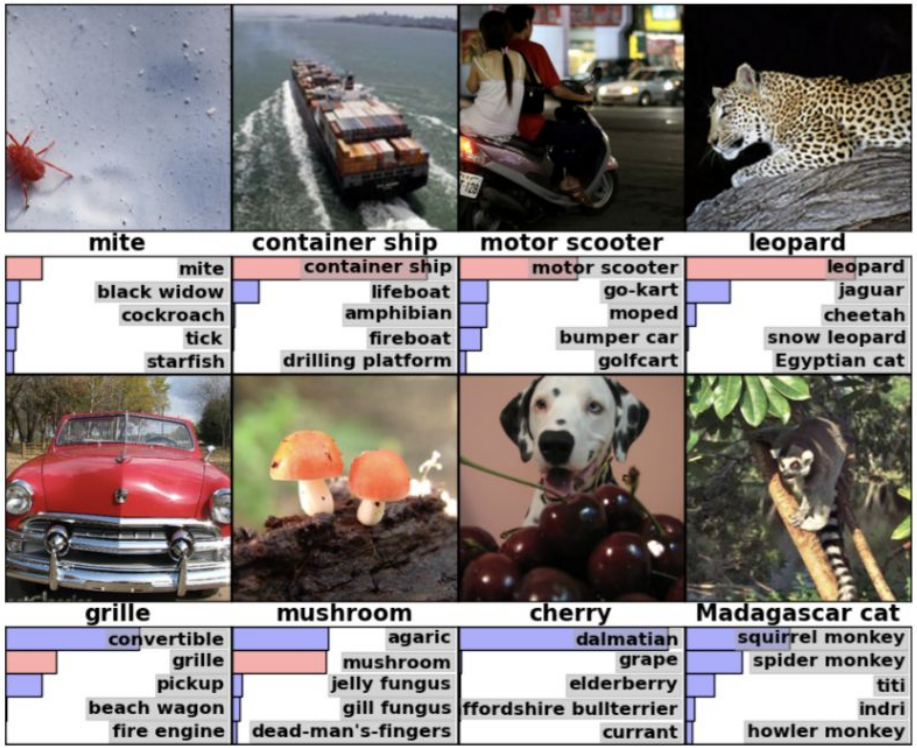
Компьютерное зрение

Лекция 2. Сверточные нейронные сети. Базовые блоки, принципы и приложения

18.06.2020

Руслан Алиев

ImageNet Challenge



Сверточные нейронные сети (CNNs)

Мотивация

- изображения представляют собой объекты с большим количеством признаков
- изображение в формате RGB размера 640x480 будет иметь ~1млн признаков
- число параметров полносвязной сети с внутренним слоем из 10 нейронов равно ~10млн
- большое число параметров модели существенно затрудняет процесс обучения

Операция свертки

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u,j-v]$$

F - исходное изображение
H - фильтр размера kxk
G - результат на выходе свертки
i,j - координаты пикселя в районе которого применяется операция свертки

