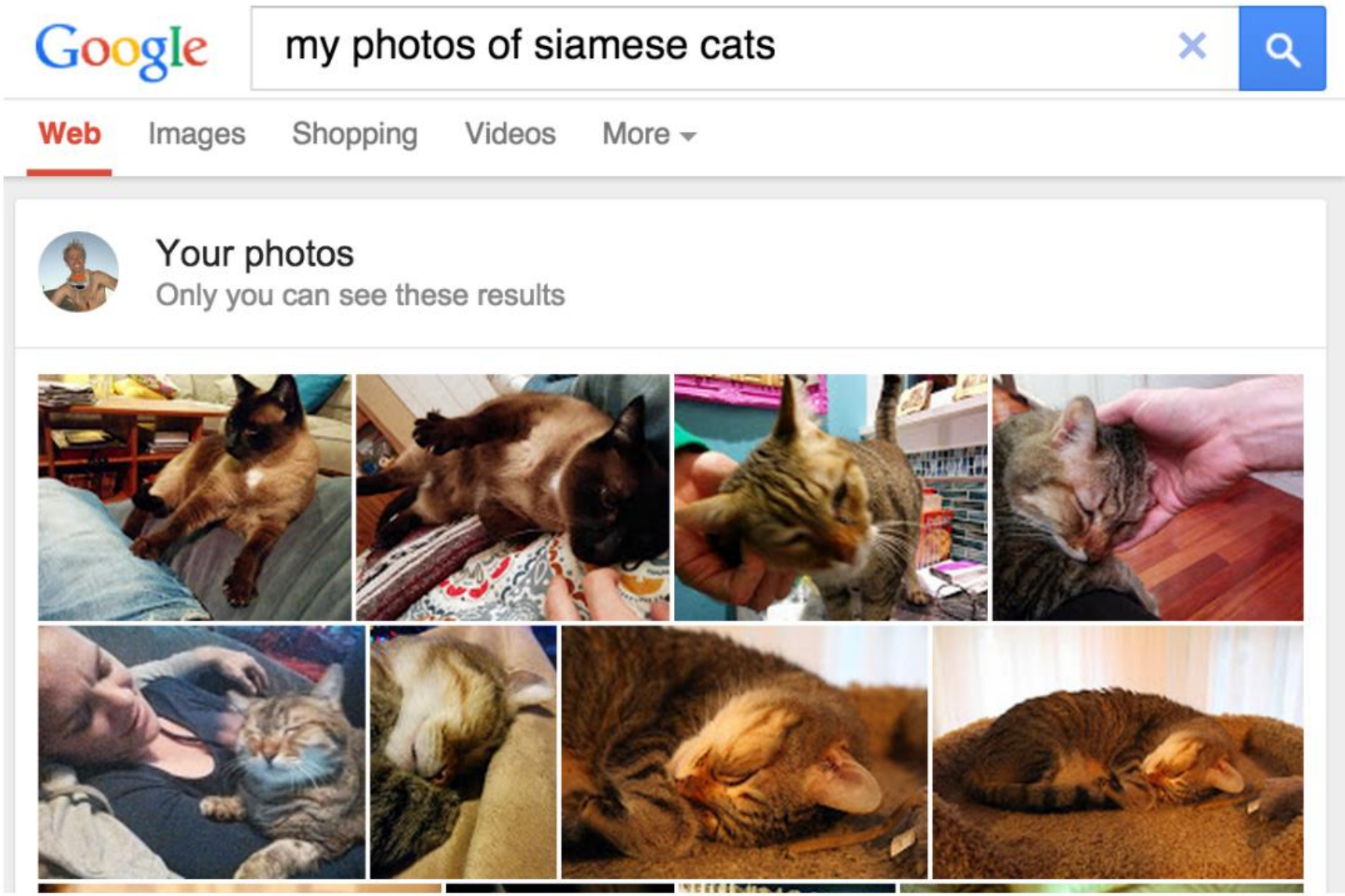
Глубокие нейронные сети

Вступление

В мае 2013 года Google [запустил поиск личных фотографий](), дав пользователям возможность находить фотографии в своих библиотеках на основе объектов, присутствующих на изображениях.



Эта функция, позже включенная в Google Фото в 2015 году, была широко воспринята как переломный момент, доказательство концепции того, что программное обеспечение компьютерного зрения может классифицировать изображения в соответствии с человеческими стандартами, добавляя ценность несколькими способами:

- Пользователям больше не нужно было помечать фотографии метками, такими как «пляж», для категоризации содержимого изображений, что устраняет ручную задачу, которая может стать довольно утомительной при управлении наборами из сотен или тысяч изображений.
- Пользователи могут исследовать свою коллекцию фотографий по-новому, используя условия поиска, чтобы находить фотографии с объектами, которые они, возможно, никогда не помечали. Например, они могут выполнить поиск по запросу «пальма», чтобы отобразить все фотографии из отпуска, на которых на заднем плане были пальмы.
- Программное обеспечение потенциально может «видеть» таксономические различия, которые сами конечные пользователи могут не заметить (например, различение сиамских и абиссинских кошек), эффективно расширяя знания пользователей в предметной области.

Как работает классификация изображений

Классификация изображений - это задача обучения с учителем: определить набор целевых классов (объекты для идентификации на изображениях) и обучить модель распознавать их, используя размеченные примеры фотографий. Ранние модели компьютерного зрения полагались на исходные пиксельные данные в качестве входных данных. Однако, как показано на рисунке, сами по себе необработанные пиксельные данные не обеспечивают достаточно стабильного представления, чтобы охватить мириады вариаций объекта, захваченного на изображении. Положение объекта, фон за объектом, окружающее освещение, угол наклона камеры и фокус камеры - все это может вызвать колебания в необработанных данных пикселей; эти различия достаточно значительны, и их нельзя исправить, взяв средневзвешенные значения пиксельных значений RGB.

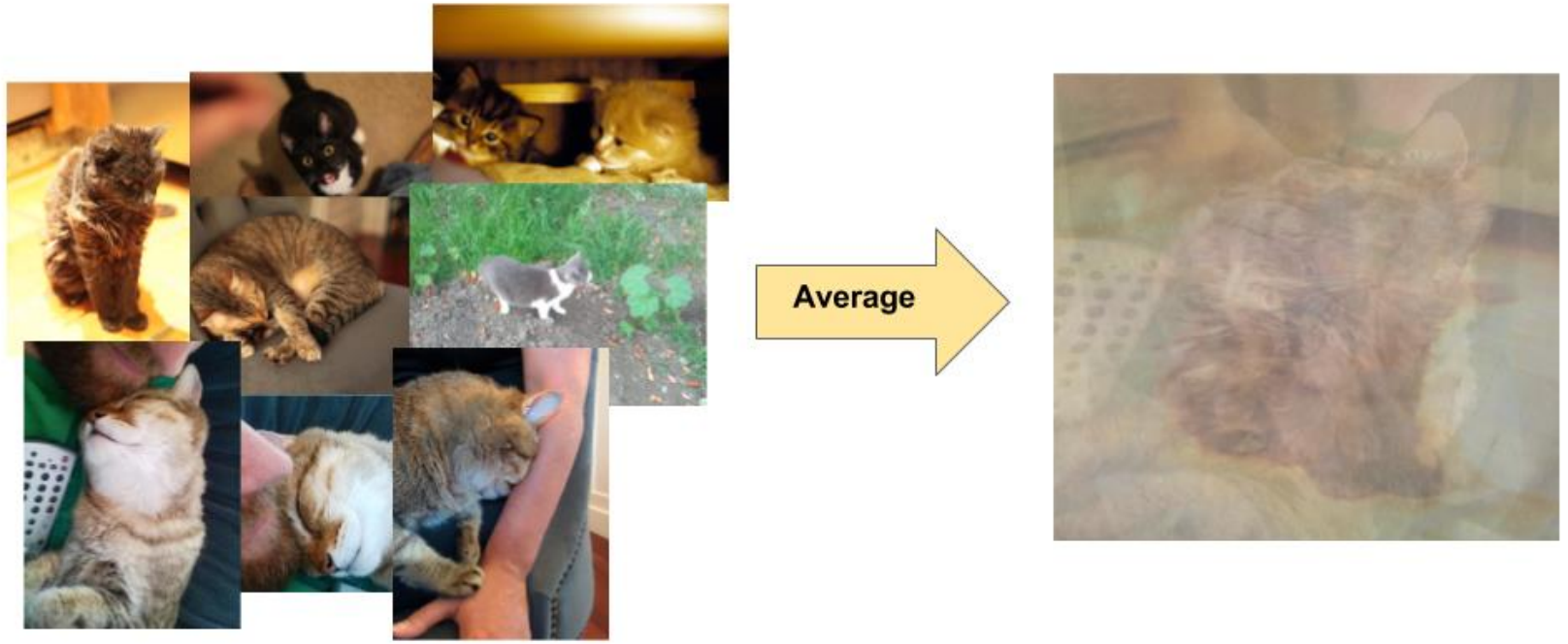


Рисунок 2. ***Слева****: кошек можно запечатлеть в разных позах, с разными фонами и условиями освещения.* ***Справа****: усреднение пиксельных данных для обучения не дает никакой значимой информации.*

Чтобы моделировать объекты более гибко, в классические модели компьютерного зрения добавлены новые функции, полученные на основе данных пикселей, такие как гистограммы цвета , текстуры и формы. Обратной стороной этого подхода было то, что разработка функций (фичей) стала настоящим бременем, так как нужно было настраивать очень много входных данных. Какие цвета были наиболее актуальны для классификатора кошек? Насколько гибкими должны быть определения формы? Поскольку функции необходимо было настроить так точно, создание надежных моделей было довольно сложной задачей, и точность их была не высокой.

