Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

М. М. Лукашевич

ЦИФРОВАЯ ОБРАБОТКА ИЗОБРАЖЕНИЙ И РАСПОЗНАВАНИЕ ОБРАЗОВ

Рекомендовано УМО по образованию в области информатики и радиоэлектроники в качестве пособия для специальности 1-40 02 01 «Вычислительные машины, системы и сети»

Минск БГУИР 2023

УДК 004.932(076)
ББК 32.813.5я73
Л84

Р е ц е н з е н т ы:

кафедра информационных систем управления
Белорусского государственного университета
(протокол №8 от 16.01.2020);

старший научный сотрудник государственного научного учреждения
«Объединенный институт проблем информатики
Национальной академии наук Беларуси»
кандидат технических наук, доцент Ю. И. Голуб

Лукашевич, М. М.

Л84 Цифровая обработка изображений и распознавание образов : пособие / М. М. Лукашевич. – Минск : БГУИР, 2023. – 72 с. : ил.

ISBN 978-985-543-581-6.

Содержит теоретические сведения о принципах цифровой обработки изображений и распознавания образов на основе глубоких нейронных сетей. Приведены задания для выполнения практических работ.

УДК 004.932(076)
ББК 32.813.5я73

ISBN 978-985-543-581-6

Содержание

Введение

Технология глубокого обучения (Deep Learning) стала одним из наиболее востребованных IT-трендов. На этой технологии основано большое количество инноваций. В современном мире глубокое обучение применяется в различных сферах жизни и бизнеса. На данный момент глубокое обучение используется в области аэрокосмической и оборонной промышленности, автомобилестроении и здравоохранении.

Исследовательская и консалтинговая компания Grand View Research (GVR) в 2016 году оценила глобальный рынок глубокого обучения в 272 млн дол. США [1]. Согласно отчету Grand View Research Inc предполагается, что к 2025 году объем мирового рынка глубокого обучения достигнет 10,2 млрд дол. США. Значительные усовершенствования в алгоритмах машинного обучения способствуют росту отрасли. Улучшения в области быстрого хранения информации, высокой вычислительной мощности и распараллеливания способствовали быстрому внедрению технологии глубокого обучения в отрасли автомобилестроения и здравоохранения.

Организации используют нейронные сети глубокого обучения для извлечения ценной информации из огромных объемов данных для предоставления инновационных продуктов и улучшения качества обслуживания клиентов. Кроме того, растущая потребность в человеческом и машинном взаимодействии предлагает новые задачи, которые наиболее эффективно можно решить с помощью глубокого обучения. Также Grand View Research прогнозирует развитие приложений, связанных с распознаванием изображений и голоса, интеллектуальным анализом данных и медицинской диагностикой [2].

Технология глубокого обучения широко используется для распознавания шаблонов в неструктурированных данных, включая звук, текст, изображения и видео. Самой популярной задачей для нейронных сетей глубокого обучения является детекция объектов на изображениях. В 2016 году сегмент распознавания образов на изображениях доминировал в отрасли.

Одним из наиболее широко используемых приложений, основанных на глубоком обучении, является распознавания лиц от Facebook. Приложение для мгновенного перевода Google Translate app также применяет технологию Deep Learning для визуального перевода, а приложение LipNet, разработанное с применением технологий нейронных сетей, может распознать по губам речь человека.

Многие компании, работающие в отрасли автомобилестроения, используют алгоритмы глубокого обучения в своей технологии для проектирования беспилотных автомобилей, чтобы распознавать дорожные знаки и препятствия

на дороге [3]. Однако данная задача не решена до конца и требует дальнейшей работы. Так, например, при тестировании автопилота автомобилей Tesla были выявлены дефекты определения полос движения из-за некорректного распознавания разметки [4].

Многие задачи, в которых применяется глубокое обучение, основываются на анализе изображений. Анализ изображений включает в себя детектирование, классификацию объектов и сегментацию. На данный момент существует большое количество алгоритмов глубокого обучения, каждый из которых имеет свои преимущества для решения той или иной задачи. Правильный выбор архитектуры нейронной сети, алгоритмов детекции объектов изображения и набора обучающих данных позволяет проектировать системы, успешно решающие поставленные задачи. Однако проблемой является определение архитектуры, подходящей для конкретной задачи.

1 ГЛУБОКОЕ ОБУЧЕНИЕ И МАШИННОЕ ОБУЧЕНИЕ

Машинное обучение (Machine Learning) – обширный подраздел искусственного интеллекта, математическая дисциплина, использующая разделы математической статистики, численных методов оптимизации, теории вероятностей, дискретного анализа и извлекающая знания из данных [5].

Одной из базовых задач в машинном зрении является задача классификации изображения – определения категорий объектов, которые находятся на изображении. В зависимости от конкретной задачи на изображении может быть аннотирован как один объект, так и несколько.

Весь спектр задач, которые приходится решать при распознавании на изображениях, можно подразделить на две группы:

1) распознавание или классификация изображений;
2) поиск и распознавание объектов (специфических локальных областей) на изображениях.

Это разделение связано с особенностями реализации процесса распознавания. В первой группе задач распознавание или классификация производится для всего изображения целиком. То есть все изображение целиком в процессе распознавания относят к одному из нескольких классов. Таким образом, решением задачи распознавания в этой группе является реализация отображения «изображение – номер класса».

Описанное выше отображение представлено на рисунке 1.1 и реализуется в виде следующих двух отображений:

1) отображение «изображение – признаки»;
2) отображение «признаки – класс».

В задачах второй группы процесс распознавания оказывается включенным в более общую технологию обработки изображения, связанную с поиском распознаваемых геометрических объектов на всей области наблюдения. Объекты в данной ситуации представляют собой относительно небольшие локальные области, появление которых может произойти в любой точке изображения.

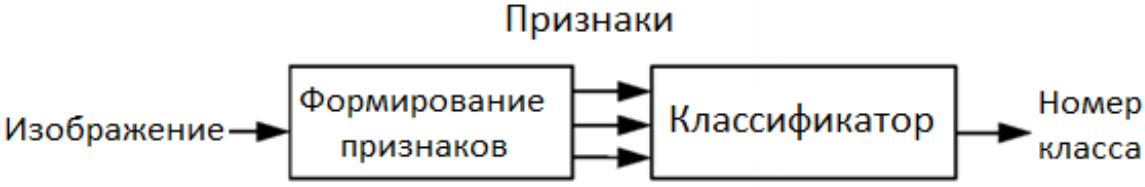


Рисунок 1.1 – Схема решения задачи распознавания образов

Причем информация о том, имеются ли объекты на изображении, каково их количество, ориентация, размеры и другие признаки, чаще всего отсутствует. Результатами решения задачи распознавания в этой ситуации являются не только класс найденного объекта, но также и его характеристики: положение, ориентация объекта в плоскости, размер.

Неопределенность в целом ряде характеристик объектов делает задачу их поиска и распознавания на изображении в математическом и вычислительном плане более сложной по сравнению с задачами первой группы. Это приводит к тому, что процесс ее решения не укладывается в приведенную схему, а производится в соответствии со схемой на рисунке 1.2 и включает в себя трудно формализуемую задачу выделения фрагментов («областей интереса»).

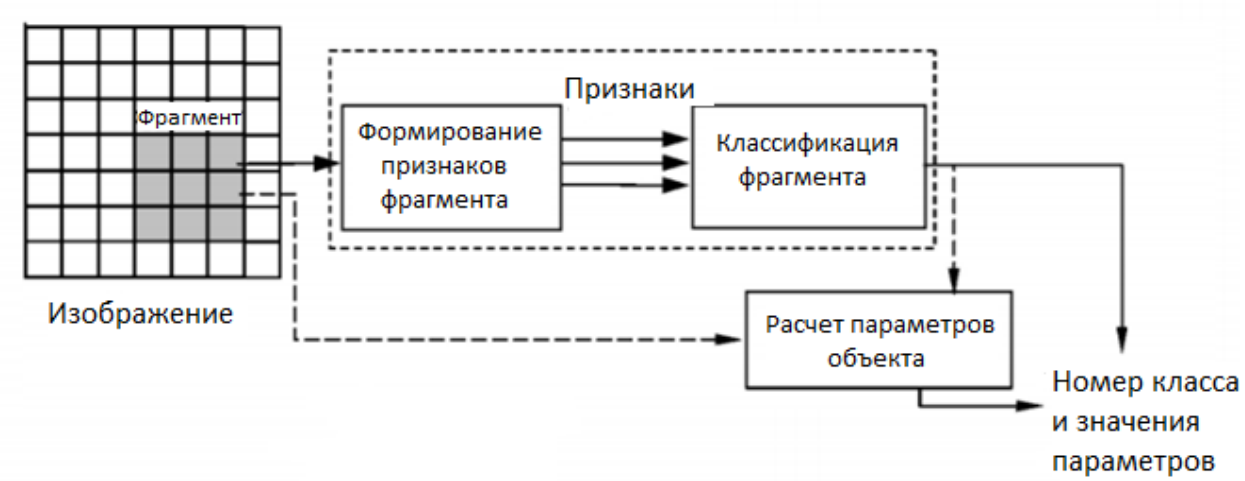


Рисунок 1.2 – Схема поиска и распознавания объектов на изображении

В соответствии с данной схемой анализу подвергается каждый фрагмент на изображении. По текущему фрагменту, выделенному окном обработки, производятся формирование признаков и классификация. В зависимости от результатов классификации происходит расчет дополнительных параметров объекта.

Можно выделить три основных подхода в детекции объектов:

1) эвристические методы;
2) метод сравнения с шаблоном;
3) методы с обучением по прецедентам.

Эвристические методы можно разделить на полную эвристическую модель и поиск характерных инвариантных признаков. Полная эвристическая модель заключается в том, что экспертом составляется набор правил, описывающих изображение объекта (строится модель), согласно которым производится обнаруже-

ние. А в поиске характерных инвариантных признаков эвристически описываются не изображения искомого объекта в целом, а его характерные признаки, инвариантные относительно возможных искажений (изменение освещения, поворот, масштабирование).

В методе сравнения с шаблоном составляется шаблон для изображения всего объекта или его характерных признаков. Также вводится функция проверки соответствия. Это более универсальный подход в смысле обобщения других видов объекта, однако он требует наличия весьма точного шаблона изображения объекта. Шаблон может быть сложной структурой, допускающей различные деформации и преобразования, таким образом способствуя инвариантности системы к пространственным искажениям изображения объекта и изменениям освещенности.

В методах с обучением по прецедентам модель автоматически строится на основе набора изображений объекта, составленных заранее из возможных входных данных системы. Это общий подход. Задача распознавания объектов на изображение сводится к задаче классификации.

Модель строится автоматически по заранее собранному набору прецедентов – изображений, о которых известно, являются ли они изображениями объекта или нет. Наблюдением в данном случае является некоторый «вектор признаков», полученный из исходного изображения некоторым преобразованием, отображающим изображения в пространство действительных векторов.

Гипотеза, подлежащая проверке, – принадлежность изображения к классу изображений искомого объекта. Таким образом, система распадается на два модуля: модуль преобразования изображения в вектор признаков и модуль классификации.

Задачей модуля преобразования является наиболее полное и информативное представление изображения в виде числового вектора. Задачей модуля классификации является проверка гипотезы принадлежности изображения классу изображений объекта на основании наблюдения, которым является вектор признаков.

Модуль преобразования и модуль классификации тесно связаны. Главная цель модуля преобразования – представить изображение в форме, наиболее удобной для модуля классификации. Основные требования, предъявляемые к модулю преобразования, – скорость, наиболее полное и информативное представление данных, масштабируемость (преобразование корректно работает с изображением разных размеров, размерность вектора признаков не меняется) [6].

В классическом машинном обучении важные экспертные знания вводятся вручную, однако затем система обучается, организуя вывод данных на основании

самостоятельно изученных признаков. Этот тип машинного обучения широко используется для решения простых задач распознавания объектов. При проектировании таких систем большая часть времени тратится на выбор верного обучающего набора данных. Когда знания экспертов удается формализовать, тогда для получения выходных данных используется обычный классификатор (рисунок 1.3).

Глубокое обучение – это часть более широкого семейства методов машинного обучения – обучения представлениям, где векторы признаков располагаются сразу на множестве уровней.

Под словом «глубокая» в данном случае понимают глубину графа вычислительной модели. На каждом уровне представлены абстрактные признаки, основанные на признаках предыдущего уровня. Таким образом, чем глубже мы продвигаемся, тем выше уровень абстракции. В процессе обучения и анализа такая сеть комбинирует несколько типов входных данных для составления конечного образа (например, образы глаза, рта, носа) (см. рисунок 1.3).

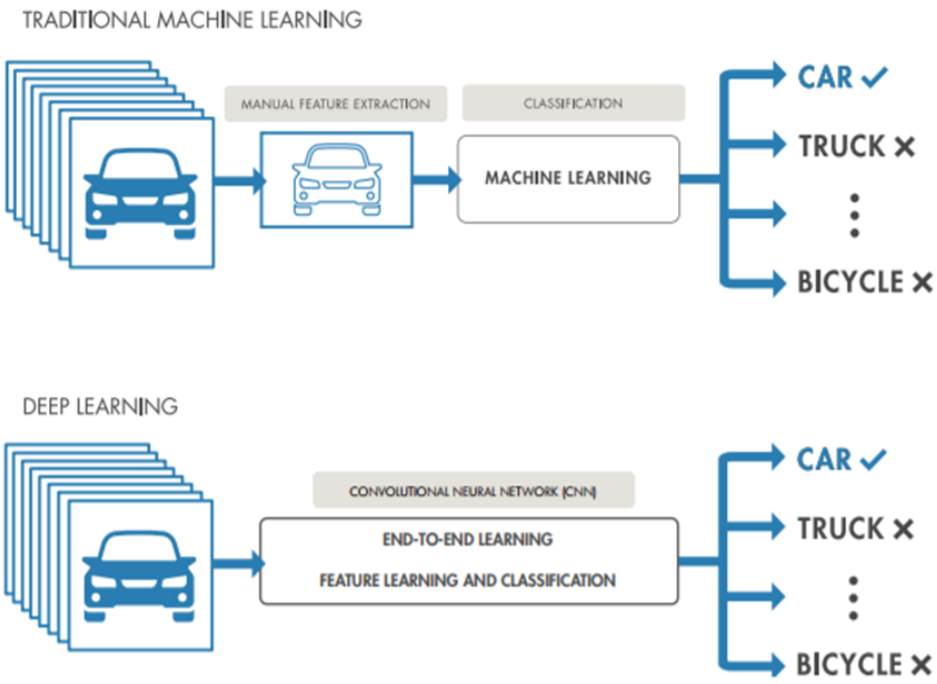


Рисунок 1.3 – Основные отличия машинного обучения и глубокого обучения [7]

В таблице 1.1 представлены достоинства и недостатки методов машинного обучения и методов глубокого обучения.

Таблица 1.1 – Сравнение машинного обучения и глубокого обучения

Машинное обучение		**Глубокое обучение**	
достоинства	**недостатки**	**достоинства**	**недостатки**
Хорошие результаты на небольших наборах данных	Необходимо пробовать различные алгоритмы вычисления признаков и классификатор для достижения лучшего результата	Извлекает признаки и классифицирует автоматически	Требует очень большого набора данных
Высокая скорость обучения модели	Точность ограничена	Практически неограниченная точность	Вычислительная сложность

На рисунке 1.4 изображена связь различных частей системы искусственного интеллекта (ИИ) между собой в рамках разных подходов к ИИ.

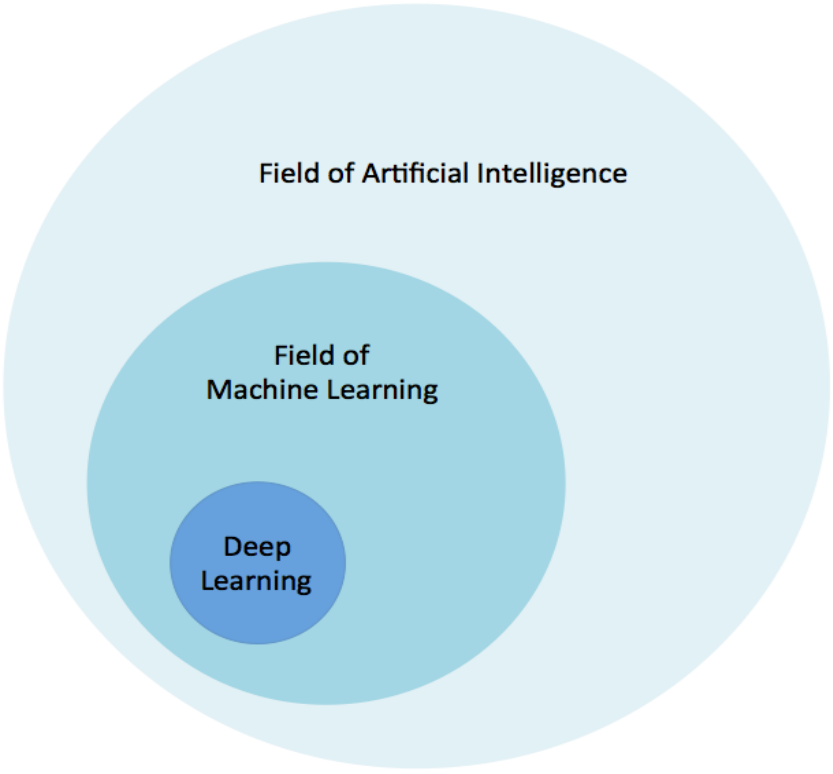


Рисунок 1.4 – Глубокое обучение как раздел машинного обучения [7]

На рисунке 1.5 представлены различные подходы к ИИ, серым цветом на нем показаны компоненты, способные обучаться на данных.

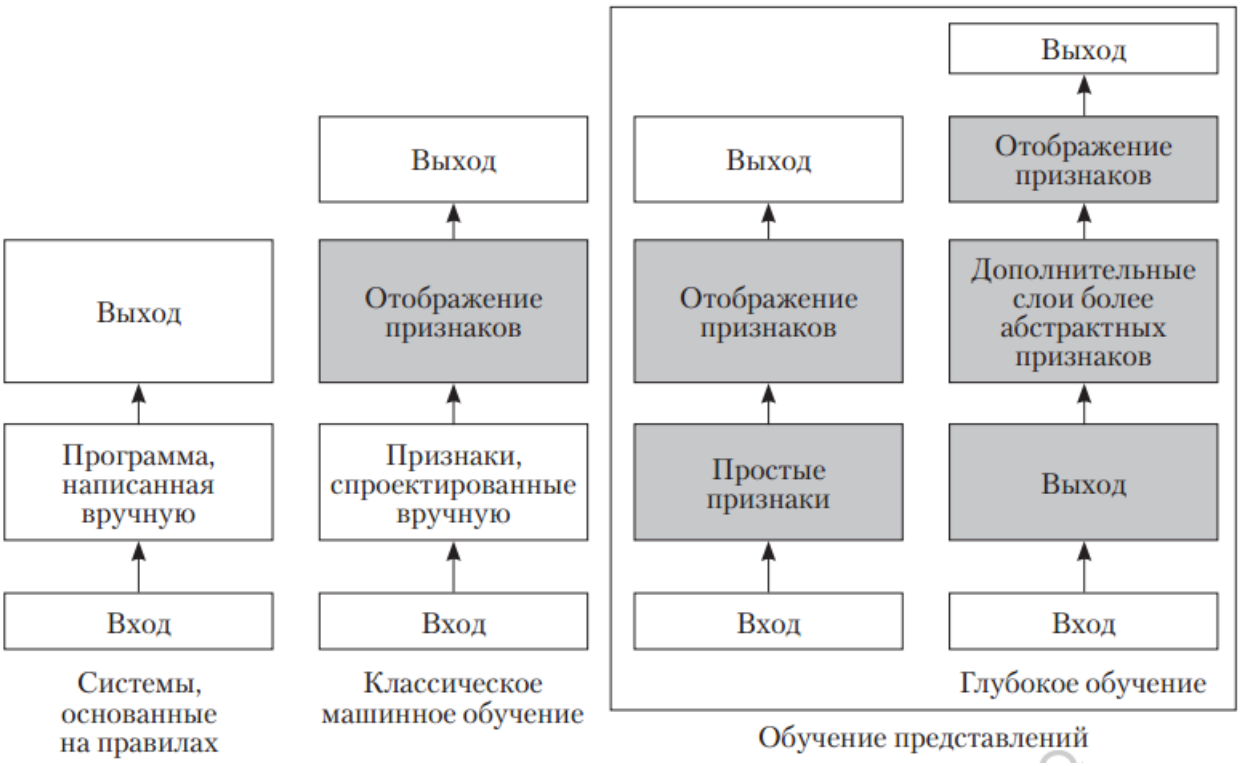


Рисунок 1.5 – Различные подходы к ИИ

В настоящее время теория и практика машинного обучения переживают настоящую «глубокую революцию», вызванную успешным применением методов Deep Learning, представляющих собой третье поколение нейронных сетей. В отличие от классических (второго поколения) нейронных сетей 80–90-х годов прошлого века, новые парадигмы обучения позволили избавиться от ряда проблем, которые сдерживали распространение и успешное применение традиционных нейронных сетей.

Наиболее успешные современные промышленные методы компьютерного зрения и распознавания речи построены на использовании глубоких сетей, а гиганты IT-индустрии, такие как Apple, Google, Facebook, скупают коллективы исследователей, занимающихся глубокими нейросетями.

