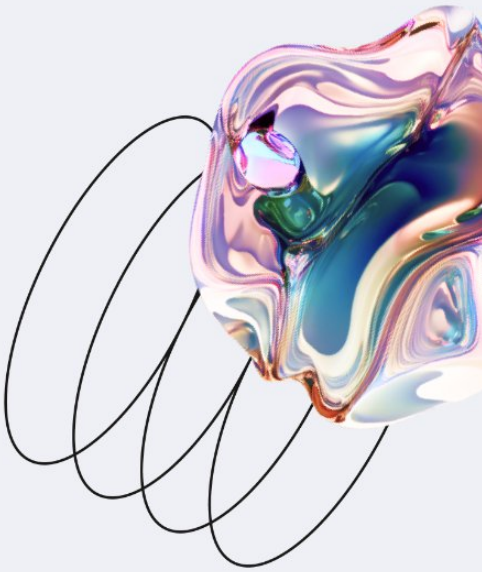
GeekBrains



Сверточные нейронные сети

Архитектура нейронных сетей

Оглавление

Введение

Всем привет! Это наша третья лекция на курсе архитектура нейронных сетей и сегодня мы с вами поговорим о сверточных нейронных сетях. Эти сети лежат в основе большинства систем компьютерного зрения и используют механизмы глубокого обучения. Давайте разбираться как же искусственный интеллект может видеть и воспринимать графическую информацию.

За последние несколько десятилетий глубокое обучение оказалось очень мощным инструментом благодаря его способности обрабатывать большие объемы данных. Интерес к использованию скрытых слоев превзошел традиционные методы, особенно в распознавании образов. Одной из самых популярных глубоких нейронных сетей являются сверточные нейронные сети (также известные как CNN или ConvNet) в глубоком обучении, особенно когда речь идет о приложениях компьютерного зрения.

С 1950-х годов, на заре ИИ, исследователи изо всех сил пытались создать систему, которая могла бы понимать визуальные данные. В последующие годы эта область стала известна как Компьютерное зрение. В 2012 году компьютерное зрение совершило квантовый скачок, когда группа исследователей из Университета Торонто разработала модель искусственного интеллекта, которая превзошла лучшие алгоритмы распознавания изображений, причем с большим отрывом.

Система искусственного интеллекта, получившая название AlexNet (названная в честь ее главного создателя Алекса Крижевского), выиграла конкурс компьютерного зрения ImageNet 2012 года с поразительной точностью в 85 процентов. Занявший второе место набрал по тесту скромные 74 процента.

В основе AlexNet лежали сверточные нейронные сети — особый тип нейронной сети, примерно имитирующий человеческое зрение. С годами CNN стали очень важной частью многих приложений компьютерного зрения и, следовательно, частью любого онлайн-курса по компьютерному зрению. Давайте посмотрим на-то как они устроены и как их применять.

Словарь терминов

Сверточная нейронная сеть — класс Нейронных сетей (Neural Network), который специализируется на обработке данных, имеющих топологию в виде сетки, например изображений
Цифровое изображение – это двоичное представление визуальных данных
Функция активации — это нелинейная математическая функция, которая применяется к выходу нейрона или нейронной сети после выполнения линейных операций (например, умножения на веса и суммирования входов)

Сверточные нейронные сети

Введение в CNN

Ян ЛеКун, директор группы исследований искусственного интеллекта Facebook , является пионером в области сверточных нейронных сетей. В 1988 году он построил первую сверточную нейронную сеть под названием LeNet. LeNet использовалась для задач распознавания символов, таких как чтение почтовых индексов и цифр.

Задумывались ли вы когда-нибудь, как распознавание лиц работает в социальных сетях, или как обнаружение объектов помогает в создании беспилотных автомобилей , или как выявление заболеваний осуществляется с помощью визуальных изображений в здравоохранении? Все это возможно благодаря сверточным нейронным сетям (CNN). Вот пример сверточных нейронных сетей, который иллюстрирует, как они работают:

Представьте, что есть изображение птицы, и вы хотите определить, действительно ли это птица или какой-то другой объект. Первое, что вы делаете, — это передаете пиксели изображения в виде массивов на входной слой нейронной сети (многослойные сети, используемые для классификации вещей). Скрытые слои выполняют извлечение объектов, выполняя различные вычисления и манипуляции. Существует несколько скрытых слоев, таких как слой свертки, слой ReLU и слой пула, которые выполняют извлечение объектов из изображения. Наконец, есть полностью связанный слой, который идентифицирует объект на изображении.

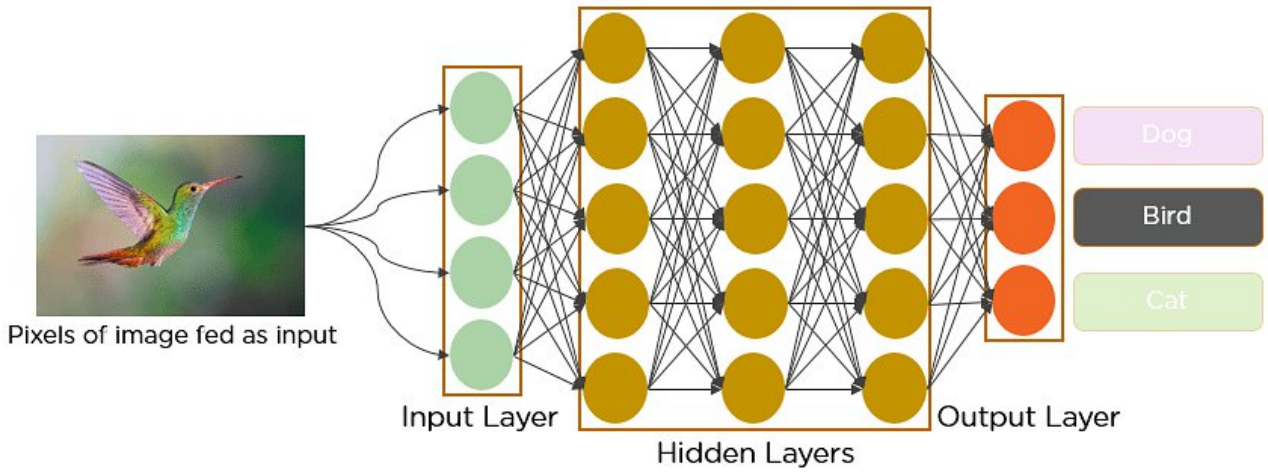
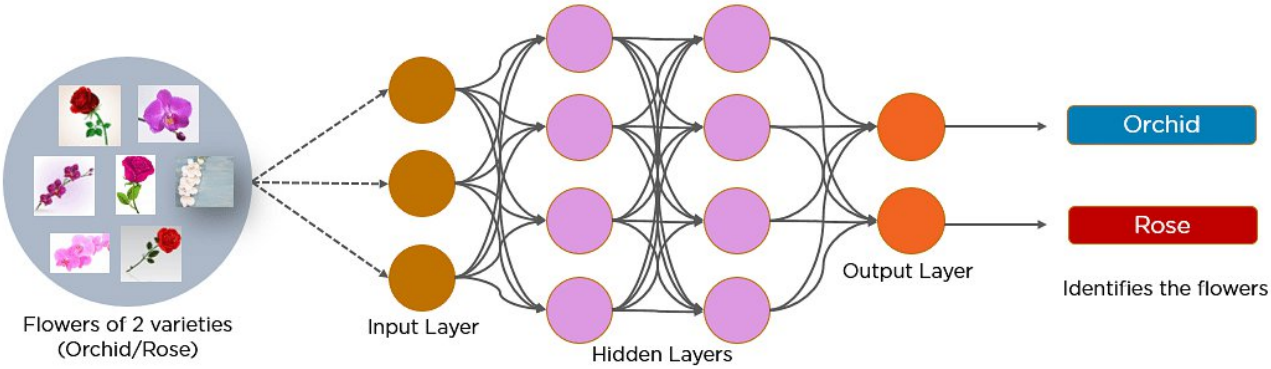


Рис. Сверточная нейронная сеть для идентификации изображения птицы.

