



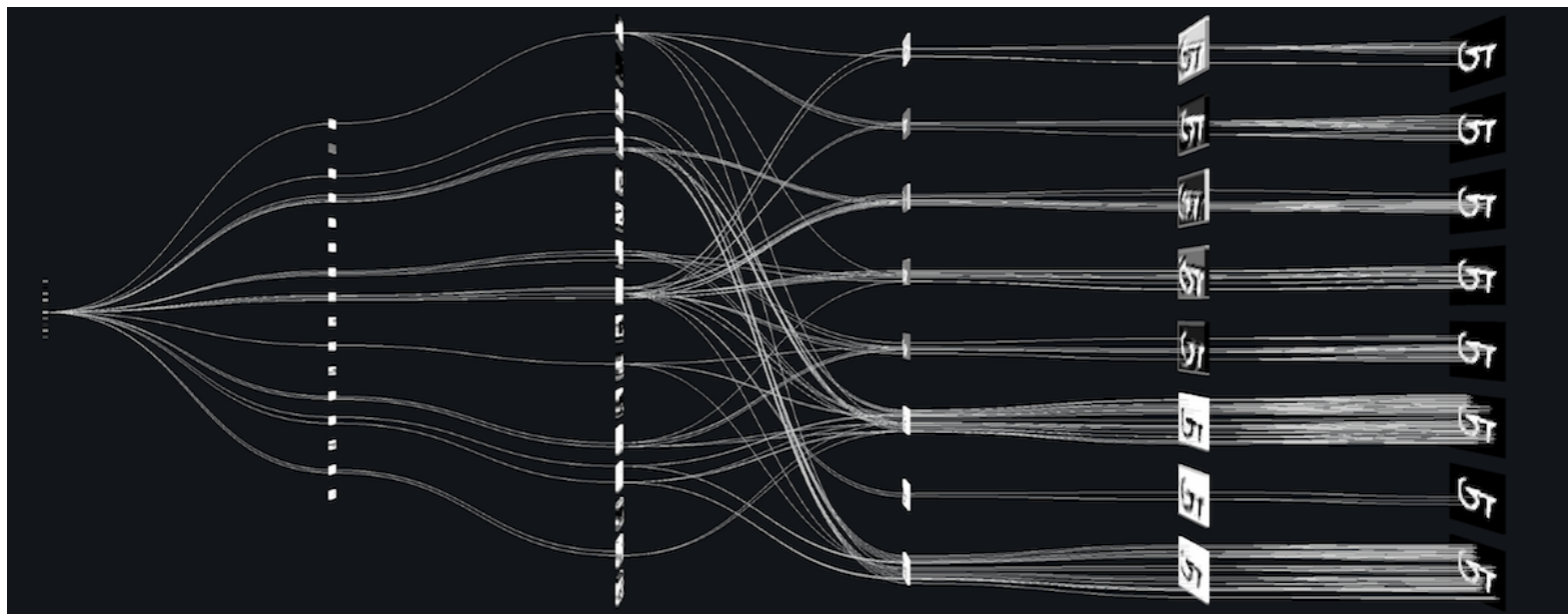
freetonic 8 сентября 2016 в 14:50

# Что такое свёрточная нейронная сеть

Программирование \*, Обработка изображений \*, Машинное обучение \*

Перевод

Автор оригинала: Adit Deshpande



## Введение

Свёрточные нейронные сети (СНС). Звучит как странное сочетание биологии и математики с примесью информатики, но как бы оно не звучало, эти сети — одни из самых влиятельных инноваций в области компьютерного зрения. Впервые нейронные сети привлекли всеобщее внимание в 2012 году, когда Алекс Крижевски благодаря им выиграл конкурс ImageNet (грубо говоря, это ежегодная олимпиада по машинному зрению), снизив рекорд ошибок классификации с 26% до 15%, что тогда стало прорывом. Сегодня глубинное обучение лежит в основе услуг многих компаний: Facebook использует нейронные сети для алгоритмов автоматического проставления тегов, Google — для поиска среди фотографий пользователя, Amazon — для генерации рекомендаций товаров, Pinterest — для персонализации домашней страницы пользователя, а Instagram — для поисковой инфраструктуры.

Но классический, и, возможно, самый популярный вариант использования сетей это обработка изображений. Давайте посмотрим, как СНС используются для классификации изображений.

## Задача

Задача классификации изображений — это приём начального изображения и вывод его класса (кошка, собака и т.д.) или группы вероятных классов, которая лучше всего характеризует изображение. Для людей это один из первых навыков, который они начинают осваивать с рождения.



What We See

```
08 02 22 97 38 15 00 40 00 75 04 05 07 78 52 12 50 77 91 08
49 49 99 40 17 81 18 57 60 87 17 40 98 43 69 48 04 56 62 00
81 49 31 73 55 79 14 29 93 71 40 67 53 88 30 03 49 13 36 65
52 70 95 23 04 60 11 42 69 24 68 56 01 32 56 71 37 02 36 91
22 31 16 71 51 67 63 89 41 92 36 54 22 40 40 28 66 33 13 80
24 47 32 60 99 03 45 02 44 75 33 53 78 36 84 20 35 17 12 50
32 98 81 28 64 23 67 10 26 38 40 67 59 54 70 66 18 38 64 70
67 26 20 68 02 62 12 20 95 63 94 39 63 08 40 91 66 49 94 21
24 55 58 05 66 73 99 26 97 17 78 78 96 83 14 88 34 89 63 72
21 36 23 09 75 00 76 44 20 45 35 14 00 61 33 97 34 31 33 95
78 17 53 28 22 75 31 67 15 94 03 80 04 62 16 14 09 53 56 92
16 39 05 42 96 35 31 47 55 58 88 24 00 17 54 24 36 29 85 57
86 56 00 48 35 71 89 07 05 44 44 37 44 60 21 58 51 54 17 58
19 80 81 68 05 94 47 69 28 73 92 13 86 52 17 77 04 89 55 40
04 52 05 83 97 35 99 16 07 97 57 32 16 26 26 79 33 27 98 66
88 36 68 87 57 62 20 72 03 46 33 67 46 55 12 32 63 93 53 69
04 42 16 73 38 25 39 11 24 94 72 18 08 46 29 32 40 62 76 36
20 69 36 41 72 30 23 88 34 62 99 69 82 67 59 85 74 04 36 16
20 73 35 29 78 31 90 01 74 31 49 71 48 86 81 16 23 57 05 54
01 70 54 71 83 51 54 69 16 92 33 48 41 43 52 01 89 19 67 48
```

What Computers See

Мы овладеваем им естественно, без усилий, став взрослыми. Даже не думая, мы можем быстро и легко распознать пространство, которое нас окружает, вместе с объектами. Когда мы видим изображение или просто смотрим на происходящее вокруг, чаще всего мы можем сразу охарактеризовать место действия, дать каждому объекту ярлык, и все это происходит бессознательно, незаметно для разума. Эти навыки быстрого распознавания шаблонов, обобщения уже полученных знаний, и адаптации к различной запечатлённой на фото обстановке не доступны нашим электронным приятелям.

## Вводы и выводы

Когда компьютер видит изображение (принимает данные на вход), он видит массив пикселей. В зависимости от разрешения и размера изображения, например, размер массива может быть 32x32x3 (где 3 — это значения каналов RGB). Чтобы было понятней, давайте представим, у нас есть цветное изображение в формате JPG, и его размер 480x480. Соответствующий массив будет 480x480x3. Каждому из этих чисел присваивается значение от 0 до 255, которое описывает интенсивность пикселя в этой точке. Эти цифры, оставаясь бессмысленными для нас, когда мы определяем что на изображении, являются единственными вводными данными, доступными компьютеру. Идея в том, что вы даете компьютеру эту матрицу, а он выводит числа, которые описывают вероятность класса изображения (.80 для кошки, .15 для собаки, .05 для птицы и т.д.).

## Чего мы хотим от компьютера

Теперь, когда мы определили задачу, ввод и вывод, давайте думать о том, как подбираться к решению. Мы хотим, чтобы компьютер мог различать все данные ему изображения и распознавать уникальные особенности, которые делают собаку собакой, а кошку кошкой. У нас и этот процесс происходит подсознательно. Когда мы смотрим на изображение собаки, мы можем отнести его к конкретному классу, если у изображения есть характерные особенности, которые можно идентифицировать, такие как лапы или уши.