Представление изображения (Image Representation)

- Изображения состоят из пикселей, значения пикселей от 0 до 255.
- Два распространенных типа:

Grayscale Image:

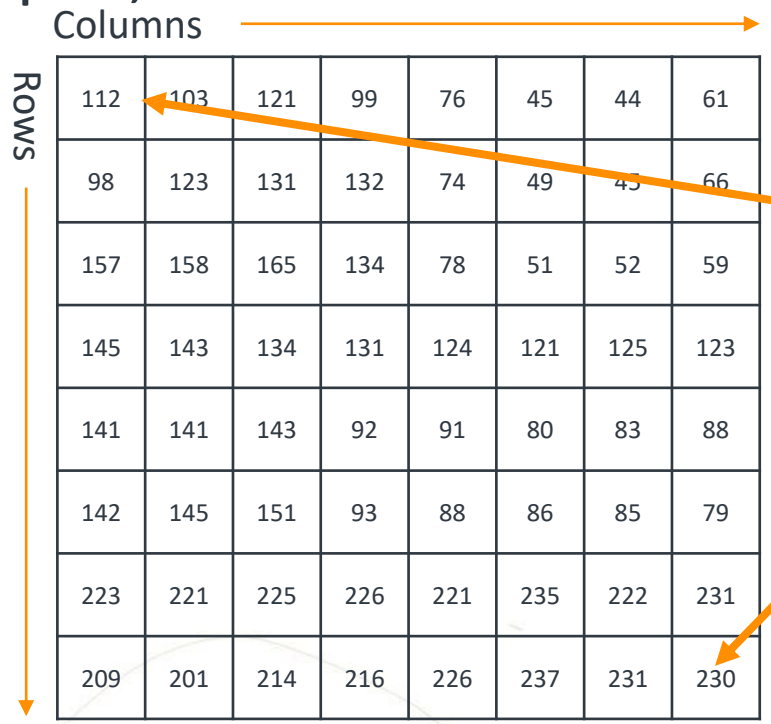
- Один канал
- Значения пикселей в диапазоне 0-255
  - 0: Самая темная интенсивность
  - 255: Самая яркая интенсивность

Color Image:

- 3 канала
  - Красный, зеленый и синий
- RGB - это обычное представление
- Значения пикселей в диапазоне 0-255

Представление изображения

На изображении в оттенках серого размером N x M, **Im** (M: количество строк, N: количество столбцов)

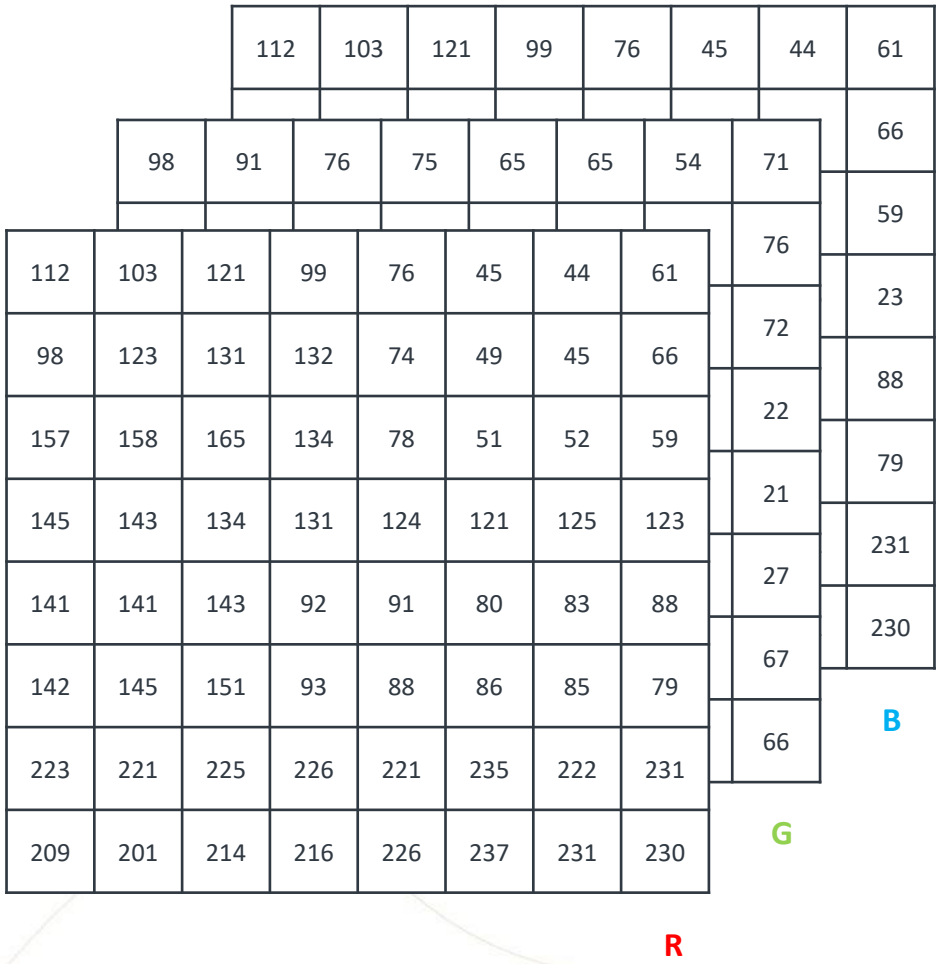


Im [m, n] => m пикселей вниз, n пикселей вправо

Im [0, 0]

Im [M-1, N-1]

Цветные изображения (Color Images)



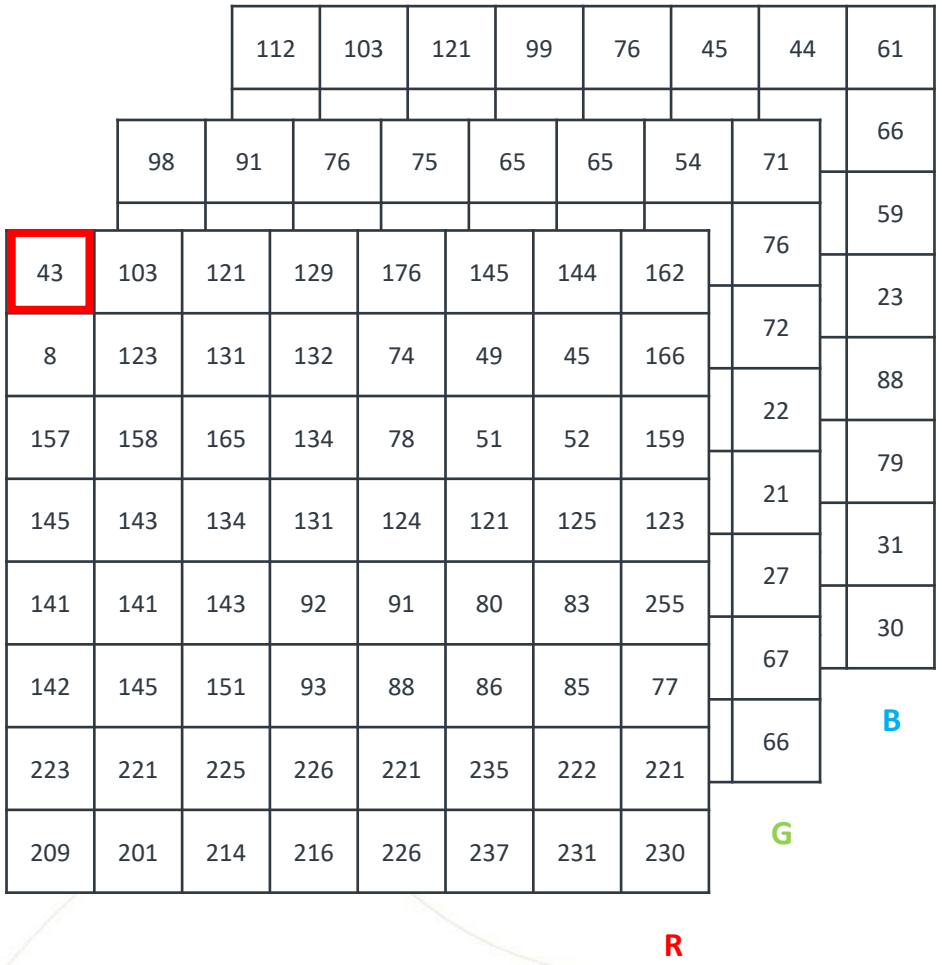
На цветном изображении размером N x M x 3, Im