Глубокие архитектуры, основанные на искусственных нейронных сетях, берут свое начало с неокогнитрона, разработанного Кунихико Фукусимой в 1980 году. В 1989 году Яну Лекуну удалось использовать алгоритм обратного распространения ошибки для обучения глубоких нейросетей для решения задачи распознавания рукописных ZIP-кодов [8].

Несмотря на успешный опыт, для обучения модели потребовалось три дня, что существенно ограничивало применение этого метода. Низкая скорость обу-

чения связана со многими факторами, включая проблему исчезающего градиента, которую в 1991 году анализировали Юрген Шмидхубер и Сепп Хохрайтер. Из-за этих проблем нейронные сети в 1990-х годах уступили место методу опорных векторов.

Термин «глубокое обучение» приобрел популярность после публикации Джеффри Хинтона и Руслана Салахутдинова в середине 2000-х годов, в которой они показали, что можно эффективно предобучать многослойную нейронную сеть, если обучать каждый слой отдельно при помощи ограниченной машины Больцмана, а затем дообучать при помощи метода обратного распространения ошибки. Также на рост популярности этой группы алгоритмов оказало влияние развитие распределенных вычислений и повышение производительности видеокарт.

2 ОСНОВНЫЕ АРХИТЕКТУРЫ ГЛУБОКИХ НЕЙРОННЫХ СЕТЕЙ

Число архитектур и алгоритмов, которые используются для Deep Learning, очень разнообразно и постоянно растет, т. к. эта область сейчас активно развивается.

В этом разделе будут рассмотрены три основные и наиболее популярные архитектуры, которые зачастую служат основой для других глубоких архитектур обучения, а также архитектуры для решения задач сегментации изображений и детекции объектов на изображениях [9–11]:

- классические нейросети прямого распространения (Feedforward Neural Networks, FNN);
- сверточные нейросети (Convolutional Neural Networks, CNN);
- рекуррентные нейросети (Recurrent Neural Networks, RNN).

Данные архитектуры широко применяются при решении различных задач, но имеют свою специализацию, т. е. те области применения, где они зарекомендовали себя лучше других:

1 Feedforward Neural Network в основном используется для анализа данных, в которых можно выявить повторяющийся паттерн.

2 Convolutional Neural Network активно используется при распознавании объектов, анализе фото или видео, обработке естественного языка.

3 Recurrent Neural Network используется там, где важно делать выводы, основываясь не только на настоящих данных, но и помнить предыдущие. Например, при распознавании эмоций, обработке речи или анализе текста.

2.1 Классические нейросети прямого распространения (Fully Connected Feed-Forward Neural Network, FFNN)

Многослойный персептрон (Multilayer Perceptron, MLP) – нейронная сеть прямого распространения сигнала (без обратных связей), в которой входной сигнал преобразуется в выходной, проходя последовательно через несколько слоев (рисунок 2.1).

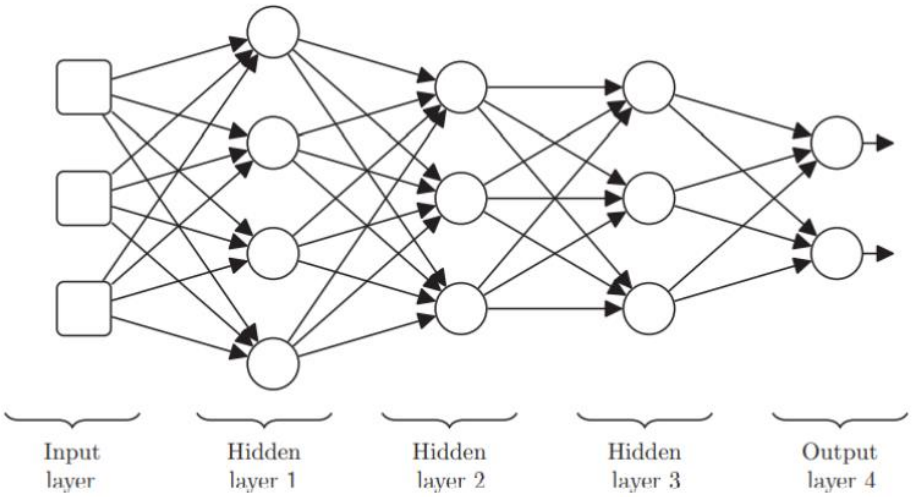


Рисунок 2.1 – Многослойный персептрон

Данная архитектура хорошо зарекомендовала себя для классификации, но есть определенные трудности в обучении:

- *много параметров.* Для сети, у которой на входе картинка 100×100, три скрытых слоя по 100 нейронов каждый и выход на 10 классов, число параметров будет примерно 1M (10 000 × 100 + 100 × 100 + 100 × 100 + 100 × 10). Чтобы обучить нейросеть с миллионом параметров, нужно очень много обучающих примеров, которые не всегда есть. Кроме того, сеть, у которой много параметров, имеет дополнительную склонность переобучаться;

- *затухающие градиенты.* При использовании метода обратного распространения ошибки ошибка с выходов отправляется на вход, распределяется по всем весам и отправляется дальше по сети. Далее градиент (производная ошибки) прогоняется через нейросеть обратно. Когда в нейросети много слоев, от этого градиента в самом конце может остаться очень маленькая часть – веса на входе будет практически невозможно изменить [30–32, 42].

Автокодировщик (Autoencoder, AE) – сеть прямого распространения с так называемым бутылочным горлышком в середине. В отличие от многослойного персептрона выходной слой автокодировщика должен содержать столько же нейронов, сколько и входной слой. Цель этой нейросети – взять какой-то вход, прогнать через себя и на выходе сгенерировать тот же самый вход, т. е. чтобы они совпадали. Такая сеть учится создавать компактное описание входных данных и может использоваться для уменьшения размерности и получения новых высокоуровневых признаков. Рассмотрим принцип работы автокодировщика на примере рисунка 2.2.

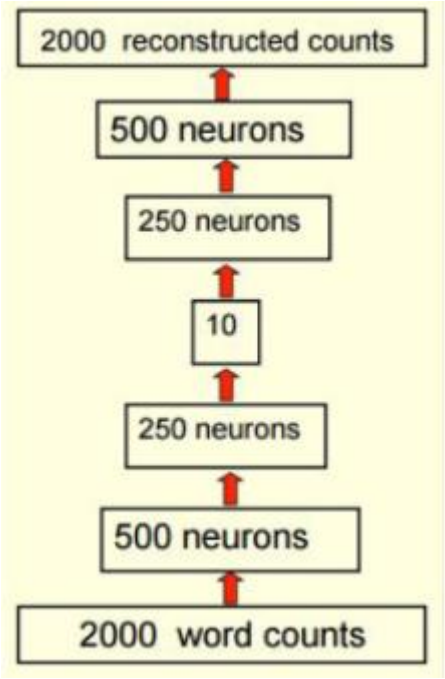


Рисунок 2.2 – Принцип работы автокодировщика

Если мы сможем обучить такую сеть, которая берет вход, прогоняет через себя и генерирует точно такой же выход, это значит, что этих 10 нейронов в середине достаточно для описания этого входа. То есть можно очень сильно уменьшить пространство, сократить объем данных, экономно закодировать любые входные данные в новых терминах 10 векторов [30–32, 42].

Ограниченная машина Больцмана (Restricted Boltzmann Machine, RBM). Несмотря на то что данная сеть неглубокая и не Feed-Forward, она часто связана с FNN-сетями, т. к. ее можно использовать для обучения глубоких сетей.

RBM состоит из двух слоев: скрытого и видимого. Нейроны одного слоя никак не связаны между собой, однако любой нейрон одного слоя имеет связь со всеми нейронами другого слоя. Особенность этой модели в том, что при данном состоянии нейронов одной группы состояния нейронов другой группы будут независимы друг от друга (рисунок 2.3).

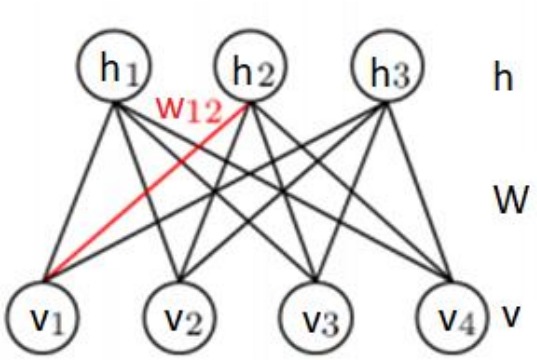


Рисунок 2.3 – Топология RBM

RBM – генеративная модель, которая учится генерировать данные с заданными характеристиками. Она стохастическая, что обозначает, что она каждый раз будет генерировать аналоги сигналов, но немного другие. Например, если такая модель обучена генерировать рукописные единицы, то каждый раз получим какое-то количество отличных единиц [30–32].

Глубокая сеть доверия (Deep belief network, DBN) – многослойная сеть, в котором каждая соседняя пара слоев работает как отдельная RBM. Потом эти RBM стыкуются, т. е. объединяются в одну нейросеть (рисунок 2.4) [42].

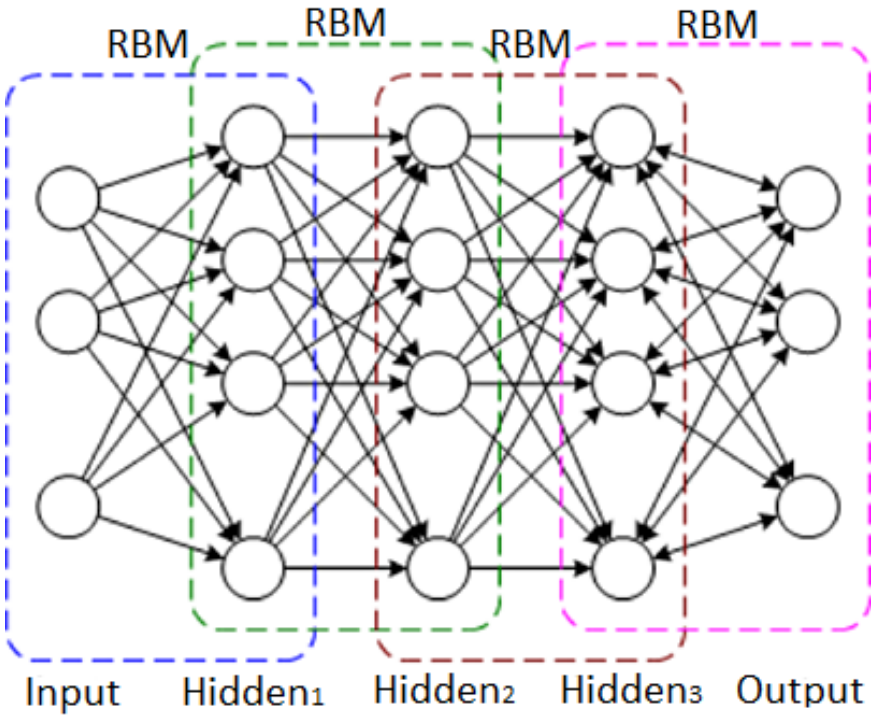


Рисунок 2.4 – Топология DBN

2.2 Сверточные нейросети (Convolutional Neural Networks, CNN)

Перед сверточными нейросетями стоит три основные задачи:

1) классификация, где нейросеть выдает класс объекта;

2) детекция, где нейросеть находит еще Bounding box (где объект находится на картинке);

3) сегментация, где происходит попиксельная классификация (рисунок 2.5).

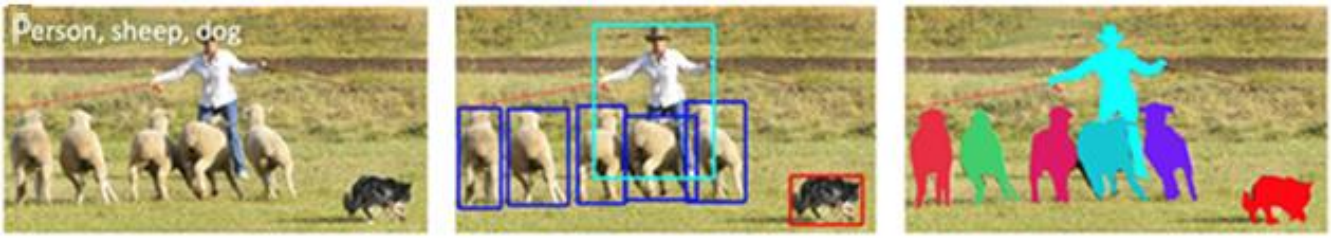


Рисунок 2.5 – Классические задачи для CNN [12]

Преимущество сетей CNN в сравнении с обычной полносвязной сетью FNN связано с уменьшением количества параметров. Соответственно, такую нейросеть проще обучить.

Архитектура сверточных нейронных сетей

В состав CNN входят:

- сверточные слои: каждая плоскость в сверточном слое – это один нейрон, который реализует операцию свертки (convolution) и является матричным фильтром небольшого размера;

- слои субдискретизации (Subsampling, spatian pooling) уменьшают размер изображения (например, в два раза);

- полносвязные слои (MLP) на выходе модели используются для классификации (рисунок 2.6).

В CNN слои свертки и субдискретизации состоят из нескольких «уровней» нейронов, называемых картами признаков (feature maps). Каждый нейрон такого слоя соединен с небольшим участком предыдущего слоя, называемым рецептивным полем. В случае изображения карта признаков является двумерным массивом нейронов, или просто матрицей.

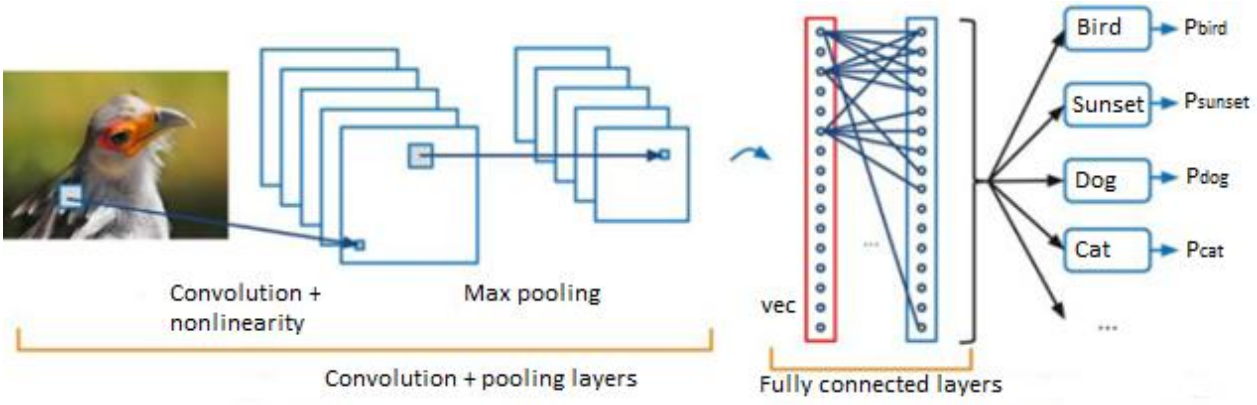


Рисунок 2.6 – Распознавание изображения в CNN [12]

Задача обучения сверточного слоя аналогична задаче в обычных нейросетях – найти веса, т. е. фактически найти ту самую матрицу свертки, которая полностью эквивалентна весам в нейронах. В слое свертки каждой карте признаков соответствует одно ядро свертки, также называемое ядром или фильтром. Каждый нейрон в качестве своего выходного значения осуществляет операцию свертки со своим рецептивным слоем.

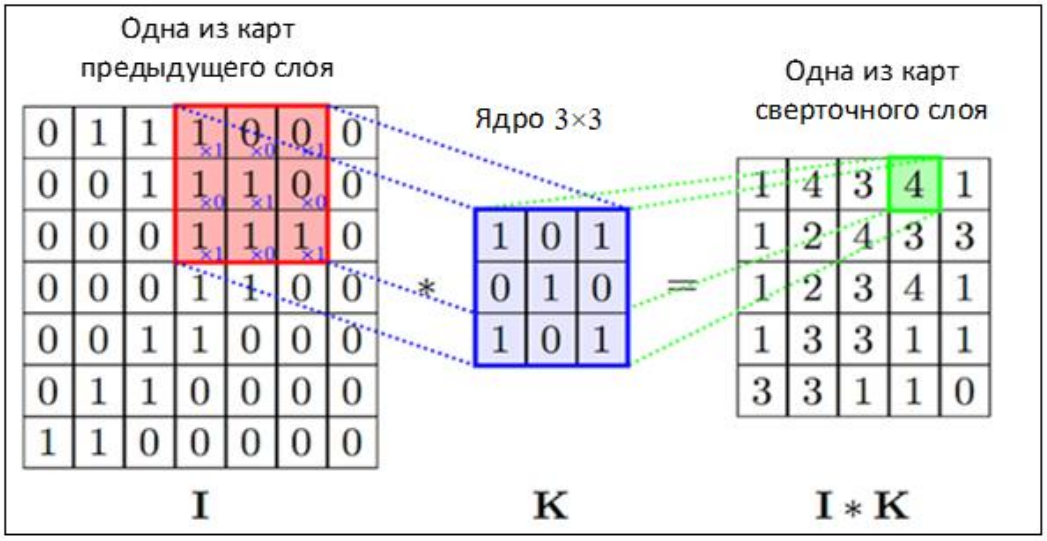


Рисунок 2.7 – Свертка

Примечание – Окном размера ядра K проходим с заданным шагом (обычно 1) все изображение I, на каждом шаге поэлементно умножаем содержимое окна на ядро K, результат суммируется и записывается в матрицу результата [44].

Так как ядро свертки для каждой карты признаков одно, то это позволяет нейронной сети научиться выделять признаки вне зависимости от их расположения во входном изображении и также приводит к значительному уменьшению

параметров. Далее построенные плоскости передаются на следующие входы (рисунок 2.8).

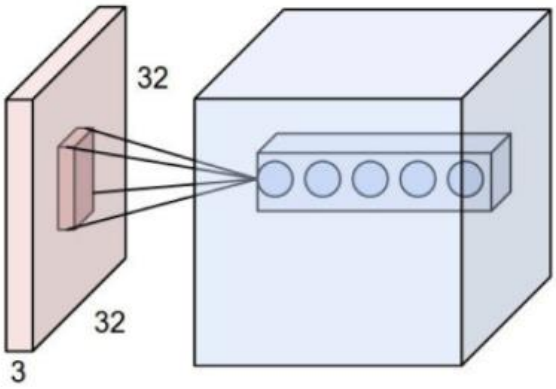


Рисунок 2.8 – Сверточный слой [13]

Слой субдискретизации осуществляет уплотнение карт признаков предыдущего слоя и не изменяет количества карт. Каждая карта признаков слоя соединена с соответствующей картой признаков предыдущего слоя, каждый нейрон выполняет «сжатие» своего рецептивного поля посредством какой-либо функции.

Наиболее популярными видами этого слоя являются Max Pooling (из рецептивного слоя выбирается максимальное значение) (рисунок 2.9), Average Pooling (выбирается среднее значение) и L2 Pooling (выбирается норма L2).

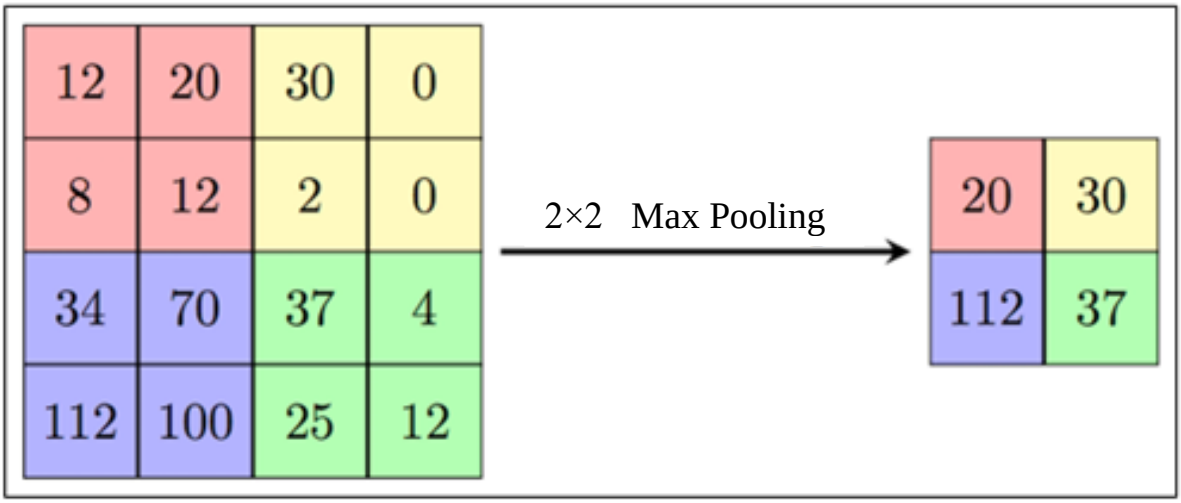


Рисунок 2.9 – Операция Max Pooling [25]

Визуализация сверточных слоев представлена на рисунке 2.10.