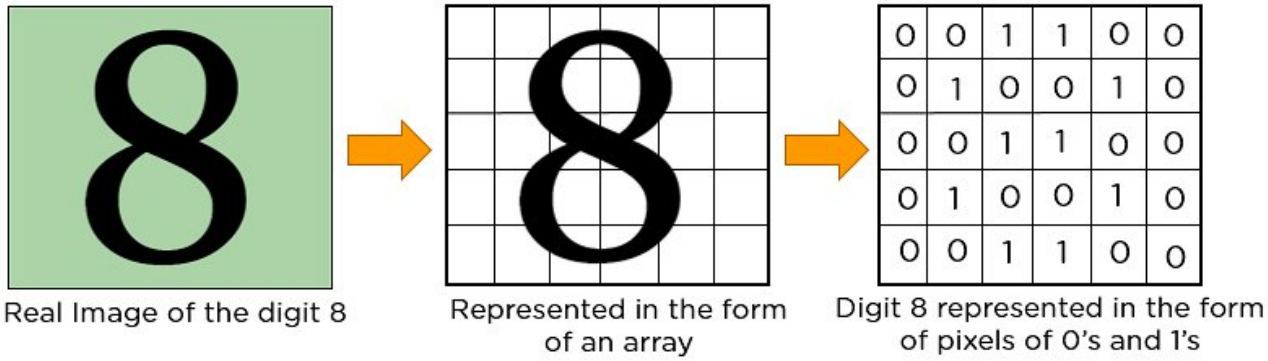
Что такое сверточная нейронная сеть?

Сверточная нейронная сеть(CNN) — это нейронная сеть прямого распространения, которая обычно используется для анализа визуальных изображений путем обработки данных с топологией, напоминающей сетку. Он также известен как ConvNet. Сверточная нейронная сеть используется для обнаружения и классификации объектов на изображении.

Ниже представлена нейронная сеть , которая идентифицирует два типа цветов: орхидею и розу.



В CNN каждое изображение представлено в виде массива значений пикселей.



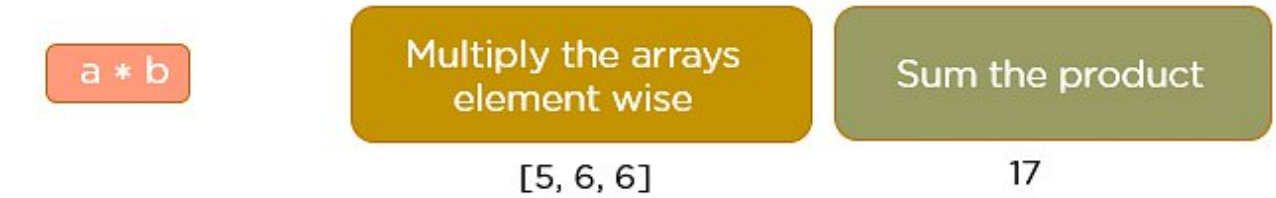
Операция свертки составляет основу любой сверточной нейронной сети. Давайте разберемся в операции свертки, используя две одномерные матрицы a и b.

a = [5,3,7,5,9,7]

b = [1,2,3]

В операции свертки массивы умножаются поэлементно, а произведение суммируется для создания нового массива, который представляет a*b.

Первые три элемента матрицы a умножаются на элементы матрицы b. Произведение суммируется для получения результата.

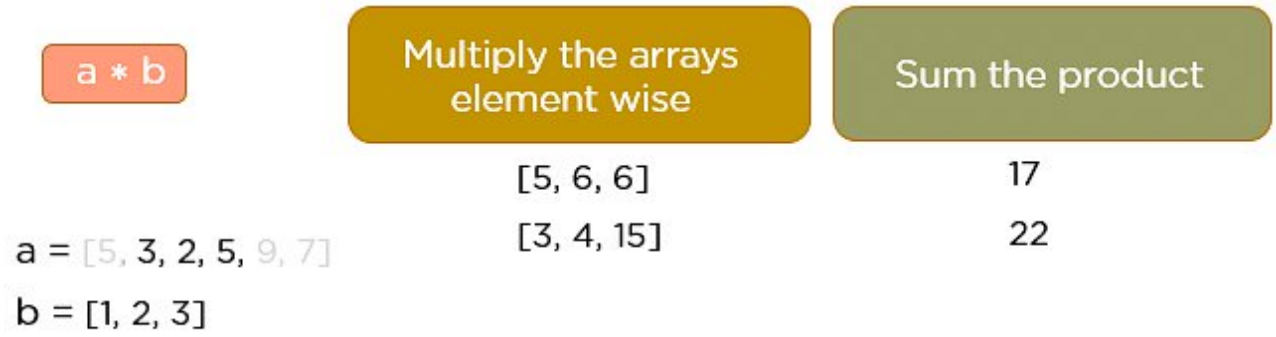


a = [5, 3, 2, 5, 9, 7]

b = [1, 2, 3]

a * b = [17,]

Следующие три элемента матрицы a умножаются на элементы матрицы b и произведение суммируется.



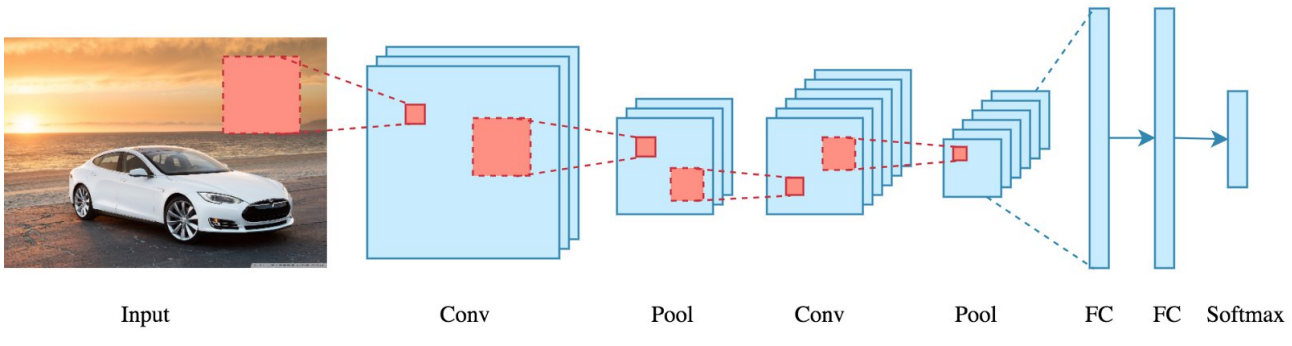
a = [5, 3, 2, 5, 9, 7]

b = [1, 2, 3]

a * b = [17, 22]

Этот процесс продолжается до тех пор, пока операция свертки не завершится.

В математике **свертка** — это математическая операция над двумя функциями, в результате которой получается третья функция, выражающая, как форма одной изменяется другой.

