки автоматических методов визуальной оценки содержания асбестового волокна в горной породе на асbestовых карьерах. Среди методов компьютерного зрения в настоящее время наибольший интерес представляет исследование методов на основе глубокого обучения нейронных сетей.

Основными трендами обозначенной области являются использование продвинутых архитектур сверточных нейронных сетей; поиск баланса между использованием сверточных слоев и таких современных идей, как слои трансформеры или слои миксеры. Последние подразумевают отказ от интуитивных предположений, лежащих в основе использования операции свертка для анализа изображений. Однако, такие архитектуры, как правило не ориентированы на их использование в рамках низкопроизводительных конечных устройств. В обзоре дается анализ архитектур нейронных сетей, ориентированных на использование в низкопроизводительных устройствах.

Также в обзоре показаны некоторые методы оптимизации архитектур для конечных устройств. Такие методы позволяют значительно снизить вычислительную сложность и требования к памяти. Например, это возможно за счет перехода к низко разрядным вычислениям или использования аппарата разряженных матриц. Отдельно отмечается возможность достижения высоких величин оптимизации при использовании т.н. «устройств ускорителей нейронных сетей».

Среди других трендов в рассматриваемой области показаны особенности современных подходов к регуляризации обучения. К основным направлениям тут относятся поиск методов замены операции «батч-нормализация», использование стратегий мета-обучения, в том числе полу-контролируемого обучения и исследование новых методов расширения наборов данных. Также к трендам в данной области следует причислить поиск новых эвристических правил обучения тех или иных архитектур глубоких нейронных сетей. В обзоре показано, что на сегодняшний день использования компьютерного зрения в геологии, в том числе в задачах оценки производительности открытых горнодобывающих работ, в значительной степени отстает от общего развития. При этом задача выделения прожилок полезной породы на кусках камней не решается. Наиболее близкой задачей к поставленной является оценка так называемой фрагментации – то есть оценка размеров фрагментов скальной породы на снимке. В подавляющем большинстве случаев эта задача решается как проблема экземплярной сегментации (instant segmentation). В этой задаче часто используются устаревшие подходы, такие как метод водораздела. Метод используется или напрямую или для результатов решения задачи семантической сегментации. Указанная проблема семантической сегментации может быть решена или в рамках каких-либо подходов классического компьютерного зрения или при помощи глубоких нейронных сетей. Показано, что среди таковых используются архитектуры семейства U-Net (2015 года). Также в небольшом количестве присутствуют работы, где задача экземплярной сегментации решена

на прямую при помощи архитектуры Mask-R-CNN (2017 год). Указанные архитектуры являются классическими в задачах глубокого обучения нейронных сетей. При этом более современные архитектуры нейронных сетей как правило игнорируются.

Показана возможность использования двух типов архитектур семантической сегментации: архитектур типа энкодер-декодер и архитектур с расширенной сверткой. Оба типа архитектур могут комбинироваться при необходимости. Также в обзоре проанализированы архитектуры нейронных сетей экземплярной сегментации и лежащие в их основе архитектуры для решения задач обнаружения объектов. Показано, что в этой области могут быть как многоэтапные, так и одноэтапные подходы. Последние являются значительно более быстрыми, хотя и обладают несколько меньшей чувствительностью к низкоразмерным объектам. Такие архитектуры могут быть успешно использованы на низкопроизводительных устройствах, в том числе в реальном масштабе времени. В обоих подходах могут быть применены как полностью сверточные блоки, так и использованы слои трансформеры.

Таким образом проведенный литературный обзор позволяет сделать вывод о перспективных направлениях исследований возможностей использования современных архитектур глубоких нейронных сетей для решения задач семантической и экземплярной сегментации, а также обнаружения объектов, в том числе с использованием комбинаций сверточных блоков и блоков трансформеров/миксеров с целью визуальной оценки содержания асBESTового волокна в горной породе на асBESTовых карьерах Свердловской области (на примере карьеров ПАО «УралАсBEST»). В рамках исследований следует делать упор на поиск оптимальных архитектур глубоких нейронных сетей для низкопроизводительных устройств, а также методов их обучения, в том числе с использованием метаобучения.

СПИСОК ЛИТЕРАТУРЫ

- [1] Fu Y, Aldrich C. Deep learning in mining and mineral processing operations: a review // IFAC-PapersOnLine. 2020. Т. 53, № 2. С. 11920–11925.
- [2] Zhou Wenyan, Wang Hao, Wan Zhibo. Ore Image Classification Based on Improved CNN // Computers and Electrical Engineering. 2022. Т. 99. с. 107819.
- [3] Research on intelligent identification of rock types based on faster R-CNN method / Xiaobo Liu, Huaiyuan Wang, Hongdi Jing [и др.] // Ieee Access. 2020. Т. 8. С. 21804–21812.
- [4] Mineral Rock Classification Using Convolutional Neural Network / Shanmuk Srinivas Amiripallia, Grandhi Nageshwara Rao, Jahnavi Beharaa [и др.] // Recent Trends in Intensive Computing; IOS Press: Amsterdam, The Netherlands. 2021.
- [5] Karimpouli Sadegh, Tahmasebi Pejman. Segmentation of digital rock images using deep convolutional autoencoder networks // Computers & geosciences. 2019. Т. 126. С. 142–150.
- [6] Deep convolutional neural network for fast determination of the rock strength parameters using drilling data / Mingming He, Zhiqiang Zhang, Jie Ren [и др.] // International Journal of Rock Mechanics and Mining Sciences. 2019. Т. 123. с. 104084.
- [7] Automated lithology classification from drill core images using convolutional neural networks / Fatimah Alzubaidi, Peyman Mostaghimi, Paweł Swietojanski [и др.] // Journal of Petroleum Science and Engineering. 2021. Т. 197. с. 107933.
- [8] Object-Oriented Open-Pit Mine Mapping Using Gaofen-2 Satellite Image and Convolutional Neural Network, for the Yuzhou City, China / Tao Chen, Naixun Hu, Ruiqing Niu [и др.] // Remote Sensing. 2020. Т. 12, № 23. с. 3895.
- [9] Baek Jieun, Choi Yosoon. Deep neural network for predicting ore production by truck-haulage systems in open-pit mines // Applied Sciences. 2020. Т. 10, № 5. с. 1657.
- [10] Exploring deep learning for dig-limit optimization in open-pit mines / Jacob Williams, Jagjit Singh, Mustafa Kumral [и др.] // Natural Resources Research. 2021. Т. 30, № 3. С. 2085–2101.
- [11] A computer vision system for terrain recognition and object detection tasks in mining and construction environments / Godfred Somua-Gyimah, Samuel Frimpong, Wedam Nyaaba [и др.] // SME Annual Conference. 2019.
- [12] Lookup: Vision-only real-time precise underground localisation for autonomous mining vehicles / Fan Zeng, Adam Jacobson, David Smith [и др.] // 2019 International conference on robotics and automation (ICRA) / IEEE. 2019. С. 1444–1450.
- [13] Maitre Julien, Bouchard Kévin, Bédard L Paul. Mineral grains recognition using computer vision and machine learning // Computers & Geosciences. 2019. Т. 130. С. 84–93.

- [14] Measuring blast fragmentation at Nui Phao open-pit mine, Vietnam using the Mask R-CNN deep learning model / Trong Vu, Tran Bao, Quoc Viet Hoang [и др.] // Mining Technology. 2021. Т. 130, № 4. С. 232–243.
- [15] Computer vision system for the automatic asbestos content control in stones / Vasily Zyuzin, Mikhail Ronkin, Sergey Porshnev [и др.] // Journal of Physics: Conference Series / IOP Publishing. Т. 1727. 2021. с. 012014.
- [16] Automatic Asbestos Control Using Deep Learning Based Computer Vision System / Vasily Zyuzin, Mikhail Ronkin, Sergey Porshnev [и др.] // Applied Sciences. 2021. Т. 11, № 22. с. 10532.
- [17] Investigation of Object Detection Based Method for Open-Pit Blast Quality Estimation / Mikhail Ronkin, Alexey Kalmykov, Kirill Reshetnikov [и др.] // 2022 Ural-Siberian Conference on Biomedical Engineering, Radioelectronics and Information Technology (US-BEREIT) / IEEE. 2022. С. 248–251.
- [18] Sangaiah Arun Kumar. Deep learning and parallel computing environment for bioengineering systems. Academic Press, 2019.
- [19] Automatic coal and gangue segmentation using u-net based fully convolutional networks / Rong Gao, Zhaoyun Sun, Wei Li [и др.] // Energies. 2020. Т. 13, № 4. с. 829.
- [20] Ore image segmentation method using U-Net and Res_Unet convolutional networks / Xiaobo Liu, Yuwei Zhang, Hongdi Jing [и др.] // RSC advances. 2020. Т. 10, № 16. С. 9396–9406.
- [21] A deep convolutional neural network model for intelligent discrimination between coal and rocks in coal mining face / Lei Si, Xiangxiang Xiong, Zhongbin Wang [и др.] // mathematical Problems in Engineering. 2020. Т. 2020.
- [22] Rock classification in petrographic thin section images based on concatenated convolutional neural networks / Cheng Su, Sheng-jia Xu, Kong-yang Zhu [и др.] // Earth Science Informatics. 2020. Т. 13, № 4. С. 1477–1484.
- [23] Luzin V. P. Complex Investigation of the Longitudinal fiber Chrisolit-Asbestos Field (In Russian) [Kompleksnye Issledovaniya Prodol'novoloknistogo Hrizotilasbesta Bazhenovskogo Mestorozhdeniya] // Available online: http://resources.krc.karelia.ru/krc/doc/publ2011/miner_tech_ocenka_118-126.pdf (accessed on 8 October 2021).
- [24] Ore image segmentation method based on u-net and watershed / H. Li, C. Pan, A. Chen, Z.and Wulamu [и др.] // Comput. Mater. Contin. 2020. Т. 65. С. 563–578.
- [25] Ronneberger Olaf, Fischer Philipp, Brox Thomas. U-net: Convolutional networks for biomedical image segmentation // International Conference on Medical image computing and computer-assisted intervention / Springer. 2015. С. 234–241.