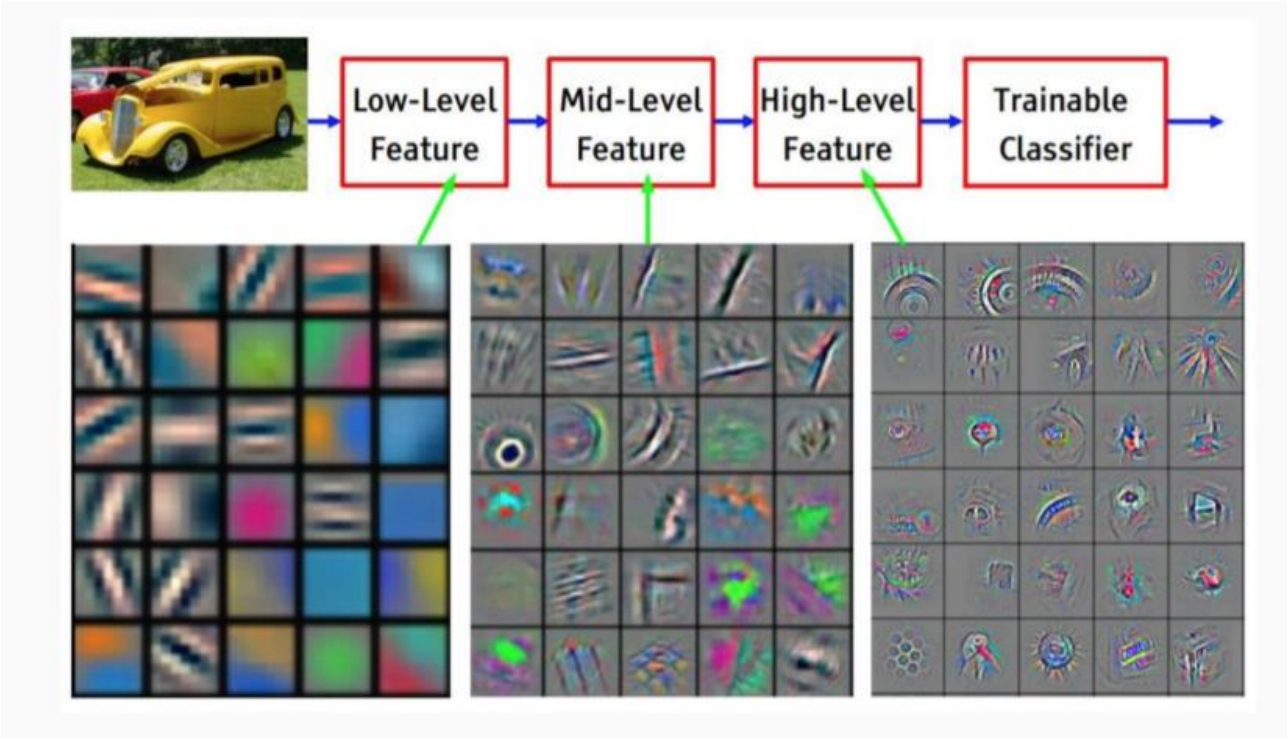


Рисунок 2.10 – Визуализация сверточных слоев, слайд LeCun

С помощью слоя субдискретизации достигается устойчивость к небольшим сдвигам входного изображения, а также уменьшается размерность предыдущих слоев (рисунок 2.11).

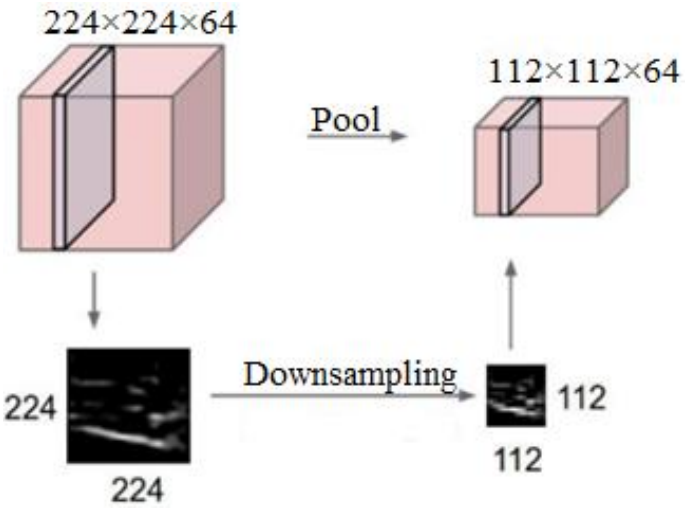


Рисунок 2.11 – Pooling-слой (Downsampling) [14]

Полносвязный слой – обычный скрытый слой многослойного перцептрона, соединенный со всеми нейронами предыдущего слоя.

Таким образом, сверточные слои учат иерархические признаки изображений (базовые детекторы, линии разного наклона, градиенты), а spatian pooling дает некоторую инвариантность к перемещениям. Сначала выделяются простые признаки, из них комбинируются сложные признаки, из них еще более сложные, еще более сложные и в конце можно скомбинировать какой-то очень сложный признак – конкретный человек, конкретная машина, слон (что-либо конкретное) (рисунок 2.12).

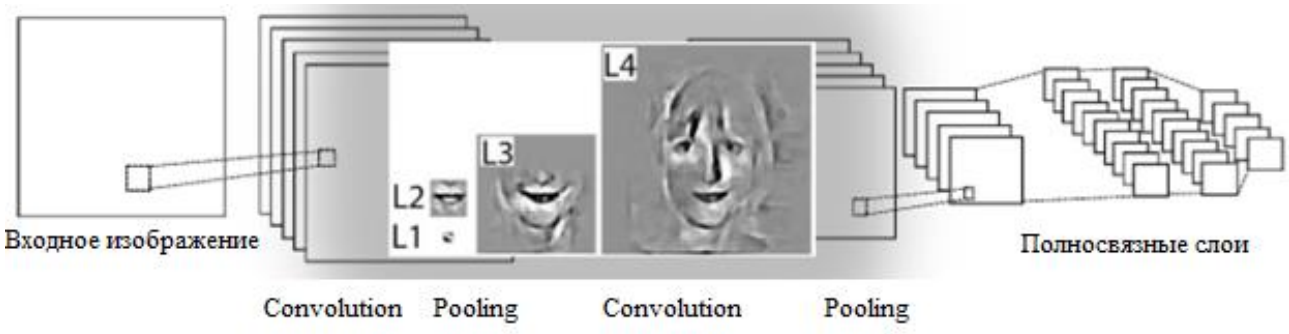


Рисунок 2.12 – Последовательность действий в CNN

Рассмотрим наиболее популярные архитектуры сверточных нейронных сетей: LeNet-5, AlexNet, ZF Net, VGG Net, GoogLeNet, ResNet.

LeNet-5

Основы современной архитектуры CNN были заложены в одной из первой широко известной сверточной сети – LeNet-5 Яна ЛеКуна (1998 год), архитектура которой представлена на рисунке 2.13. Архитектура LeNet применялась для считывания почтовых индексов, цифр и т. п. Данная архитектура является классической для CNN [28].

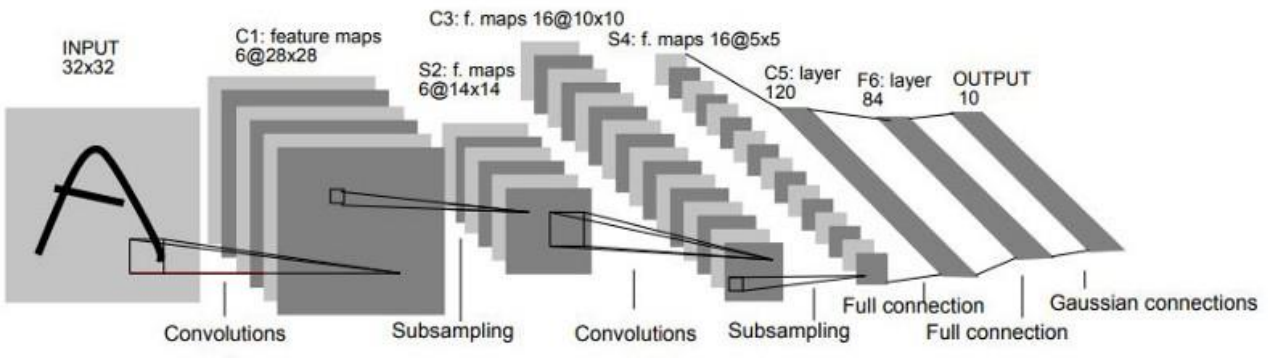


Рисунок 2.13 – Архитектура LeNet-5 [28]

AlexNet

В 2012 году на конкурсе ILSVRC [23] классификации изображений впервые победила нейронная сеть AlexNet, достигнув в top-5 ошибки 15,31 %. Для сравнения уточним, что метод, не использующий сверточные нейронные сети, получил ошибку 26,1 %. Архитектура AlexNet [23] была одной из первых глубоких сетей, которая значительно повысила точность классификации ImageNet по сравнению с традиционными методологиями. Данная архитектура состоит из пяти сверточных слоев, за которыми следуют три полностью связанных слоя (рисунок 2.14).

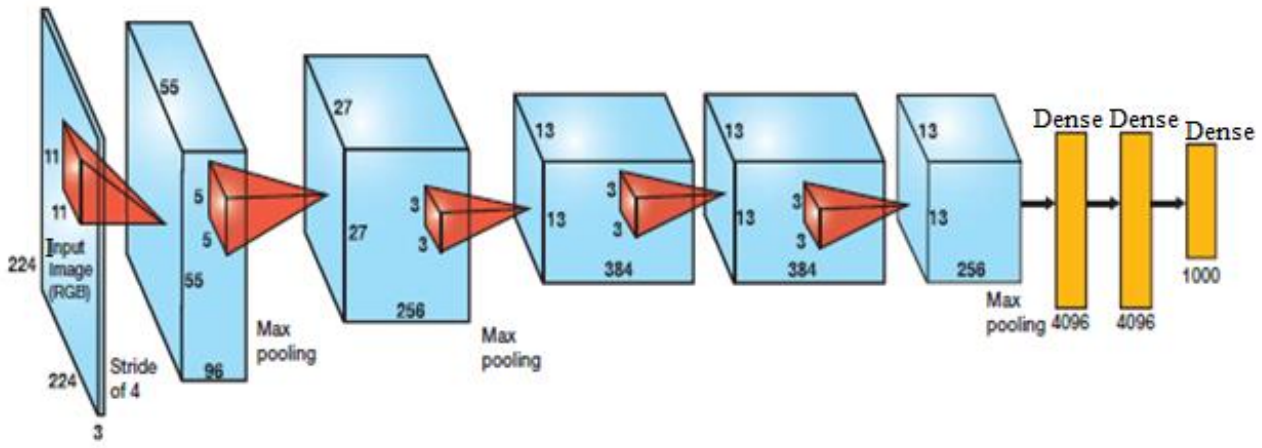


Рисунок 2.14 – Архитектура AlexNet [23]

В AlexNet из-за количества параметров сети обучение происходило на двух GPU, что позволило сократить время обучения в сравнении с обучением на CPU. Также оказалось, что использование функции активации ReLU [24] вместо более традиционных функций сигмоиды и гиперболического тангенса позволило снизить количество эпох (циклов) обучения в шесть раз.

Формула выпрямленной линейной функции активации ReLU и ее график представлены на рисунке 2.15.

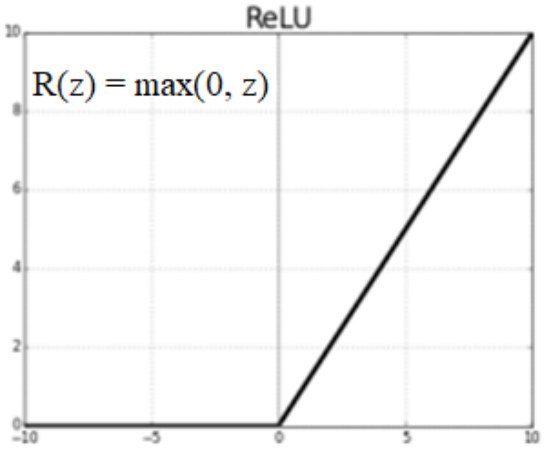


Рисунок 2.15 – Функция активации ReLU

Производная ReLU равна либо единице, либо нулю, и поэтому не может произойти разрастание или затухание градиентов. Более того, использование данной функции приводит к прореживанию весов, что приводит к удалению излишних связей и нейронов, не приводящих к увеличению ошибки классификации сетью. В развитие была предложена функция активации Leaky ReLU [27], которая не имеет такого недостатка, как затухание (рисунок 2.16).

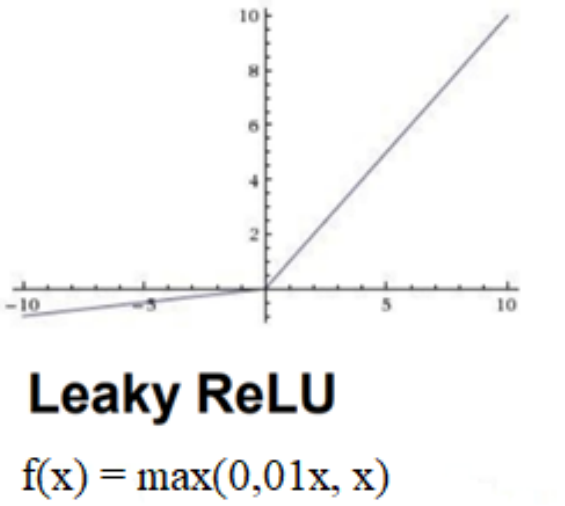


Рисунок 2.16 – Функция активации Leaky ReLU

Помимо прочего, в AlexNet была применена техника отсева (Dropout), которая заключалась в случайном отключении каждого нейрона на заданном слое с вероятностью p на каждой эпохе. После обучения сети на стадии распознавания

веса слоев, к которым был применен dropout, должны быть умножены на 1/p. Dropout выступает в роли регуляризатора, не позволяя сети переобучаться.

ZF Net

Победителем ILSVRC 2013 стала сверточная нейронная сеть ZF Net Мэтью Зеллера и Роба Фергюса из top-5 с ошибкой 11,2 %. Данная архитектура – улучшенная версия AlexNex: здесь увеличили размеры средних сверточных слоев и уменьшили шаг и размер фильтра на первом слое [45].

Основным достижением данной архитектуры является создание техники визуализации фильтров – сети развертки (deconvolutional network), состоящей из операций, в каком-то смысле обратных операциям сети. В итоге сеть развертки отображает скрытый слой сети на оригинальное изображение.

Чтобы изучить поведение фильтра на определенном изображении с помощью обученной нейронной сети, необходимо сначала осуществить вывод сетью, после чего в слое изучаемого фильтра обнулить все веса, кроме весов самого фильтра, и затем подать полученную активацию на слой сети развертки. В сети развертки последовательно применяются операции Unpooling, ReLU и фильтрации. Unpooling частично восстанавливает вход соответствующего слоя субдискретизации, запоминая координаты, которые выбрал слой субдискретизации. ReLU – обычный слой, применяющих функцию ReLU. Слой фильтрации выполняет операцию свертки с весами соответствующего слоя свертки, но веса каждого фильтра «перевернуты» вертикально и горизонтально.

Таким образом, исходная активация фильтра движется в обратном направлении, пока не будет отображена в оригинальном пространстве изображения.

GoogLeNet

Inveption-v1 – победитель ILSVRC 2014 из top-5 с ошибкой 6,7 %, также известный как GoogLeNet [46].

Создатели этой сети во главе с Christian Szegedy исходили из факта, что после каждого слоя сети необходимо сделать выбор – будет ли следующий слой сверткой с фильтром 3×3, 5×5, 1×1 или же слоем субдискретизации. Каждый из таких слоев полезен: фильтр 1×1 выявляет корреляцию между каналами, в то время как фильтры большего размера реагируют на более глобальные признаки, а слой субдискретизации позволяет уменьшить размерность без больших потерь информации.

Вместо того чтобы выбирать, какой именно слой должен быть следующим, предлагается использовать все слои сразу, параллельно друг другу, а затем объ-

единить полученные результаты в один. Чтобы избежать роста числа параметров, перед каждым слоем свертки используется свертка 1×1, которая уменьшает число карт признаков. Такой блок слоев назвали модулем Inception (рисунок 2.17).

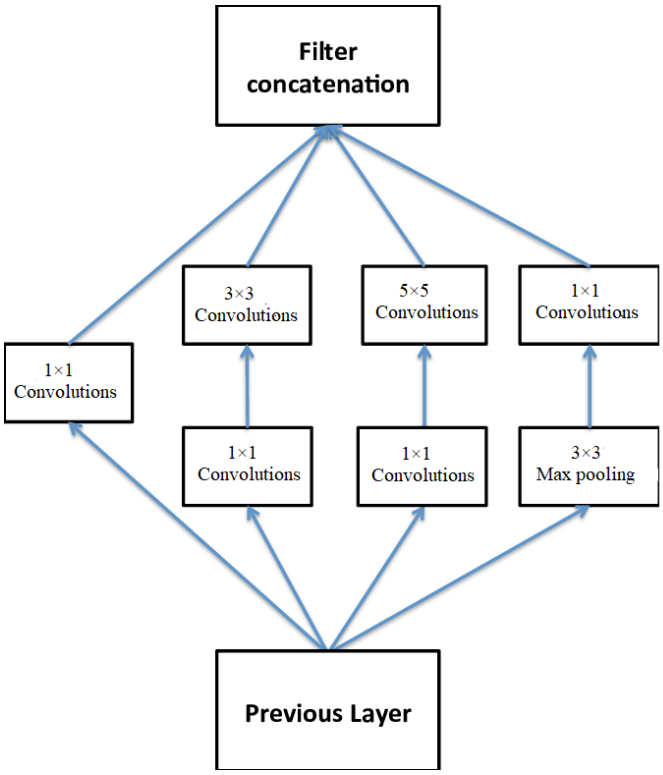


Рисунок 2.17 – Модуль Inception [46]

Также в GoogLeNet отказались от использования полносвязного слоя в конце сети, используя вместо него слой Average Pooling, благодаря чему резко уменьшилось число параметров в сети.

Таким образом, GoogLeNet, состоящая из более чем ста базовых слоев, имеет почти в 12 раз меньше параметров, чем AlexNet (около 7 млн параметров против 138 млн).

VGG 16/19

VGG Net – модель сверточной нейронной сети 2014 года, в которой отказались от использования фильтров размером более чем 3×3. Авторы показали, что слой с фильтрами 7×7 эквивалентен трем слоям с фильтрами 3×3, причем в последнем случае используется на 55 % меньше параметров.

Аналогично слой с фильтром 5×5 эквивалентен двум слоям с фильтром 3×3, которые экономят 22 % параметров сети. Визуальное представление такой декомпозиции можно увидеть на рисунке 2.18.

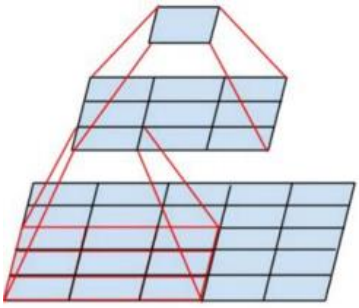


Рисунок 2.18 – Декомпозиция фильтра 5×5

На соревновании ILSVRC 2014 ансамбль из двух VGG Net из top-5 получил ошибку 7,3 %. Хотя данная модель и не победила в соревновании, благодаря своей простоте она используется в более сложных сетях, предназначенных для детектирования предметов, семантической сегментации или маскирования объектов. Архитектура VGG16 представлена на рисунке 2.19 [47].

Рисунок 2.19 – Архитектура VGG16 [47]

Разработчикам удалось наглядно продемонстрировать, что глубина является ключевым фактором для производительности. Их сеть содержит 16 сверточных и полносвязных слоев и имеет чрезвычайно однородную архитектуру, которая выполняет свертывание 3×3 и пулинг 2×2 от начала до конца. Исходная модель доступна в режиме Plug and Play во фреймворке для глубокого обучения Caffe.

В процессе обучения данная сверточная сеть принимает RGB-изображение с фиксированным размером 224×224. Единственная предобработка, которая производится, – это вычитание среднего значения RGB, вычисленного на обучающем наборе, из RGB-значения каждого пикселя.

Изображение передается через стек сверточных слоев, в котором используются фильтры с очень маленьким локальным полем восприятия: 3×3 (является наименьшим размером для захвата таких понятий как «левый/правый», «вверх/вниз», «центр»). В одной из конфигураций также использованы фильтры свертки 1×1, которые можно рассматривать как линейное преобразование входных каналов (за ними следует нелинейность). Шаг свертки фиксирован в 1 пиксель; пространственное дополнение входного сверточного слоя такое, что пространственное разрешение сохраняется после свертки, т. е. заполнение составляет 1 пиксель для фильтра размером 3×3 для каждого конволюционного слоя. Пространственное объединение осуществляется с помощью пяти max-pooling слоев, которые следуют за сверточными слоями (не все конволюционные слои сопровождаются max-pooling слоем). Max-pooling выполняется над окном размером 2×2 с шагом 2.

За стеком сверточных слоев (которые имеют разную глубину в разных архитектурах) следуют три полносвязных слоя (Fully-Connected, FC): первые два имеют по 4096 каналов каждый, третий выполняет 1000-классовую ILSVRC-классификацию и, следовательно, содержит 1000 каналов (по одному для каждого класса). Последний слой – soft-max. Конфигурация полностью подключенных слоев одинакова во всех сетях. Все скрытые слои оснащены активационными ReLU-слоями. Следует отметить, что ни одна из сетей не имеет нормализации локальной реакции (LRN-нормализации), т. к. такая нормализация не улучшает производительность в наборе данных ILSVRC, но увеличивает потребление памяти и время вычислений [48].

ResNet

Победителем ILSVRC 2015 из top-5 с ошибкой в 3,57 % стал ансамбль из шести сетей типа ResNet (Residual Network), разработанный в Microsoft Research. Ключевые особенности – интенсивное использование пакетной нормализации и специальные skip-соединения. В конце архитектуры отсутствую полносвязные слои [26].