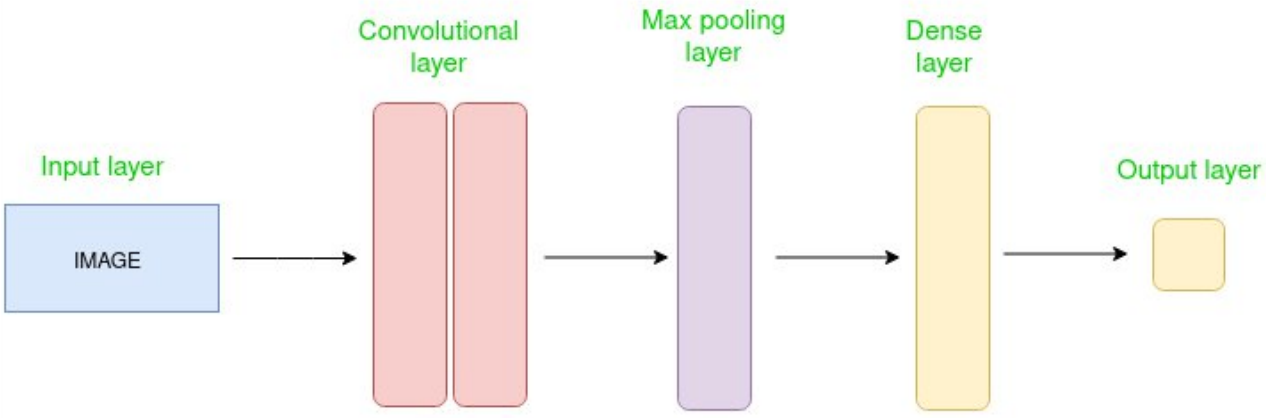


Но на самом деле нам не нужно углубляться в математическую часть, чтобы понять, что такое CNN и как она работает.

Суть в том, что роль сверточной сети состоит в том, чтобы привести изображения к форме, которую легче обрабатывать, не теряя при этом особенностей, которые имеют решающее значение для получения хорошего прогноза.

Архитектура сверточной нейронной сети.

Сверточная нейронная сеть состоит из нескольких слоев, таких как входной слой, сверточный слой, слой пула и полностью связанные слои.



Простая архитектура CNN

Сверточный слой применяет фильтры к входному изображению для извлечения признаков, слой объединения снижает дискретизацию изображения для сокращения вычислений, а полностью связный слой делает окончательный прогноз. Сеть изучает оптимальные фильтры посредством обратного распространения ошибки и градиентного спуска.

Слои, используемые для построения ConvNets

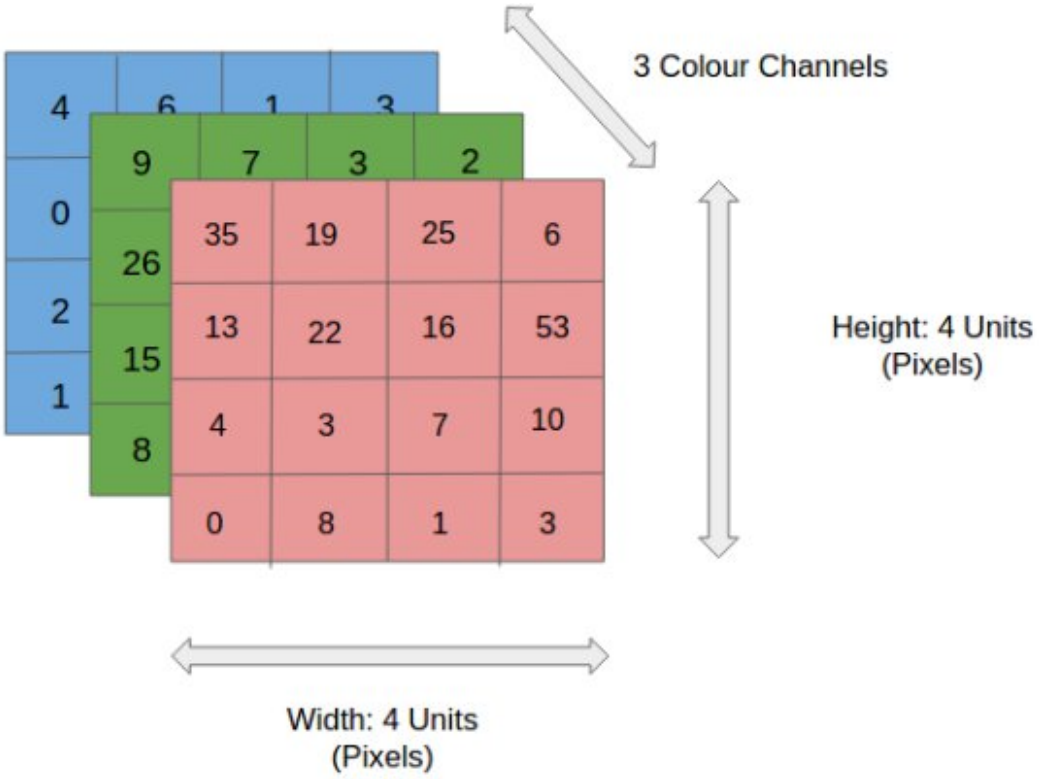
Полная архитектура сверточных нейронных сетей также известна как covnets. Ковнеты — это последовательность слоев, каждый из которых преобразует один объем в другой с помощью дифференцируемой функции.
Давайте возьмем пример, запустив сетку изображения размером 32 x 32 x 3.

- **Входные слои:** это слой, в котором мы вводим входные данные в нашу модель. В CNN, как правило, входными данными будет изображение или последовательность изображений. Этот слой содержит необработанные входные данные изображения шириной 32, высотой 32 и глубиной 3.
- **Сверточные слои:** это слой, который используется для извлечения объекта из входного набора данных. Он применяет к входным изображениям набор обучаемых фильтров, известных как ядра. Фильтры/ядра представляют собой матрицы меньшего размера, обычно формы 2×2, 3×3 или 5×5. он скользит по данным входного изображения и вычисляет скалярное произведение между весом ядра и соответствующим патчем входного изображения. Результатом этого слоя являются карты объектов объявлений. Предположим, мы используем в общей сложности 12 фильтров для этого слоя и получим выходной объем размером 32 x 32 x 12.