- N: # of rows,
- M: # of columns,
- 3 channels

Im [m, n, c] :

- **m pixels down,**
- **n pixels right,**
- **c : 0,1, or 2 (“R”, “G”, or “B”)**

Цветные изображения (Color Images)



On a N x M x 3 color image, Im

- N: # of rows,
- M: # of columns,
- 3 channels

- **Im** [0, 0, 0] => 43

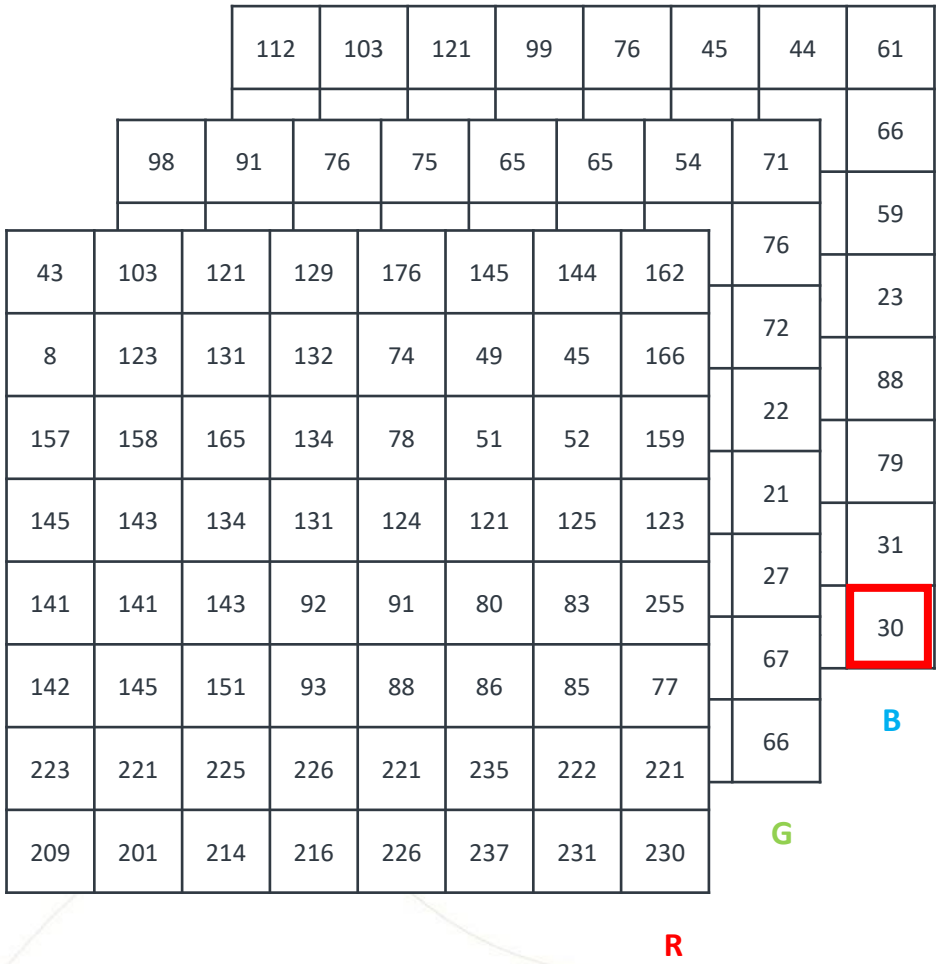
Color Images

On a N x M x 3 color image, **Im**

- N: # of rows,
- M: # of columns,
- 3 channels

- **Im** [0, 0, 0] => 43
- **Im** [0, 0, 1] => 98

Color Images



On a N x M x 3 color image, **Im**

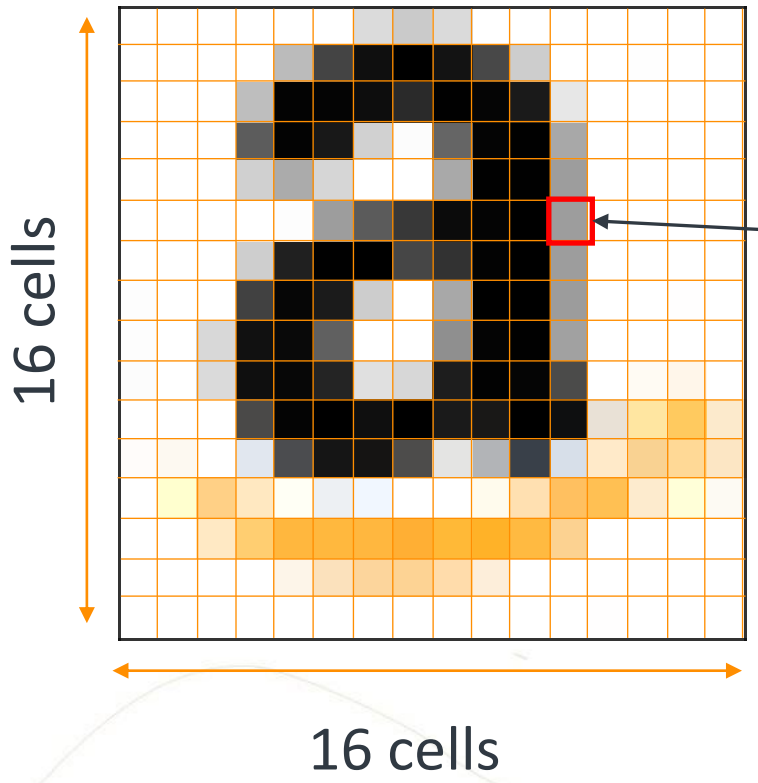
- N: # of rows,
- M: # of columns,
- 3 channels

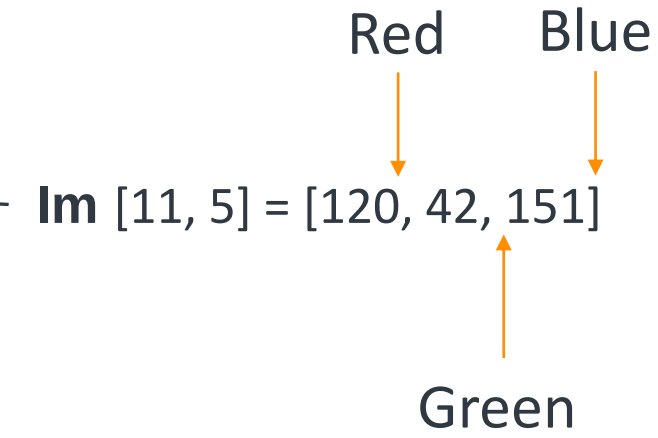
- **Im** [0, 0, 0] => 43
- **Im** [0, 0, 1] => 98
- **Im** [N-1, M-1, 2] => 30

Цветные изображения (Color Images)

Например, на изображении 16x16 RGB.

Im :





Проблемы многослойных персептронов (MLP)

