

Лекция 4

Сверточные нейронные сети

План

- Что такое операция свертка
- Основные блоки
- Общий вид архитектуры сверточных нейронных сетей
- Визуализация слоев, Deep Dream
- Style Transfer

ImageNet Challenge

			
mite	container ship	motor scooter	leopard
mite	container ship	motor scooter	leopard
black widow	lifeboat	go-kart	jaguar
cockroach	amphibian	moped	cheetah
tick	fireboat	bumper car	snow leopard
starfish	drilling platform	golfcart	Egyptian cat
			
grille	mushroom	cherry	Madagascar cat
convertible	agaric	dalmatian	squirrel monkey
grille	mushroom	grape	spider monkey
pickup	jelly fungus	elderberry	titi
beach wagon	gill fungus	ffordshire bullterrier	indri
fire engine	dead-man's-fingers	currant	howler monkey

Deep Dream



Style Transfer



PRISMA

Сверточные нейронные сети

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

1 <small>$\times 1$</small>	1 <small>$\times 0$</small>	1 <small>$\times 1$</small>	0	0
0 <small>$\times 0$</small>	1 <small>$\times 1$</small>	1 <small>$\times 0$</small>	1	0
0 <small>$\times 1$</small>	0 <small>$\times 0$</small>	1 <small>$\times 1$</small>	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

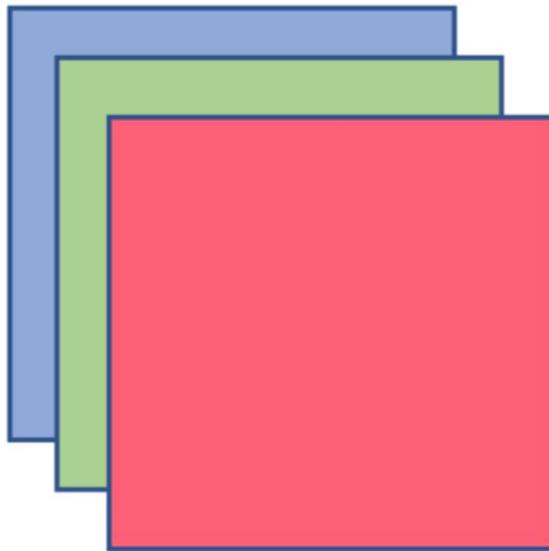
Convolved
Feature

Как посчитать свертку

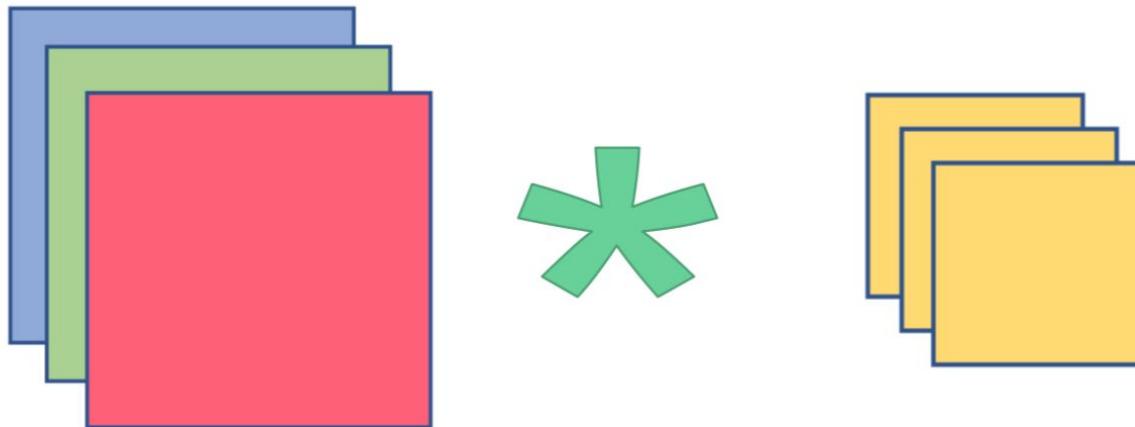
1. Имея входное изображение, наложить на него маленькое окно (ядро свертки). Найти произведение ядра и участка изображения которое накрыло ядро
2. Сдвинуть окно и повторить процедуру
3. Повторить шаги 1 и 2 для всех остальных выходных каналов уже с другими ядрами

Объемная свертка (Volume Convolution, Conv2d)

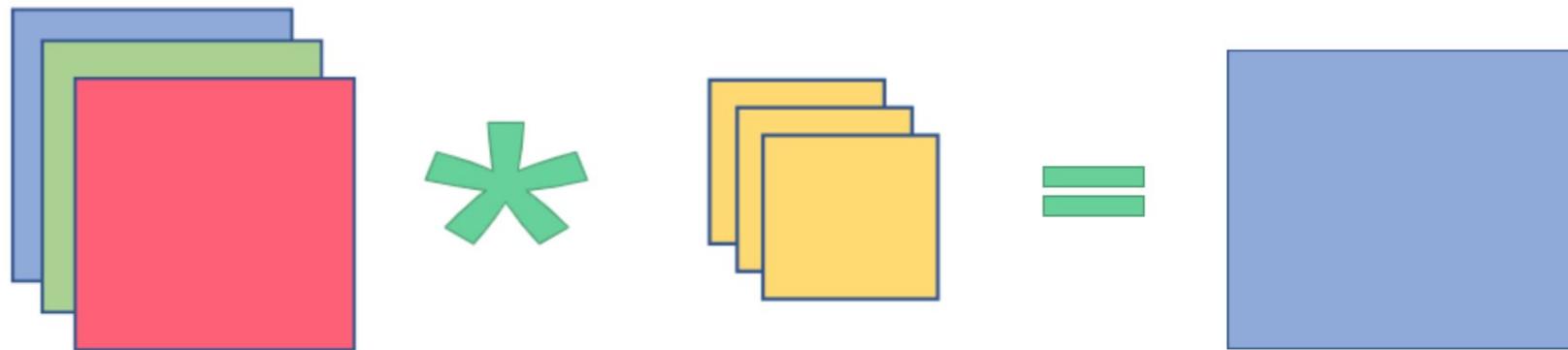
- Картинка имеет 3 канала, т.е. это 3d матрица



Объемная свертка (Volume Convolution, Conv2d)



Объемная свертка (Volume Convolution, Conv2d)



Как применить свертку?

0	0	0	0	0	0	...
0	156	155	156	158	158	...
0	153	154	157	159	159	...
0	149	151	155	158	159	...
0	146	146	149	153	158	...
0	145	143	143	148	158	...
...

