

References

CS231n: Convolutional Neural Networks for Visual Recognition



Fei-Fei Li & Andrej Karpathy & Justin Johnson

Fast R-CNN

Rich feature hierarchies for accurate object detection and semantic segmentation

Ross Girshick
Microsoft Research
rbg@microsoft.com

Ross Girshick Jeff Donahue Trevor Darrell Jitendra Malik
UC Berkeley
{rbg,jdonahue,trevor,malik}@eecs.berkeley.edu

Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun

Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun

Selective Search for Object Recognition

J.R.R. Uijlings^{*1,2}, K.E.A. van de Sande^{†2}, T. Gevers², and A.W.M. Smeulders²

¹University of Trento, Italy

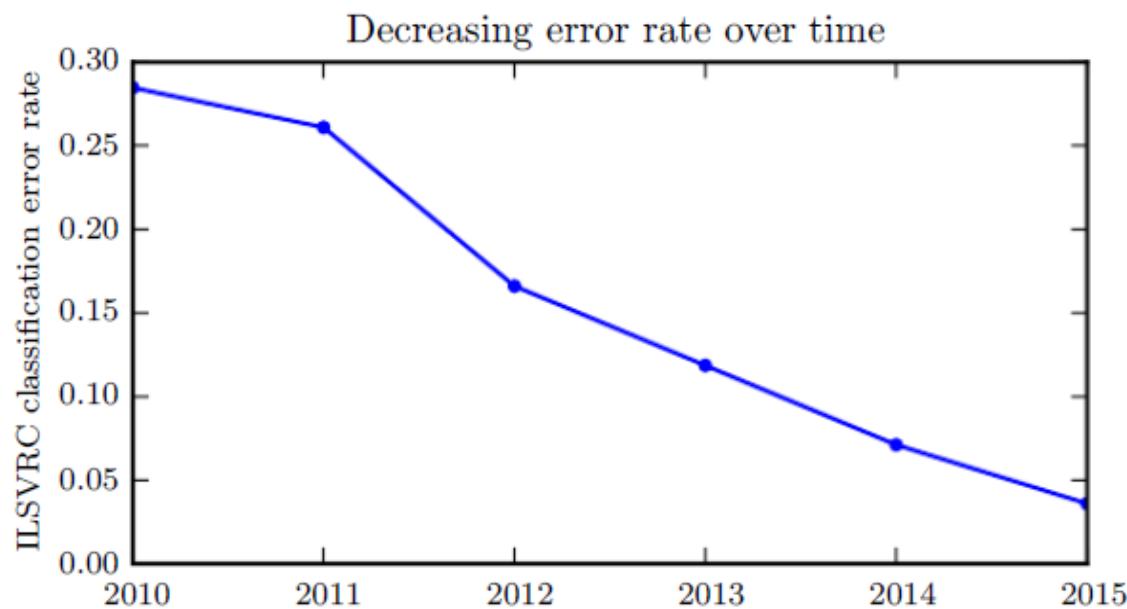
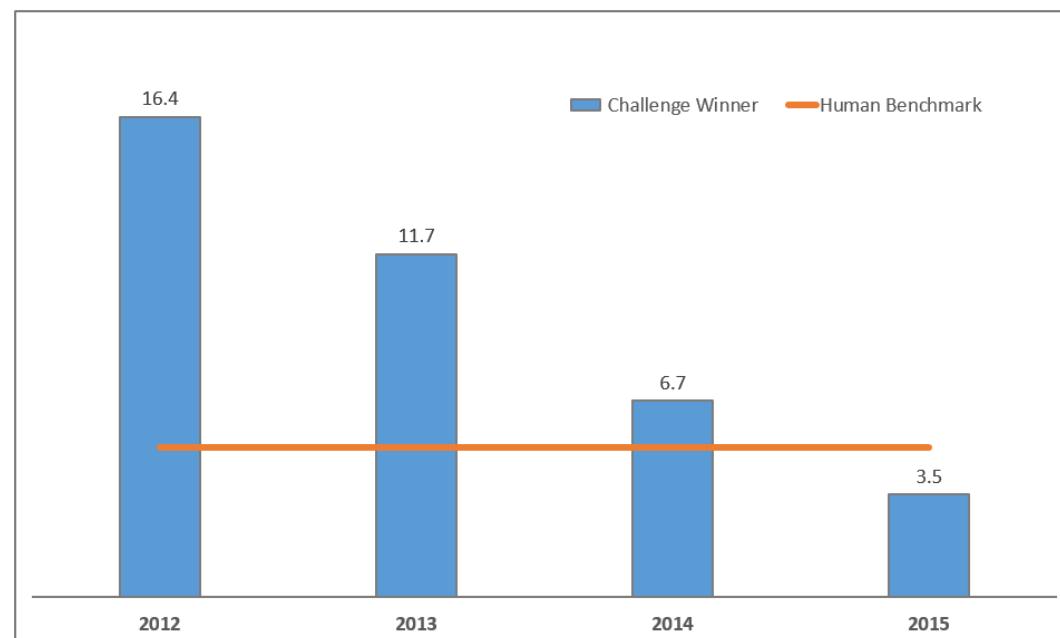
²University of Amsterdam, the Netherlands

Fully Convolutional Networks for Semantic Segmentation

Jonathan Long* Evan Shelhamer* Trevor Darrell
UC Berkeley
{jonlong,shelhamer,trevor}@cs.berkeley.edu

“Кульминационной точкой в стремительном росте глубокого обучения был момент , когда сверточная нейронная сеть впервые и с большим отрывом справилась с этой задачей, снизив уровень ошибок с 26,1% до 15,3%.

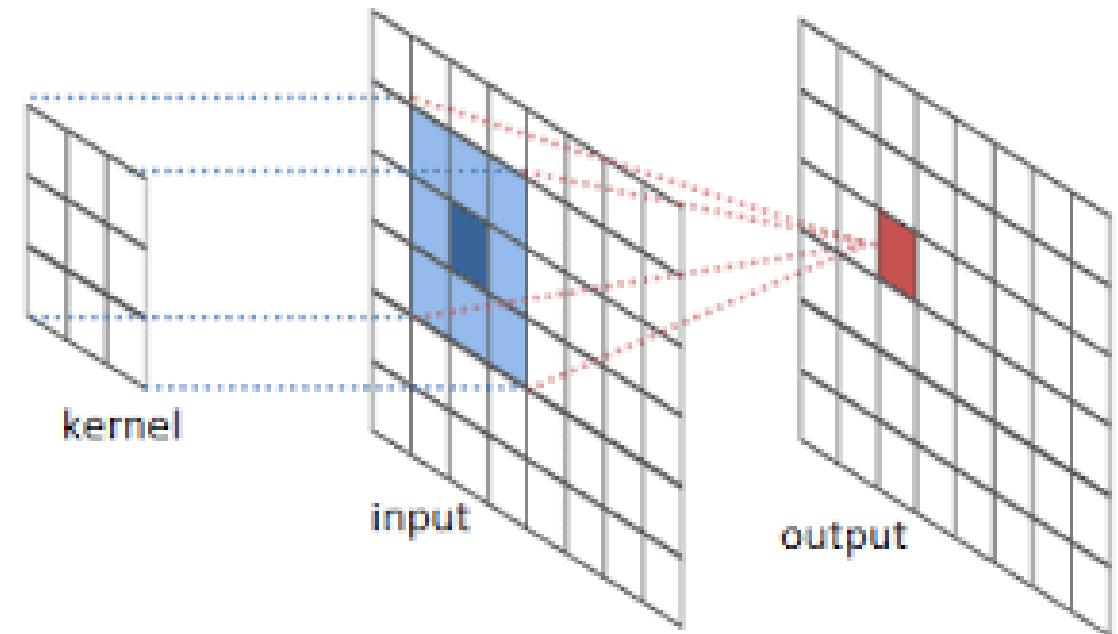
(Krizhevsky *et al.*, 2012), это означает, что сверточная нейронная сеть создает список возможных категорий/классов для каждого изображения, и правильная категория появляется уже в первых пяти записях этого списка для всех, кроме 15,3% тестовых примеров.. **С тех пор, победителем подобных соревнований всегда выходила сверточная нейронная сеть**, и на момент написания статьи, достижения в области глубокого обучения привели к тому, что ошибки 5 лучших результатов в этом конкурсе снизились до 3,6% ». – Ref: Deep Learning Book by Y Bengio et al



Что такое сверточная нейронная сеть?

Сверточные сети - это просто нейронные сети, которые используют свертку вместо общего умножения матриц, по крайней мере, на одном из их слоев.

Свертка - это математическая операция, имеющая линейную форму



Типы входов

- Входы имеют структуру
 - Цветные изображения трехмерны и поэтому имеют объем
 - Речевые сигналы во временной области являются 1-мерными, в то время как представления в частотной области (например, MFCC) принимают 2-мерную форму. Их также можно рассматривать как временную последовательность.
 - Медицинские изображения (такие как КТ / MR / и т.д.) являются многомерными
 - Видеоролики имеют дополнительное временное измерение по сравнению со стационарными изображениями
 - Речевые сигналы можно смоделировать как двумерные
 - Последовательности переменной длины и данные временных рядов также многомерны
- Таким образом, имеет смысл моделировать их как тензоры вместо векторов.
- Классификатор должен принять тензор в качестве входных данных и выполнить необходимый процесс машинного обучения. В случае изображения этот тензор будет выражать объем.

Сверточные нейронные сети повсюду

- Поиск изображений
- обнаружение
- Автопилотируемые автомобили
- Семантическая сегментация
- Определение лица
- Оценка позы
- Определение заболеваний
- Распознавание речи
- Обработка текста
- Обработка данных со спутника

Применение СНН к изображениям

- Почему для обработки изображений лучше всего подходит СНН?
- Пиксели на изображении соотносятся друг с другом. Однако соседние пиксели коррелируют с более сильными, а удаленные пиксели не влияют не так сильно
Субъективные особенности имеют значение: Субъективные поля восприятия
- Аффинные преобразования: класс изображения не меняется с переводом. Мы можем создать детектор объектов, который может искать конкретный объект (например, ребро) в любом месте плоскости изображения, перемещаясь поперек. Сверточный слой может иметь несколько таких фильтров, составляющих измерение глубины слоя.

ПОЛНОСТЬЮ СВЯЗАННЫЕ СЛОИ

- Полностью связанные слои (такие как скрытые слои традиционной нейронной сети) не зависят от структуры ввода
 - Они принимают входные данные как векторы и генерируют выходной вектор
 - Нет необходимости обмениваться параметрами, если это не навязано в определенных архитектурах. Это увеличивает количество параметров при увеличении входных и / или выходных размеров.
- Предположим, мы должны выполнить классификацию изображения размером 100x100x3.
- Если мы реализуем использование нейронной сети с прямой связью, которая имеет входной, скрытый и выходной слои, где: скрытые единицы (n_h) = 1000, выходные классы = 10:
 - Входной слой = $10k$ пикселей * 3 = $30k$, весовая матрица для скрытого для входного слоя = $1k * 30k = 30$ М и размер матрицы выходного слоя = $10 * 1000 = 10k$
 - Мы можем решить эту проблему путем извлечения функций, использующих предварительную обработку и предоставления низкоразмерных входных данных для нейронной сети. Но для этого нужны специальные инженерные функции и, следовательно, знание предметной области.