Обучение по очень многим параметрам …

 # hidden units: 100
 # parameters: 100x120,000 =12 million!!!

 # inputs: 200x200x3 =120,000

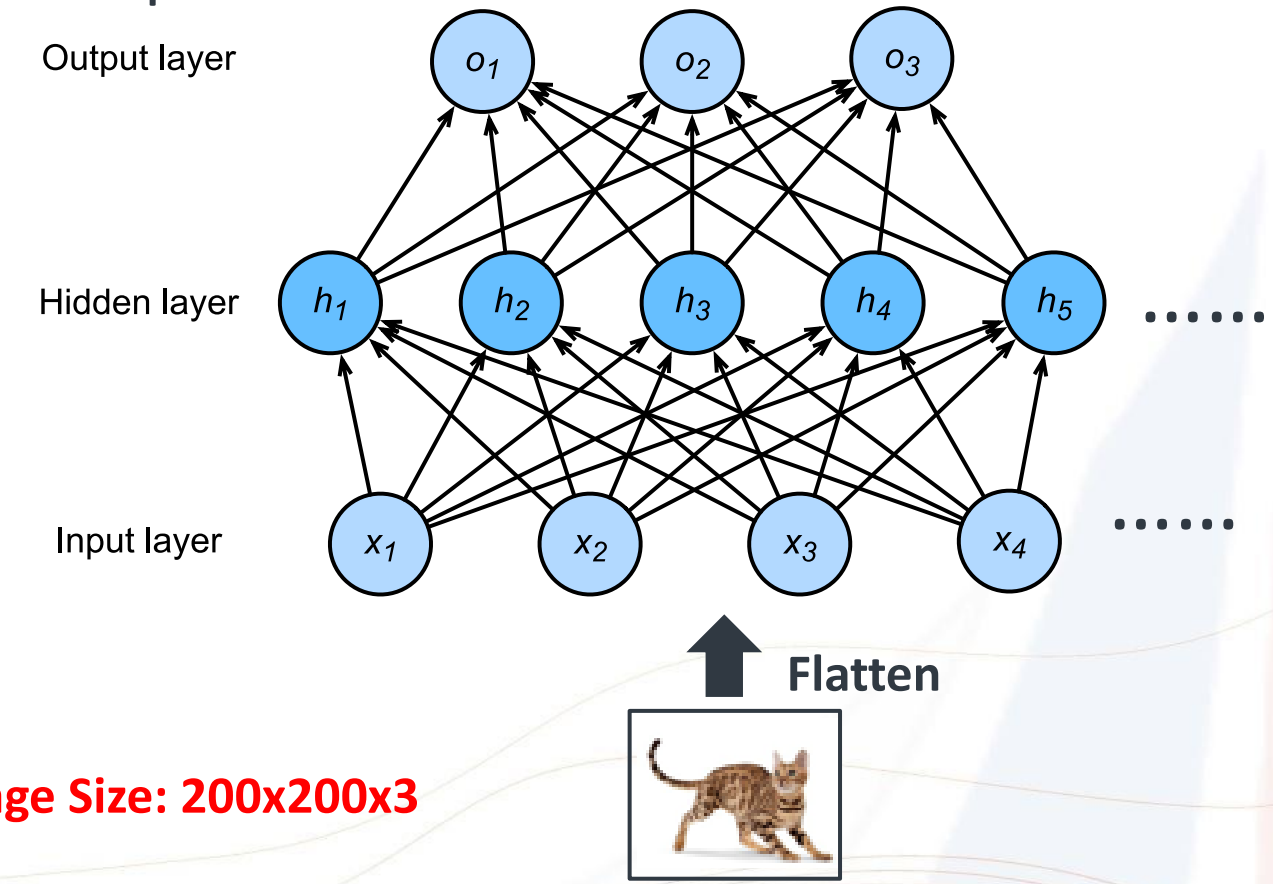


Image Size: 200x200x3

Можем ли мы уменьшить параметры?

Where is Waldo?

Наша система технического зрения …

1. Инвариантность перевода :

Аналогично реагируют на один и тот же объект, независимо от его местоположения.

Our vision system…

2. Локальность (Locality):

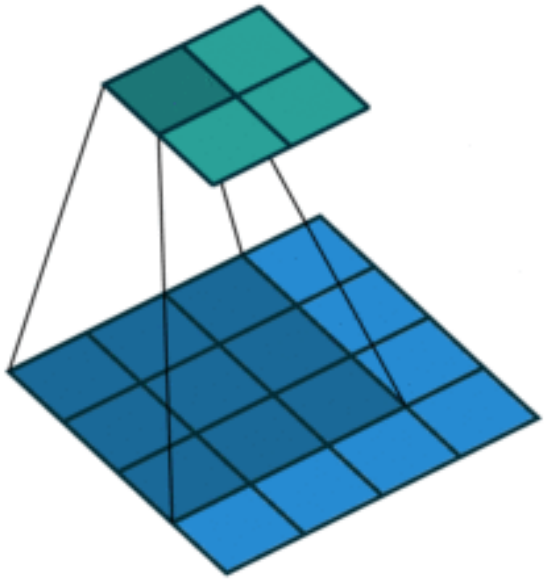
Ориентация на несколько локальных областей (регионов), а не на более крупный масштаб

Filters

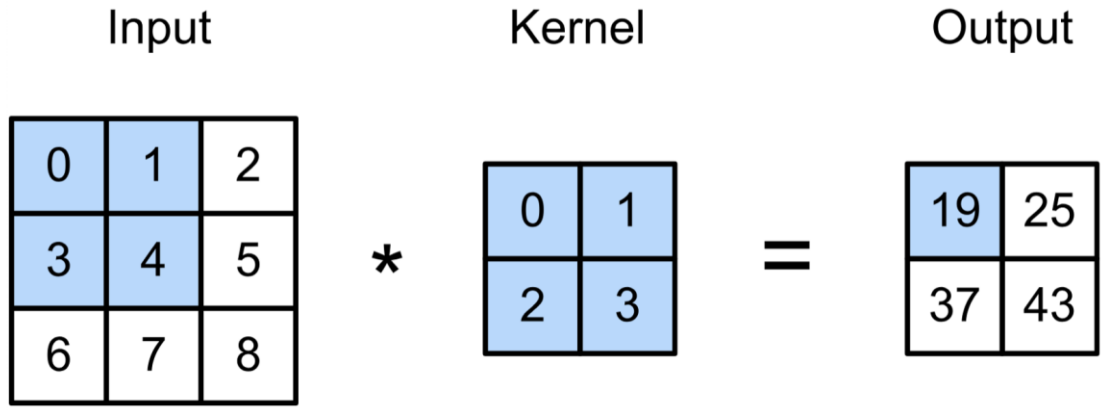
- Каждый пиксель изображения может быть реализован как $x_{i,j}$
- Каждый «вес» (“weight”) в окне фильтра (или свертки, или ядра:convolution, or kernel) может быть определен следующим образом: $w_{a,b}$

$$h_{i,j} = \sum_{a=-\Delta}^{\Delta} \sum_{b=-\Delta}^{\Delta} w_{a,b} x_{i+a,j+b}$$

2D Convolution Example



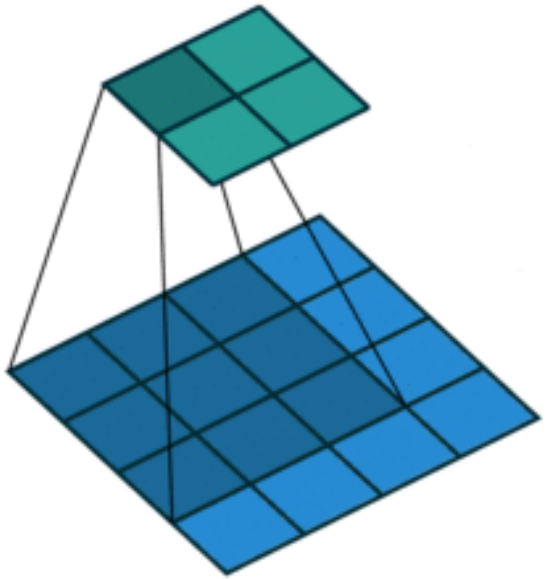
(vdumoulin@ Github)



$$0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3 = 19$$

$$h_{i,j} = \sum_{a=-\Delta}^{\Delta} \sum_{b=-\Delta}^{\Delta} w_{a,b} x_{i+a,j+b}$$

2D Convolution Example



(vdumoulin@ Github)

Input				Kernel			Output	
0	1	2	*	0	1	=	19	25
3	4	5		2	3		37	43
6	7	8						

- $\mathbf{X} : n_h \times n_w$ input matrix
- $\mathbf{W} : k_h \times k_w$ kernel matrix
- b : the bias scalar
- $\mathbf{Y} : (n_h - k_h + 1) \times (n_w - k_w + 1)$ output matrix

$$\mathbf{Y} = \mathbf{X} \star \mathbf{W} + b$$

- W and b are the trainable parameters

Filters

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

(wikipedia)

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Что будет после фильтров?

Filters

(wikipedia)

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Edge Detection — Определение границ

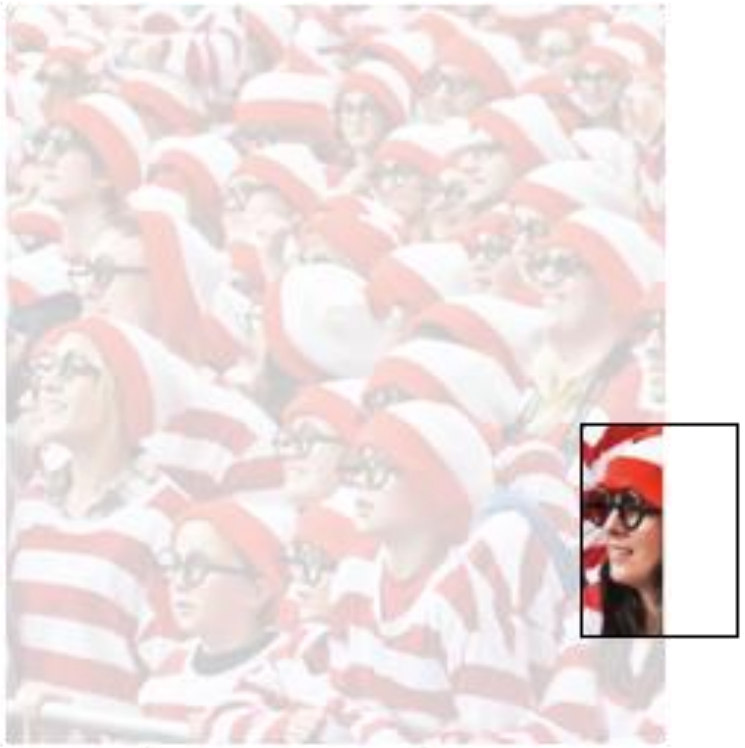
$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Sharpen — резкость

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Gaussian Blur — Гауссово размытие

Что мы делаем у границы?