Input Channel #1 (Red)

0	0	0	0	0	0	...
0	167	166	167	169	169	...
0	164	165	168	170	170	...
0	160	162	166	169	170	...
0	156	156	159	163	168	...
0	155	153	153	158	168	...
...

Input Channel #2 (Green)

0	0	0	0	0	0	...
0	163	162	163	165	165	...
0	160	161	164	166	166	...
0	156	158	162	165	166	...
0	155	155	158	162	167	...
0	154	152	152	157	167	...
...

Input Channel #3 (Blue)

-1	-1	1
0	1	-1
0	1	1

Kernel Channel #1



308

1	0	0
1	-1	-1
1	0	-1

Kernel Channel #2



-498

0	1	1
0	1	0
1	-1	1

Kernel Channel #3



164

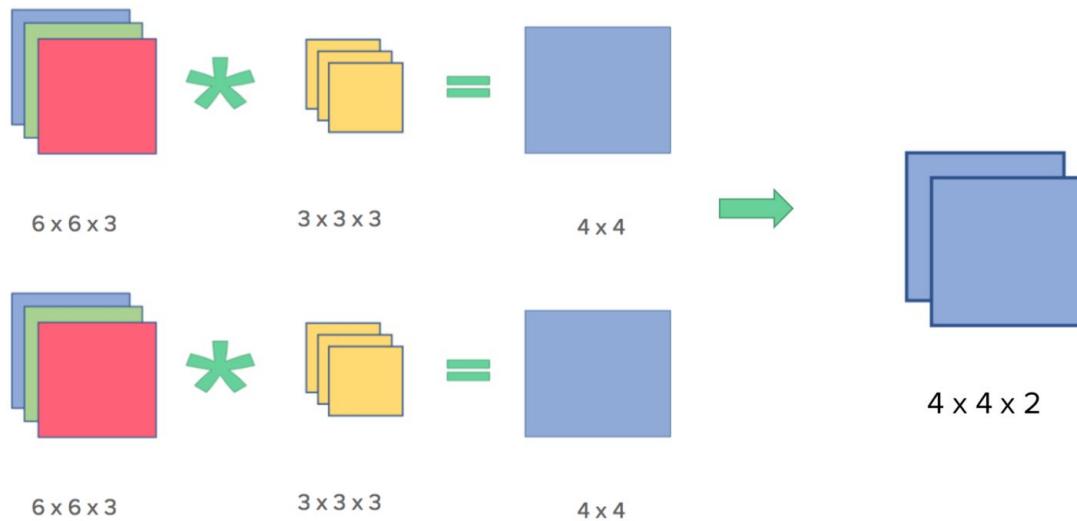
+ 1 = -25

Bias = 1
↑

-25				...
				...
				...
				...
...

Output

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. Свертка как способ извлекать локальные признаки
2. Свертка как способ уменьшить сложность модели

Свертка как детектор признаков



Согласны? Узнали? :)

Свертка как уменьшение сложности модели

Давайте сравним многослойный перцепtron и CNN.

Размер картинки на входе $3 \times 28 \times 28 = 3 \times 784$. Число классов на выходе равно 10.

Число параметров в 3x слойном перцептроне с размером скрытого слоя 32 равно

$$3 \times 784 \times 32 + 32 \times 32 + 32 \times 32 + 32 \times 10 = 77632$$

Свертка как уменьшение сложности модели

Давайте сравним многослойный перцепtron и CNN.

Размер картинки на входе $3 \times 28 \times 28 = 3 \times 784$. Число классов на выходе равно 10.

Число параметров в 10 слойной CNN с размером 5 фильтрами в каждом слое размером 3×3 , шагом 1 и no padding + полно связанный слой в конце

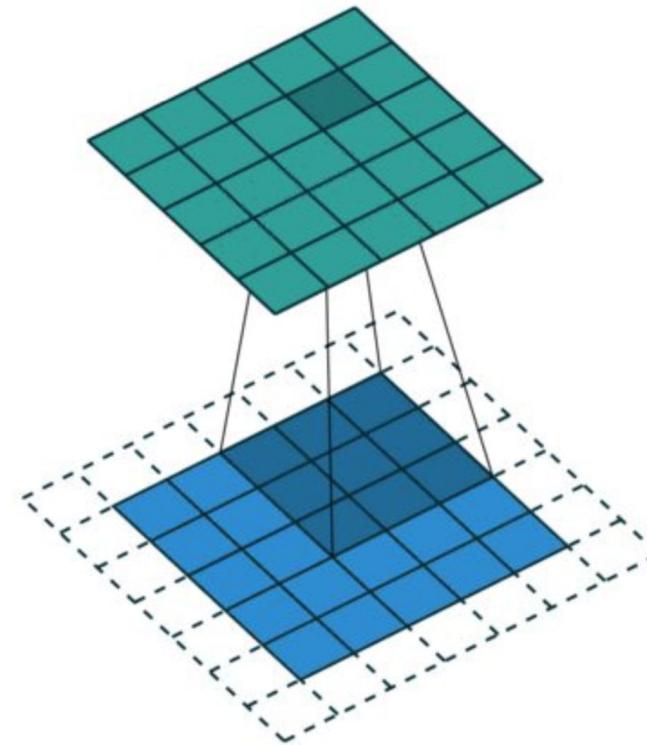
$$((3 \times 3 \times 3 \times 5) + 9 * (3 \times 3 \times 5 \times 5)) + ((5 \times 8 \times 8) * 10) \sim 5360$$

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)

