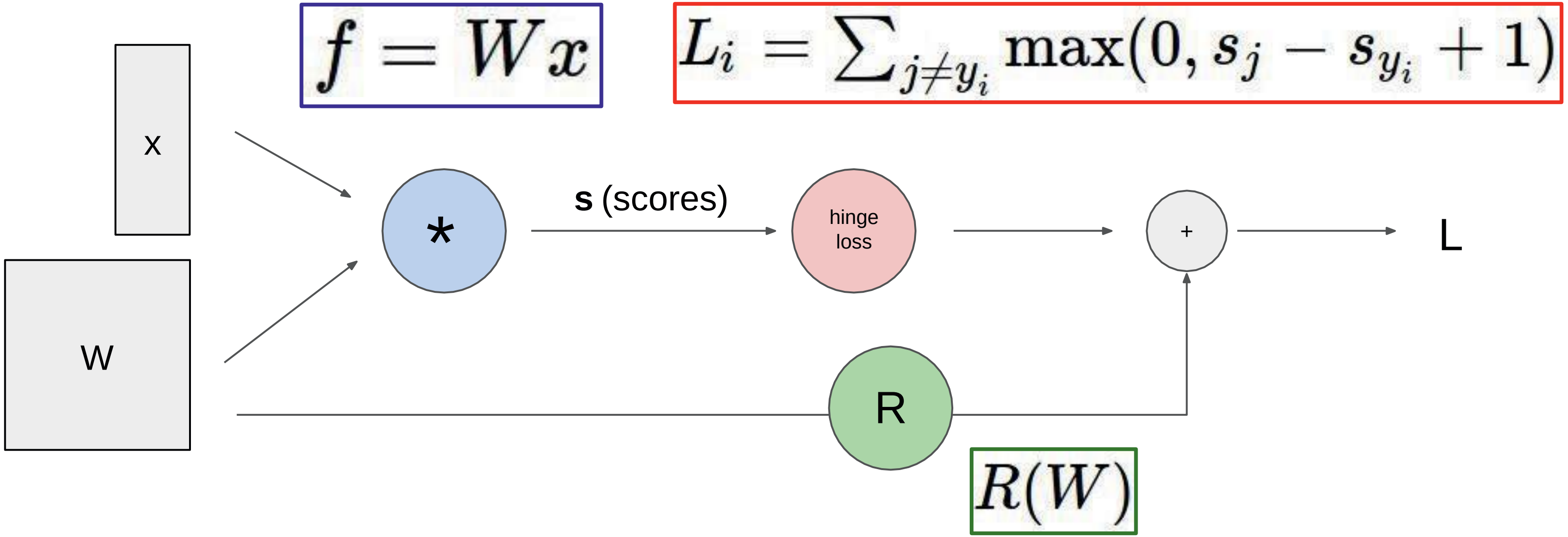
Лекция 7:
Обучение сетей,
Часть первая

Вспоминаем...

Computational graphs

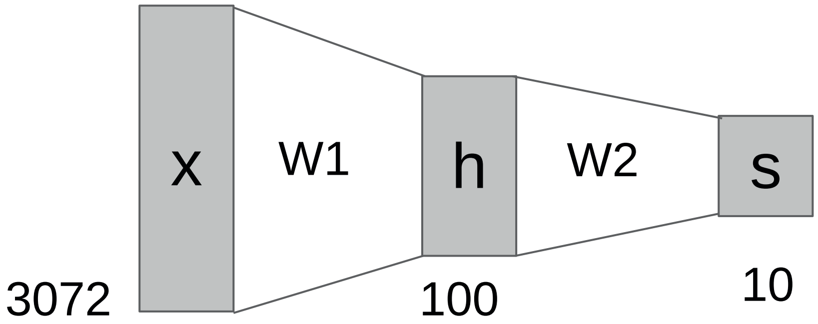


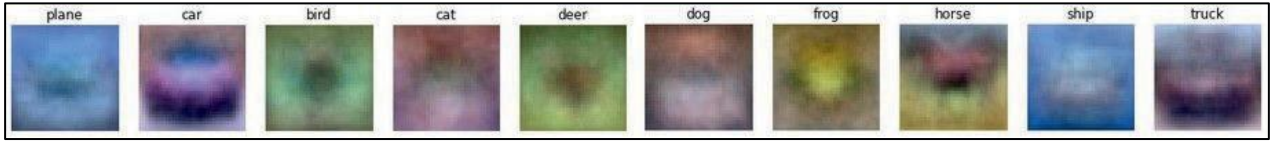
Вспоминаем...

Neural Networks

Linear score function: $$f = Wx$$

2-layer Neural Network $$f = W_2 \max(0, W_1 x)$$





Вспоминаем...