Исходное окно свертки может игнорировать этого Уолли на границе …

Padding

Padding добавляет строки / столбцы вокруг ввода.

Input

0	0	0	0	0
0	0	1	2	0
0	3	4	5	0
0	6	7	8	0
0	0	0	0	0

*

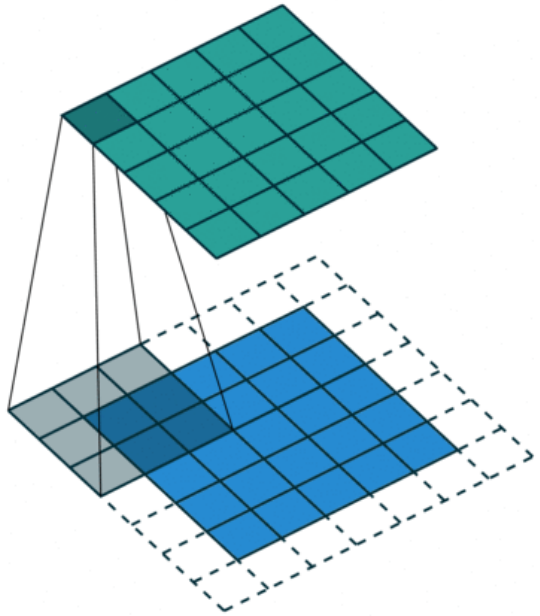
Kernel

0	1
2	3

=

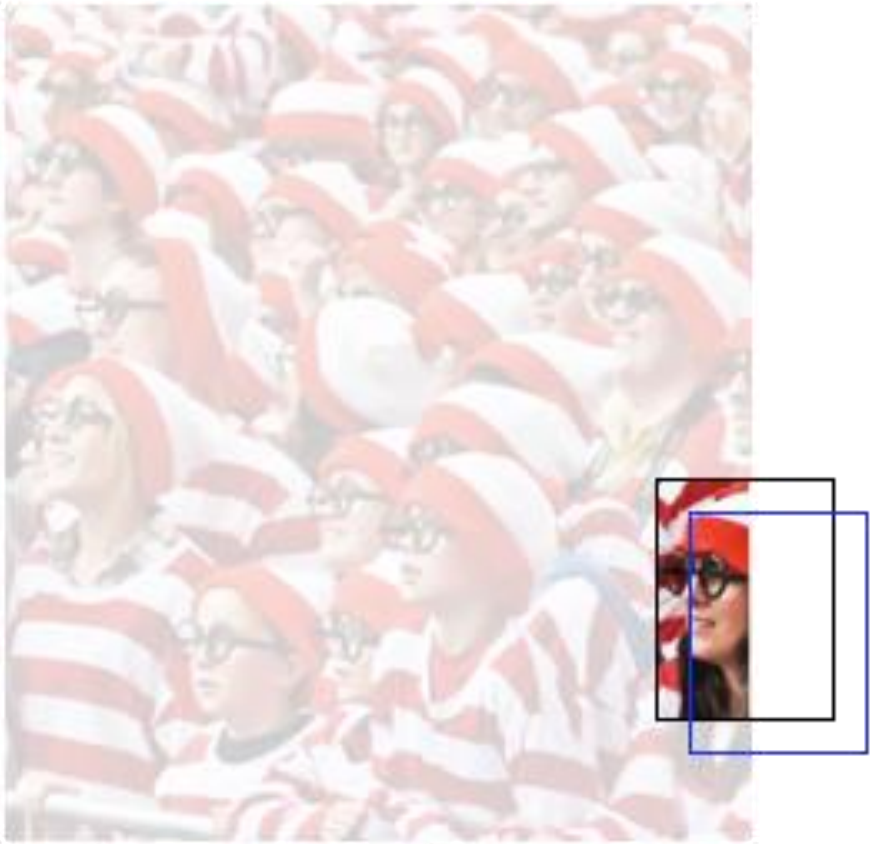
Output

0	3	8	4
9	19	25	10
21	37	43	16
6	7	8	0



(vdumoulin@ Github)

Как насчет двух почти одинаковых окон?

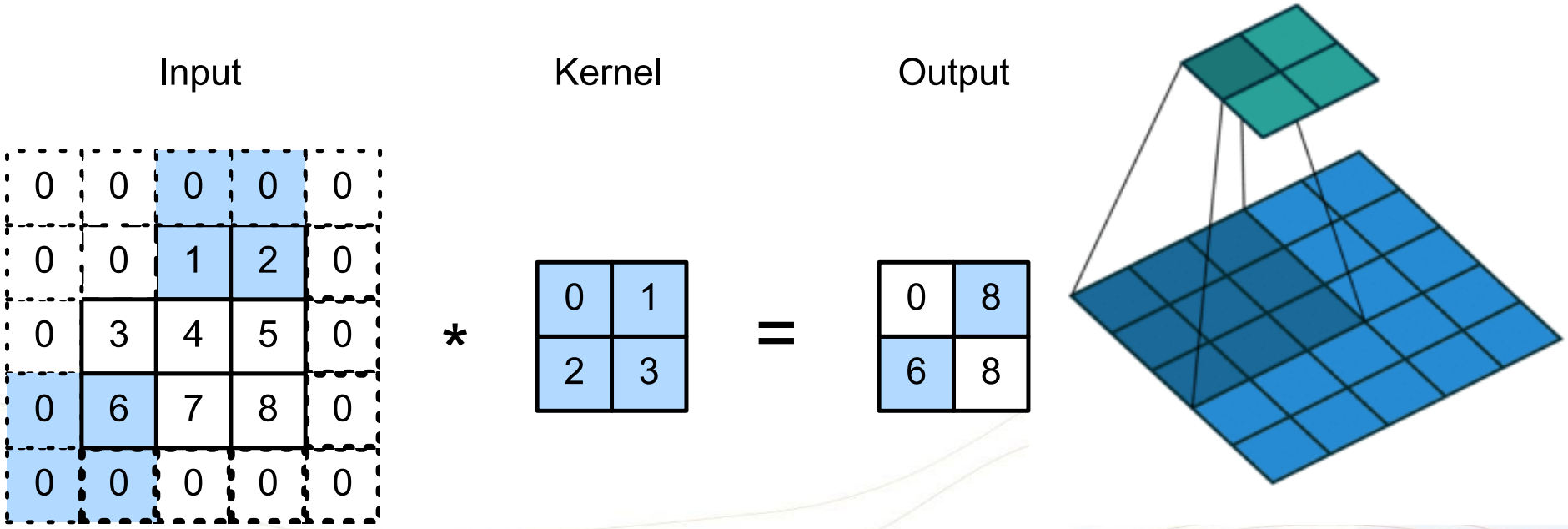


Слишком дорого с вычислительной точки зрения, чтобы сдвигать по одному пикселю за раз …

Stride

Stride это количество «единиц», на которое ядро сдвигается за слайд по строкам / столбцам..

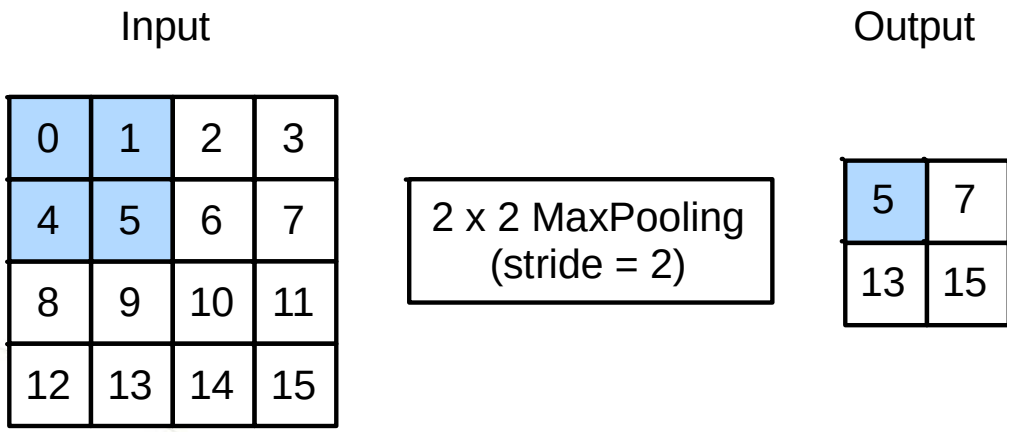
Например, 3 шага для высоты и 2 для ширины.