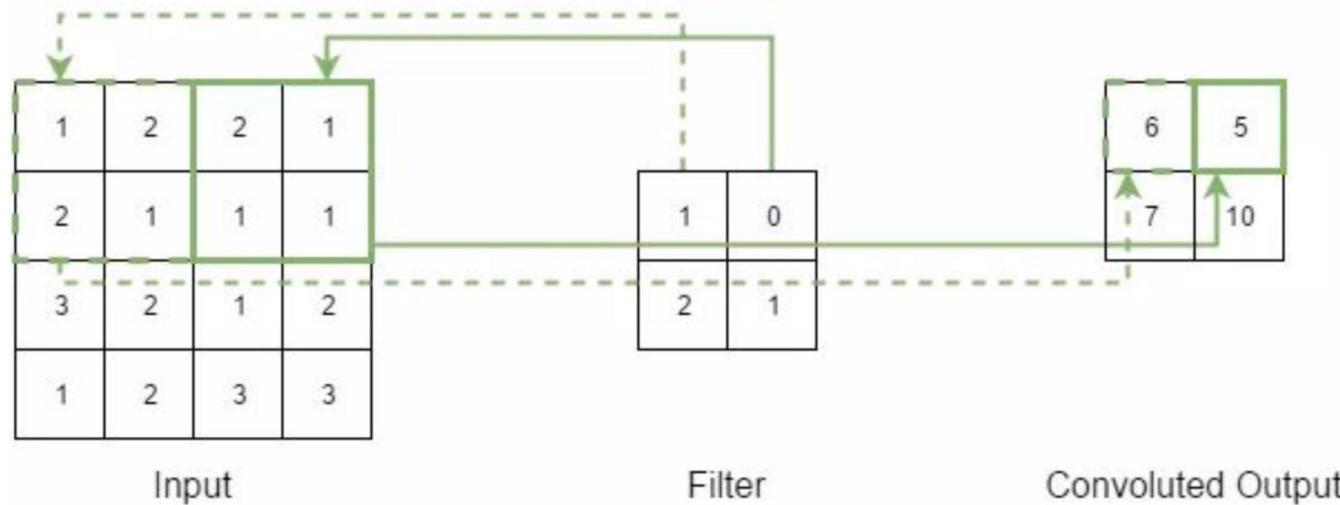
- при последовательном применении фильтров размер выхода будет уменьшаться с каждым шагом на размер фильтра
- при этом теряется информация о краях изображения

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



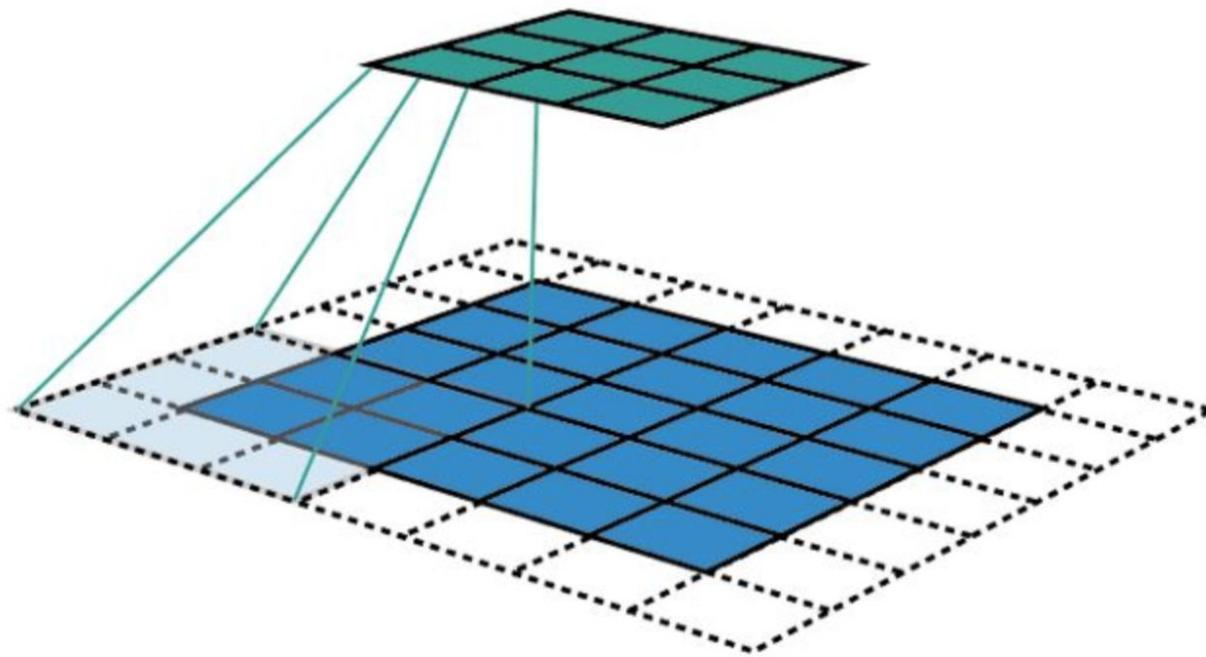
Шаг (Stride)

- шаг может быть разным по x и у
- определяет смещение фильтра на очередной итерации
- при увеличении шага уменьшается размер выхода
- шаг > 1 в начальных слоях
- шаг > 1 увеличивает receptive field (далее)

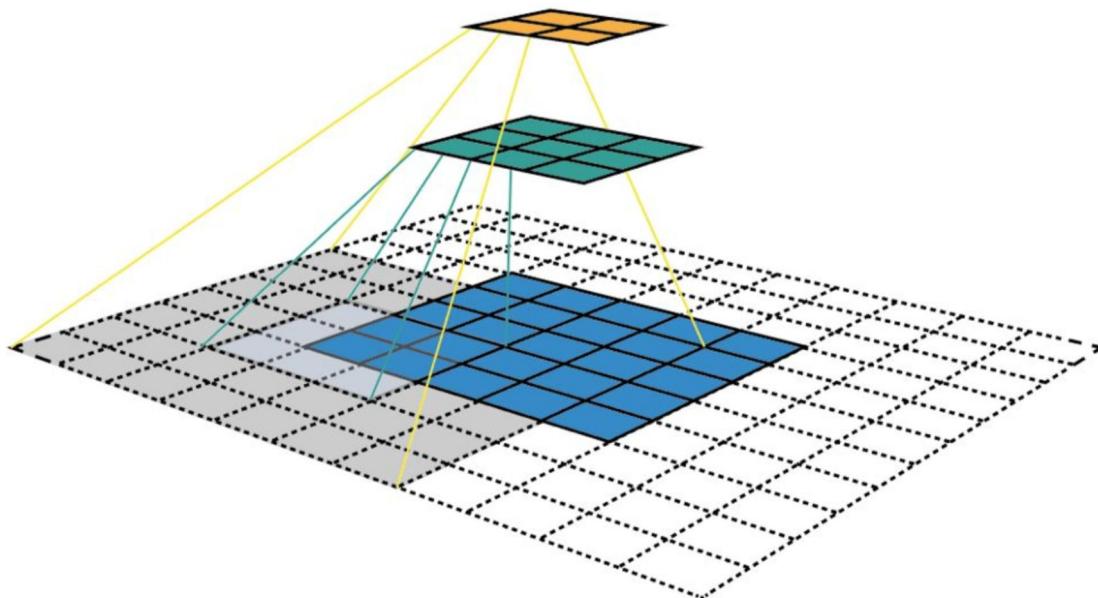
Область видимости фильтра (reception field)