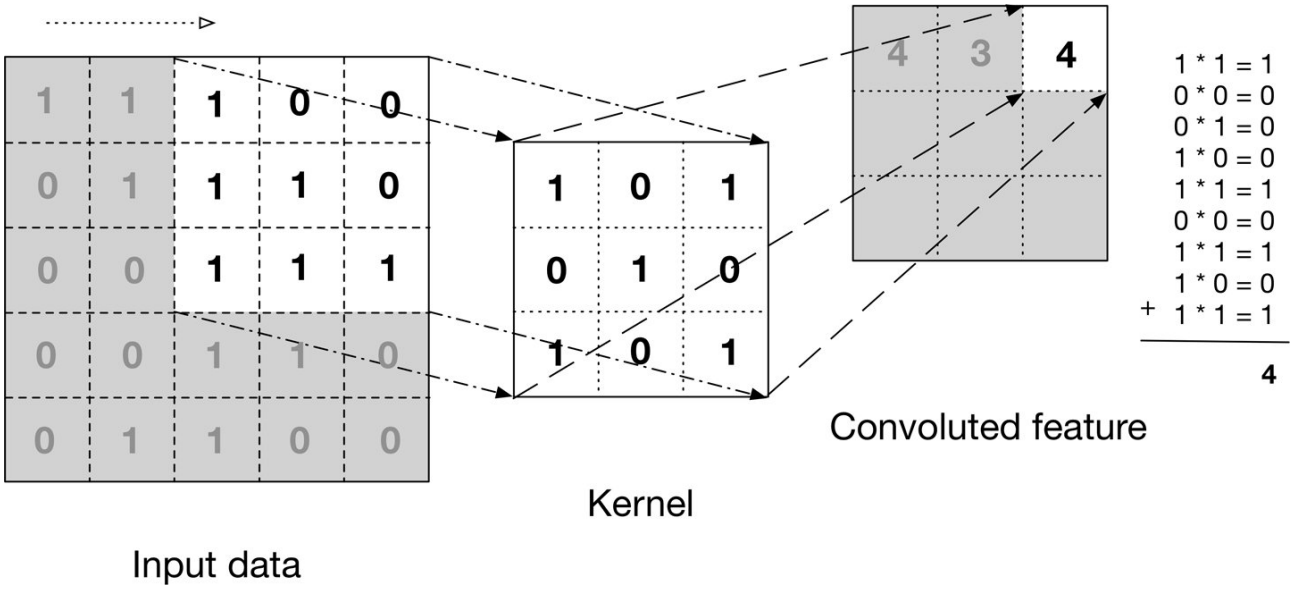
- **Слой активации:** добавляя функцию активации к выходным данным предыдущего слоя, слои активации добавляют нелинейность в сеть. он применит поэлементную функцию активации к выходным данным слоя свертки. Некоторые распространенные функции активации: **RELU** : max(0, x), **Tanh** , **Leaky RELU** и т. д. Объем остается неизменным, поэтому выходной объем будет иметь размеры 32 x 32 x 12.
- **Слой пула:** этот слой периодически вставляется в косети, и его основная функция — уменьшить размер тома, что ускоряет вычисления, уменьшает объем памяти, а также предотвращает переобучение. Двумя распространенными типами слоев пула являются **максимальный пул** и **средний пул** . Если мы используем максимальный пул с фильтрами 2 x 2 и шагом 2, результирующий объем будет иметь размер 16x16x12.
- **Сглаживание:** полученные карты объектов сводятся в одномерный вектор после слоев свертки и объединения, поэтому их можно передать в полностью связанный слой для категоризации или регрессии.
- **Полностью связанные слои:** он принимает входные данные из предыдущего слоя и вычисляет окончательную задачу классификации или регрессии.
- **Выходной уровень:** выходные данные полностью связанных слоев затем передаются в логистическую функцию для задач классификации, таких как сигмоид или softmax, которая преобразует выходные данные каждого класса в оценку вероятности каждого класса.

Как это работает?

Прежде чем мы перейдем к работе CNN, давайте рассмотрим основы, например, что такое изображение и как оно представлено. Изображение RGB — это не что иное, как матрица значений пикселей, имеющая три плоскости, тогда как изображение в оттенках серого такое же, но имеет одну плоскость. Взгляните на это изображение, чтобы понять больше.



Для простоты давайте будем использовать изображения в оттенках серого, пытаясь понять, как работают CNN.



На изображении показано, что такое свертка. Мы берем фильтр/ядро (матрица 3×3) и применяем его к входному изображению, чтобы получить свернутый признак. Эта свернутая функция передается на следующий слой.

Сверточные нейронные сети состоят из нескольких слоев искусственных нейронов. Искусственные нейроны, грубая имитация своих биологических аналогов, представляют собой математические функции, которые вычисляют взвешенную сумму нескольких входных данных и выводят значение активации. Когда вы вводите изображение в ConvNet, каждый уровень генерирует несколько функций активации, которые передаются на следующий уровень.

Первый слой обычно извлекает основные элементы, такие как горизонтальные или диагональные края. Эти выходные данные передаются на следующий слой, который обнаруживает более сложные элементы, такие как углы или комбинационные края. По мере того, как мы продвигаемся глубже в сеть, она может идентифицировать еще более сложные функции, такие как объекты, лица и т. д.

На основе карты активации финального слоя свертки слой классификации выводит набор показателей достоверности (значения от 0 до 1), которые определяют, насколько вероятно, что изображение принадлежит к «классу». Например, если у вас есть ConvNet, который обнаруживает кошек, собак и лошадей, выходные данные последнего слоя — это вероятность того, что входное изображение содержит любое из этих животных.

Слой свертки

Слой свертки является основным строительным блоком CNN. На него ложится основная часть вычислительной нагрузки сети.

Этот уровень выполняет скалярное произведение двух матриц, где одна матрица представляет собой набор обучаемых параметров, также известных как ядро, а

другая матрица представляет собой ограниченную часть рецептивного поля. Ядро пространственно меньше изображения, но оно более глубокое. Это означает, что если изображение состоит из трех (RGB) каналов, высота и ширина ядра будут пространственно малы, но глубина распространяется на все три канала.

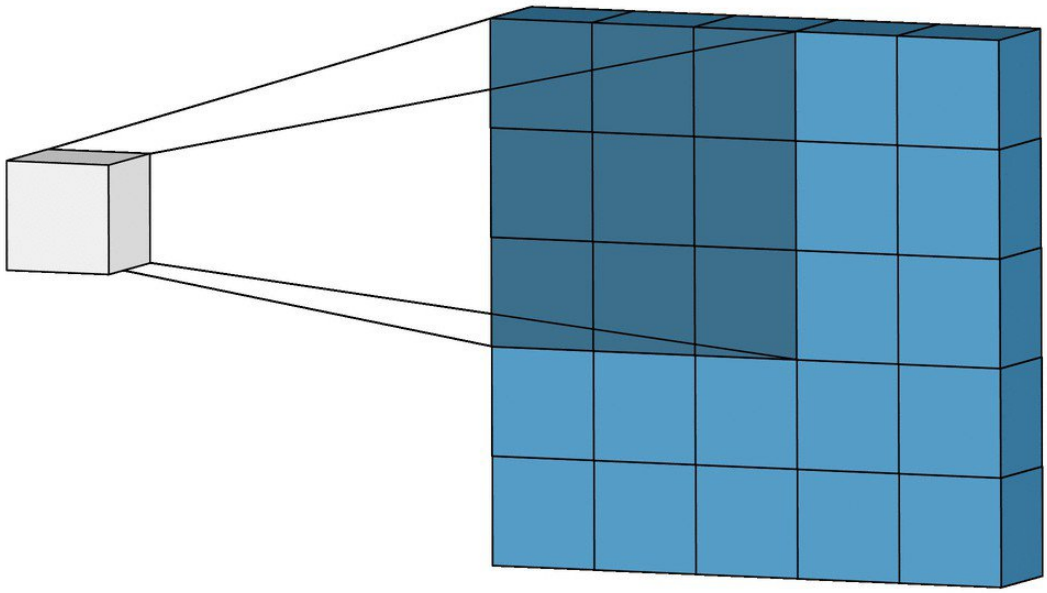


Иллюстрация операции свертки (источник)

