

ML для изображений и сверточные нейросети

Иван Карпухин, ведущий исследователь-разработчик

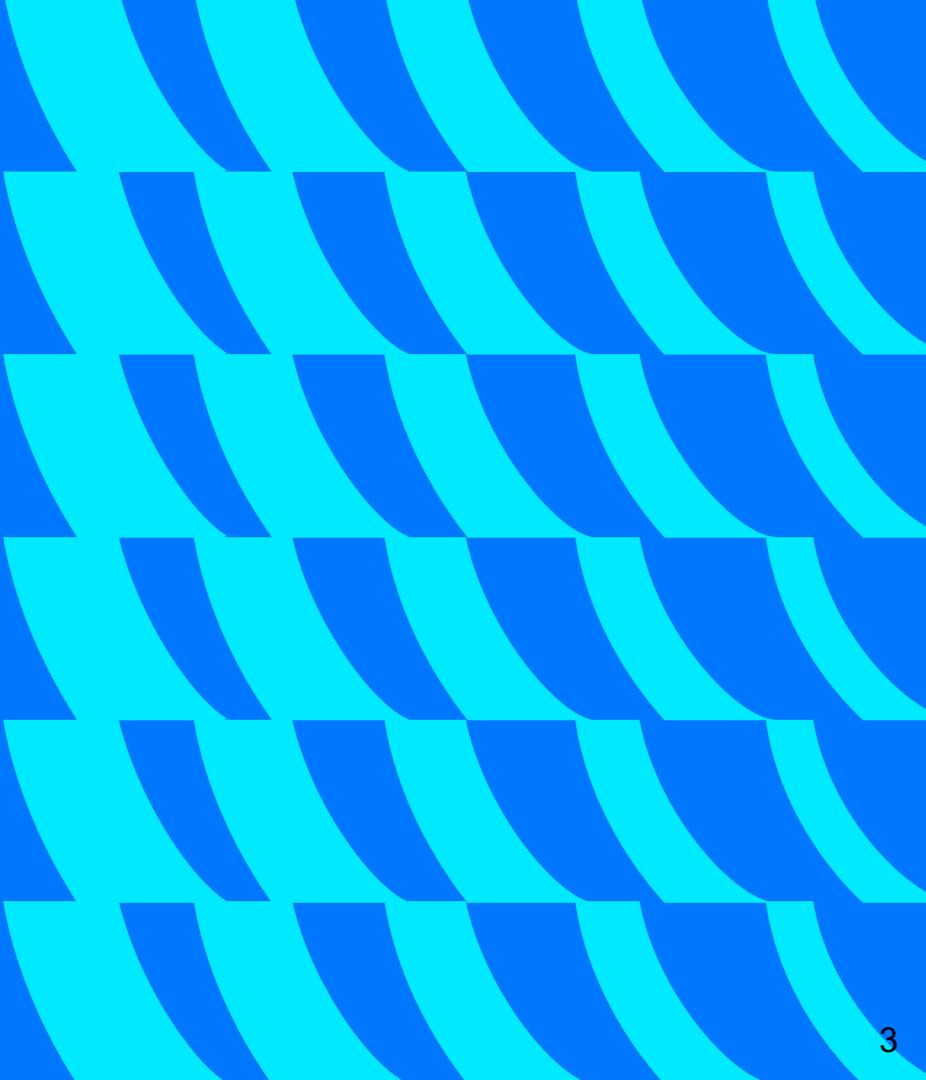


План лекции

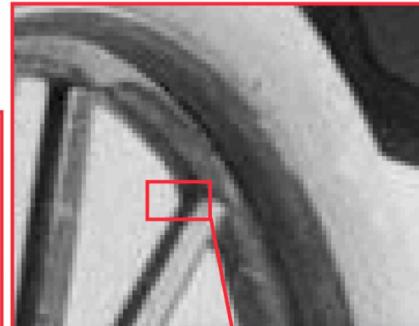
Семинар

- Подробно рассмотрим CNN
 - Как устроены изображения
 - Сверточный слой
 - Пулинг
 - Batchnorm
 - Convolutional Neural Network (CNN)
- Вспомним методы оптимизации (если успеем)
 - Momentum/Adam
 - Регуляризация
- Обучение CNN на PyTorch

Как устроены изображения

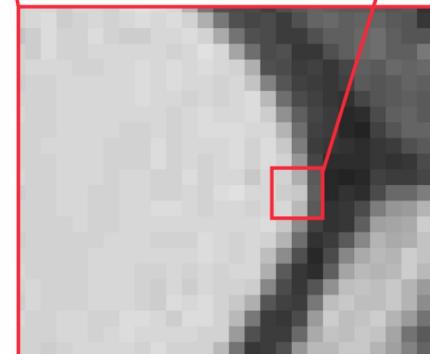


Изображение = пиксели



[214, 176, 90]
[207, 204, 97]
[218, 186, 93]

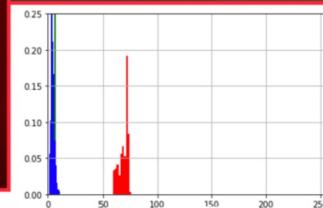
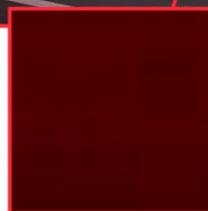
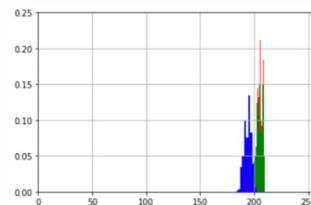
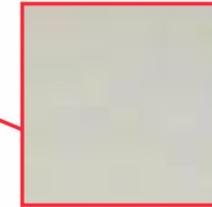
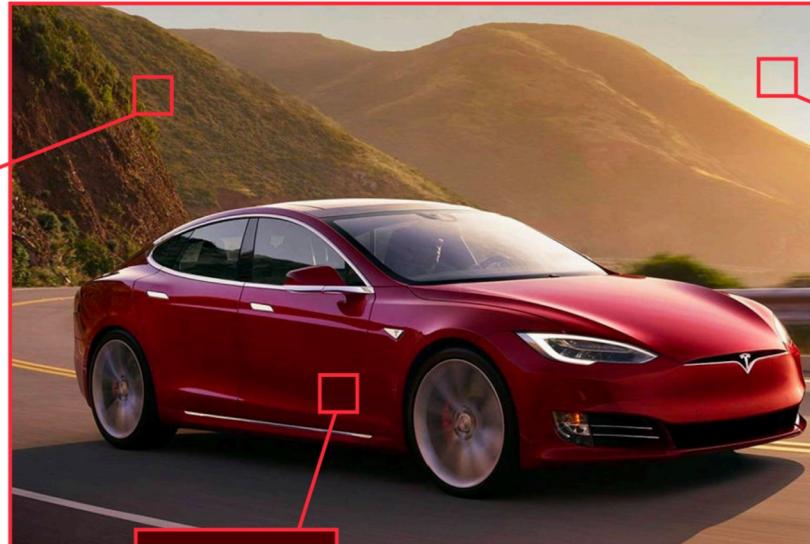
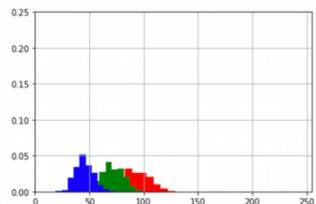
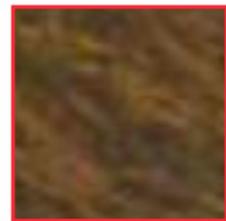
214	176	90
207	204	97
218	186	93



Представление цвета



Представление цвета



ML на изображениях



Как засунуть изображение в модель ML

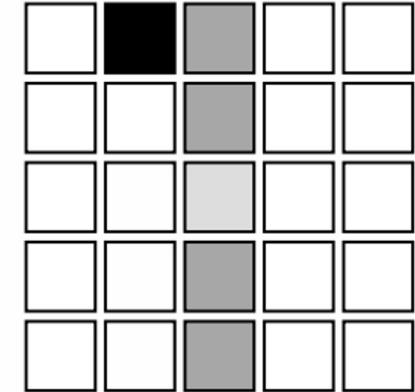
- Для многих ML моделей (линейных/лесов/...) хорошо подходят "табличные" данные
 - объект = набор численных признаков
- Как превратить изображение в набор признаков?

Как засунуть изображение в модель ML

- Для многих ML моделей (линейных/лесов/...) хорошо подходят "табличные" данные
 - объект = набор численных признаков
- Как превратить изображение в набор признаков?
 - **Признаки = исходные значения яркости пикселей**

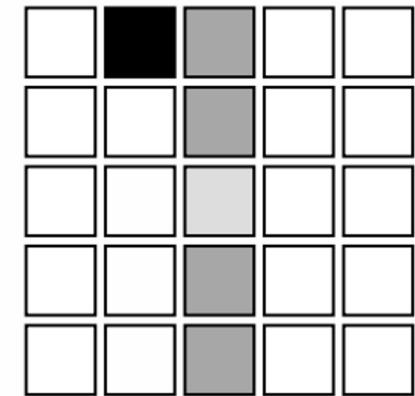
Признаки = яркости всех пикселей

- Хотим сделать классификатор grayscale-изображений цифр от 0 до 9
- Размер изображений - 5 x 5 пикселей
- Используем полносвязную нейросеть с 1 скрытым слоем
- Какова размерность весов этого слоя?

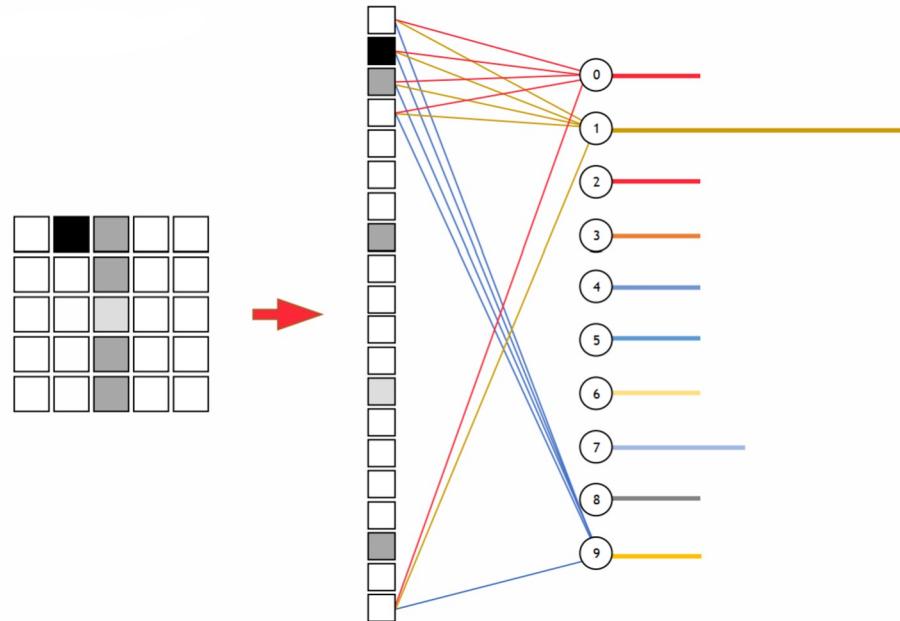


Признаки = яркости всех пикселей

- Хотим сделать классификатор grayscale-изображений цифр от 0 до 9
- Размер изображений - 5 x 5 пикселей
- Используем полносвязную нейросеть с 1 скрытым слоем
- Какова размерность весов этого слоя?
 - Число входов = 25
 - Число выходов = 10
 - Число весов ~ 250 (без bias)

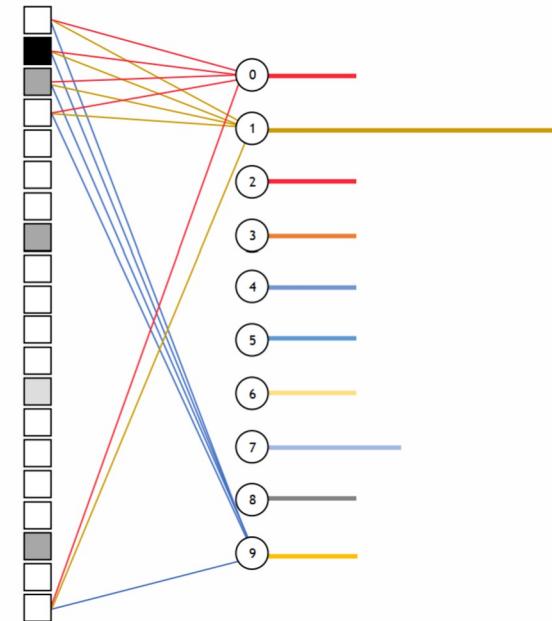
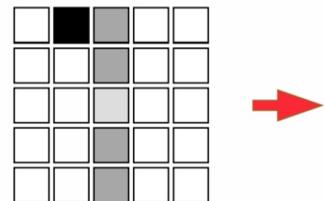


Признаки = яркости всех пикселей



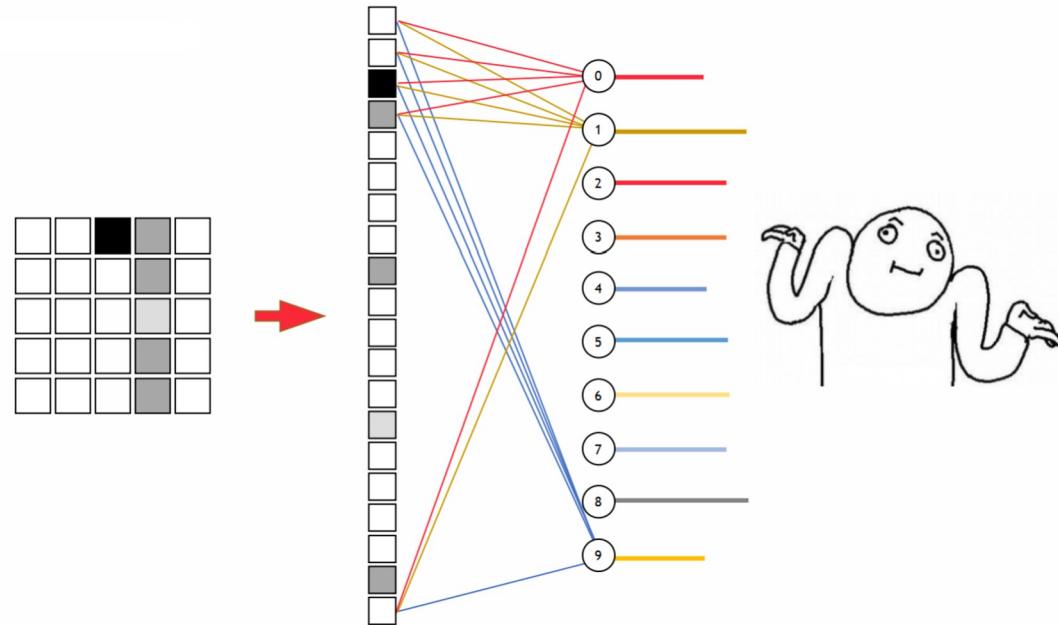
Признаки = яркости всех пикселей

К чему приведет
смещение объекта на 1
пиксель?



Признаки = яркости всех пикселей

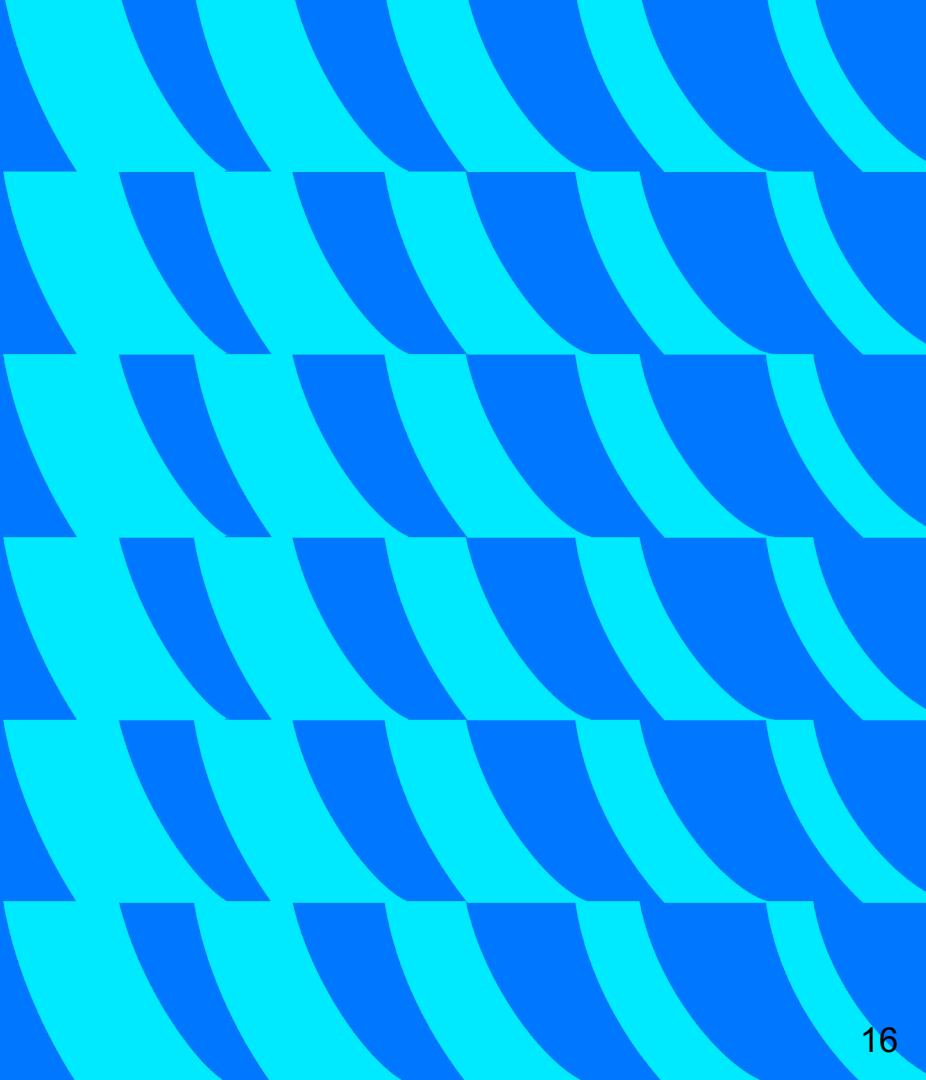
Проблема: нет
инвариантности к
смещениям