Convolutional Neural Networks

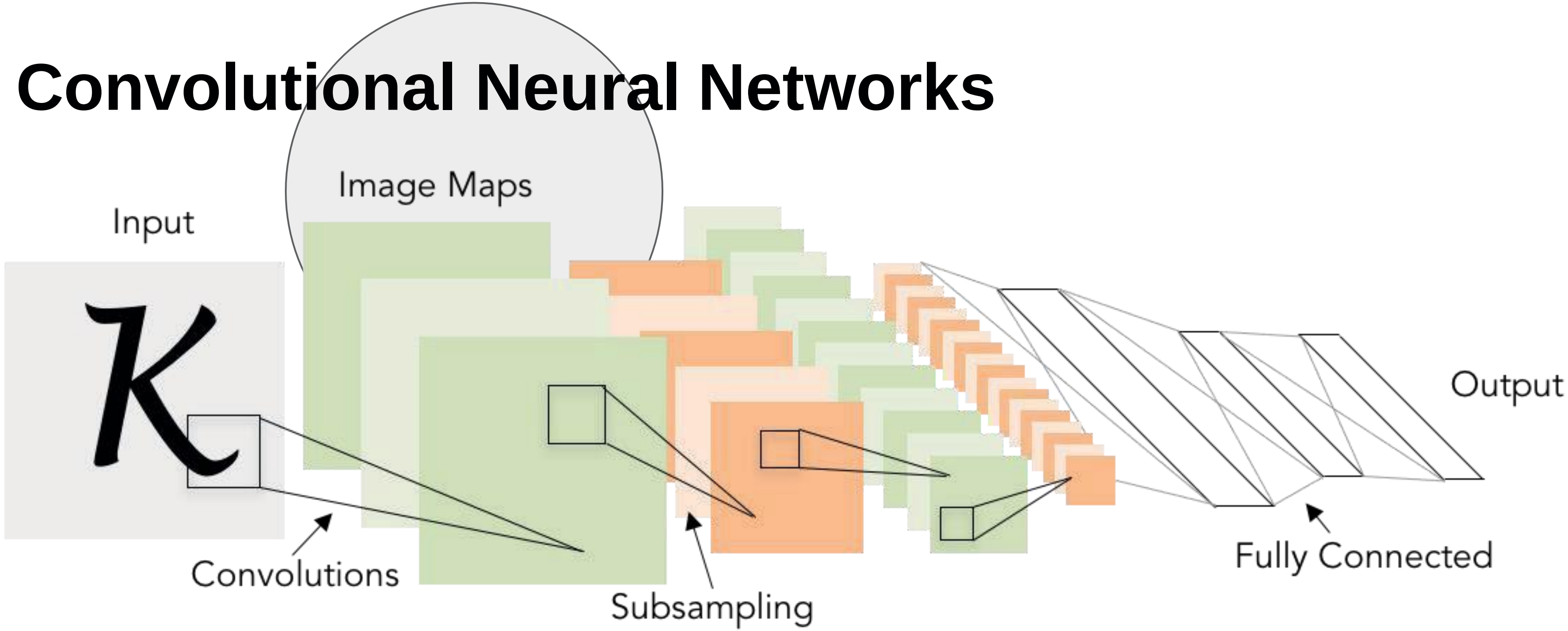


Illustration of LeCun et al. 1998 from CS231n 2017 Lecture 1

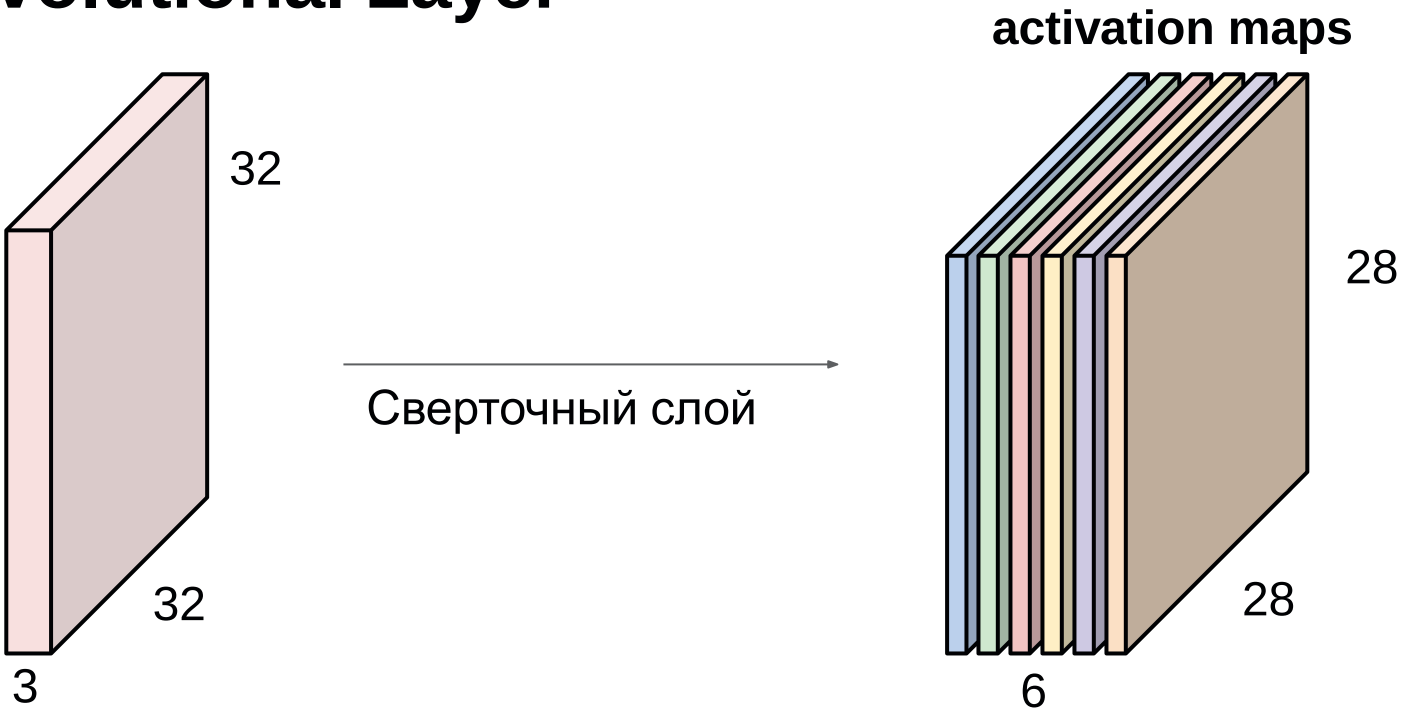
Вспоминаем...