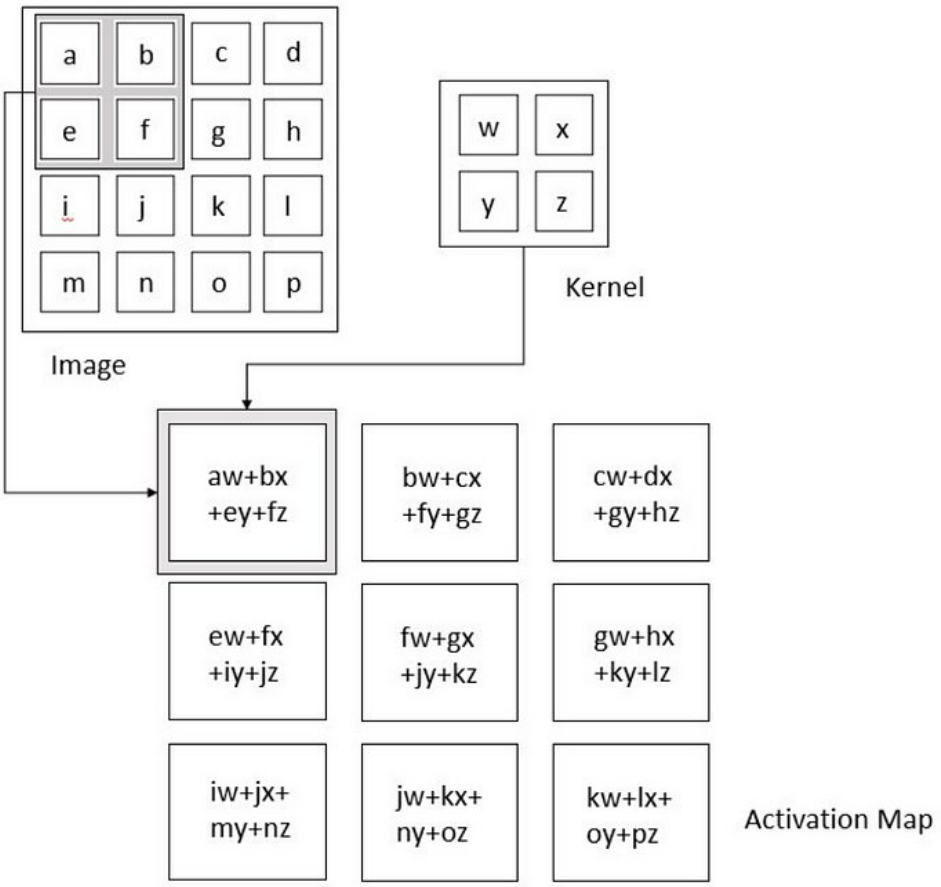
Во время прямого прохода ядро скользит по высоте и ширине изображения, создавая представление изображения этой рецептивной области. Это создает двумерное представление изображения, известное как карта активации, которая дает ответ ядра в каждом пространственном положении изображения. Скользящий размер ядра называется шагом.

Если у нас есть входные данные размером W x W x D и количество ядер Dout с пространственным размером F с шагом S и количеством заполнения P, то размер выходного объема можно определить по следующей формуле:

$$W_{out} = \frac{W - F + 2P}{S} + 1$$

Формула для слоя свертки

Это даст выходной объем размером *Wout* x *Wou* t x *Dout* .



Операция свертки (Источник: глубокое обучение Яна Гудфеллоу, Йошуа Бенджио и Аарона Курвиля)

Мотивация свертки

Свертка использует три важные идеи, которые мотивировали исследователей компьютерного зрения: разреженное взаимодействие, совместное использование параметров и эквивариантное представление. Опишем каждый из них подробно.

Тривиальные слои нейронной сети используют умножение матрицы на матрицу параметров, описывающих взаимодействие между блоком ввода и вывода. Это означает, что каждый выходной блок взаимодействует с каждым входным блоком. Однако сверточные нейронные сети имеют *редкое взаимодействие.* Это достигается за счет уменьшения размера ядра по сравнению с входными данными. Например, изображение может иметь миллионы или тысячи пикселей, но при его обработке с использованием ядра мы можем обнаружить значимую информацию, состоящую из десятков или сотен пикселей. Это означает, что нам нужно хранить меньше параметров, что не только снижает требования к памяти модели, но и повышает статистическую эффективность модели.

Если вычисление одного признака в пространственной точке (x1, y1) полезно, то оно также должно быть полезно и в какой-то другой пространственной точке, скажем (x2, y2). Это означает, что для одного двумерного среза, то есть для создания одной карты активации, нейроны вынуждены использовать один и тот же набор весов. В традиционной нейронной сети каждый элемент весовой матрицы используется один

раз, а затем никогда не используется повторно, в то время как сеть свертки имеет *общие параметры* , т. е. для получения выходных данных веса, применяемые к одному входу, такие же, как веса, применяемые в другом месте.

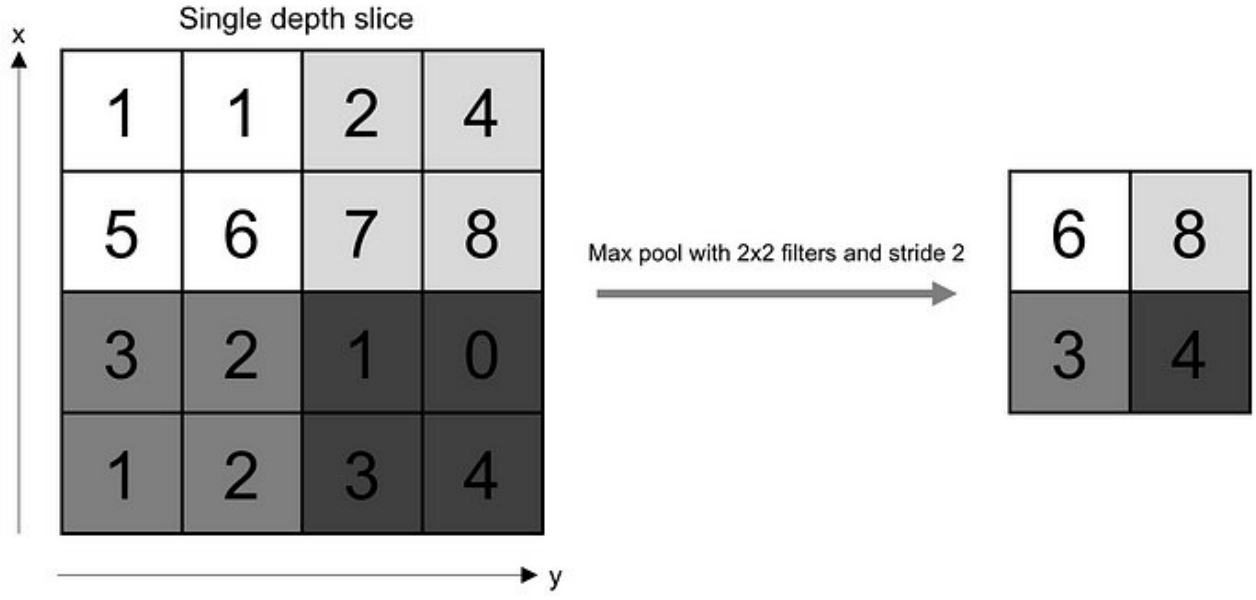
Благодаря совместному использованию параметров слои сверточной нейронной сети будут обладать свойством *эквивалентности трансляции* . В нем говорится, что если мы каким-то образом изменили входные данные, выходные данные также изменятся таким же образом.

Слой пула

Подобно сверточному слою, слой объединения отвечает за уменьшение пространственного размера свернутого объекта. Это делается для **уменьшения вычислительной мощности, необходимой для обработки данных,** за счет уменьшения размеров.

Уровень объединения заменяет выходные данные сети в определенных местах, получая сводную статистику близлежащих выходных данных. Это помогает уменьшить пространственный размер представления, что уменьшает требуемый объем вычислений и веса. Операция объединения обрабатывается для каждого фрагмента представления индивидуально.