как лапы или четыре ноги. Аналогичным образом компьютер может выполнять классификацию изображений через поиск характеристик базового уровня, например границ и искривлений, а затем с помощью построения более абстрактных концепций через группы свёрточных слоев. Это общее описание того, что делают СНС. Теперь перейдём к специфике.

## Биологические связи

В начале немного истории. Когда вы впервые слышали термин сверточные нейронные сети, возможно подумали о чем-то связанном с нейронами или биологией, и отчасти были правы. В каком-то смысле. СНС — это действительно прототип зрительной коры мозга. Зрительная кора имеет небольшие участки клеток, которые чувствительны к конкретным областям поля зрения. Эту идею детально рассмотрели с помощью потрясающего эксперимента Хьюбел и Визель в 1962 году ([видео](#)), в котором показали, что отдельные мозговые нервные клетки реагировали (или активировались) только при визуальном восприятии границ определенной ориентации. Например, некоторые нейроны активировались, когда воспринимали вертикальные границы, а некоторые — горизонтальные или диагональные. Хьюбел и Визель выяснили, что все эти нейроны сосредоточены в виде стержневой архитектуры и вместе формируют визуальное восприятие. Эту идею специализированных компонентов внутри системы, которые решают конкретные задачи (как клетки зрительной коры, которые ищут специфические характеристики) и используют машины, и эта идея — основа СНС.

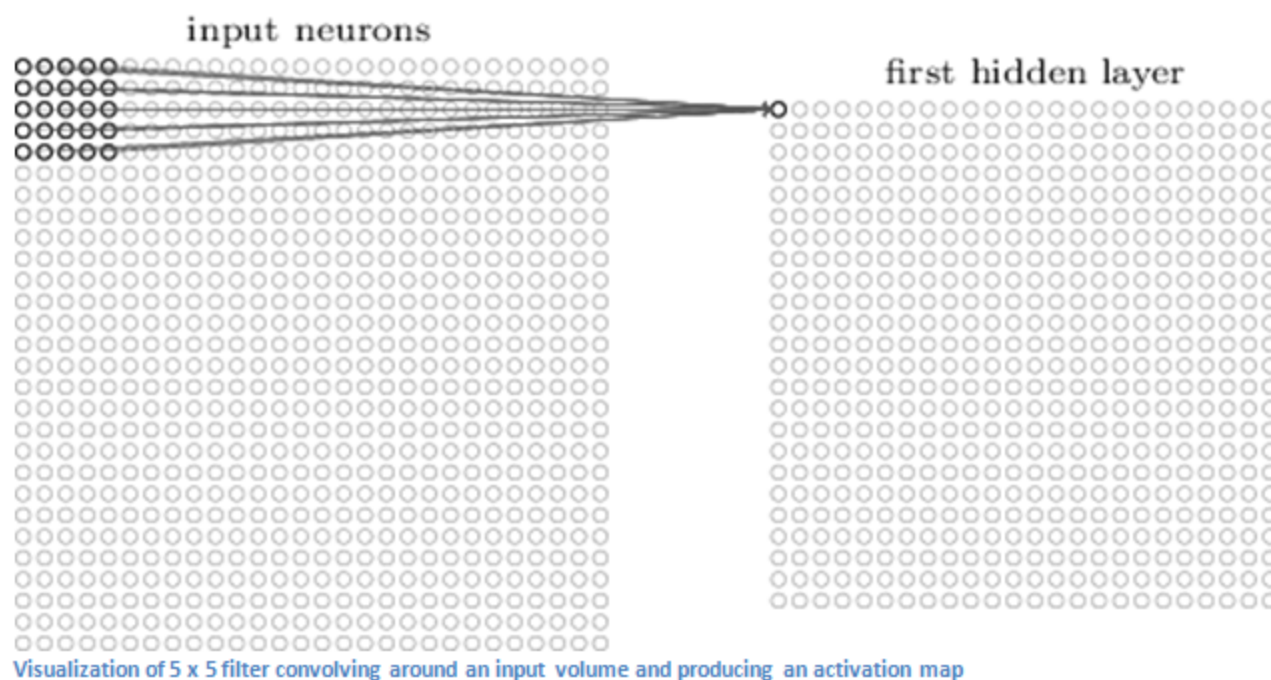
## Структура

Вернёмся к специфике. Что конкретно делают СНС? Берётся изображение, пропускается через серию свёрточных, нелинейных слоев, слоев объединения и полносвязных слоёв, и генерируется вывод. Как мы уже говорили, выводом может быть класс или вероятность классов, которые лучше всего описывают изображение. Сложный момент — понимание того, что делает каждый из этих слоев. Так что давайте перейдем к самому важному.

## Первый слой — математическая часть

Первый слой в СНС всегда свёрточный. Вы же помните, какой ввод у этого свёрточного слоя? Как уже говорилось ранее, вводное изображение — это матрица  $32 \times 32 \times 3$  с пиксельными значениями. Легче всего понять, что такое свёрточный слой, если представить его в виде фонарика, который светит на верхнюю левую часть изображения. Допустим свет, который излучает этот фонарик, покрывает площадь  $5 \times 5$ . А теперь давайте представим, что фонарик движется по всем областям вводного изображения. В терминах компьютерного обучения этот фонарик называется фильтром (иногда нейроном или ядром), а области, на которые он светит, называются рецептивным полем (полем восприятия). То есть наш фильтр — это матрица (такую матрицу ещё называют матрицей весов или матрицей параметров). Заметьте, что глубина у фильтра должна быть такой же, как и глубина вводного изображения (тогда есть гарантия математической верности), и размеры этого фильтра —  $5 \times 5 \times 3$ . Теперь давайте за пример возьмем позицию, в которой находится фильтр. Пусть это будет левый верхний угол. Поскольку фильтр производит свёртку, то есть передвигается по вводному изображению, он умножает значения

фильтра на исходные значения пикселей изображения (поэлементное умножение). Все эти умножения суммируются (всего 75 умножений). И в итоге получается одно число. Помните, оно просто символизирует нахождение фильтра в верхнем левом углу изображения. Теперь повторим этот процесс в каждой позиции. (Следующий шаг — перемещение фильтра вправо на единицу, затем еще на единицу вправо и так далее). Каждая уникальная позиция введенного изображения производит число. После прохождения фильтра по всем позициям получается матрица  $28 \times 28 \times 1$ , которую называют функцией активации или картой признаков. Матрица  $28 \times 28$  получается потому, что есть 784 различных позиции, которые могут пройти через фильтр  $5 \times 5$  изображения  $32 \times 32$ . Эти 784 числа преобразуются в матрицу  $28 \times 28$ .



(Небольшая ремарка: некоторые изображения, в том числе то, что вы видите выше, взяты из потрясающей книги "Нейронные сети и глубинное обучение" Майкла Нильсена ("[Neural Networks и Deep Learning](#)", by Michael Nielsen). Настоятельно рекомендую).

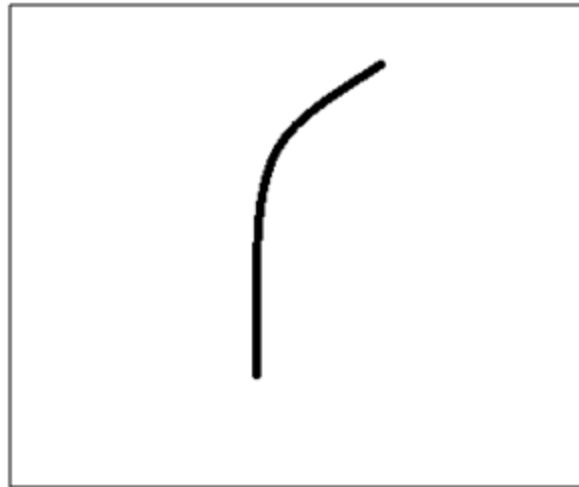
Допустим, теперь мы используем два  $5 \times 5 \times 3$  фильтра вместо одного. Тогда выходным значением будет  $28 \times 28 \times 2$ .