Свертывание

Свертывание в одномерном пространстве:

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n - k]$$

Свертывание в двумерном пространстве:

$$y[n_1, n_2] = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} x[k_1, k_2]h[(n_1 - k_1), (n_2 - k_2)]$$

Input

a	b	c	d
e	f	g	h
i	j	k	l

Kernel

w	x
y	z

Output

$$aw + bx +$$

$$ey + fz +$$

$$bw + cx +$$

$$fy + gz +$$

$$cw + dx +$$

$$gy + hz +$$

$$ew + fx +$$

$$iy + jz +$$

$$fw + gx +$$

$$jy + kz +$$

$$gw + hx +$$

$$ky + lz +$$

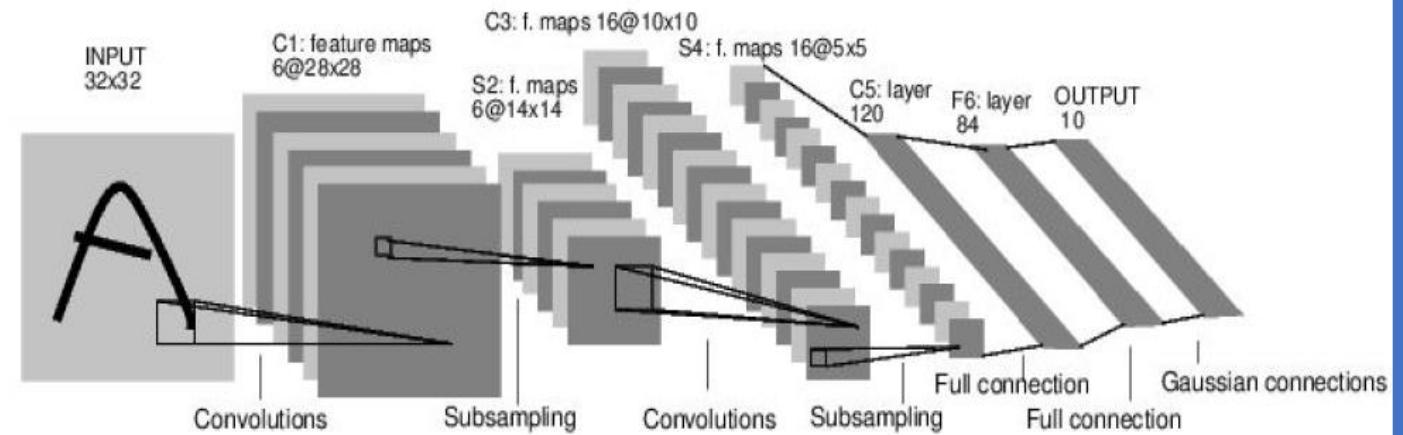
СНН

Типы слоев

- Слой свертывания
- Слой пула
- Полностью связанный слой

Case Study: LeNet-5

[LeCun et al., 1998]



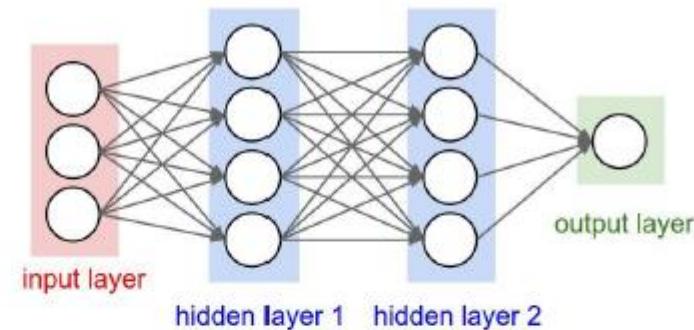
Conv filters were 5x5, applied at stride 1

Subsampling (Pooling) layers were 2x2 applied at stride 2
i.e. architecture is [CONV-POOL-CONV-POOL-CONV-FC]

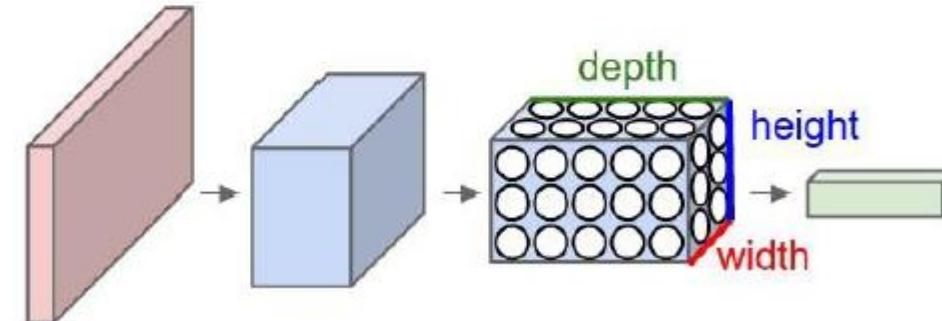
Слой свертывания

- Слой в обычной нейронной сети принимает на вход вектор и выводит, также, вектор.
- Сверточный слой принимает тензор (3д объем для изображений RGB) в качестве входных данных и генерирует тензор в качестве выходных данных

Regular neural network (fully connected):



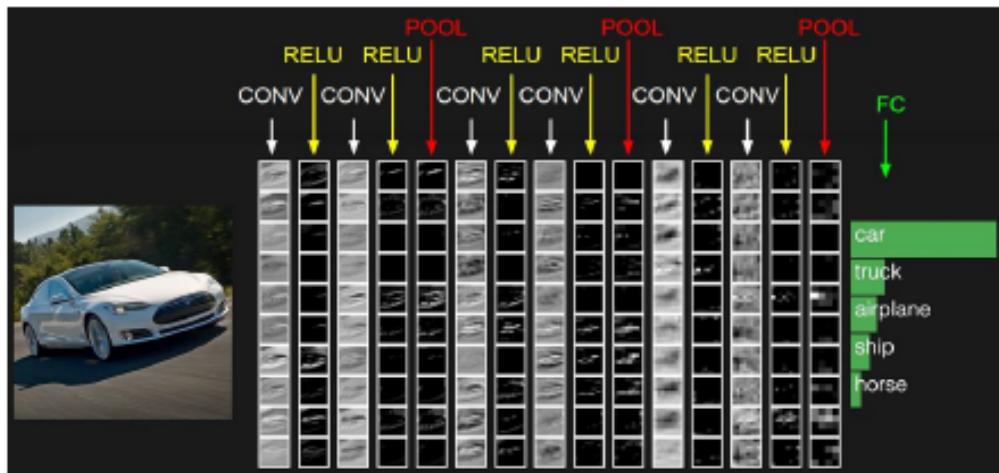
Convolutional neural network:



Each layer takes a 3d volume, produces 3d volume with some smooth function that may or may not have parameters.

Convolutional Neural Networks: Layers

- **INPUT** [32x32x3] will hold the raw pixel values of the image, in this case an image of width 32, height 32, and with three color channels R,G,B.
- **CONV** layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume. This may result in volume such as [32x32x12] if we decided to use 12 filters.
- **RELU** layer will apply an elementwise activation function, such as the $\max(0,x)$ thresholding at zero. This leaves the size of the volume unchanged ([32x32x12]).
- **POOL** layer will perform a downsampling operation along the spatial dimensions (width, height), resulting in volume such as [16x16x12].
- **FC** (i.e. fully-connected) layer will compute the class scores, resulting in volume of size [1x1x10], where each of the 10 numbers correspond to a class score, such as among the 10 categories of CIFAR-10. As with ordinary Neural Networks and as the name implies, each neuron in this layer will be connected to all the numbers in the previous volume.



Вход содержит значения пикселей изображения , в данном случае высотой в 32, шириной в 32 и трехканальной системой цветов RGB

Слой свертывания вычисляет выходные значение нейронов, соединенных в ближней области, для каждого из них вычисляя скалярное произведение между весами и небольшой областью, с которой они соединены во входном объеме.

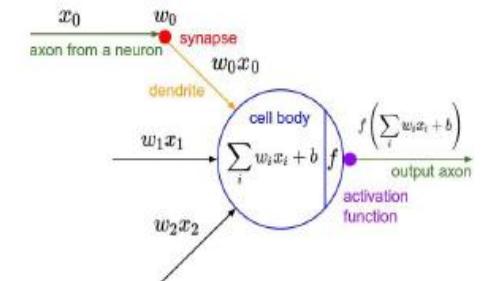
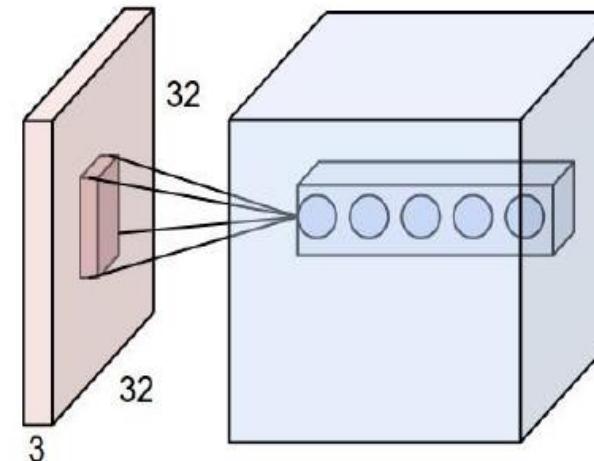
Слой выпрямленной линейной единицы применяет функцию активации поэлементно, например $\max(0,x)$ с порогом в нуле. Этот этап не изменяет значение объема ([32x32x12])

Последностью соединенный слой подсчитает классовые очки, полученные в виде объема (1x1x10), где каждый из десяти чисел соответствует числу очков определенного класса, как среди 10-ти категорий CIFAR-10. Как и в случае с обычной нейросетью, каждый нейрон этого слоя соединен со всеми числами предыдущего объема

Локальные рецептивные поля

- Фильтр (ядро) применяется к входному изображению как скользящее окно по ширине и высоте.
- Глубина фильтра совпадает с глубиной ввода.
- Для каждой позиции фильтра вычисляется скалярное произведение фильтра и входные данные.
(Активация)
- Двухмерное расположение этих активаций называется картой активации.
- Количество таких фильтров составляет глубину свертки слоя

Dealing with Images: Local Connectivity



Same neuron. Just more focused (narrow “receptive field”).