Существует несколько функций объединения, таких как среднее значение прямоугольной окрестности, норма L2 прямоугольной окрестности и средневзвешенное значение, основанное на расстоянии от центрального пикселя. Однако наиболее популярным процессом является максимальное объединение, которое сообщает о максимальном выходе из окрестности.



Операция объединения (Источник: O'Reilly Media)

Если у нас есть карта активации размером W x W x D , ядро пула пространственного размера F и шаг S , то размер выходного объема можно определить по следующей формуле:

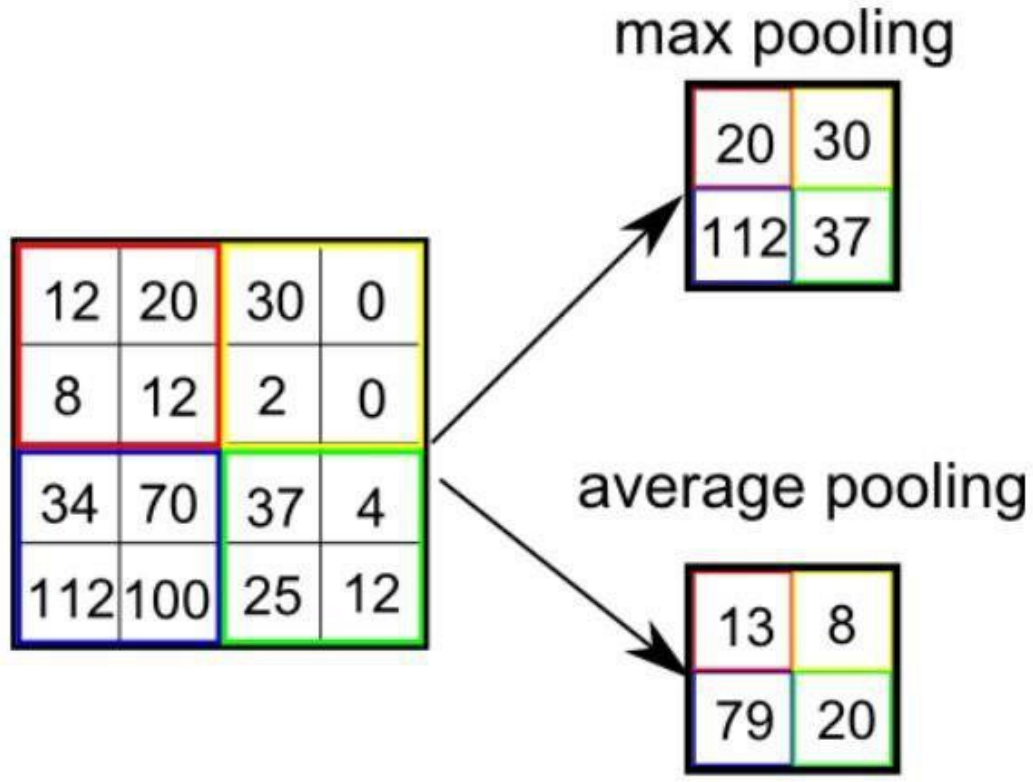
$$W_{out} = \frac{W - F}{S} + 1$$

Формула для слоя заполнения

Это даст выходной объем размером Wout x Wout x D.

Во всех случаях объединение обеспечивает некоторую трансляционную инвариантность, что означает, что объект будеТ узнаваем независимо от того, где он появляется в кадре.
Существует два типа пула: средний пул и максимальный пул. Итак, что мы делаем в Max Pooling, так это находим максимальное значение пикселя из части изображения, покрытой ядром. Max Pooling также выполняет функцию **шумоподавления** . Он полностью отбрасывает шумные активации, а также выполняет шумоподавление и уменьшение размерности.

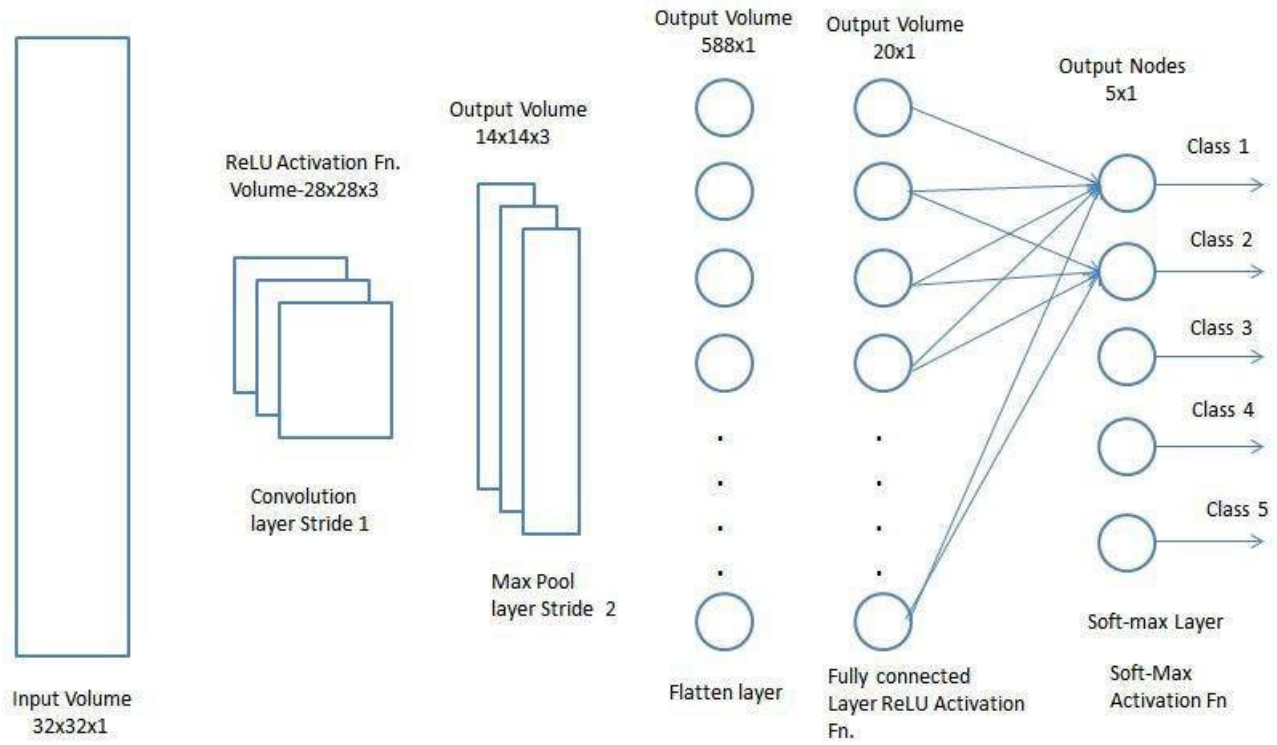
С другой стороны, **метод Average Pooling** возвращает **среднее значение всех значений** из части изображения, покрытой ядром. Среднее объединение просто выполняет уменьшение размерности в качестве механизма подавления шума. Следовательно, мы можем сказать, что **Max Pooling работает намного лучше, чем Average Pooling** .



Полностью связный слой

Нейроны этого слоя имеют полную связь со всеми нейронами предыдущего и последующего слоя, как это видно в обычном FCNN. Вот почему его можно вычислить, как обычно, путем умножения матриц с последующим эффектом смещения.

Полностью связанный слой(FC Layer) помогает сопоставить представление между входом и выходом.