Свертка двух матриц

Input =

0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Filter =

1	0	1
0	1	0
1	0	1

0	0	1
0	0	1
0	1	1

dot

1	0	1
0	1	0
1	0	1

Свертка двух матриц

Input =

0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Filter =

1	0	1
0	1	0
1	0	1

0	0	1
0	0	1
0	1	1

dot

1	0	1
0	1	0
1	0	1

= 2

Свертка двух матриц

Input =

0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Filter =

1	0	1
0	1	0
1	0	1

Свертка двух матриц

Input =

0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Filter =

1	0	1
0	1	0
1	0	1

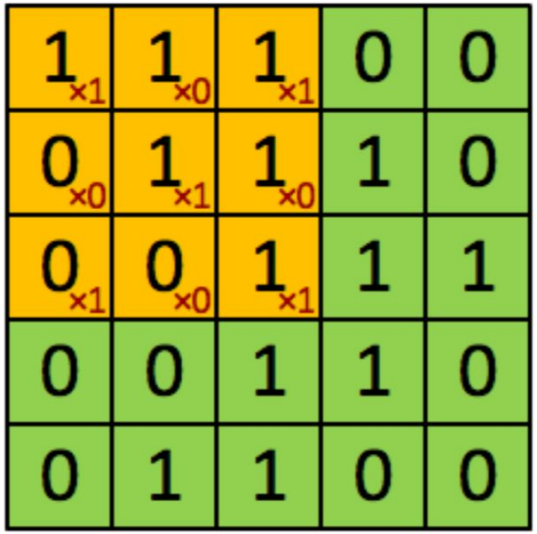
0	1	1
0	1	1
1	1	0

dot

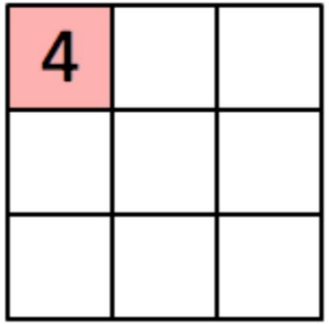
1	0	1
0	1	0
1	0	1

= 3

Как посчитать свертку



Image



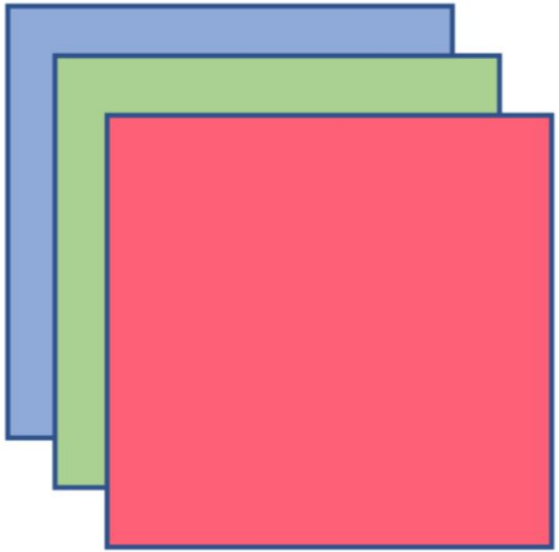
Convolved Feature

Как посчитать свертку

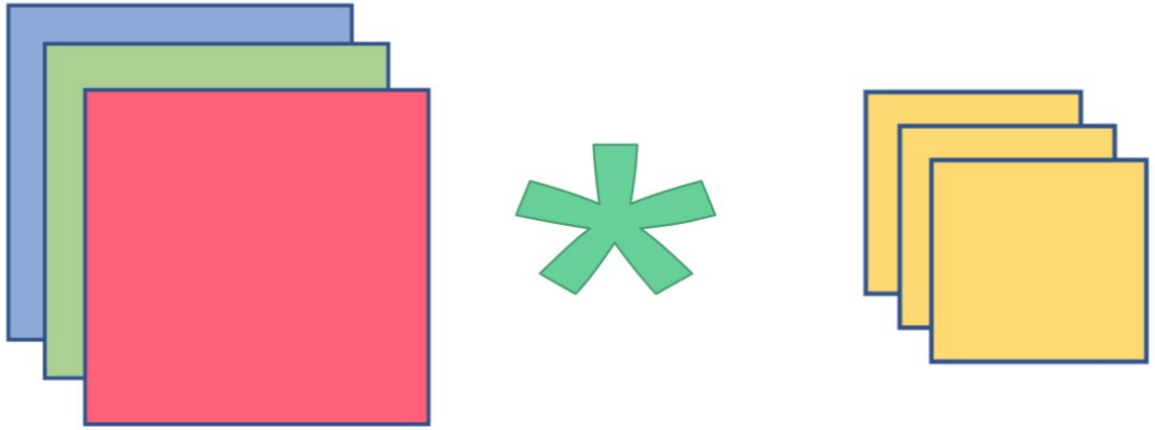
1. Имея входное изображение, наложить на него маленькое окно (ядро свертки). Найти произведение ядра и участка изображения которое накрыло ядро
2. Сдвинуть окно и повторить процедуру
3. Повторить шаги 1 и 2 для всех остальных выходных каналов уже с другими ядрами

Объемная свертка (Volume Convolution,Conv2d)

- Картинка имеет 3 канала, т.е. это 3d матрица

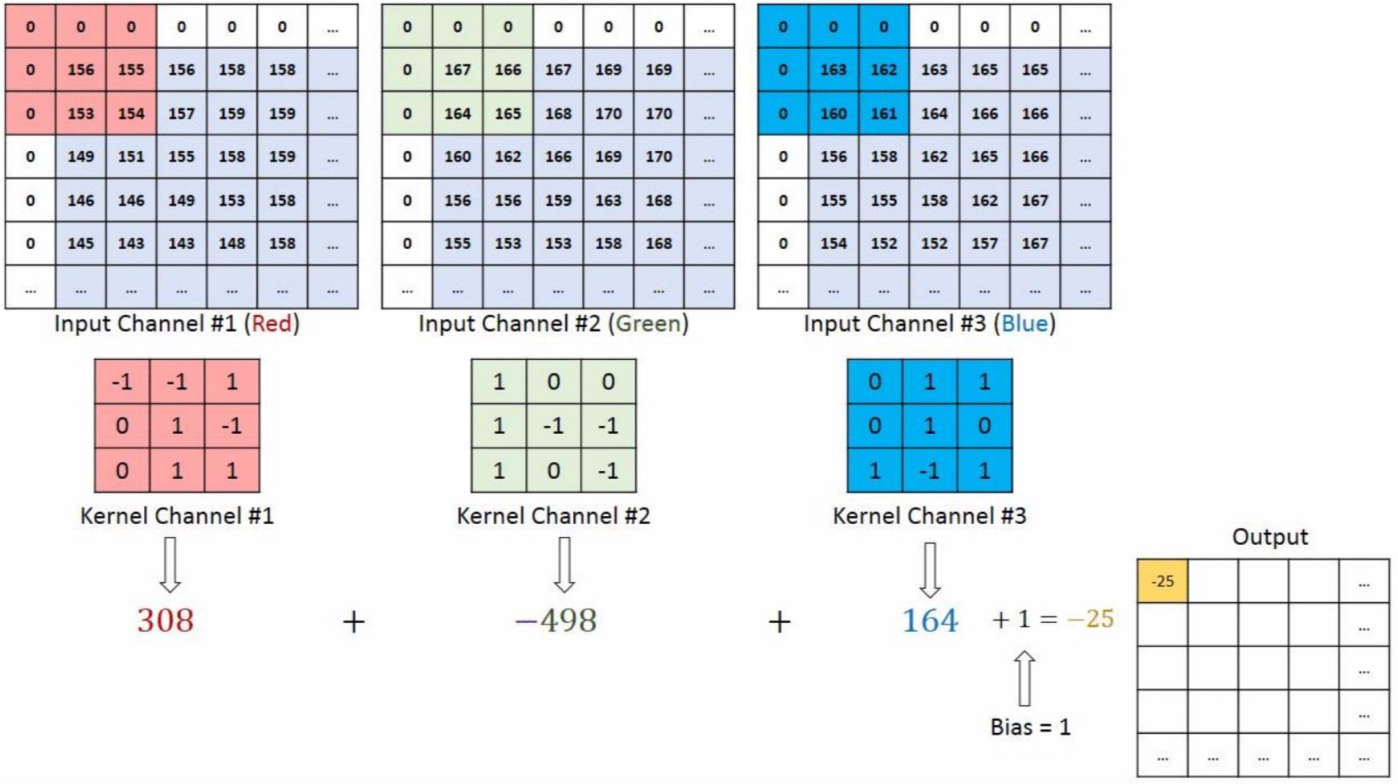


Объемная свертка (Volume Convolution,Conv2d)

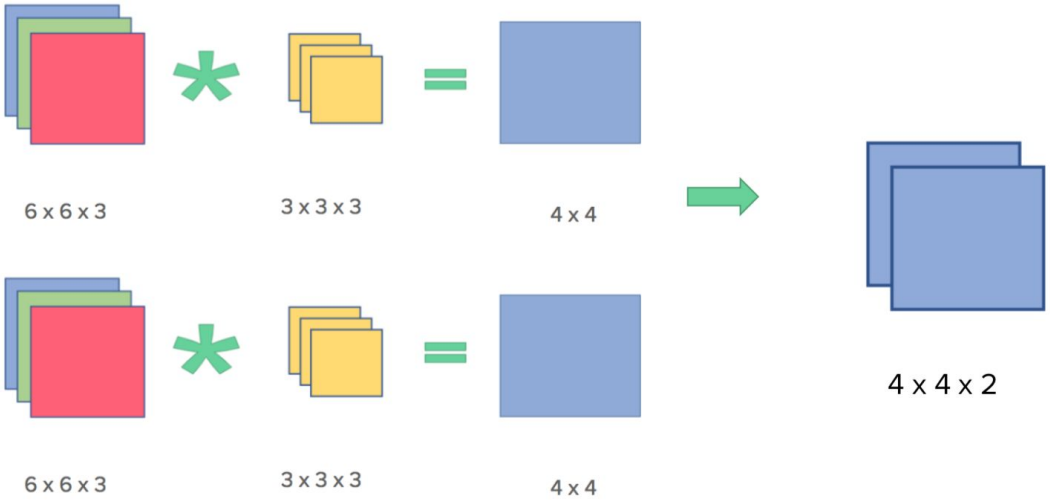


Объемная свертка (Volume Convolution,Conv2d)

Как применить свертку?