Признаки = яркости всех пикселей

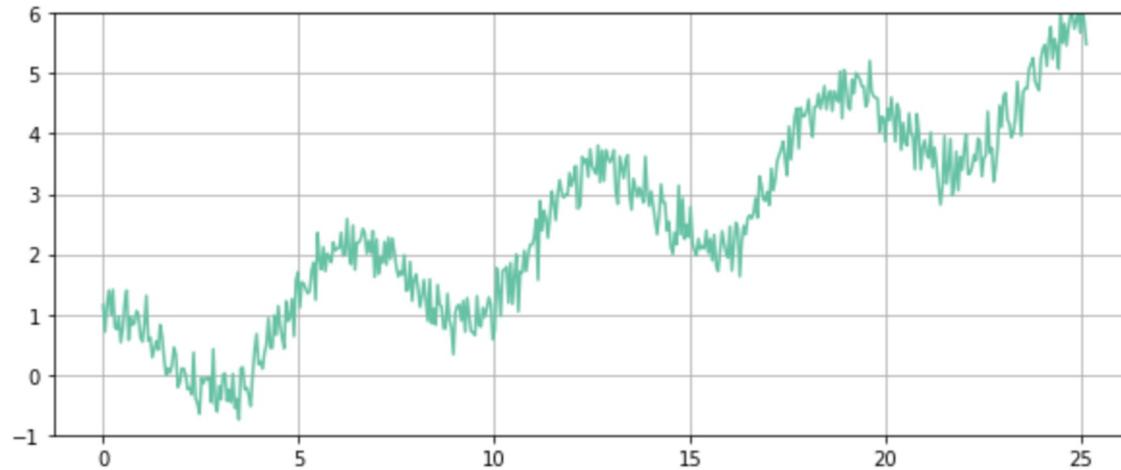
- Пример: ImageNet
- Изображения 224x224x3
- 1000 классов
- Число параметров однослойной нейросети:
 - $224 \times 224 \times 3 \times 1000 \sim 150\ 000\ 000$

Свертка



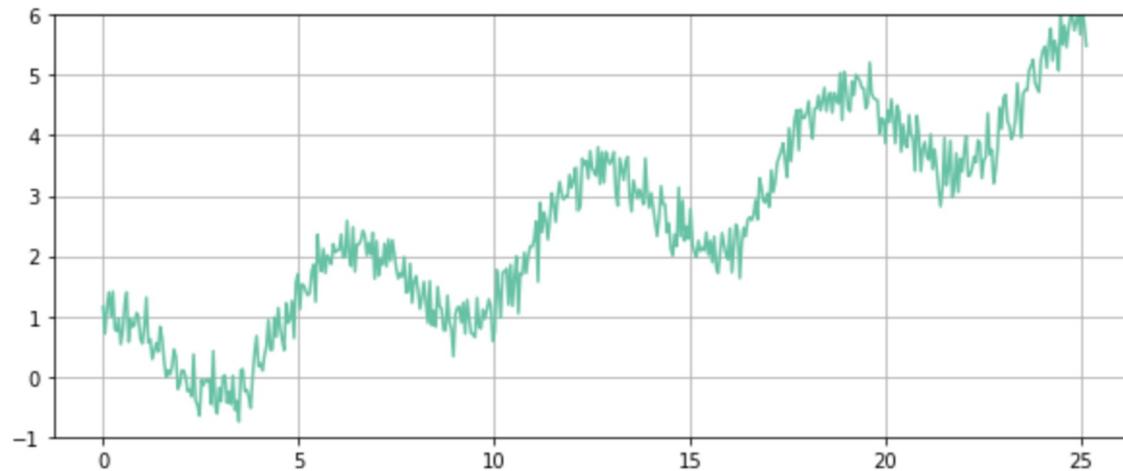
Пример

- Имеется некоторый сигнал
 - Например, показания физического датчика за промежуток времени
- Сигнал явно шумный - как его "почистить"?



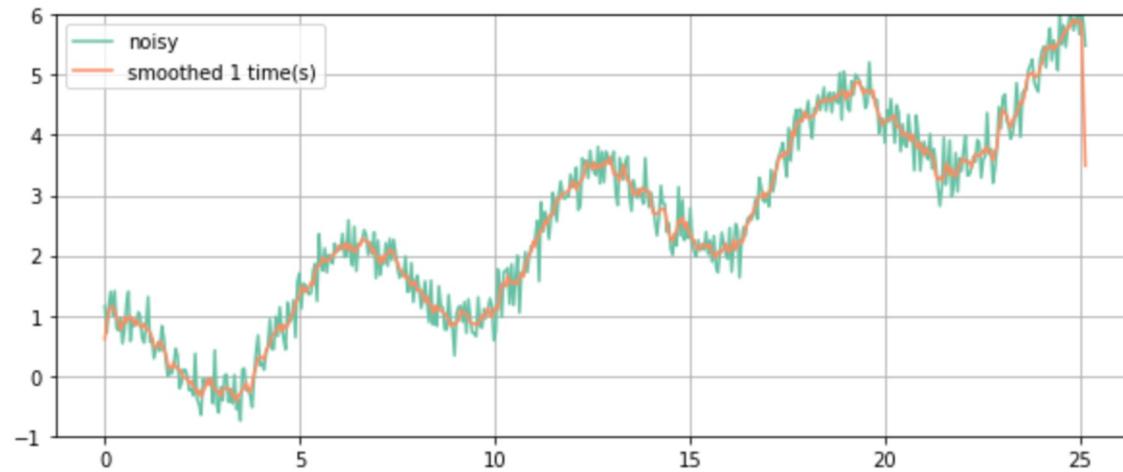
Пример

- Применим к "сглаживание" с помощью скользящего среднего
- Выберем размер окна; например, 5
- Каждое значение сигнала заменим средним по окрестности размера 5



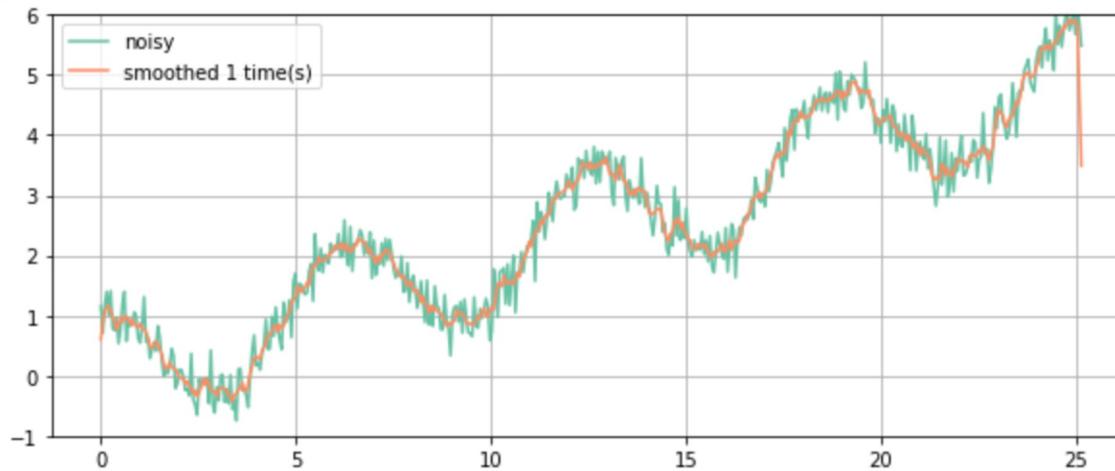
Пример

- Применим к "сглаживание" с помощью скользящего среднего
- Выберем размер окна; например, 5
- Каждое значение сигнала заменим средним по окрестности размера 5



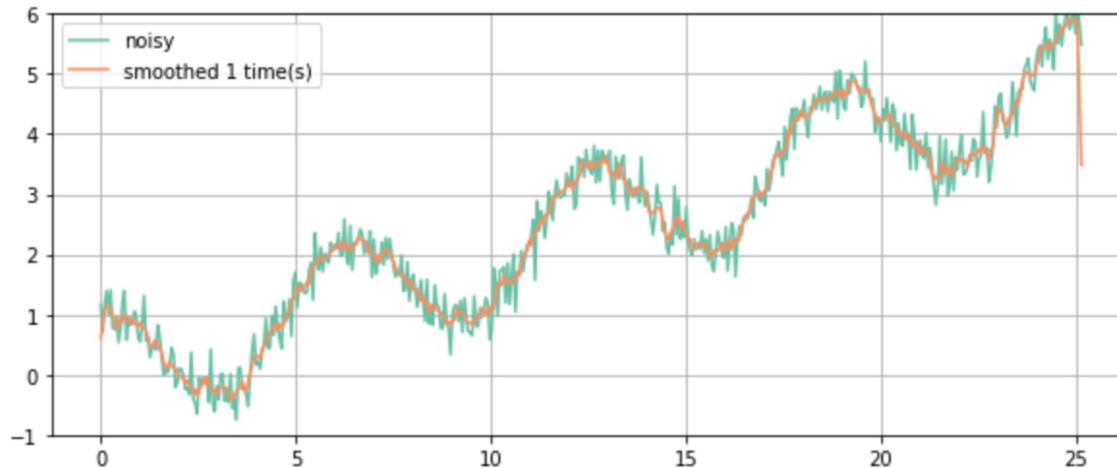
Пример

- Сигнал стал более "гладким"
- Появились краевые эффекты - почему?



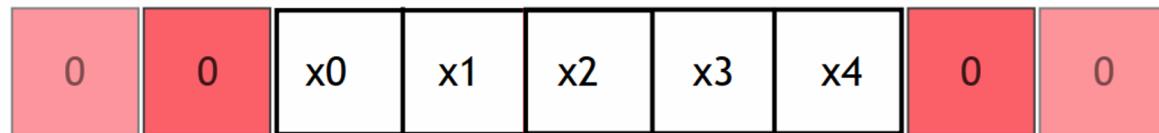
Пример

- Сигнал стал более "гладким"
- Появились краевые эффекты - почему?
 - Для первых и последних точек нет полного числа соседей, поэтому пришлось добавить нули по краям сигнала

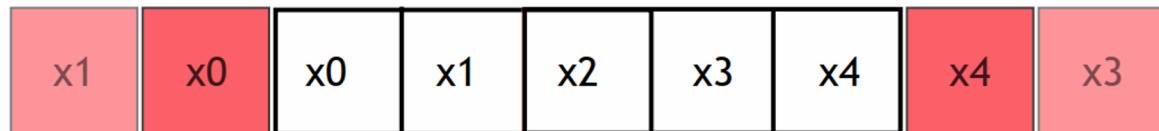


Padding

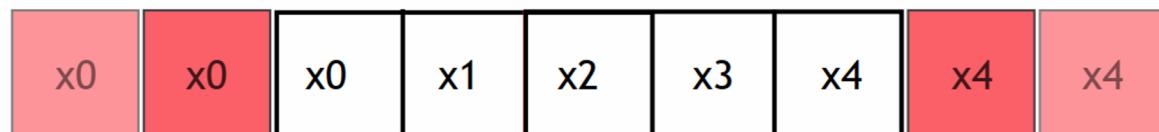
- Zero padding



- Reflect padding

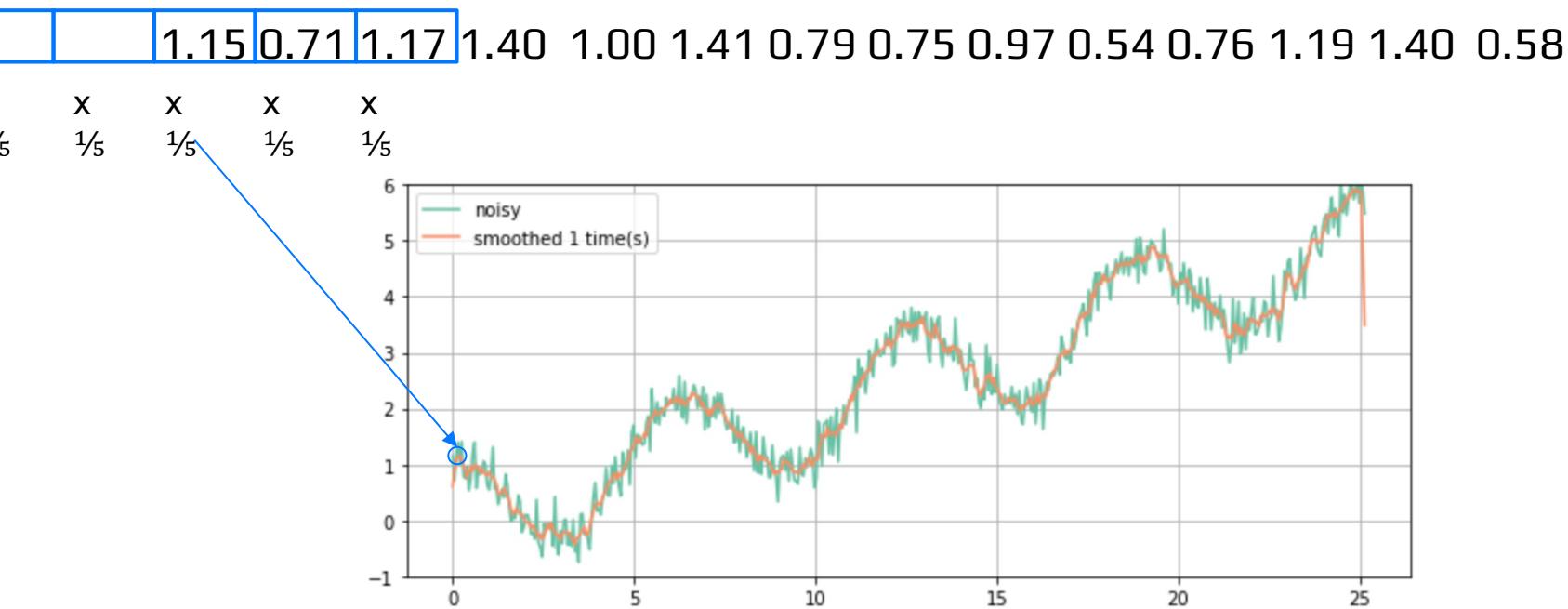


- Replicate padding



Пример

- Что именно мы сделали?

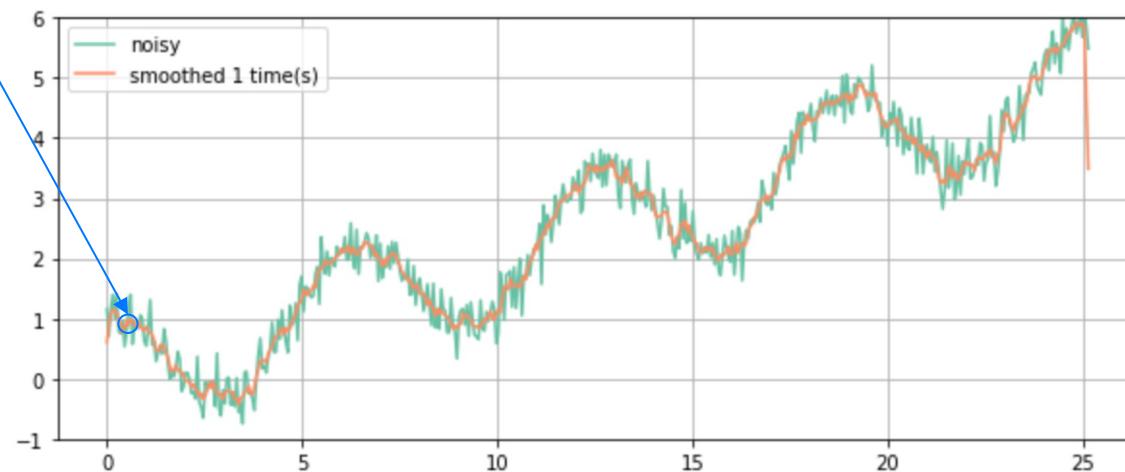


Пример

- Что именно мы сделали?

	1.15	0.71	1.17	1.40	1.00	1.41	0.79	0.75	0.97	0.54	0.76	1.19	1.40	0.58
--	------	------	------	------	------	------	------	------	------	------	------	------	------	------

X X X X X
 $\frac{1}{5}$ $\frac{1}{5}$ $\frac{1}{5}$ $\frac{1}{5}$ $\frac{1}{5}$

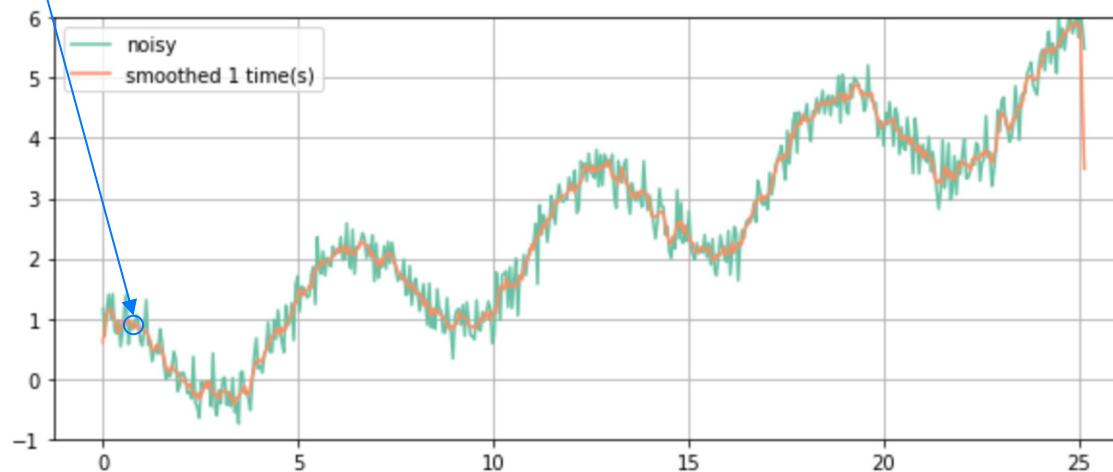


Пример

- Что именно мы сделали?