- 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

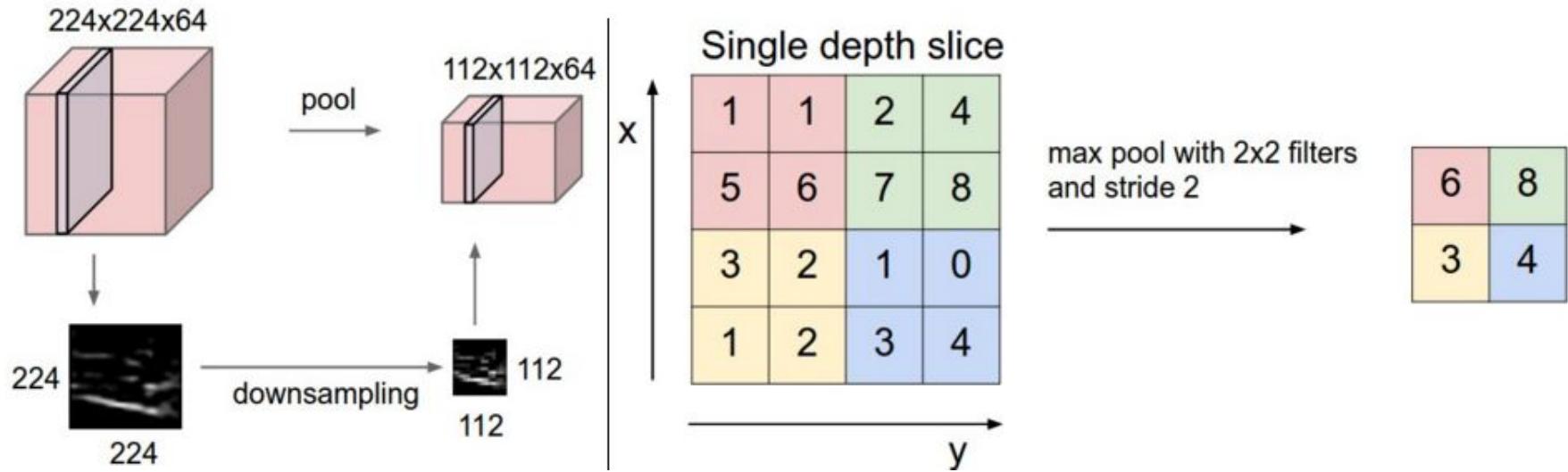
Pooling Layer

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)

Pooling Layer



Pooling Layer

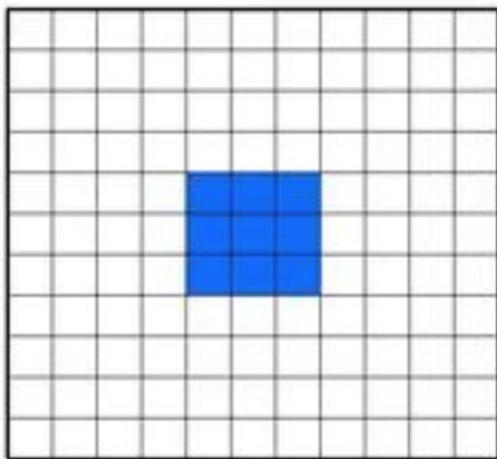
- pooling слой вставляется после активаций сверточного слоя
- уменьшает пространственный размер данных
- как результат уменьшается число параметров сети
- необучаемый слой - не содержит параметров

Расширение (dilation)

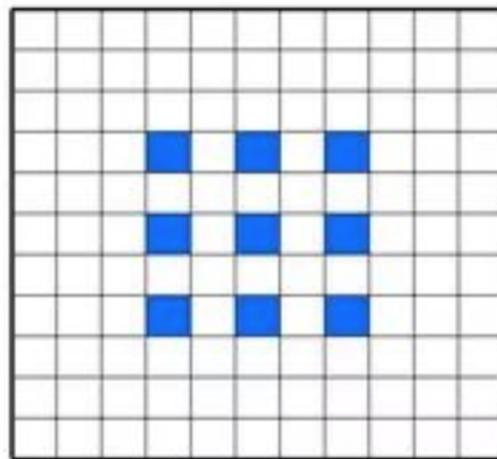
- увеличение размера фильтра (области видимости) без увеличения числа параметров (весов)
- веса фильтра “раздвигаются” в пространстве
- фильтр имеет разреженную структуру
- свободные позиции фильтра заполняются нулями

Расширение (dilation)

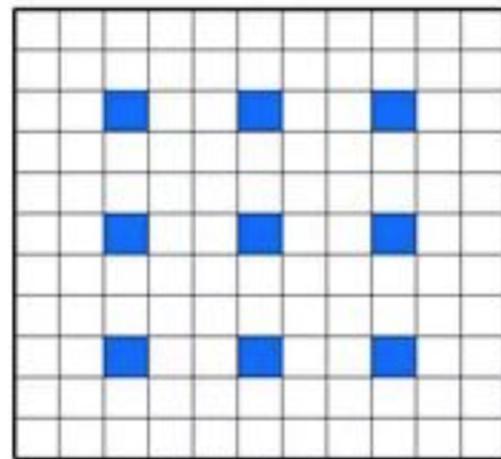
D = 1



D = 2



D = 3



Математика

- Размер входного тензора $n \times n \times c$
- Размер фильтра $f \times f \times c$
 - f обычно равно нечетному числу, например 3, 5, 7
- Размер картинки (тензора) на выходе $m \times m$

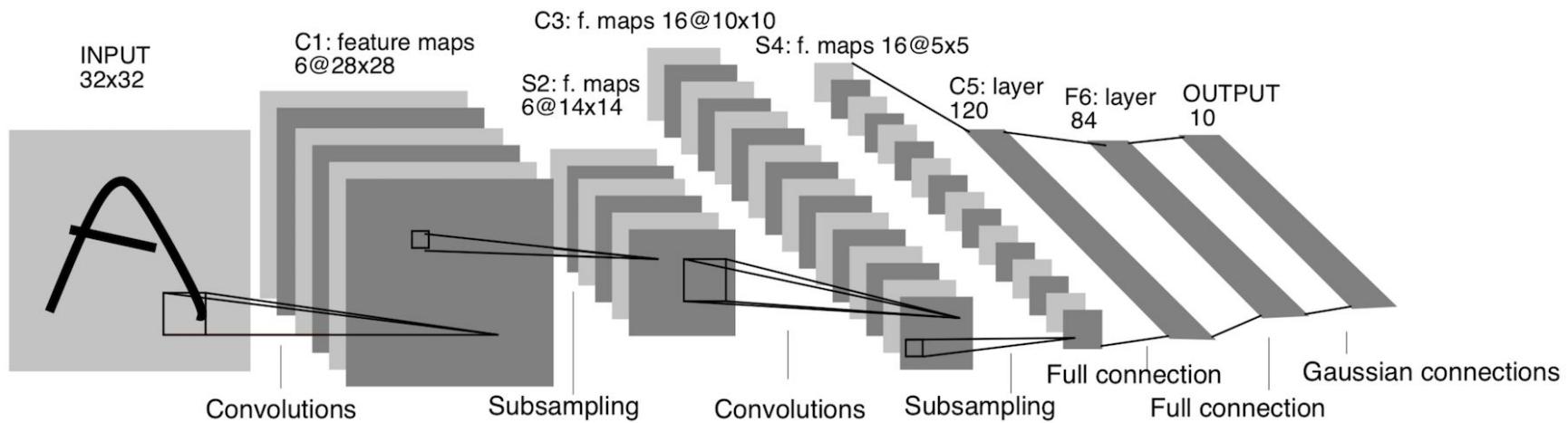
$$m = \left\lfloor \frac{n + 2p - f}{s} \right\rfloor + 1$$