Convolutional Layer

Вспоминаем...

Convolutional Layer

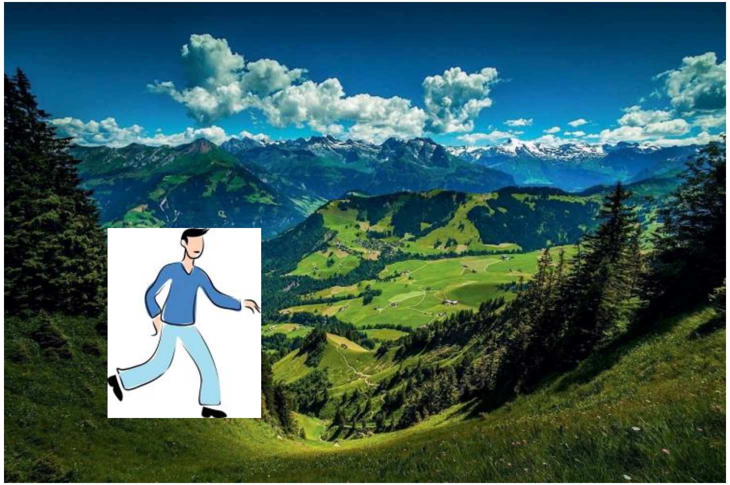
Например, для 6 фильтров 5x5, получим 6 независимых карт активации:

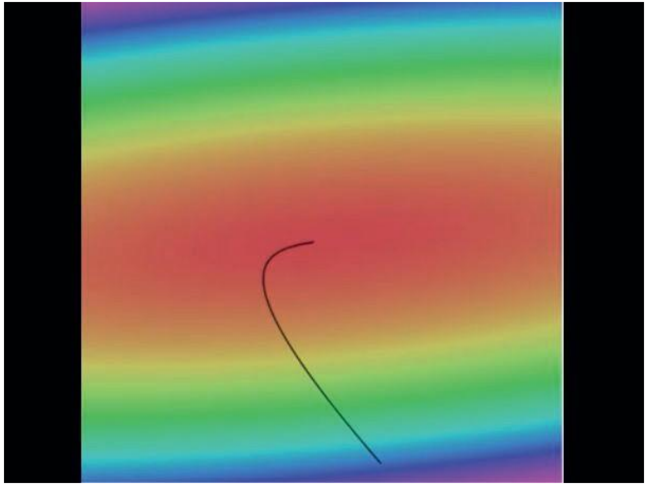


Соберем из них “новое изображение” размером 28x28x6

Вспоминаем...

Оптимизация для подбора параметров сети





```
# Vanilla Gradient Descent

while True:
  weights_grad = evaluate_gradient(loss_fun, data, weights)
  weights += - step_size * weights_grad # perform parameter update
```

Вспоминаем...

Mini-batch SGD

Цикл:

1. **Выбираем случайный** batch поднабор данных
2. **Прямой (forward)** прямой проход по графу сети, получаем loss
3. **Обратный** проход **(backprop)** для расчета градиентов
4. **Обновим параметры** на основе градиентов

Вспоминаем...