1.15 0.71 1.17 1.40 1.00 1.41 0.79 0.75 0.97 0.54 0.76 1.19 1.40 0.58

X X X X X
 $\frac{1}{5}$ $\frac{1}{5}$ $\frac{1}{5}$ $\frac{1}{5}$ $\frac{1}{5}$

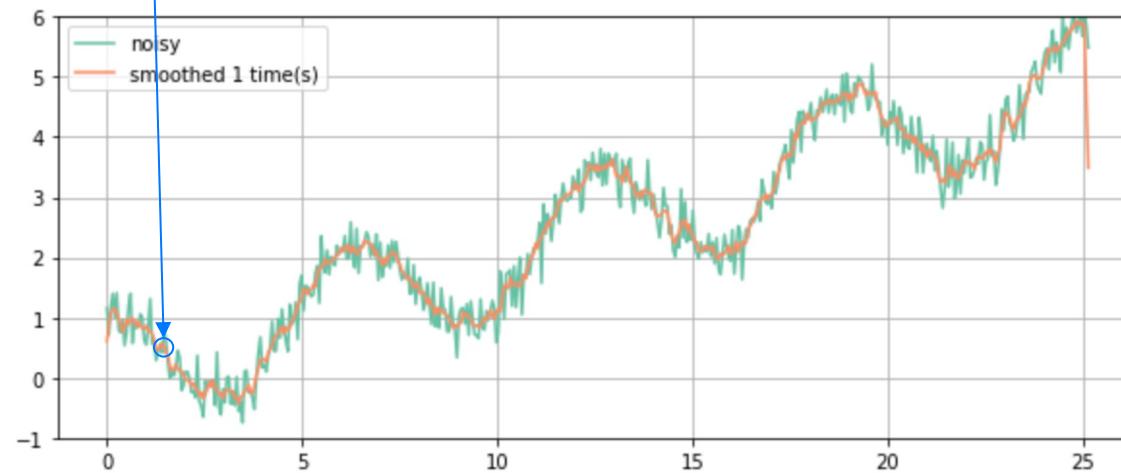


Пример

- Что именно мы сделали?

1.15 0.71 1.17 1.40 1.00 1.41 0.79 0.75 0.97 0.54 0.76 1.19 1.40 0.58

X X X X X
 $\frac{1}{5}$ $\frac{1}{5}$ $\frac{1}{5}$ $\frac{1}{5}$ $\frac{1}{5}$



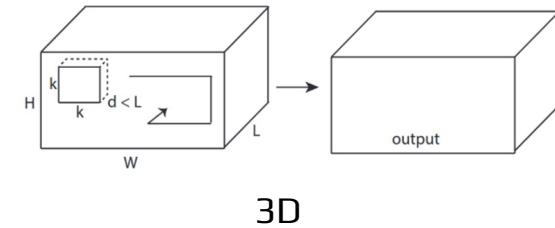
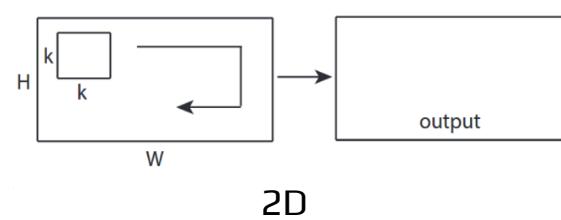
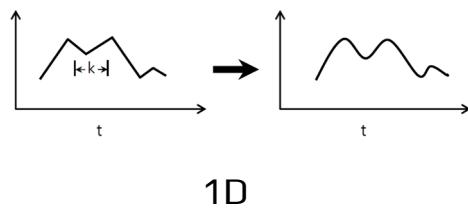
Свертка

- Свертка - операция, при которой
 - Ядро свертки "скользит" по входному сигналу
 - В каждой точке приложения ядра вычисляется скалярное произведение
 - Результат записывается в выходной сигнал
- Дискретная формула свертки (1D) сигнала y с ядром w размера K :

$$y_i = (x * w)_i = \sum_{k=0}^{K-1} x_{i+k} \cdot w_k$$

Свертка

- Сглаживание сигнала в прошлом примере = одномерная свертка
- Свертка может быть и более высокой размерности
- Размерность свертки = количество направлений, вдоль которых движется ядро свертки, проходя по сигналу



Свертка - двумерный случай

Входной сигнал X_{ij}

X00	X01	X02	X03
X10	X11	X12	X13
X20	X21	X22	X23
X30	X31	X32	X33

$$Y_{ij} = (X * W)_{ij} = \sum_{u=0}^{K-1} \sum_{v=0}^{K-1} X_{i+u, j+v} W_{uv}$$

Результат свертки Y_{ij}

Y00	Y01
Y10	Y11

Ядро свертки W
Размер **3x3**
Веса W_{ij}

Свертка

- 2D-свертки с разными ядрами используются в обработке изображений
 - Сглаживание
 - Выделение границ и углов
 - Удаление шумов
 - ...

Свертки для обработки изображений

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

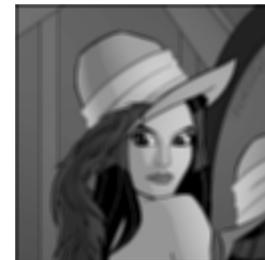


Смещение на 1 пиксель вправо



0	0	0
-1	0	1
0	0	0

???



Размытие окном 3x3

0	0	0
1	0	0
0	0	0



Выделение вертикальных границ

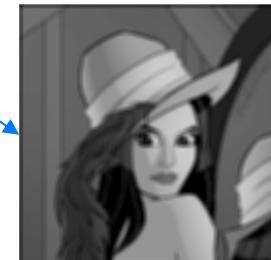
Свертки для обработки изображений

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9



Смещение на 1 пиксель вправо

0	0	0
-1	0	1
0	0	0



Размытие окном 3x3

0	0	0
1	0	0
0	0	0



Выделение горизонтальных границ



Свертки для обработки изображений

- Если во входном изображении больше 1 канала (например, RGB)?



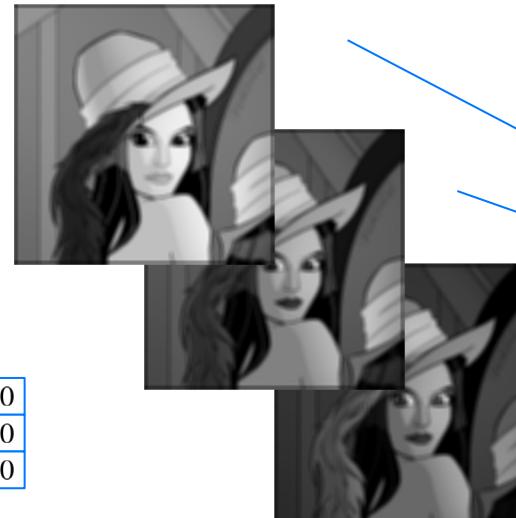
Тензор $H \times W \times 3$

$$\begin{matrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{matrix}$$

$$\begin{matrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{matrix}$$

$$\begin{matrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{matrix}$$

3 двумерных
свертки



3 Тензора $H \times W$

Σ

Тензор $H \times W$

Свертки для обработки изображений

- Если во входном изображении больше 1 канала (например, RGB)?



Тензор $H \times W \times 3$

$$\begin{matrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{matrix}$$

...

$$\begin{matrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{matrix}$$

К двумерных
сверток



Тензор $H \times W \times K$

Свертка - свойства

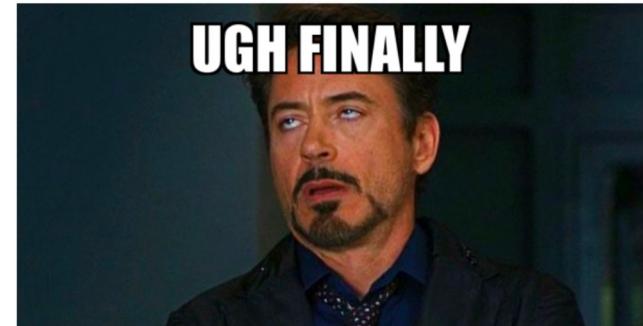
- Ассоциативность:
 - $a * (b * c) = (a * b) * c$
- Коммутативность:
 - $a * b = b * a$
- Линейность
 - $(a + b) * c = a * c + b * c$
 - $(ka * b) = k(a * b)$

Свертка

- В области обработки изображений используются свертки с явно заданными ядрами
- Что получится, если сделать веса ядер свертки обучаемыми?

Свертка

- В области обработки изображений используются свертки с явно заданными ядрами
- Что получится, если сделать веса ядер свертки обучаемыми?
 - Сверточный слой

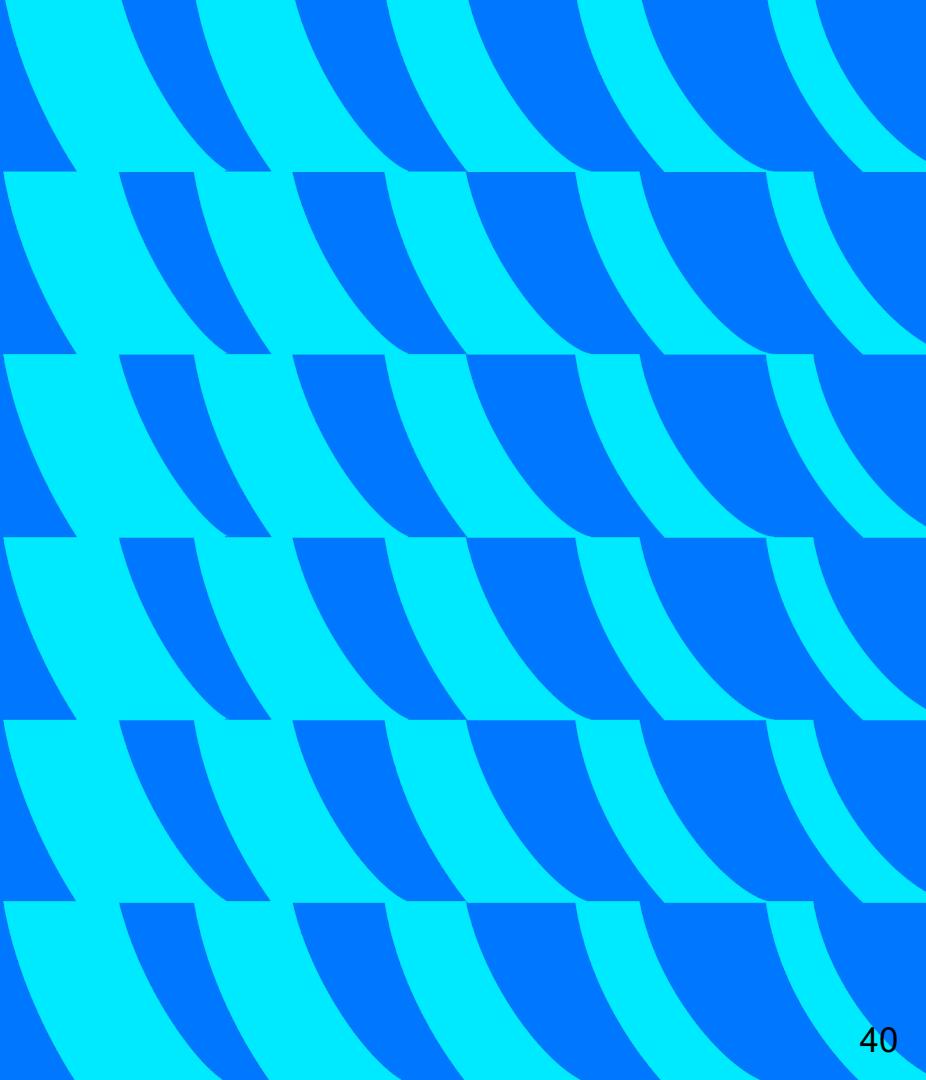


Вопросы



Сверточный слой

.....



Сверточный слой

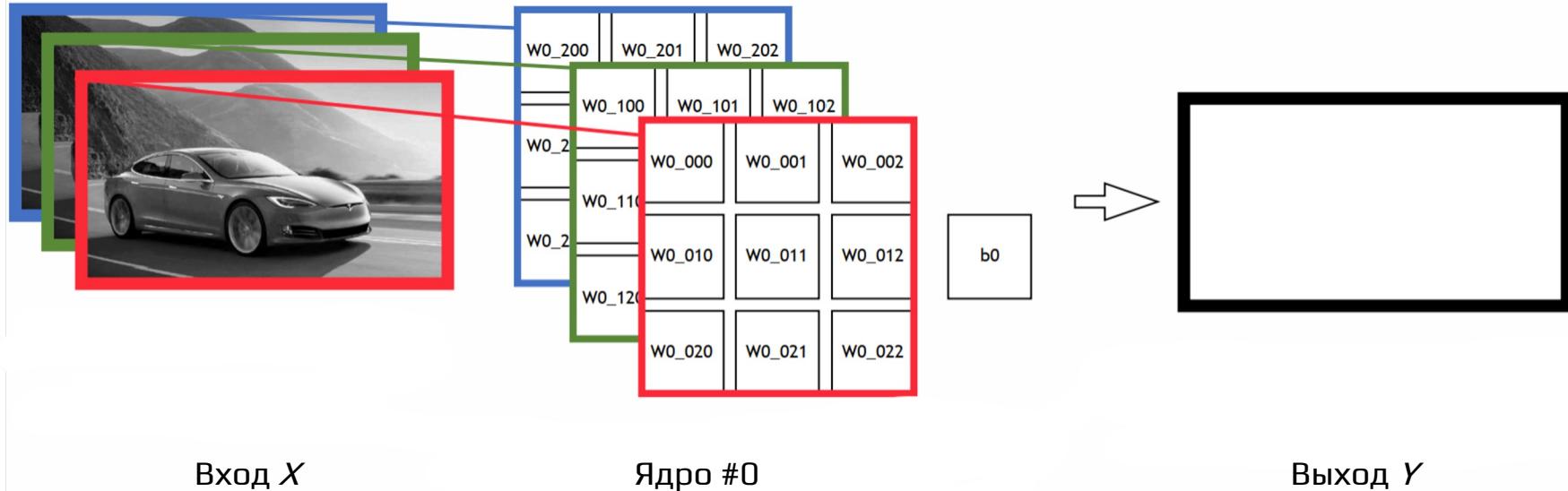
- Сверточный слой состоит из нескольких ($K \geq 1$) ядер свертки
 - К каждому результату свертки можно добавить bias
- На вход подается тензор X размера $H \times W \times D$
- Каждое из K ядер сворачивается с входным тензором
- В итоге получается новый тензор Y размера $H' \times W' \times K$
 - Применив padding, можно сделать $H = H'$, $W = W'$