The parameters on each filter are spatially “shared”
(if a feature is useful in one place, it’s useful elsewhere)

Fig Credit: Lex Fridman, MIT, 6.S094

Операция свертки между фильтром и изображением

- Слой свертки вычисляет скалярные произведения между фильтром и частью изображения, когда он скользит вдоль изображения
- Размер шага слайда называется шагом
- Без какого-либо заполнения процесс свертки уменьшает пространственные размеры выходного значения

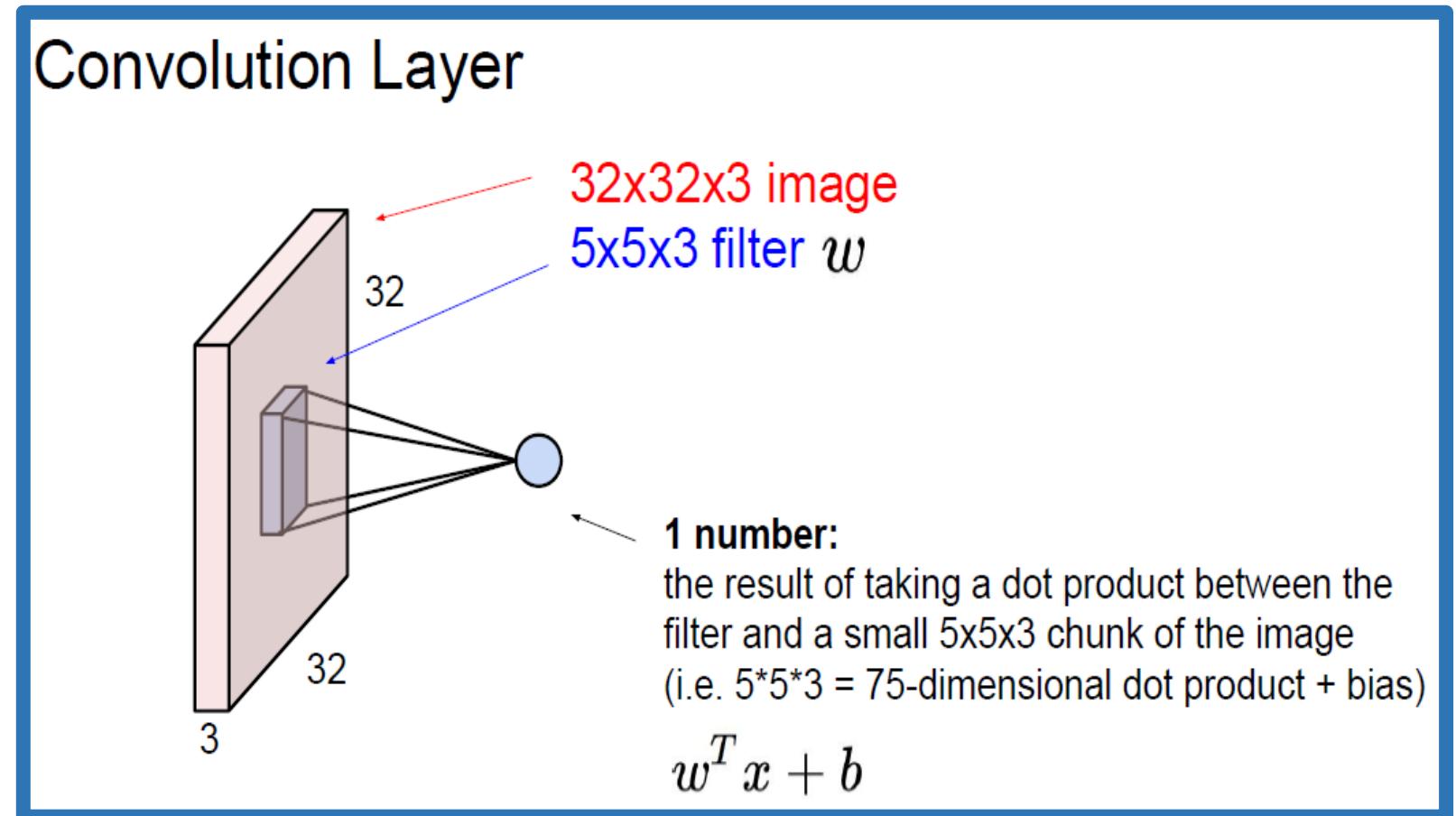


Fig Credit: A Karpathy, CS231n

Карты активации

- Пример:
 - Рассмотрим изображение размером $32 \times 32 \times 3$ и фильтр $5 \times 5 \times 3$.
 - Свертка происходит между $5 \times 5 \times 3$ фрагментом изображения с фильтром: $wTx + b$
 - В этом примере мы получаем 75-мерный вектор и элемент смещения
 - В этом примере с шагом 1 мы получаем активацию $28 \times 28 \times 1$ для 1 фильтра без заполнения
 - Если бы у нас было 6 фильтров, мы бы получили $28 \times 28 \times 6$ без заполнения
- В приведенном выше примере у нас есть карта активации 28×28 на фильтр.
- Карты активации являются входными данными для следующего слоя сети.
- Без какого-либо дополнения, площадь поверхности 2D карты активации меньше, чем площадь входной поверхности для шага ≥ 1

Укладка сверточных слоев

Preview: ConvNet is a sequence of Convolutional Layers, interspersed with activation functions

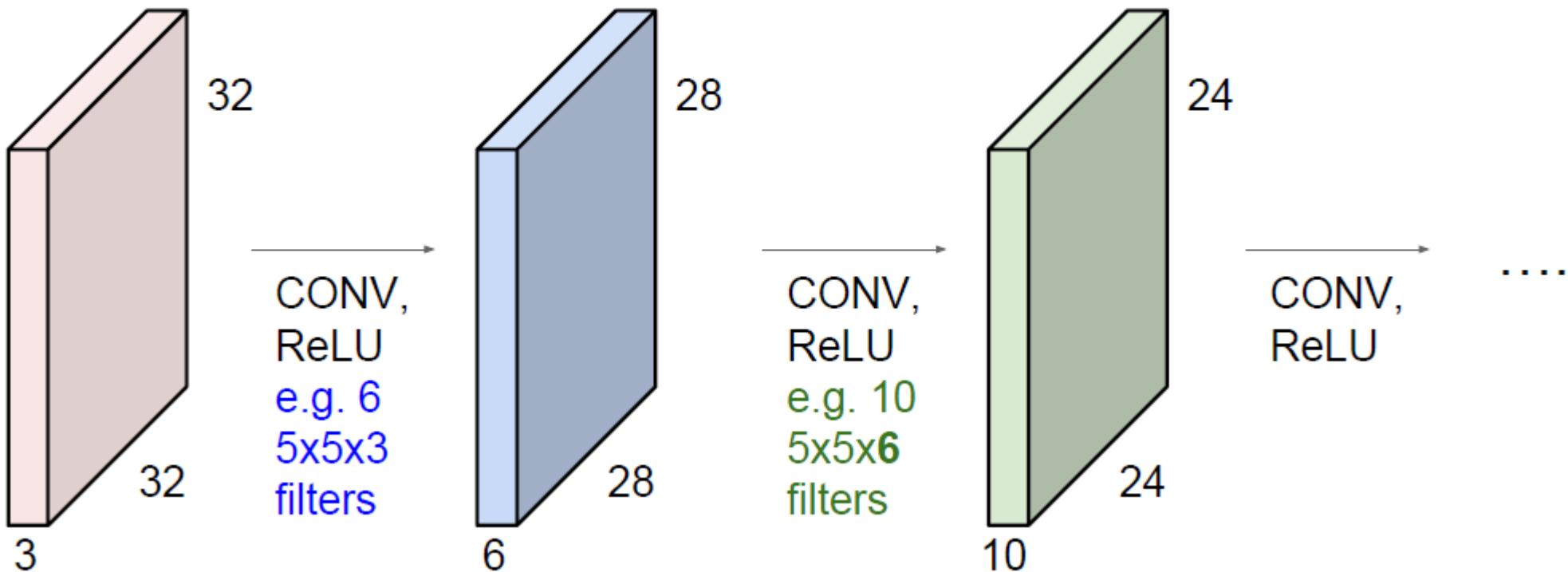
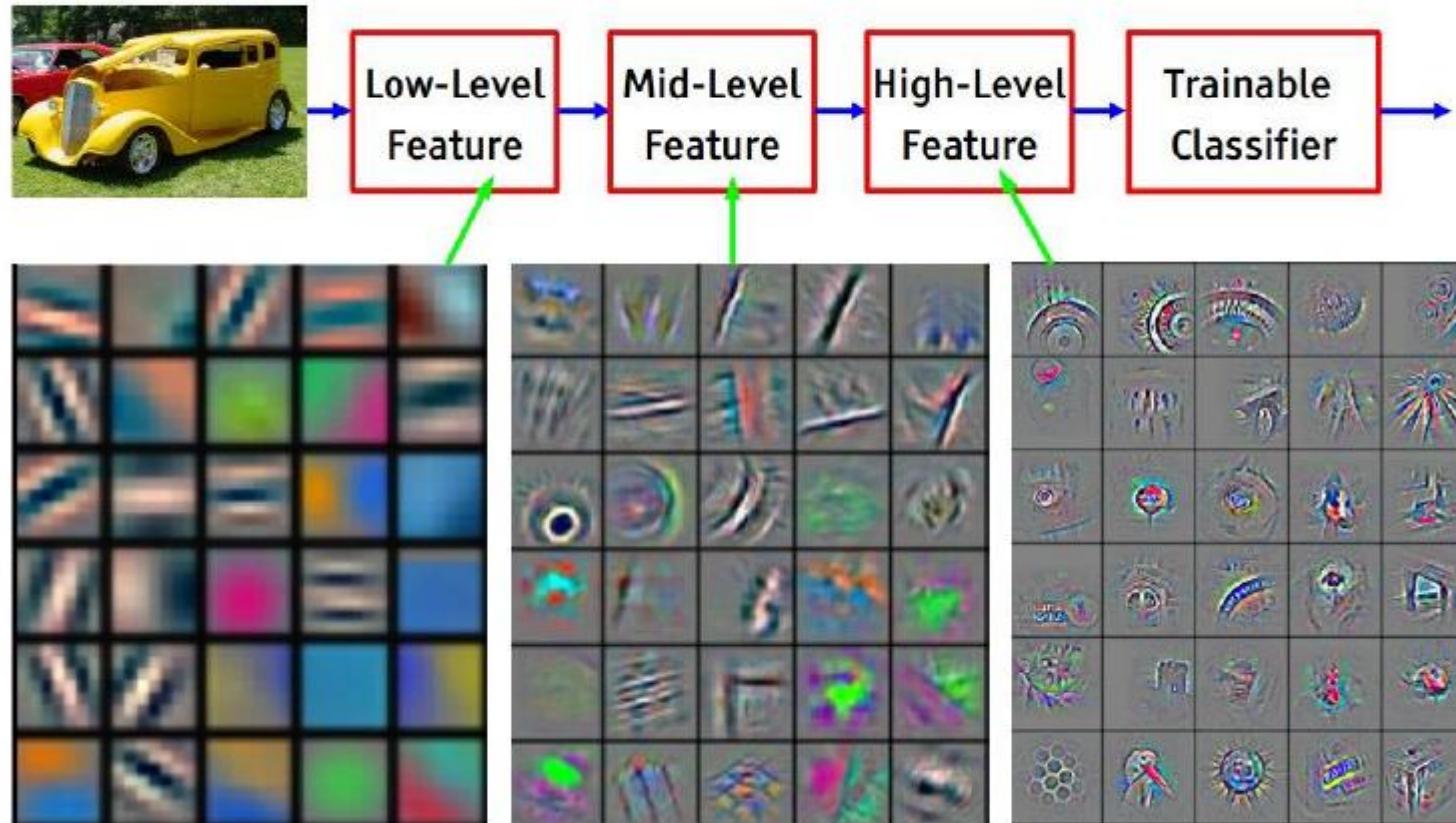


Fig Credit: A Karpathy, CS231n

Представление объекта в виде иерархии

[From recent Yann LeCun slides]



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Заполнение

- Пространственный (x, y) экстент выходного сигнала, создаваемого сверточным слоем, меньше соответствующих размеров входного сигнала (за исключением специального случая фильтра 1×1 с шагом 1).
- По мере добавления большего количества слоев и использования больших шагов размеры выходной поверхности продолжают уменьшаться, и это может повлиять на точность.
- Зачастую, нам может потребоваться сохранить пространственный экстент во время начальных слоев и уменьшить их в более позднее время.
- Заполнение ввода подходящими значениями (обычно заполнение нулями) помогает сохранить пространственный размер

Нулевое заполнение границы

0	0	0	0	0	0			
0								
0								
0								
0								

e.g. input 7×7

3×3 filter, applied with **stride 1**

pad with 1 pixel border => what is the output?