Авторы ResNet заметили, что с повышением числа слоев сверточная нейронная сеть может начать деградировать – у нее понижается точность на валидационном множестве. Так как падает точность и на тренировочном множестве, можно сделать вывод, что проблема состоит не в переобучении сети. Есть предположение, что если сверточная нейронная сеть достигла своего предела точности в некотором слое, то все следующие слои должны будут выродиться в тождественное преобразование, но из-за сложности обучения глубоких сетей этого не происходит. Для того чтобы «помочь» сети, было решено ввести пропускающие соединения (Shortcut Connections), изображенные на рисунке 2.20.

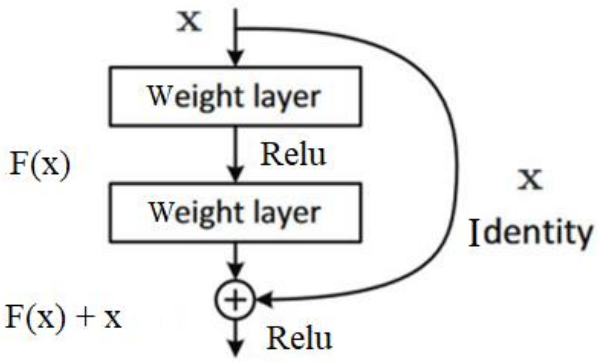


Рисунок 2.20 – Пропускающее соединение

Перед входом в вычислительный блок (стопку слоев) данные в их текущем виде сохраняются и передаются на выход блока. Таким образом, если в блоке вычисляется функция F(x) над входом x, то выходом блока будет activation(F(x) + x). Тогда, если окажется, что слой избыточен, то сети достаточно сделать F(x) = 0 для всех x, и по сети дальше будет распространяться x без изменений.

На рассмотренных примерах архитектур CNN можно проследить постепенное улучшение точности распознавания изображений, что на практике привело к постепенному вытеснению классических методов компьютерного зрения архитектурами CNN.

На рисунке 2.21 можно увидеть постепенное снижение показателей top-5 ошибки с появлением новых моделей CNN [12, 14]. Уровень погрешности оценки со стороны человека на тот же набор данных составляет 5,1 %, что означает, что в классификации образов современные CNN превзошли возможности человека.

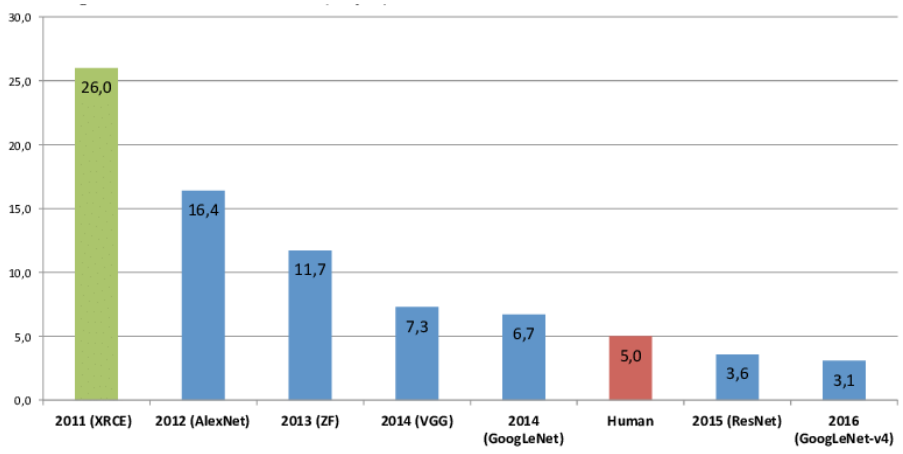


Рисунок 2.21 – Top-5 ошибок в различных моделях CNN

2.3 Рекуррентные нейросети (Recurrent Neural Networks, RNN)

Главное отличие RNN от FNN в наличии циклических связей, при которых скрытый слой свои же значения отправляет сам на себя на следующем шаге. Эта маленькая особенность дает большие преимущества при сравнении с FNN (таблица 2.1).

Таблица 2.1 – Сравнительный анализ возможностей FNN и RNN

FNN	RNN
- универсальный аппроксиматор: однослойная нейросеть с конечным числом нейронов может аппроксимировать непрерывную функцию на компактных подмножествах R^N (теорема Цыбенко); - не имеют естественной возможности учесть порядок во времени; - не обладают памятью, кроме полученной во время обучения	- тьюринг-полны́: можно реализовать любую вычислимую функцию; - обладают определенным видом памяти и гораздо лучше подходят для работы с последовательностями, моделированием контекста и временными зависимостями

Для обучения RNN используется специальный вариант метода обратного распространения ошибки (Backpropagation through time, BPTT) и «разворачивание» нейросети (рисунок 2.22). Идея очень простая – цикл RNN разворачивается на несколько шагов и получается обычная глубокая нейросеть, которую после этого обучают Backpropagation.

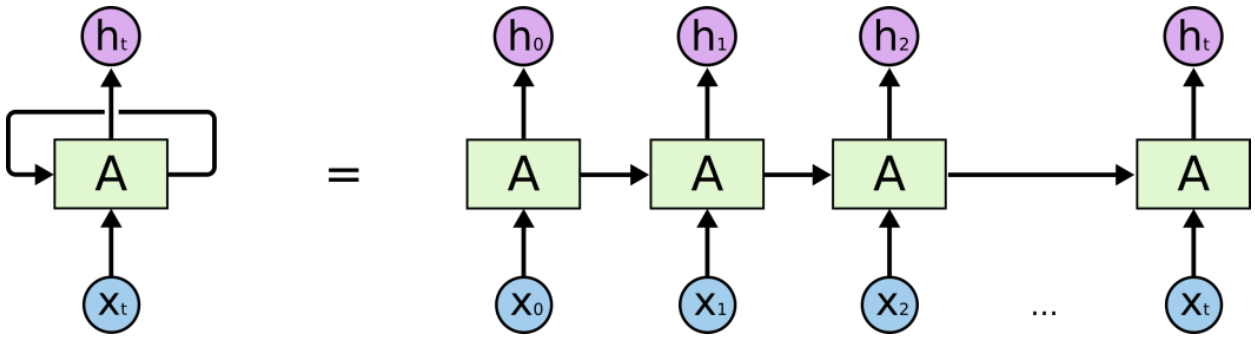


Рисунок 2.22 – Backpropagation through time, BPTT [43]

Из-за этой особенности в RNN есть проблема с затуханием градиентов при большой глубине. Для ее решения вместо простых нейронов используют более сложные ячейки памяти – LSTM и GRU, в которых появляются память и gate, контролирующие, когда эту память нужно сбросить, перезаписать или сохранить.

Рассмотренные архитектуры широко применяются при решении различных задач, но имеют свою специализацию, т. е. те области применения, где они зарекомендовали себя лучше других [11]:

- FNN в основном используется для анализа данных, в которых можно выявить повторяющийся паттерн;
- CNN активно используется при распознавании объектов, анализе фото или видео, обработке естественного языка;
- RNN используется там, где важно делать выводы, основываясь не только на настоящих данных, но и на прошедших (распознавание эмоций, обработка речи, анализ текста).

2.4 Основные архитектуры детектирования объектов на изображении

Скользящее окно

Задается конфигурация «окна обработки» – двумерной области, охватывающей конечное множество отсчетов входного изображения. В процессе обработки это окно смещается по изображению, последовательно занимая все возможные положения в плоскости дискретных аргументов. Для каждого положения окна по содержащимся в нем входным отсчетам вычисляется значение выходного отсчета, соответствующего центру окна.

К окну, которое перемещается по картинке, можно применить классификационную нейронную сеть, которая предобучена на тех классах, объекты которых нужно детектировать, и, таким образом, если классификационная сеть говорит, что в данном окне объект есть, то он и помечается соответствующей рамкой и классом.

Но в этом подходе есть определенный недостаток: чтобы пройти скользящим окном на разных масштабах по изображению, нужно многократно применить классификационную нейронную сеть. А это означает, что данный процесс будет идти очень медленно [21].

Selective search

Объекты могут встречаться в любом масштабе внутри изображения. Кроме того, некоторые объекты имеют менее четкие границы, чем другие объекты. Поэтому в Selective search учитываются все масштабы объектов. Это достигается с помощью иерархического алгоритма. Затем регионы, которые схожи (цвет, текстура) и находятся рядом, объединяются в один регион.

Целью Selective search является создание набора возможных мест размещения объектов для использования в практической системе распознавания объектов. Найденные регионы попадают в классификатор. Количество регионов гораздо меньше чем в методе скользящего окна, поэтому скорость детекции объектов увеличивается. Данный метод является основой для многих подходов [22].

R-CNN

Основным назначением R-CNN (Region-based Convolutional Neural Networks) является обнаружение объектов и определение их класса. Решение этой задачи сводится к выделению всех объектов на изображении. В процессе Selective search генерируется 2000 различных регионов, которые с наибольшей вероятностью содержат объект. После того как регионы сгенерированы, они приводятся к размеру, пригодному для обработки с помощью CNN, которая извлекает вектор признаков каждого региона.

Далее вектор признаков подается на вход набору линейных классификаторов и выполняется классификация. Этот вектор также подается на вход регрессии для вычисления максимально точных координат ограничивающей рамки. Далее выполняется алгоритм подавления немаксимумов (non-maximum suppression), который отклоняет регион, если он в значительной степени совпадает с другим регионом, имеющим более высокую оценку [16].

Fast R-CNN

Fast R-CNN является улучшением R-CNN, т. к. R-CNN имеет ряд недостатков, связанных с высокими временными затратами [17].

В R-CNN требуется натренировать сверточную нейронную сеть в два этапа. Затем требуется натренировать набор SVM по одной для каждого класса и линейные регрессоры (по одному для каждого класса) для восстановления позиций объекта.

Временные затраты на этапе детектирования сильно снижают практическую пользу данного подхода. Причина, из-за которой детектирование работает так медленно, заключается в том, что при помощи selective search генерируется 2000 претендентов на изображении, и каждый претендент должен быть пропущен через сверточную сеть, чтобы вычислить для него вектор особенностей. Процесс достаточно затратный, особенно для сетей с большим числом сверточных слоев. Поэтому авторы предлагают подавать на вход сети полное изображение, но при этом последний слой max-pool заменить на слой нового типа RoI pooling.

RoI pooling сопоставляет регионы и карту признаков и вычисляет max pooling [9]. После RoI pooling слоя данные через два полносвязных слоя подаются параллельно:

- на слой для оценки принадлежности данного претендента одному из классов объектов;
- слой, реализующий регрессию, которая уточняет границы объекта.

Faster R-CNN

В этой архитектуре была разработана специальная сеть для предложения регионов (region proposal network, RPN). Ее поместили после последнего сверточного слоя. Эта сеть позволяет генерировать предлагаемые регионы на основе лишь последней свертки, а затем – классификация и регрессия [18]. Из семейства R-CNN Faster R-CNN является наиболее точной и быстрой. На рисунке 2.23 представлены результаты теста скорости алгоритмов (в секундах).

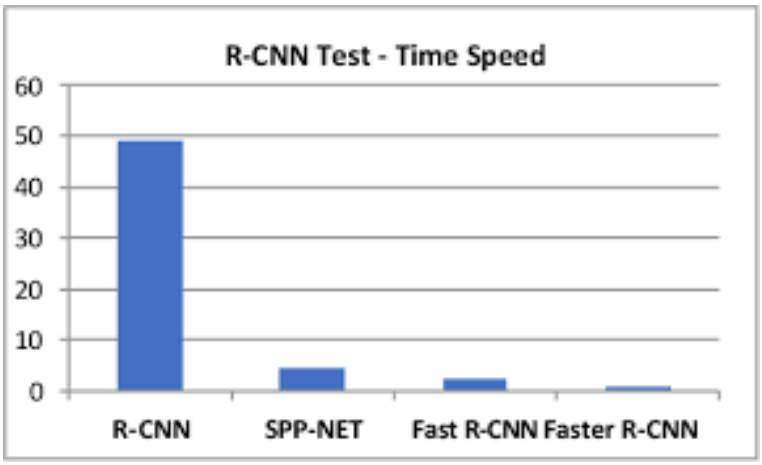


Рисунок 2.23 – Результаты скорости алгоритмов

YOLO

YOLO применяет единую нейронную сеть к полному изображению. Эта сеть делит изображение на регионы и предсказывает ограничивающие поля и вероятности для каждого региона.

Модель имеет ряд преимуществ перед системами на основе классификаторов. Алгоритм представлен на рисунке 2.24. Он просматривает весь образ во время тестирования, поэтому его прогнозы сообщаются глобальным контекстом изображения. Данный подход достаточно быстрый: более чем в 1000 раз быстрее, чем R-CNN, и в 100 раз быстрее, чем Fast R-CNN, однако проигрывает в точности [15].

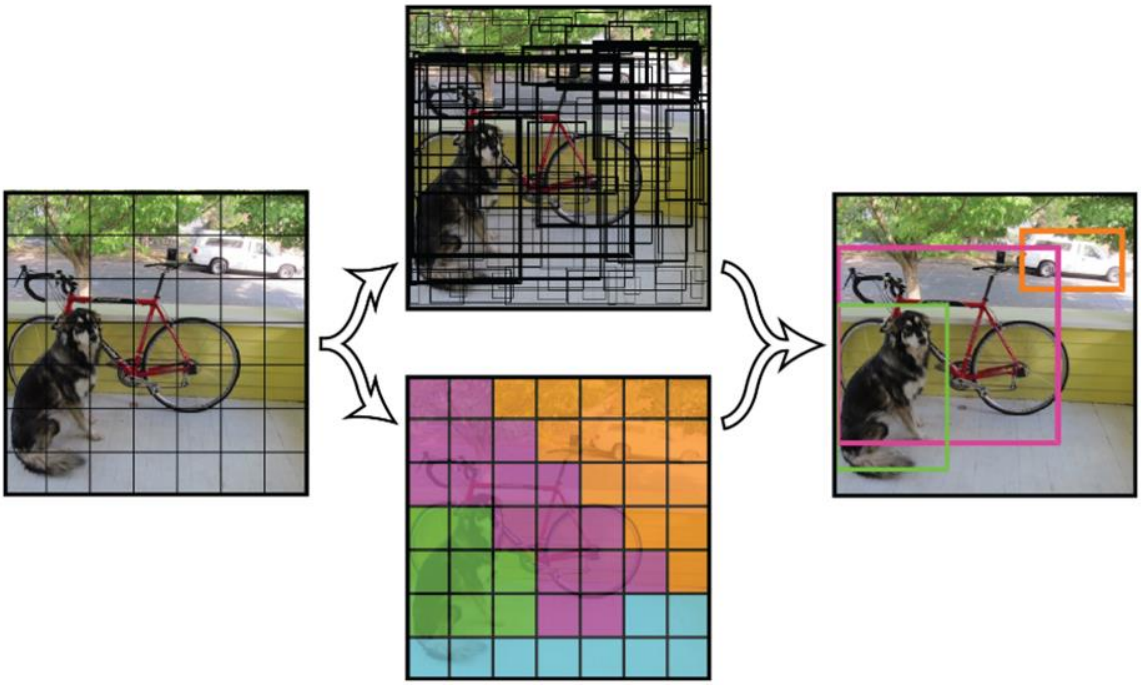


Рисунок 2.24 – Структура алгоритма YOLO [15]

2.5 Основные архитектуры семантического сегментирования изображений

До того как глубокое обучение захватило компьютерное зрение, для семантической сегментации использовались классификаторы Texton Forest и Random Forest, так же как и в классифицировании изображений огромным успехом пользовались сверточные нейронные сети (CNN).

Изначально одним из популярных методов глубокого обучения являлась классификация с помощью патчей, когда каждый пиксель отдельно относится к определенному классу, используя патч изображения вокруг него. Применение патчей было обусловлено тем, что в классификационных сетях обычно присутствуют полностью связанные слои, из-за чего требуются изображения фиксированного размера.

В 2014 году благодаря полностью сверточной сети (FCN), разработанной Джонатаном Лонгом, Эваном Шелхамером и Тревором Дарелом, началась популяризация использования архитектур CNN для предсказаний без каких-либо полносвязных слоев. Данные подходы позволяли создавать карты сегментации для изображения любого размера, а также выполнять данное построение намного быстрее, чем подход классификации патчей. Практически все последующие современные подходы к семантической сегментации приняли данную парадигму.

Помимо полностью связанных слоев одной из основных проблем использования CNN для сегментации являются слои объединения (pooling layers). Данные слои увеличивают поле зрения и позволяют агрегировать контекст, отказываясь от позиционной информации. Однако семантическая сегментация требует точного выравнивания карт классов, поэтому позиционная информация необходима. Для решения данной проблемы возникли два разных класса архитектур.

Первая из них – архитектура типа «шифратор – дешифратор». Шифратор постепенно уменьшает пространственный размер с помощью слоев объединения, а дешифратор восстанавливает детали и пространственный размер объекта. Обычно присутствуют соединения от шифратора к дешифратору, для того чтобы дешифратор точнее восстанавливал детали.

Одной их самых популярных архитектур данного класса является U-Net архитектура [40] (рисунок 2.25).

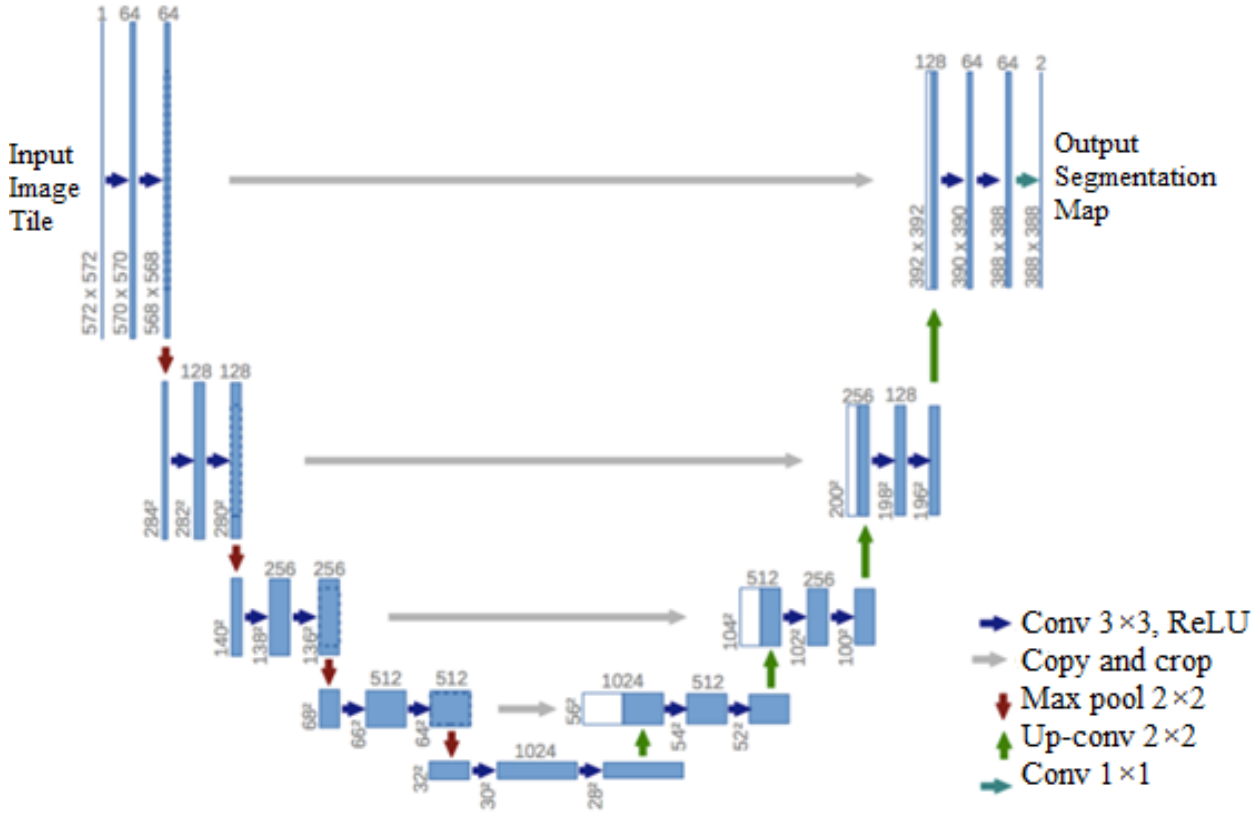


Рисунок 2.25 – U-Net: архитектура типа «шифратор – дешифратор»

Архитектуры второго класса используют расширенные свертки (рисунок 2.26) и отказываются от слоев объединения.