- [26] U-net and its variants for medical image segmentation: A review of theory and applications / Nahian Siddique, Sidike Paheding, Colin P Elkin [и др.] // Ieee Access. 2021. Т. 9. С. 82031–82057.
- [27] U-Net-Based Medical Image Segmentation / Xiao-Xia Yin, Le Sun, Yuhan Fu [и др.] // Journal of Healthcare Engineering. 2022. Т. 2022.
- [28] A State-of-the-art Survey of U-Net in Microscopic Image Analysis: from Simple Usage to Structure Mortification / Jian Wu, Wanli Liu, Chen Li [и др.] // arXiv preprint arXiv:2202.06465. 2022.
- [29] Gu Wenchao, Bai Shuang, Kong Lingxing. A review on 2D instance segmentation based on deep neural networks // Image and Vision Computing. 2022. с. 104401.
- [30] Hafiz Abdul Mueed, Bhat Ghulam Mohiuddin. A survey on instance segmentation: state of the art // International journal of multimedia information retrieval. 2020. С. 1–19.
- [31] Mask r-cnn / Kaiming He, Georgia Gkioxari, Piotr Dollár [и др.] // Proceedings of the IEEE international conference on computer vision. 2017. С. 2961–2969.
- [32] Deep residual learning for image recognition / Kaiming He, Xiangyu Zhang, Shaoqing Ren [и др.] // Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. С. 770–778.
- [33] Feature pyramid networks for object detection / Tsung-Yi Lin, Piotr Dollár, Ross Girshick [и др.] // Proceedings of the IEEE conference on computer vision and pattern recognition. 2017. С. 2117–2125.
- [34] Ramesh C Sreya [и др.]. A review on instance segmentation using mask R-CNN // Proceedings of the International Conference on Systems, Energy & Environment (ICSEE). 2021.
- [35] Beucher Serge. Use of watersheds in contour detection // Proceedings of the International Workshop on Image Processing / CCETT. 1979.
- [36] A method of blasted rock image segmentation based on improved watershed algorithm / Qinpeng Guo, Yuchen Wang, Shijiao Yang [и др.] // Scientific Reports. 2022. Т. 12, № 1. С. 1–21.
- [37] Automatic muck pile characterization from UAV images / Fabian Schenk, Alexander Tscharf, Gerhard Mayer [и др.] // ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences. 2019. Т. 4, № W5. С. 163–170.
- [38] Bamford Thomas, Esmaeili Kamran, Schoellig Angela P. A deep learning approach for rock fragmentation analysis // International Journal of Rock Mechanics and Mining Sciences. 2021. Т. 145. с. 104839.
- [39] Jocher Glenn. ultralytics/yolov5: v3.1 - Bug Fixes and Performance Improvements. 2020. . URL: <https://github.com/ultralytics/yolov5>.

- [40] Hossain Sabir, Lee Deok-jin. Deep learning-based real-time multiple-object detection and tracking from aerial imagery via a flying robot with GPU-based embedded devices // Sensors. 2019. Т. 19, № 15. с. 3371.
- [41] A survey of modern deep learning based object detection models / Syed Sahil Abbas Zaidi, Mohammad Samar Ansari, Asra Aslam [и др.] // Digital Signal Processing. 2022. с. 103514.
- [42] Review the state-of-the-art technologies of semantic segmentation based on deep learning / Yujian Mo, Yan Wu, Xinneng Yang [и др.] // Neurocomputing. 2022. Т. 493. С. 626–646.
- [43] Image segmentation using deep learning: A survey / Shervin Minaee, Yuri Y Boykov, Fatih Porikli [и др.] // IEEE transactions on pattern analysis and machine intelligence. 2021.
- [44] Yuan Xiaohui, Shi Jianfang, Gu Lichuan. A review of deep learning methods for semantic segmentation of remote sensing imagery // Expert Systems with Applications. 2021. Т. 169. с. 114417.
- [45] Segmenter: Transformer for semantic segmentation / Robin Strudel, Ricardo Garcia, Ivan Laptev [и др.] // Proceedings of the IEEE/CVF International Conference on Computer Vision. 2021. С. 7262–7272.
- [46] Swin transformer: Hierarchical vision transformer using shifted windows / Ze Liu, Yutong Lin, Yue Cao [и др.] // Proceedings of the IEEE/CVF International Conference on Computer Vision. 2021. С. 10012–10022.
- [47] Fukushima Kunihiko. Cognitron: A self-organizing multilayered neural network // Biological cybernetics. 1975. Т. 20, № 3. С. 121–136.
- [48] Hubel David H, Wiesel Torsten N. Receptive fields and functional architecture of monkey striate cortex // The Journal of physiology. 1968. Т. 195, № 1. С. 215–243.
- [49] Hubel David H, Wiesel Torsten N. Receptive fields of single neurones in the cat's striate cortex // The Journal of physiology. 1959. Т. 148, № 3. С. 574–591.
- [50] Roberts Lawrence G. Machine perception of three-dimensional solids. Ph.D. thesis: Massachusetts Institute of Technology. 1963.
- [51] Fukushima Kunihiko, Miyake Sei. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition // Competition and cooperation in neural nets. Springer, 1982. С. 267–285.
- [52] Аксёнов С.В., Новосельцев В.Б. Повышение качества распознавания сцен нейронной сетью "Неокогнитрон". 2006. № 7.
- [53] Hubel David H, Wiesel Torsten N. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex // The Journal of physiology. 1962. Т. 160, № 1. С. 106–154.

- [54] Marr David. Vision: A computational investigation into the human representation and processing of visual information, henry holt and co // Inc., New York, NY. 1982. T. 2, № 4.2.
- [55] Zhang Wei [и др.]. Shift-invariant pattern recognition neural network and its optical architecture // Proceedings of annual conference of the Japan Society of Applied Physics. 1988.
- [56] A shift-invariant neural network for the lung field segmentation in chest radiography / Akira Hasegawa, Shih-Chung B Lo, Jyh-Shyan Lin [и др.] // Journal of VLSI signal processing systems for signal, image and video technology. 1998. T. 18, № 3. C. 241–250.
- [57] A survey of convolutional neural networks: analysis, applications, and prospects / Zewen Li, Fan Liu, Wenjie Yang [и др.] // IEEE Transactions on Neural Networks and Learning Systems. 2021.
- [58] Phoneme recognition using time-delay neural networks / Alex Waibel, Toshiyuki Hanazawa, Geoffrey Hinton [и др.] // IEEE transactions on acoustics, speech, and signal processing. 1989. T. 37, № 3. C. 328–339.
- [59] Backpropagation applied to handwritten zip code recognition / Yann LeCun, Bernhard Boser, John S Denker [и др.] // Neural computation. 1989. T. 1, № 4. C. 541–551.
- [60] LeCun Yann [и др.]. Generalization and network design strategies // Connectionism in perspective. 1989. T. 19. C. 143–155.
- [61] Bridle John S. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters // Advances in neural information processing systems. 1990. C. 211–217.
- [62] Bridle John S. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition // Neurocomputing. Springer, 1990. C. 227–236.
- [63] Face recognition: A convolutional neural-network approach / Steve Lawrence, C Lee Giles, Ah Chung Tsoi [и др.] // IEEE transactions on neural networks. 1997. T. 8, № 1. C. 98–113.
- [64] Gradient-based learning applied to document recognition / Yann LeCun, Léon Bottou, Yoshua Bengio [и др.] // Proceedings of the IEEE. 1998. T. 86, № 11. C. 2278–2324.
- [65] The MNIST database of handwritten digits. URL: <http://yann.lecun.com/exdb/mnist/>.
- [66] Weng Juyang, Ahuja Narendra, Huang Thomas S. Cresceptron: a self-organizing neural network which grows adaptively // [Proceedings 1992] IJCNN International Joint Conference on Neural Networks / IEEE. T. 1. 1992. C. 576–581.
- [67] Schmidhuber Jürgen. Deep learning in neural networks: An overview // Neural networks. 2015. T. 61. C. 85–117.