Сверточный слой с *К*различных ядер

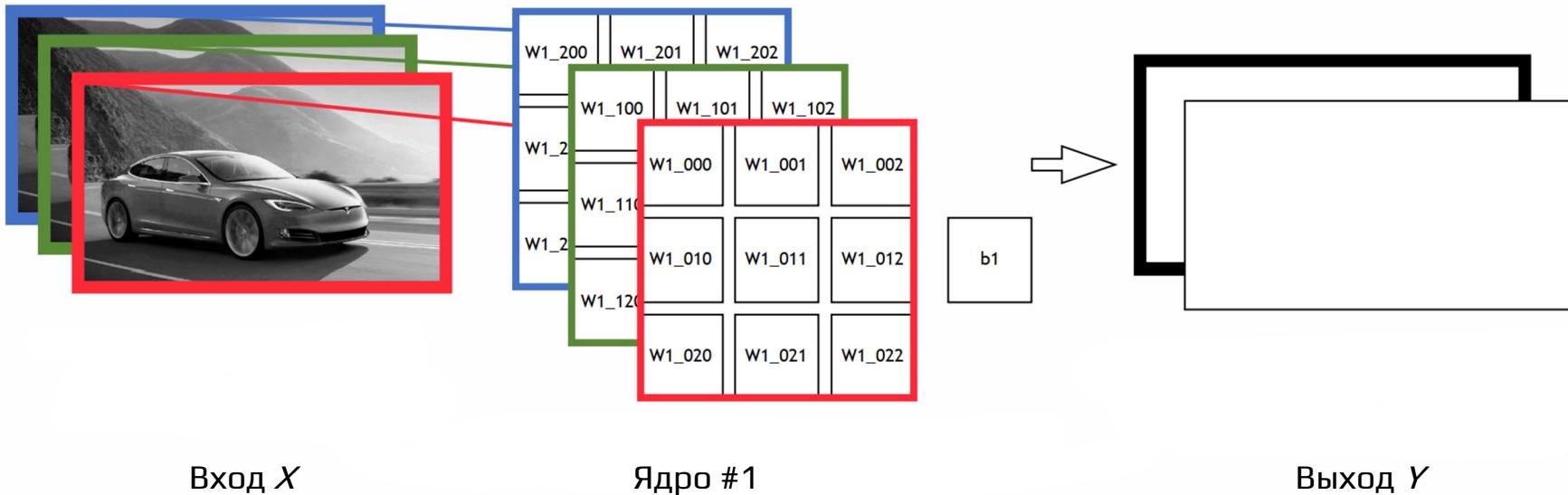


Вход X

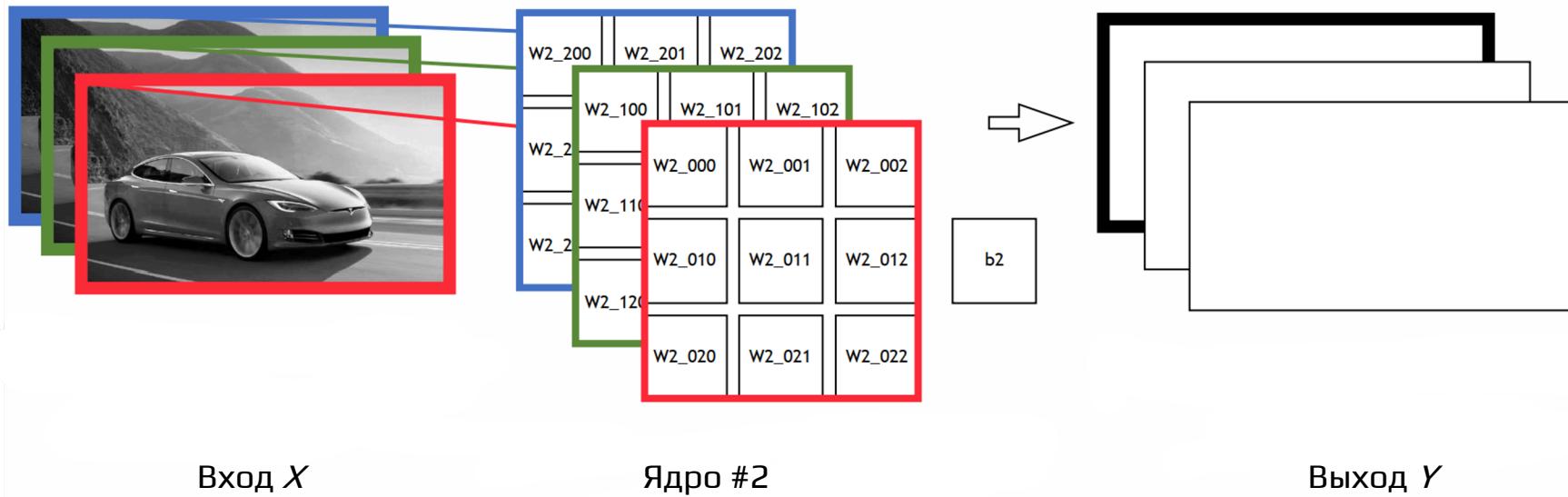
Сверточный слой с Кразличных ядер



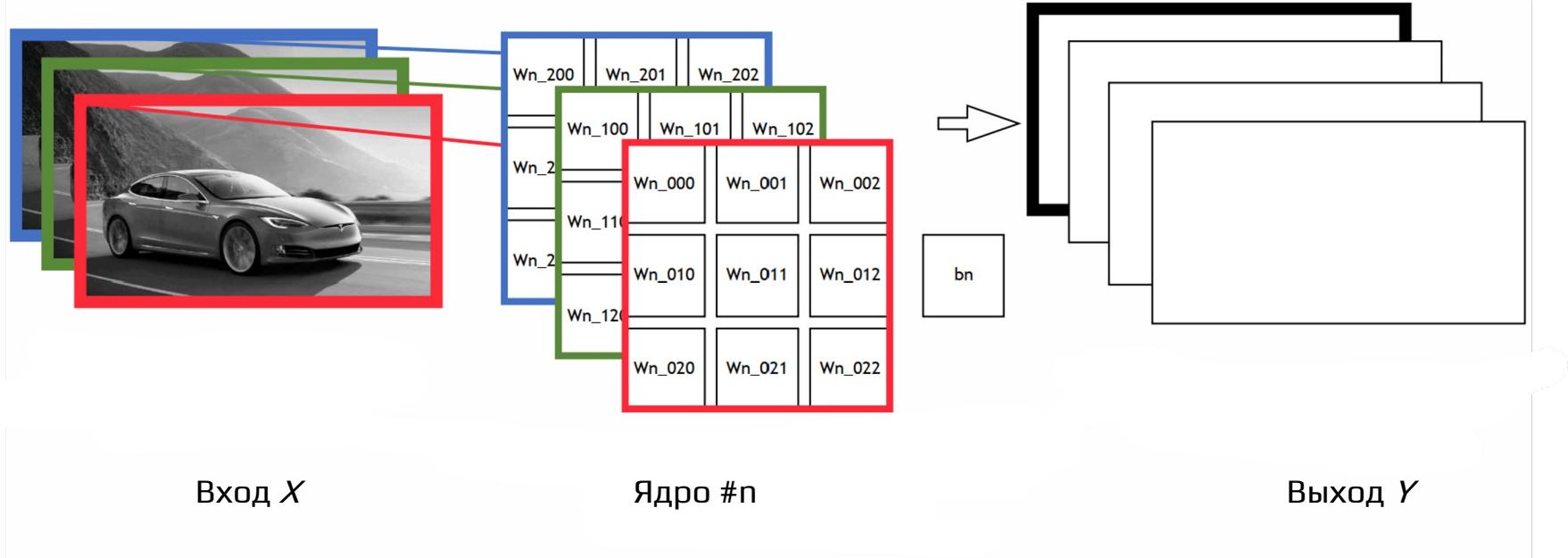
Сверточный слой с Кразличных ядер



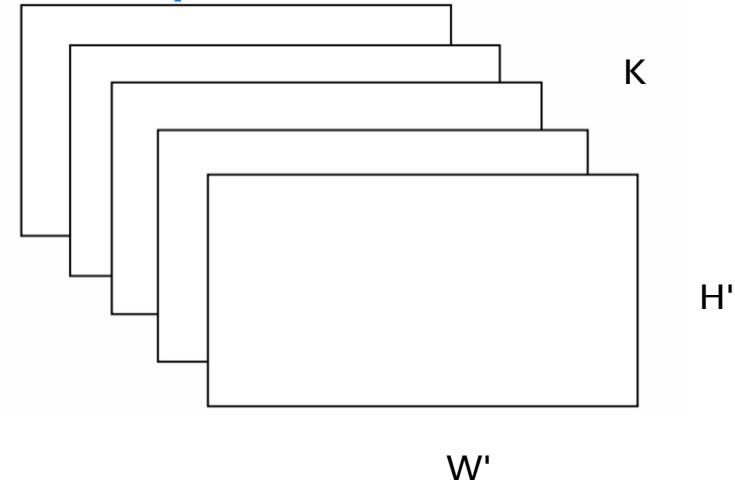
Сверточный слой с Кразличных ядер



Сверточный слой с Кразличных ядер



Сверточный слой с Кразличных ядер



Вход X

Выход Y

Сверточный слой

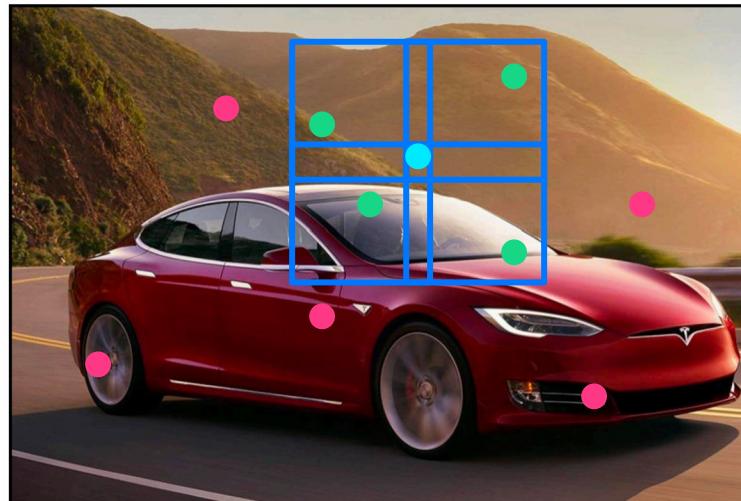
- Сколько всего обучаемых весов у сверточного слоя, если
 - размер входного тензора $H \times W \times D$
 - количество ядер K
 - размер ядра 3×3
 - используется $bias$?

Сверточный слой

- Сколько всего обучаемых весов у сверточного слоя, если
 - размер входного тензора $H \times W \times D$
 - количество ядер K
 - размер ядра 3×3
 - используется $bias$?
- Число весов на одно ядро свертки: $3 \times 3 \times D$ (weights) + 1 (bias)
- Всего ядер $K \Rightarrow$ весов $K \times (3 \times 3 \times D + 1)$

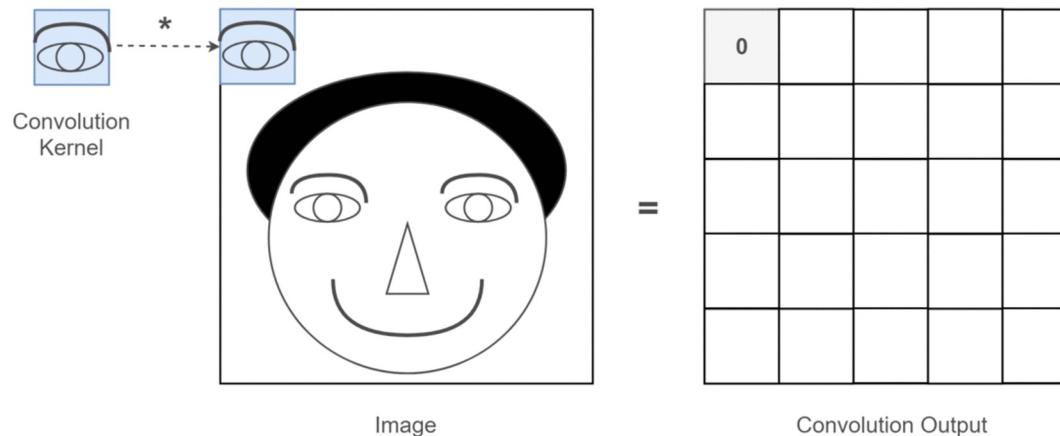
Сверточный слой - отличия от полносвязного

- Сверточный слой обрабатывает входной сигнал *локально*
 - "Взаимодействуют" только те ячейки, которые находятся рядом (попадают в одно окно свертки)



Сверточный слой - отличия от полносвязного

- Сверточный слой позволяет сделать сеть инвариантной к смещениям объектов
 - Ядра сверточ применяются одинаковым образом ко всей "площади" входного тензора



Сверточный слой - параметры

- Число ядер (оно же: "число фильтров", "ширина слоя")
- Размер ядра (3×3 , 5×5 , 3×5 , ...)
- Padding (добавление значений по краям входного сигнала)
- Stride (величина шага скользящего окна, бывает и >1)

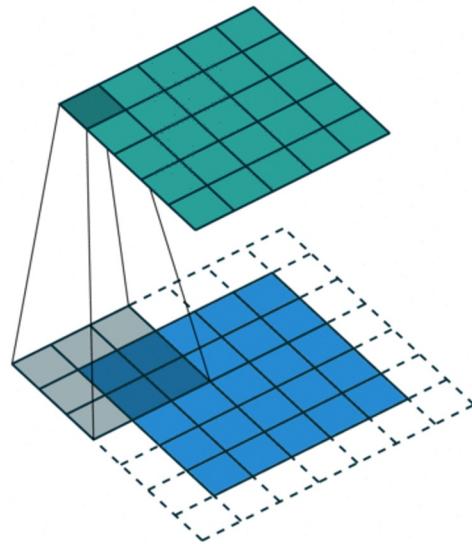
Сверточный слой - параметры

- **Число ядер (оно же: "число фильтров", "ширина слоя")**
- **Размер ядра (3x3, 5x5, 3x5, ...)**
- Padding (добавление значений по краям входного сигнала)
- Stride (величина шага скользящего окна, бывает и >1)

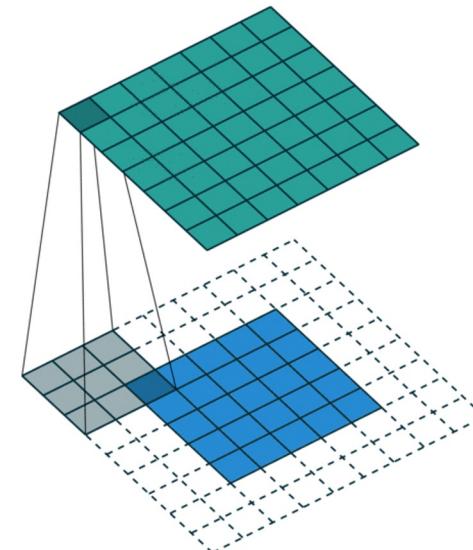
Сверточный слой - параметры

- Число ядер (оно же: "число фильтров", "ширина слоя")
- Размер ядра (3×3 , 5×5 , 3×5 , ...)
- **Padding (добавление значений по краям входного сигнала)**
- Stride (величина шага скользящего окна, бывает и >1)

Сверточный слой - Padding



Размер ядра 3x3
Padding = 1

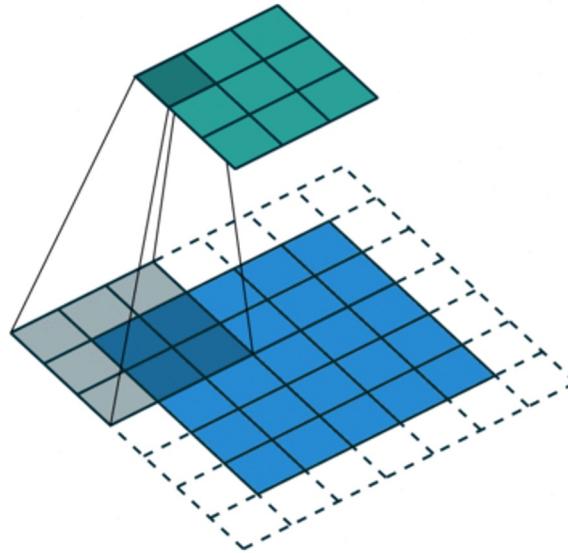


Размер ядра 3x3
Padding = 2

Сверточный слой - параметры

- Число ядер (оно же: "число фильтров", "ширина слоя")
- Размер ядра (3×3 , 5×5 , 3×5 , ...)
- Padding (добавление значений по краям входного сигнала)
- **Stride (величина шага скользящего окна, бывает и >1)**

Сверточный слой - Stride



Размер ядра 3×3
Padding = 1
Stride = 2

Сверточный слой - параметры

- Число ядер (оно же: "число фильтров", "ширина слоя")
- Размер ядра (3×3 , 5×5 , 3×5 , ...)
- Padding (добавление значений по краям входного сигнала)
- Stride (величина шага скользящего окна, бывает и >1)

Сверточный слой - параметры

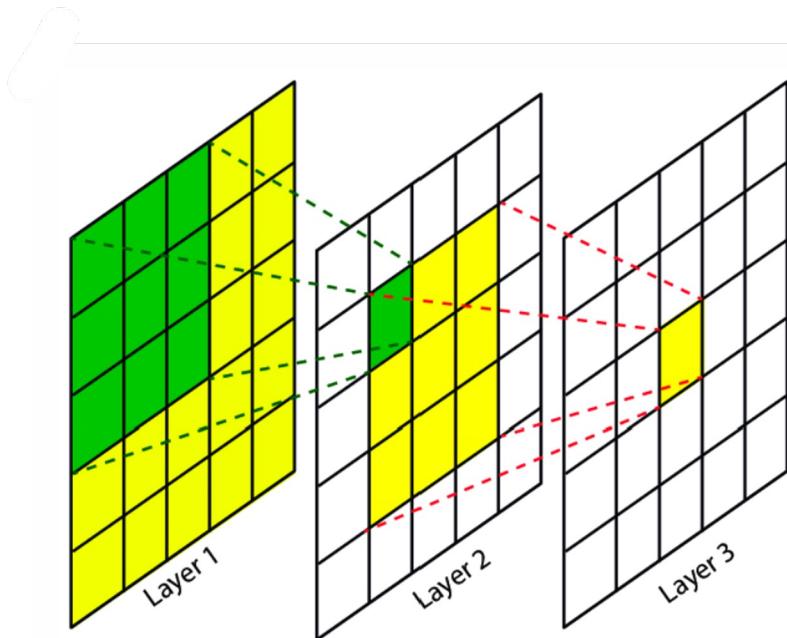
- Число ядер (оно же: "число фильтров", "ширина слоя")
- Размер ядра (3×3 , 5×5 , 3×5 , ...)
- Padding (добавление значений по краям входного сигнала)
- Stride (величина шага скользящего окна, бывает и >1)

Влияют на H/W выходного тензора

Сверточный слой - рецептивное поле

Рецептивное поле (receptive field)

нейрона - размер области исходного сигнала (изображения), которая может вносить вклад в активацию данного нейрона



Рецептивное поле нейрона на Layer 2: 3×3

Рецептивное поле нейрона на Layer 3: 5×5

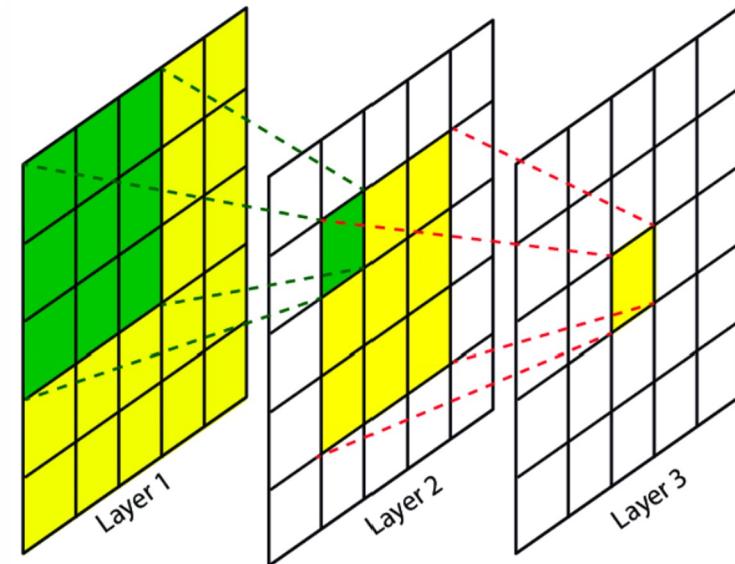
Сверточный слой - рецептивное поле

Рецептивное поле (receptive field)

нейрона - размер области исходного сигнала (изображения), которая может вносить вклад в активацию данного нейрона

Как меняется РП выхода слоя при:

- Увеличении ядер сверток?