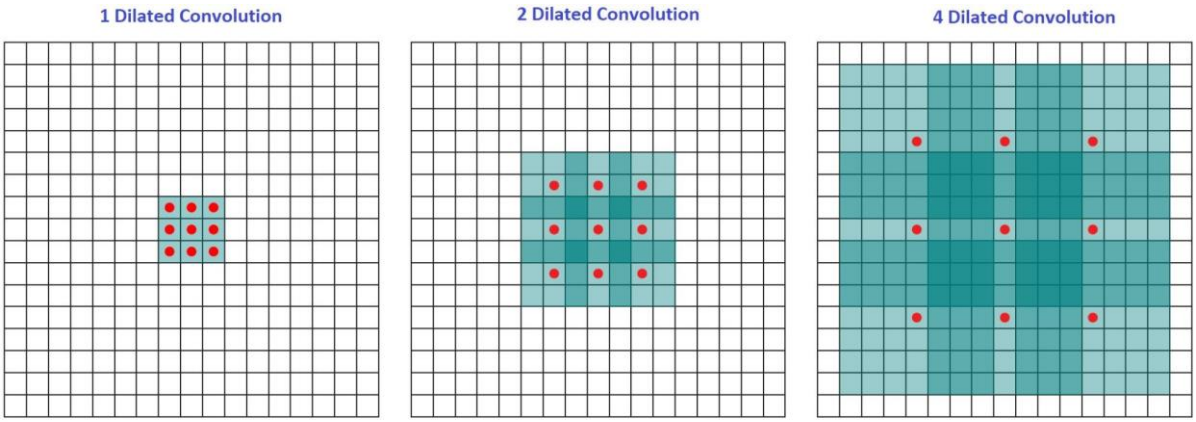
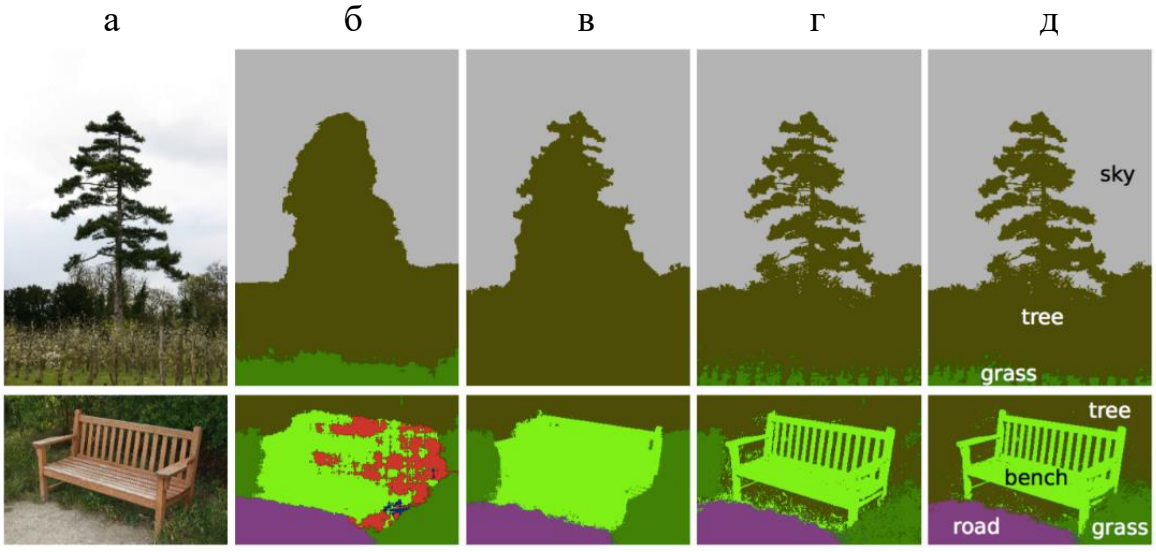


Рисунок 2.26 – Расширенные свертки

Примечание – Коэффициент, равный единице, соответствует нормальным сверткам [53].

Для улучшения сегментации обычно используется постобработка условно случайного поля (CRF), варианты которого изображены на рисунку 2.27. Данная обработка основана на том, что пиксели с аналогичной интенсивностью, как правило, помечены как один и тот же класс. CRF могут улучшить результаты обработки на 1–2 %.



а – Image; б – Unary classifiers; в – Robust P^n CRF; г – Fully connected CRF, MCMC inference, 36 hrs; д – Fully connected CRF, our approach, 0,2 seconds

Рисунок 2.27 – Широко используемые варианты CRF

Ниже приведена эволюция архитектур сегментации, начавшаяся с FCN.

Полностью сверточные сети (FCN)

Ключевой вклад данного типа нейронных сетей был в том, что он популяризовал использование сквозных сверточных сетей для семантической сегментации. Сети такого типа позволяли повторное использование натренированных сетей. Повышение дискретизации достигалось с использованием деконволюционных слоев, а внедрение сокращенных соединений приводило к смягчению грубости дискретизации.

Полностью связные слои в классификационных сетях можно рассматривать как свертки с ядрами, которые охватывают все области ввода (рисунок 2.28). Они являются эквивалентом исходной сети классификации при перекрытии входных патчей, однако именно они более эффективны, т. к. вычисление делится на перекрывающиеся области патчей.

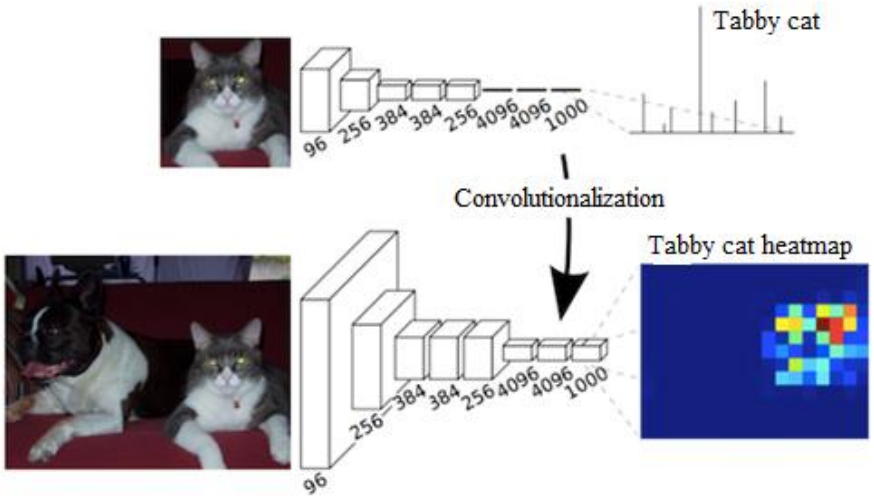


Рисунок 2.28 – Полностью связанные слои как свертка

После свертывания полностью подключенных слоев в предварительно сконфигурированной сети изображений, как VGG (Visual Geometry Group), частота дискретизации функциональных карт все еще должна быть повышена из-за операций объединения в CNN. Вместо использования простой билинейной интерполяции данную операцию могут выполнить деконволюционные слои.

Однако повышение частоты дискретизации даже с деконволюционными слоями приводит к грубым картам сегментации из-за потери информации во время объединения. Поэтому сокращенные соединения вводятся с функциональных карт с более высоким разрешением.

SegNet: архитектура типа «шифратор – дешифратор»

Ключевой вклад нейронных сетей SegNet [41] был в том, что индексы максимального объединения (max pooling), переданные в дешифратор, улучшают разрешение сегментации.

FCN, несмотря на деконволюционные слои и некоторое количество сокращенных соединений, производит достаточно грубые карты сегментации. Таким образом, в SegNet (рисунок 2.29) вводятся более короткие соединения. Однако вместо копирования функций шифратора, как в FCN, копируются индексы максимального объединения – процесса дискретизации на основе выборки.

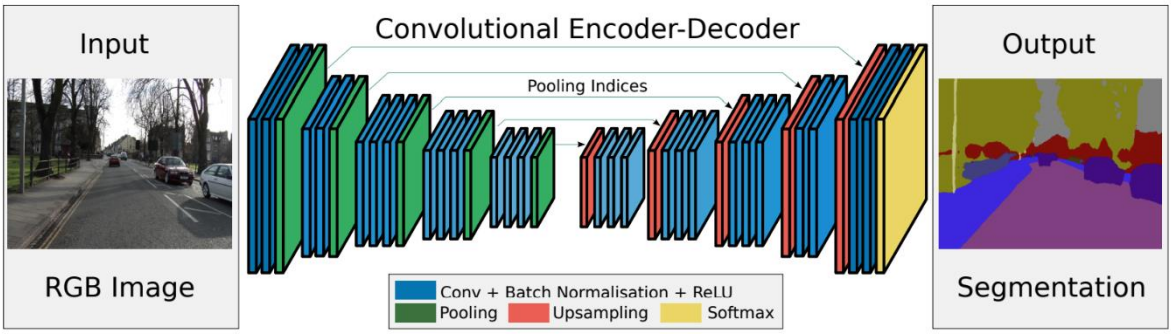


Рисунок 2.29 – SegNet-архитектура

Расширенные свертки

Расширенные свертки в таком случае используются в сверточном слое для более плотных предсказаний, а также позволяют проводить многоуровневое агрегирование контекста по расширенным сверткам.

Операция объединения в классификационных сетях помогает увеличить восприимчивое поле, однако вместе с тем она уменьшает разрешение результата. Для исправления данной ситуации используется расширенный слой свертки (рисунок 2.30).

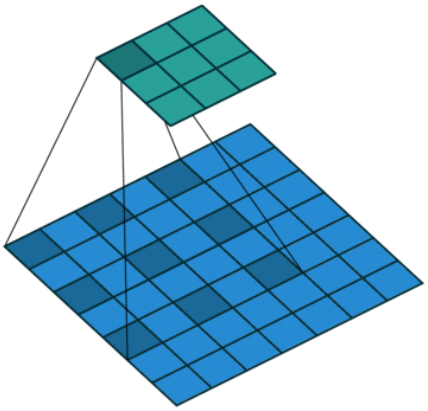


Рисунок 2.30 – Расширенная свертка

Расширенный сверточный слой позволяет экспоненциально увеличить поле зрения без уменьшения пространственных размеров.

Последние два слоя объединения из предварительно натренированной классификационной сети (например, VGG) удаляются, а последующие сверточные слои заменяются на расширенные свертки. С данным модулем, называемым frontend, наблюдаются плотные предсказания без какого-либо увеличения числа параметров.

Модуль, называемый контекстным, обучается отдельно, в качестве входных данных используются выходы frontend-модуля. Данный модуль представляет собой каскад, необходимый для того, чтобы агрегировать контекст с множеством масштабов и улучшить предсказания из frontend [54].

DeepLab (первая и вторая версия)

Особенности DeepLab архитектуры:

- использование расширенных сверток;
- предложение сложного пула пространственной пирамиды (Atrous Spatial Pyramid Pooling, ASSP);
- использование полностью подключенной CRF.

Расширенные свертки увеличивают поле зрения без увеличения числа параметров. Сеть модифицируется, как и в архитектуре VGG (рисунок 2.31).

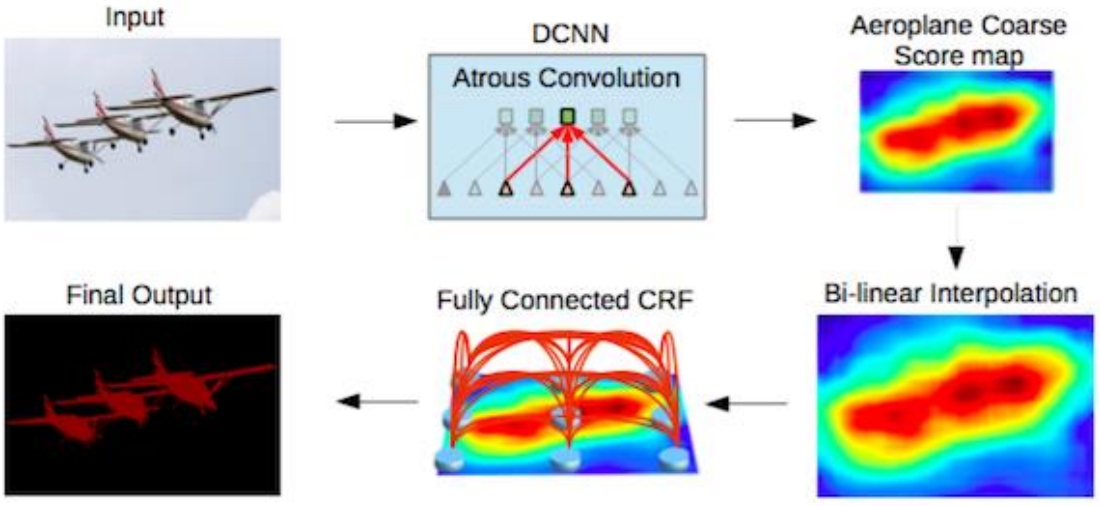


Рисунок 2.31 – Стадии DeepLab2 [49]

Многомасштабная обработка достигается либо путем передачи нескольких масштабированных версий исходных изображений в параллельные ветви CNN (пирамида изображений), и/или с использованием нескольких параллельных расширенных сверточных слоев с разной частотой дискретизации (ASSP).

Структурированное прогнозирование выполняется с помощью подключенного CRF, который обучается отдельно.

RefineNet

Особенности архитектуры (рисунок 2.32):

- архитектура «шифратор – дешифратор» с хорошо продуманными блоками дешифратора;
- все компоненты соответствуют схеме остаточного соединения.

Подход и использование расширенных сверток не лишен недостатков. Расширенные свертки являются дорогостоящими с точки зрения вычислений и памяти, поскольку они применяются на большом количестве функциональных карт с высоким разрешением, что в свою очередь затрудняет получение высокоточных прогнозов. Например, прогнозы DeepLab составляют 1/8 часть от исходного размера.

В данном подходе часть шифратора реализована блоками ResNet-101. Дешифратор имеет блоки RefineNet, каждый из которых объединяет функции высокого разрешения от шифратора и функции низкого разрешения от предыдущего блока RefineNet.

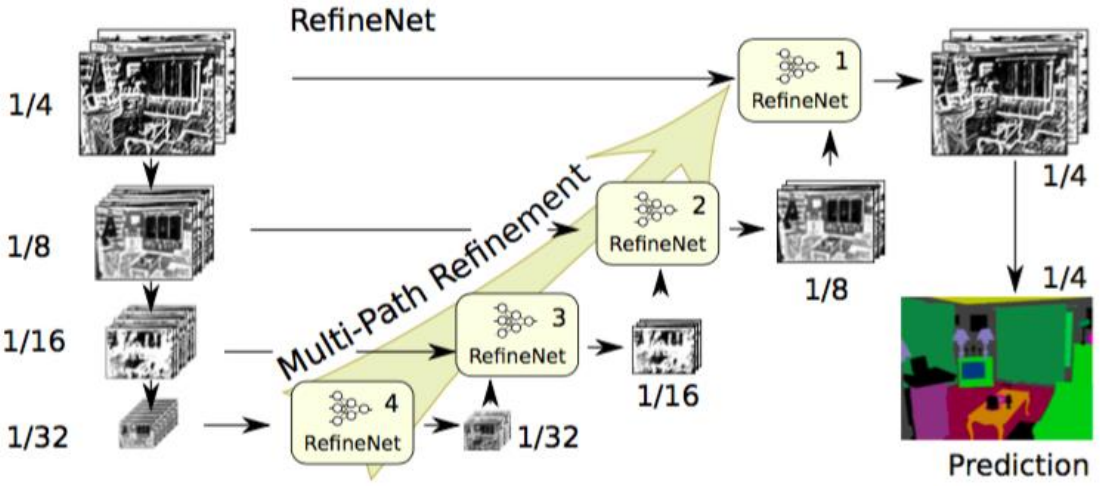
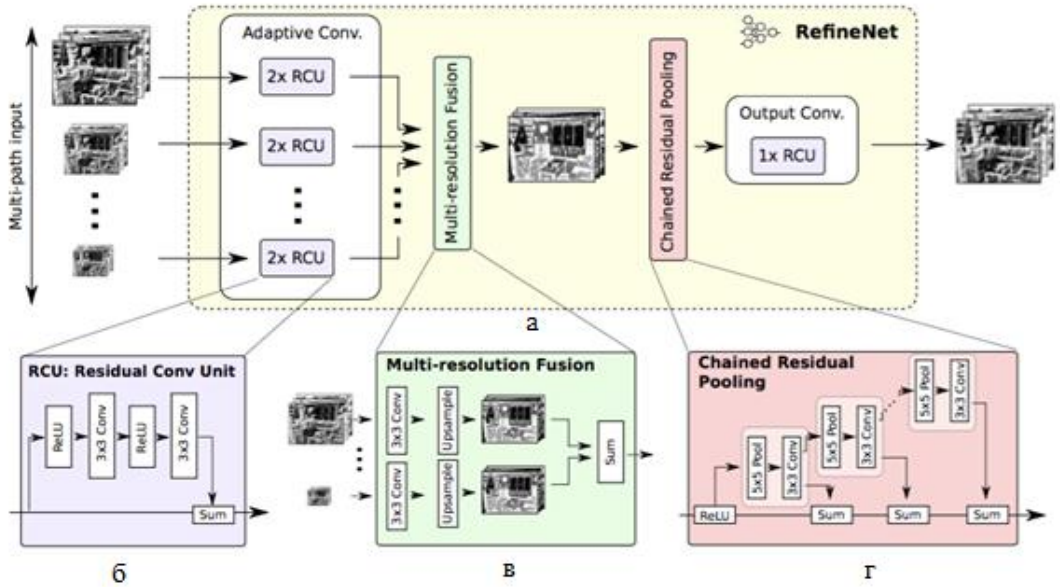


Рисунок 2.32 – Архитектура RefineNet [50]

Каждый RefineNet блок (рисунок 2.33) имеет компонент, который должен сглаживать функции множественного разрешения, повышая дискретизацию функций с более низким разрешением, и компонент для захвата контекста на основе 5×5 с шагом 1 повторяющихся объединительных слоев. Все эти компоненты используют схему остаточного соединения, следуя образцу карты идентификации [50].



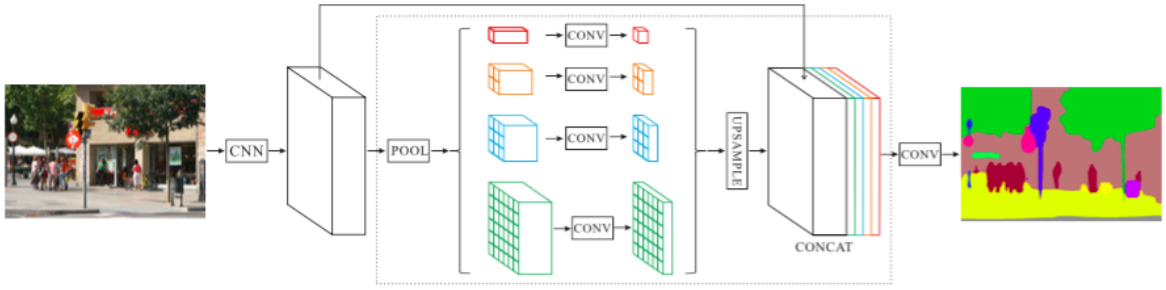
а – RefineNet; б – RCU; в – Multi-Resolution Fusion; г – Chained Residual Pooling

Рисунок 2.33 – Блок RefineNet [50]

PSPNet

Особенности архитектуры (рисунок 2.34):

- предложение модуля пула пирамиды для агрегирования контекста;
- использование вспомогательных потерь.



а б в г

а – Input Image; б – Feature Map; в – Pyramid Pooling Module; г – Final Prediction

Рисунок 2.34 – Архитектура PSPNet [51]

Категории глобальной сцены необходимы, т. к. они дают ключ к распределению классов сегментации. Модуль пула пирамид фиксирует данную информацию, применяя большие объединительные слои ядра.

Расширенные свертки используются для модификации ResNet, к которой добавляется модуль пула пирамид. Данный модуль объединяет карты функций из ResNet с выходом параллельных слоев объединения с ядрами, покрывающими все изображение, половину или небольшую его часть.

Вспомогательные потери, дополнительные для потерь на основной ветке, применяются после четвертого этапа ResNet (т. е. ввода в модуль пула пирамид). Такая идея также называется наблюдением [51].

Large Kernel Matters: улучшение семантической сегментации глобальной сверточной сетью

Суть данного подхода заключается в предложении архитектуры типа «шифратор – дешифратор» с очень большими свертками ядра.

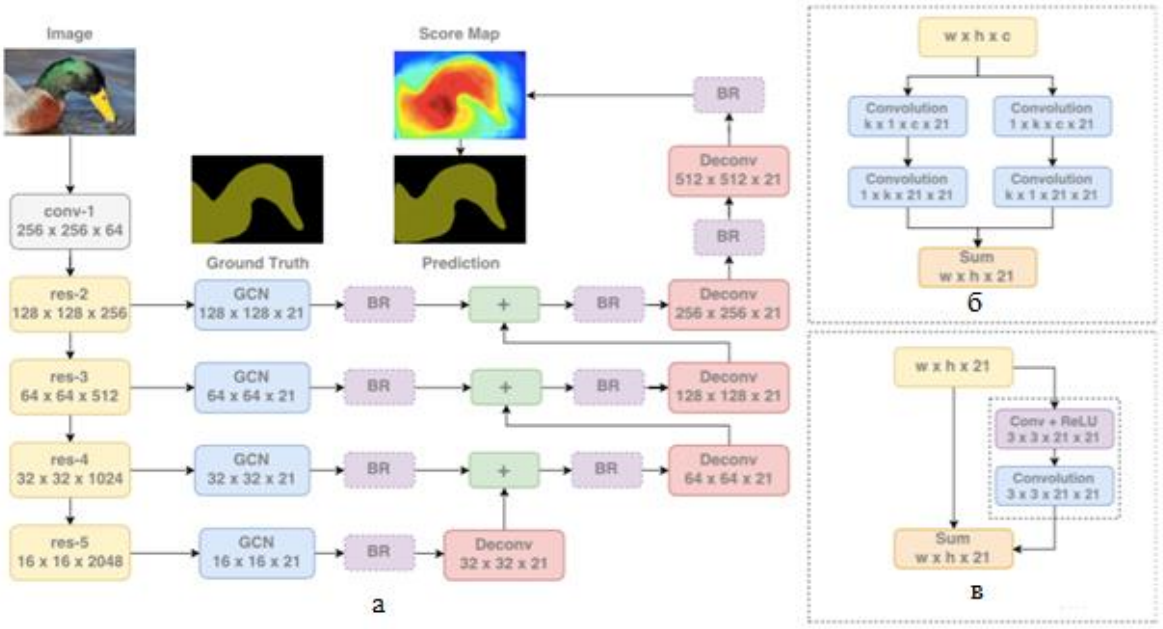
Семантическая сегментация требует как сегментации, так и классификации сегментированных объектов. Так как полностью подключенные слои не могут присутствовать в архитектуре сегментации, вместо них используются свертки с очень большими ядрами.