## Первый слой

Давайте поговорим о том, что эта свертка на самом деле делает на высоком уровне. Каждый фильтр можно рассматривать как идентификатор свойства. Когда я говорю свойство, я имею в виду прямые границы, простые цвета и кривые. Подумайте о самых простых характеристиках, которые имеют все изображения в общем. Скажем, наш первый фильтр  $7 \times 7 \times 3$ , и он будет детектором кривых. (Сейчас давайте игнорировать тот факт, что у фильтра глубина 3, и рассмотрим только верхний слой фильтра и изображения, для простоты). У фильтра пиксельная структура, в которой численные значения выше вдоль области, определяющей форму кривой (помните, фильтры, о которых мы говорим, это просто числа!).

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

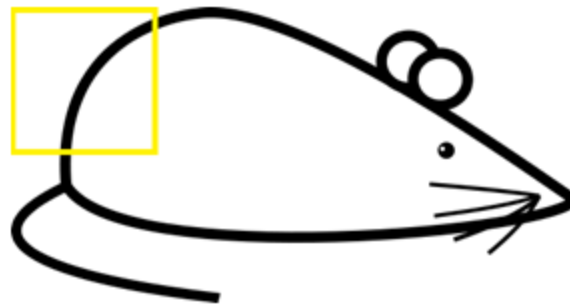


Visualization of a curve detector filter

Вернемся к математической визуализации. Когда у нас в левом верхнем углу вводного изображения есть фильтр, он производит умножение значений фильтра на значения пикселей этой области. Давайте рассмотрим пример изображения, которому мы хотим присвоить класс, и установим фильтр в верхнем левом углу.



Original image



Visualization of the filter on the image

Помните, всё что нам нужно, это умножить значения фильтра на исходные значения пикселей изображения.



Visualization of the receptive field

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

Pixel representation of the receptive field

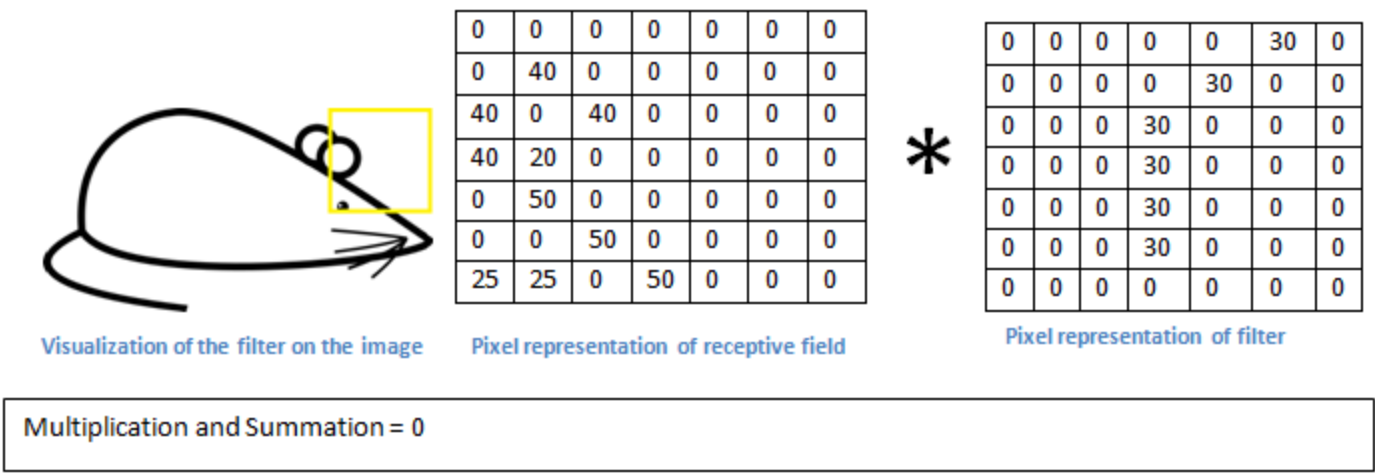
\*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation =  $(50 \cdot 30) + (50 \cdot 30) + (50 \cdot 30) + (20 \cdot 30) + (50 \cdot 30) = 6600$  (A large number!)

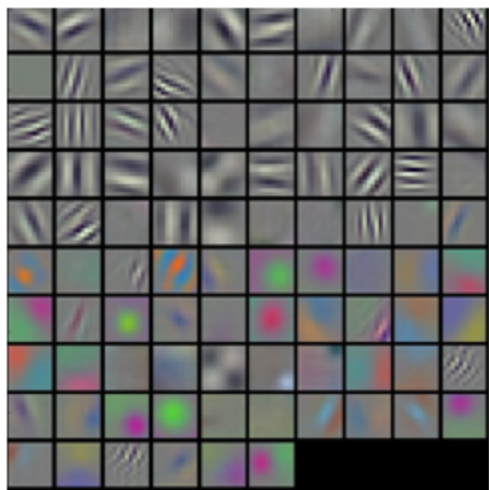
По сути, если на вводимом изображении есть форма, в общих чертах похожая на кривую, которую представляет этот фильтр, и все умноженные значения суммируются, то результатом будет большое значение! Теперь давайте посмотрим, что произойдёт, когда мы переместим фильтр.



Значение намного ниже! Это потому, что в новой области изображения нет ничего, что фильтр определения кривой мог засечь. Помните, что вывод этого свёрточного слоя — карта свойств. В самом простом случае, при наличии одного фильтра свертки (и если этот фильтр — детектор кривой), карта свойств покажет области, в которых больше вероятности наличия кривых. В этом примере в левом верхнем углу значение нашей 28 x 28 x 1 карты свойств будет 6600. Это высокое значение показывает, что, возможно, что-то похожее на кривую присутствует на изображении, и такая вероятность активировала фильтр. В правом верхнем углу значение у карты свойств будет 0, потому что на картинке не было ничего, что могло активировать фильтр (проще говоря, в этой области не было кривой). Помните, что это только для одного фильтра. Это фильтр, который обнаруживает линии с изгибом наружу. Могут быть другие фильтры для линий, изогнутых внутрь или просто прямых. Чем больше фильтров, тем больше глубина карты свойств, и тем больше информации мы имеем о вводимой картинке.

Ремарка: Фильтр, о котором я рассказал в этом разделе, упрощён для упрощения математики свёртывания. На рисунке ниже видны примеры фактических визуализаций фильтров первого свёрточного слоя обученной сети. Но идея здесь та же. Фильтры на первом слое сворачиваются вокруг вводимого изображения и "активируются" (или производят большие значения), когда специфическая черта, которую они ищут, есть во вводимом изображении.





Visualizations of filters

(Заметка: изображения выше — из Стэнфордского курса [231N](#), который преподают Андрей Карпатый и Джастин Джонсон (Andrej Karpathy and Justin Johnson). Рекомендую тем, кто хочет лучше изучить СНС).

## Идём глубже по сети

Сегодня в традиционной сверточной нейронной сетевой архитектуре существуют и другие слои, которые перемежаются со свёрточными. Я очень рекомендую тем, кто интересуется темой, почитать об этих слоях, чтобы понять их функциональность и эффекты. Классическая архитектура СНС будет выглядеть так:

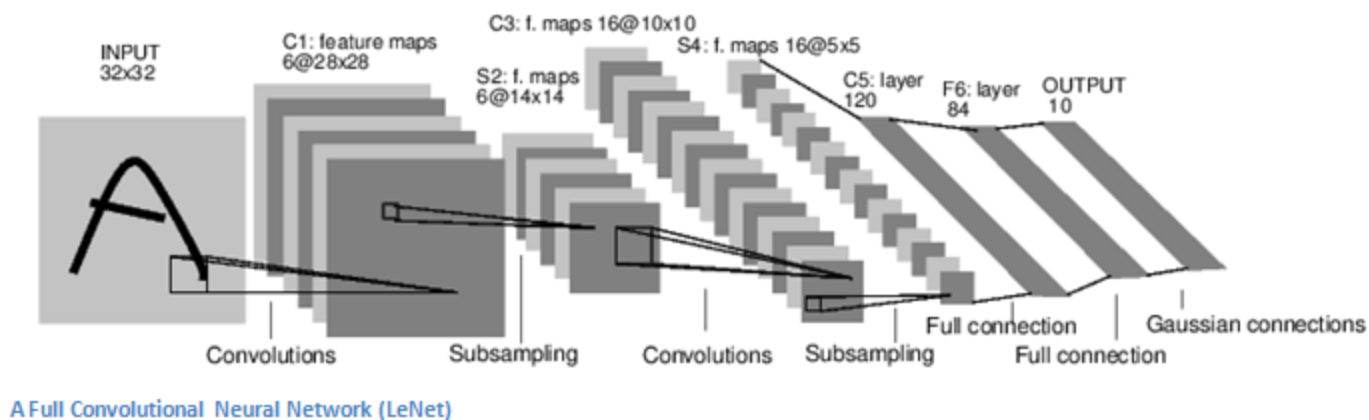
Input -> Conv -> ReLU -> Conv -> ReLU -> Pool -> ReLU -> Conv -> ReLU -> Pool -> Fully Connected