Добавление FC Layer — это (обычно) дешевый способ изучить нелинейные комбинации высокоуровневых признаков, представленных выходными данными сверточного слоя. FC Layer изучает возможную нелинейную функцию в этом пространстве.

Теперь, когда мы преобразовали наше входное изображение в подходящий формат, мы сведем изображение в вектор-столбец. Сглаженный вывод подается в нейронную сеть с прямой связью, и затем к каждой итерации обучения применяется обратное распространение. После серии эпох (epochs) модель способна различать доминирующие и некоторые низкоуровневые признаки изображений и классифицировать их с помощью **техники Softmax**.

Слои нелинейности

Поскольку свертка — это линейная операция, а изображения далеки от линейных, слои нелинейности (также эти слои называются функцией активации, мы с вами про них уже говорили) часто размещаются непосредственно после сверточного слоя, чтобы внести нелинейность в карту активации.

Существует несколько типов нелинейных операций, наиболее популярными из которых являются:

1. Сигмовидная

Сигмовидная нелинейность имеет математическую форму σ(к) = 1/(1+ек). Он принимает действительное число и «сжимает» его в диапазон от 0 до 1.
Однако очень нежелательным свойством сигмовидной является то, что при активации на любом хвосте градиент становится почти нулевым. Если локальный градиент становится очень маленьким, то при обратном распространении он эффективно «убьет» градиент. Кроме того, если данные, поступающие в нейрон, всегда положительны, то на выходе сигмоиды будут либо все положительные, либо все отрицательные значения, что приведет к зигзагообразной динамике обновлений градиента веса.

2. Tanh

Тан сжимает действительное число в диапазон [-1, 1]. Как и сигмовидная, активация насыщает, но, в отличие от сигмовидных нейронов, ее выходной сигнал центрирован по нулю.

3. ReLU

Выпрямленный линейный блок (ReLU) стал очень популярным в последние несколько лет. Он вычисляет функцию f(к)=max (0,к). Другими словами, активация просто пороговая при нуле.
По сравнению с сигмовидной и танской ReLU более надежен и ускоряет сходимость в шесть раз.
К сожалению, недостатком является то, что ReLU может быть хрупким во время обучения. Большой градиент, проходящий через него, может обновить его таким образом, что нейрон никогда не будет обновляться дальше. Однако мы можем справиться с этим, установив правильную скорость обучения.

Примеры приложений, использующих CNN

Ниже приведены некоторые приложения сверточных нейронных сетей, используемые сегодня:
1. Обнаружение объектов. Благодаря CNN у нас теперь есть сложные модели, такие как R-CNN , Fast R-CNN и Faster R-CNN , которые являются преобладающим конвейером для многих моделей обнаружения объектов, используемых в автономных транспортных средствах, обнаружения лиц и многого другого.
2. Семантическая сегментация. В 2015 году группа исследователей из Гонконга разработала сеть глубокого анализа на базе CNN, позволяющую включать обширную информацию в модель сегментации изображений. Исследователи из Калифорнийского университета в Беркли также создали полностью сверточные сети , которые улучшили современную семантическую сегментацию.
3. Подписи к изображениям. CNN используются с рекуррентными нейронными сетями для написания подписей к изображениям и видео. Это можно использовать во многих приложениях, таких как распознавание активности или описание видео и изображений для людей с ослабленным зрением. YouTube активно использует его, чтобы придать смысл огромному количеству видео, регулярно загружаемых на платформу.

Ограничения CNN

Несмотря на мощь и сложность ресурсов CNN, они дают глубокие результаты. В основе всего этого лежит просто распознавание закономерностей и деталей, которые настолько малы и незаметны, что остаются незамеченными для человеческого

глаза. Но когда дело доходит до **понимания** содержания изображения, это терпит неудачу.

Давайте посмотрим на этот пример. Когда мы передаем изображение ниже в CNN, он обнаруживает человека около 30 лет и ребенка, вероятно, около 10 лет. Но когда мы смотрим на одно и то же изображение, мы начинаем думать о нескольких разных сценариях. Может быть, это день отца и сына, пикник или, может быть, они в походе. Может быть, это школьная площадка, и ребенок забил гол, и его отец счастлив, поэтому поднимает его.

Эти ограничения более чем очевидны, когда дело доходит до практического применения. Например, CNN широко использовались для модерации контента в социальных сетях. Но, несмотря на огромные ресурсы изображений и видео, на которых они обучались, он до сих пор не способен полностью заблокировать и удалить нежелательный контент. Как выяснилось, на Фейсбуке была отмечена **30-тысячелетняя статуя с обнаженной натурой** .

Несколько исследований показали, что CNN, обученные на ImageNet и других популярных наборах данных, не могут обнаруживать объекты, когда видят их в разных условиях освещения и под новыми углами.

Означает ли это, что CNN бесполезны? Однако, несмотря на ограничения сверточных нейронных сетей, нельзя отрицать, что они произвели революцию в искусственном интеллекте. Сегодня CNN используются во многих **приложениях компьютерного зрения,** таких как распознавание лиц, поиск и редактирование изображений, дополненная реальность и многое другое. Как показывают достижения в области ConvNets, наши достижения замечательны и полезны, но мы все еще очень далеки от **воспроизведения ключевых компонентов человеческого интеллекта .**

Что такое фреймворк TensorFlow?

Google разработал TensorFlow в ноябре 2015 года. Они определяют его как платформу машинного обучения с открытым исходным кодом, доступную каждому, по нескольким причинам.

- **Открытый исходный код** : выпущен под лицензией с открытым исходным кодом Apache 2.0. Это позволяет исследователям, организациям и разработчикам вносить свой вклад в библиотеку, опираясь на нее без каких-либо ограничений.
- **Фреймворк машинного обучения** : это означает, что он имеет набор библиотек и инструментов, которые поддерживают процесс построения моделей машинного обучения.
- **Для всех** : использование TensorFlow упрощает реализацию моделей машинного обучения с помощью распространенных языков программирования, таких как Python. Более того, встроенные библиотеки, такие как Keras, еще больше упрощают создание надежных моделей глубокого обучения.

Все эти функции делают Tensorflow хорошим кандидатом для создания нейронных сетей.