Сверточный слой



$$V(x,y,t) = \sum_{i=x-\delta}^{x+\delta} \sum_{j=y-\delta}^{y+\delta} \sum_{s=1}^{S} K^t(i-x+\delta, j-y+\delta, s) \cdot U(i,j,s)$$

Что делает свертка

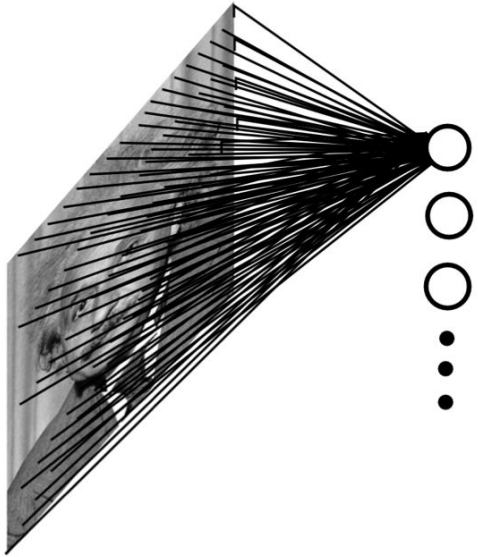
Двигать окно по изображению - что это нам дает?

1. Local connectivity
   свертка как способ извлекать локальные признаки

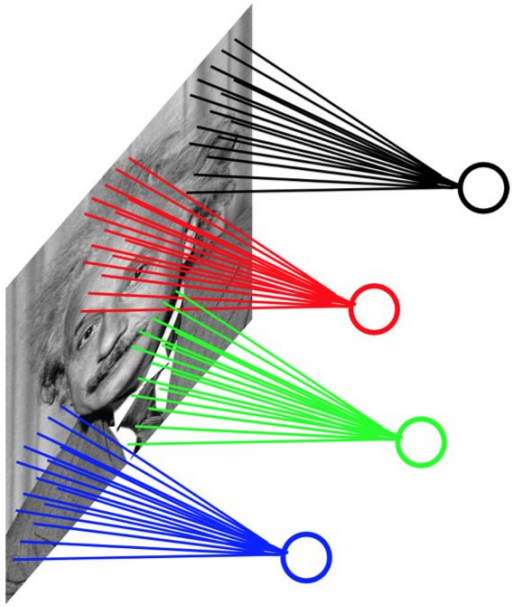
2. Weight (Parameter) sharing
   свертка как способ уменьшить сложность модели

Свертка как детектор признаков

Local connectivity

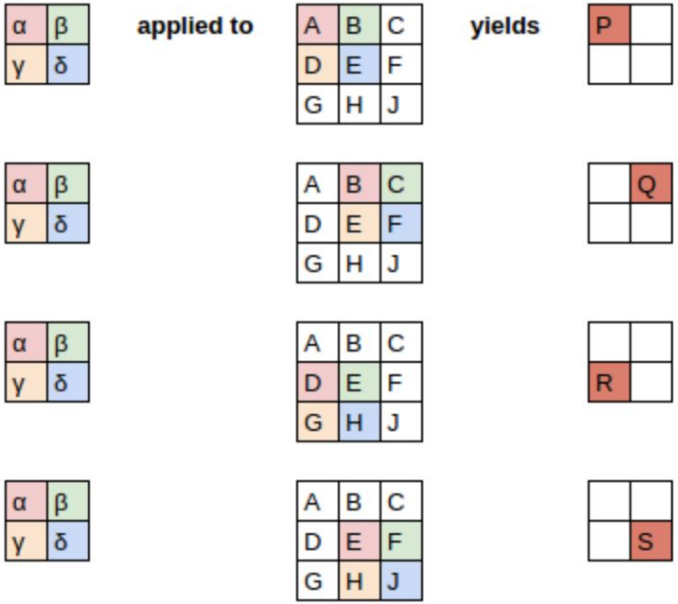


Fully Connected: 200x200 image, 40K hidden units, ~2B parameters

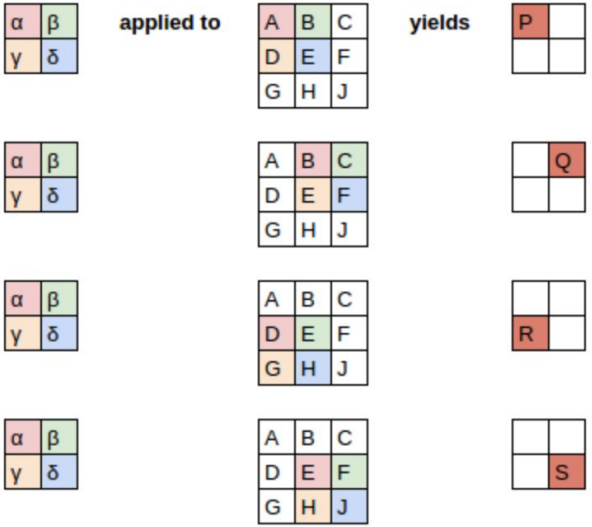


Locally Connected: 200x200 image, filter size 10x10, 4M parameters!