Еще одна причина в пользу больших ядер заключается в том, что, хотя более глубинные сети, такие как ResNet, имеют очень большую восприимчивую

область, исследования показывают, что такие сети имеют тенденцию собирать информацию из гораздо меньшего региона (действительная восприимчивая подача).

Большие ядра вычислительно затратны и имеют множество параметров. Поэтому свертка размером k × k аппроксимируется суммой сверток 1 × k + k × 1 и k × 1 + 1 × k. Данный модуль называется глобальной сверточной сетью (GCN).

Архитектура этого подхода представляет собой часть шифратора, состоящую из ResNet без каких-либо расширенных сверток, и часть дешифратора, состоящую из GCN (рисунок 2.35) и сверток. Также используется простой остаточный блок (BR) [52].



а – Whole Pipeline; б – Global Convolutional Network; в – Boundary Refinement

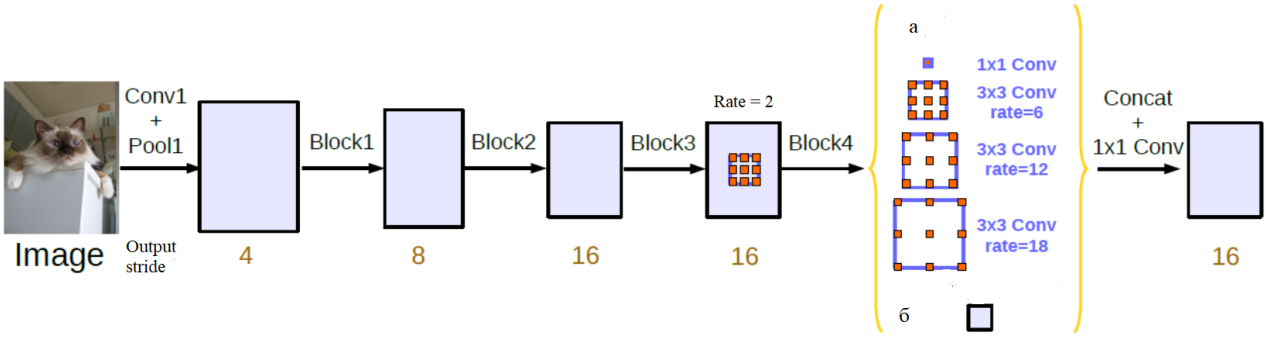
Рисунок 2.35 – GCN-архитектура [52]

DeepLab третьей версии

Особенности архитектуры:

- улучшение ASPP;
- модуль, который использует расширенные свертки в каскаде.

Модель ResNet, как и в DeepLab второй версии, модифицируется для использования расширенных сверток. Улучшенный ASPP (рисунок 2.36) включает в себя конкатенацию функций уровня изображения, свертки 1×1 и трех сверток 3×3 с разными коэффициентами. После каждого из параллельных слоев используется нормализация пакета.



а – Atrous Spatial Pyramid Pooling; б – Image Pooling

Рисунок 2.36 – DeepLabv3 ASPP [53]

Каскадный модуль представляет из себя блок ResNet, отличие заключается лишь в том, что слои свертки компонентов сделаны расширенными с различными коэффициентами. Данный модуль аналогичен контекстному модулю, используемому в расширенных свертках, однако он применяется непосредственно на картах промежуточных объектов вместо карт убеждений (данные карты являются окончательными картами признаков CNN с каналами, равными количеству классов) [53].

U-Net

U-Net соответствует архитектуре «кодер – декодер». Кодировщик постепенно уменьшает пространственное измерение с помощью объединения слоев, а декодер постепенно восстанавливает детали объекта и пространственное измерение. Также существуют быстрые соединения от кодера к декодеру, чтобы помочь декодеру лучше восстановить детали объекта. Архитектура сети схематично изображена на рисунке 2.25.

Сеть не имеет полностью соединенных слоев и использует только действительную часть каждой свертки, т. е. карта сегментации содержит только пиксели, для которых полный контекст доступен во входном изображении. Для качественной сегментации U-Net увеличивает количество данных путем деформации имеющихся изображений.

Данный тип нейронной сети был разработан с учетом особенностей обработки медицинских изображений и позволяет достичь высокой точности сегментации на небольших наборах данных [40].

3 ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ ГЛУБОКИХ НЕЙРОННЫХ СЕТЕЙ ПРИ РЕШЕНИИ ЗАДАЧ СЕГМЕНТАЦИИ, КЛАССИФИКАЦИИ ИЗОБРАЖЕНИЙ

Компьютерное зрение – это междисциплинарная научная область, которая занимается методами создания компьютеров, способными получать высокий уровень понимания цифровых изображений или видео. С точки зрения инженерии она стремится автоматизировать задачи, которые может выполнять зрительная система человека.

Задачи компьютерного зрения включают методы получения, обработки, анализа и понимания цифровых изображений и извлечения многомерных данных из реального мира для получения числовой или символической информации, например, в форме решений. Понимание в этом контексте означает преобразование зрительных образов (ввод сетчатки) в описания мира, которые могут взаимодействовать с другими мыслительными процессами и вызывать соответствующие действия. Такое понимание изображения можно рассматривать как отделение символической информации от данных изображения с использованием моделей, построенных с помощью геометрии, физики, статистики и теории обучения.

Рисунок 3.1 – Дисциплины, используемые в компьютерном зрении

Именно глубокое обучение позволило так быстро развиться компьютерному зрению за последние несколько лет.

Существуют различные уровни детализации, в которых компьютер получает представление об изображениях. Для каждого из этих уровней в области компьютерного зрения определена проблема. Такими уровнями являются:

- классификация изображений;
- классификация с локализацией;
- обнаружение объекта;
- семантическая сегментация;
- сегментация экземпляра [55].

Классификация изображений

Самым фундаментальным строительным блоком в компьютерном зрении является проблема классификации изображений, когда для данного изображения мы ожидаем, что компьютер выведет дискретную метку, которая относится к основному объекту на изображении.

В классификации изображений предполагается, что в изображении есть только один объект, а не несколько. Схематично данный процесс изображен на рисунке 3.2.

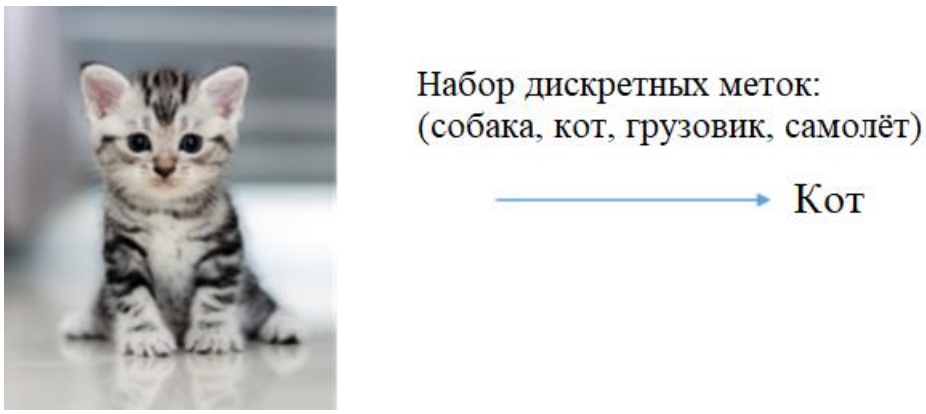


Рисунок 3.2 – Схематическое обозначение задачи классификации изображений

Классификация с локализацией

При локализации вместе с дискретной меткой мы также ожидаем, что вычисление локализует, где именно объект присутствует на изображении. Эта локализация обычно реализуется с использованием ограничивающей рамки, которая может быть идентифицирована некоторыми числовыми параметрами относительно границы изображения. Даже в этом случае предполагается, что на изоб-

ражение должно быть только один объект. Схематично данный процесс изображен на рисунке 3.3 [55].

Рисунок 3.3 – Различие между классификацией и классификацией с локализацией

Обнаружение объектов

Обнаружение объектов расширяет локализацию до следующего уровня, теперь на изображении нет ограничения на количество объектов, которые на нем размещены. Задача состоит в том, чтобы классифицировать и локализовать все объекты на изображении. Здесь снова локализация осуществляется с использованием концепции ограничивающего прямоугольника. Схематично данный процесс изображен на рисунке 3.4 [55].

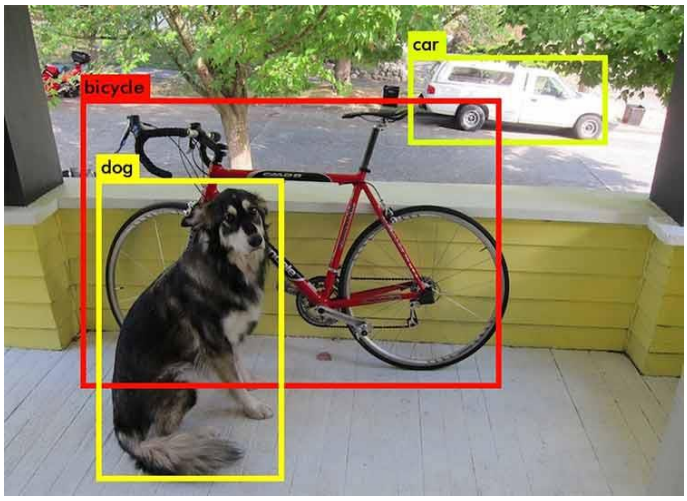


Рисунок 3.4 – Пример обнаружения объектов

Семантическое сегментирование

Цель семантической сегментации изображения состоит в том, чтобы пометить каждый пиксель изображения соответствующим классом представляемого изображения. Поскольку мы прогнозируем каждый пиксель изображения, эту задачу обычно называют плотным прогнозом.

В отличие от предыдущих задач ожидаемый результат в семантической сегментации – это не просто метки и параметры ограничивающего прямоугольника. Сам вывод представляет собой изображение с высоким разрешением (как правило, того же размера, что и входное изображение), в котором каждый пиксель классифицируется по определенному классу. Таким образом, это классификация изображений на уровне пикселей. Схематично данный процесс изображен на рисунке 3.5 [55].

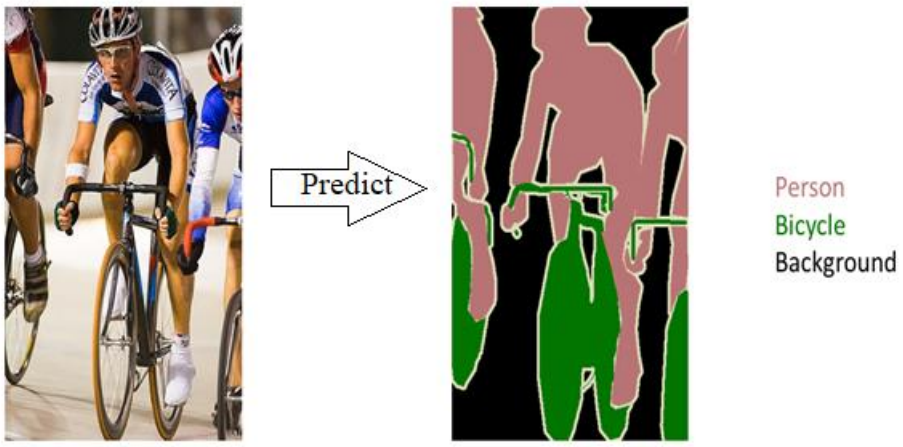


Рисунок 3.5 – Ожидаемый результат семантического сегментирования [49]

Сегментация экземпляра

Сегментация экземпляров находится на один уровень выше семантической сегментации, т. к. наряду с классификацией на уровне пикселей мы ожидаем, что в ней компьютер будет классифицировать каждый экземпляр класса отдельно.

Например, на изображении более пяти человек отображены технически пять экземпляров класса «Человек». Все пять классифицируются отдельно (разными цветами). Но семантическая сегментация не делает различий между экземплярами определенного класса. Схематично данный процесс изображен на рисунке 3.6.

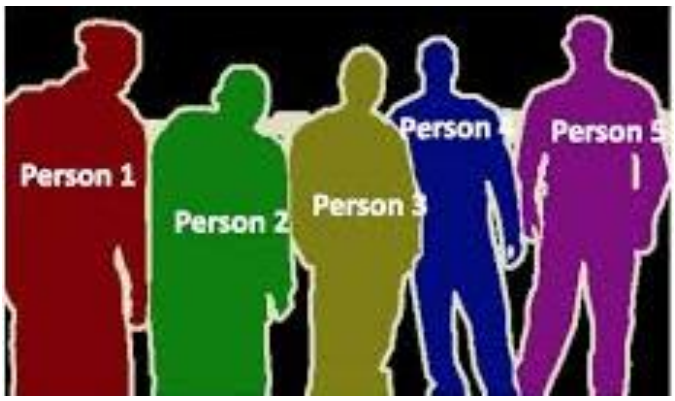


Рисунок 3.6 – Ожидаемый результат сегментации экземпляров

Для обучения нейронных сетей необходимо использовать для обучения большое количество картинок. До 2010 года основная проблема заключалась в том, что не существовало достаточно большого dataset, который способен был бы научить сеть с большим количеством параметров распознавать изображения. Самые большие базы данных, которые существовали до этого времени, – PASCAL VOC (20 категорий объектов) и Caltech 101 (9,146 изображений и 101 категория). Тем же, кто не сумел найти свои объекты ни в одной из этих баз данных, приходилось строить свои базы данных. Однако в 2010 году появилась база ImageNet, которая насчитывает 15 млн изображений, разделенных на 22 тыс. категорий. Это решило проблему обучения нейронных сетей.

Одновременно с появлением базы ImageNet возникло соревнование ImageNet (международный challenge), в котором могут принять участие все желающие. Из-за того что на изображениях в базе ImageNet может присутствовать несколько объектов и лишь один из них аннотирован, в ImageNet основной оценкой ошибки является ошибка из top-5. При ее использовании считается, что алгоритм не ошибся, если правильная категория объекта находится среди пяти категорий, выданных алгоритмом как наиболее вероятные. Вследствие этого многие нейронные сети для задачи классификации оцениваются именно с помощью ошибки из top-5.

Сферами применения семантической сегментации являются автономные транспортные средства, картографирование, точное земледелие, медицинская диагностика изображений и др.

Автономные транспортные средства. Автономное вождение представляет собой сложную робототехническую задачу, которая требует восприятия, планирования и выполнения в постоянно меняющихся условиях. Эта задача также должна выполняться с предельной точностью, поскольку безопасность

имеет первостепенное значение. Семантическая сегментация предоставляет информацию о свободном пространстве на дорогах, а также обнаруживает разметку полос и дорожные знаки.

Картографирование. Проблемы семантической сегментации также могут рассматриваться как проблемы классификации, когда каждый пиксель классифицируется как один из ряда классов объектов. Таким образом, существует возможность использования сегментации для картографирования используемых земель со спутниковых изображений. Информация о земном покрове важна для различных применений, таких как мониторинг районов обезлесения и урбанизации. Чтобы распознать тип земного покрова (например, городские районы, сельское хозяйство, водные ресурсы и т. д.), классификацию земного покрова можно рассматривать как многоклассовую задачу семантической сегментации. Обнаружение дорог и зданий также является важной темой исследований для управления движением, городского планирования и мониторинга дорог.

Точное земледелие. Роботы для точного земледелия могут уменьшить количество гербицидов, которые необходимо распылять на полях, а семантическая сегментация сельскохозяйственных культур и сорняков помогает им в режиме реального времени инициировать действия по прополке. Такие передовые методы визуального изображения для сельского хозяйства могут уменьшить ручной мониторинг сельского хозяйства.

Медицинская диагностика изображений. Машины могут дополнять анализ, выполняемый рентгенологами, значительно сокращая время, необходимое для проведения диагностических тестов [55].

4 ФРЕЙМВОРКИ И БИБЛИОТЕКИ ДЛЯ РАБОТЫ С ГЛУБОКИМИ НЕЙРОННЫМИ СЕТЯМИ

К программным инструментам глубокого обучения относят программные библиотеки для обучения нейронных сетей, позволяющие использовать готовые алгоритмы создания и обучения нейросетевых моделей. Существующие инструменты глубокого обучения имеют различный функционал и требуют от пользователя разного уровня знаний и навыков. Правильный выбор инструмента – важная задача, позволяющая добиться необходимого результата за наименьшее время и с меньшей затратой сил.

Рассмотрим общие сведения о наиболее популярных открытых библиотеках и проведем сравнительный анализ библиотек по глубинному обучению [29].

Theano [35]

Theano – это расширение языка Python, позволяющее эффективно вычислять математические выражения, содержащие многомерные массивы. Библиотека реализована на языке Python, поддерживается на операционных системах Windows, Linux и Mac OS. В состав Theano входит компилятор, который переводит математические выражения, написанные на языке Python, в эффективный код на языках C или CUDA. Библиотека позволяет использовать возможности GPU без изменения кода программы, что делает ее незаменимой при выполнении ресурсоемких задач. Возможна реализация многослойных полностью связанных сетей (Multi-Layer Perceptron), сверточных нейросетей (CNN), рекуррентных нейронных сетей (Recurrent Neural Networks, RNN), автокодировщиков и ограниченных машин Больцмана.

Модели искусственного интеллекта на основе Theano достигают высокой точности в вычислительных операциях, требующих большой вычислительной мощности.

Особенности:

- фреймворк обеспечивает превосходную точность даже при минимальных значениях;
- модульное тестирование является важной особенностью Theano.

Достоинства:

- фреймворк Theano обеспечивает эффективную поддержку всех приложений с интенсивным использованием данных, но требует объединения с другими библиотеками;
- платформа отлично оптимизирована для работы как с CPU, так и с GPU.

Недостаток: для текущей версии Theano не запланирован выпуск обновлений и добавление функционала [29].

TensorFlow [34]

TensorFlow – довольно молодой фреймворк для глубокого машинного обучения, разрабатываемый в Google Brain. Долгое время фреймворк разрабатывался в закрытом режиме под названием DistBelief, но после глобального рефакторинга 9 ноября 2015 года был выпущен в Open Source. В библиотеке языки программирования C++ и Python поддерживаются на операционных системах Linux, Mac OS X, Windows. Библиотека учитывает нужды комплексных вычислений: она обслуживает вычисления, распределенные на CPU/GPU, и несколько систем, заботясь о дублировании.

Разработанная компанией Google, TensorFlow – это надежная платформа с открытым исходным кодом, поддерживающая глубокое обучение, доступ к которой можно получить даже со смартфона.

TensorFlow – это отличный инструмент для создания и разработки статистических программ. Поскольку фреймворк предлагает распределенное обучение, все модели ИИ будут обучаться намного эффективнее на любом уровне абстракции, который предпочитает пользователь.

Особенности:

- масштабируемый мультипрограммный интерфейс;
- постоянное развитие платформы за счет огромного сообщества энтузиастов и открытого исходного кода;
- платформа предоставляет обширную документацию.

Достоинства:

- TensorFlow основан на Python;
- система использует вычислительную графическую абстракцию для создания моделей ИИ;
- существует возможность визуализации процесса тренировки нейронных сетей с помощью TensorBoard, который позволяет строить граф сети, графики с различными параметрами и графики с распределением признаков в тренировочных данных.

Недостаток: для принятия решения или прогнозирования фреймворк передает входные данные через несколько узлов – этот процесс занимает много времени [29].

Torch [36]

Torch – библиотека для научных вычислений с широкой поддержкой алгоритмов машинного обучения. Разрабатываются Idiap Research Institute, New York University и NEC Laboratories America, начиная с 2000 года, распространяются под лицензией BSD. Библиотека реализована на языке Lua с использованием C и CUDA. На данный момент поддерживаются операционные системы Linux, FreeBSD, Mac OS X. Основные модули также работают и на Windows. Библиотека состоит из набора модулей, каждый из которых отвечает за различные стадии работы с нейросетями. Так, например, модуль nn обеспечивает конфигурирование нейросети (определению слоев, и их параметров), модуль optim содержит реализации различных методов оптимизации, применяемых для обучения, а gnuplot предоставляет возможность визуализации данных (построение графиков, показ изображений и т. д.). Установка дополнительных модулей позволяет расширить функционал библиотеки [29].

Caffe [33]

Библиотека Caffe для глубокого обучения разработана Yangqing Jia в процессе подготовки своей диссертации в университете Беркли в 2013 году. С указанного момента Caffe активно поддерживается Центром Зрения и Обучения Беркли и сообществом разработчиков на GitHub.

Caffe реализована с использованием языка программирования C++, имеются обертки на Python и MATLAB. Официально поддерживаемые операционные системы – Linux и OS X, также имеется неофициальный порт на Windows. Разработчики Caffe поддерживают возможности создания, обучения и тестирования полностью связанных и сверточных нейросетей. Для ускорения вычислений Caffe может быть запущена на GPU (система графических процессоров) с использованием базовых возможностей технологии CUDA или библиотеки примитивов глубокого обучения cuDNN.