- [68] LeCun Yann, Bengio Yoshua, Hinton Geoffrey. Deep learning // nature. 2015. Т. 521, № 7553. С. 436–444.
- [69] Efficient backprop / Yann LeCun, Leon Bottou, G Orr [и др.] // Neural Networks: Tricks of the Trade. New York: Springer. 1998.
- [70] Hinton Geoffrey E, Osindero Simon, Teh Yee-Whye. A fast learning algorithm for deep belief nets // Neural computation. 2006. Т. 18, № 7. С. 1527–1554.
- [71] Hinton Geoffrey E. To recognize shapes, first learn to generate images // Progress in brain research. 2007. Т. 165. С. 535–547.
- [72] Goodfellow Ian, Bengio Yoshua, Courville Aaron. Deep learning. MIT press, 2016.
- [73] Ackley David H, Hinton Geoffrey E, Sejnowski Terrence J. A learning algorithm for Boltzmann machines // Cognitive science. 1985. Т. 9, № 1. С. 147–169.
- [74] Parallel distributed processing / James L McClelland, David E Rumelhart, PDP Research Group [и др.]. MIT press Cambridge, MA, 1986. Т. 2.
- [75] Hopfield John J. Neural networks and physical systems with emergent collective computational abilities // Proceedings of the national academy of sciences. 1982. Т. 79, № 8. С. 2554–2558.
- [76] Hinton Geoffrey E, Salakhutdinov Ruslan R. Reducing the dimensionality of data with neural networks // science. 2006. Т. 313, № 5786. С. 504–507.
- [77] Ballard Dana H. Modular learning in neural networks. // AAAI. Т. 647. 1987. С. 279–284.
- [78] Why does unsupervised pre-training help deep learning? / Dumitru Erhan, Aaron Courville, Yoshua Bengio [и др.] // Proceedings of the thirteenth international conference on artificial intelligence and statistics / JMLR Workshop and Conference Proceedings. 2010. С. 201–208.
- [79] Greedy layer-wise training of deep networks / Yoshua Bengio, Pascal Lamblin, Dan Popovici [и др.] // Advances in neural information processing systems. 2007. С. 153–160.
- [80] Oh Kyoung-Su, Jung Keechul. GPU implementation of neural networks // Pattern Recognition. 2004. Т. 37, № 6. С. 1311–1314.
- [81] Steinkraus Dave, Buck Ian, Simard PY. Using GPUs for machine learning algorithms // Eighth International Conference on Document Analysis and Recognition (ICDAR'05) / IEEE. 2005. С. 1115–1120.
- [82] Chellapilla Kumar, Puri Sidd, Simard Patrice. High performance convolutional neural networks for document processing // Tenth international workshop on frontiers in handwriting recognition / Suvisoft. 2006.
- [83] Bengio Yoshua, LeCun Yann [и др.]. Scaling learning algorithms towards AI // Large-scale kernel machines. 2007. Т. 34, № 5. С. 1–41.

- [84] What is the best multi-stage architecture for object recognition? / Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato [и др.] // 2009 IEEE 12th international conference on computer vision / IEEE. 2009. C. 2146–2153.
- [85] Nair Vinod, Hinton Geoffrey E. Rectified linear units improve restricted boltzmann machines // Icml. 2010.
- [86] Activation functions / Mohit Goyal, Rajan Goyal, P Venkatappa Reddy [и др.] // Deep learning: Algorithms and applications. Springer, 2020. C. 1–30.
- [87] Machine learning and deep learning frameworks and libraries for large-scale data mining: a survey / Giang Nguyen, Stefan Dlugolinsky, Martin Bobák [и др.] // Artificial Intelligence Review. 2019. Т. 52, № 1. C. 77–124.
- [88] Deep, big, simple neural nets for handwritten digit recognition / Dan Claudiu Cireşan, Ueli Meier, Luca Maria Gambardella [и др.] // Neural computation. 2010. Т. 22, № 12. C. 3207–3220.
- [89] Flexible, high performance convolutional neural networks for image classification / Dan Claudiu Ciresan, Ueli Meier, Jonathan Masci [и др.] // Twenty-second international joint conference on artificial intelligence. 2011.
- [90] The CIFAR-10 dataset. URL: <https://www.cs.toronto.edu/~kriz/cifar.html>.
- [91] Theano: A CPU and GPU math compiler in Python / James Bergstra, Olivier Breuleux, Frédéric Bastien [и др.] // Proc. 9th python in science conf. Т. 1. 2010. C. 3–10.
- [92] The deep learning compiler: A comprehensive survey / Mingzhen Li, Yi Liu, Xiaoyan Liu [и др.] // IEEE Transactions on Parallel and Distributed Systems. 2020. Т. 32, № 3. C. 708–727.
- [93] Krizhevsky Alex, Hinton Geoff. Convolutional deep belief networks on cifar-10 // Unpublished manuscript. 2010. Т. 40, № 7. C. 1–9.
- [94] A committee of neural networks for traffic sign classification / Dan Cireşan, Ueli Meier, Jonathan Masci [и др.] // The 2011 international joint conference on neural networks / IEEE. 2011. C. 1918–1921.
- [95] Ciregan Dan, Meier Ueli, Schmidhuber Jürgen. Multi-column deep neural networks for image classification // 2012 IEEE conference on computer vision and pattern recognition / IEEE. 2012. C. 3642–3649.
- [96] ImageNet: A Large-Scale Hierarchical Image Database / J. Deng, W. Dong, R. Socher [и др.] // CVPR09. 2009.
- [97] ImageNet Large Scale Visual Recognition Challenge / Olga Russakovsky, Jia Deng, Hao Su [и др.] // International Journal of Computer Vision (IJCV). 2015. Т. 115, № 3. C. 211–252.

- [98] ImageNet web site. URL: <https://image-net.org/challenges/LSVRC/2012/index.php>.
- [99] Krizhevsky Alex, Sutskever Ilya, Hinton Geoffrey E. Imagenet classification with deep convolutional neural networks // Advances in neural information processing systems. 2012. T. 25. C. 1097–1105.
- [100] Image Classification on ImageNet. PapersWithCode.com. URL: <https://paperswithcode.com/sota/image-classification-on-imagenet>.
- [101] Prabhu Vinay Uday, Birhane Abeba. Large image datasets: A pyrrhic win for computer vision? // arXiv preprint arXiv:2006.16923. 2020.
- [102] Poggio Tomaso, Girosi Federico. A theory of networks for approximation and learning: Tech. Rep.: : Massachusetts INST of TECH Cambridge Artificial Intelligence LAB, 1989.
- [103] Girosi Federico, Jones Michael, Poggio Tomaso. Regularization theory and neural networks architectures // Neural computation. 1995. T. 7, № 2. C. 219–269.
- [104] Tihonov Andrei Nikolajevits. Solution of incorrectly formulated problems and the regularization method // Soviet Math. 1963. T. 4. C. 1035–1038.
- [105] Tibshirani Robert. Regression shrinkage and selection via the lasso // Journal of the Royal Statistical Society: Series B (Methodological). 1996. T. 58, № 1. C. 267–288.
- [106] Zou Hui, Hastie Trevor. Regularization and variable selection via the elastic net // Journal of the royal statistical society: series B (statistical methodology). 2005. T. 67, № 2. C. 301–320.
- [107] Moradi Reza, Berangi Reza, Minaei Behrouz. A survey of regularization strategies for deep models // Artificial Intelligence Review. 2020. T. 53, № 6. C. 3947–3986.
- [108] Krogh Anders, Hertz John. A simple weight decay can improve generalization // Advances in neural information processing systems. 1991. T. 4.
- [109] Three mechanisms of weight decay regularization / Guodong Zhang, Chaoqi Wang, Bowen Xu [и др.] // arXiv preprint arXiv:1810.12281. 2018.
- [110] Best practices for convolutional neural networks applied to visual document analysis. / Patrice Y Simard, David Steinkraus, John C Platt [и др.] // Icdar. T. 3. 2003.
- [111] Baird Henry S. Document image defect models // Structured Document Image Analysis. Springer, 1992. C. 546–556.
- [112] Shorten Connor, Khoshgoftaar Taghi M. A survey on image data augmentation for deep learning // Journal of Big Data. 2019. T. 6, № 1. C. 1–48.
- [113] Lewy Dominik, Mańdziuk Jacek. An overview of mixing augmentation methods and augmentation strategies // Artificial Intelligence Review. 2022. C. 1–59.

- [114] Generative adversarial networks (GANs) for image augmentation in agriculture: A systematic review / Yuzhen Lu, Dong Chen, Ebenezer Olaniyi [и др.] // Computers and Electronics in Agriculture. 2022. Т. 200. с. 107208.
- [115] Autoaugment: Learning augmentation strategies from data / Ekin D Cubuk, Barret Zoph, Dandelion Mane [и др.] // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019. С. 113–123.
- [116] A Survey of Automated Data Augmentation Algorithms for Deep Learning-based Image Classification Tasks / Zihan Yang, Richard O Sinnott, James Bailey [и др.] // arXiv preprint arXiv:2206.06544. 2022.
- [117] Randaugment: Practical automated data augmentation with a reduced search space / Ekin D Cubuk, Barret Zoph, Jonathon Shlens [и др.] // Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops. 2020. С. 702–703.
- [118] Augment your batch: Improving generalization through instance repetition / Elad Hoffer, Tal Ben-Nun, Itay Hubara [и др.] // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020. С. 8129–8138.
- [119] Khalifa Nour Eldeen, Loey Mohamed, Mirjalili Seyedali. A comprehensive survey of recent trends in deep learning for digital images augmentation // Artificial Intelligence Review. 2021. С. 1–27.
- [120] mixup: Beyond empirical risk minimization / Hongyi Zhang, Moustapha Cisse, Yann N Dauphin [и др.] // arXiv preprint arXiv:1710.09412. 2017.
- [121] Going deeper with convolutions / Christian Szegedy, Wei Liu, Yangqing Jia [и др.] // Proceedings of the IEEE conference on computer vision and pattern recognition. 2015. С. 1–9.
- [122] Rethinking the inception architecture for computer vision / Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe [и др.] // Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. С. 2818–2826.
- [123] Distilling the knowledge in a neural network / Geoffrey Hinton, Oriol Vinyals, Jeff Dean [и др.] // arXiv preprint arXiv:1503.02531. 2015. Т. 2, № 7.
- [124] Glorot Xavier, Bengio Yoshua. Understanding the difficulty of training deep feedforward neural networks // Proceedings of the thirteenth international conference on artificial intelligence and statistics / JMLR Workshop and Conference Proceedings. 2010. С. 249–256.
- [125] Delving deep into rectifiers: Surpassing human-level performance on imagenet classification / Kaiming He, Xiangyu Zhang, Shaoqing Ren [и др.] // Proceedings of the IEEE international conference on computer vision. 2015. С. 1026–1034.