Hardware + Software

PyTorch

TensorFlow

Поехали:
обучение нейронных сетей
Training Neural Networks

Общий план

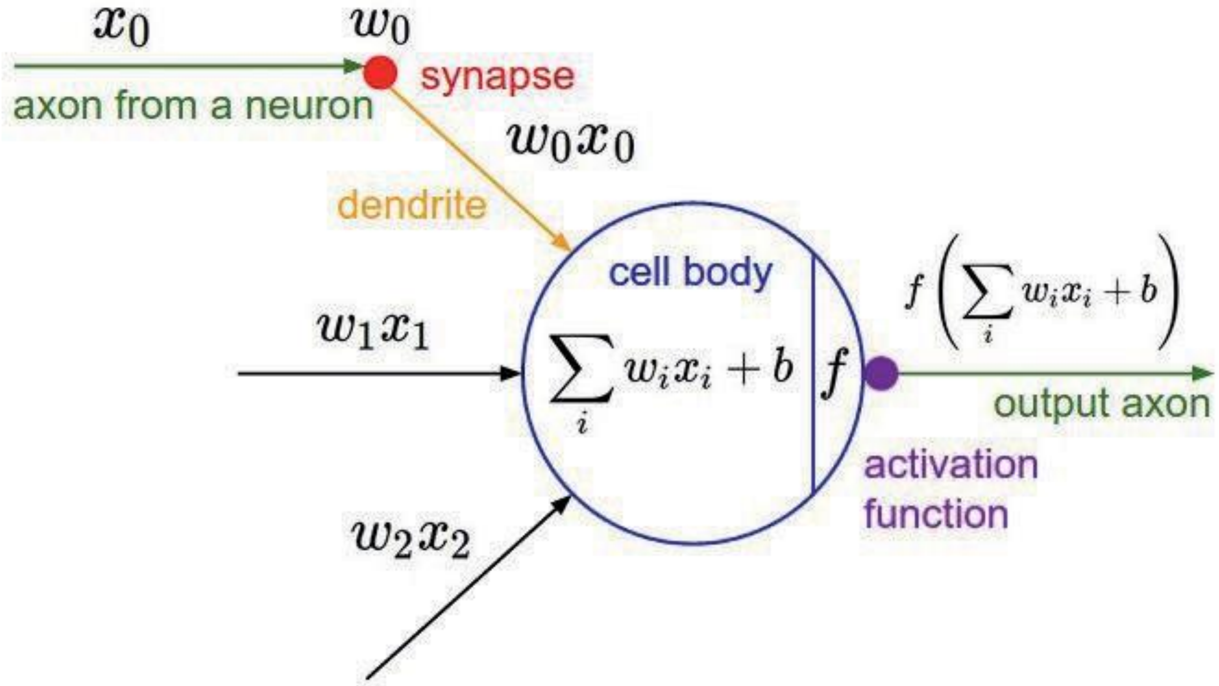
1. **Инициализация**
   Функции активации, подготовка данных, инициализация весов, регуляризация, проверка градиентов
2. **Динамика обучения**
   transfer learning, *мониторинг процесса обучения, обновление весов, оптимизация гиперпараметров*
3. **Тестирование**
   Ансамбли моделей, аугментация на этапе тестирования

Часть 1

- Функции активации / Activation Functions
- Подготовка данных / Data Preprocessing
- Инициализация весов / Weight Initialization
- Пакетная нормализация / Batch Normalization
- Transfer learning

Функции активации
Activation Functions

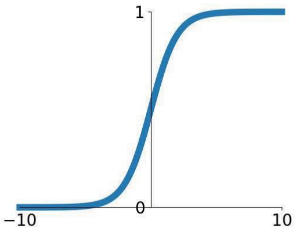
Функции активации



Функции активации

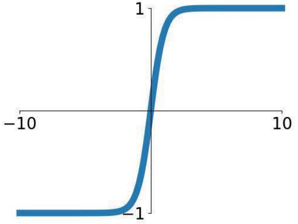
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



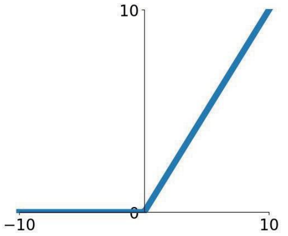
tanh

$$\tanh(x)$$



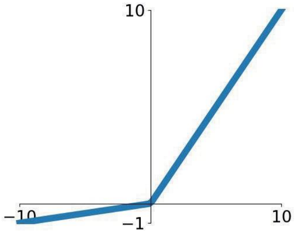
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

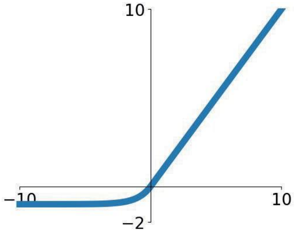


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

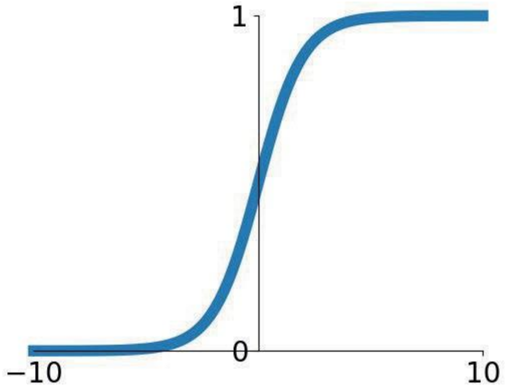
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Функции активации

$$\sigma(x) = 1/(1+e^{-x})$$

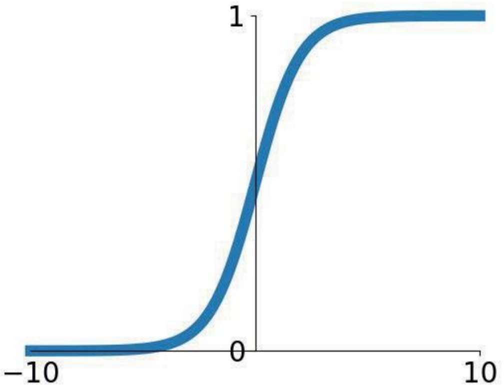


Sigmoid

- Отображает все в отрезок [0,1]
- Исторически имеет нейробиологическую трактовку насыщения, приводящего к активации биологического нейрона