7×7 output!

in general, common to see CONV layers with stride 1, filters of size $F \times F$, and zero-padding with $(F-1)/2$. (will preserve size spatially)

e.g. $F = 3 \Rightarrow$ zero pad with 1

$F = 5 \Rightarrow$ zero pad with 2

$F = 7 \Rightarrow$ zero pad with 3

Гиперпараметры сверточного слоя

- Размер фильтра

Summary. To summarize, the Conv Layer:

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
 - Number of filters K ,
 - their spatial extent F ,
 - the stride S ,
 - the amount of zero padding P .
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
 - $W_2 = (W_1 - F + 2P)/S + 1$
 - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
 - $D_2 = K$
- With parameter sharing, it introduces $F \cdot F \cdot D_1$ weights per filter, for a total of $(F \cdot F \cdot D_1) \cdot K$ weights and K biases.
- In the output volume, the d -th depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the d -th filter over the input volume with a stride of S , and then offset by d -th bias.

Common settings:

- $K = (\text{powers of 2, e.g. } 32, 64, 128, 512)$
- $F = 3, S = 1, P = 1$
 - $F = 5, S = 1, P = 2$
 - $F = 5, S = 2, P = ?$ (whatever fits)
 - $F = 1, S = 1, P = 0$

- # Фильтры

- Шаг

- Заполнение

Слой пула

- Объединение - операция понижающей выборки
- Обоснование состоит в том, что «значение», встроенное в фрагмент изображения, может быть зафиксировано с использованием небольшого подмножества «важных» пикселей
- Максимальный пул и средний пул являются двумя наиболее распространенными операциями
- Слой пула не имеет обучаемых параметров

Pooling layer

- makes the representations smaller and more manageable
- operates over each activation map independently:

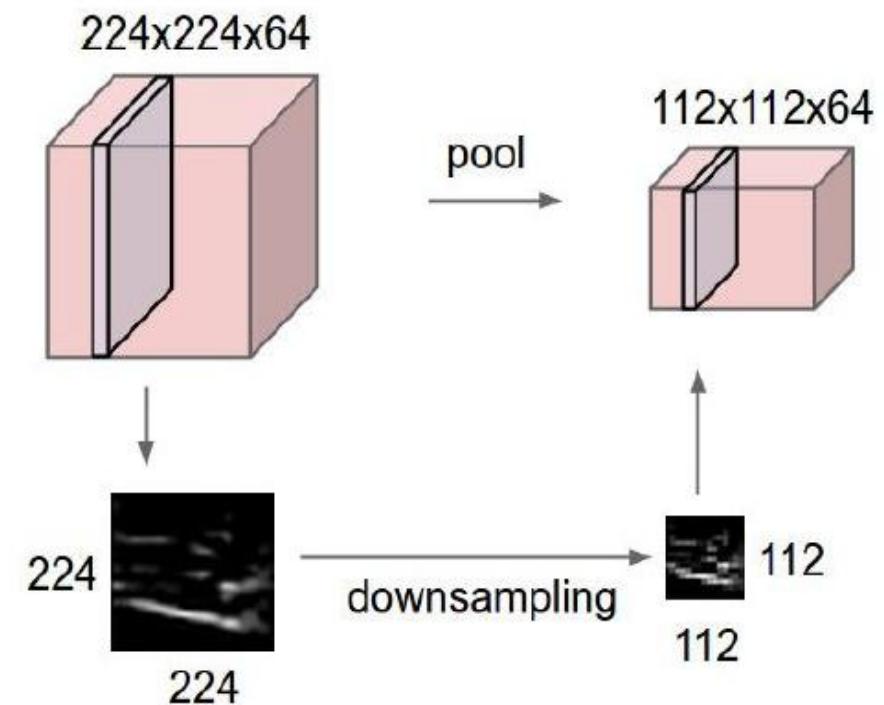
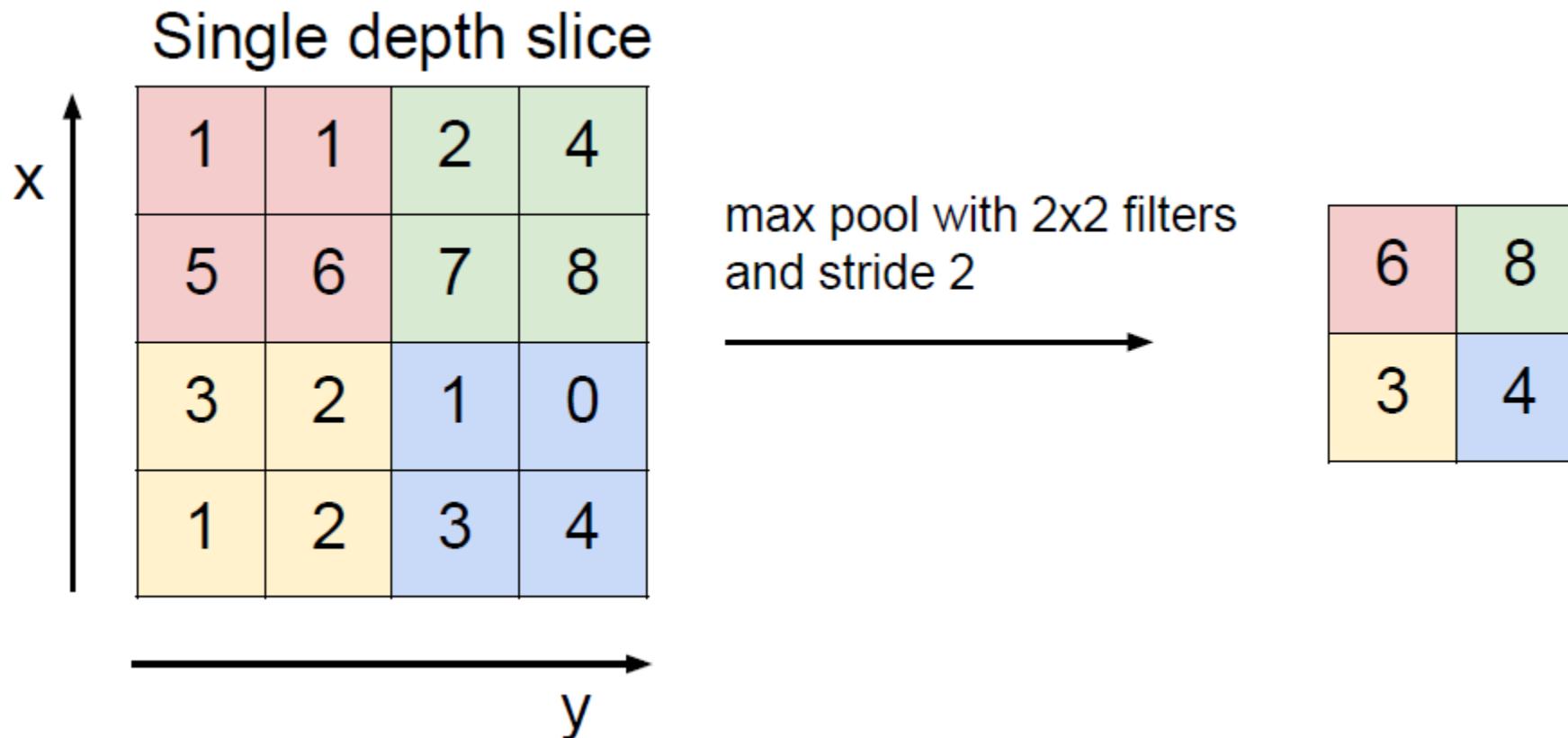


Fig Credit: A Karpathy, CS231n

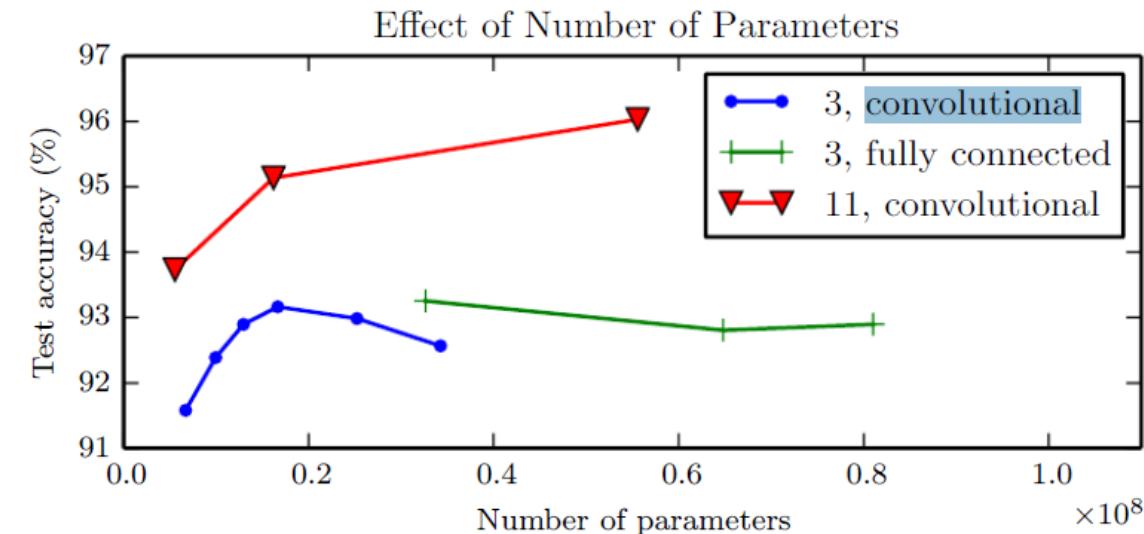
Иллюстрация максимального пула



Популярные сетевые архитектуры

Текущая тенденция: более глубокие модели

- CNN полностью превосходят другие подходы для основных задач CV
- Более глубокие модели работают лучше
- Увеличение количества параметров в слоях CNN без увеличения их глубины не эффективно при увеличении производительности тестового набора.
- Мелкие модели соответствуют более 20 миллионам параметров, в то время как глубокие модели могут использовать более 60 миллионов.
- Ключевой момент: модель работает лучше, когда она спроектирована так, чтобы отражать состав более простых функций, чем одна сложная функция. Это также можно объяснить, рассматривая вычисления как цепочку зависимостей.