- [126] Narkhede Meenal V, Bartakke Prashant P, Sutaone Mukul S. A review on weight initialization strategies for neural networks // Artificial Intelligence Review. 2021. C. 1–32.
- [127] Improving neural networks by preventing co-adaptation of feature detectors / Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky [и др.] // arXiv preprint arXiv:1207.0580. 2012.
- [128] Labach Alex, Salehinejad Hojjat, Valaee Shahrokh. Survey of dropout methods for deep neural networks // arXiv preprint arXiv:1904.13310. 2019.
- [129] Baldi Pierre, Sadowski Peter J. Understanding dropout // Advances in neural information processing systems. 2013. T. 26. C. 2814–2822.
- [130] Dropout: a simple way to prevent neural networks from overfitting / Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky [и др.] // The journal of machine learning research. 2014. T. 15, № 1. C. 1929–1958.
- [131] Self-normalizing neural networks / Günter Klambauer, Thomas Unterthiner, Andreas Mayr [и др.] // Advances in neural information processing systems. 2017. T. 30.
- [132] Efficient object localization using convolutional networks / Jonathan Tompson, Ross Goroshin, Arjun Jain [и др.] // Proceedings of the IEEE conference on computer vision and pattern recognition. 2015. C. 648–656.
- [133] Ghiasi Golnaz, Lin Tsung-Yi, Le Quoc V. Dropblock: A regularization method for convolutional networks // Advances in neural information processing systems. 2018. T. 31.
- [134] Wu Haibing, Gu Xiaodong. Max-pooling dropout for regularization of convolutional neural networks // International Conference on Neural Information Processing / Springer. 2015. C. 46–54.
- [135] Deep networks with stochastic depth / Gao Huang, Yu Sun, Zhuang Liu [и др.] // European conference on computer vision / Springer. 2016. C. 646–661.
- [136] Ioffe Sergey, Szegedy Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift // International conference on machine learning / PMLR. 2015. C. 448–456.
- [137] Normalization techniques in training dnns: Methodology, analysis and application / Lei Huang, Jie Qin, Yi Zhou [и др.] // arXiv preprint arXiv:2009.12836. 2020.
- [138] Summers Cecilia, Dinneen Michael J. Four things everyone should know to improve batch normalization // arXiv preprint arXiv:1906.03548. 2019.
- [139] Dive into Deep Learning / Aston Zhang, Zachary C. Lipton, Mu Li [и др.] // arXiv preprint arXiv:2106.11342. 2021.

- [140] Lubana Ekdeep Singh, Dick Robert P, Tanaka Hidenori. Beyond BatchNorm: Towards a General Understanding of Normalization in Deep Learning // arXiv preprint arXiv:2106.05956. 2021.
- [141] Towards understanding regularization in batch normalization / Ping Luo, Xinjiang Wang, Wenqi Shao [и др.] // arXiv preprint arXiv:1809.00846. 2018.
- [142] Understanding the disharmony between dropout and batch normalization by variance shift / Xiang Li, Shuo Chen, Xiaolin Hu [и др.] // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019. C. 2682–2690.
- [143] Ba Jimmy Lei, Kiros Jamie Ryan, Hinton Geoffrey E. Layer normalization // arXiv preprint arXiv:1607.06450. 2016.
- [144] Wu Yuxin, He Kaiming. Group normalization // Proceedings of the European conference on computer vision (ECCV). 2018. C. 3–19.
- [145] Micro-batch training with batch-channel normalization and weight standardization / Siyuan Qiao, Huiyu Wang, Chenxi Liu [и др.] // arXiv preprint arXiv:1903.10520. 2019.
- [146] Sik-Ho Tsang. Summary: My Paper Reading Lists, Tutorials & Sharings. 2020. URL: <https://sh-tsang.medium.com/overview-my-reviewed-paper-lists-tutorials-946ce59fbf9e>.
- [147] Zeiler Matthew D, Fergus Rob. Visualizing and understanding convolutional networks // European conference on computer vision / Springer. 2014. C. 818–833.
- [148] Simonyan Karen, Zisserman Andrew. Very deep convolutional networks for large-scale image recognition // arXiv preprint arXiv:1409.1556. 2014.
- [149] Lin Min, Chen Qiang, Yan Shuicheng. Network in network // arXiv preprint arXiv:1312.4400. 2013.
- [150] Inception-v4, inception-resnet and the impact of residual connections on learning / Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke [и др.] // Thirty-first AAAI conference on artificial intelligence. 2017.
- [151] A survey of the recent architectures of deep convolutional neural networks / Asifullah Khan, Anabia Sohail, Umme Zahoor [и др.] // Artificial Intelligence Review. 2020. Т. 53, № 8. С. 5455–5516.
- [152] Identity mappings in deep residual networks / Kaiming He, Xiangyu Zhang, Shaoqing Ren [и др.] // European conference on computer vision / Springer. 2016. C. 630–645.
- [153] A state-of-the-art survey on deep learning theory and architectures / Md Zahangir Alom, Tarek M Taha, Chris Yakopcic [и др.] // Electronics. 2019. Т. 8, № 3. с. 292.
- [154] Aggregated residual transformations for deep neural networks / Saining Xie, Ross Girshick, Piotr Dollár [и др.] // Proceedings of the IEEE conference on computer vision and pattern recognition. 2017. C. 1492–1500.

- [155] Sik-Ho Tsang. Review: Trimps-Soushen — Winner in ILSVRC 2016 (Image Classification). 2018. URL: <https://towardsdatascience.com/review-trimps-soushen-winner-in-ilsvrc-2016-image-classification-dfbc423111dd>.
- [156] Wu Zifeng, Shen Chunhua, Van Den Hengel Anton. Wider or deeper: Revisiting the resnet model for visual recognition // Pattern Recognition. 2019. Т. 90. С. 119–133.
- [157]
- [158] Mobilenets: Efficient convolutional neural networks for mobile vision applications / Andrew G Howard, Menglong Zhu, Bo Chen [и др.] // arXiv preprint arXiv:1704.04861. 2017.
- [159] Densely connected convolutional networks / Gao Huang, Zhuang Liu, Laurens Van Der Maaten [и др.] // Proceedings of the IEEE conference on computer vision and pattern recognition. 2017. С. 4700–4708.
- [160] Big transfer (bit): General visual representation learning / Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai [и др.] // European conference on computer vision / Springer. 2020. С. 491–507.
- [161] Kolesnikov Alexander, Beyer Lucas, Zhai Xiaohua [и др.]. Big Transfer (BiT): General Visual Representation Learning Supplementary Material.
- [162] Bag of tricks for image classification with convolutional neural networks / Tong He, Zhi Zhang, Hang Zhang [и др.] // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019. С. 558–567.
- [163] Da Silva Icamaan B Viegas, Adeodato Paulo JL. PCA and Gaussian noise in MLP neural network training improve generalization in problems with small and unbalanced data sets // The 2011 International Joint Conference on Neural Networks / IEEE. 2011. С. 2664–2669.
- [164] Dozat Timothy. Incorporating nesterov momentum into adam. 2016.
- [165] Wightman Ross, Touvron Hugo, Jégou Hervé. Resnet strikes back: An improved training procedure in timm // arXiv preprint arXiv:2110.00476. 2021.
- [166] Large batch optimization for deep learning: Training bert in 76 minutes / Yang You, Jing Li, Sashank Reddi [и др.] // arXiv preprint arXiv:1904.00962. 2019.
- [167] Revisiting resnets: Improved training and scaling strategies / Irwan Bello, William Fedus, Xianzhi Du [и др.] // Advances in Neural Information Processing Systems. 2021. Т. 34. С. 22614–22627.
- [168] Hu Jie, Shen Li, Sun Gang. Squeeze-and-excitation networks // Proceedings of the IEEE conference on computer vision and pattern recognition. 2018. С. 7132–7141.
- [169] Mishra Rahul, Gupta Hari Prabhat, Dutta Tania. A survey on deep neural network compression: Challenges, overview, and solutions // arXiv preprint arXiv:2010.03954. 2020.