Функции активации

$$\sigma(x) = 1/(1 + e^{-x})$$

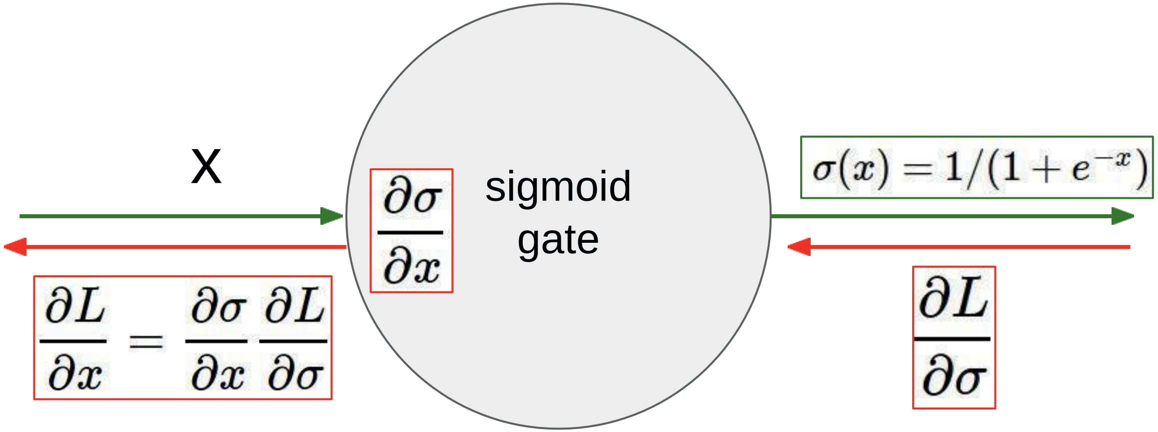


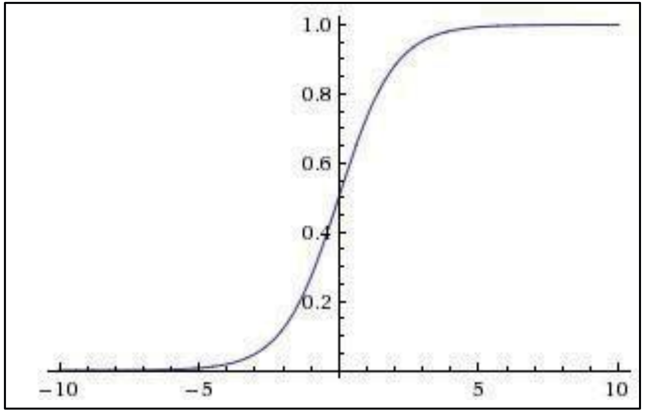
Sigmoid

- Отображает все в отрезок [0,1]
- Исторически имеет нейробиологическую трактовку насыщения, приводящего к активации биологического нейрона

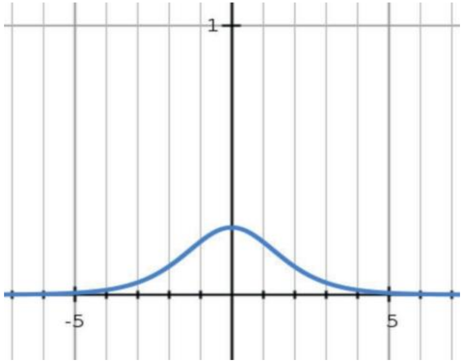
Три проблемы:

1. При насыщении нейрона градиенты «умирают» (dead gradients)





$$\frac{\partial \sigma(x)}{\partial x} = \sigma(x)\,(1 - \sigma(x)\,)$$



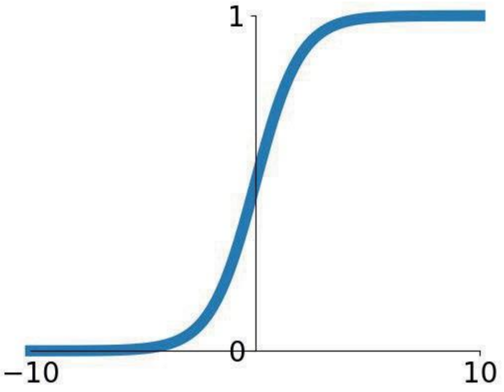
Что будет при x = -10?

Что будет при x = 0?

Что будет при x = 10?