Weight sharing



Weight sharing



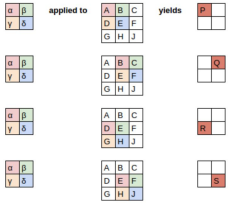
$$\alpha * A + \beta * B + \gamma * D + \delta * E + b = P$$

$$\alpha * B + \beta * C + \gamma * E + \delta * F + b = Q$$

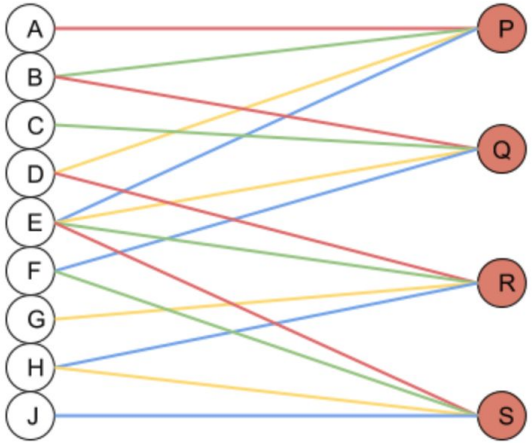
$$\alpha * D + \beta * E + \gamma * G + \delta * H + b = R$$

$$\alpha * E + \beta * F + \gamma * H + \delta * J + b = S$$

Weight sharing



$$\alpha * A + \beta * B + \gamma * D + \delta * E + b = P$$
$$\alpha * B + \beta * C + \gamma * E + \delta * F + b = Q$$
$$\alpha * D + \beta * E + \gamma * G + \delta * H + b = R$$
$$\alpha * E + \beta * F + \gamma * H + \delta * J + b = S$$



$$\begin{bmatrix} \alpha & \beta & 0 & \gamma & \delta & 0 & 0 & 0 & 0 \\ 0 & \alpha & \beta & 0 & \gamma & \delta & 0 & 0 & 0 \\ 0 & 0 & 0 & \alpha & \beta & 0 & \gamma & \delta & 0 \\ 0 & 0 & 0 & 0 & \alpha & \beta & 0 & \gamma & \delta \end{bmatrix} * \begin{bmatrix} A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ J \end{bmatrix} + \begin{bmatrix} b \\ b \\ b \\ b \end{bmatrix} = \begin{bmatrix} \alpha A+\beta B+0C+\gamma D+\delta E+0F+0G+0H+0J+b \\ 0A+\alpha B+\beta C+0D+\gamma E+\delta F+0G+0H+0J+b \\ 0A+0B+0C+\alpha D+\beta E+0F+\gamma G+\delta H+0J+b \\ 0A+0B+0C+0D+\alpha E+\beta F+0G+\gamma H+\delta J+b \end{bmatrix} = \begin{bmatrix} \alpha A+\beta B+\gamma D+\delta E+b \\ \alpha B+\beta C+\gamma E+\delta F+b \\ \alpha D+\beta E+\gamma G+\delta H+b \\ \alpha E+\beta F+\gamma H+\delta J+b \end{bmatrix} = \begin{bmatrix} P \\ Q \\ R \\ S \end{bmatrix}$$

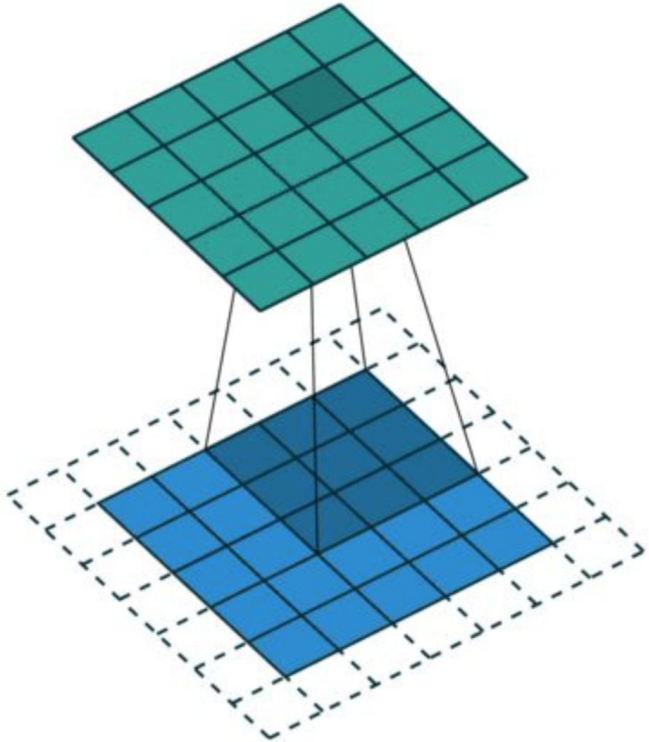
A B C D E F G H J

Как применить свертку

Вопросы на которые мы еще не ответили включают:

1. Размер изображения на выходе меньше чем на входе и мы не знаем как хорошо обрабатывать границы
2. Накладываем ядро свертки на каждый участок изображения или можем двигаться с интервалом?

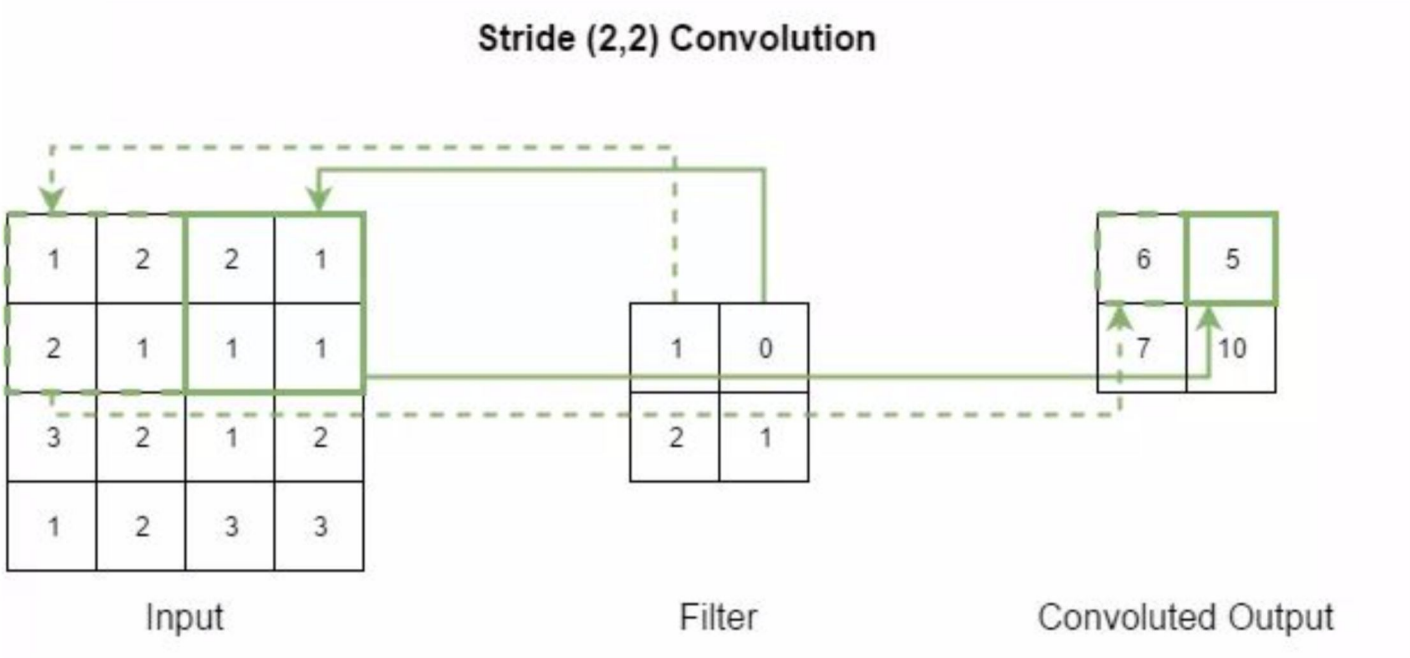
Отступы (padding)