Рецептивное поле нейрона на Layer 2: 3×3

Рецептивное поле нейрона на Layer 3: 5×5

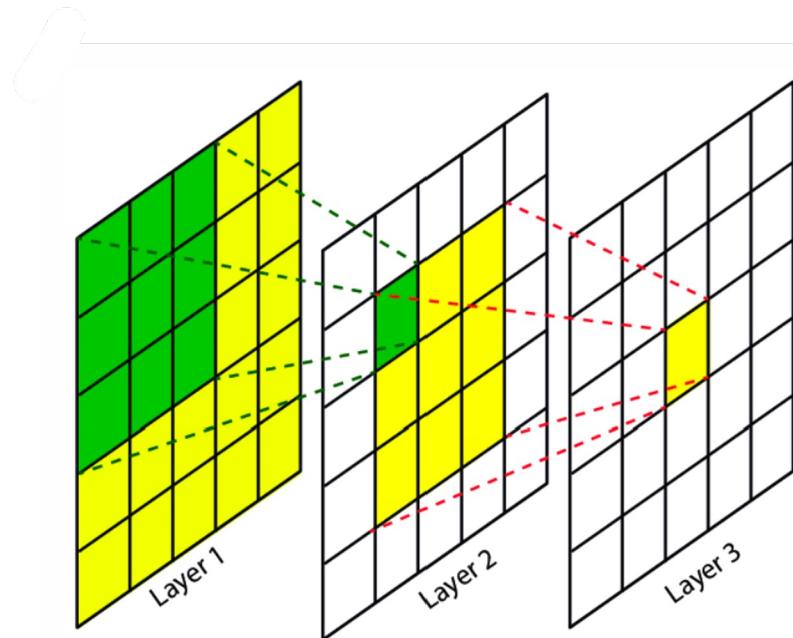
Сверточный слой - рецептивное поле

Рецептивное поле (receptive field)

нейрона - размер области исходного сигнала (изображения), которая может вносить вклад в активацию данного нейрона

Как меняется РП выхода слоя при:

- Увеличении ядер сверток? Растет



Рецептивное поле нейрона на Layer 2: 3x3

Рецептивное поле нейрона на Layer 3: 5x5

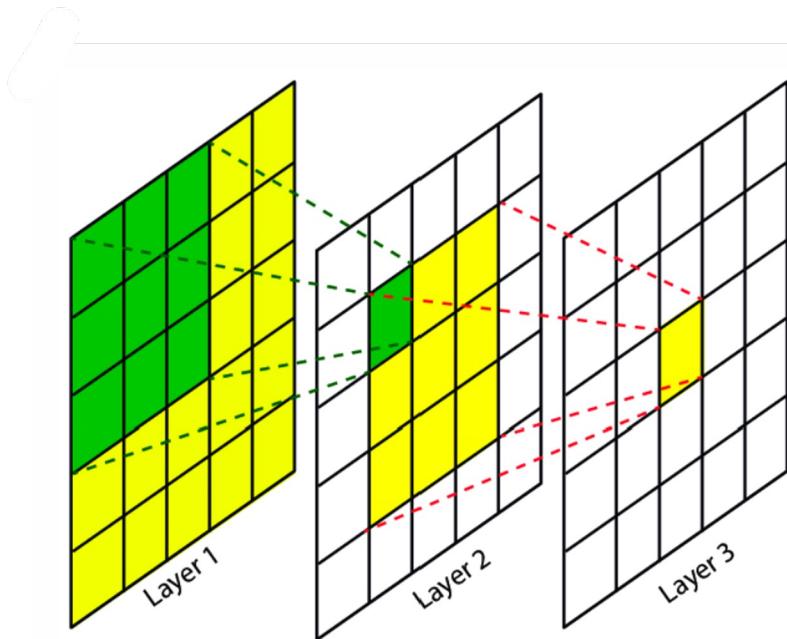
Сверточный слой - рецептивное поле

Рецептивное поле (receptive field)

нейрона - размер области исходного сигнала (изображения), которая может вносить вклад в активацию данного нейрона

Как меняется РП выхода слоя при:

- Увеличении ядер сверток? Растет
- Добавлении padding?



Рецептивное поле нейрона на Layer 2: 3x3

Рецептивное поле нейрона на Layer 3: 5x5

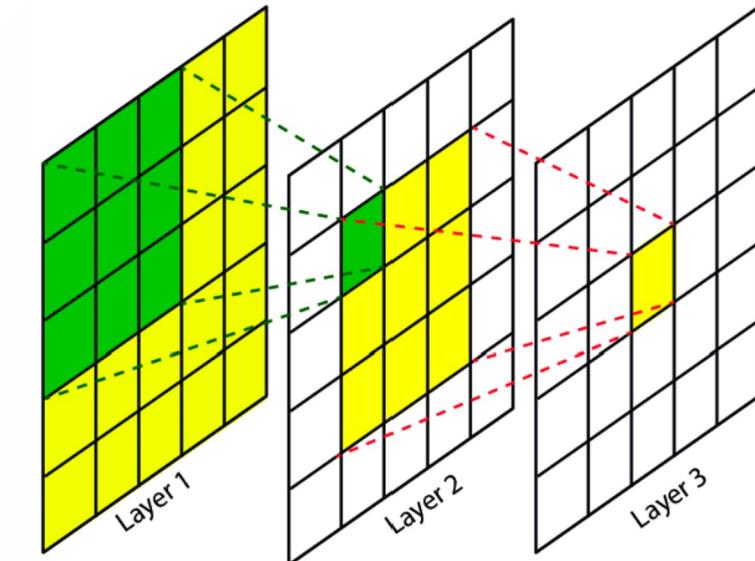
Сверточный слой - рецептивное поле

Рецептивное поле (receptive field)

нейрона - размер области исходного сигнала (изображения), которая может вносить вклад в активацию данного нейрона

Как меняется РП выхода слоя при:

- Увеличении ядер сверток? Растет
- Добавлении padding? Никак



Рецептивное поле нейрона на Layer 2: 3x3

Рецептивное поле нейрона на Layer 3: 5x5

Сверточный слой - параметры

- Число ядер (оно же: "число фильтров", "ширина слоя")
- Размер ядра (3×3 , 5×5 , 3×5 , ...)
- Padding (добавление значений по краям входного сигнала)
- Stride (величина шага скользящего окна, бывает и >1)

Определяют receptive field ячеек
выходного тензора (нейронов)

Сверточный слой

```
import torch
```

```
conv_layer = torch.nn.Conv2d()
```

Init signature:
torch.nn.Conv2d(
 in_channels,
 out_channels,
 kernel_size,
 stride=1,
 padding=0,
 dilation=1,
 groups=1,
 bias=True,
 padding_mode='zeros',
)

Сверточный слой - backprop

- Градиент по весам сверточного слоя = свертка входного сигнала с градиентом по выходу слоя (в одномерном случае)
- Еще можно посмотреть, например, [тут](#)

Вопросы



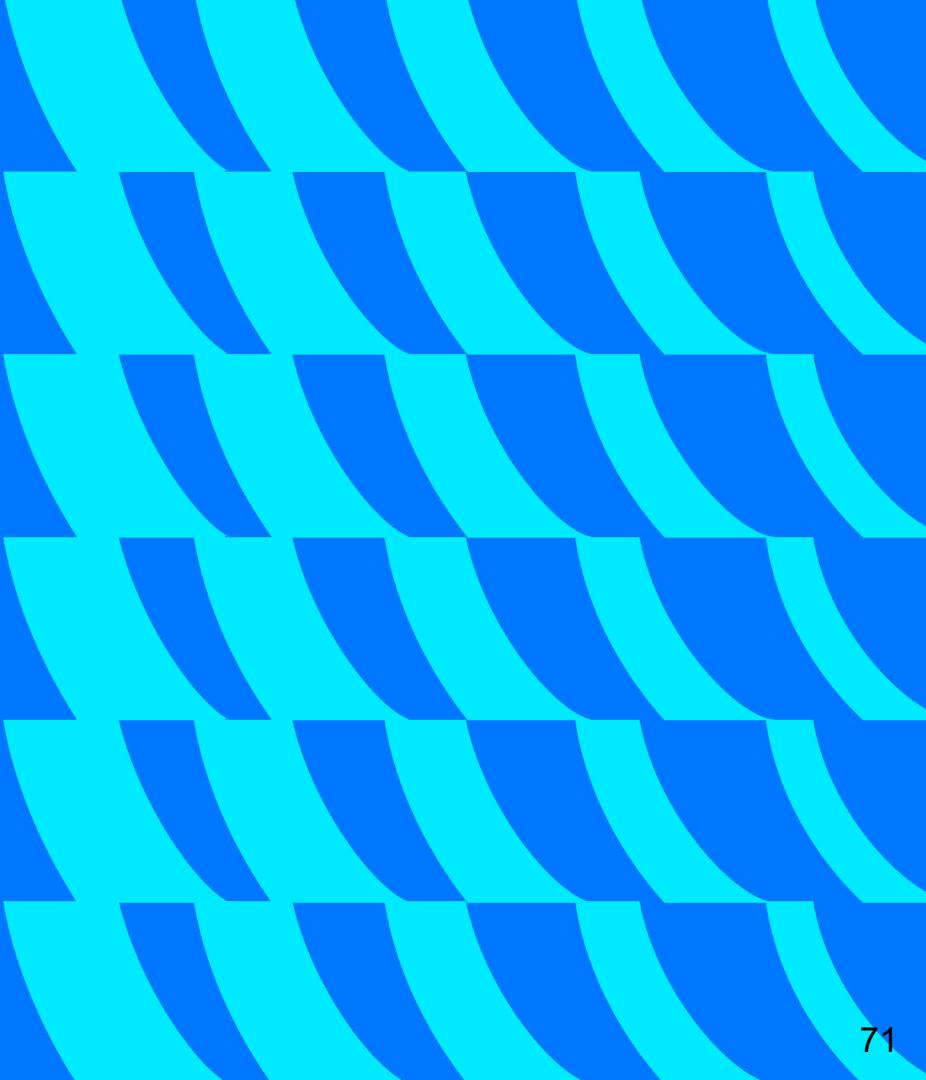
CNN

- Чтобы получить сверточную нейросеть, помимо сверточного слоя понадобятся:
 - Слои активации (Tanh, ReLU, ...)
 - Слои пулинга
 - (опционально) BatchNorm
 - (опционально) Полносвязные выходные слои

CNN

- Чтобы получить сверточную нейросеть, помимо сверточного слоя понадобятся:
 - Слои активации (Tanh, ReLU, ...)
 - **Слои пулинга**
 - (опционально) BatchNorm
 - (опционально) Полносвязные выходные слои

Пулинг



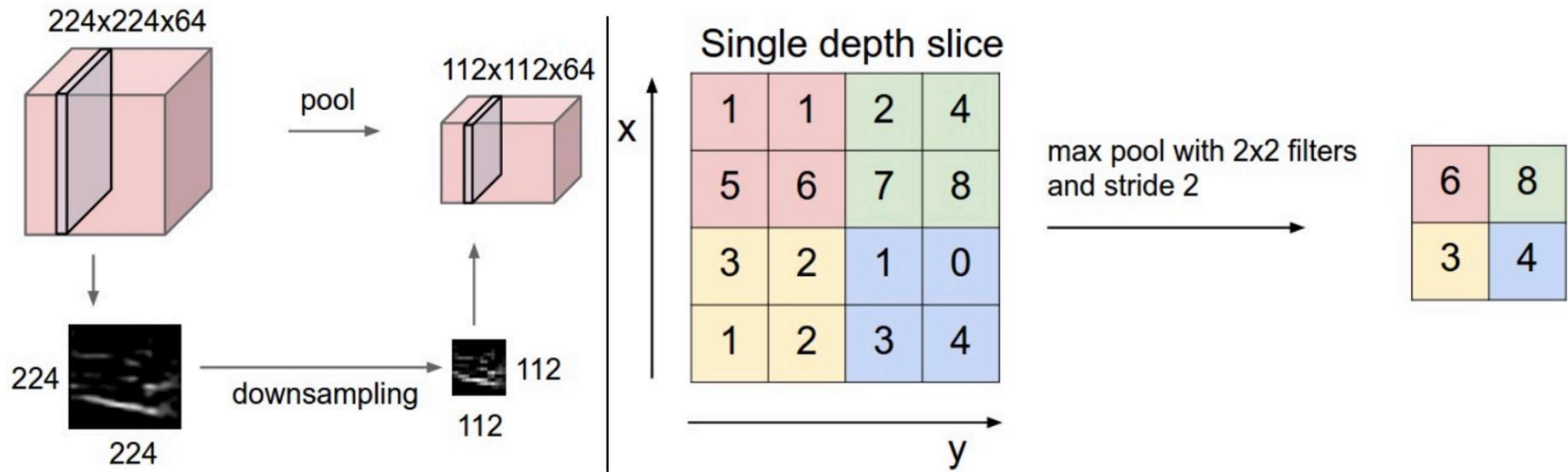
Проблема памяти

- Обычно число каналов сверточных слоев растет по мере увеличения глубины сети
 - Например, в сети ResNet18 количество сверток в одном слое растет так: 64 - 128 - 256 - 512
- Чем больше размер тензоров, тем больше потребление памяти
- Слой Pooling позволяет уменьшать H/W тензоров

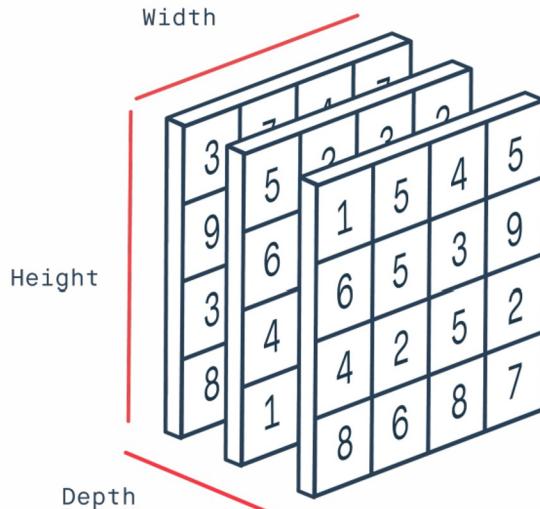
Pooling

- Разобьем тензор на части по ширине и высоте
- В каждой части независимо от других посчитаем 1 статистику (среднее - AveragePooling, максимум - MaxPooling)
- Склейм полученные статистики обратно в тензор
- Получится тензор прежней глубины, но меньшего размера по ширине и высоте

Pooling

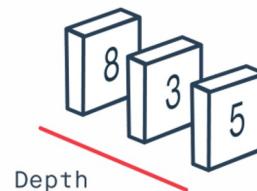


Global Pooling



Height x Width x Depth

- Можно посчитать статистики сразу по всему каналу, а не по регионам
- В результате получится тензор размера $1 \times 1 \times D$, который можно отправить в полносвязный слой
- Это будет **Global Pooling**



$1 \times 1 \times \text{Depth}$

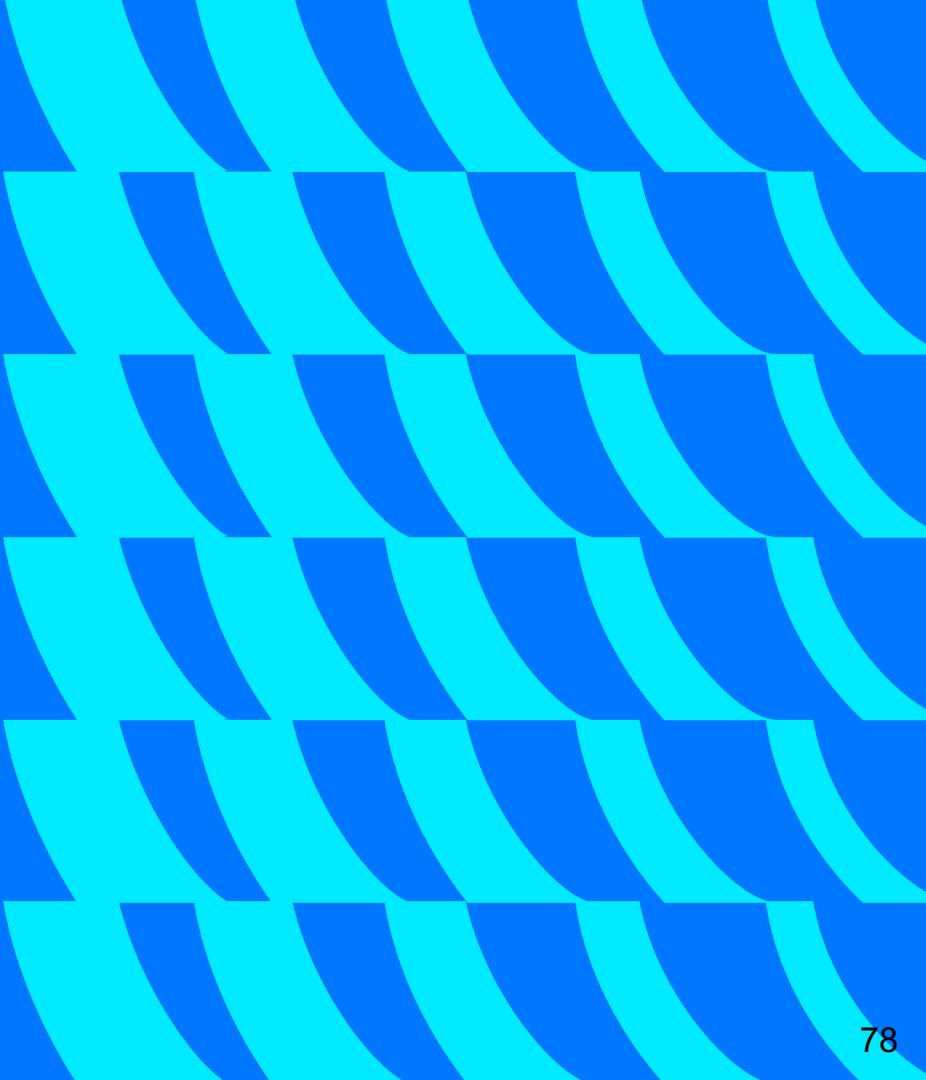
Сверточный слой

- Чтобы получить сверточную нейросеть, помимо сверточного слоя понадобятся:
 - Слои активации (Tanh, ReLU, ...)
 - Слои пулинга
 - (опционально) Полносвязные выходные слои

Вопросы



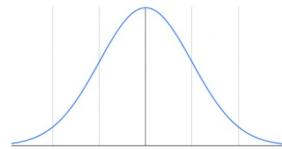
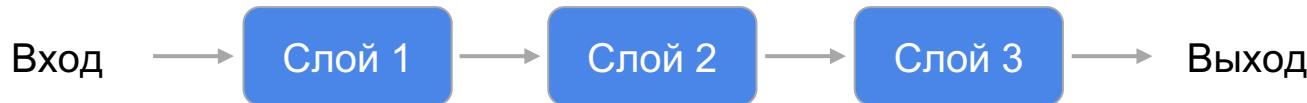
Batchnorm