Функции активации

$$\sigma(x) = 1/(1 + e^{-x})$$



Sigmoid

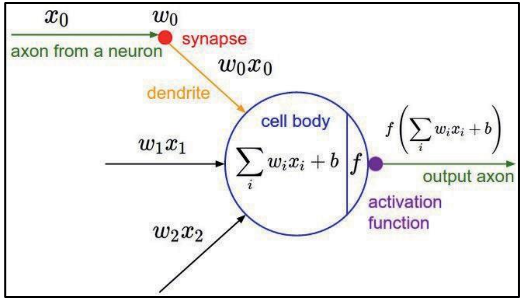
- Отображает все в отрезок [0,1]
- Исторически имеет нейробиологическую трактовку насыщения, приводящего к активации биологического нейрона

Три проблемы:

1. При насыщении нейрона градиенты «умирают» (dead gradients)
2. Отклик сигмоиды не центрирован относительно нуля

Что будет при всегда положительном входе нейрона...

$$f\left(\sum_i w_i x_i + b\right)$$

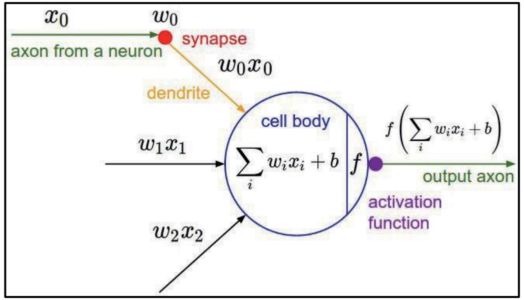


Что можно сказать про градиенты по **w**?

$$\frac{\partial L}{\partial w} = \sigma(\textstyle\sum_i w_i x_i + b)(1 - \sigma(\sum_i w_i x_i + b))x \times upstream\_gradient$$

Что будет при всегда положительном входе нейрона...

$$f\left(\sum_i w_i x_i + b\right)$$



Что можно сказать про градиенты по **w**?

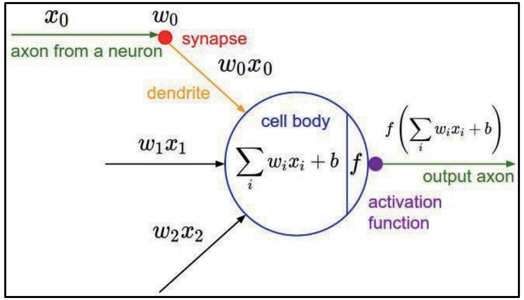
Локальные градиенты – положительные

Все x - положительные

$$\frac{\partial L}{\partial w} = \boxed{\sigma(\textstyle\sum_i w_i x_i + b)(1 - \sigma(\sum_i w_i x_i + b))}\boxed{x} \times upstream\_gradient$$

Что будет при всегда положительном входе нейрона...

$$f\left(\sum_i w_i x_i + b\right)$$



Что можно сказать про градиенты по **w**?

Локальные градиенты – положительные

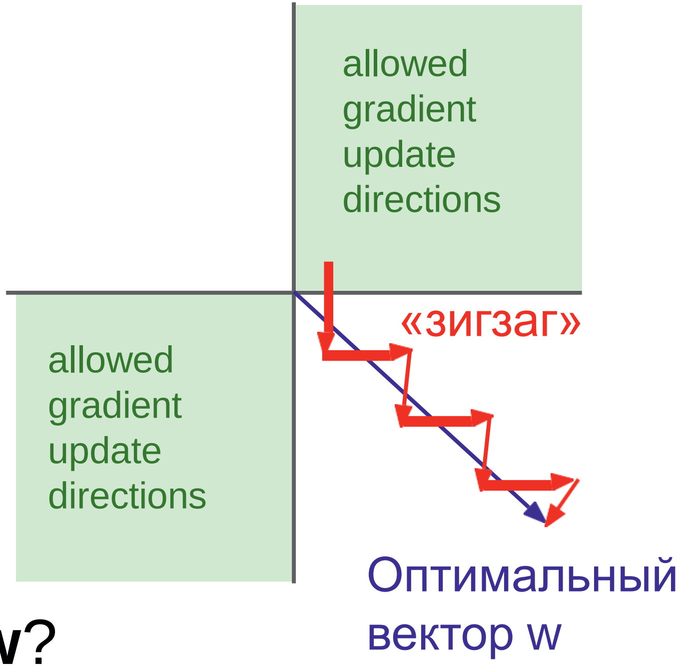
Все x – положительные

Значит знак **для всех $\mathbf{w}_i$** совпадает со знаком восходящих (upstream) градиетнов!

$$\boxed{\frac{\partial L}{\partial w}} = \boxed{\sigma(\textstyle\sum_i w_i x_i + b)(1 - \sigma(\sum_i w_i x_i + b))}\boxed{x} \times \boxed{upstream\_gradient}$$

Что будет при всегда положительном входе нейрона...

$$f\left(\sum_i w_i x_i + b\right)$$



Что можно сказать про градиенты по **w**?

Все либо положительные, либо отрицательные…

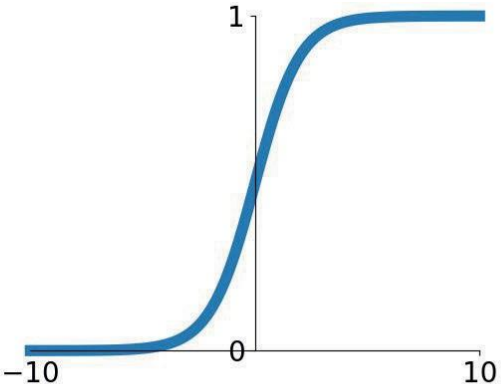
(Это для одного элемента выборки!