- где S - размер шага, p размер padding с одной стороны

Сверточный слой (Convolution layer)

- Содержит набор из k фильтров одинакового размера (f, f, c)
- Размер входа (bs, n, n, c)
- Размер выхода (bs, m, m, k)
- На выходе применяется нелинейная активация

LeNet (1998)



Сверточные нейронные сети

- в нейронных сетях для обработки изображений полно связные слои заменяют на сверточные
- полно связные слои (+dropout) могут использоваться в выходном слое
- Сверточный блок обычно состоит из:
Conv2d -> BatchNormalization -> Нелинейная активация (ReLU)->Pooling

Batch Normalization

Batch normalization

- приводит значения активаций на выходе слоя к нулевому среднему и единичной дисперсии
- во время обучения среднее и дисперсия оцениваются для каждого батча
- такая нормировка дает прирост как в качестве, так и в скорости сходимости процесса обучения
- повышается обобщающая способность сети

Batch normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_1 \dots m\}$;

Parameters to be learned: γ, β

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

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

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

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad // \text{normalize}$$

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

Batch normalization

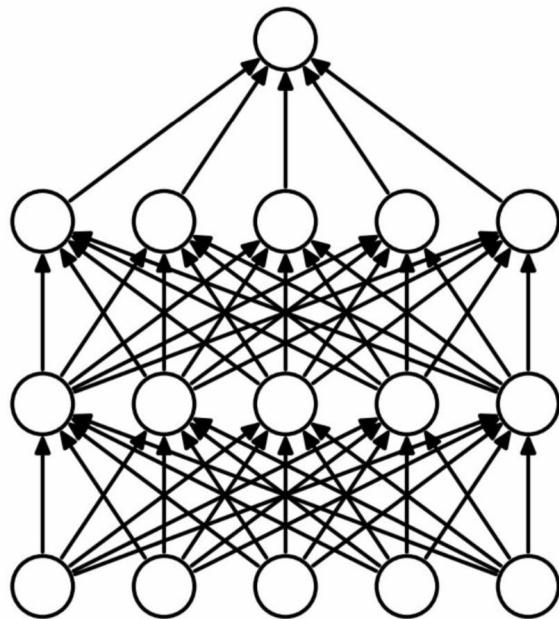
CLASS `torch.nn.BatchNorm2d(num_features, eps= $1e-05$, momentum=0.1, affine=True,
track_running_stats=True)`

Dropout

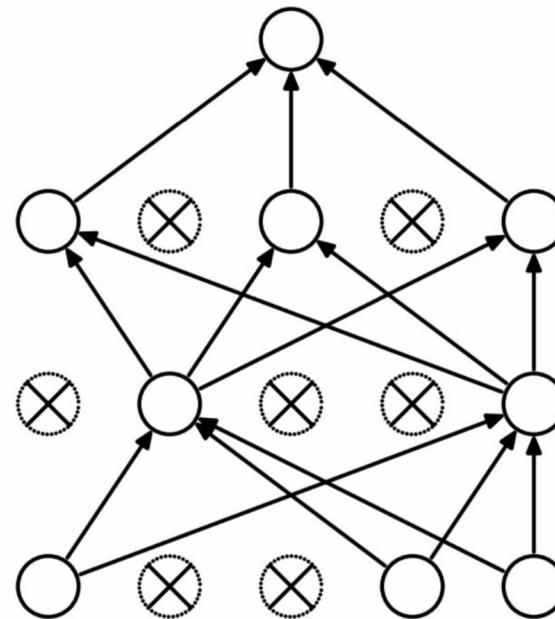
Dropout

- в процессе обучения случайным образом выбранная часть входов слоя зануляется
- доля зануляемых входов задается параметром слоя
- на этапе предсказания значения входов перевзвешиваются пропорционально доли зануленных входово
- такой подход позволяет модели быть более устойчивой к переобучению