Case Study: AlexNet

[Krizhevsky et al. 2012]

Full (simplified) AlexNet architecture:

[227x227x3] INPUT

[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0

[27x27x96] MAX POOL1: 3x3 filters at stride 2

[27x27x96] NORM1: Normalization layer

[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2

[13x13x256] MAX POOL2: 3x3 filters at stride 2

[13x13x256] NORM2: Normalization layer

[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1

[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1

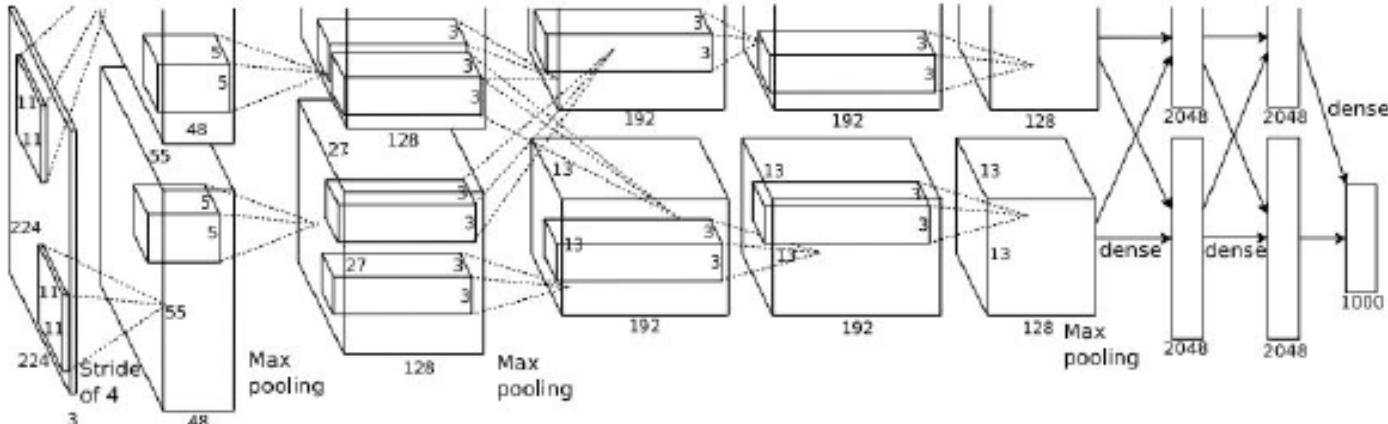
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1

[6x6x256] MAX POOL3: 3x3 filters at stride 2

[4096] FC6: 4096 neurons

[4096] FC7: 4096 neurons

[1000] FC8: 1000 neurons (class scores)



Details/Retrospectives:

- first use of ReLU
- used Norm layers (not common anymore)
- heavy data augmentation
- dropout 0.5
- batch size 128
- SGD Momentum 0.9
- Learning rate 1e-2, reduced by 10 manually when val accuracy plateaus
- L2 weight decay 5e-4
- 7 CNN ensemble: 18.2% -> 15.4%

VGG Net

INPUT: [224x224x3] memory: $224 \times 224 \times 3 = 150K$ params: 0 (not counting biases)

CONV3-64: [224x224x64] memory: $224 \times 224 \times 64 = 3.2M$ params: $(3 \times 3 \times 3) \times 64 = 1,728$

CONV3-64: [224x224x64] memory: $224 \times 224 \times 64 = 3.2M$ params: $(3 \times 3 \times 64) \times 64 = 36,864$

POOL2: [112x112x64] memory: $112 \times 112 \times 64 = 800K$ params: 0

CONV3-128: [112x112x128] memory: $112 \times 112 \times 128 = 1.6M$ params: $(3 \times 3 \times 64) \times 128 = 73,728$

CONV3-128: [112x112x128] memory: $112 \times 112 \times 128 = 1.6M$ params: $(3 \times 3 \times 128) \times 128 = 147,456$

POOL2: [56x56x128] memory: $56 \times 56 \times 128 = 400K$ params: 0

CONV3-256: [56x56x256] memory: $56 \times 56 \times 256 = 800K$ params: $(3 \times 3 \times 128) \times 256 = 294,912$

CONV3-256: [56x56x256] memory: $56 \times 56 \times 256 = 800K$ params: $(3 \times 3 \times 256) \times 256 = 589,824$

CONV3-256: [56x56x256] memory: $56 \times 56 \times 256 = 800K$ params: $(3 \times 3 \times 256) \times 256 = 589,824$

POOL2: [28x28x256] memory: $28 \times 28 \times 256 = 200K$ params: 0

CONV3-512: [28x28x512] memory: $28 \times 28 \times 512 = 400K$ params: $(3 \times 3 \times 256) \times 512 = 1,179,648$

CONV3-512: [28x28x512] memory: $28 \times 28 \times 512 = 400K$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$

CONV3-512: [28x28x512] memory: $28 \times 28 \times 512 = 400K$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$

POOL2: [14x14x512] memory: $14 \times 14 \times 512 = 100K$ params: 0

CONV3-512: [14x14x512] memory: $14 \times 14 \times 512 = 100K$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$

CONV3-512: [14x14x512] memory: $14 \times 14 \times 512 = 100K$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$

CONV3-512: [14x14x512] memory: $14 \times 14 \times 512 = 100K$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$

POOL2: [7x7x512] memory: $7 \times 7 \times 512 = 25K$ params: 0

FC: [1x1x4096] memory: 4096 params: $7 \times 7 \times 512 \times 4096 = 102,760,448$

FC: [1x1x4096] memory: 4096 params: $4096 \times 4096 = 16,777,216$

FC: [1x1x1000] memory: 1000 params: $4096 \times 1000 = 4,096,000$

TOTAL memory: $24M * 4 \text{ bytes} \approx 93\text{MB} / \text{image}$ (only forward! ~ 2 for bwd)

TOTAL params: 138M parameters

ConvNet Configuration		
B	C	D
13 weight layers	16 weight layers	16 weight layers
put (224 × 224 RGB image)		
conv3-64	conv3-64	conv3-64
conv3-64	conv3-64	conv3-64
maxpool		
conv3-128	conv3-128	conv3-128
conv3-128	conv3-128	conv3-128
maxpool		
conv3-256	conv3-256	conv3-256
conv3-256	conv3-256	conv3-256
conv1-256	conv3-256	conv3-256
maxpool		
conv3-512	conv3-512	conv3-512
conv3-512	conv3-512	conv3-512
conv1-512	conv3-512	conv3-512
maxpool		
conv3-512	conv3-512	conv3-512
conv3-512	conv3-512	conv3-512
conv1-512	conv3-512	conv3-512
maxpool		
FC-4096		
FC-4096		
FC-1000		
soft-max		

VGG net

INPUT: [224x224x3] memory: $224 \times 224 \times 3 = 150K$ params: 0 (not counting biases)

CONV3-64: [224x224x64] memory: $224 \times 224 \times 64 = 3.2M$ params: $(3 \times 3 \times 3) \times 64 = 1,728$

CONV3-64: [224x224x64] memory: $224 \times 224 \times 64 = 3.2M$ params: $(3 \times 3 \times 64) \times 64 = 36,864$

POOL2: [112x112x64] memory: $112 \times 112 \times 64 = 800K$ params: 0

CONV3-128: [112x112x128] memory: $112 \times 112 \times 128 = 1.6M$ params: $(3 \times 3 \times 64) \times 128 = 73,728$

CONV3-128: [112x112x128] memory: $112 \times 112 \times 128 = 1.6M$ params: $(3 \times 3 \times 128) \times 128 = 147,456$

POOL2: [56x56x128] memory: $56 \times 56 \times 128 = 400K$ params: 0

CONV3-256: [56x56x256] memory: $56 \times 56 \times 256 = 800K$ params: $(3 \times 3 \times 128) \times 256 = 294,912$

CONV3-256: [56x56x256] memory: $56 \times 56 \times 256 = 800K$ params: $(3 \times 3 \times 256) \times 256 = 589,824$

CONV3-256: [56x56x256] memory: $56 \times 56 \times 256 = 800K$ params: $(3 \times 3 \times 256) \times 256 = 589,824$

POOL2: [28x28x256] memory: $28 \times 28 \times 256 = 200K$ params: 0

CONV3-512: [28x28x512] memory: $28 \times 28 \times 512 = 400K$ params: $(3 \times 3 \times 256) \times 512 = 1,179,648$

CONV3-512: [28x28x512] memory: $28 \times 28 \times 512 = 400K$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$

CONV3-512: [28x28x512] memory: $28 \times 28 \times 512 = 400K$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$

POOL2: [14x14x512] memory: $14 \times 14 \times 512 = 100K$ params: 0

CONV3-512: [14x14x512] memory: $14 \times 14 \times 512 = 100K$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$

CONV3-512: [14x14x512] memory: $14 \times 14 \times 512 = 100K$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$

CONV3-512: [14x14x512] memory: $14 \times 14 \times 512 = 100K$ params: $(3 \times 3 \times 512) \times 512 = 2,359,296$

POOL2: [7x7x512] memory: $7 \times 7 \times 512 = 25K$ params: 0

FC: [1x1x4096] memory: 4096 params: $7 \times 7 \times 512 \times 4096 = 102,760,448$

FC: [1x1x4096] memory: 4096 params: $4096 \times 4096 = 16,777,216$

FC: [1x1x1000] memory: 1000 params: $4096 \times 1000 = 4,096,000$

Note:

Most memory is in early CONV

Most params are in late FC

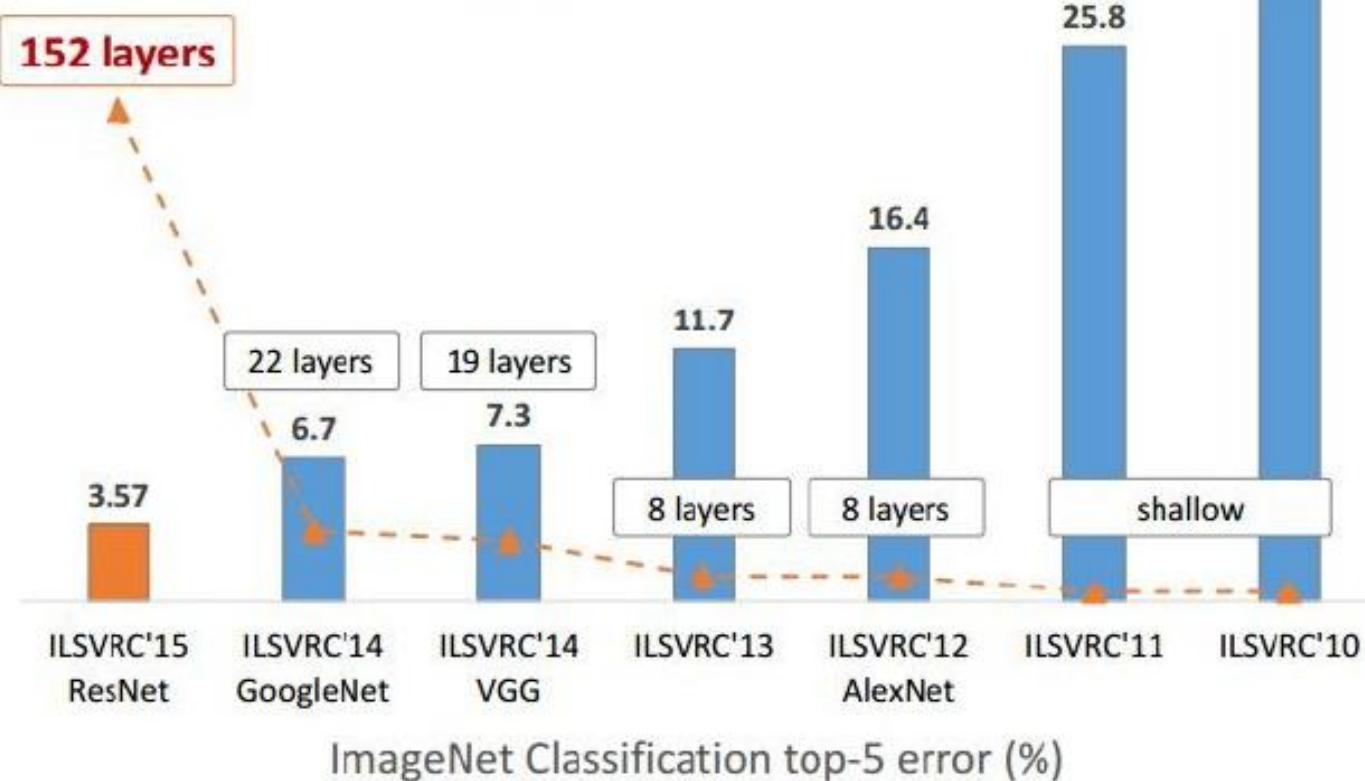
TOTAL memory: $24M * 4$ bytes $\approx 93MB / \text{image}$ (only forward! ~ 2 for bwd)

TOTAL params: 138M parameters

ResNet

Microsoft
Research

Revolution of Depth



Эволюция
глубины



Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

(slide from Kaiming He's recent presentation)

Основные задачи компьютерного зрения

Основная задача CV	Описание задания	Выход	Метрики
Классификация	Учитывая изображение, назначьте ярлык	Метка класса	Точность
Локализация	Определить ограничивающий прямоугольник, содержащий объект на данном изображении	Прямоугольник (x1, y1, x2, y2)	Соотношение пересечения с объединением (перекрытием) реальным значением и ограничительной рамкой
Обнаружение объекта	По изображению определите все объекты и их расположение на изображении.	Для каждого объекта: (ярлык, коробка)	Среднее значение лучшего перекрытия (MAP), Средняя точность (mAP)
Семантическая сегментация	Для данного изображения назначьте каждый пиксель метке класса, чтобы мы могли рассматривать изображение как набор помеченных сегментов.	Набор сегментов изображения	Классификационные метрики, пересечение по общему перекрытию
Сегментация экземпляра	То же, что семантическая сегментация, но каждый экземпляр класса сегмента определяется однозначно	Набор сегментов изображения	

Локализация объектов

- Учитывая изображение, содержащее интересующий объект, определите ограничивающую рамку для объекта
- Классифицировать объект

**Classification
+ Localization**



Классификация + Локализация: Задание

Classification + Localization: Task

Classification: C classes

Input: Image

Output: Class label

Evaluation metric: Accuracy



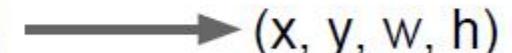
CAT

Localization:

Input: Image

Output: Box in the image (x, y, w, h)

Evaluation metric: Intersection over Union



(x, y, w, h)

Classification + Localization: Do both

Классификация + Локализация: ImageNet

Classification + Localization: ImageNet

1000 classes (same as classification)

1000 классов (то же, что и классификация)

Each image has 1 class, at least one bounding box

Каждое изображение принадлежит одному классу

~800 training images per class

800 тренировочных картинок на класс

Algorithm produces 5 (class, box) guesses

Алгоритм выдает 5 предположений

Example is correct if at least one one guess has correct class AND bounding box at least 0.5 intersection over union (IoU)

Пример корректен, если хотя бы одна попытка соответствует верному классу и выдает обрамляющую коробку минимум 0.5 IoU

Krizhevsky et. al. 2012



Локализация как Регрессия

Idea #1: Localization as Regression

Input: image



Neural Net
→

Выход:
Координаты
коробки

Output:
Box coordinates
(4 numbers)

Потеря:

Loss:
L2 distance

Correct output:
box coordinates
(4 numbers)

Верный выход:
Координаты
коробки

Only one object,
simpler than detection

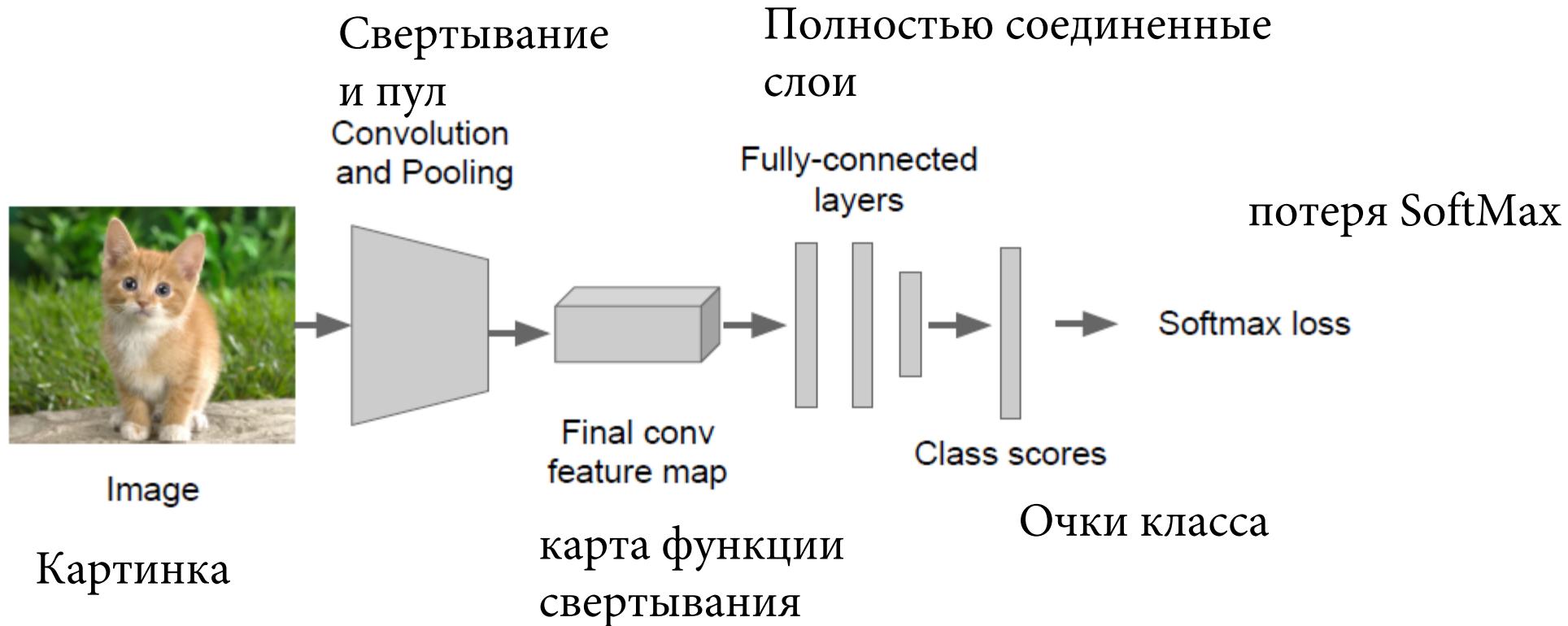
Только один объект, проще, чем
обнаружение

Простой рецепт Классификации + Локализации

Simple Recipe for Classification + Localization

Step 1: Train (or download) a classification model (AlexNet, VGG, GoogLeNet)

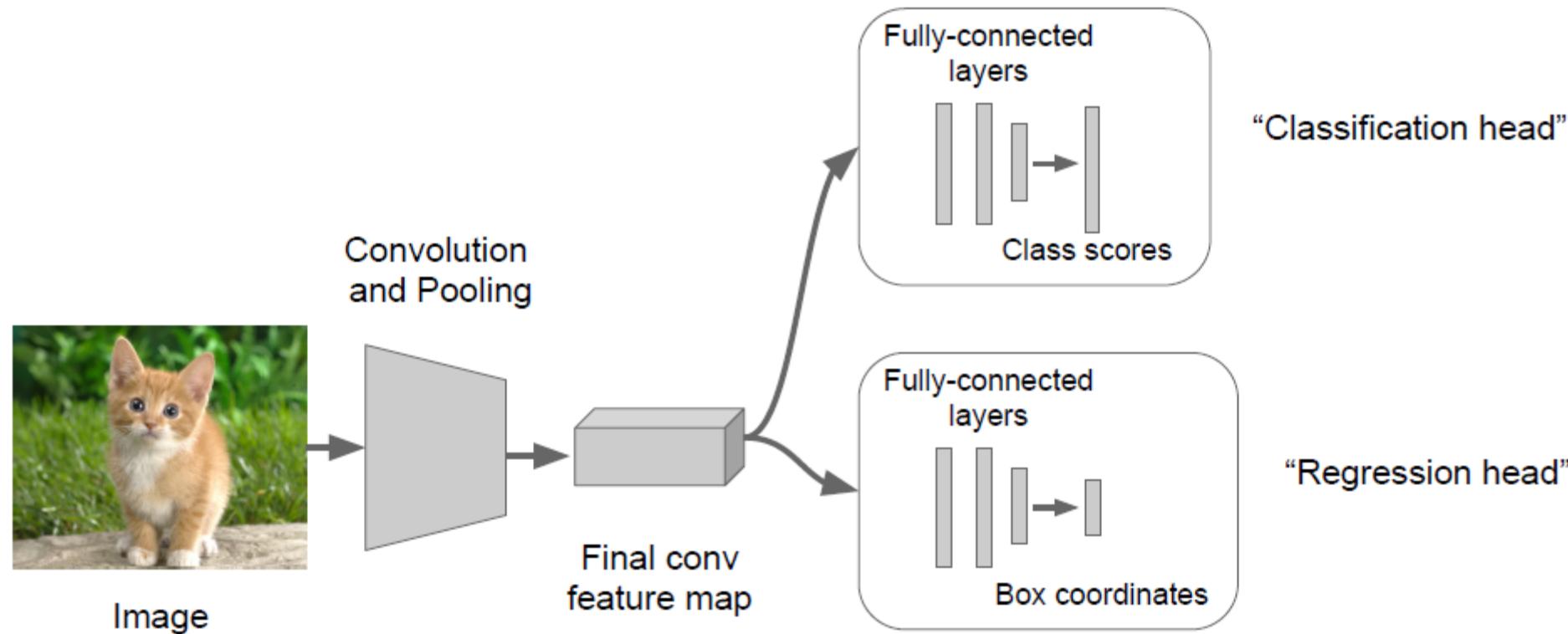
Протренируйте или скачайте классификационную модель