Минибатчи спасают ситуацию)

Функции активации

$$\sigma(x) = 1/(1+e^{-x})$$

- Отображает все в отрезок [0,1]
- Исторически имеет нейробиологическую трактовку насыщения, приводящего к активации биологического нейрона



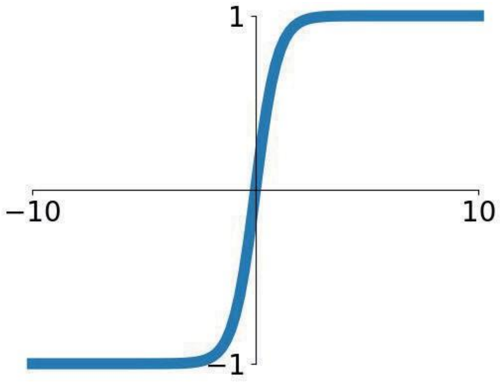
Sigmoid

Три проблемы:

1. При насыщении нейрона градиенты «умирают» (dead gradients)
2. Отклик сигмоиды не центрирован относительно нуля
3. Считать exp() немного затратно…

И особенно неудобно в квантованных сетках, например, в int16!

Функции активации

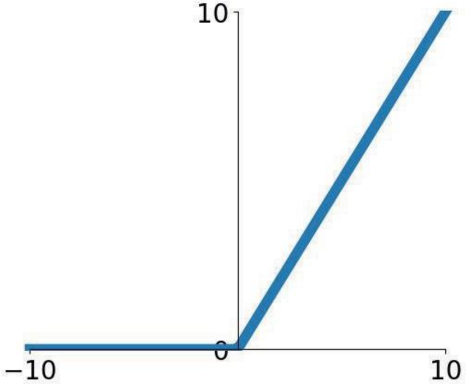


tanh(x)

- Отображает числовую прямую в отрезок [-1,1]
- Центрирована в нуле (хорошо!)
- Насышение все еще убивает градиеты :(
- Считать все еще не очень удобно…

[LeCun et al., 1991]

Функции активации



ReLU
(Rectified Linear Unit)

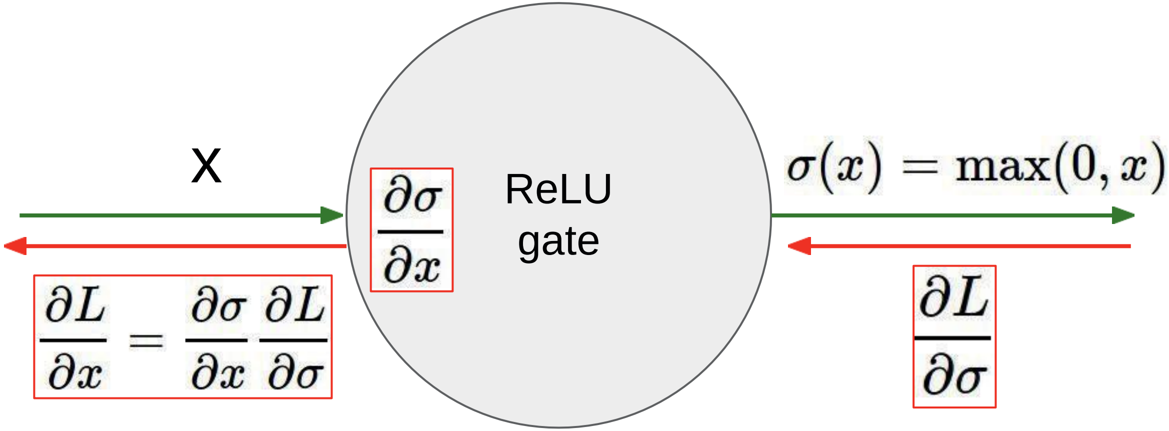
Вычисляем **f(x) = max(0,x)**

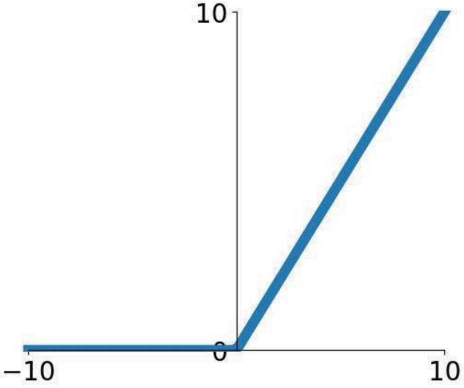
- Без насыщения в положительной части
- Очень простая вычислительно
- На практике сходится кратно быстрее чем sigmoid/tanh, примерно в 6 раз