Последний слой, хоть и находится в конце, один из важных — мы перейдём к нему позже. Давайте подытожим то, в чём мы уже разобрались. Мы говорили о том, что умеют определять фильтры первого свёрточного слоя. Они обнаруживают свойства базового уровня, такие как границы и кривые. Как можно себе представить, чтобы предположить какой тип объекта изображён на картинке, нам нужна сеть, способная распознавать свойства более высокого уровня, как например руки, лапы или уши. Так что давайте подумаем, как выглядит выходной результат сети после первого свёрточного слоя. Его размер  $28 \times 28 \times 3$  (при условии, что мы используем три фильтра  $5 \times 5 \times 3$ ). Когда картинка проходит через один свёрточный слой, выход первого слоя становится входным значением 2-го слоя. Теперь это немного сложнее визуализировать. Когда мы говорили о первом слое, входом были только данные исходного изображения. Но когда мы перешли ко 2-му слою, входным значением для него стала одна или несколько карт свойств — результат обработки предыдущим слоем. Каждый набор входных данных описывает позиции, где на исходном изображении встречаются определенные базовые признаки.

Теперь, когда вы применяете набор фильтров поверх этого (пропускаете картинку через второй свёрточный слой), на выходе будут активированы фильтры, которые представляют свойства более высокого уровня. Типами этих свойств могут быть, допустим, (комбинация прямой границы с изгибом)







## Обучение (или "Что заставляет эту штуку работать")

Это один из аспектов нейронных сетей, о котором я специально до сих пор не упоминал. Вероятно, это самая важная часть. Возможно, у вас появилось множество вопросов. Откуда фильтры первого свёрточного слоя знают, что нужно искать границы и кривые? Откуда полносвязный слой знает, что ищет карта свойств? Откуда фильтры каждого слоя знают, какие хранить значения? Способ, которым компьютер способен корректировать значения фильтра (или весов) — это обучающий процесс, который называют методом обратного распространения ошибки.

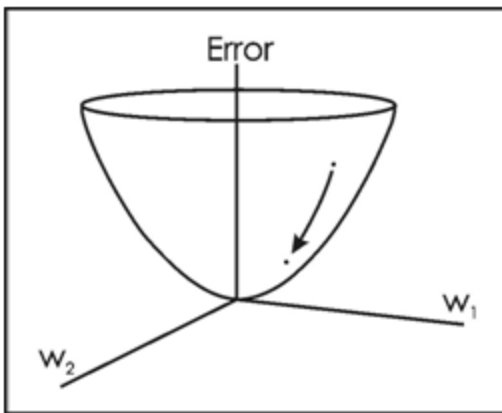
Перед тем, как перейти к объяснению этого метода, поговорим о том, что нейронной сети нужно для работы. Когда мы рождаемся, наши головы пусты. Мы не понимаем как распознать кошку, собаку или птицу. Ситуация с СНС похожа: до момента построения сети, веса или значения фильтра случайны. Фильтры не умеют искать границы и кривые. Фильтры верхних слоёв не умеют искать лапы и клювы. Когда мы становимся старше, родители и учителя показывают нам разные картинки и изображения и присваивают им соответствующие ярлыки. Та же идея показа картинки и присваивания ярлыка используется в обучающем процессе, который проходит СНС. Давайте представим, что у нас есть набор обучающих картинок, в котором тысячи изображений собак, кошек и птиц. У каждого изображения есть ярлык с названием животного.

Метод обратного распространения ошибки можно разделить на 4 отдельных блока: прямое распространение, функцию потерь, обратное распространение и обновление веса. Во время прямого распространения, берётся тренировочное изображение — как помните, это матрица  $32 \times 32 \times 3$  — и пропускается через всю сеть. В первом обучающем примере, так как все веса или значения фильтра были инициализированы случайным образом, выходным значением будет что-то вроде  $[.1 \ .1 \ .1 \ .1 \ .1 \ .1 \ .1 \ .1 \ .1 \ .1]$ , то есть такое значение, которое не даст предпочтения какому-то определённом числу. Сеть с такими весами не может найти свойства базового уровня и не может обоснованно определить класс изображения. Это ведёт к функции потерь. Помните, то, что мы используем сейчас — это обучающие данные. У таких данных есть и изображение и ярлык. Допустим, первое обучающее изображение — это цифра 3. Ярлыком изображения будет  $[0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0]$ . Функция потерь может быть выражена по-разному, но часто используется СКО (среднеквадратическая ошибка), это  $1/2$  умножить на (реальность

— предсказание) в квадрате.

$$E_{total} = \sum \frac{1}{2}(target - output)^2$$

Примем это значение за переменную  $L$ . Как вы догадываетесь, потеря будет очень высокой для первых двух обучающих изображений. Теперь давайте подумаем об этом интуитивно. Мы хотим добиться того, чтобы спрогнозированный ярлык (вывод свёрточного слоя) был таким же, как ярлык обучающего изображения (это значит, что сеть сделала верное предположение). Чтобы такого добиться, нам нужно свести к минимуму количество потерь, которое у нас есть. Визуализируя это как задачу оптимизации из математического анализа, нам нужно выяснить, какие входы (веса, в нашем случае) самым непосредственным образом способствовали потерям (или ошибкам) сети.



One way of visualizing this idea of minimizing the loss is to consider a 3-D graph where the weights of the neural net (there are obviously more than 2 weights, but let's go for simplicity) are the independent variables and the dependent variable is the loss. The task of minimizing the loss involves trying to adjust the weights so that the loss decreases. In visual terms, we want to get to the lowest point in our bowl shaped object. To do this, we have to take a derivative of the loss (visual terms: calculate the slope in every direction) with respect to the weights.

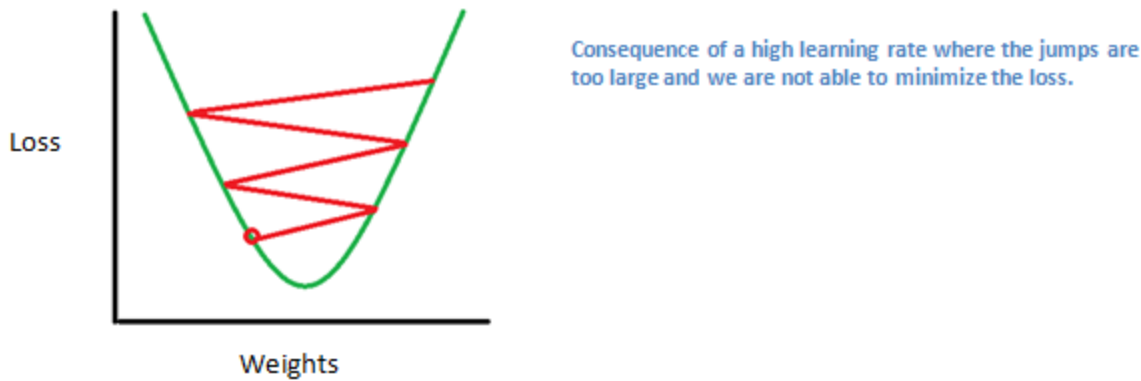
(Один из способов визуализировать идею минимизации потери — это трёхмерный график, где веса нейронной сети (очевидно их больше, чем 2, но тут пример упрощен) это независимые переменные, а зависимая переменная — это потеря. Задача минимизации потерь — отрегулировать веса так, чтобы снизить потерю. Визуально нам нужно приблизиться к самой нижней точке чашеподобного объекта. Чтобы добиться этого, нужно найти производную потери (в рамках нарисованного графика — рассчитать угловой коэффициент в каждом направлении) с учётом весов).