Dropout



(a) Standard Neural Net

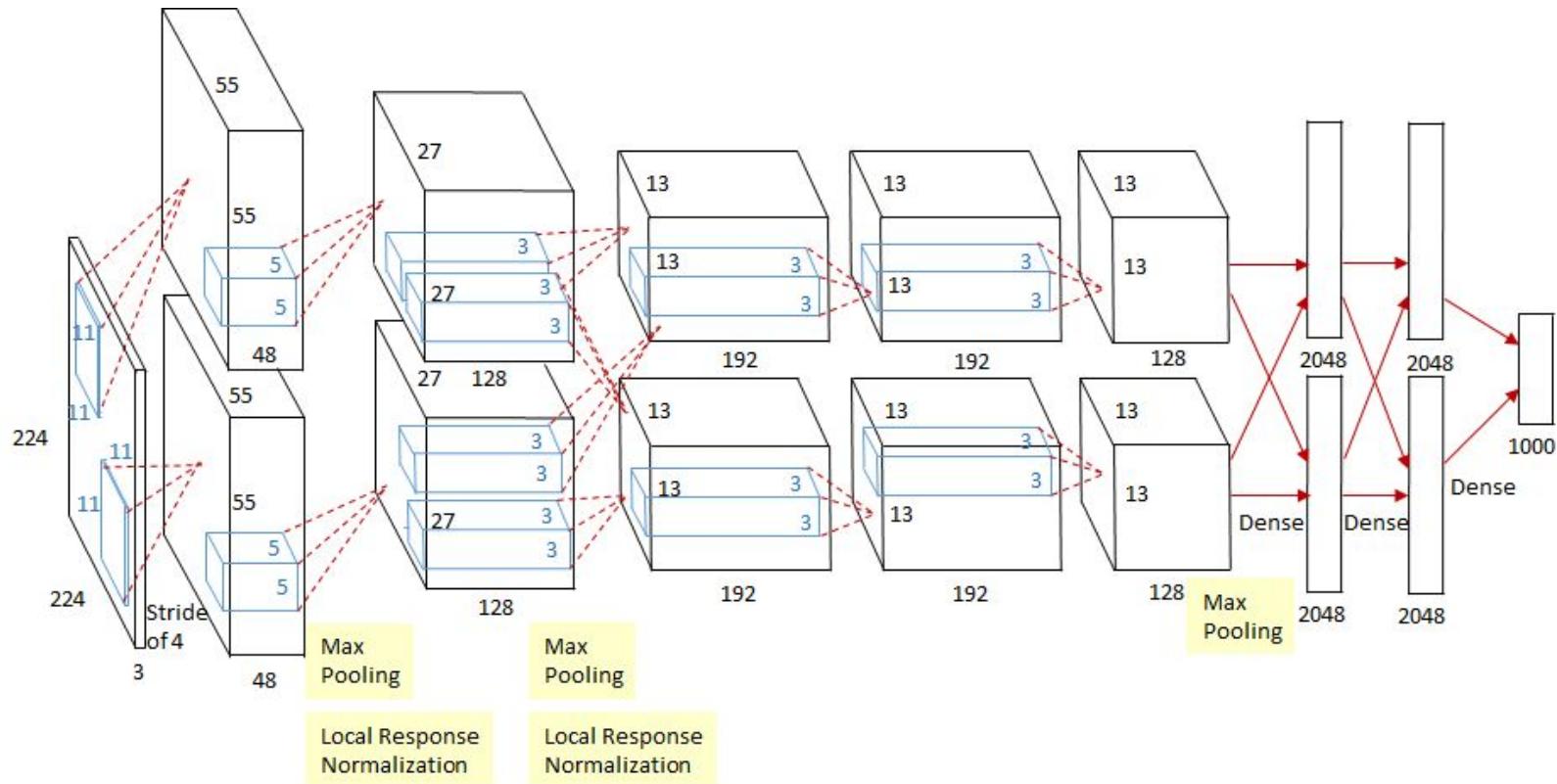


(b) After applying dropout.

Dropout

CLASS `torch.nn.Dropout(p=0.5, inplace=False)`

AlexNet (2012)

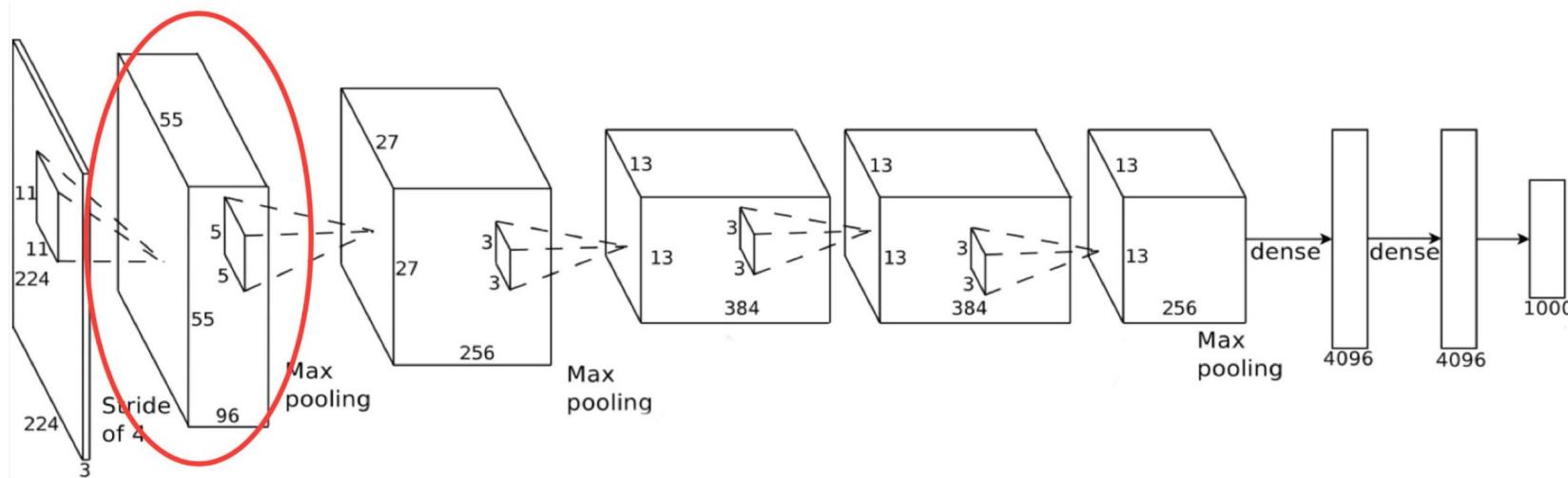


Визуализация сверточных сетей

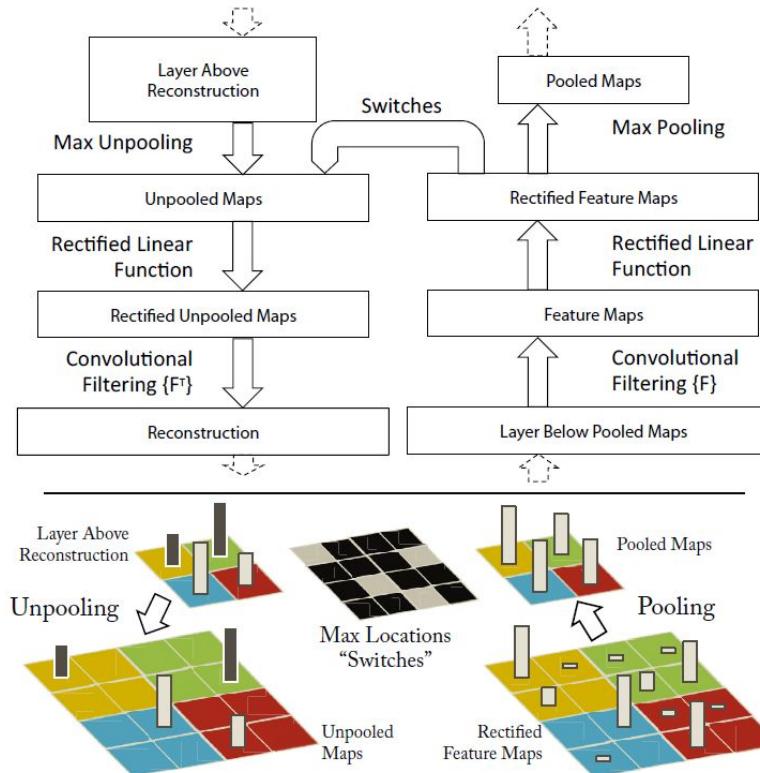
Визуализация сверточных слоев

- рассматриваем фильтры на различных слоях
- на первом слое фильтры имеют простые геометрические формы
- с увеличением глубины, фильтры выучивают более сложные паттерны и визуально становятся похожи на части объектов