Pooling

Pooling используется для уменьшения размера (h и w) карты признаков (feature map).

- может работать с padding и stride
- Нет весовых коэффициентов



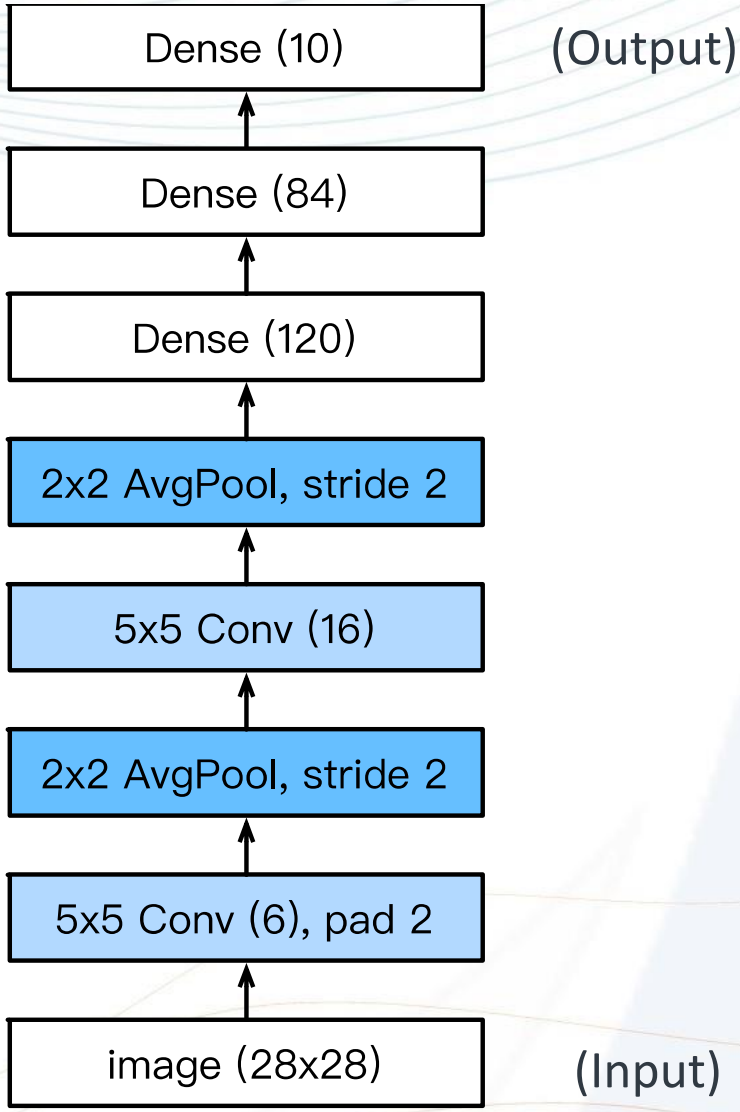
Max Pooling: Возвращает максимальное значение в окне объединения.

Average Pooling: Возвращает среднее значение в окне

CNN Layers

Простая сверточная нейронная сеть (CNN) обычно состоит из следующих слоев:

- Свертка
- Объединение
- Полносвязный



Сверточные нейронные сети

Прорыв в построении моделей для классификации изображений произошел с открытием того, что сверточную нейронную сеть (CNN) можно использовать для постепенного извлечения все более и более глубоких уровней представлений содержимого изображения. Вместо предварительной обработки данных для получения таких функций, как текстуры и формы, CNN принимает в качестве входных данных только необработанные пиксельные данные изображения и «учится» **извлекать эти функции** и в конечном итоге делает вывод, какой объект они составляют.
Для начала CNN получает входную карту характеристик: трехмерную матрицу, в которой размер первых двух измерений соответствует длине и ширине изображений в пикселях. Размер третьего измерения - 3 (соответствует трем каналам цветного изображения: красному, зеленому и синему). CNN состоит из набора модулей, каждый из которых выполняет три операции.

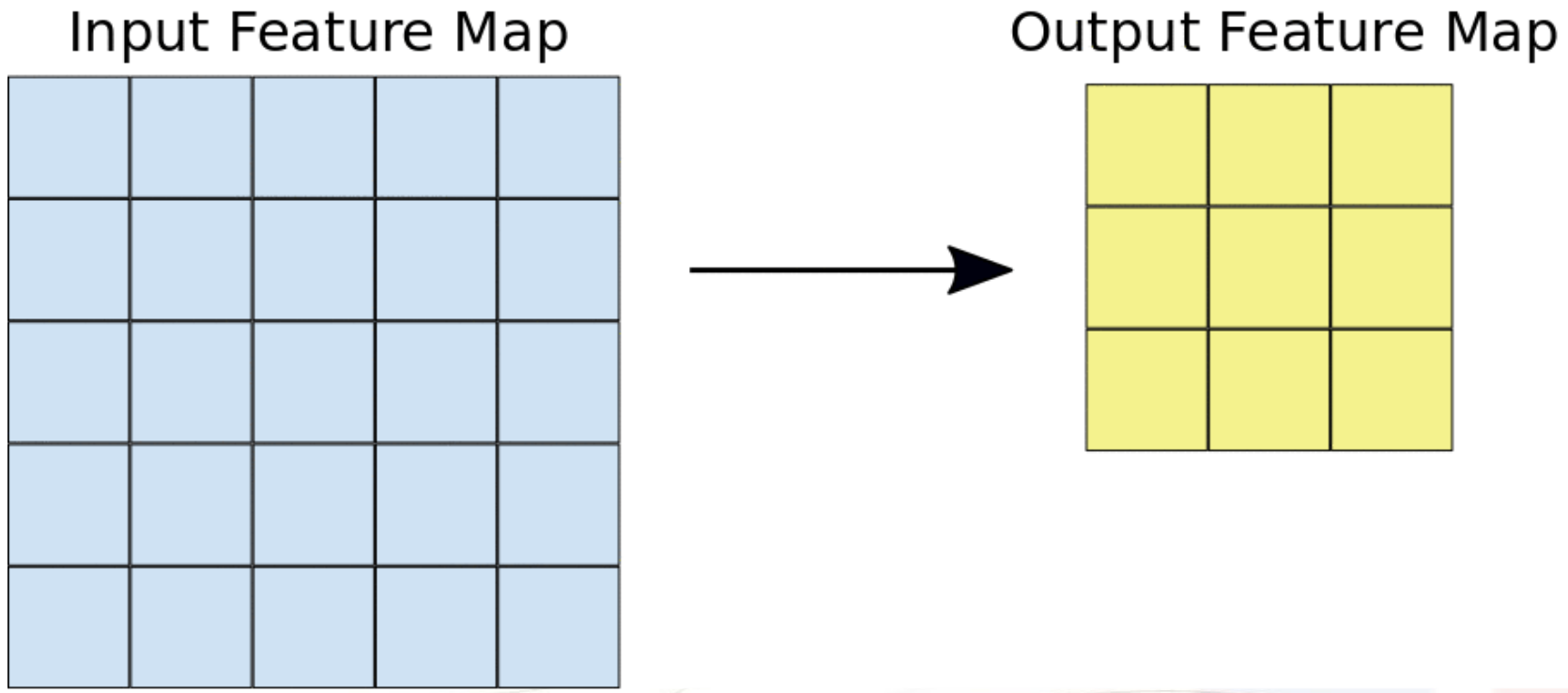
Идея свёрточных нейронных сетей заключается в чередовании свёрточных слоёв (англ. convolution layers) и субдискретизирующих слоёв (англ. subsampling layers или англ. pooling layers, слоёв подвыборки). Структура сети — однонаправленная (без обратных связей), принципиально многослойная. Для обучения используются стандартные методы, чаще всего **метод обратного распространения ошибки**. Функция активации нейронов (передаточная функция) — любая, по выбору исследователя.

1. Свертка

Свертка извлекает пиксели input feature map, и применяет к ним фильтры, чтобы вычислить новые функции, вычисляя функцию свертки (которая может иметь различный размер и глубину, чем input feature map). Свертки определяются двумя параметрами:

- Размер извлекаемых фрагментов (обычно 3x3 или 5x5 пикселей).
- Глубина выходной карты функций соответствует количеству применяемых фильтров.