Это математический эквивалент  $dL/dW$ , где  $W$  — веса определенного слоя. Теперь нам нужно выполнить **обратное распространение** через сеть, которое определяет, какие веса оказали большее влияние на потери, и найти способы, как их настроить, чтобы уменьшить потери. После того, как мы вычислим производную, перейдём к последнему этапу — обновлению весов. Возьмём все фильтровые веса и обновим их так, чтобы они менялись в направлении градиента.

$$w = w_i - \eta \frac{dL}{dW}$$

$w$  = Weight  
 $w_i$  = Initial Weight  
 $\eta$  = Learning Rate

**Скорость обучения** — это параметр, который выбирается программистом. Высокая скорость обучения означает, что в обновлениях веса делались более крупные шаги, поэтому образцу может потребоваться меньше времени, чтобы набрать оптимальный набор весов. Но слишком высокая скорость обучения может привести к очень крупным и недостаточно точным скачкам, которые помешают достижению оптимальных показателей.



Процесс прямого распространения, функцию потерь, обратное распространение и обновление весов, обычно называют одним периодом дискретизации (или epoch — эпохой). Программа будет повторять этот процесс фиксированное количество периодов для каждого тренировочного изображения. После того, как обновление параметров завершится на последнем тренировочном образце, сеть в теории должна быть достаточно хорошо обучена и веса слоёв настроены правильно.

## Тестирование

Наконец, чтобы увидеть, работает ли СНС, мы берём другой набор изображений и ярлыков и пропускаем изображения через сеть. Сравниваем выходы с реальностью и смотрим, работает ли наша сеть.

## Как компании используют СНС

Данные, данные, данные. Компании, у которых тонны этого шестибуквенного магического добра, имеют закономерное преимущество перед остальными конкурентами. Чем больше тренировочных данных, которые можно скормить сети, тем больше можно создать обучающих итераций, больше обновлений весов и перед уходом в продакшн, получить лучше обученную сеть. Facebook (и Instagram) могут использовать все фотографии миллиарда пользователей, которые у них сегодня есть, Pinterest — информацию из 50 миллиардов пинов, Google — данные поиска, а Amazon — данные о миллионах продуктов, которые ежедневно покупаются. И теперь вы знаете какое волшебство они используют в своих целях.

## Ремарка



которые не обсуждались в этой статье, относятся нелинейные (nonlinear) слои и слои объединения (pooling), а также гиперпараметры сети, такие как размеры фильтров, шаги и отступы. Сетевая архитектура, пакетная нормализация, исчезающие градиенты, выпадение, методы инициализации, невыпуклая оптимизация, сдвиг, варианты функций потерь, расширение данных, методы регуляризации, машинные особенности, модификации обратного распространения и много других необсуждённых (пока ;- ) тем.

(Перевод [Наталии Басс](#))

**Теги:** [глубинное обучение](#), [машинное зрение](#), [нейронные сети](#), [мозг](#), [зрение](#)

**Хабы:** [Программирование](#), [Обработка изображений](#), [Машинное обучение](#)

◆ +91

👁 225K

📖 743



## Редакторский дайджест

Присылаем лучшие статьи раз в месяц



626.5

0

Карма

Рейтинг

**Рахим Давлеткалиев** [@freetonik](#)

Пользователь

[Сайт](#) [Twitter](#) [Github](#) [Instagram](#)

Комментарии 74



**DistortNeo**

08.09.2016 в 16:00

Эх, а ведь самая интересная часть свёрточных нейронных сетей — нелинейные преобразования, не затронуты вообще. Именно в них и заключается магия, потому что только линейными преобразованиями, которыми являются свёртки, результата не добиться.

Обратите внимание на схему:

Input -> Conv -> **ReLU** -> Conv ->...

ReLU — это просто преобразование  $\max(x, 0)$ , т.е. если  $x > 0$ , то оставляем  $x$ , а если  $x < 0$ , то заменяем на 0.

Почему в качестве нелинейного слоя в свёрточных нейронных сетях используется именно ReLU? Да чисто из практических соображений. Существует большое количество нелинейных преобразований, в т.ч. и более эффективных (размер сети получается меньше). Но именно с ReLU есть возможность обучения сравнительно большой сети за разумное время.

Дело в том, что обучение сети — это математическая задача минимизации ошибки, как это написано в статье. Но размерности задачи настолько огромны, что долгое время не существовало эффективных методов минимизации, и нейронные сети были похоронены. Однако, в последнее десятилетие была создана теория разреженных представлений, созданы математические методы приближённого решения задач  $L_0$  и  $L_1$  минимизации. А потом выяснилось, что задача обучения свёрточной нейронной сети с нелинейностью ReLU сводится к эффективно решаемым задачам из теорий разреженных представлений, что и привело к буму нейросетей.

◆ +8 Ответить

• ○  basilbasilbasil  
08.09.2016 в 16:38

ReLU — это просто преобразование  $\max(x, 0)$ , т.е. если  $x > 0$ , то оставляем  $x$ , а если  $x < 0$ , то заменяем на 0.

да, не раскрыта тема использования гиперболического тангенса.

◆ +1 Ответить

НЛО прилетело и опубликовало эту надпись здесь

• ○  Seiten  
08.09.2016 в 20:43

А откуда у вас такие детальные познания? Что такого почитать если основы уже ясны понятны? При чем хотелось бы именно в духе того что вы написали. Ну типа причина того почему именно используется ReLU и тут бах срыв покровов и тонна ценной информации, которой почему то нигде нет.

◆ +1 Ответить

• • ○  bertmsk  
09.09.2016 в 01:18

Вот кстати да. Нигде нет собсно самой мякотки — почему ReLU (ну это уже выяснили), почему ядро свертки  $3 \times 3$ ,  $5 \times 5$  или  $11 \times 11$ , нужно или нет ресайзить исходные картинки, и если надо в какое разрешение и aspect-ratio, как выбрать топологию сети, сколько слоёв и т.д. и т.п.

◆ 0 Ответить

• ○  DistortNeo

почему ядро свертки  $3 \times 3$ ,  $5 \times 5$  или  $11 \times 11$

А это просто объяснить через аналогию с фонариком. Вырезаете из картинки кусочек определённого размера и пытаетесь понять, можно ли только по этому кусочку сделать обработку или классификацию. Если нет (кусочек слишком мал) — берите размер фильтра побольше. Если же вам кажется, что размер кусочка можно уменьшить, не потеряв в точности классификации — смело уменьшайте. Чем меньше лишних данных принимает алгоритм машинного обучения, тем легче он обучается и более точные результаты выдаёт.

нужно или нет ресайзить исходные картинки, и если надо в какое разрешение и aspect-ratio

Зависит от конкретной задачи. В общем случае да, нужно подавать на вход нормализованные данные. Вместо обучения нейросети на нахождение объектов разных размеров (отличие в разы) эффективнее обучать на нахождение объектов фиксированного размера и масштабировать саму картинку.

как выбрать топологию сети, сколько слоёв

Если честно, то:

1. Методом научного тыка: попробовать разные варианты и выбрать наиболее удачный.
2. Поизучать научную литературу по данной тематике и выбрать готовую успешную топологию из статьи, авторы которой потратили кучу времени и добились успешных результатов, применяя один из этих двух методов.

+1 Ответить



НЛО прилетело и опубликовало эту надпись здесь



**DistortNeo**

09.09.2016 в 02:01

Ну так работа обязывает во всём этом разбираться. Что почитать — к сожалению, ничего посоветовать не могу. В моём случае это научные статьи: читаете статью, смотрите ссылки, которые кажутся интересными, читаете их и так пока не надоест. Со временем в голове начинает складываться целостная картина.

0 Ответить



**Idot**

09.09.2016 в 08:25

а ведь самая интересная часть свёрточных нейронных сетей



Жду появления статьи о том как научить нейронную сеть играть в крестики-нолики.

Эксперты — АУ!!!