CNN

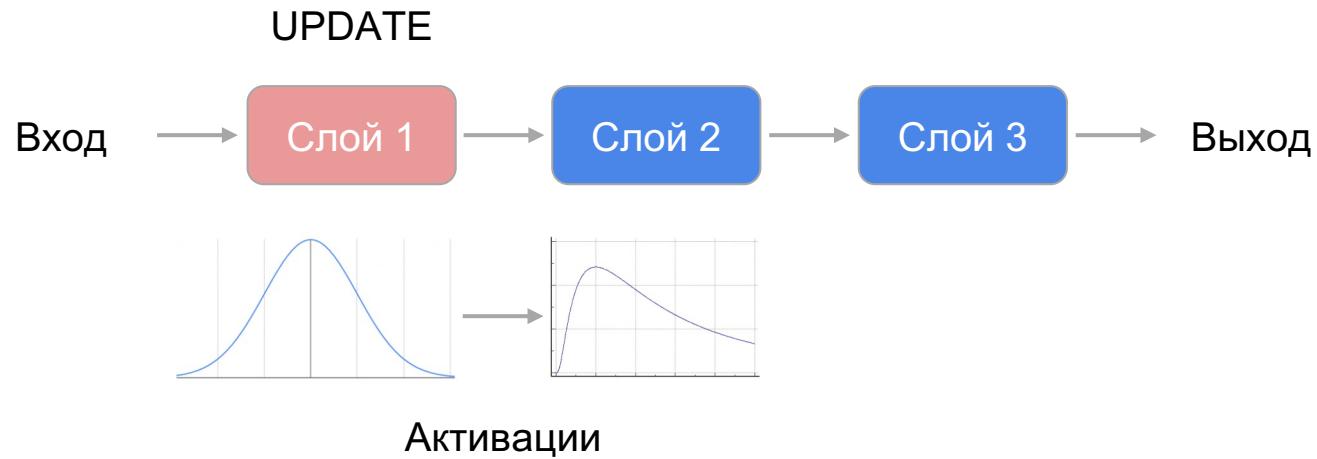
- Чтобы получить сверточную нейросеть, помимо сверточного слоя понадобятся:
 - Слои активации (Tanh, ReLU, ...)
 - Слои пулинга
 - **(опционально) BatchNorm**
 - **(опционально) Полносвязные выходные слои**

Проблема

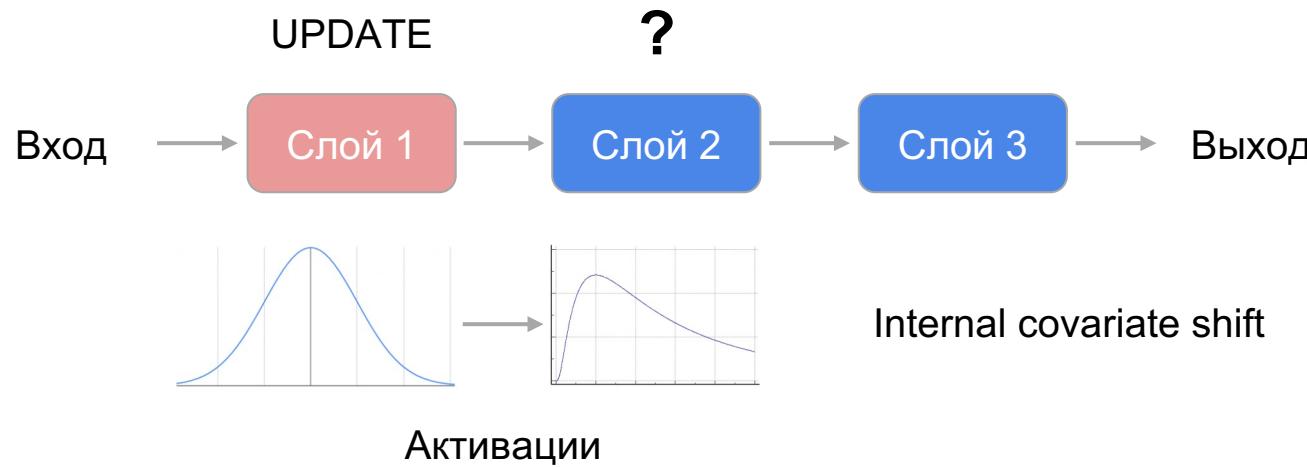


Активации

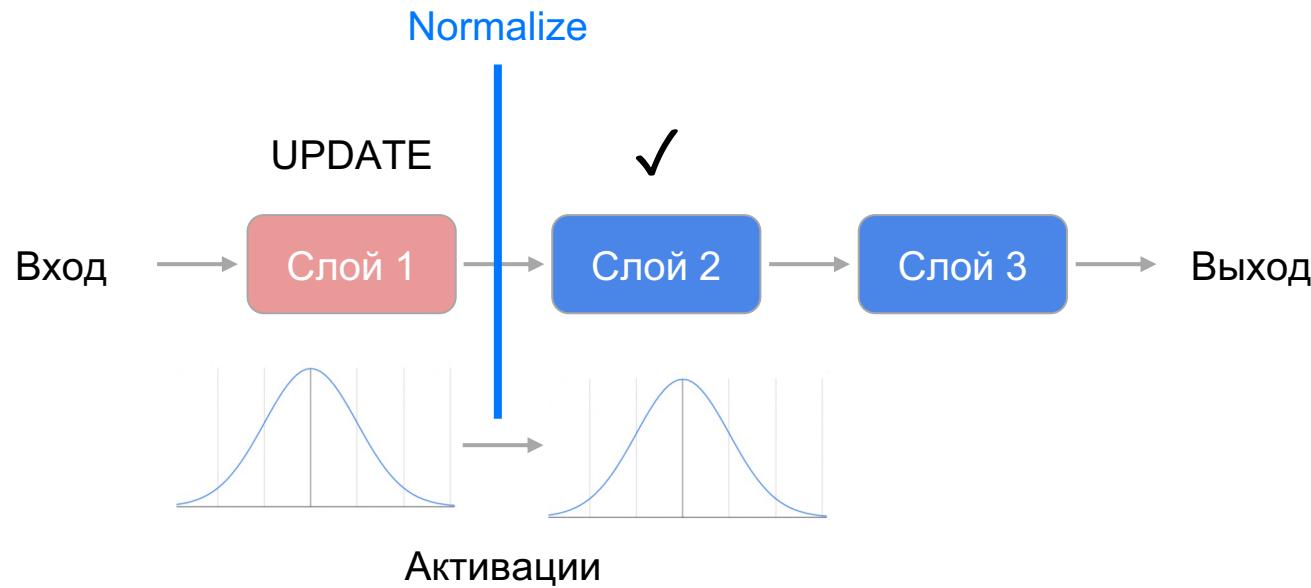
Проблема



Проблема



Нормировка



Batch normalization

Во время тренировки:

- Вычесть среднее по батчу
- Разделить на STD батча
- Умножить на обучаемый scale
- Добавить обучаемый bias

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\dots m}\}$;

Parameters to be learned: γ, β

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

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

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

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

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

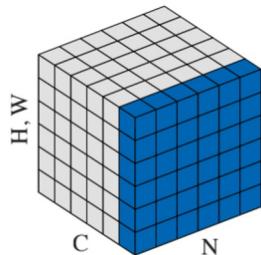
Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

Batch normalization

Во время тренировки:

- Вычесть среднее по батчу
- Разделить на STD батча
- Умножить на обучаемый scale
- Добавить обучаемый bias

В свёрточных сетях:



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

Parameters to be learned: γ, β

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

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

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

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

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

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

Batch normalization

Во время тренировки:

- Вычесть среднее по батчу
- Разделить на STD батча
- Умножить на обучаемый scale
- Добавить обучаемый bias

Следствия:

- Поведение зависит от данных батча
- Поведение зависит от размера батча

Что делать в inference если нет батча?

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\dots m}\}$;

Parameters to be learned: γ, β

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

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

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

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

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

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

Batch normalization

Во время тренировки:

- Вычесть среднее по батчу
- Разделить на STD батча
- Умножить на обучаемый scale
- Добавить обучаемый bias

При тестировании используются средние и STD усредненные по traininig set

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\dots m}\}$;

Parameters to be learned: γ, β

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

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

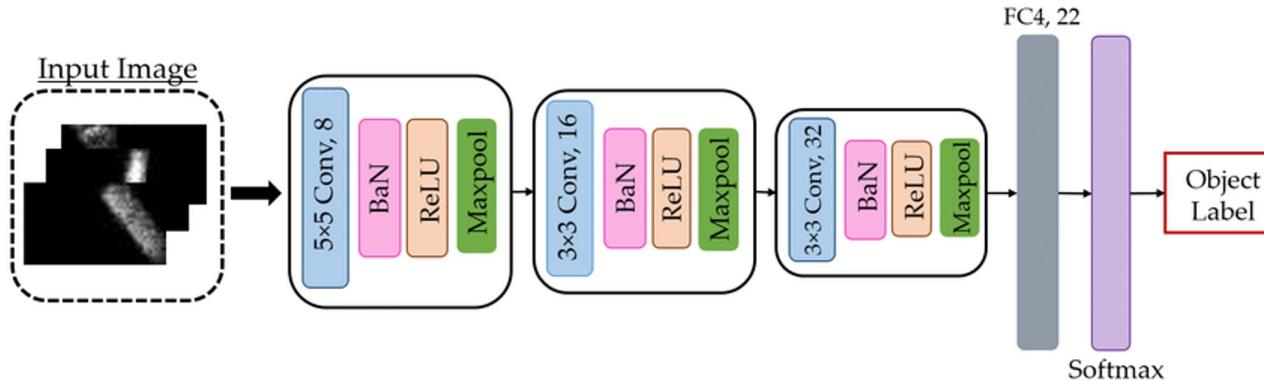
$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \quad // \text{mini-batch variance}$$

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

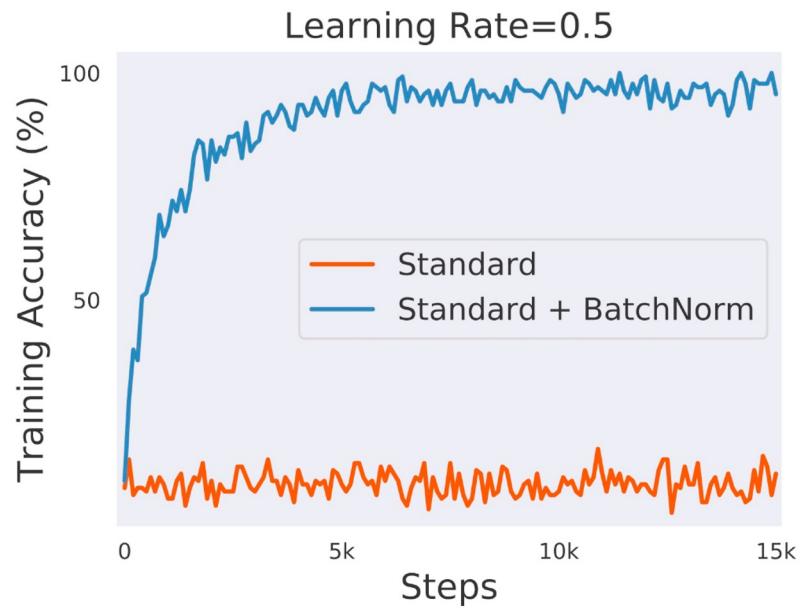
$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) \quad // \text{scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

Пример



Пример

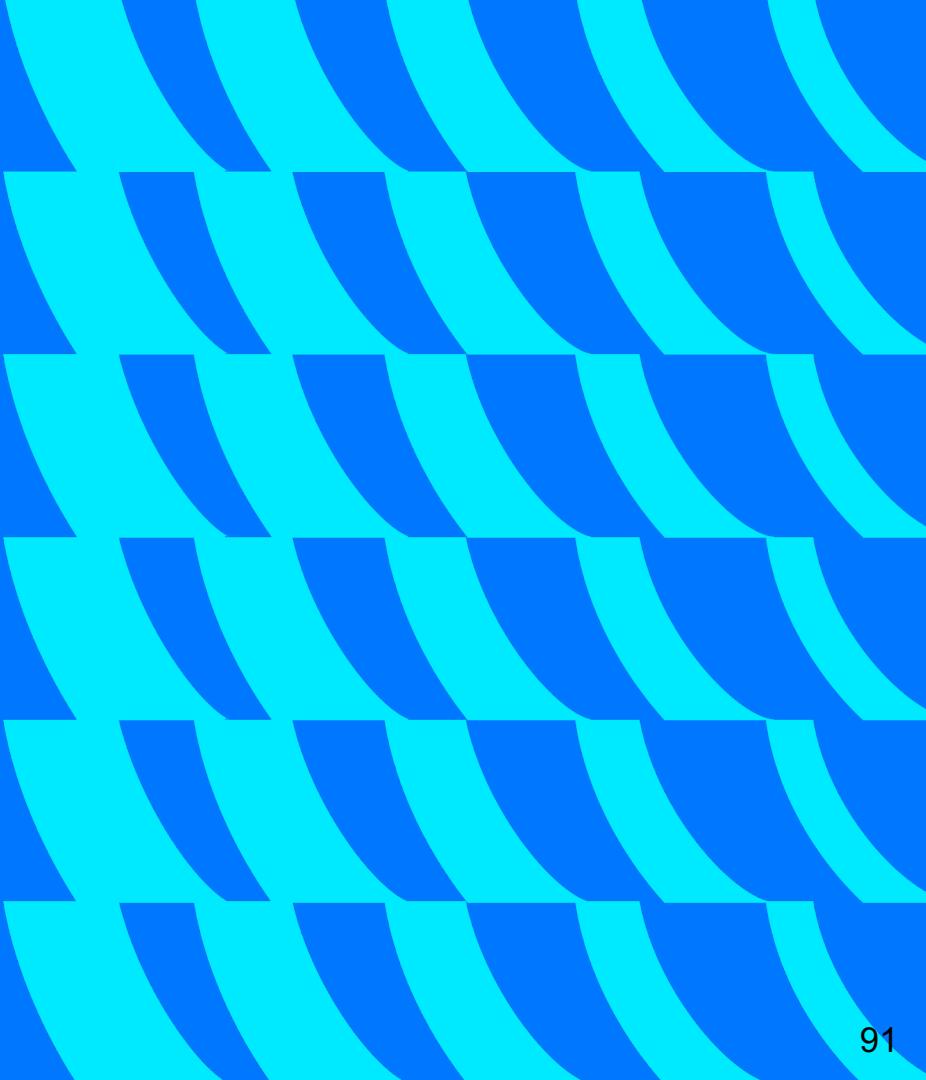


Batchnorm summary

- BN ускоряет обучение CNN и FC сетей
- BN позволяет использовать больший Learning Rate
- BN вносит разницу между train и test
- При больших batch size разница меньше, а статистики устойчивые

CNN

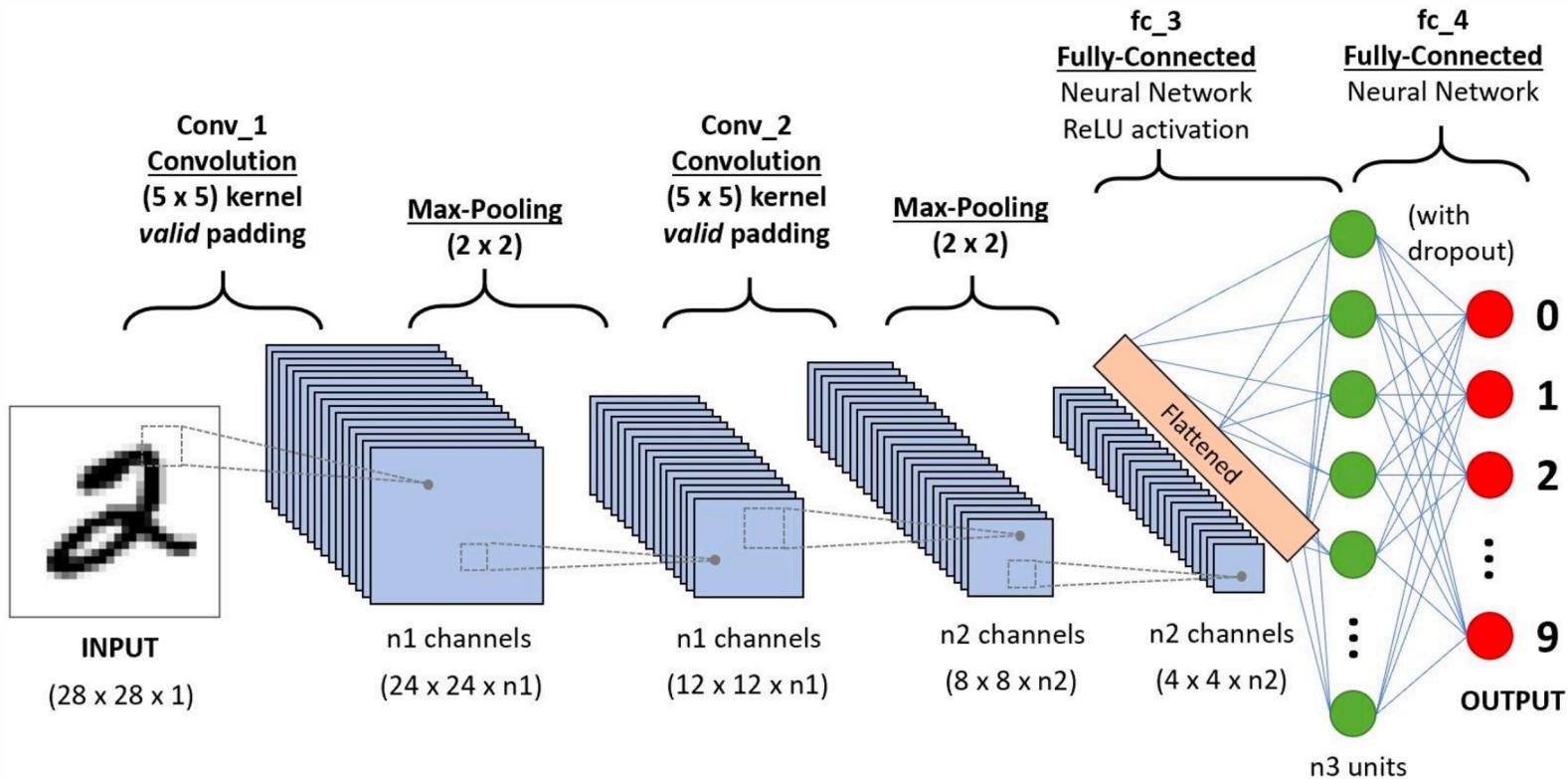
.....