Отступы (padding)

- чтобы размер на выходе совпадал с размером на входе, размер исходного изображения увеличивают, заполняя пространство по периметру например определенными значениями
- zero padding
- replication padding
- reflection padding

Шаг (Stride)


Stride (2,2) Convolution
1 2 2 1
2 1 1 1
3 2 1 2
1 2 3 3
1 0
2 1
6 5
7 10
Input
Filter
Convoluted Output

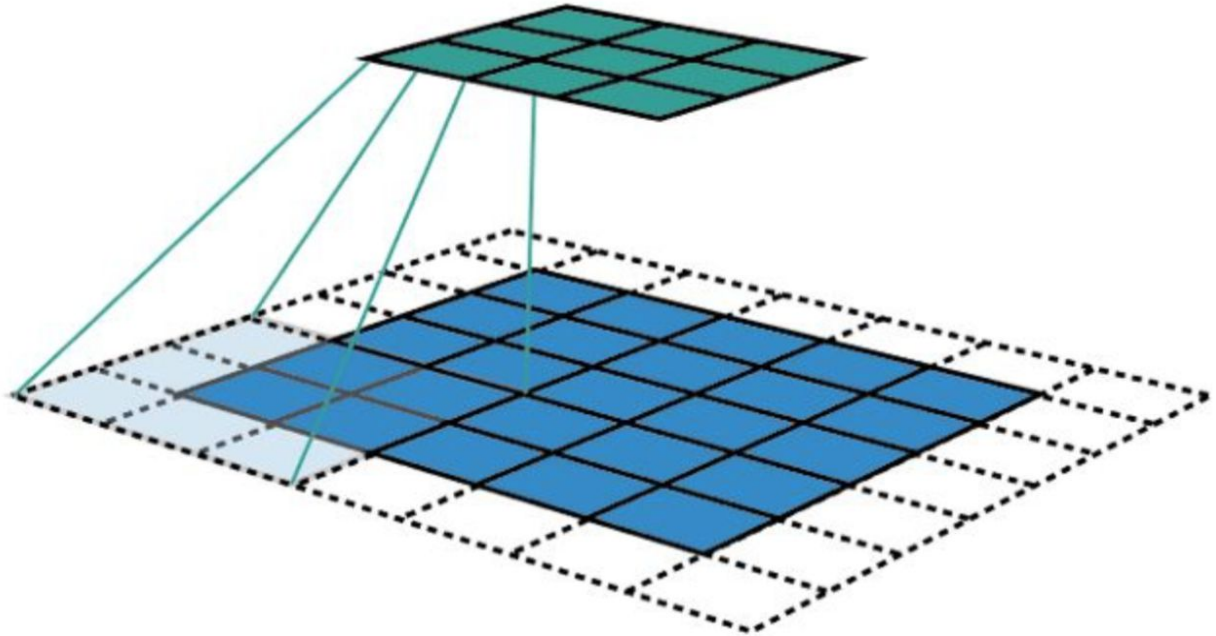
Шаг (Stride)

- шаг может быть разным по оси x и y, но в основном используют одинаковый
- определяет смещение фильтра на очередной итерации
- при увеличении шага уменьшается размер выхода
- шаг > 1 в начальных слоях
- шаг > 1 увеличивает receptive field (далее)

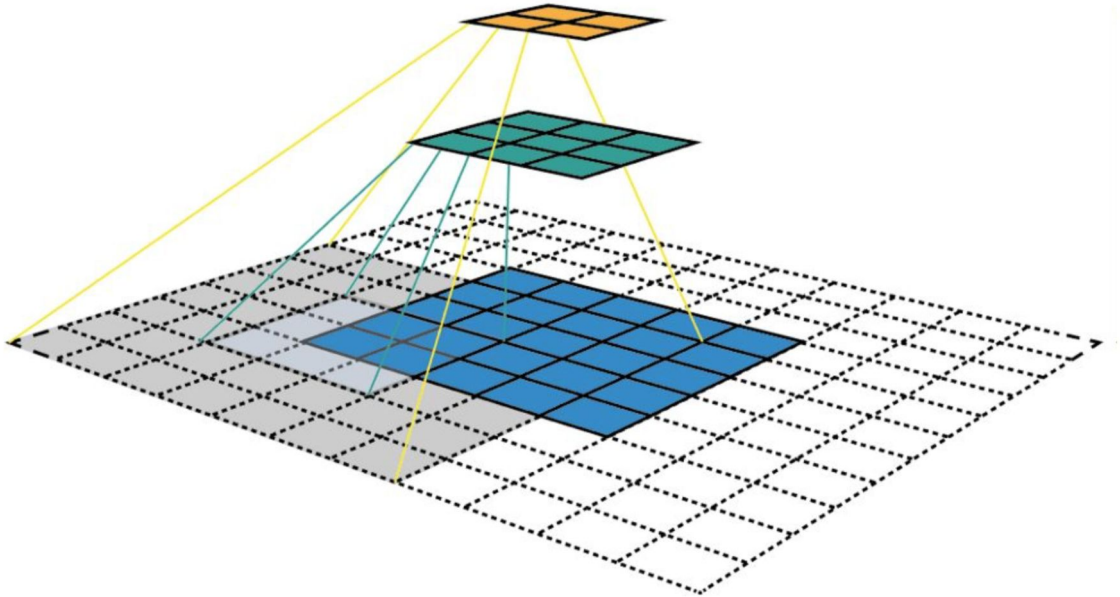
Область видимости фильтра (reception field)