- [170] Minerva: Enabling low-power, highly-accurate deep neural network accelerators / Brandon Reagen, Paul Whatmough, Robert Adolf [и др.] // 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA) / IEEE. 2016. C. 267–278.
- [171] SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and < 0.5 MB model size / Forrest N Iandola, Song Han, Matthew W Moskewicz [и др.] // arXiv preprint arXiv:1602.07360. 2016.
- [172] He Kaiming, Sun Jian. Convolutional neural networks at constrained time cost // Proceedings of the IEEE conference on computer vision and pattern recognition. 2015. C. 5353–5360.
- [173] Hollemans Matthijs. New mobile neural network architectures. 2020.
- [174] Shufflenet: An extremely efficient convolutional neural network for mobile devices / Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin [и др.] // Proceedings of the IEEE conference on computer vision and pattern recognition. 2018. C. 6848–6856.
- [175] Shufflenet v2: Practical guidelines for efficient cnn architecture design / Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng [и др.] // Proceedings of the European conference on computer vision (ECCV). 2018. C. 116–131.
- [176] Mobilenetv2: Inverted residuals and linear bottlenecks / Mark Sandler, Andrew Howard, Menglong Zhu [и др.] // Proceedings of the IEEE conference on computer vision and pattern recognition. 2018. C. 4510–4520.
- [177] Striving for simplicity: The all convolutional net / Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox [и др.] // arXiv preprint arXiv:1412.6806. 2014.
- [178] Repvgg: Making vgg-style convnets great again / Xiaohan Ding, Xiangyu Zhang, Ningning Ma [и др.] // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021. C. 13733–13742.
- [179] Itti Laurent, Koch Christof. Computational modelling of visual attention // Nature reviews neuroscience. 2001. Т. 2, № 3. C. 194–203.
- [180] Bahdanau Dzmitry, Cho Kyunghyun, Bengio Yoshua. Neural machine translation by jointly learning to align and translate // arXiv preprint arXiv:1409.0473. 2014.
- [181] An attentive survey of attention models / Sneha Chaudhari, Varun Mithal, Gungor Polatkan [и др.] // arXiv preprint arXiv:1904.02874. 2019.
- [182] Cheng Jianpeng, Dong Li, Lapata Mirella. Long short-term memory-networks for machine reading // arXiv preprint arXiv:1601.06733. 2016.
- [183] A decomposable attention model for natural language inference / Ankur P Parikh, Oscar Täckström, Dipanjan Das [и др.] // arXiv preprint arXiv:1606.01933. 2016.

- [184] Attention is all you need / Ashish Vaswani, Noam Shazeer, Niki Parmar [и др.] // Advances in neural information processing systems. 2017. C. 5998–6008.
- [185] Weng Lilian. Attention? Attention! 2021. URL: <http://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html>.
- [186] Yang Xiao. An Overview of the Attention Mechanisms in Computer Vision // Journal of Physics: Conference Series / IOP Publishing. T. 1693. 2020. c. 012173.
- [187] Cordonnier Jean-Baptiste, Loukas Andreas, Jaggi Martin. On the relationship between self-attention and convolutional layers // arXiv preprint arXiv:1911.03584. 2019.
- [188] Cbam: Convolutional block attention module / Sanghyun Woo, Jongchan Park, Joon-Young Lee [и др.] // Proceedings of the European conference on computer vision (ECCV). 2018. C. 3–19.
- [189] Searching for mobilenetv3 / Andrew Howard, Mark Sandler, Grace Chu [и др.] // Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019. C. 1314–1324.
- [190] Non-local neural networks / Xiaolong Wang, Ross Girshick, Abhinav Gupta [и др.] // Proceedings of the IEEE conference on computer vision and pattern recognition. 2018. C. 7794–7803.
- [191] Gcnet: Non-local networks meet squeeze-excitation networks and beyond / Yue Cao, Jiarui Xu, Stephen Lin [и др.] // Proceedings of the IEEE/CVF international conference on computer vision workshops. 2019. C. 0–0.
- [192] Weng Lilian. Neural Architecture Search. 2020. URL: <http://lilianweng.github.io/lil-log/2020/08/06/neural-architecture-search.html>.
- [193] Elskens Thomas, Metzen Jan Hendrik, Hutter Frank. Neural architecture search: A survey // The Journal of Machine Learning Research. 2019. T. 20, № 1. C. 1997–2017.
- [194] Guyon I., et al. Analysis of the AutoML Challenge series 2015-2018 // Hutter F., Kotthoff L., Vanschoren J. (eds) Automated Machine Learning. The Springer Series on Challenges in Machine Learning / Springer. 2019.
- [195] Hutter Frank, Kotthoff Lars, Vanschoren Joaquin. Automated machine learning: methods, systems, challenges. Springer Nature, 2019.
- [196] He Xin, Zhao Kaiyong, Chu Xiaowen. AutoML: A Survey of the State-of-the-Art // Knowledge-Based Systems. 2021. T. 212. c. 106622.
- [197] Kyriakides George, Margaritis Konstantinos. An introduction to neural architecture search for convolutional networks // arXiv preprint arXiv:2005.11074. 2020.
- [198] Learning transferable architectures for scalable image recognition / Barret Zoph, Vijay Vasudevan, Jonathon Shlens [и др.] // Proceedings of the IEEE conference on computer vision and pattern recognition. 2018. C. 8697–8710.

- [199] Progressive neural architecture search / Chenxi Liu, Barret Zoph, Maxim Neumann [и др.] // Proceedings of the European conference on computer vision (ECCV). 2018. C. 19–34.
- [200] Zoph Barret, Le Quoc V. Neural architecture search with reinforcement learning // arXiv preprint arXiv:1611.01578. 2016.
- [201] Tan Mingxing, Le Quoc. Efficientnet: Rethinking model scaling for convolutional neural networks // International Conference on Machine Learning / PMLR. 2019. C. 6105–6114.
- [202] Tan Mingxing, Le Quoc V. Efficientnetv2: Smaller models and faster training // arXiv preprint arXiv:2104.00298. 2021.
- [203] Designing network design spaces / Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick [и др.] // Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020. C. 10428–10436.
- [204] A survey on visual transformer / Kai Han, Yunhe Wang, Hanting Chen [и др.] // arXiv preprint arXiv:2012.12556. 2020.
- [205] A Survey of Transformers / Tianyang Lin, Yuxin Wang, Xiangyang Liu [и др.] // arXiv preprint arXiv:2106.04554. 2021.
- [206] Bert: Pre-training of deep bidirectional transformers for language understanding / Jacob Devlin, Ming-Wei Chang, Kenton Lee [и др.] // arXiv preprint arXiv:1810.04805. 2018.
- [207] Sun Chen, Shrivastava Abhinav, Singh Saurabh [и др.]. Revisiting Unreasonable Effectiveness of Data in Deep Learning Era. 2017.
- [208] An image is worth 16x16 words: Transformers for image recognition at scale / Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov [и др.] // arXiv preprint arXiv:2010.11929. 2020.
- [209] Transformers in vision: A survey / Salman Khan, Muzammal Naseer, Munawar Hayat [и др.] // arXiv preprint arXiv:2101.01169. 2021.
- [210] Scaling vision transformers / Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby [и др.] // arXiv preprint arXiv:2106.04560. 2021.
- [211] Gidaris Spyros, Singh Praveer, Komodakis Nikos. Unsupervised representation learning by predicting image rotations // arXiv preprint arXiv:1803.07728. 2018.
- [212] Jing Longlong, Tian Yingli. Self-supervised visual feature learning with deep neural networks: A survey // IEEE transactions on pattern analysis and machine intelligence. 2020.
- [213] Bottleneck transformers for visual recognition / Aravind Srinivas, Tsung-Yi Lin, Niki Parmar [и др.] // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021. C. 16519–16529.

- [214] Do Vision Transformers See Like Convolutional Neural Networks? / Maithra Raghu, Thomas Unterthiner, Simon Kornblith [и др.] // arXiv preprint arXiv:2108.08810. 2021.
- [215] Training data-efficient image transformers & distillation through attention / Hugo Touvron, Matthieu Cord, Matthijs Douze [и др.] // International Conference on Machine Learning / PMLR. 2021. C. 10347–10357.
- [216] Volo: Vision outlouker for visual recognition / Li Yuan, Qibin Hou, Zihang Jiang [и др.] // arXiv preprint arXiv:2106.13112. 2021.
- [217] Coatnet: Marrying convolution and attention for all data sizes / Zihang Dai, Hanxiao Liu, Quoc V Le [и др.] // Advances in Neural Information Processing Systems. 2021. T. 34. C. 3965–3977.
- [218] Maxvit: Multi-axis vision transformer / Zhengzhong Tu, Hossein Talebi, Han Zhang [и др.] // arXiv preprint arXiv:2204.01697. 2022.
- [219] Mehta Sachin, Rastegari Mohammad. Mobilevit: light-weight, general-purpose, and mobile-friendly vision transformer // arXiv preprint arXiv:2110.02178. 2021.
- [220] Meta-learning in neural networks: A survey / Timothy M Hospedales, Antreas Antoniou, Paul Micaelli [и др.] // IEEE transactions on pattern analysis and machine intelligence. 2021.
- [221] Naveed Humza. Survey: Image mixing and deleting for data augmentation // arXiv preprint arXiv:2106.07085. 2021.
- [222] A convnet for the 2020s / Zhuang Liu, Hanzi Mao, Chao-Yuan Wu [и др.] // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022. C. 11976–11986.
- [223] Can Attention Enable MLPs To Catch Up With CNNs? / Meng-Hao Guo, Zheng-Ning Liu, Tai-Jiang Mu [и др.] // arXiv preprint arXiv:2105.15078. 2021.
- [224] Mlp-mixer: An all-mlp architecture for vision / Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov [и др.] // arXiv preprint arXiv:2105.01601. 2021.
- [225] Resmlp: Feedforward networks for image classification with data-efficient training / Hugo Touvron, Piotr Bojanowski, Mathilde Caron [и др.] // IEEE Transactions on Pattern Analysis and Machine Intelligence. 2022.
- [226] Metaformer is actually what you need for vision / Weihao Yu, Mi Luo, Pan Zhou [и др.] // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022. C. 10819–10829.
- [227] Repmlp: Re-parameterizing convolutions into fully-connected layers for image recognition / Xiaohan Ding, Chunlong Xia, Xiangyu Zhang [и др.] // arXiv preprint arXiv:2105.01883. 2021.