CNN

- CNN = convolutional neural network
- Типичная структура CNN:
((сверточный слой \Rightarrow активация) * k \Rightarrow пулинг) * m \Rightarrow линейный слой
- В современных архитектурах как правило используются свертки 3x3

CNN - LeNet5



CNN

- Первый сверточный слой видит "сырые" данные
 - Его веса худо-бедно "понятны"
- Чем глубже в сеть, тем более абстрактным становится представление исходного объекта

Визуализация ядер первого сверточного слоя



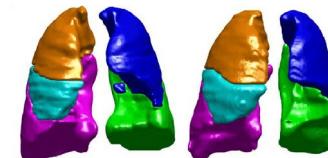
<https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>

CNN

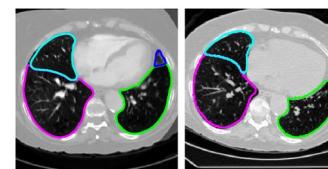
- Сверточные сети хорошо работают на данных с "пространственными размерностями"
 - Звук (1d)
 - Изображения (2d)
 - Сканы КТ (3d)
 - ...



Cat



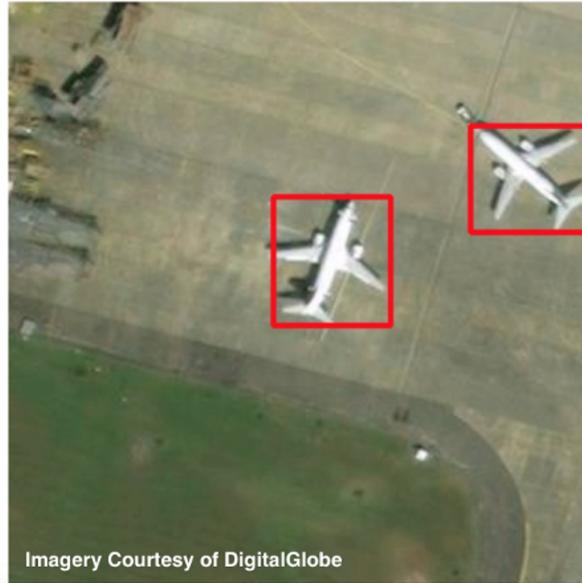
Dog



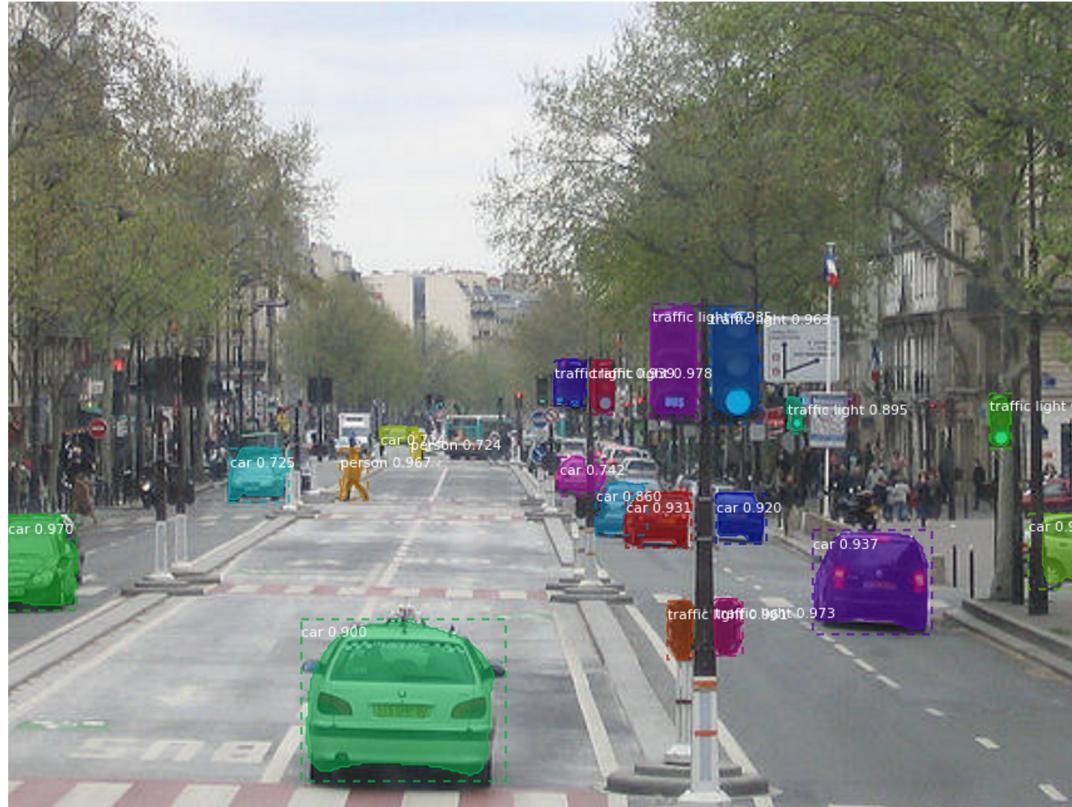
CNN

- Применительно к изображениям, CNN хорошо работают в задачах:
 - Классификации / Регрессии
 - Детектирования
 - Сегментации
 - Генерации

CNN - Object Detection



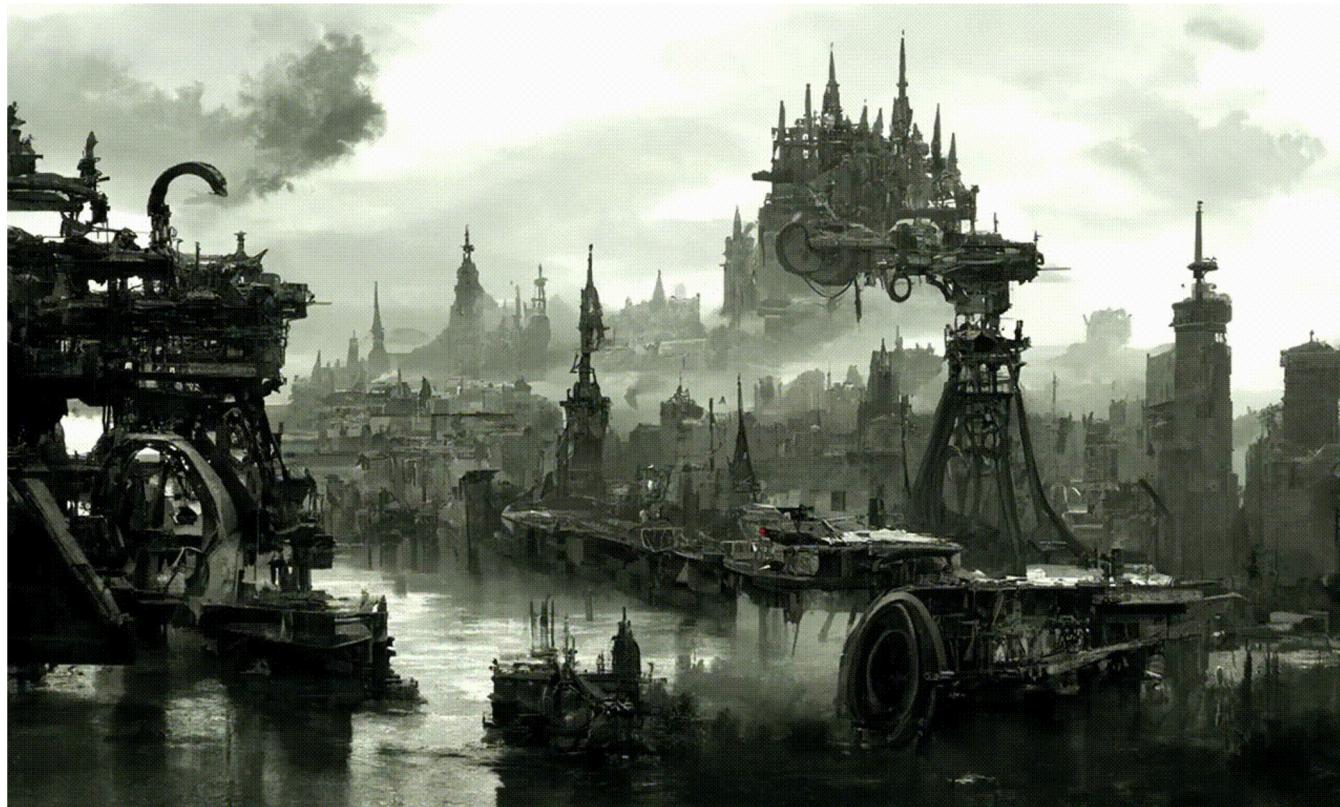
CNN - Instance Segmentation



CNN - Generative models



CNN - Generative models



Курс по компьютерному зрению

- Эти и другие темы обсудим в рамках курса Computer Vision
- Записывайтесь!

Резюме по CNN

.....

Итоги

- Сверточный слой
 - Основан на операции свертки
 - Сам "извлекает признаки"
 - Хорошо работает для данных с локальной связностью
 - Эффективнее полносвязного слоя по числу весов
- Сверточная нейросеть
 - Базовые слои - сверточный, активация, пулинг, полносвязный(е) в конце
 - Чем глубже слой в сети, тем с более абстрактными признаками он работает

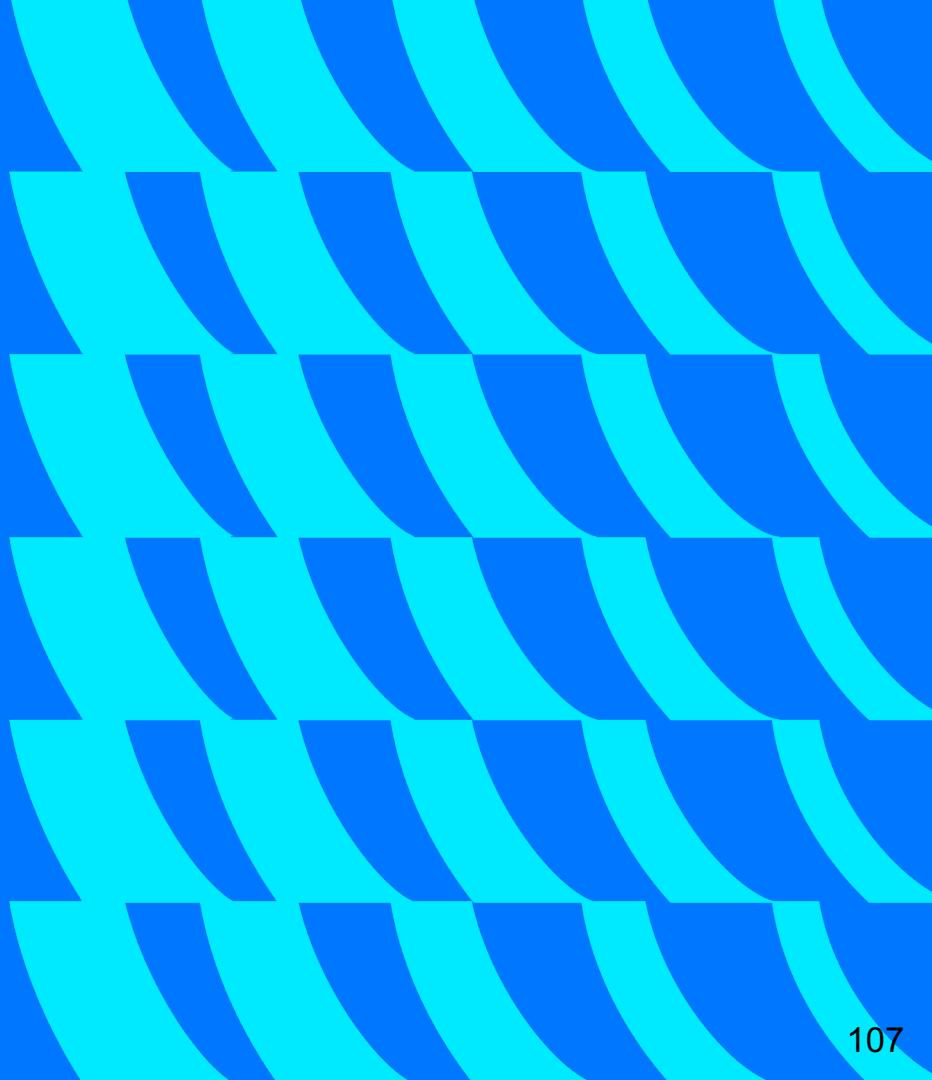
Почитать еще

- <https://cs231n.github.io/convolutional-networks/>

Контрольные вопросы

- Пусть имеется БД с записями о пациентах, содержащая признаки вида "возраст", "пол", "вес" и т.д, а также поставленный диагноз. Почему использовать сверточную нейросеть для предсказания диагноза - плохая идея?
- Сколько операций (FLOPS) потребуют вычисления в сверточном слое с 32 свертками размера 3x3, padding = 1, stride = 1, на входе цветное (RGB) изображение размера 64x64?
- Каким будет размер тензора на выходе слоя MaxPooling(2, 2), если на вход пришел тензор с $H \times W \times D = 32 \times 16 \times 256$?
- Составьте сверточный слой, который обратит порядок каналов входного изображения с 3 каналами (напр. RGB → BGR)

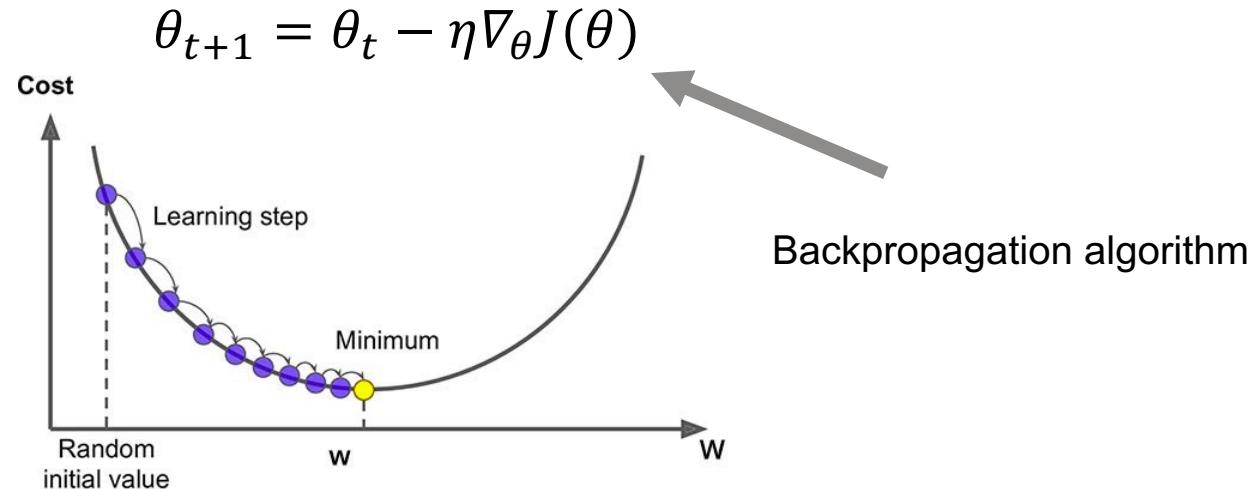
Оптимизация (recap)



Обычный gradient descent

Функция потерь: $J(\theta) = \sum_{i=1}^N l(f_\theta(x_i), y_i)$, где N — все доступные тренировочные данные

η — шаг алгоритма (learning rate)



Обычный gradient descent

Функция потерь: $J(\theta) = \sum_{i=1}^N l(f_\theta(x_i), y_i)$, где N — все доступные тренировочные данные

η — шаг алгоритма (learning rate)

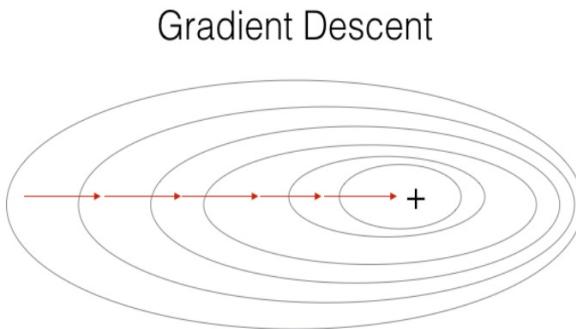
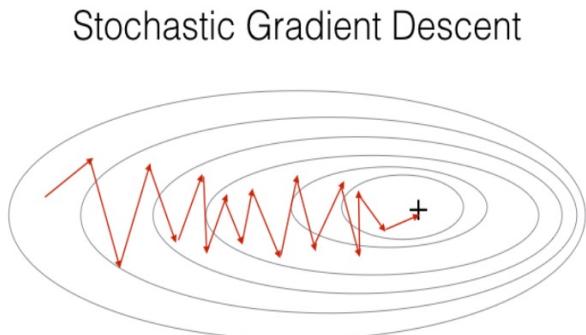
$$\theta_{t+1} = \theta_t - \eta \nabla_{\theta} J(\theta)$$

Что делать, если N очень большой?