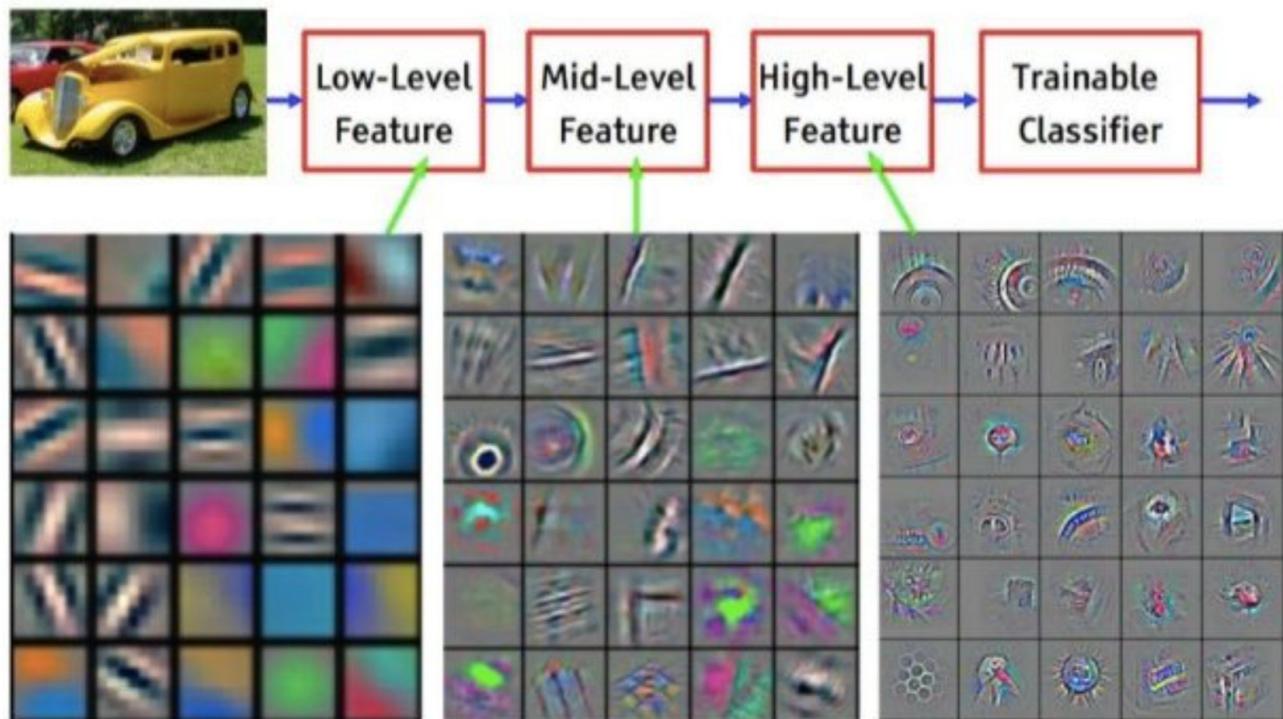
Визуализация сверточных слоев



Визуализация сверточных слоев



Визуализация сверточных слоев



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

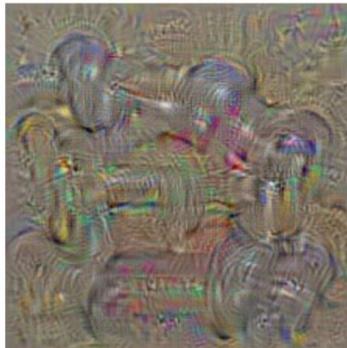
Image Optimization

- выбираем предобученную сеть
- выбираем класс и фиксируем его на выходном слое
- решаем обратную задачу - подбираем такое изображение, которое максимизирует вес выбранного класса

$$\arg \max_I S_c(I) - \lambda \|I\|_2^2$$

- I - изображение
- Sc - вес выбранного класса
- lambda - параметр регуляризации

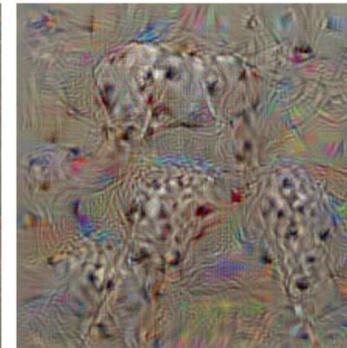
Image Optimization



dumbbell



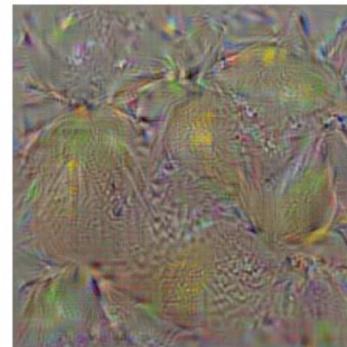
cup



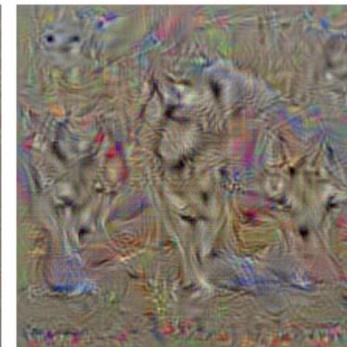
dalmatian



bell pepper



lemon



husky

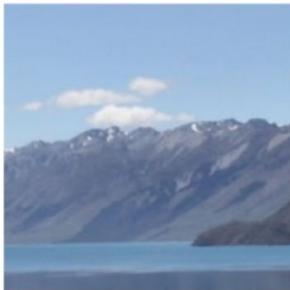
Deep Dream

- выбираем предобученную сеть
- фиксируем интересующий нас сверточный слой
- вычисляем активации на выбранном слое
- распространяем обратно значения активаций как значения градиента
- обновляем изображение с учетом полученных градиентов и повторяем итерацию

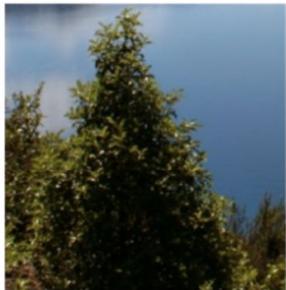
Deep Dream

- результат будет зависеть от того, на каких данных обучалась нейронная сеть
- детали рисунка будут зависеть от глубины слоя с которого распространяются активации