- [228] Global filter networks for image classification / Yongming Rao, Wenliang Zhao, Zheng Zhu [и др.] // Advances in Neural Information Processing Systems. 2021. Т. 34. С. 980–993.
- [229] Adaptive fourier neural operators: Efficient token mixers for transformers / John Guibas, Morteza Mardani, Zongyi Li [и др.] // arXiv preprint arXiv:2111.13587. 2021.
- [230] What Makes for Hierarchical Vision Transformer? / Yuxin Fang, Xinggang Wang, Rui Wu [и др.] // arXiv preprint arXiv:2107.02174. 2021.
- [231] AMixer: Adaptive Weight Mixing for Self-attention Free Vision Transformers / Yongming Rao, Wenliang Zhao, Jie Zhou [и др.] // European Conference on Computer Vision / Springer. 2022. С. 50–67.
- [232] Sultana Farhana, Sufian Abu, Dutta Paramartha. Evolution of image segmentation using deep convolutional neural network: a survey // Knowledge-Based Systems. 2020. Т. 201. с. 106062.
- [233] Thoma Martin. A survey of semantic segmentation // arXiv preprint arXiv:1602.06541. 2016.
- [234] Learning hierarchical features for scene labeling / Clement Farabet, Camille Couprie, Laurent Najman [и др.] // IEEE transactions on pattern analysis and machine intelligence. 2012. Т. 35, № 8. С. 1915–1929.
- [235] Jadon Shruti. A survey of loss functions for semantic segmentation // 2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB) / IEEE. 2020. С. 1–7.
- [236] On the usage of average Hausdorff distance for segmentation performance assessment: hidden error when used for ranking / Orhun Utku Aydin, Abdel Aziz Taha, Adam Hilbert [и др.] // European Radiology Experimental. 2021. Т. 5, № 1. С. 1–7.
- [237] Unified Focal loss: Generalising Dice and cross entropy-based losses to handle class imbalanced medical image segmentation / Michael Yeung, Evis Sala, Carola-Bibiane Schönlieb [и др.] // Computerized Medical Imaging and Graphics. 2022. Т. 95. с. 102026.
- [238] Ma Jun. Segmentation loss odyssey // arXiv preprint arXiv:2005.13449. 2020.
- [239] A formal evaluation of PSNR as quality measurement parameter for image segmentation algorithms / Fernando A Fardo, Victor H Conforto, Francisco C de Oliveira [и др.] // arXiv preprint arXiv:1605.07116. 2016.
- [240] Correlation maximized structural similarity loss for semantic segmentation / Shuai Zhao, Boxi Wu, Wenqing Chu [и др.] // arXiv preprint arXiv:1910.08711. 2019.
- [241] Road extraction by using atrous spatial pyramid pooling integrated encoder-decoder network and structural similarity loss / Hao He, Dongfang Yang, Shicheng Wang [и др.] // Remote Sensing. 2019. Т. 11, № 9. с. 1015.

- [242] DCAN: deep contour-aware networks for accurate gland segmentation / Hao Chen, Xiaojuan Qi, Lequan Yu [и др.] // Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. 2016. C. 2487–2496.
- [243] Wang Ziyang. Deep learning in medical ultrasound image segmentation: A review // arXiv preprint arXiv:2002.07703. 2020.
- [244] Improved U-net: fully convolutional network model for skin-lesion segmentation / Karshiev Sanjar, Olimov Bekhzod, Jaeil Kim [и др.] // Applied Sciences. 2020. Т. 10, № 10. с. 3658.
- [245] Dumoulin Vincent, Visin Francesco. A guide to convolution arithmetic for deep learning // arXiv preprint arXiv:1603.07285. 2016.
- [246] Long Jonathan, Shelhamer Evan, Darrell Trevor. Fully convolutional networks for semantic segmentation // Proceedings of the IEEE conference on computer vision and pattern recognition. 2015. C. 3431–3440.
- [247] Sanderson Edward, Matuszewski Bogdan J. FCN-transformer feature fusion for polyp segmentation // Annual Conference on Medical Image Understanding and Analysis / Springer. 2022. C. 892–907.
- [248] Medical Images Segmentation. PapersWithCode.com. URL: <https://paperswithcode.com/task/medical-image-segmentation>.
- [249] Yu Fisher, Koltun Vladlen. Multi-scale context aggregation by dilated convolutions // arXiv preprint arXiv:1511.07122. 2015.
- [250] Semantic image segmentation with deep convolutional nets and fully connected crfs / Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos [и др.] // arXiv preprint arXiv:1412.7062. 2014.
- [251] Pyramid scene parsing network / Hengshuang Zhao, Jianping Shi, Xiaojuan Qi [и др.] // Proceedings of the IEEE conference on computer vision and pattern recognition. 2017. C. 2881–2890.
- [252] Spatial pyramid pooling in deep convolutional networks for visual recognition / Kaiming He, Xiangyu Zhang, Shaoqing Ren [и др.] // IEEE transactions on pattern analysis and machine intelligence. 2015. Т. 37, № 9. С. 1904–1916.
- [253] Deformable convolutional networks / Jifeng Dai, Haozhi Qi, Yuwen Xiong [и др.] // Proceedings of the IEEE international conference on computer vision. 2017. С. 764–773.
- [254] Spatial transformer networks / Max Jaderberg, Karen Simonyan, Andrew Zisserman [и др.] // Advances in neural information processing systems. 2015. Т. 28.

- [255] Deformable convnets v2: More deformable, better results / Xizhou Zhu, Han Hu, Stephen Lin [и др.] // Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019. C. 9308–9316.
- [256] Adaptive deformable convolutional network / Feng Chen, Fei Wu, Jing Xu [и др.] // Neurocomputing. 2021. T. 453. C. 853–864.
- [257] Internimage: Exploring large-scale vision foundation models with deformable convolutions / Wenhui Wang, Jifeng Dai, Zhe Chen [и др.] // arXiv preprint arXiv:2211.05778. 2022.
- [258] Scene parsing through ade20k dataset / Bolei Zhou, Hang Zhao, Xavier Puig [и др.] // Proceedings of the IEEE conference on computer vision and pattern recognition. 2017. C. 633–641.
- [259] Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs / Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos [и др.] // IEEE transactions on pattern analysis and machine intelligence. 2017. T. 40, № 4. C. 834–848.
- [260] Rethinking atrous convolution for semantic image segmentation / Liang-Chieh Chen, George Papandreou, Florian Schroff [и др.] // arXiv preprint arXiv:1706.05587. 2017.
- [261] Noh Hyeonwoo, Hong Seunghoon, Han Bohyung. Learning deconvolution network for semantic segmentation // Proceedings of the IEEE international conference on computer vision. 2015. C. 1520–1528.
- [262] Badrinarayanan Vijay, Handa Ankur, Cipolla Roberto. Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling // arXiv preprint arXiv:1505.07293. 2015.
- [263] Punn Narinder Singh, Agarwal Sonali. Modality specific U-Net variants for biomedical image segmentation: A survey // arXiv preprint arXiv:2107.04537. 2021.
- [264] Semantic Segmentation Models. PapersWithCode.com. URL: <https://paperswithcode.com/methods/category/segmentation-models>.
- [265] PASCAL VOC dataset and challenge. URL: <http://host.robots.ox.ac.uk/pascal/VOC/>.
- [266] Semantic Segmentation on PASCAL VOC 2012 test. URL: <https://paperswithcode.com/sota/semantic-segmentation-on-pascal-voc-2012>.
- [267] Deep semantic segmentation of natural and medical images: a review / Saeid Asgari Taghanaki, Kumar Abhishek, Joseph Paul Cohen [и др.] // Artificial Intelligence Review. 2021. T. 54, № 1. C. 137–178.
- [268] A survey of semi-and weakly supervised semantic segmentation of images / Man Zhang, Yong Zhou, Jiaqi Zhao [и др.] // Artificial Intelligence Review. 2020. T. 53, № 6. C. 4259–4288.

- [269] Unet++: A nested u-net architecture for medical image segmentation / Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh [и др.] // Deep learning in medical image analysis and multimodal learning for clinical decision support. Springer, 2018. C. 3–11.
- [270] Chaurasia Abhishek, Culurciello Eugenio. Linknet: Exploiting encoder representations for efficient semantic segmentation // 2017 IEEE Visual Communications and Image Processing (VCIP) / IEEE. 2017. C. 1–4.
- [271] Zhang Zhengxin, Liu Qingjie, Wang Yunhong. Road extraction by deep residual u-net // IEEE Geoscience and Remote Sensing Letters. 2018. T. 15, № 5. C. 749–753.
- [272] A comprehensive study on colorectal polyp segmentation with ResUNet++, conditional random field and test-time augmentation / Debesh Jha, Pia H Smedsrød, Dag Johansen [и др.] // IEEE journal of biomedical and health informatics. 2021. T. 25, № 6. C. 2029–2040.
- [273] The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation / Simon Jégou, Michal Drozdzal, David Vazquez [и др.] // Proceedings of the IEEE conference on computer vision and pattern recognition workshops. 2017. C. 11–19.
- [274] Mdu-net: Multi-scale densely connected u-net for biomedical image segmentation / Jiawei Zhang, Yuzhen Jin, Jilan Xu [и др.] // arXiv preprint arXiv:1812.00352. 2018.
- [275] Attention u-net: Learning where to look for the pancreas / Ozan Oktay, Jo Schlemper, Loic Le Folgoc [и др.] // arXiv preprint arXiv:1804.03999. 2018.
- [276] Ma-net: A multi-scale attention network for liver and tumor segmentation / Tongle Fan, Guanglei Wang, Yan Li [и др.] // IEEE Access. 2020. T. 8. C. 179656–179665.
- [277] Encoder-decoder with atrous separable convolution for semantic image segmentation / Liang-Chieh Chen, Yukun Zhu, George Papandreou [и др.] // Proceedings of the European conference on computer vision (ECCV). 2018. C. 801–818.
- [278] Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation / Chenxi Liu, Liang-Chieh Chen, Florian Schroff [и др.] // Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019. C. 82–92.
- [279] Attention deeplabv3+: Multi-level context attention mechanism for skin lesion segmentation / Reza Azad, Maryam Asadi-Aghbolaghi, Mahmood Fathy [и др.] // European conference on computer vision / Springer. 2020. C. 251–266.
- [280] Denseaspp for semantic segmentation in street scenes / Maoke Yang, Kun Yu, Chi Zhang [и др.] // Proceedings of the IEEE conference on computer vision and pattern recognition. 2018. C. 3684–3692.