0 Ответить



MichaelBorisov  
13.09.2016 в 21:23



Оптимальная игра в крестики-нолики неплохо реализуется в виде обычного алгоритма. Нейронные же сети следует применять там, где ничто другое не помогает, как, например, в задачах машинного зрения.

0 Ответить



НЛО прилетело и опубликовало эту надпись здесь

MichaelBorisov  
15.09.2016 в 20:44



Может вам лучше сделать забивалку гвоздей, в которой в качестве ударного элемента использовался бы микроскоп?

0 Ответить



НЛО прилетело и опубликовало эту надпись здесь

Dark\_Daiver  
20.09.2016 в 21:17



Ну как бы разные сетки в последние годы берут первые места в соревнованиях по распознаванию/локализации объектов  
[https://rodrigob.github.io/are\\_we\\_there\\_yet/build/classification\\_datasets\\_results.html](https://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html)

+1 Ответить



НЛО прилетело и опубликовало эту надпись здесь

Dark\_Daiver  
20.09.2016 в 21:31



Вобщем-то самое прямое. И к науке и к бизнесу. Для решения задач классификации/расознавания/etc есть огромное количество подходов разной степени паршивости. Для того, чтобы эту самую степень паршивости определить требуется некоторый объективный критерий. Один из таких критериев — точность алгоритма на известных и публично доступных датасетах (MNIST, CIFAR, ImageNet, и т.д.).

+1 Ответить



НЛО прилетело и опубликовало эту надпись здесь

>Все «нейросетки», которые попадались мне, удается обмануть легко  
Если вы работали с компьютерным зрением, то должны понимать, что там практически нет задач которые были бы решены полностью. Всегда есть некоторый предел. Новые методы этот предел отодвигают. Для многих задач, на текущий момент самые продвинутые методы основанны на Deep Learning.  
Та же фигня с распознаванием голоса.

На телефоне точно есть Machine Learning. Именно с его помощью происходит поиск лиц в кадре. В те времена когда я интересовался Computer Vision лучше всего эту задачу решали каскады. Теперь, возможно это делают сети.  
Всякие распознавалки текста с большой вероятностью используют внутри собственно сверточные сети.  
Prisma — это тоже сети.  
Подозреваю что можно еще много примеров найти, но мне лень.

>И ещё разок. Котенка никто не учит, что такое собака.  
А какая разница как кто учит котенка? «Биологические» нейронные сети и то что мы называем Deep Learning это довольно разные вещи.

0 Ответить

НЛО прилетело и опубликовало эту надпись здесь

> Я и возражал. Deep Learning и «нейросети» не имеют никакого отношения ни к живым организмам, ни к ИИ.  
Ну я этого нигде не утверждал. И это не мешает сетям показывать лучший результат во многих задачах.  
Подозреваю, чтобы сделать «распознавание как у котенка» надо бы хорошо понимать как собственно оно у котенка работает. И даже после этого нет гарантии что получится нечто лучше существующих методов.

>Но опять, сделайте ловушку для тараканов, станете миллиардером  
Для начала — я не специалист по Deep Learning/Computer Vision. Ну и подозреваю, что себестоимость роботизированных систем для истребления тараканов и стоимость RnD просто не позволят выйти даже в плюс, не то чтобы заработать миллиарды.

Если что — я никого не минусую.

0 Ответить

НЛО прилетело и опубликовало эту надпись здесь

Могу перестать спорить, это уже становится довольно скучно и однообразно, если честно =(

RnD дорогой не потому, что Deep learning такой плохой, а потому, что кейз «распознай таракана в темноте между кучей объектов» довольно сложен и не тривиален. Но это не самая большая проблема. Я даже думаю что некоторые успехи можно даже получить используя классические методы компьютерного зрения. Большая часть средств уйдет на сам инструмент уничтожения тараканов. Не окупится это все дело потому, что себестоимость железки будет большой, относительно классических средств. При этом физически уничтожать тараканов это не так эффективно как их травить. Т.е. мы получаем дорогое и не эффективное средство которое надо разработать, против вроде как хорошего и дешевого которое уже существует.  
Не стоит сваливать на DL/CV недостатки бизнес-модели.

0 Ответить

НЛО прилетело и опубликовало эту надпись здесь



**Dark\_Daiver**

21.09.2016 в 10:46

RnD — [https://en.wikipedia.org/wiki/Research\\_and\\_development](https://en.wikipedia.org/wiki/Research_and_development).

DL/CV — Deep Learning / Computer Vision, мне просто надоело писать длинные названия

Мне кажется вам стоит несколько лучше разобраться в той сфере о которой вы пытаетесь делать выводы.

Можно начать с этого <https://habrahabr.ru/post/274725/>

На этом у меня все.

0 Ответить



**Dark\_Daiver**

21.09.2016 в 09:49

Да, если что, на гиктаймс проскакивала статья про отпугивание котов при помощи DL. Как первое приближение можете взять локализацию для таракановушки оттуда

0 Ответить



**bit**

08.12.2016 в 12:19

Вы с животными видимо только по книжкам общались, да в зоопарке за решёткой видели?

Когда собака выкармливает котят или кошка щенков, система даёт сбой?

В быту кошки с собаками порой неплохо ладят.

А котёнок любит спину дыбить даже на руку — этому его кто учил?

0 Ответить


НЛО прилетело и опубликовало эту надпись здесь

это прилетело и спускалось от поднимов здесь

 **petroslasuk**  
08.09.2016 в 16:22

Еще будучи студентом, использовал этот тип НС для решения задачи распознавания цифр из базы MNIST. При этом весь процесс распознавания запускал на видеокарте, используя технологию CUDA. Тогда получилось достичь точности распознавания около 97,8%, но в научных работах людям удавалось достичь точности около 99%. Подозреваю, что здесь большую роль играют начальные настройки НС, когда только начинаем процесс обучения.

0 Ответить

 **DistortNeo**  
08.09.2016 в 16:34

Большую роль играет ещё и training data set.

0 Ответить

 **samodum**  
09.09.2016 в 10:12

Он же написал, что training data set — это MNIST

0 Ответить

 **DistortNeo**  
09.09.2016 в 10:19

MNIST — это просто набор изображений. Training data set они становятся только после соответствующей предобработки, учитывающей особенности алгоритма машинного обучения.

0 Ответить

 **Alex\_ME**  
08.09.2016 в 17:40


Скажите, Вы реализовывали все сами или использовали какие-то готовые решения для нейронных сетей? Если да, то какие позволяют использовать видеокарту?

0 Ответить

 **petroslasuk**  
08.09.2016 в 18:10

Писал все сам. Использовал язык C и SDK CUDA, тогда еще версия была примерно 2.0. Компилировал всё в Visual Studio.

0 Ответить

 **supersonic\_snail**  
09.09.2016 в 15:20

> Писал все сам.

Вполне возможно, что в этом и проблема, если вы не делали gradient checking. Очень просто

сделать какую-то мелкую ошибку в паре прямое-обратное распространение, типа индекса сдвинутого на 1, с которой все в принципе работает, но не так, как могло бы.

Из-за того, что сеть по сути функция, а обратное распространение считает градиент, его правильность можно очень просто проверить численно через определение градиента ( $df(x)/dx = (f(x+eps) - f(x-eps)) / (2*eps)$ ). В данном случае  $f(x)$  — cost function, которые минимизируется,  $x$  — параметры сети. Результаты, полученные численно и обратным распространением, должны совпадать числа до 5-6. Если разница больше, то что-то не так.

0 Ответить



**petrostasuk**  
12.09.2016 в 09:30

Я также не откидал этот вариант, что в коде кроется ошибка. Те ошибки, что получилось найти — исправил. Но наверное не все. А готовые библиотеки не использовал, так как это был дипломный проект в университете.

0 Ответить



**Dark\_Daiver**  
08.09.2016 в 18:14

Их довольно много, на самом деле — TensorFlow/Theano/Keras(поверх предыдущих двух)/etc

0 Ответить



**iroln**  
09.09.2016 в 17:37

какие-то готовые решения для нейронных сетей? Если да, то какие позволяют использовать видеокарту?