Во время свертки фильтры (матрицы того же размера, что и размер плитки) эффективно скользят по сетке входной карты признаков по горизонтали и вертикали, по одному пикселю за раз, извлекая каждый соответствующую плитку



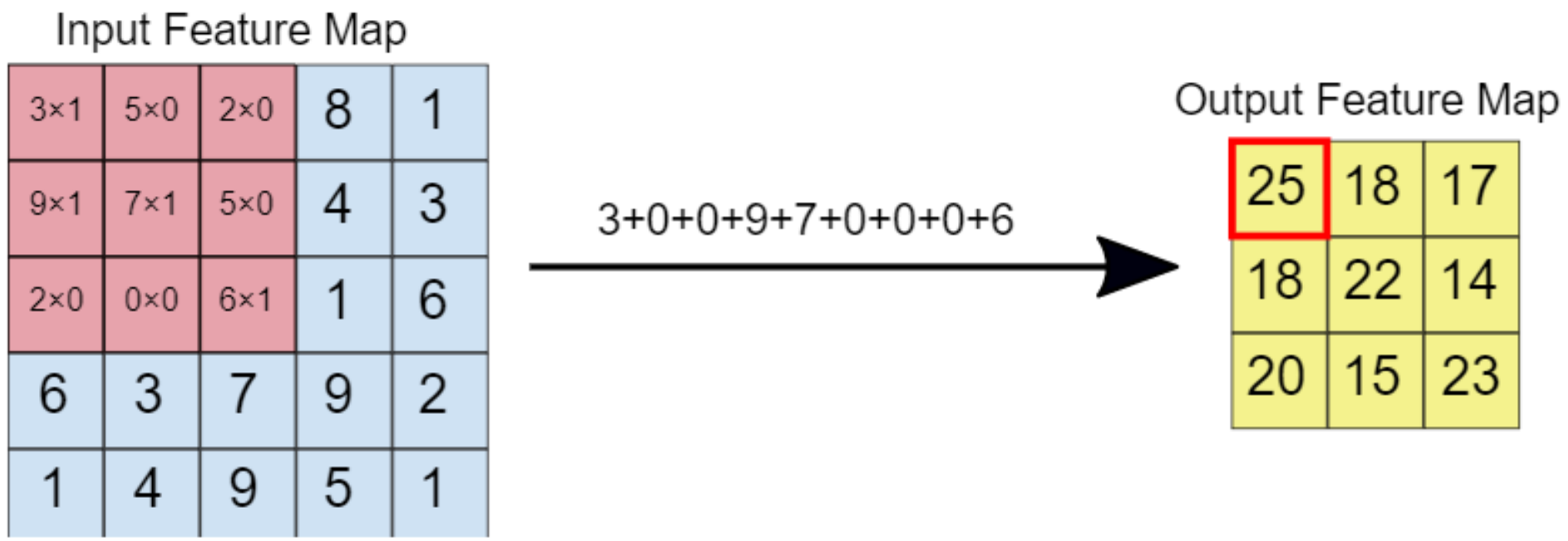
Для каждой пары фильтр-плитка CNN выполняет поэлементное умножение матрицы фильтра и матрицы изображения, а затем суммирует все элементы результирующей матрицы, чтобы получить одно значение. Каждое из этих результирующих значений для каждой пары фильтр-плитка затем выводится в новой матрице признаков

Input Feature Map

3	5	2	8	1
9	7	5	4	3
2	0	6	1	6
6	3	7	9	2
1	4	9	5	1

Convolutional Filter

1	0	0
1	1	0
0	0	1



Во время обучения CNN «изучает» оптимальные значения весовых коэффициентов (матрицы весовых коэффициентов) фильтров, которые позволяют извлекать значимые особенности (текстуры, края, формы) из входной карты признаков. По мере того, как количество фильтров (глубина выходной карты признаков), применяемых к входу, увеличивается, увеличивается и количество функций, которые CNN может извлечь. Однако компромисс заключается в том, что фильтры составляют большую часть ресурсов, расходуемых CNN, поэтому время обучения также увеличивается по мере добавления дополнительных фильтров. Кроме того, каждый фильтр, добавленный в сеть, обеспечивает меньшую пошаговую ценность, чем предыдущий, поэтому инженеры стремятся создавать сети, в которых используется минимальное количество фильтров, необходимых для извлечения функций, необходимых для точной классификации изображений.

2. ReLU

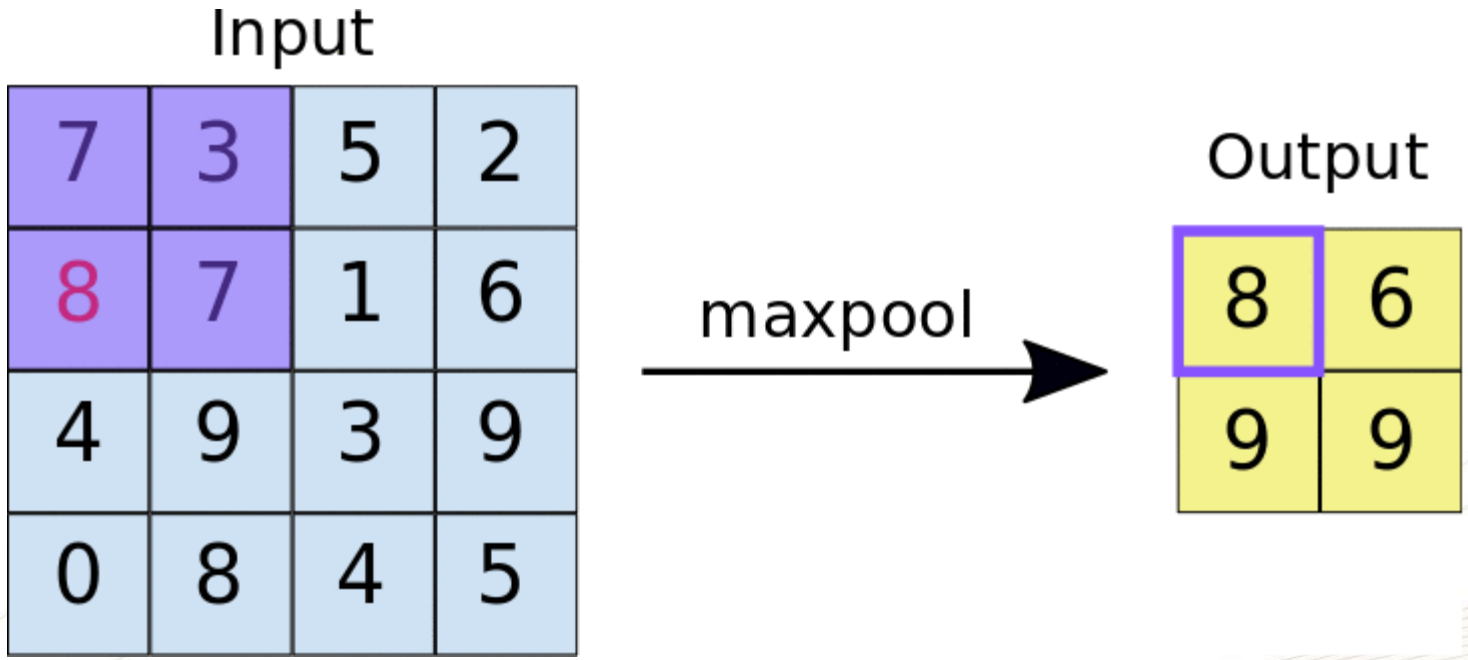
После каждой операции свертки CNN применяет преобразование Rectified Linear Unit (ReLU) к свернутому элементу, чтобы внести нелинейность в модель. Функция ReLU $F(x) = max(0, x)$, возвращает *x* для всех значений *x* > 0 и возвращает 0 для всех значений *x* ≤ 0.
ReLU используется как функция активации во множестве нейронных сетей.