- [281] Pyramid attention network for semantic segmentation / Hanchao Li, Pengfei Xiong, Jie An [и др.] // arXiv preprint arXiv:1805.10180. 2018.
- [282] Real-time semantic segmentation benchmarks. PapersWithCode.com. URL: <https://paperswithcode.com/task/real-time-semantic-segmentation/latest>.
- [283] Gao Roland. Rethink dilated convolution for real-time semantic segmentation // arXiv preprint arXiv:2111.09957. 2021.
- [284] PP-LiteSeg: A Superior Real-Time Semantic Segmentation Model / Juncai Peng, Yi Liu, Shiyu Tang [и др.] // arXiv preprint arXiv:2204.02681. 2022.
- [285] Semantic flow for fast and accurate scene parsing / Xiangtai Li, Ansheng You, Zhen Zhu [и др.] // European Conference on Computer Vision / Springer. 2020. C. 775–793.
- [286] Zhou Xingyi, Zhuo Jiacheng, Krahenbuhl Philipp. Bottom-up object detection by grouping extreme and center points // Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019. C. 850–859.
- [287] A review of non-maximum suppression algorithms for deep learning target detection / Meiling Gong, Dong Wang, Xiaoxia Zhao [и др.] // Seventh Symposium on Novel Photoelectronic Detection Technology and Applications / SPIE. T. 11763. 2021. C. 821–828.
- [288] Overfeat: Integrated recognition, localization and detection using convolutional networks / Pierre Sermanet, David Eigen, Xiang Zhang [и др.] // arXiv preprint arXiv:1312.6229. 2013.
- [289] Rich feature hierarchies for accurate object detection and semantic segmentation / Ross Girshick, Jeff Donahue, Trevor Darrell [и др.] // Proceedings of the IEEE conference on computer vision and pattern recognition. 2014. C. 580–587.
- [290] Selective search for object recognition / Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers [и др.] // International journal of computer vision. 2013. T. 104, № 2. C. 154–171.
- [291] Object detection in 20 years: A survey / Zhengxia Zou, Zhenwei Shi, Yuhong Guo [и др.] // arXiv preprint arXiv:1905.05055. 2019.
- [292] Girshick Ross. Fast r-cnn // Proceedings of the IEEE international conference on computer vision. 2015. C. 1440–1448.
- [293] Faster r-cnn: Towards real-time object detection with region proposal networks / Shaoqing Ren, Kaiming He, Ross Girshick [и др.] // Advances in neural information processing systems. 2015. T. 28. C. 91–99.
- [294] R-fcn: Object detection via region-based fully convolutional networks / Jifeng Dai, Yi Li, Kaiming He [и др.] // Advances in neural information processing systems. 2016. T. 29.

- [295] Light-head r-cnn: In defense of two-stage object detector / Zeming Li, Chao Peng, Gang Yu [и др.] // arXiv preprint arXiv:1711.07264. 2017.
- [296] PanNet: A deep network architecture for pan-sharpening / Junfeng Yang, Xueyang Fu, Yuwen Hu [и др.] // Proceedings of the IEEE international conference on computer vision. 2017. C. 5449–5457.
- [297] Ghiasi Golnaz, Lin Tsung-Yi, Le Quoc V. Nas-fpn: Learning scalable feature pyramid architecture for object detection // Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019. C. 7036–7045.
- [298] Tan Mingxing, Pang Ruoming, Le Quoc V. Efficientdet: Scalable and efficient object detection // Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020. C. 10781–10790.
- [299] Focal loss for dense object detection / Tsung-Yi Lin, Priya Goyal, Ross Girshick [и др.] // Proceedings of the IEEE international conference on computer vision. 2017. C. 2980–2988.
- [300] You only look once: Unified, real-time object detection / Joseph Redmon, Santosh Divvala, Ross Girshick [и др.] // Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. C. 779–788.
- [301] Hui Jonathan. Real-time Object Detection with YOLO, YOLOv2 and now YOLOv3. 2018. URL: <https://jonathan-hui.medium.com/real-time-object-detection-with-yolo-yolov2-28b1b93e2088>.
- [302] Ssd: Single shot multibox detector / Wei Liu, Dragomir Anguelov, Dumitru Erhan [и др.] // European conference on computer vision / Springer. 2016. C. 21–37.
- [303] Real-Time Object Detection on COCO. URL: <https://paperswithcode.com/sota/real-time-object-detection-on-coco>.
- [304] Object Detection on COCO test-dev. URL: <https://paperswithcode.com/sota/object-detection-on-coco>.
- [305] Mask SSD: An effective single-stage approach to object instance segmentation / Hui Zhang, Yonglin Tian, Kunfeng Wang [и др.] // IEEE Transactions on Image Processing. 2019. T. 29. C. 2078–2093.
- [306] Redmon Joseph, Farhadi Ali. YOLO9000: better, faster, stronger // Proceedings of the IEEE conference on computer vision and pattern recognition. 2017. C. 7263–7271.
- [307] Redmon Joseph, Farhadi Ali. Yolov3: An incremental improvement // arXiv preprint arXiv:1804.02767. 2018.
- [308] Bochkovskiy Alexey, Wang Chien-Yao, Liao Hong-Yuan Mark. Yolov4: Optimal speed and accuracy of object detection // arXiv preprint arXiv:2004.10934. 2020.

- [309] Jocher Glenn, Chaurasia Ayush, Stoken Alex [и др.]. ultralytics/yolov5: v6.2 - YOLOv5 Classification Models, Apple M1, Reproducibility, ClearML and Deci.ai integrations. 2022. . URL: <https://doi.org/10.5281/zenodo.7002879>.
- [310] Wang Chien-Yao, Bochkovskiy Alexey, Liao Hong-Yuan Mark. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors // arXiv preprint arXiv:2207.02696. 2022.
- [311] Patrick Langechuan Liu. Single Stage Instance Segmentation — A Review. 2020. URL: <https://towardsdatascience.com/single-stage-instance-segmentation-a-review-1eeb66e0cc49>.
- [312] Polarmask: Single shot instance segmentation with polar representation / Enze Xie, Peize Sun, Xiaoge Song [и др.] // Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020. C. 12193–12202.
- [313] Instance Segmentation on COCO test-dev. URL: <https://paperswithcode.com/sota/instance-segmentation-on-coco>.
- [314] Microsoft coco: Common objects in context / Tsung-Yi Lin, Michael Maire, Serge Belongie [и др.] // European conference on computer vision / Springer. 2014. C. 740–755.
- [315] Cai Zhaowei, Vasconcelos Nuno. Cascade r-cnn: Delving into high quality object detection // Proceedings of the IEEE conference on computer vision and pattern recognition. 2018. C. 6154–6162.
- [316] Hybrid task cascade for instance segmentation / Kai Chen, Jiangmiao Pang, Jiaqi Wang [и др.] // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019. C. 4974–4983.
- [317] Path aggregation network for instance segmentation / Shu Liu, Lu Qi, Haifang Qin [и др.] // Proceedings of the IEEE conference on computer vision and pattern recognition. 2018. C. 8759–8768.
- [318] Yolact: Real-time instance segmentation / Daniel Bolya, Chong Zhou, Fanyi Xiao [и др.] // Proceedings of the IEEE/CVF international conference on computer vision. 2019. C. 9157–9166.
- [319] Yolact++: Better real-time instance segmentation / Daniel Bolya, Chong Zhou, Fanyi Xiao [и др.] // IEEE transactions on pattern analysis and machine intelligence. 2020.
- [320] Solov2: Dynamic and fast instance segmentation / Xinlong Wang, Rufeng Zhang, Tao Kong [и др.] // Advances in Neural information processing systems. 2020. T. 33. C. 17721–17732.
- [321] Sparse Instance Activation for Real-Time Instance Segmentation / Tianheng Cheng, Xinggang Wang, Shaoyu Chen [и др.] // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022. C. 4433–4442.