Deep Dream



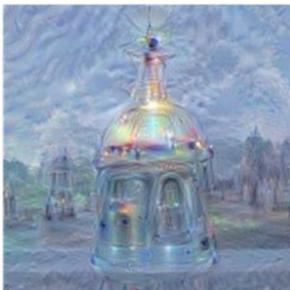
Horizon



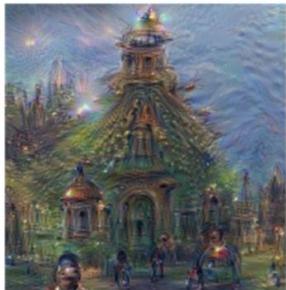
Trees



Leaves



Towers & Pagodas



Buildings

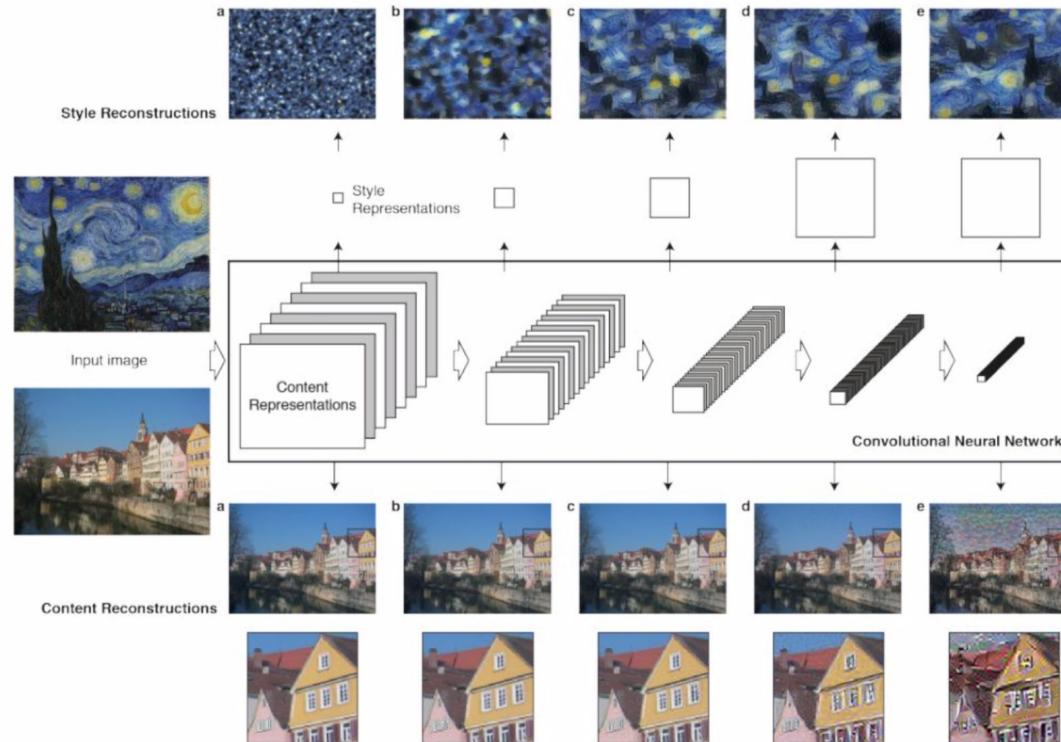


Birds & Insects

Style Transfer

- сгенерируем такое изображение, которое максимально похоже на исходное
- при этом, стиль сгенерированного изображения должен быть похож на стиль референсного изображения
- активации сгенерированного изображения должны быть близки к активациям исходного
- ковариационная матрица фильтров сгенерированного изображения должна быть близка к матрице референсного изображения

Style Transfer



Style Transfer - Content Loss

$$\mathcal{L}_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} (F_{ij}^l - P_{ij}^l)^2$$

- F - активация пикселя i,j на слое l исходного изображения
- P - активация пикселя i,j на слое l сгенерированного изображения

Style Transfer - Style Loss

$$G_{ij}^l = \sum_k F_{ik}^l F_{jk}^l.$$
$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G_{ij}^l - A_{ij}^l)^2$$

$$\mathcal{L}_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^L w_l E_l$$

i,j - индексы фильтров

k - номер активации

Fk - значение активации k-го пикселя

G, A - матрица ковариации фильтров для заданного слоя

N - число фильтров в слое

M - число активаций (размер изображения на выходе сверточного слоя)

E - мера расхождения в стиле

w - вес ошибки для заданного слоя

L - число слоев

Style Transfer - Total Loss

$$\mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{style}(\vec{a}, \vec{x})$$

р - исходное изображение

а - изображение с референсным стилем

х - сгенерированное изображение

alpha, beta - параметры обучения

Style Transfer

1

Upload photo

The first picture defines the scene you would like to have painted.



2

Choose style

Choose among predefined styles or upload your own style image.



3

Submit

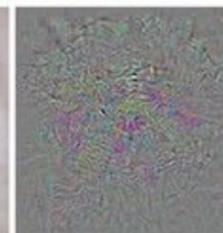
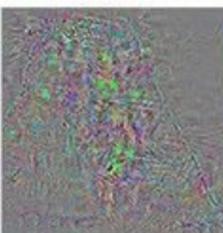
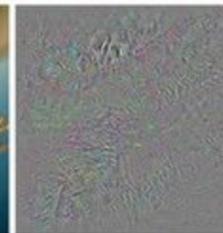
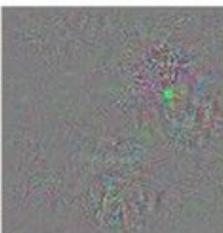
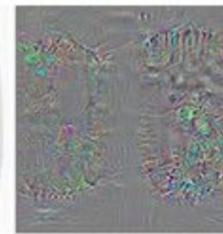
Our servers paint the image for you. You get an email when it's done.



<https://deepart.io>

Обманываем сверточную сеть

Обманываем сверточную сеть



correct

+distort

ostrich

correct

+distort

ostrich

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 $\text{stride} > 1$ помогает увеличить receptive field
- основной операцией в процессе работы сверточной нейронной сети является перемножение матриц

Recap

- с увеличением глубины слоя признаки становятся более общими (увеличивается область видимости изображения)
- для оценки стиля изображения можно использовать ковариационную матрицу фильтров для каждого слоя
- добавление специального шума может заставить нейронную сеть выдавать неверные ответы

Полезные материалы

- [A guide to convolution arithmetic for deep learning](#)
- [Introducing Activation Atlases](#)