3. Объединение

После ReLU идет этап объединения, на котором CNN понижает дискретизацию свернутой функции (для экономии времени обработки), уменьшая количество измерений карты функций, сохраняя при этом наиболее важную информацию о функциях. Общий алгоритм, используемый для этого процесса, называется max pooling.

max pooling работает аналогично свертке. Мы перемещаемся по карте функций и извлекаем фрагменты заданного размера. Для каждой плитки максимальное значение выводится на новую карту функций, а все остальные значения отбрасываются. Максимальные операции объединения принимают два параметра:

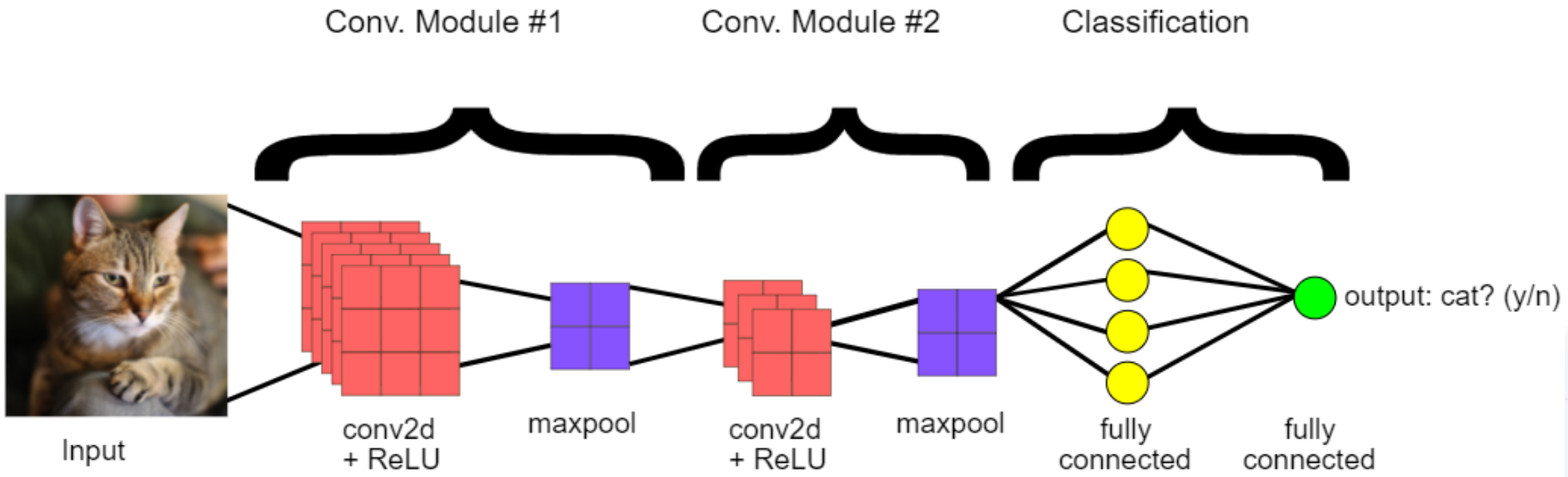
- Размер фильтра максимального объединения (обычно 2x2 пикселя)
- Шаг: расстояние в пикселях, разделяющее каждую извлеченную плитку. В отличие от свертки, когда фильтры перемещаются по карте признаков пиксель за пикселем, при максимальном объединении шаг определяет места, где извлекается каждый tiles. Для фильтра 2x2 шаг 2 указывает, что операция максимального объединения будет извлекать все неперекрывающиеся фрагменты 2x2 из карты функций



***Слева** : Max pooling, выполненное на карте функций 4x4 с фильтром 2x2 и шагом 2.*
***Справа** : результат операции максимального объединения. Обратите внимание, что получившаяся карта объектов теперь имеет размер 2x2, сохраняя только максимальные значения из каждой плитки.*

Полносвязные слои

В конце сверточной нейронной сети находится один или несколько полносвязных слоев (когда два слоя «полностью соединены», каждый узел первого уровня подключается к каждому узлу второго уровня). Их задача - выполнить классификацию на основе признаков, извлеченных свертками. Как правило, последний полностью связанный слой содержит функцию активации softmax, которая выводит значение вероятности от 0 до 1 для каждой из классификационных меток, которую модель пытается предсказать.

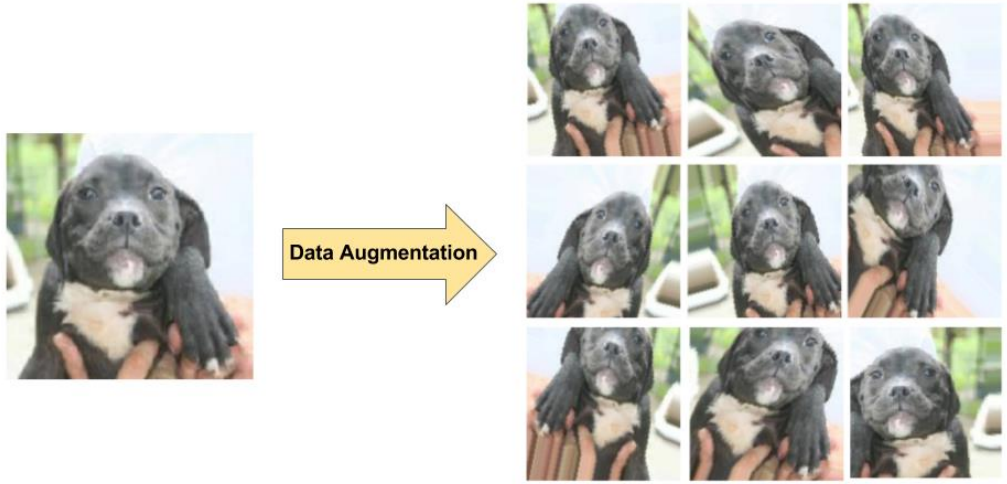


CNN содержит два модуля свертки (свертка + ReLU + объединение) для извлечения признаков и два полностью связанных слоя для классификации. Другие CNN могут содержать большее или меньшее количество сверточных модулей и большее или меньшее количество полностью связанных слоев. Инженеры часто экспериментируют, чтобы определить конфигурацию, которая дает наилучшие результаты для их модели.

Предотвращение переобучения

Как и в случае любой модели машинного обучения, ключевой проблемой при обучении сверточной нейронной сети является переобучение: модель настолько настроена на специфику обучающих данных, что ее невозможно обобщить на новые примеры.
Два метода предотвращения переобучения при построении CNN:

- Увеличение данных: искусственное увеличение разнообразия и количества обучающих примеров путем выполнения случайных преобразований существующих изображений для создания набора новых вариантов (см. рисунок). Увеличение данных особенно полезно, когда исходный набор обучающих данных относительно невелик.
- Dropout regularization: случайное удаление нейронов (а значит и синапсов) из нейронной сети на каждом шаге градиентного обучения.



*Увеличение данных на одном изображении собаки (взято из набора данных «Собаки против кошек», доступного на Kaggle). **Слева**: исходное изображение собаки из тренировочного набора. **Справа**: девять новых изображений, сгенерированных из исходного изображения с использованием случайных преобразований.*

Использование предварительно обученных моделей

Обучение сверточной нейронной сети выполнению задач классификации изображений обычно требует чрезвычайно большого количества обучающих данных и может занимать очень много времени, занимая дни или даже недели. Но что, если бы вы могли использовать существующие модели изображений, обученные на огромных наборах данных, например, с помощью TensorFlow-Slim, и адаптировать их для использования в ваших собственных задачах классификации?

Одним из распространенных методов использования предварительно обученных моделей **(трансферное обучение)** является извлечение признаков: получение промежуточных представлений, созданных предварительно обученной моделью, а затем передача этих представлений в новую модель в качестве входных данных. Например, если вы тренируете модель классификации изображений, чтобы различать разные типы овощей, вы можете передать обучающие изображения моркови, сельдерея и др. в предварительно обученную модель, а затем извлечь признаки из ее последнего сверточного слоя, которые собирают всю информацию, которую модель узнала об атрибутах более высокого уровня изображений: цвет, текстура, форма и т.д. Затем, при построении вашей новой модели классификации, вместо того, чтобы начинать с необработанных пикселей, вы можете использовать эти извлеченные функции в качестве входных данных, и добавьте свои полносвязные слои классификации в конце. Чтобы повысить производительность при использовании извлечения признаков с предварительно обученной моделью, инженеры часто проводят тонкую настройку весовых коэффициентов (параметров), применяемые к извлеченным объектам.

Преимущества

- Один из лучших алгоритмов по распознаванию и классификации изображений.
- По сравнению с полносвязной нейронной сетью (типа перцептрона) — гораздо меньшее количество настраиваемых весов, так как одно ядро весов используется целиком для всего изображения, вместо того, чтобы делать для каждого пикселя входного изображения свои персональные весовые коэффициенты. Это подталкивает нейросеть при обучении к обобщению демонстрируемой информации, а не попиксельному запоминанию каждой показанной картинки в мириадах весовых коэффициентов, как это делает перцептрон.
- Удобное распараллеливание вычислений, а следовательно, возможность реализации алгоритмов работы и обучения сети на графических процессорах.
- Относительная устойчивость к повороту и сдвигу распознаваемого изображения.
- Обучение при помощи классического метода обратного распространения ошибки.

Недостатки

- Слишком много варьируемых параметров сети; непонятно, для какой задачи и вычислительной мощности какие нужны настройки. Так, к варьируемым параметрам можно отнести: количество слоёв, размерность ядра свёртки для каждого из слоёв, количество ядер для каждого из слоёв, шаг сдвига ядра при обработке слоя, необходимость слоёв субдискретизации, степень уменьшения ими размерности, функция по уменьшению размерности (выбор максимума, среднего и т. п.), передаточная функция нейронов, наличие и параметры выходной полносвязной нейросети на выходе свёрточной. Все эти параметры существенно влияют на результат, но выбираются исследователями эмпирически.
- Однако существует несколько выверенных и прекрасно работающих конфигураций сетей, но не хватает рекомендаций, по которым нужно строить сеть для новой задачи.