- область видимости - область исходного изображения от которой зависит значение пикселя на выходе сверточного слоя
- область видимости отдельного пикселя на выходе сверточного слоя возрастает с увеличением глубины сети

Область видимости фильтра (reception field)



Область видимости фильтра (reception field)



Pooling Layer

Pooling Layer

Тензор признаков на выходе свертки говорит нам о:

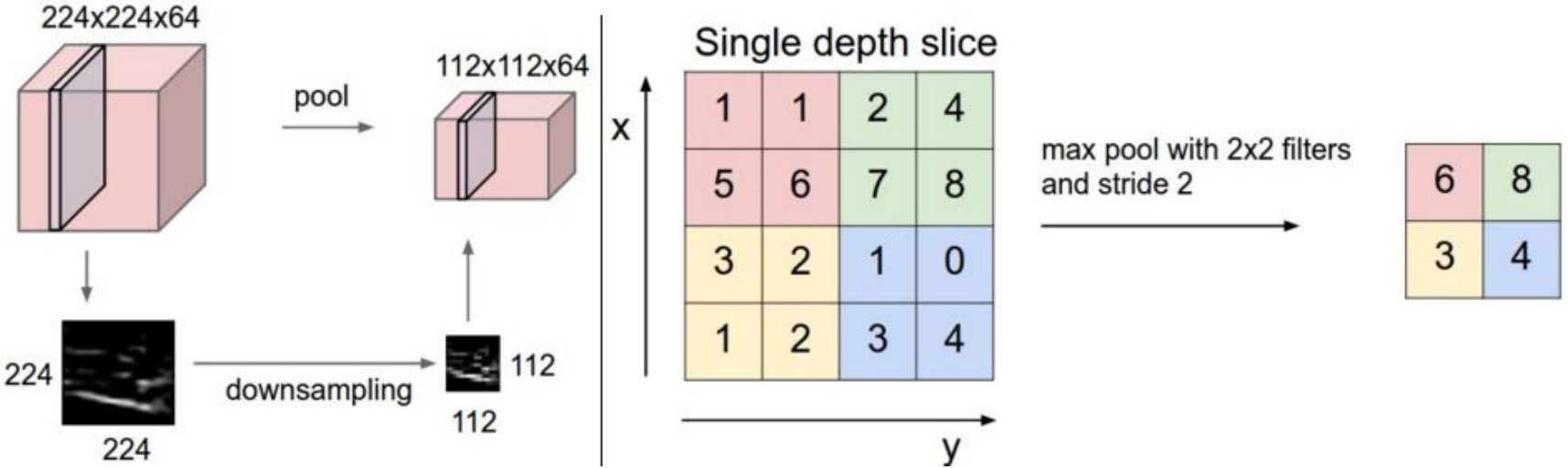
1. присутствует ли признак на изображении (большая активация)
2. если да, то где примерно был найден на изображении

Мы можем сохранить оба свойства достаточно хорошо с помощью операции pooling (например max или average)

Отступы (padding)

- при последовательном применении фильтров размер выхода будет уменьшаться с каждым шагом на размер фильтра
- при этом теряется информация о краях изображения

Pooling Layer



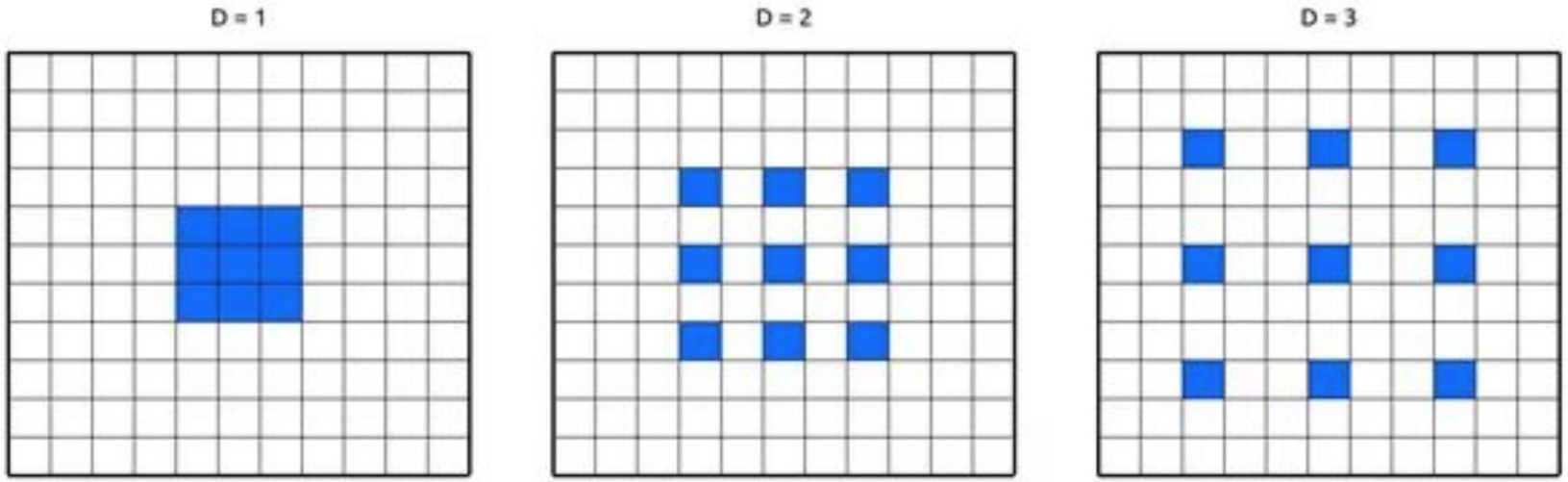
Pooling Layer

- pooling слой вствляется после активаций сверточного слоя
- уменьшает пространственный размер данных
- как результат уменьшается число параметров сети
- необучаемый слой - не содержит параметров

Расширение (dilation)