Caffe – платформа, включающая в себя предустановленные наборы обучаемых нейронных сетей. Этот фреймворк известен своими возможностями обработки изображений, также в платформу была включена поддержка пакета прикладного ПО MATLAB.

Особенности:

- все модели имеют открытый исходный код;
- фреймворк обеспечивает высокую скорость и эффективность работы.

Достоинства:

- сопряжение и поддержка языков C++ и Python, поддержка CNN;
- фреймворк специализируется на решении различных вычислительных задач.

Недостаток: Caffe не способна обрабатывать комплексные массивы данных [29].

Произведем сравнительный анализ данных библиотек по важным для нас критериям оценки.

1 Язык программирования. Начиная работать с Deep Learning, хотелось бы иметь возможность использовать удобный язык программирования. Из всех рассмотренных библиотек возможность использовать C++ есть только в Caffe и TensorFlow. Во всех остальных случаях задача усложняется новым для освоения языком Python.

2 Руководство для изучения. Theano, TensorFlow, Torch обладают отлично написанными туториалами, материалами для самообучения, которые легко понять и использовать на практике. Степень вовлеченности GitHub-сообщества также является достоверным показателем не только будущего развития инструментария, но и того, с какой скоростью/вероятностью можно будет исправить ошибку в коде с помощью StackOverflow или Git Issues. По данным GitHub в ежегодном отчете Octoverse в 2017 году TensorFlow занимает первое место среди проектов с наибольшим количеством форков и пятое место среди проектов с наибольшим количеством контрибьюторов.

3 Возможности для моделирования CNN. Была проведена оценка фреймворков в зависимости от наличия определенных возможностей для моделирования CNN: доступность встроенных слоев нейронных сетей, наличие инструментов и функций для соединения этих слоев между собой. В результате проведенного анализа были выделены библиотеки TensorFlow и его InceptionV3, а также easy-to-use временной верстки CNN в Torch.

4 Возможности для моделирования RNN. Caffe имеет минимальное количество возможностей для RNN моделирования (рекуррентные нейронные сети), в то время как Torch обладает богатыми наборами документаций и встроенных моделей. TensorFlow также имеет некоторый набор материалов по RNN.

5 Архитектура и easy-to-use модульный пользовательский интерфейс. Для того чтобы создавать и обучать новые модели, фреймворк должен иметь легкий в использовании модульный пользовательский интерфейс. TensorFlow, Torch и MXNet, например, им обладают – создают интуитивно понятную среду для разработки. Для сравнения такие фреймворки, как Caffe, требуют значительных усилий для создания нового слоя сети. Выяснилось, что TensorFlow легко отлаживается не только во время, но и после обучения благодаря TensorBoard GUI.

6 Скорость. Библиотека Torch показала один из наилучших результатов при тестировании производительности сверточных нейронных сетей [1]. TensorFlow прошел испытания с сопоставимыми результатами, а Caffe и Theano

значительно отстали от лидеров. Авторы другого исследования, сравнивая Theano, Torch и TensorFlow, выбрали Theano победителем по части обучения RNN [2].

7 Поддержка множества GPU. Большинство алгоритмов Deep Learning требуют невероятного количества FLOPS (floating point operations). Для того чтобы уменьшить время создания модели, необходимо использовать различные GPU на различных системах. Все вышеуказанные библиотеки предоставляют такую возможность.

8 Совместимость с Keras. Keras – это высокоуровневая библиотека для быстрой реализации Deep Learning алгоритмов, которая отлично подходит для знакомства аналитиков с областью. Она представляет собой надстройку над фреймворками Deeplearning4j, TensorFlow и Theano. По состоянию на сентябрь 2016 года Keras являлась второй по скорости роста системой глубинного обучения после TensorFlow Google, и третьей по размеру после TensorFlow и Caffe.

Результаты анализа характеристик библиотек сведем для удобства в таблицу 4.1 и расставим места каждой библиотеке от первого до третьего [19, 29].

Таблица 4.1 – Результаты сравнения библиотек

Наименование библиотеки	Язык	Учебные материалы	CNN	RNN	Простота	Скорость	GPU	Keras
Theano	Python	++	++	++	+	++	+	+
TensorFlow	Python, C++	+++	+++	++	+++	++	++	+
Torch	Python, Lua	+	+++	++	++	+++	++	–
Caffe	C++	+	++	–	+	+	+	–

Исходя из данных таблицы наилучшие показатели по проанализированным характеристикам у библиотек TensorFlow и Torch. Выбирая между этими двумя библиотеками, было принято решение использовать в дальнейшем TensorFlow. В связи с удобством в написании приложений на языке C++ и большим количеством материалов для самообучения TensorFlow выбран в качестве программного средства для практических заданий, предлагаемых в данном пособии.

5 БАЗЫ ИЗОБРАЖЕНИЙ

База данных MNIST (Modified National Institute of Standards and Technology) [36] – объемная база данных образцов рукописного написания цифр. База данных является стандартом, предложенным Национальным институтом стандартов и технологий США с целью калибрации и сопоставления методов распознавания изображений с помощью машинного обучения прежде всего на основе нейронных сетей. Данные состоят из заранее подготовленных примеров изображений, на основе которых проводится обучение и тестирование систем. База данных была создана после переработки оригинального набора черно-белых образцов размером 20×20 пикселей MNIST. Создатели базы данных MNIST, в свою очередь, использовали набор образцов из Бюро переписи населения США, к которому были добавлены еще тестовые образцы, написанные студентами американских университетов. Образцы из набора MNIST были нормализированы, прошли сглаживание и приведены к серому полутоновому изображению размером 28×28 пикселей. База данных MNIST содержит 60 000 изображений для обучения и 10 000 изображений для тестирования. Половина образцов для обучения и тестирования была взята из набора MNIST для обучения, а другая половина – из набора MNIST для тестирования. Производились многочисленные попытки достичь минимальной ошибки после обучения по базе данных MNIST, которые обсуждались в научной литературе. Рекордные результаты указывались в публикациях, посвященных использованию сверточных нейронных сетей, уровень ошибки был доведен до 0,23 %. Примеры изображений из базы MNIST приведены на рисунке 5.1 [37].

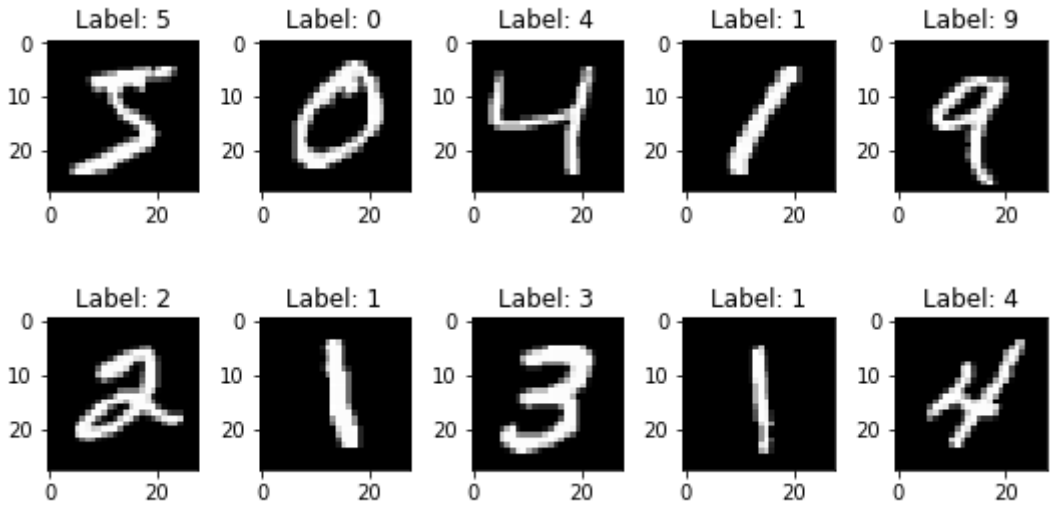


Рисунок 5.1 – Примеры изображений из базы MNIST

IAM Handwriting

База данных IAM Handwriting [38] содержит примеры рукописного текста на английском языке, которые можно использовать для обучения и тестирования систем распознавания рукописного текста, а также для проведения экспериментов по идентификации и проверке авторов.

На рисунке 5.2 представлены образцы полной формы, текстовая строка и несколько извлеченных из текста слов.

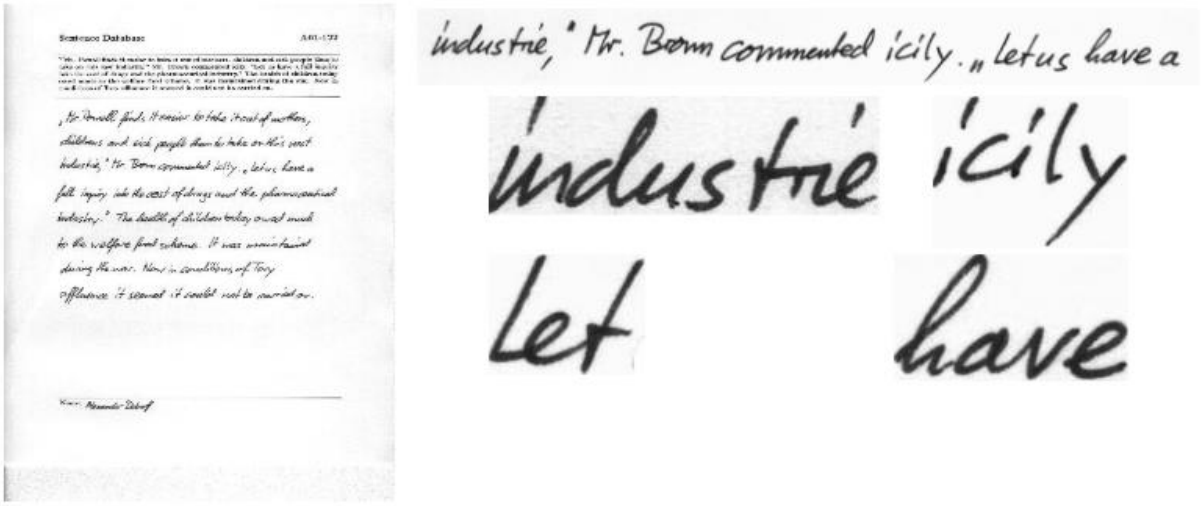


Рисунок 5.2 – Образцы полной формы, текстовая строка и некоторые извлеченные слова

Базы изображений MNIST и IAM Handwriting будет предложено использовать при выполнении лабораторных работ далее.

6 МЕТРИКИ ДЛЯ ОЦЕНКИ КАЧЕСТВА МОДЕЛИ

Матрица ошибок

Для описания метрик оценки качества модели классификации используется матрица ошибок (confusion matrix). Каждая строка матрицы представляет экземпляры в реальном классе, в то время как каждый столбец представляет экземпляры в прогнозируемом классе (или наоборот). С помощью матрицы легко увидеть, не путает ли система два класса [20]. На рисунке 6.1 представлен пример матрицы ошибок.

С помощью данной матрица можно вычислить следующие срабатывания:

- TP (True Positive) – истинно положительные;
- FP (False Positive) – ложноположительные;
- FN (False Negative) – ложноотрицательные;
- TN (True Negative) – истинно отрицательные.

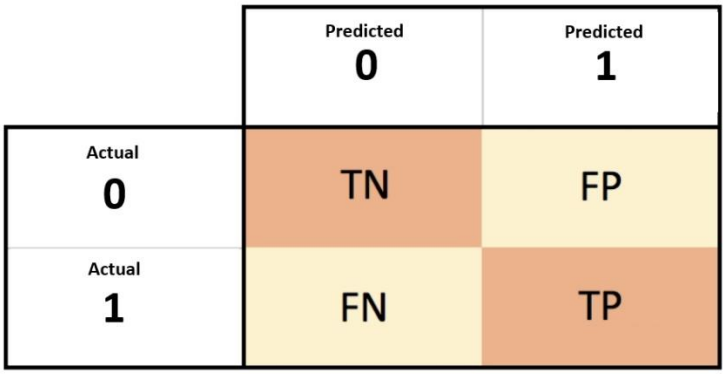


Рисунок 6.1 – Матрица ошибок

Точность и полнота

Precision (точность) можно интерпретировать как долю объектов, названных классификатором положительными и при этом действительно являющимися положительными, а recall (полнота) показывает, какую долю объектов положительного класса из всех объектов положительного класса нашел алгоритм. Точность и полнота рассчитываются по формулам

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}},$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}},$$

где TP – истинно положительные значения;

FP – ложноположительные значения;

FN – ложноотрицательные значения [57].

Intersection over union (IoU)

В контексте задачи детекции объектов на изображениях Intersection over union определяется отношением площади пересечения и области объединения прогнозируемой ограничивающей рамки и истинной ограничивающей рамки (ground truth). Считается, что детектор правильно определил объект, если IoU больше чем 0,5. Ложноположительный результат может быть, если IoU меньше чем 0,5 или произошло дублирование ограничивающей рамки. Ложноотрицательный результат возможен, если IoU больше чем 0,5, но класс объекта определен неверно [56]. На рисунке 6.2 представлены возможные варианты работы детектора.

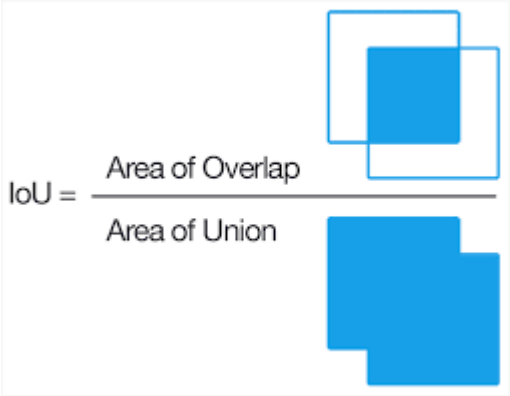


Рисунок 6.2 – Оценка точности детекции объекта

Метрика Jaccard Index

Для оценки решений сегментической сегментации часто используется метрика Jaccard Index. Метрика Jaccard Index задается формулой

$$J = \frac{TP}{TP + FP + FN} = \frac{A \cap B}{A \cup B} = \frac{A \cap B}{A + B - A \cap B},$$

где TP – True Positive;

FP – False Positive;

FN – False Negative.

Схематично формулу можно пояснить с помощью кругов Эйлера (рисунок 6.3).

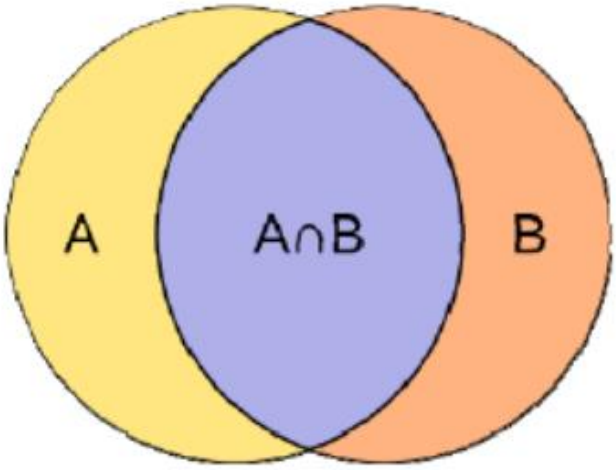


Рисунок 6.3 – Пересечение реального ядра клетки A с предсказанным ядром B

Из формулы видно, что Jaccard Index изменяется на промежутке от 0 до 1. При этом в задаче метрика считалась одновременно для всех картинок. Плохое предсказание на одной картинке значительно влияло на финальную метрику, поэтому требовалось избегать ложноположительных срабатываний [58].

Mean average precision (mAP)

Mean average precision – одна из наиболее часто используемых метрик качества ранжирования.

Для больших данных precision и recall перестают быть информативными. В этом случае для оценки качества лучше использовать Precision at k (P@k) и Recall at k (R@k) – метрики, применимые к первым наиболее подходящим k найденным результатам. В рамках задачи детекции объектов P@k – доля релевантных объектов среди первых объектов k из отранжированного списка найденных объектов R@k – доля релевантных объектов из списка топ-k найденных объектов среди всех релевантных объектов. Найденные объекты ранжируются по IoU.

Введем обозначения для следующих метрик: average precision at k (AP@k) average recall at k (AR@k). Тогда

$$\mathrm{AP@k} = \frac{1}{\mathrm{k}}\sum_{\mathrm{i}=1}^{\mathrm{k}} \mathrm{P_i@k},$$

$$\mathrm{AR@k} = \frac{1}{\mathrm{k}}\sum_{\mathrm{i}=1}^{\mathrm{k}} \mathrm{R_i@k},$$

где k – количество объектов, для которых рассчитываются данные метрики.

Average precision at k рассчитывается для каждого класса. Mean average precision (mAP) – это среднее значение AP для всех классов в данном детекторе [59].

Метрики COCO

Стандартные метрики COCO берут за основу mAP, но изменяют значение IoU и рассчитывают mAP отдельно для больших и маленьких объектов.

Average Precision (AP):

- AP с IoU от 0,5 до 0,95 с шагом 0,05;
- AP с IoU, равным 0,5;
- AP с IoU, равным 0,75.

Average Precision Across Scales:

- AP small – для объектов с площадью меньше 32^2 px;
- AP medium – для объектов с площадью от 32^2 до 96^2 px;
- AP large – для объектов с площадью больше 96^2 px.

Average Recall (AR):

- ARmax=1 AR – для 1 объекта на изображении;
- ARmax=10 AR – для 10 объектов на изображении;
- ARmax=100 для 100 объектов на изображении.

AR Across Scales:

- AR small – для объектов с площадью меньше 32^2 px;
- AR medium – для объектов с площадью от 32^2 до 96^2 px;
- AR large – для объектов с площадью больше 96^2 px [39].

7 МЕТОДОЛОГИЯ ПОСТРОЕНИЯ СИСТЕМ ИНТЕЛЛЕКТУАЛЬНОГО АНАЛИЗА ДАННЫХ

Проектирование систем для решения задач классификации изображений и распознавания объектов можно реализовывать в соответствии с методологией построения систем интеллектуального анализа данных CRISP-DM. CRoss Industry Standard Process for Data Mining (CRISP-DM) – стандарт, описывающий общие процессы и подходы к аналитике данных, используемые в промышленных data-mining проектах независимо от конкретной задачи и индустрии. CRISP-DM – наиболее распространенная и популярная методология ведения проектов интеллектуального анализа данных [60] (рисунок 7.1).

Рисунок 7.1 – Диаграммы ключевых фаз CRISP-DM

Необходимость использования методологии

Проекты интеллектуального анализа данных должны:

- надежно исполняться испытанными средствами с предсказуемыми результатом (Reliable);

- быть повторяемыми, особенно людьми с малым опытом в анализе данных (Repeatable).

Следование методике позволяет:

- эффективно использовать средства для сохранения опыта проектов, а накопленный опыт позволяет успешно повторять проекты;

- упростить процессы планирования и управления проектами, т. к. последовательность действий известна и привычна, а также имеется набор необходимых артефактов;

- достаточно просто осуществлять включение в работу новых членов команды, уменьшить взаимозависимость [60].

Основные этапы

Методология CRISP-DM состоит из шести основных этапов. Каждый из этих этапов в свою очередь делится на задачи. На выходе каждой задачи должен получаться определенный результат. Перечень этапов, а также списки основных задач приведены в таблице 7.1.

Таблица 7.1 – Списки основных задач каждого этапа CRISP-DM

Business Understanding/ Бизнес-анализ	Data Understanding/ Анализ данных	Data Preparation/ Подготовка данных	Modeling/ Моделирование	Evaluation/ Оценка решения	Deployment/ Внедрение
Determine Business Objectives/Определение бизнес-целей Assess Situation/Оценка текущей ситуации Determine Data Mining Goals/Определение целей аналитики Product Project Plan/ Подготовка плана проекта	Collecting Initial Data/Сбор данных Describe Data/Описание данных Explore Data/ Изучение данных Verify Data Quality/Проверка качества данных	Select Data/ Выборка данных Clean Data/ Очистка данных Construct Data/Генерация данных Integrate Data/Интеграция данных Format Data/ Форматирование данных	Select Modeling Techniques/Выбор алгоритмов Generate Test/Подготовка плана тестирования Build Model/ Обучение моделей Assess Model/ Оценка качества моделей	Evaluate Results/Оценка результатов Review Process/Оценка процесса Determine Next Steps/ Определение следующих шагов	Plan Deployment/Внедрение Plan Monitoring and Maintenance/ Планирование мониторинга и поддержки Produce Final Report/Подготовка отчета Review Project/Ревью проекта

Понимание бизнеса (Business Understanding)

Первая фаза процесса направлена на определение целей проекта и требований со стороны бизнеса. Затем эти знания конвертируются в постановку задачи интеллектуального анализа данных и предварительный план достижения целей проекта.

Задачи этапа:

- определить бизнес-цели;
- оценить ситуацию;
- определить цели анализа данных;
- составить план проекта.

Понимание данных (Data Understanding)

Вторая фаза начинается со сбора данных и ставит целью познакомиться с данными как можно ближе. Для этого необходимо выявить проблемы с качеством данных (такие как ошибки или пропуски), понять что за данные имеются в наличии, попробовать отыскать интересные наборы данных или сформировать гипотезы о наличии скрытых закономерностей в данных.

Задачи этапа:

- собрать исходные данные;
- описать данные;
- исследовать данные;
- проверить качество данных.