Сейчас все современные библиотеки для нейронных сетей используют GPU или кластер из GPU. Это просто необходимость. На CPU в production-применении сети не обучают и ничего не классифицируют ими, потому что это очень-очень медленно. Берите любую библиотеку: caffe, tensorflow, cntk, theano

Но надо учесть, что требуется современная мощная видеокарта (обычно NVIDIA с compute capability не меньше 3.0).

0 Ответить



**Alex\_ME**  
09.09.2016 в 17:44

Большое спасибо! Я посмотрел разные библиотеки, пока остановился на TensorFlow, показалось самой интересной, читаю tutoriales.

0 Ответить



Xirexel

Интересно, но что по можете указать по поводу искусственной нейронной сети Хопфилда? Она обладает свойством памяти, фильтрации и восстановления изображения.

0 Ответить

**basilbasilbasil**

08.09.2016 в 16:22

Скорость обучения — это параметр, который выбирается программистом. Высокая скорость обучения означает, что в обновлениях веса делались более крупные шаги, поэтому образцу может потребоваться меньше времени, чтобы набрать оптимальный набор весов. Но слишком высокая скорость обучения может привести к очень крупным и недостаточно точным скачкам, которые помешают достижению оптимальных показателей.

а почему бы не ввести скорость обучения сети как один из выходных параметров, чтобы сеть сама оптимально задавала себе её?

0 Ответить

**Dark\_Daiver**

08.09.2016 в 18:13

Большая часть продвинутых алгоритмов оптимизации и так стараются адаптивно регулировать скорость обучения.

0 Ответить

**snapdragon**

09.09.2016 в 07:31

Это параметр процесса оптимизации, а не самой модели. Если Вы хотите оптимизировать оптимизацию :) то это другой процесс которому тоже нужны параметры.

Но есть похожая рабочая идея — [синтетические градиенты](#). Цель, правда, там другая.

0 Ответить

**MichaelBorisov**

13.09.2016 в 21:30

Подозреваю, что создать алгоритм, который бы оптимально подбирал параметры оптимизации — это на данном этапе развития более сложная задача, чем подобрать параметры вручную.

+1 Ответить

**verge**

08.09.2016 в 17:40

Сделали бы вы курс по машинному обучению, даже самый вводный. С радостью бы прошел...

+1 Ответить





 08.09.2016 в 18:10

Введение в машинное обучение на coursera

И еще на Stepic есть хороший вводный курс по нейронным сетям

 0 [Ответить](#)

 **KirillFormado**  
08.09.2016 в 18:38

Ссылки не прошли( Ну можно по названию поискать.

 0 [Ответить](#)

НЛО прилетело и опубликовало эту надпись здесь

НЛО прилетело и опубликовало эту надпись здесь

 **ldot**  
09.09.2016 в 08:41

Кто-нибудь расскажет, чем свёрточная сеть принципиально отличается от Перцептерона, предложенного ещё в 50-х годах прошлого столетия?

 0 [Ответить](#)

 **numitus2**  
09.09.2016 в 15:20

У сверточной сети меньше настраиваемых параметров. И она нечувствительна к сдвигам.

 0 [Ответить](#)

 **Nashev**  
17.02.2020 в 19:21

Хороший вопрос, хоть и задан 4 года назад. Простите за некрокоммент.

Персептрон (однослойный) — это на каждом нейроне собранная сумма с входов с коэффициентами. Это, по сути, то самое, как тут описан свёрточный фильтр. И фильтров таких в свёрточной сети несколько, ровно как нейронов в персептроне.

Только персептрон применялся сразу весь, и сразу ко всем своим входам. А каждый фильтр в свёрточной сети применяется к большой куче разных фрагментов входного слоя.

То есть, получается, свёрточный слой в свёрточной сети — это фактически тот самый персептрон, но который просто последовательно коммутируют к разным фрагментам входного потока. И его выходы накапливают в качестве входов для следующего слоя нейронной сети.

 0 [Ответить](#)

 **AISSU**  
08.09.2016 в 18:10

Мне кажется, что через 10-20 лет будет придуман более простой механизм распознавания образов.

◆ 0 [Ответить](#)

 **perfect\_genius**  
08.09.2016 в 21:57

Мы хотим, чтобы компьютер мог различать все данные ему изображения и распознавать уникальные особенности, которые делают собаку собакой, а кошку кошкой.

И мы думаем, что раз мы можем, значит и компьютер сможет. Забывая, что мы всегда видим мир в движении и когда оно застывает — это кажется неестественным. Думаю, это фундаментальная ошибка обучателей нейросетей.


Например, по картинкам сеть не узнает, что шерсть колыхнется на ходу.

◆ 0 [Ответить](#)

 **grischenko**  
09.09.2016 в 07:46

Например, о динозаврах вы в детстве наверняка обучались по статичным картинкам, ничего не зная о моторике их шерсти/чешуи. Тем не менее, распознаете динозавров, я уверен, с высокой точностью. ;)

◆ 0 [Ответить](#)

 **Idot**  
09.09.2016 в 08:36

Человек знает, что динозавр — это ящер. И у человека есть опыт из которого он знает, что ящер — трёхмерный и двигается.

Поясню на знаменитом диване, который свёрточная сеть распознаёт как леопарда:  
— для человека леопард — это большая кошка, а кошка имеет четыре ноги (вход, выход, земля и питание), хвост, голову и так далее,  
— для свёрточных сетей леопард это характерная текстура, и потому диван крашенный под леопарда, для свёрточной сети и есть леопард, потому что сеть не знает, что леопард должен иметь четыре ноги, хвост, голову и так далее.

Такая же фигня со всеми объектами — обучать сеть НУЖНО НА ВИДЕО!

◆ 0 [Ответить](#)

 **DistortNeo**  
09.09.2016 в 09:47

Такова природа свёрточных сетей — они видят исключительно локально, небольшими фрагментами, без учёта контекста. Обычная свёрточная сеть способна определить текстуру, даже найти составные части (голова, конечность), но возможности абстрактного мышления у неё крайне

ограничены как концептуально, так и технически.

И видео не как набор изображений, а именно как связная последовательность, ей в данном случае не поможет никак.

♦ +1 Ответить



grischenko

09.09.2016 в 10:04

Разрешите все же подвергнуть Ваш радикальный капс сомнению.

Проблема знания о четырех ногах и возможных действиях объекта типа «ящер» лежит не в плоскости видео/статика. Анализ видео теми же алгоритмами даст вам просто **гораздо** большую коллекцию картинок, которые будут чрезвычайно схожи, а после нормализации вообще одинаковы. В результате, обучение станет низко эффективным.

Допустим, Вы видите картинку, леопарда|дивана в плохом разрешении. Издалека. Вы распознаете по контексту (ковёр на стене, часы, журнальный столик). Из этого контекста получается «комната», а в комнате леопарду делать нечего, следовательно это что-то, что может быть в комнате. О! Диван.

Надо понимать, что нахождение объектов на картинке (кадре) это еще не «компьютерное зрение». Это лишь один из слоев. Что, ни в коем случае, не уменьшает его важности.

♦ 0 Ответить



Idot

09.09.2016 в 10:28

У Путина дома живёт тигрёнок, так почему бы и леопарду не жить в комнате?

не в плоскости видео/статика. Анализ видео теми же алгоритмами даст вам просто гораздо большую коллекцию картинок, которые будут чрезвычайно схожи, а после нормализации вообще одинаковы. В результате, обучение станет низко эффективным.

Зрение должно выделять ДВИЖУЩИЕСЯ объекты. Например, нейросети лягушки вообще не видят, то что неподвижно.

♦ 0 Ответить



DistortNeo

09.09.2016 в 10:44

А теперь представьте размер нейросети, которая способна решать подобные задачи на видео. И представьте, какой объём данных понадобится, чтобы её обучить.

♦ 0 Ответить



Idot

09.09.2016 в 11:19

Прежде всего нужно обучить нейросеть выделять движущиеся объекты отделяя от неподвижных. А затем сфокусироваться уже выделенные объекты. Что потребует выделения

неподвижных. А затем анализировать уже выделенные объекты. Что потребует разделения нейронной сети на «отделы мозга».

0 Ответить



**DistortNeo**  
09.09.2016 в 13:29

Для выделения движущихся объектов давно созданы простые и быстрые алгоритмы — нейросеть здесь не нужна.