- увеличение размера фильтра (области видимости) без увеличения числа параметров (весов)
- веса фильтра “раздвигаются” в пространстве
- фильтр имеет разреженную структуру
- свободные позиции фильтра заполняются нулями
- число свободных позиций является гипер-параметром
- свертки с расширением именуются atrous (dilated) convolutions

Расширение (dilation)



Математика

- Размер картинки (cin, h, w)
- Размер фильтра (cout, cin, k1, k2)
  - Обычно k1=k2=f=нечетному числу, например 3, 5, 7
- Допустим h = n, w = n тогда размер картинки на выходе (cout, m, m)

$$m = \left\lfloor \frac{n + 2p - f}{S} \right\rfloor + 1$$

- где S - размер шага, p значение паддинга с одной стороны

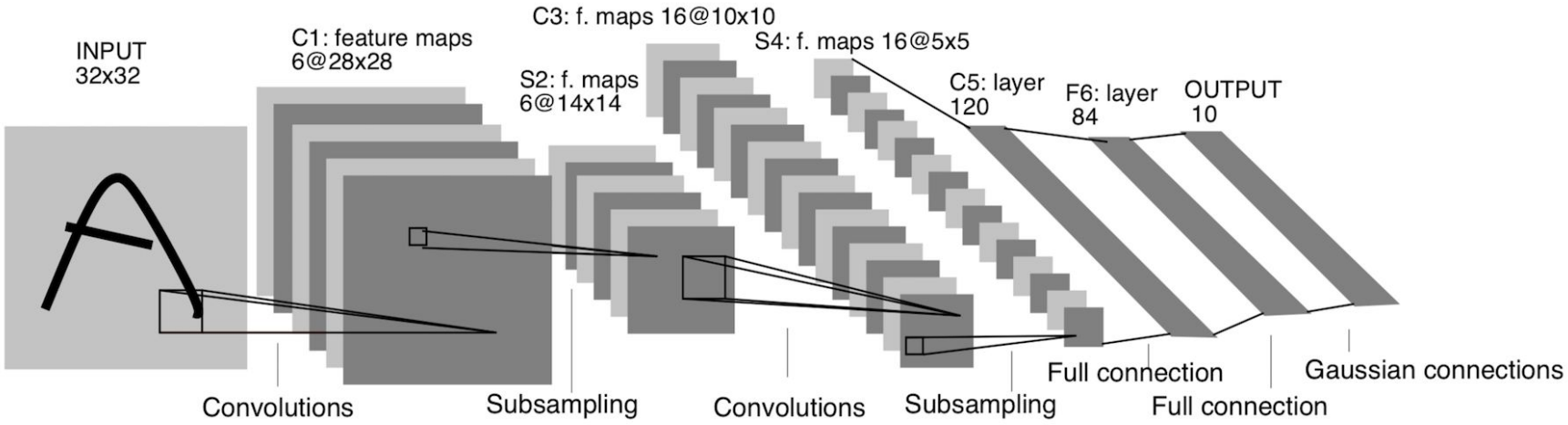
Сверточный слой (Convolution layer)

- Содержит набор из cout фильтров одинакового размера (cin, f, f)
- Размер входа (b, cin, h, w)
- Размер выхода (b, cout, h~, w~)

  где:
  - b - размер батча
  - cin - число каналов на входе
  - cout - число каналов на выходе

- На выходе применяется нелинейная активация

LeNet (1998)


INPUT
32x32
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: layer
120
F6: layer
84
OUTPUT
10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection
Gaussian connections

Сверточные нейронные сети

- в нейронных сетях для обработки изображений полносвязные слои заменяют на сверточные
- полносвязные слои (+dropout) могут использоваться в выходном слое
- сверточный блок обычно имеет следующий вид:
  Conv2d -> нормализация -> нелинейная активация (н: ReLU) -> Pooling

Invariance

- CNNs обладают интересным свойством translation (shift) invariance, т.е. если мы сдвинем объект на фото, сеть все равно можно распознать его, так как свертка - операция локальная

- Pooling layers тоже добавляют свойство инвариантности, но это invariance to small perturbations, т.е. если появляется какой-то шум после активаций, pooling позволяет снизить влияние шума

Invariance vs Equivariance

- Часто это эти понятия путают

Shift-equivariance and invariance A function $\widetilde{\mathcal{F}}$ is shift-equivariant if shifting the input equally shifts the output, meaning shifting and feature extraction are commutable.

$$\text{Shift}_{\Delta h,\Delta w}(\widetilde{\mathcal{F}}(X)) = \widetilde{\mathcal{F}}(\text{Shift}_{\Delta h,\Delta w}(X)) \quad \forall\ (\Delta h, \Delta w) \tag{1}$$

A representation is shift-invariant if shifting the input results in an *identical* representation.

$$\widetilde{\mathcal{F}}(X) = \widetilde{\mathcal{F}}(\text{Shift}_{\Delta h,\Delta w}(X)) \quad \forall\ (\Delta h, \Delta w) \tag{2}$$

Нормализация

Нормализация

- чаще всего дает прирост как в качестве, так и в скорости сходимости процесса обучения сети
- повышает обобщающую способность сети

Batch normalization

- приводит значения активаций на выходе слоя к нулевому среднему и единичной дисперсии
- во время обучения среднее и дисперсия оцениваются для каждого батча
- далее применяется афинное преобразование (optional)
- самый популярный вариант

Batch normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Batch normalization (режимы работы)

- train or eval mode
  - во время обучения используются текущие статистики батча
  - во время тестирования (inference) используются бегущие средние статистик, собранные во время обучения

- affine = True (default) or affine = False
  - применять ли scale и shift после нормализации или нет

Batch normalization (минусы)

- в случае маленького размера батча, могут появиться коррелированные по контенту изображения или зависимости между картинками, что плохо
- требуется большой размер матча, обычно >= 16
- такой размер батча иногда сложно использовать, так как потребление памяти очень сильно растет с увеличением модели
- возможное решение synchronous batch norm
  - если у вас много gpu
  - статистики батча собираются со всех видеокарт
- еще решение - убрать зависимость от размера батча, использовать другую нормализацию

Другие известные нормализации

В этих нормализациях зависимость от размера батча исчезает, делаем для каждой картинки по-отдельности