Классификация + Локализация: Задание

Simple Recipe for Classification + Localization

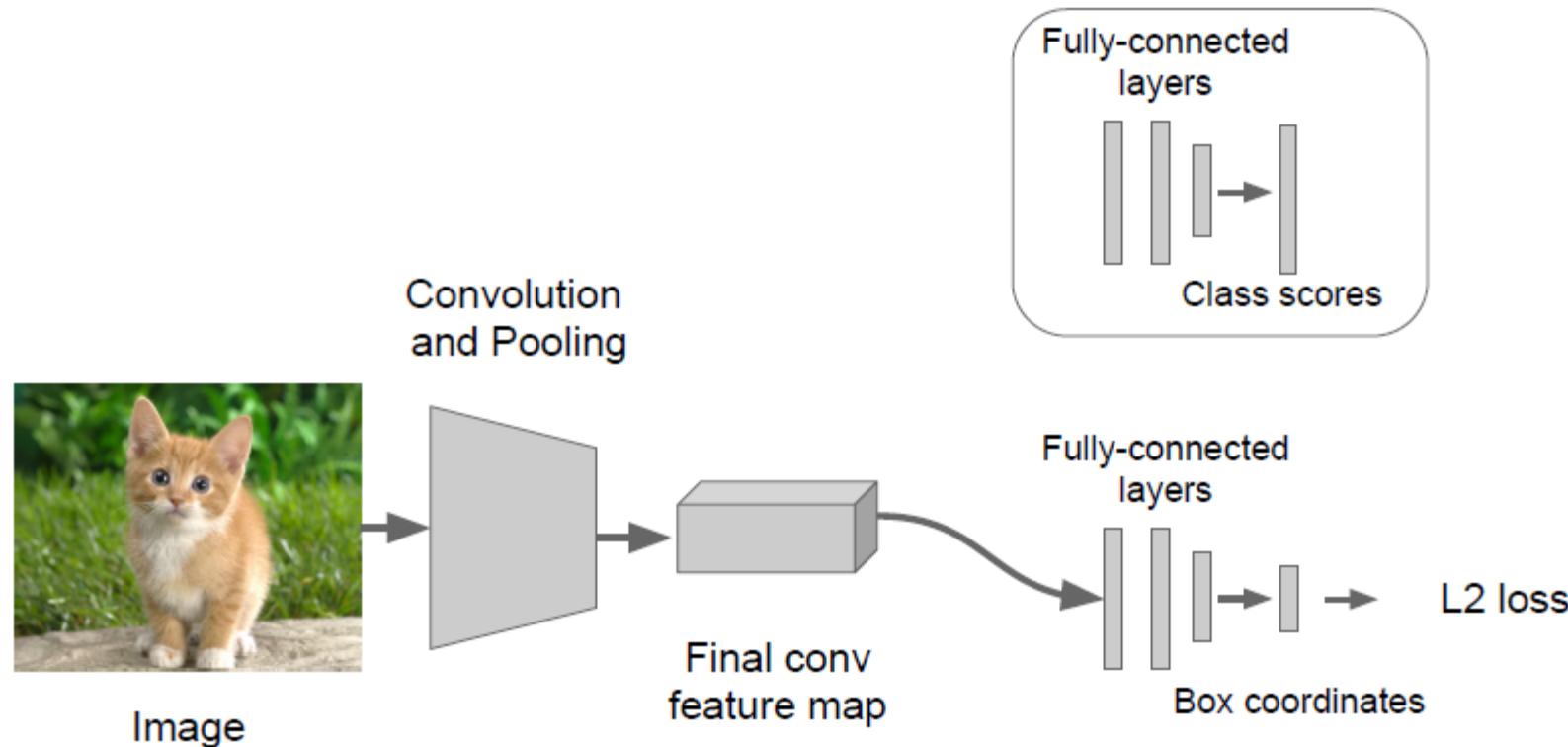
Step 2: Attach new fully-connected “regression head” to the network



Классификация + Локализация: Задание

Simple Recipe for Classification + Localization

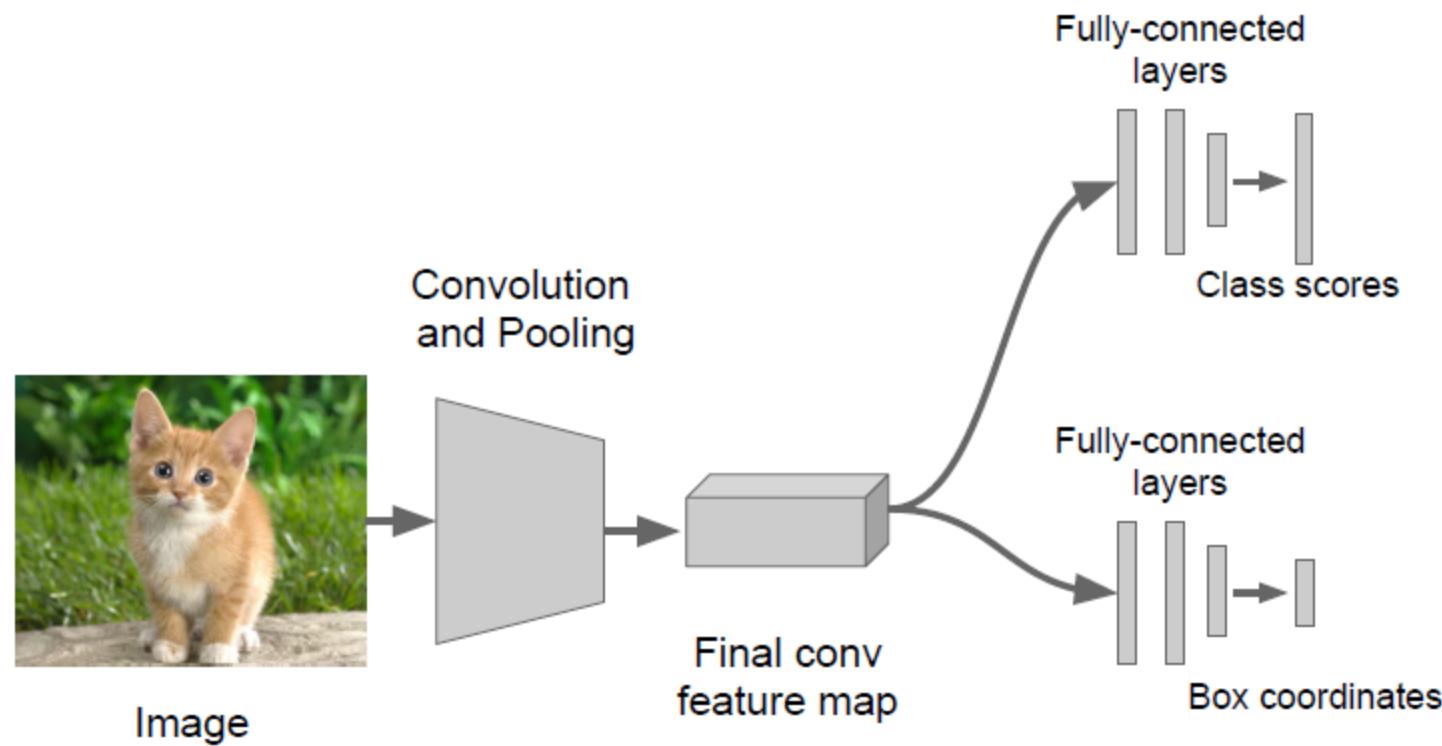
Step 3: Train the regression head only with SGD and L2 loss



Классификация + Локализация: Задание

Simple Recipe for Classification + Localization

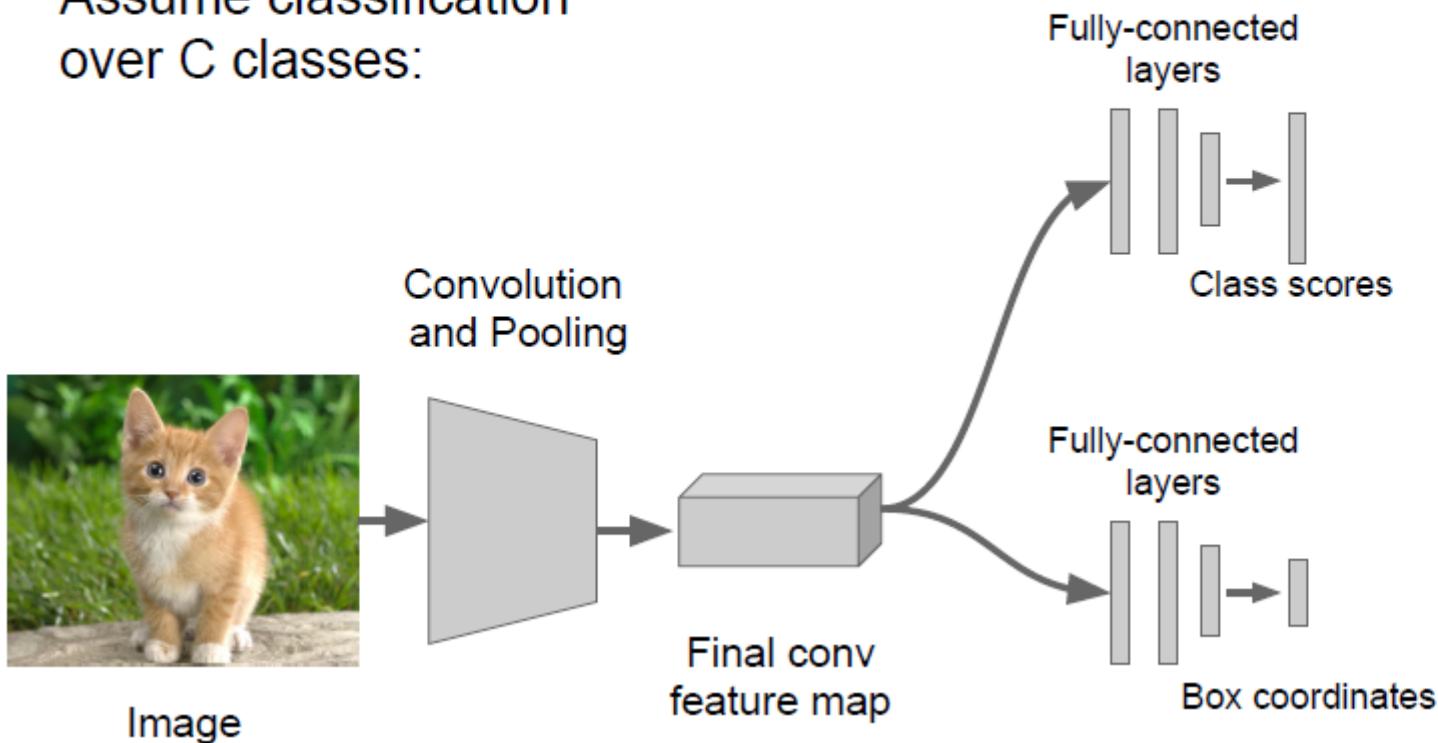
Step 4: At test time use both heads



Поклассовая/классовая агностическая регрессия

Per-class vs class agnostic regression

Assume classification over C classes:



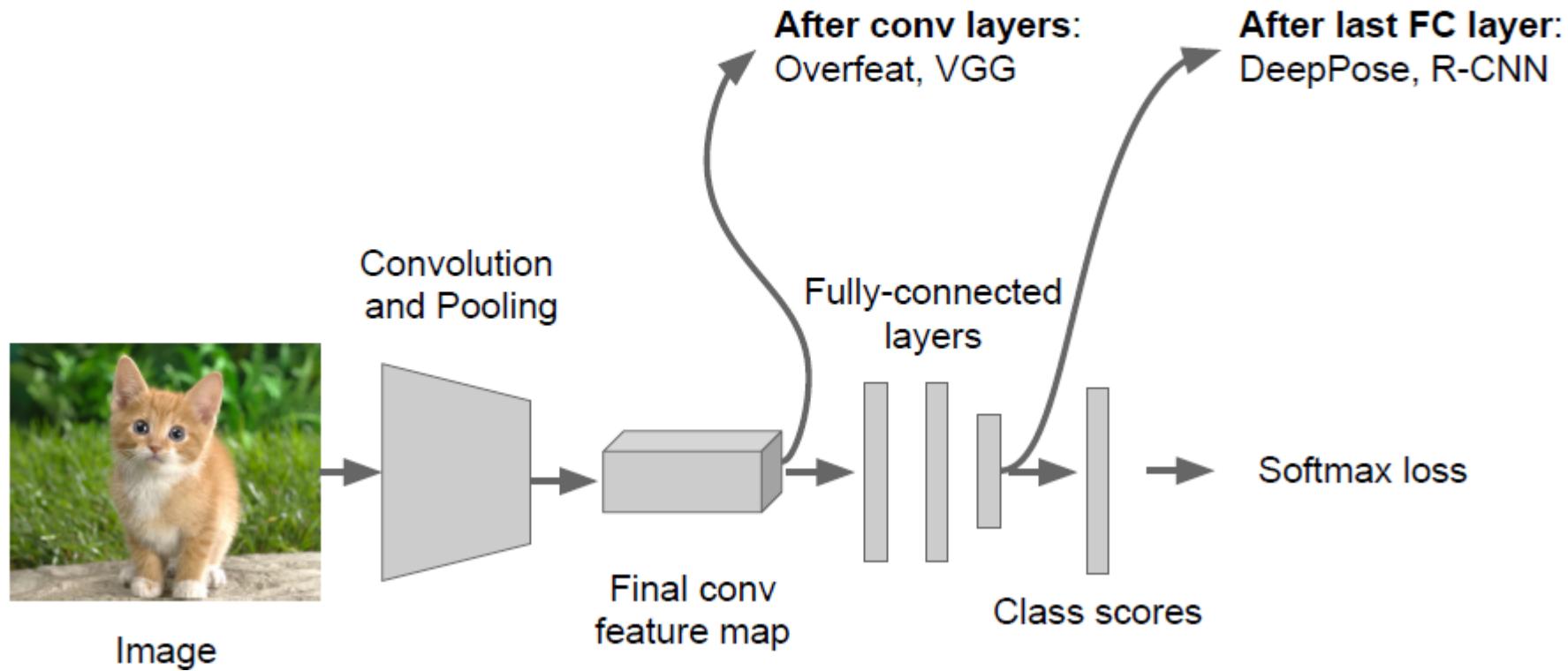
Classification head:
C numbers
(one per class)

Class agnostic:
4 numbers
(one box)

Class specific:
 $C \times 4$ numbers
(one box per class)

Куда прикреплять регрессионную головку?

Where to attach the regression head?

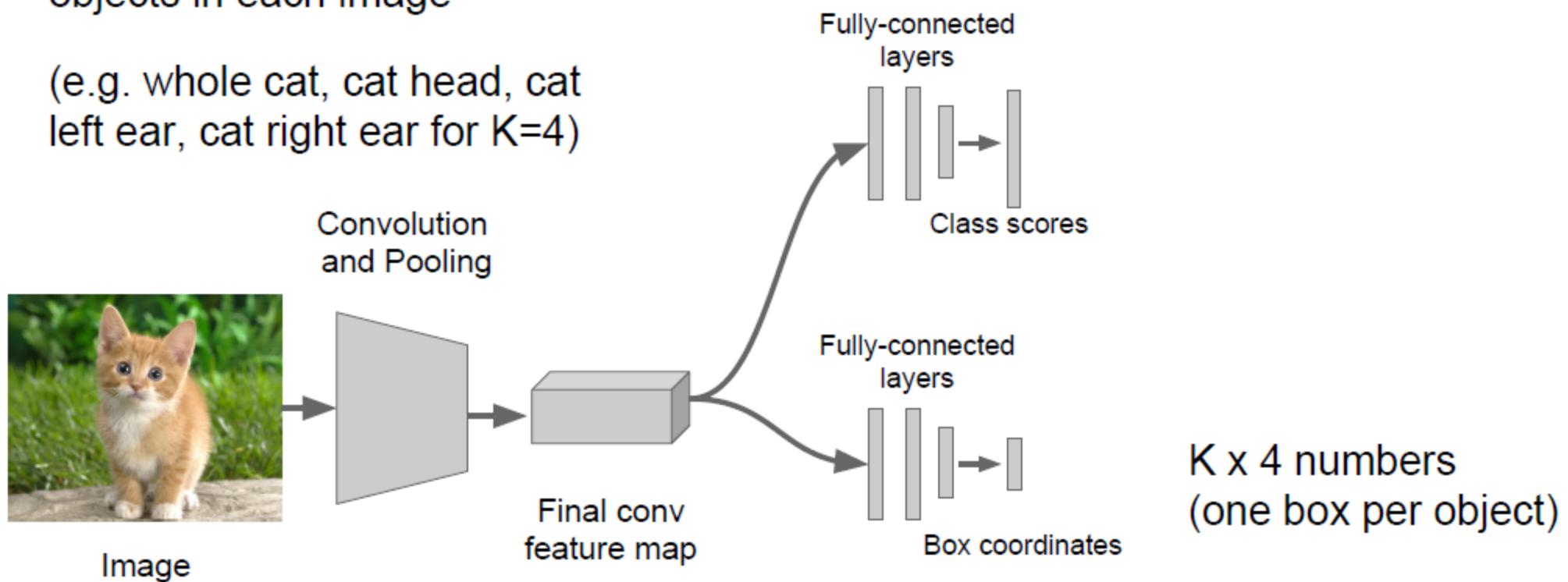


Отдельно: Нахождение нескольких объектов

Aside: Localizing multiple objects

Want to localize **exactly K** objects in each image

(e.g. whole cat, cat head, cat left ear, cat right ear for $K=4$)



Отдельно: Распознавание человеческой позы

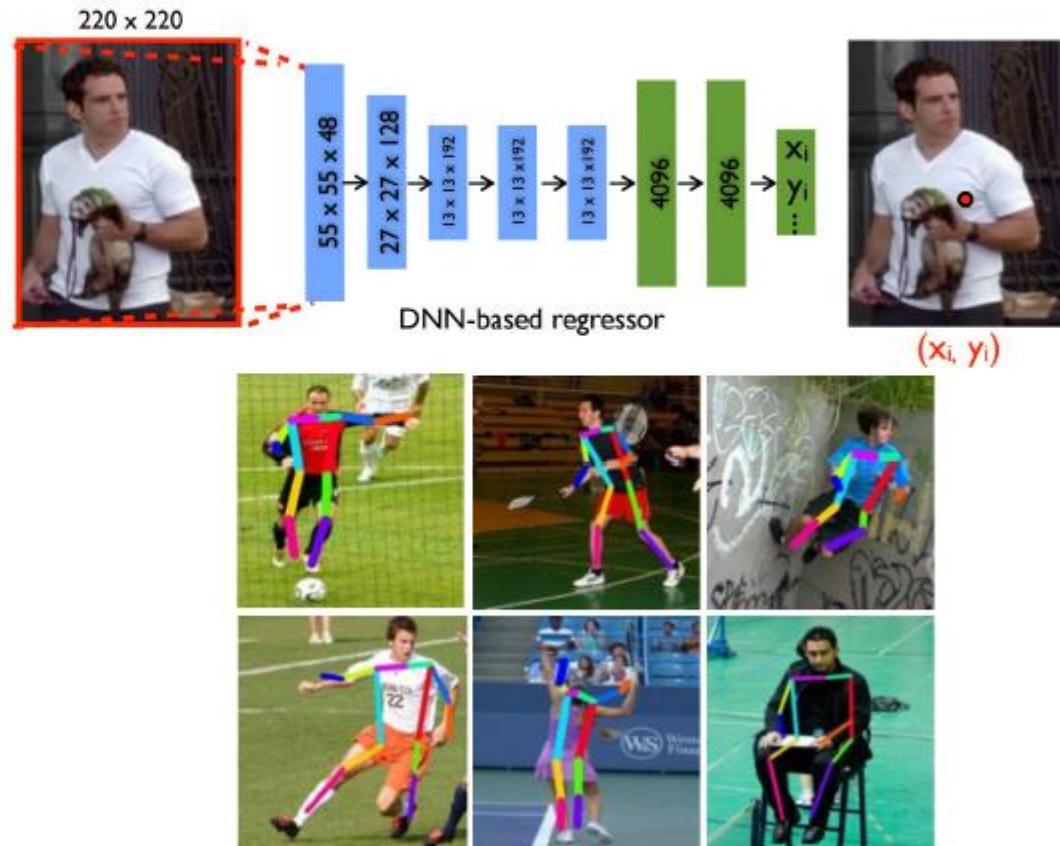
Aside: Human Pose Estimation

Represent a person by K joints

Regress (x, y) for each joint from last fully-connected layer of AlexNet

(Details: Normalized coordinates, iterative refinement)

Toshev and Szegedy, "DeepPose: Human Pose Estimation via Deep Neural Networks", CVPR 2014



Набор данных для оценки

- Задачи Imagenet предоставляют исследователям платформу для сравнения их новых алгоритмов
- PASCAL VOC 2010 отлично подходит для небольших экспериментов. Весит около 1,3 ГБ.
- Наборы данных MS COCO доступны для таких задач, как подписи к изображениям. Размер для загрузки довольно большой, но выборочная загрузка возможна.

	PASCAL VOC (2010)	ImageNet Detection (ILSVRC 2014)	MS-COCO (2014)
Number of classes	20	200	80
Number of images (train + val)	~20k	~470k	~120k
Mean objects per image	2.4	1.1	7.2