Сравните аналог: мозг человека имеет огромнейшую производительность, но математические операции все равно быстрее будет выполнять калькулятор, состоящий из сотни транзисторов.

И как вы собираетесь анализировать выделенные объекты? Объём данных (всевозможные видеофрагменты распознаваемых объектов), достаточных для обучения нейросети, будет настолько большой, а скорость обработки настолько медленной, что эффективнее будет использовать биологическую реализацию нейросети (живого человека).

+1 Ответить



**Idot**  
09.09.2016 в 13:34

Обучать нейросеть нужно именно на выделенных объектах. Иначе будет полнейшая хрень вроде дивана леопардового окраса.

Сам метод обучения на статических картинках — ущербный, потому что объект это не цветные пятна с характерным узором, а трёхмерный объект.

0 Ответить



**DistortNeo**  
09.09.2016 в 13:41

Обучать нейросеть нужно именно на выделенных объектах.

Обучайте. Я что, против? Я просто указал на то, что для забивания гвоздя достаточно использования молотка.

Сам метод обучения на статических картинках — ущербный, потому что объект это не цветные пятна с характерным узором, а трёхмерный объект.

Может и ущербный, но при современном уровне развития техники и математического аппарата только он и является доступным.

К тому же не забывайте про фотографии или про ситуацию, когда леопард спит.

0 Ответить

Idot

Леопард спит — это не пятна в вытянутые горизонтально, а это трёхмерный объект принявший позу для сна.

А с определением «леопард спит — это пятна вытянутые горизонтально» будет получен тот самый диван, потому что он вытянут горизонтально и имеет пятна,

0

Ответить

DistortNeo

09.09.2016 в 13:52

Трёхмерных объектов не существует, есть только двухмерные изображения, то которым можно сделать выводы о трёхмерности.

А чучело леопарда — это леопард? А чучело леопарда, оформленное в виде дивана?

0

Ответить

Idot

09.09.2016 в 14:04

Упоротый лис — это тоже лис.

0

Ответить

fireSparrow

09.09.2016 в 15:20

Скажите, каков ваш опыт работы в области машинного зрения в целом, и какие успехи в применении того подхода, который вы описываете?

Может быть, напишете подробную статью? Уверен не только мне было бы интересно узнать больше подробностей о столь новаторском подходе, особенно если он действительно у вас показывает лучшую эффективность, чем стандартный.

0

Ответить

samodum

09.09.2016 в 10:18

Когда леопард лежит, то ты не видишь ни четырёх лап, ни хвоста

0

Ответить

Idot

09.09.2016 в 10:35

... но, тем не менее известно, что они у него есть и он может их прятать. Если не видно ни лап, ни хвоста, то объект вероятно леопард, и на 100% уверенности в этом нет. Уверенность возникает — когда он пошевелился.

0

Ответить

 **DistortNeo**  
09.09.2016 в 10:42

Или человек в костюме леопарда.

 +1 [Ответить](#)



 **Nashev**  
13.09.2016 в 20:33

Темпоральные сети уже придуманы, и тоже активно развиваются. Видео, аудио и прочие потоки информации не останутся без внимания!

 0 [Ответить](#)



 **gugura**  
09.09.2016 в 10:43

С ростом вычислительных мощностей можно будет и на трехмерных видео обучать.  
То что сейчас делают разработчики алгоритмов — детские игрушки, у самого простого таракана миллион нейронов.

 0 [Ответить](#)



НЛО прилетело и опубликовало эту надпись здесь

 **AlekseiMorozov19730316Ru**  
09.09.2016 в 10:43

>некоторые нейроны активировались,  
>когда воспринимали вертикальные границы,  
>а некоторые — горизонтальные или диагональные

То есть у этих нейронов дендриты «выросли» так, что в итоге нейрон умеет реагировать на определённый паттерн. Но «вырастить» все возможные паттерны на всех уровнях невозможно. Поэтому для ассоциирования на верхних уровнях у нейронной сети и возникло ширококвещательное распространение паттернов.

 0 [Ответить](#)



Только полноправные пользователи могут оставлять комментарии. [Войдите](#), пожалуйста.



Расширенная гарантия  
с iPort+ Business.

iPort+ | Apple  
Partners  
Reseller

ООО «Портативная техника», юрид. адрес: 190031,  
Санкт-Петербург, наб. реки Фонтанки, д.109,  
литер А, пом. 13Н, ОГРН № 1057811930296.

Суперсила  
профессионалов.



MacBook Pro

## ПОХОЖИЕ ПУБЛИКАЦИИ

2 июня 2020 в 12:10

Книга «Генеративное глубокое обучение. Творческий потенциал нейронных сетей»

+5

4.1K

31

0

2 ноября 2018 в 12:00

I see you: машинное обучение и искусственные нейронные сети в изучении зрения дрозофил

+17

6.7K

40

1 +1

8 октября 2018 в 14:38

Обучение и тестирование нейронных сетей на PyTorch с помощью Ignite

+36

25K

157

4 +4

## МИНУТОЧКУ ВНИМАНИЯ

Разместить



Мегапост

Инфобез от первого лица: снимаем проклятие знания



Мегапост

Город, которого нет: OFDMA, MU-MIMO и 802.11ax

## ЗАКАЗЫ

### Собрать мой приложение на Firebase

1000 руб./за проект · 5 откликов · 23 просмотра

### Собрать верстку калькулятора

18000 руб./за проект · 10 откликов · 127 просмотров

### Разработка Screen Mirroring App на iOS

90000 руб./за проект · 4 отклика · 12 просмотров

### Поиск картинок в интернете (много)

500 руб./в час · 29 откликов · 97 просмотров

### Собрать информацию о 197 университетов

120000 руб./за проект · 10 откликов · 97 просмотров

[Больше заказов на Хабр Фрилансе](#)

## ЛУЧШИЕ ПУБЛИКАЦИИ ЗА СУТКИ

вчера в 15:34

### О форме Земли: тыква или дыня? Геодезия и отвага

◆ +55

👁 4.5K

🔖 18

💬 8 +8

вчера в 14:02

### Доказательная медицина в оториноларингологии за последние 10 лет (мифы и что вы не знали про это)

◆ +46

👁 8.7K

🔖 71

💬 40 +40

вчера в 10:05

### Вторая лунная гонка. Что получают завоеватели?

◆ +44

👁 9.6K

🔖 33

💬 33 +33

вчера в 11:58

### Смерть легионера. Серия «Жертвы Везувия», часть 1

◆ +43

👁 5.2K

🔖 9

💬 9 +9


вчера в 20:57

### Металлогалогенные лампы: маленькое домашнее солнце

Реклама


✕

# MacBook Pro. Суперсила профессионалов.



Доступны в лизинг и Trade-in  
для юридических лиц.

ООО «Портативная техника»,  
юр.адрес: 190031, Санкт-Петербург,  
наб. реки Фонтанки, д.109, литер А,  
пом. 13Н, ОГРН № 1057811930296.

**iPort** 

## Ваш аккаунт

Войти

Регистрация

## Разделы

Публикации

Новости

Хабы

Компании

Авторы

Песочница

## Информация

Устройство сайта

Для авторов

Для компаний

Документы

Соглашение

Конфиденциальность

## Услуги

Реклама

Тарифы

Контент

Семинары

Мегапроекты



[О сайте](#)

[Техническая поддержка](#)

[Вернуться на старую версию](#)

© 2006–2021 «Habr»





















«Сбер» оказался недоволен производительностью российских серверов на базе процессоров «Эльбрус-8С»

 31K  118 **+118**

Непереводимые слова: 7 русских лексем, которых не хватает в английском

 15K  54 **+54**

Авторы коллективного иска против Apple утверждают, что Apple Watch опасно носить

 5.6K  5 **+5**

Кризис видеокарт — что делать и кто виноват

 5.7K  32 **+32**

Госкорпорация Ростех показала беспилотник С-70 «Охотник» с плоским реактивным соплом

 5.8K  14 **+14**

Пазл DevSecOps: как из знаний, инструментов и процессов сложить слово «безопасность»

Турбо

## РАБОТА

Data Scientist

144 вакансии

[Все вакансии](#)