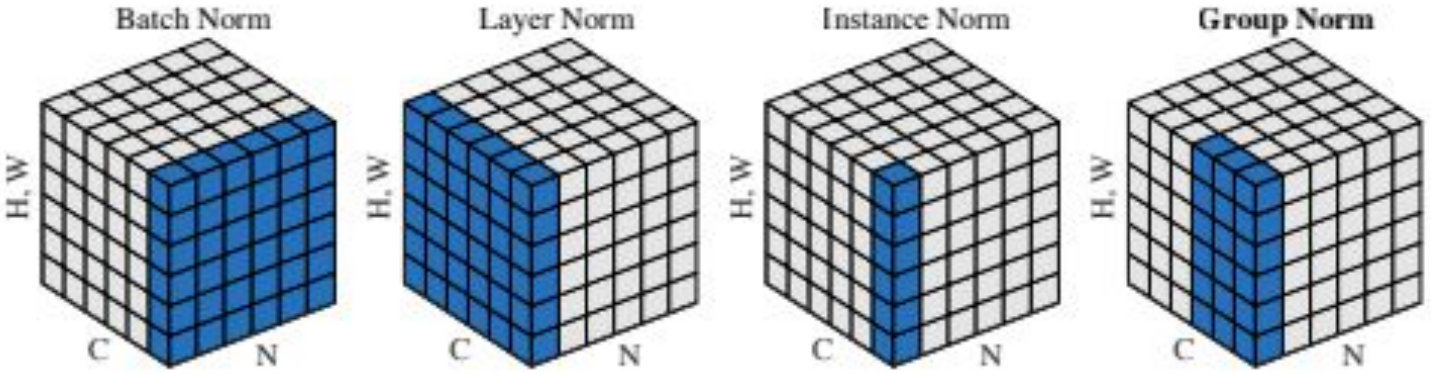
- Layer normalization
  - среднее и дисперсия считаются по всем каналам по всем пикселям Local response normalization
- Local response normalization
  - среднее и дисперсия считаются по группам каналов для каждого пикселя
  - практически потеряла популярность
- Group normalization
  - среднее и дисперсия считаются по группам каналов по всем пикселям

Другие известные нормализации

В этих нормализациях зависимость от размера батча исчезает, делаем для каждой картинки по-отдельности

- Layer normalization
  - среднее и дисперсия считаются по всем каналам по всем пикселям Local response normalization
- Local response normalization
  - среднее и дисперсия считаются по группам каналов для каждого пикселя
  - практически потеряла популярность
- Group normalization
  - среднее и дисперсия считаются по группам каналов по всем пикселям

Другие известные нормализации

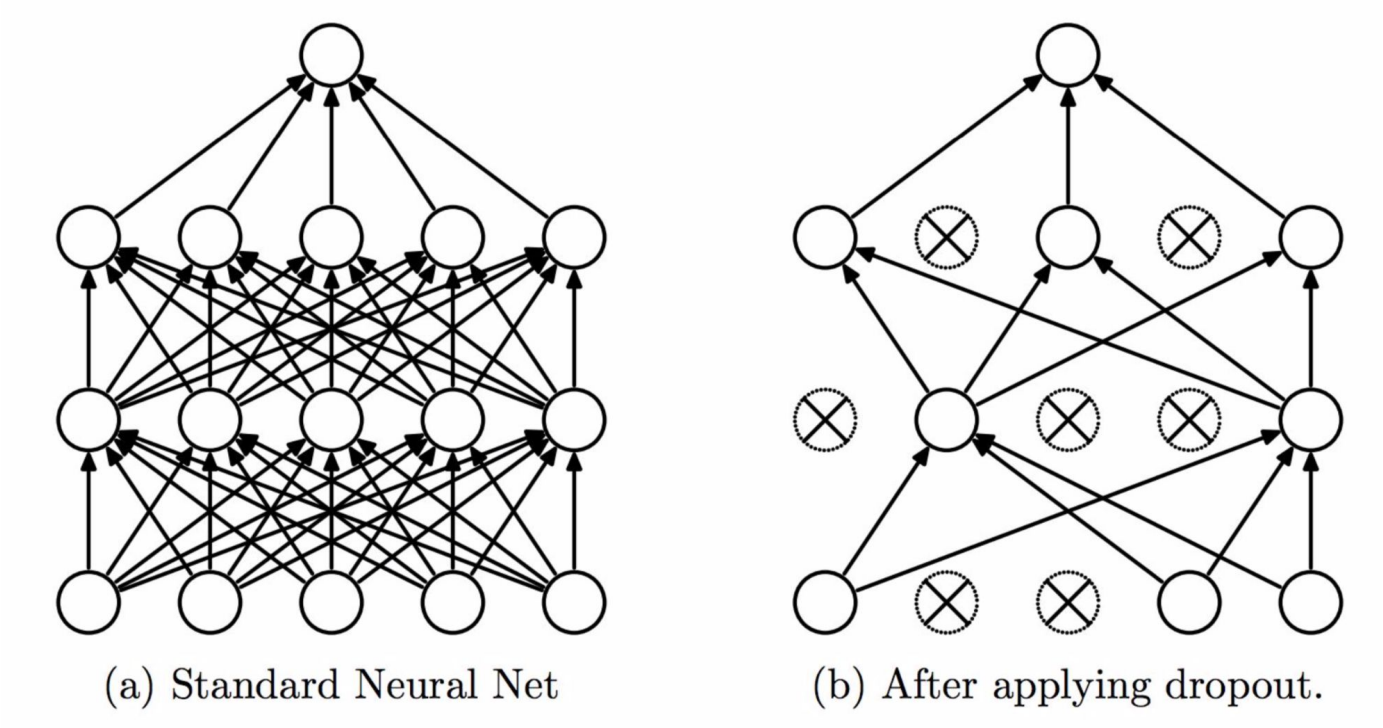


Dropout

Dropout

- в процессе обучения случайным образом выбранная часть нейронов (каналов в случае CNN) слоя зануляется
- доля зануляемых входов является гипер-параметром p
- на этапе обучения значения выходов делится на p
- на этапе тестирования dropout не используется
- является способом регуляризации, т.е. позволяет модели быть более устойчивой к переобучению

Dropout



(a) Standard Neural Net

(b) After applying dropout.

Recap

- в отличие от полносвязных, сверточные сети содержат меньше параметров, как результат менее склонны к переобучению
- основные параметры свертки: размер фильтра, receptive field
- pooling layer и stride > 1 помогает увеличить receptive field
- основной операцией в процессе работы сверточной нейронной сети является перемножение матриц