Stochastic gradient descent

Обновляем параметры для одного элемента данных (x_i, y_i) $J(\theta) = l(f_\theta(x_i), y_i)$

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta J(\theta) \quad \mathbb{E}_i(-\nabla_\theta l(f_\theta(x_i), y_i)) = \frac{1}{N} \sum_{i=1}^N \nabla_\theta l(f_\theta(x_i), y_i) = -\nabla_\theta J(\theta)$$

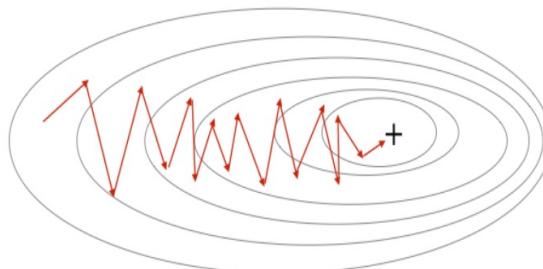


Mini-batch gradient descent

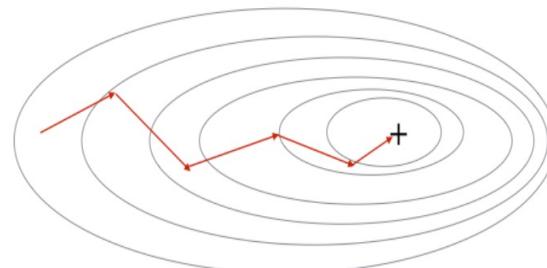
На практике используем подмножество входных данных на каждом шаге - минибатч

$$J(\theta) = \sum_{i=1}^B l(f_\theta(x_i), y_i)$$

Stochastic Gradient Descent



Mini-Batch Gradient Descent



Mini-batch gradient descent

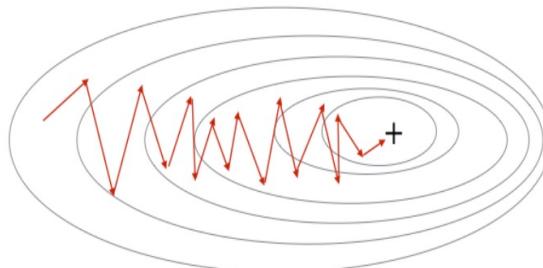
Вычислительно эффективнее

Размер батча B обычно берут 64, 256, 512, 1024

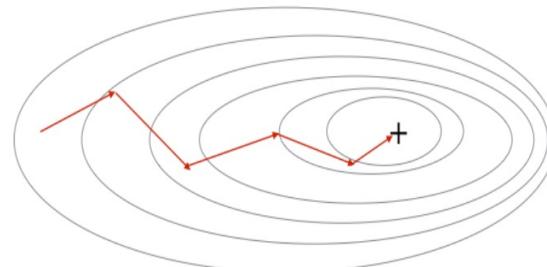
В дальнейшем рассматриваем именно такой вариант функции потерь

Часто под SGD имеют ввиду этот метод

Stochastic Gradient Descent



Mini-Batch Gradient Descent

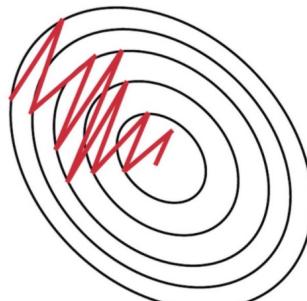


Momentum

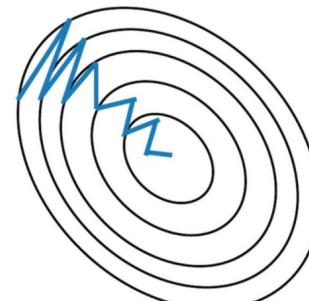
Рядом с оптимумом часто возникают области, где SGD осциллирует

Часто используют $v_t = \gamma v_{t-1} + \eta \nabla_{\theta} J(\theta)$

$$\theta_{t+1} = \theta_t - v_t$$



Stochastic Gradient
Descent **without**
Momentum



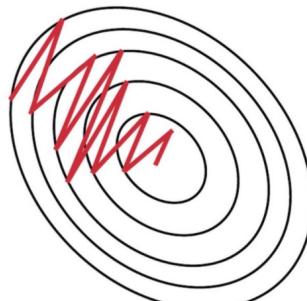
Stochastic Gradient
Descent **with**
Momentum

Momentum

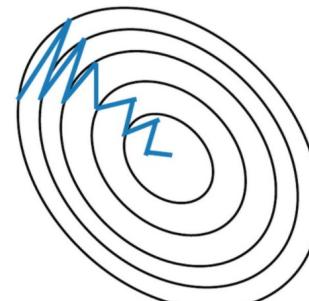
Рядом с оптимумом часто возникают области, где SGD осциллирует

Часто используют $v_t = \gamma v_{t-1} + \eta \nabla_{\theta} J(\theta)$

$$\theta_{t+1} = \theta_t - v_t$$

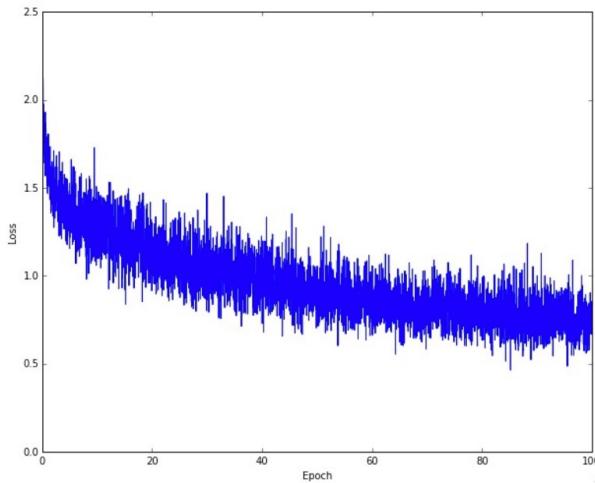
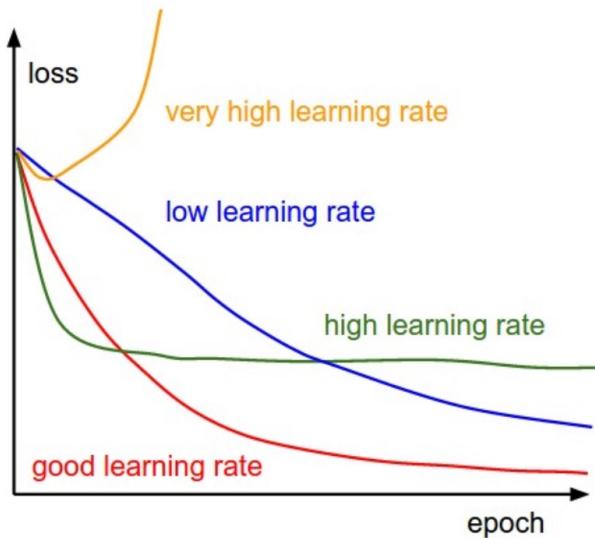


Stochastic Gradient
Descent **without**
Momentum



Stochastic Gradient
Descent **with**
Momentum

Влияние learning rate



Adam

Автоматически подбирает learning rate для каждого параметра

$$g_t = \nabla_{\theta} J(\theta)$$

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \quad \text{первый момент градиента}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \quad \text{второй момент градиента}$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

поправки моментов

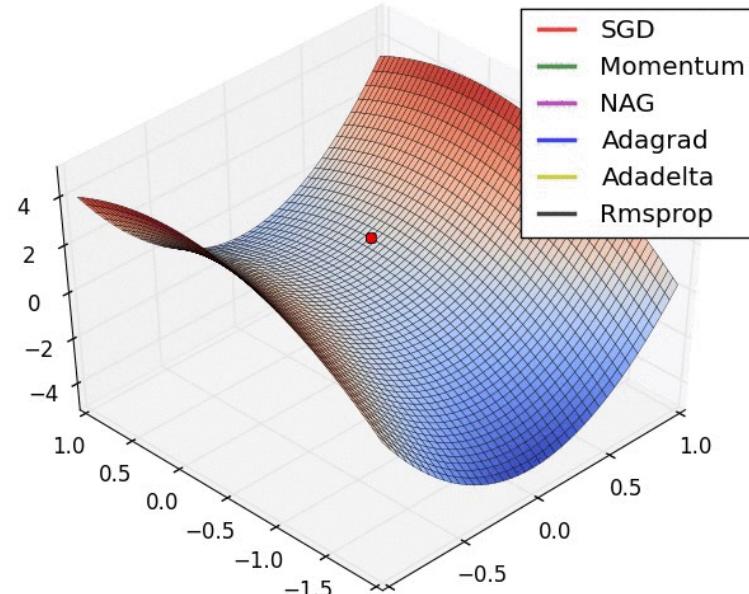
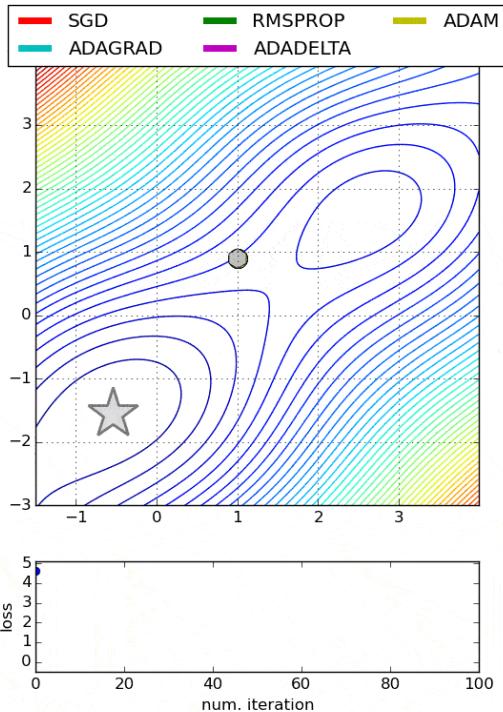
$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$$

$$\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$$

Наиболее успешный (в среднем) на практике метод

Сравнение методов



Всё уже реализовано в PyTorch

```
CLASS torch.optim.SGD(params, lr=<required parameter>, momentum=0, dampening=0,  
    weight_decay=0, nesterov=False, *, maximize=False, foreach=None,  
    differentiable=False) [SOURCE]
```



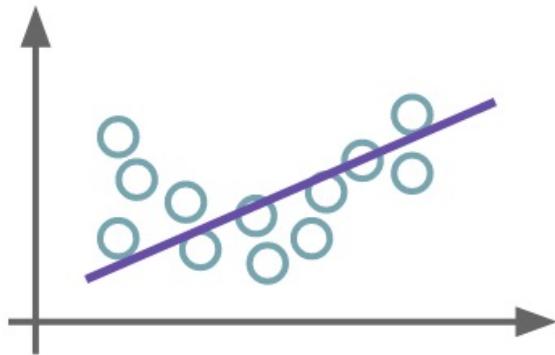
```
CLASS torch.optim.Adam(params, lr=0.001, betas=(0.9, 0.999), eps=1e-08,  
    weight_decay=0, amsgrad=False, *, foreach=None, maximize=False,  
    capturable=False, differentiable=False, fused=False) [SOURCE]
```



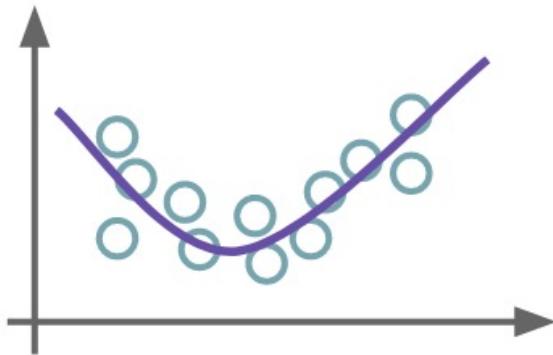
Регуляризация (регар)



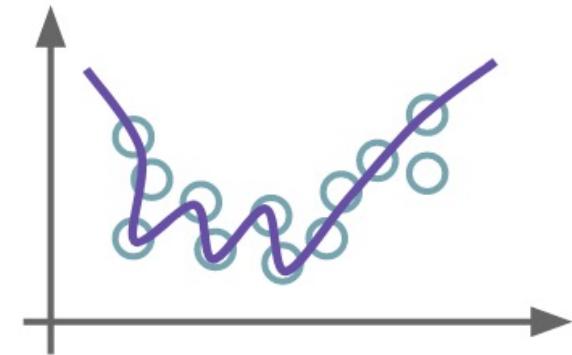
Переобучение



Underfitting

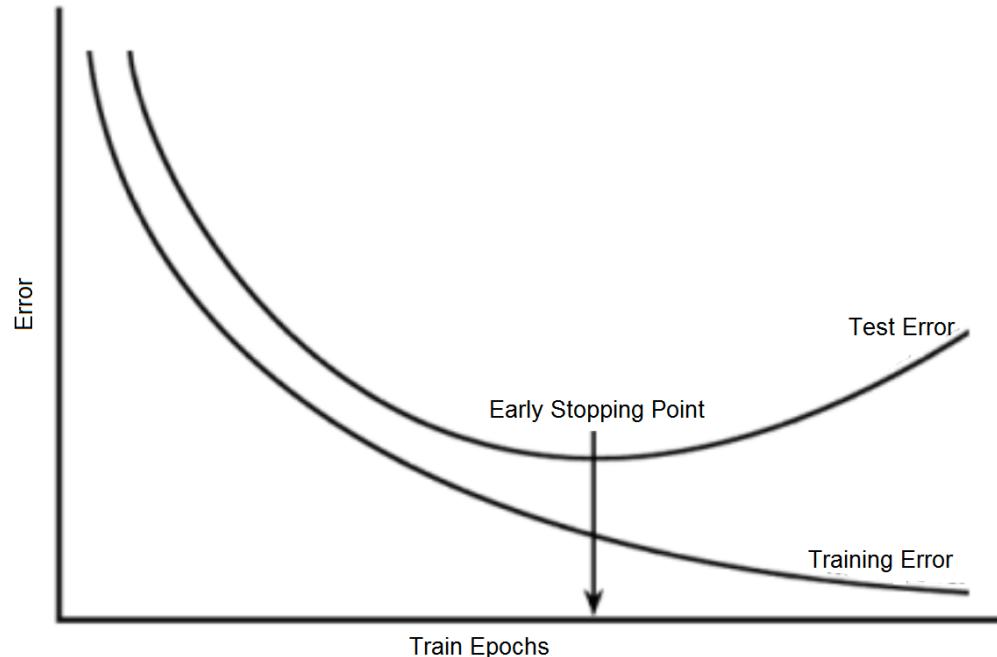


Balanced

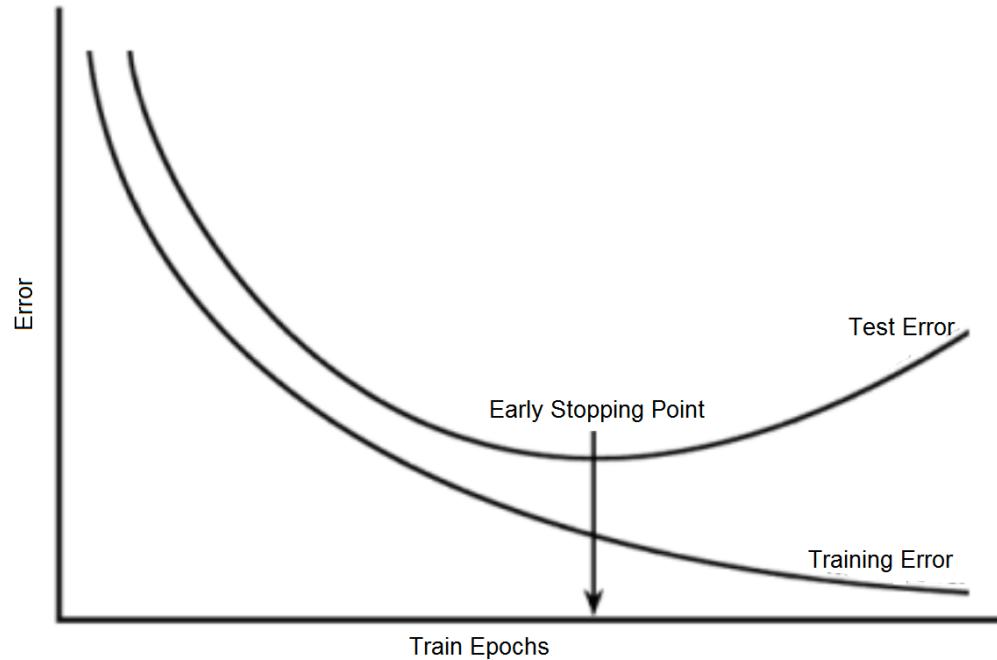


Overfitting

Early stopping

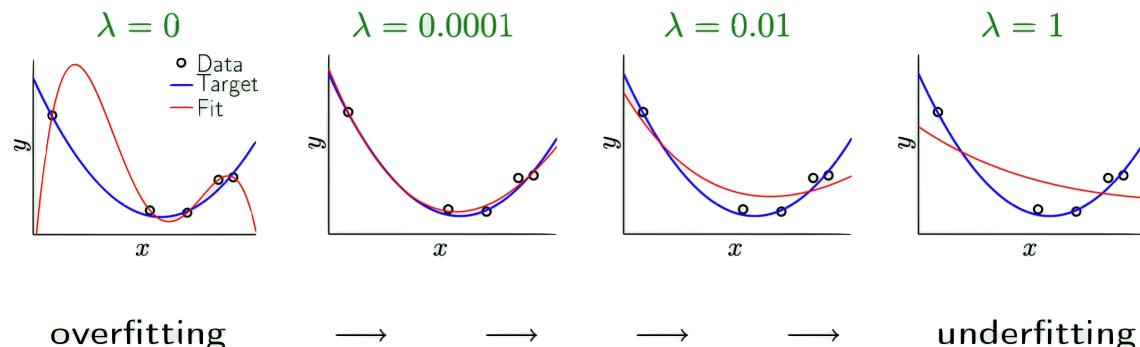


Early stopping



L2 регуляризация

- $\theta^* = \operatorname{argmin}_{\theta} \sum_{i=1}^N l(f_{\theta}(x_i), y_i) + \lambda \sum_{j=1}^T \theta_j^2$
- Интуиция: хотим, чтобы параметры модели были не очень большими
- λ — параметр регуляризации



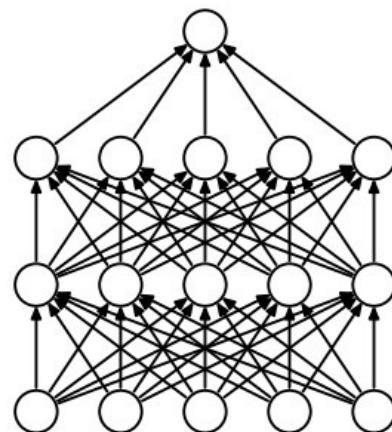
L2 регуляризация

```
CLASS torch.optim.SGD(params, lr=<required parameter>, momentum=0, dampening=0,  
    weight_decay=0, nesterov=False, *, maximize=False, foreach=None,  
    differentiable=False) [SOURCE]
```

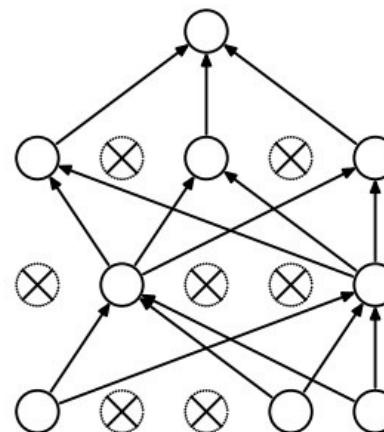


Dropout

CLASS `torch.nn.Dropout($p=0.5$, inplace=False)` [SOURCE]



(a) Standard Neural Net



(b) After applying dropout.

Вопросы