Подготовка данных (Data Preparation)

Фаза подготовки данных ставит целью получить итоговый набор данных, которые будут использоваться при моделировании, из исходных разнородных и разноформатных данных. Задачи подготовки данных могут выполняться много раз без какого-либо наперед заданного порядка. Они включают в себя отбор таблиц, записей и атрибутов, а также конвертацию и очистку данных для моделирования.

Задачи этапа:

- отобрать данные;
- очистить данные;
- сделать производные данные;
- объединить данные;
- привести данные в нужный формат.

Моделирование (Modeling)

В этой фазе к данным применяются разнообразные методики моделирования, строятся модели и их параметры настраиваются на оптимальные значения. Обычно для решения любой задачи анализа данных существует несколько различных подходов. Некоторые подходы накладывают особые требования на представление данных. Таким образом, часто бывает нужен возврат на шаг назад к фазе подготовки данных.

Задачи этапа:

- выбрать методику моделирования;
- сделать тесты для модели;
- построить модель;
- оценить модель.

Оценка (Evaluation)

На этом этапе проекта уже построена модель и получены количественные оценки ее качества. Перед тем как внедрять эту модель, необходимо убедиться, что мы достигли всех поставленных бизнес-целей. Основной целью этапа является поиск важных бизнес-задач, которым не было уделено должного внимания.

Задачи этапа:

- оценить результаты;
- сделать ревью процесса;
- определить следующие шаги.

Развертывание (Deployment)

В зависимости от требований фаза развертывания может быть простой, например, составление финального отчета или сложной (например, автоматизация процесса анализа данных для решения бизнес-задач). Обычно развертывание – это забота клиента. Однако даже если аналитик не принимает участие в развертывании, важно дать понять клиенту, что ему нужно сделать для того, чтобы начать использовать полученные модели.

Задачи этапа:

- запланировать развертывание;
- запланировать поддержку и мониторинг развернутого решения;
- сделать финальный отчет;
- сделать ревью проекта [60].

8 ПРАКТИЧЕСКАЯ ЧАСТЬ

8.1 Знакомство с TensorFlow и TensorBoard

Цель работы:

1) визуализировать процесс обучения и построить граф вычислений средствами TensorBoard;

2) рассмотреть преимущества TensorFlow, понятия «сессия», «граф» и «тензор», значение метода градиентного спуска и функции потерь при обучении;

3) реализовать сохранение полученной модели, а также отобразить полученные веса.

Задания:

1 Реализовать простейшее математическое выражение средствами TensorFlow (например y = ax + b); построить граф в TensorBoard.

2 Реализовать линейную или логистическую регрессию средствами TensorFlow.

Исходные данные: искомая функция, коэффициенты которой необходимо будет найти:

а) $2 \cdot x - 3 = y$;

б) $1/(1 + np.exp(- (x \cdot k + b))) = y$.

Сгенерировать данные, добавить к ним шум, далее, используя метод градиентного спуска, получить коэффициенты.

Построить граф в TensorBoard, отобразить сгенерированные данные, исходную и полученную функцию.

3 Обучить простейший нейрон средствами TensorFlow таким образом, чтобы на выходе получалась логическая единица независимо от значения на входе. Построить граф в TensorBoard, отобразить график в TensorBoard обучения, т. е. вывести функцию потерь.

Рекомендуемые источники: [61, 62].

8.2 Решение задачи классификации на основе глубоких сетей

Цель работы: решить задачу классификации изображений на основе глубоких нейронных сетей.

Задание: реализовать и обучить нейронную сеть – многослойный перцептрон – для классификации рукописных цифр.

Исходные данные: база данных изображений рукописных цифр MNIST.

Образцы данной базы нормализованы, сглажены и приведены к полутоновым изображениям. Размер изображений 28×28 пикселей.

Технические задания:

1 Реализовать нейросетевой классификатор с использованием библиотека TensorFlow, языка программирования Python (желательно).

2 Использовать функции активации: ReLU для скрытых слоев и Softmax для выходного слоя.

3 В качестве функции потерь использовать перекрестную энтропию.

4 Использовать оптимизатор AdamOptimizer.

Рекомендуемые источники: [10, 63].

8.3 Решение задачи автоматической обработки текстов на основе глубоких сетей

Цель работы: реализовать и обучить рекуррентную нейронную сеть для распознавания рукописного текста.

Задание: реализовать и обучить RNN+CNN для распознавания рукописного текста.

Исходные данные: база данных изображений рукописного текста IAM Handwriting Database.

Технические задания:

1 Решить задачу автоматической обработки текстов с использованием библиотек TensorFlow, OpenCV (желательно), языка программирования Python (желательно).

2 Использовать функции активации: ReLU для сверточных слоев и CTC для выходного слоя.

3 В качестве функции потерь использовать CTC.

4 Использовать оптимизатор RMSPropOptimizer.

Рекомендуемые источники: [10, 64, 65, 66].

8.4 Решение задачи шумоподавления с помощью автокодировщика

Цель работы: реализовать и обучить автокодировщик для решения задачи шумоподавления.

Задание: реализовать шумоподавляющий автокодировщик, продемонстрировать его способность к шумоподавлению.

Исходные данные: база данных изображений рукописных цифр MNIST.

Технические задания:

1 Решить задачу автоматической обработки текстов с использованием библиотеки TensorFlow.

2 Реализовать автокодер, состоящий из двух частей (рисунок 8.1):

а) **энкодер** – отвечает за сжатие входа в latent-space, представлен функцией кодирования h = f (x);

б) **декодер** – предназначен для восстановления ввода из latent-space, представлен функцией декодирования h = f (x).

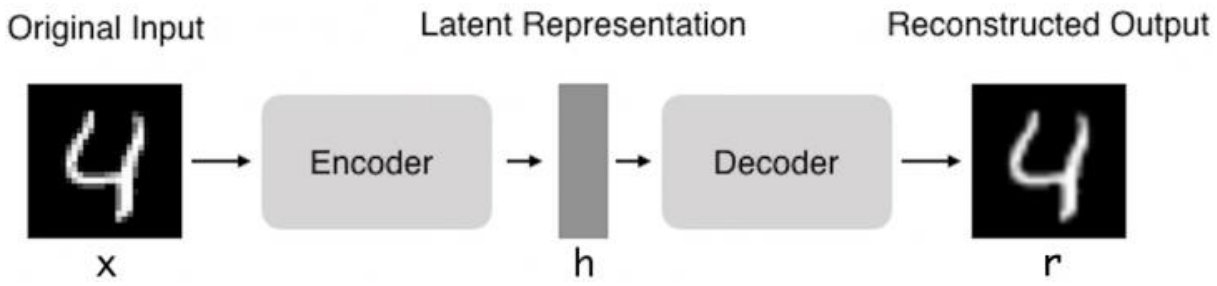


Рисунок 8.1 – Условная схема автокодера

Результат работы представлен на рисунке 8.2.

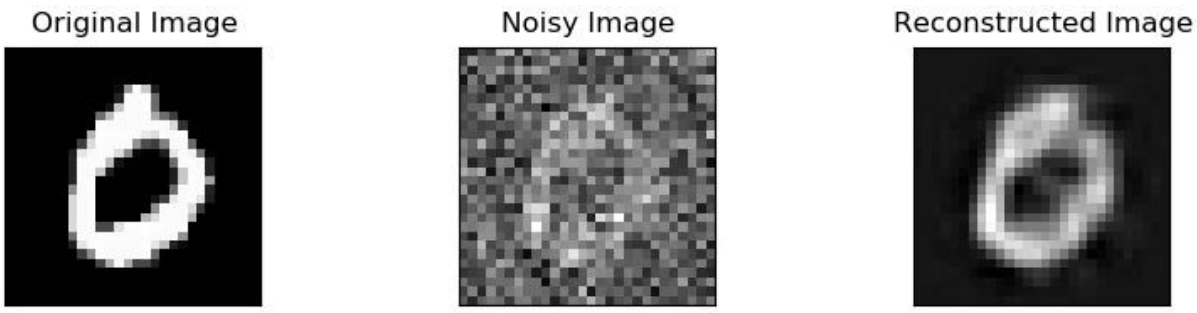


Рисунок 8.2 – Результаты работы автокодера

Рекомендуемые источники: [61, 67, 68].

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Grant View Reserch [Электронный ресурс]. – Режим доступа : https://www.grandviewresearch.com/industry-analysis/artificial-intelligence-in-agriculture-market. – Дата доступа : 17.12.2019.

2 Grant View Reserch [Электронный ресурс]. – Режим доступа : https://www.grandviewresearch.com/press-release/global-deep-learning-market. – Дата доступа : 23.11.2019.

3 Сайт компании Stfalcon [Электронный ресурс]. – Режим доступа : https://stfalcon.com/ru/blog/post/5-fascinating-applications-of-deep-learning. – Дата доступа : 23.11.2019.

4 Сикорский, О. С. Обзор сверточных нейронных сетей для задачи классификации изображений / О. С. Сикорский // Новые информационные технологии в автоматизированных системах. – 2017. – № 20. – С. 37–42.

5 Машинное обучение [Электронный ресурс]. – Режим доступа : http://www.machinelearning.ru/wiki/index.php?title=%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5. – Дата доступа : 23.11.2019.

6 Journal «Proceedings of TUSUR» [Электронный ресурс]. – Режим доступа : https://journal.tusur.ru/storage/47679/074.pdf?1467885811. – Дата доступа : 23.11.2019.

7 MathWorks - Makers of MATLAB and Simulink - MATLAB & Simulink [Электронный ресурс]. – Режим доступа : https://www.mathworks.com. – Дата доступа : 23.11.2019.

8 Yann LeCun's Home [Электронный ресурс]. – Режим доступа : http://yann.lecun.com/exdb/publis/pdf/lecun-89e.pdf. – Дата доступа : 23.11.2019.

9 Гудфеллоу, Я. Глубокое обучение / Я. Гудфеллоу, И. Бенджио. – М. : ДМК-пресс, 2017. – 652 с.

10 Николенко, С. И. Глубокое обучение / С. И. Николенко, А. А. Кадурин, Е. О. Архангельская. – СПб. : Питер, 2018. – 481 с.

11 Головко, В. А. Нейронные сети: обучение, организация и применение : учеб. пособие для вузов / В. А. Головко. – М. : ИПРЖР, 2001. – 256 с.

12 PaddlePaddle [Электронный ресурс]. – Режим доступа : http://www.paddlepaddle.org/docs/develop/book/03.image_classification/index.html. – Дата доступа : 23.11.2019.

13 CS231n Convolutional Neural Networks for Visual Recognition [Электронный ресурс]. – Режим доступа : http://cs231n.github.io/convolutional-networks. – Дата доступа : 23.11.2019.

14 Обзор сверточных нейронных сетей для задачи классификации изображений [Электронный ресурс]. – Режим доступа : https://cyberleninka.ru/article/n/obzor-svyortochnyh-neyronnyh-setey-dlya-zadachi-klassifikatsii-izobrazheniy. – Дата доступа : 23.11.2019.

15 GitHub [Электронный ресурс]. – Режим доступа :https://github.com/pjreddie/darknet/wiki/YOLO:-Real-Time-Object-Detection. – Дата доступа : 23.11.2019.

16 GitHub [Электронный ресурс]. – Режим доступа : https://github.com/rbgirshick/rcnn. – Дата доступа : 23.11.2019.

17 GitHub [Электронный ресурс]. – Режим доступа : https://github.com/rbgirshick/fast-rcnn. – Дата доступа : 23.11.2019.

18 GitHub [Электронный ресурс]. – Режим доступа : https://github.com/rbgirshick/py-faster-rcnn. – Дата доступа : 23.11.2019.

19 Быстрый старт: обзор основных Deep Learning фреймворков [Электронный ресурс]. – Режим доступа : http://ai-news.ru/2017/04/bystryj_start_obzor_osnovnyh_deep_learning_frejmvorkov.html. – Дата доступа : 23.11.2019.

20 Confusion matrix – Wikipedia [Электронный ресурс]. – Режим доступа : https://en.wikipedia.org/wiki/Confusion_matrix. – Дата доступа : 23.11.2019.

21 Журнал «Компьютерная Оптика» [Электронный ресурс]. – Режим доступа : http://www.computeroptics.smr.ru/KO/PDF/KO14-15_1/05.pdf. – Дата доступа : 23.11.2019.

22 Selective Search for Object Recognition [Электронный ресурс]. – Режим доступа : http://www.huppelen.nl/publications/selectiveSearchDraft.pdf. – Дата доступа : 23.11.2019.

23 Electronic Proceedings of the Neural Information Processing Systems Conference [Электронный ресурс]. – Режим доступа : http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf. – Дата доступа : 23.11.2019.

24 Proceedings of Machine Learning Research [Электронный ресурс]. – Режим доступа : http://proceedings.mlr.press/v15/glorot11a/glorot11a.pdf. – Дата доступа : 23.11.2019.

25 GitHub [Электронный ресурс]. – Режим доступа : https://github.com/ajinkya933/Computer_Vision?files=1. – Дата доступа : 23.11.2019.

26 Deep Residual Learning for Image Recognition [Электронный ресурс]. – Режим доступа : https://arxiv.org/pdf/1512.03385.pdf. – Дата доступа : 23.11.2019.

27 Stanford Artificial Intelligence Laboratory [Электронный ресурс]. – Режим доступа : https://ai.stanford.edu/~amaas/papers/relu_hybrid_icml2013_final.pdf. – Дата доступа : 23.11.2019.

28 Gradient-Based Learning Applied to Documents Recognition [Электронный ресурс]. – Режим доступа : http://www.dengfanxin.cn/wp-content/uploads/2016/03/1998Lecun.pdf. – Дата доступа : 23.11.2019.

29 Топ-9 фреймворков в мире искусственного интеллекта. Часть 1 [Электронный ресурс]. – Режим доступа : https://medium.com/nuances-of-programming/топ-9-фреймворков-в-мире-искусственного-интеллекта-часть-1-40015cfb98dd. – Дата доступа : 23.11.2019.

30 Хайкин, С. Нейронные сети. Полный курс / С. Хайкин. – М. : Изд. дом «Вильямс», 2006. – 1104 с.

31 Осовский, С. Нейронные сети для обработки информации / С. Осовский. – М. : Финансы и статистика, 2002. – 344 с.

32 Головко, В. А. Нейроинтеллект: теория и применения. Книга 1 : Организация и обучение нейронных сетей с прямыми и обратными связями / В. А. Головко. – Брест : Изд. БПИ, 1999. – 264 с.

33 Caffe | Deep Learning Framework [Электронный ресурс]. – Режим доступа : http://caffe.berkeleyvision.org. – Дата доступа : 23.11.2019.

34 TensorFlow [Электронный ресурс]. – Режим доступа : https://www.tensorflow.org. – Дата доступа : 23.11.2019.

35 Theano · GitHub [Электронный ресурс]. – Режим доступа : https://github.com/Theano. – Дата доступа : 23.11.2019.

36 MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges [Электронный ресурс]. – Режим доступа : http://yann.lecun.com/exdb/mnist. – Дата доступа : 23.11.2019.

37 MNIST (база данных) – Википедия [Электронный ресурс]. – Режим доступа : https://ru.wikipedia.org/wiki/MNIST_(%D0%B1%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85). – Дата доступа : 23.11.2019.

38 IAM Handwriting Database – Computer Vision and Artificial Intelligence [Электронный ресурс]. – Режим доступа : http://www.fki.inf.unibe.ch/databases/iam-handwriting-database. – Дата доступа : 23.11.2019.

39 COCO – Common Objects in Context [Электронный ресурс]. – Режим доступа : http://cocodataset.org/#detection-eval. – Дата доступа : 23.11.2019.

40 arXiv.org e-Print archive [Электронный ресурс]. – Режим доступа : https://arxiv.org/pdf/1505.04597.pdf. – Дата доступа : 23.11.2019.

41 Badrinarayanan, V. Segnet: A deepconvolutional encoder-decoder architecture for image segmentation / V. Badrinarayanan, A. Kendall, R. Cipolla // IEEE transactions on pattern analysis and machine intelligence. – 2017. – Vol. 39, No. 12. – P. 2481–2495.

42 Введение в архитектуры нейронных сетей / Конференции Олега Бунина (Онтико) corporate blog / Habr [Электронный ресурс]. – Режим доступа : https://habr.com/en/company/oleg-bunin/blog/340184. – Дата доступа : 23.11.2019.

43 Understanding LSTM Networks – colah's blog [Электронный ресурс]. – Режим доступа : http://colah.github.io/posts/2015-08-Understanding-LSTMs. – Дата доступа : 23.11.2019.

44 Сверточная нейронная сеть, часть 1: структура, топология, функции активации и обучающее множество / Хабр [Электронный ресурс]. – Электронные данны – Режим доступа : https://habr.com/ru/post/348000. – Дата доступа : 23.11.2019.

45 arXiv.org e-Print archive [Электронный ресурс]. – Режим доступа : https://arxiv.org/abs/1311.2901. – Дата доступа : 23.11.2019.

46 arXiv.org e-Print archive [Электронный ресурс]. – Режим доступа : https://arxiv.org/abs/1409.4842. – Дата доступа : 23.11.2019.

47 Simonyan, K. Very deep convolution-al networks for large-scale image recognition / K. Simonyan, A. Zisserman // CoRR, 2014. – N. 1409.1556. – 14 c.

48 Howard, A. G. MobileNets: Efficient Con-volutional Neural Networks for Mobile Vision Applications / A. G. Howard [et al.] // CoRR, 2017. – N. 1704.04861. – 9 p.

49 arXiv.org e-Print archive [Электронный ресурс]. – Режим доступа : https://arxiv.org/pdf/1606.00915.pdf. – Дата доступа : 23.11.2019.

50 arXiv.org e-Print archive [Электронный ресурс]. – Режим доступа : https://arxiv.org/abs/1611.06612. – Дата доступа : 23.11.2019.

51 arXiv.org e-Print archive [Электронный ресурс]. – Режим доступа : https://arxiv.org/pdf/1612.01105.pdf. – Дата доступа : 23.11.2019.

52 [Электронный ресурс]. – Режим доступа : https://arxiv.org/pdf/1703.02719.pdf. – Дата доступа : 23.11.2019.

53 arXiv.org e-Print archive [Электронный ресурс]. – Режим доступа : https://arxiv.org/abs/1706.05587. – Дата доступа : 23.11.2019.

54 arXiv.org e-Print archive [Электронный ресурс]. – Режим доступа : https://arxiv.org/pdf/1603.07285.pdf. – Дата доступа : 23.11.2019..

55 Понимание семантической сегментации с помощью UNET – Машинное Обучение–2020 [Электронный ресурс]. – Режим доступа : https://ru.sciencewal.com/28944-understanding-semantic-segmentation-with-unet-6be4f42d4b47-42. – Дата доступа : 23.11.2019.

56 Intersection over Union (IoU) for object detection – PyImageSearch [Электронный ресурс]. – Режим доступа : https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection. – Дата доступа : 23.11.2019.

57 Оценка классификатора (точность, полнота, F-мера) [Электронный ресурс]. – Режим доступа : http://bazhenov.me/blog/2012/07/21/classification-performance-evaluation.html. – Дата доступа : 23.11.2019.

58 Intersection over Union (IoU) for object detection – PyImageSearch [Электронный ресурс]. – Режим доступа : https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection. – Дата доступа : 23.11.2019.

59 Метрики качества ранжирования / E-Contenta corporate blog / Habr [Электронный ресурс]. – Режим доступа : https://habr.com/en/company/econtenta/blog/303458. – Дата доступа : 23.11.2019.

60 IBM SPSS Modeler CRISP-DM Guide – CRISP_DM.pdf [Электронный ресурс]. – Режим доступа : ftp://ftp.software.ibm.com/software/analytics/spss/documentation/modeler/14.2/en/CRISP_DM.pdf. – Дата доступа: 23.11.2019.

61 GitHub – tensorflow/tensorflow: An Open Source Machine Learning Framework for Everyone [Электронный ресурс]. – Режим доступа : https://github.com/tensorflow/tensorflow. – Дата доступа : 23.11.2019.

62 Hello, TensorFlow. Библиотека машинного обучения от Google / Habr [Электронный ресурс]. – Режим доступа : https://habr.com/post/305578. – Дата доступа : 23.11.2019.

63 A Step by Step Backpropagation Example – Matt Mazur [Электронный ресурс]. – Электронные данные. – Режим доступа : https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example. – Дата доступа : 23.11.2019.

64 Сверточная сеть на python. Часть 1. Определение основных параметров модели / Open Data Science corporate blog / Habr [Электронный ресурс]. – Режим доступа : https://habr.com/en/company/ods/blog/344008. – Дата доступа : 23.11.2019.

65 Введение в архитектуры нейронных сетей / Конференции Олега Бунина (Онтико) corporate blog / Habr [Электронный ресурс]. – Режим доступа : https://habr.com/company/oleg-bunin/blog/340184. – Дата доступа : 23.11.2019.

66 An Intuitive Explanation of Connectionist Temporal Classification [Электронный ресурс]. – Режим доступа : https://towardsdatascience.com/intuitively-understanding-connectionist-temporal-classification-3797e43a86c. – Дата доступа : 23.11.2019.

67 Department of Computer Science, University of Toronto [Электронный ресурс]. – Режим доступа : https://www.cs.toronto.edu/~hinton/science.pdf. – Дата доступа : 23.11.2019.

68 Deep Learning [Электронный ресурс]. – Режим доступа : http://www.deeplearningbook.org/contents/autoencoders.html. – Дата доступа : 23.11.2019.