- [322] Real-time instance segmentation benchmarks. PapersWithCode.com. URL: <https://paperswithcode.com/sota/real-time-instance-segmentation-on-mscoco>.
- [323] Solo: Segmenting objects by locations / Xinlong Wang, Tao Kong, Chunhua Shen [и др.] // European Conference on Computer Vision / Springer. 2020. C. 649–665.
- [324] WikiChip . Ryzen Threadripper - AMD. 2022. URL: https://en.wikichip.org/wiki/amd/ryzen_threadripper.
- [325] WikiChip . Xeon Phi 7295 - Intel. 2021. URL: https://en.wikichip.org/wiki/intel/xeon_phi/7295.
- [326] Grigory Sapunov Grigory Sapunov . Hardware for Deep Learning. Part 2: CPU. 2018. URL: <https://blog.inten.to/cpu-hardware-for-deep-learning-b91f53cb18af>.
- [327] Intel . Open Vino. URL: <https://software.intel.com/content/www/us/en/develop/tools/openvino-toolkit.html>.
- [328] Georgia Tech and Facebook Artificial Intelligence Research. NNpack acceleration package for neural networks on multi-core CPUs. 2022. URL: <https://github.com/Maratyszcza/NNPACK>.
- [329] Intel. oneDNN Intel math kernel library for deep neural networks (Intel MKL-DNN) and deep neural network library (DNNL). 2022. URL: <https://github.com/oneapi-src/oneDNN>.
- [330] Hadjis Stefan, Abuzaid Firas, Zhang Ce [и др.]. Caffe con Troll: Shallow Ideas to Speed Up Deep Learning. 2015. URL: <https://arxiv.org/abs/1504.04343>.
- [331] Intel. Intel DLboost project. 2022. URL: <https://www.intel.com/content/www/us/en/artificial-intelligence/deep-learning-boost.html>.
- [332] Intel. Boosting deep learning training and inference performance on Intel Xeon and Intel Xeon Phi processors. 2022. URL: <https://software.intel.com/content/www/us/en/develop/articles/boosting-deep-learning-training-inference-performance-on-xeon-and-xeon-phi.html>.
- [333] An Updated Survey of Efficient Hardware Architectures for Accelerating Deep Convolutional Neural Networks / Maurizio Capra, Beatrice Bussolino, Alberto Marchisio [и др.] // Future Internet. 2020. T. 12, № 7. URL: <https://www.mdpi.com/1999-5903/12/7/113>.
- [334] Dumas II Joseph D. Computer architecture: Fundamentals and principles of computer design. CRC Press, 2018.
- [335] Grigory Sapunov Grigory Sapunov . Hardware for Deep Learning. Part 3: GPU. 2018. URL: <https://blog.inten.to/hardware-for-deep-learning-part-3-gpu-8906c1644664>.
- [336] NVIDIA Corporation. Artificial Neural Network. 2022. URL: <https://developer.nvidia.com/discover/artificial-neural-network>.

- [337] Vanholder Han. Efficient inference with tensorrt // GPU Technology Conference. T. 1. 2016. c. 2.
- [338] Evaluating the Performance of NVIDIA's A100 Ampere GPU for Sparse and Batched Computations / Hartwig Anzt, Yuhsiang M. Tsai, Ahmad Abdelfattah [и др.] // 2020 IEEE/ACM Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS). 2020. C. 26–38.
- [339] NVIDIA Corporation. Mixed-Precision Programming with CUDA 8. 2022. URL: <https://developer.nvidia.com/blog/mixed-precision-programming-cuda-8/>.
- [340] Tian Rong, Zhao Zijing, Liu Weijie [и др.]. SAMP: A Toolkit for Model Inference with Self-Adaptive Mixed-Precision. 2022. URL: <https://arxiv.org/abs/2209.09130>.
- [341] Linux Foundation. Automatic Mixed Precision package - torch.amp. 2022. URL: <https://pytorch.org/docs/stable/amp.html>.
- [342] Honka Tapio. AUTOMATIC MIXED PRECISION QUANTIZATION OF NEURAL NETWORKS USING ITERATIVE CORRELATION COEFFICIENT ADAPTATION. 2021.
- [343] Pruning and quantization for deep neural network acceleration: A survey / Tailin Liang, John Glossner, Lei Wang [и др.] // Neurocomputing. 2021. Т. 461. C. 370–403. URL: <https://www.sciencedirect.com/science/article/pii/S0925231221010894>.
- [344] Wimmer Paul, Mehnert Jens, Condurache Alexandru Paul. Dimensionality Reduced Training by Pruning and Freezing Parts of a Deep Neural Network, a Survey. 2022. URL: <https://arxiv.org/abs/2205.08099>.
- [345] Can FPGAs Beat GPUs in Accelerating Next-Generation Deep Neural Networks? / Eriko Nurvitadhi, Ganesh Venkatesh, Jaewoong Sim [и др.] // Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. FPGA '17. New York, NY, USA: Association for Computing Machinery, 2017. c. 5–14. URL: <https://doi.org/10.1145/3020078.3021740>.
- [346] Sun Wei, Li Ang, Geng Tong [и др.]. Dissecting Tensor Cores via Microbenchmarks: Latency, Throughput and Numerical Behaviors. 2022. URL: <https://arxiv.org/abs/2206.02874>.
- [347] Tsai Yuhsiang Mike, Cojean Terry, Anzt Hartwig. Evaluating the Performance of NVIDIA's A100 Ampere GPU for Sparse Linear Algebra Computations. 2020. URL: <https://arxiv.org/abs/2008.08478>.
- [348] Time-Based Roofline for Deep Learning Performance Analysis / Yunsong Wang, Charlene Yang, Steven Farrell [и др.] // 2020 IEEE/ACM Fourth Workshop on Deep Learning on Supercomputers (DLS). 2020. C. 10–19.
- [349] A GPU-outperforming FPGA accelerator architecture for binary convolutional neural networks / Yixing Li, Zichuan Liu, Kai Xu [и др.] // ACM Journal on Emerging Technologies in Computing Systems (JETC). 2018. Т. 14, № 2. C. 1–16.

- [350] Accelerating Neural Network Inference on FPGA-Based Platforms—A Survey / Ran Wu, Xinmin Guo, Jian Du [и др.] // Electronics. 2021. Т. 10, № 9. URL: <https://www.mdpi.com/2079-9292/10/9/1025>.
- [351] Optimizing Memory Efficiency for Deep Convolutional Neural Networks on GPUs / Chao Li, Yi Yang, Min Feng [и др.] // SC '16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. 2016. С. 633–644.
- [352] Habib Gousia, Qureshi Shaima. Optimization and acceleration of convolutional neural networks: A survey // Journal of King Saud University - Computer and Information Sciences. 2022. Т. 34, № 7. С. 4244–4268. URL: <https://www.sciencedirect.com/science/article/pii/S1319157820304845>.
- [353] Mittal Sparsh. A Survey on optimized implementation of deep learning models on the NVIDIA Jetson platform // Journal of Systems Architecture. 2019. Т. 97. С. 428–442. URL: <https://www.sciencedirect.com/science/article/pii/S1383762118306404>.
- [354] Hu Yunxiang, Liu Yuhao, Liu Zhuovuan. A Survey on Convolutional Neural Network Accelerators: GPU, FPGA and ASIC // 2022 14th International Conference on Computer Research and Development (ICCRD). 2022. С. 100–107.
- [355] Kim Joo-Young. Chapter Five - FPGA based neural network accelerators // Hardware Accelerator Systems for Artificial Intelligence and Machine Learning / под ред. Shiho Kim, Ganesh Chandra Deka. Elsevier, 2021. Т. 122 из Advances in Computers. С. 135–165. URL: <https://www.sciencedirect.com/science/article/pii/S0065245820300899>.
- [356] Mittal Sparsh, Vibhu. A survey of accelerator architectures for 3D convolution neural networks // Journal of Systems Architecture. 2021. Т. 115. с. 102041. URL: <https://www.sciencedirect.com/science/article/pii/S1383762121000400>.
- [357] Omondi Amos R, Rajapakse Jagath Chandana. FPGA implementations of neural networks. Springer, 2006. Т. 365.
- [358] Deep learning for computer architects / Brandon Reagen, Robert Adolf, Paul Whatmough [и др.] // Synthesis Lectures on Computer Architecture. 2017. Т. 12, № 4. С. 1–123.
- [359] Wang Teng, Wang Chao, Zhou Xuehai [и др.]. A Survey of FPGA Based Deep Learning Accelerators: Challenges and Opportunities. 2019. URL: <https://arxiv.org/abs/1901.04988>.
- [360] Gemmini: Enabling Systematic Deep-Learning Architecture Evaluation via Full-Stack Integration / Hasan Genc, Seah Kim, Alon Amid [и др.] // 2021 58th ACM/IEEE Design Automation Conference (DAC). 2021. С. 769–774.
- [361] Designing efficient accelerator of depthwise separable convolutional neural network on FPGA / Wei Ding, Zeyu Huang, Zunkai Huang [и др.] // Journal of Systems Architecture. 2019. Т. 97. С. 278–286. URL: <https://www.sciencedirect.com/science/article/pii/S1383762118304612>.

- [362] Hu Yu Hen, Kung Sun-Yuan. Systolic Arrays // Handbook of Signal Processing Systems / под ред. Shuvra S. Bhattacharyya, Ed F. Deprettere, Rainer Leupers [и др.]. Cham: Springer International Publishing, 2019. С. 939–977. URL: https://doi.org/10.1007/978-3-319-91734-4_26.
- [363] Liu Zhi-Gang, Whatmough Paul N., Mattina Matthew. Systolic Tensor Array: An Efficient Structured-Sparse GEMM Accelerator for Mobile CNN Inference // IEEE Computer Architecture Letters. 2020. Т. 19, № 1. С. 34–37.
- [364] NoC-Based DNN Accelerator: A Future Design Paradigm / Kun-Chih (Jimmy) Chen, Masoumeh Ebrahimi, Ting-Yi Wang [и др.] // Proceedings of the 13th IEEE/ACM International Symposium on Networks-on-Chip. NOCS '19. New York, NY, USA: Association for Computing Machinery, 2019. 8 с. URL: <https://doi.org/10.1145/3313231.3352376>.