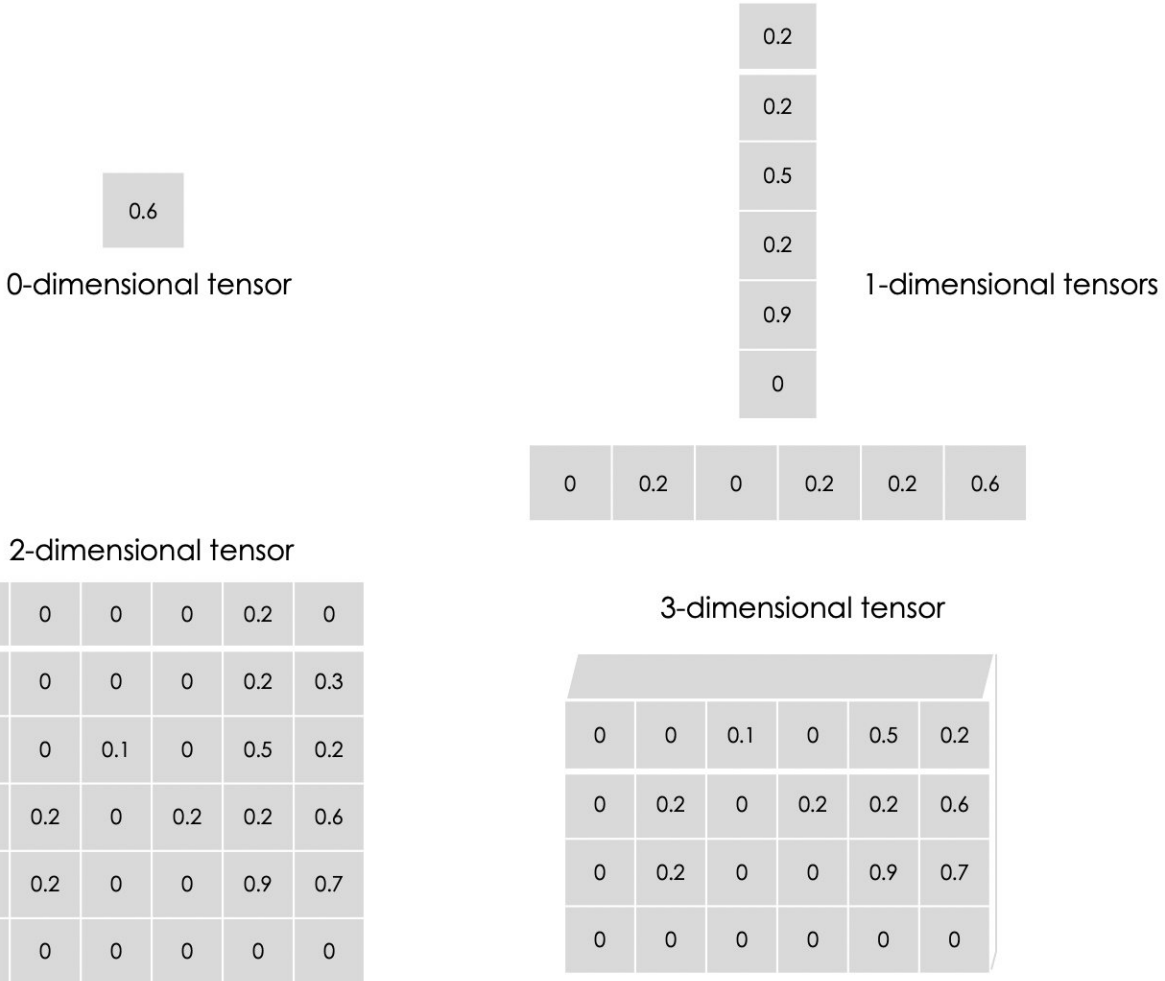
Кроме того, установка Tensorflow 2 проста и может быть выполнена следующим образом с помощью ***pip*** менеджера пакетов Python , как описано в официальной документации .

Теперь давайте подробнее рассмотрим основные компоненты для создания этих сетей.

Что такое тензоры?

В основном мы имеем дело с многомерными данными при построении моделей машинного и глубокого обучения. Тензоры — это многомерные массивы единого типа, используемые для представления различных характеристик данных.

Ниже приведено графическое представление различных типов размерностей тензоров.



- 0-мерный тензор содержит одно значение.
- Одномерный тензор, также известный как тензор «ранга 1», представляет собой список значений.
- Двумерный тензор — это тензор «ранга 2».
- Наконец, мы можем иметь N-мерный тензор, где N представляет количество измерений внутри тензора. В предыдущих случаях N равно соответственно 0, 1 и 2.

Ниже приведена иллюстрация нуля в трехмерном тензоре. Каждый тензор создается с помощью функции Constant() из TensorFlow.

```
# Zero dimensional tensor
zero_dim_tensor = tf.constant(20)
print(zero_dim_tensor)

# One dimensional tensor
one_dim_tensor = tf.constant([12, 20, 53, 26, 11, 56])
print(one_dim_tensor)

# Two dimensional tensor
two_dim_array = [[3, 6, 7, 5],
                 [9, 2, 3, 4],
                 [7, 1, 10,6],
                 [0, 8, 11,2]]

two_dim_tensor = tf.constant(two_dim_array)
print(two_dim_tensor)
```

Успешное выполнение предыдущего кода должно сгенерировать приведенные ниже выходные данные, и мы можем заметить, что ключевое слово «tf.Tensor» означает, что результатом является тензор. Он имеет три параметра:

- Фактическое значение тензора.
- shape() тензора, который равен 0, 6 на 1 и 4 на 4 соответственно для первого, второго и третьего тензоров.
- Тип данных, представленный атрибутом **dtype** , и все тензоры — int32.

```
tf.Tensor(20, shape=(), dtype=int32)
tf.Tensor([12 20 53 26 11 56], shape=(6,), dtype=int32)
tf.Tensor(
[[ 3  6  7  5]
 [ 9  2  3  4]
 [ 7  1 10  6]
 [ 0  8 11  2]], shape=(4, 4), dtype=int32)
```

Тензоры против матриц: различия

Многие путают тензоры с матрицами. Несмотря на то, что эти два объекта внешне похожи, они обладают совершенно разными свойствами. Этот раздел дает лучшее понимание разницы между матрицами и тензорами.

- Мы можем думать о матрице как о тензоре только с двумя измерениями.

- С другой стороны, тензоры — это более общий формат, который может иметь любое количество измерений.

В отличие от матриц, тензоры больше подходят для задач глубокого обучения по следующим причинам:

- Они могут работать с любым количеством измерений, что делает их более подходящими для многомерных данных.
- Способность тензоров быть совместимыми с широким спектром типов, форм и размеров данных делает их более универсальными, чем матрицы.
- Tensorflow обеспечивает поддержку графических процессоров и TPU для ускорения вычислений. Используя тензоры, инженеры по машинному обучению могут автоматически воспользоваться этими преимуществами.
- Тензоры изначально поддерживают трансляцию, заключающуюся в выполнении арифметических операций между тензорами разной формы, что не всегда возможно при работе с матрицами.

TensorFlow: константы, переменные и заполнители

Константы — не единственные типы тензоров. Существуют также переменные и заполнители, которые являются строительными блоками вычислительного графа.

Вычислительный граф — это, по сути, представление последовательности операций и потока данных между ними.

Теперь давайте поймем разницу между этими типами тензоров.

Константы

Константы — это тензоры, значения которых не изменяются в ходе выполнения вычислительного графа. Они создаются с помощью функции **tf.constant()** и в основном используются для хранения фиксированных параметров, которые не требуют каких-либо изменений во время обучения модели.

Переменные

Переменные — это тензоры, значение которых можно изменить во время выполнения вычислительного графа, и они создаются с помощью функции **tf.Variable()** . Например, в случае нейронных сетей веса и смещения могут быть определены как переменные, поскольку их необходимо обновлять в процессе обучения.

Плейсхолдеры

Они использовались в первой версии Tensorflow как пустые контейнеры, не имеющие конкретных значений. Они просто используются для заполнения места для данных, которые будут использоваться в будущем. Это дает пользователям свободу использовать различных наборов данных и размеров бэтчей во время обучения и проверки модели.

В версии 2 Tensorflow плейсхолдеоы были заменены функцией **tf.function()** , которая представляет собой более Pythonic и динамичный подход к загрузке данных в вычислительный граф.

Заключение

На этой лекции мы рассмотрели фундаментальные принципы работы сверточных нейронных сетей, а также важные аспекты их применения в различных областях. Мы изучили, как сверточные слои позволяют автоматически извлекать иерархии признаков из входных данных, что делает их особенно эффективными для обработки изображений и видео. Механизмы пулинга и активации, а также архитектурные решения, такие как использование предварительно обученных моделей, дополнили наше понимание функционирования этих сетей.