- Не центрирована в нуле…

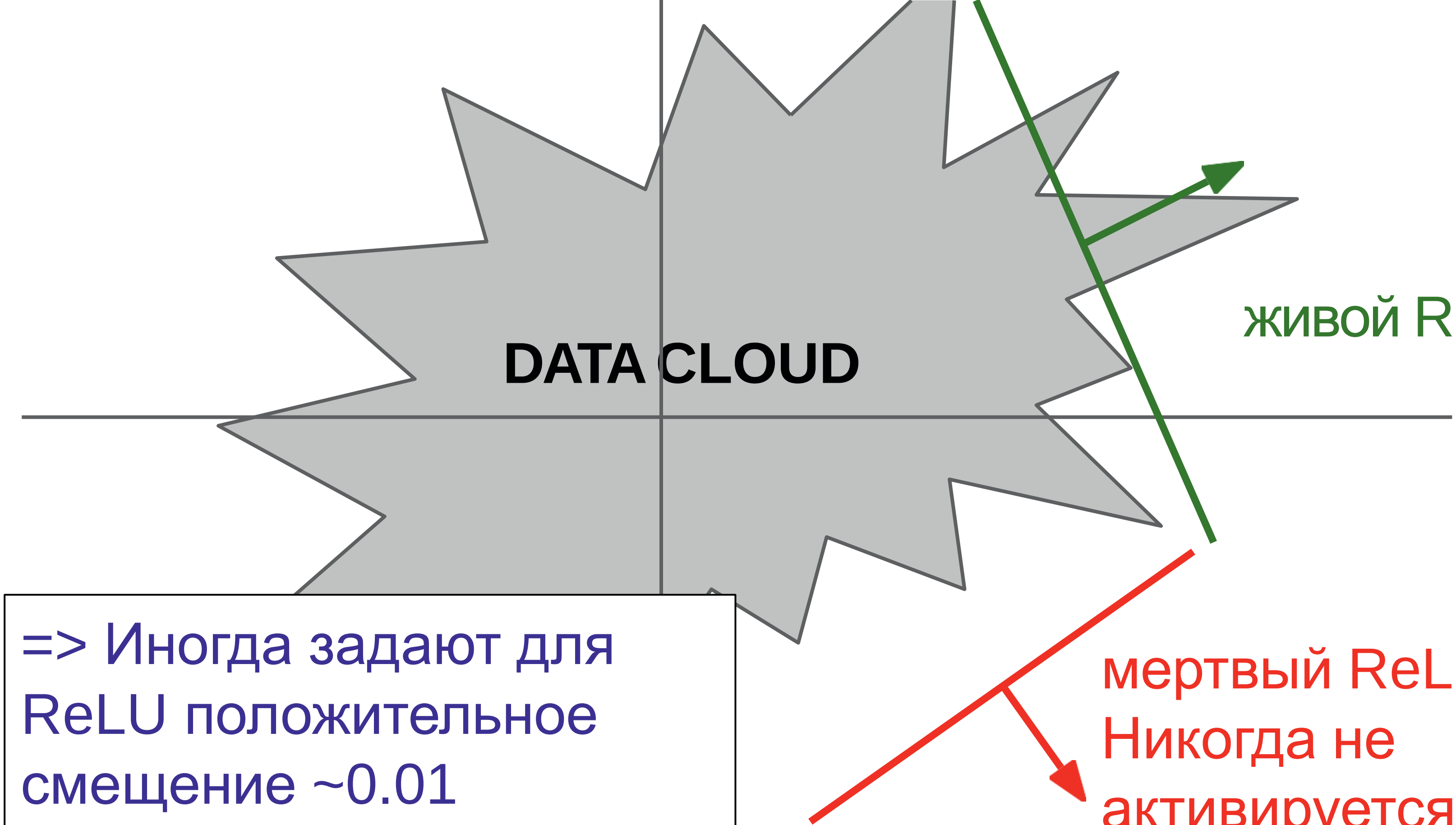
[Krizhevsky et al., 2012]

Функции активации





Что будет при x = -10?
Что будет при x = 0?
Что будет при x = 10?

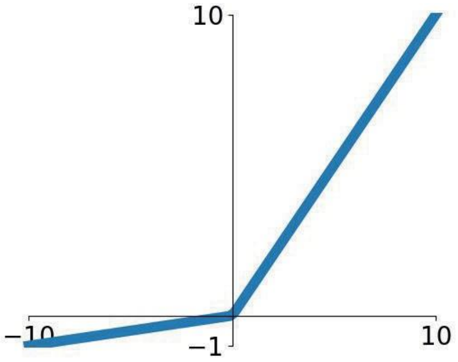


=> Иногда задают для ReLU положительное смещение ~0.01

мертвый ReLU
Никогда не активируется
=> Обновления не будет

Функции активации

[Mass et al., 2013]
[He et al., 2015]



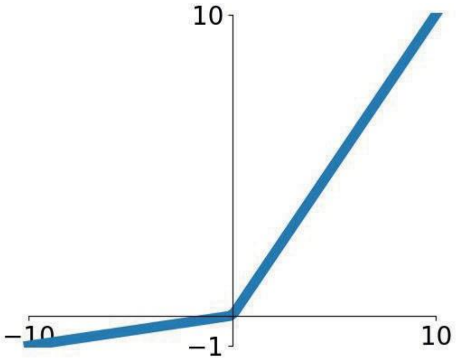
- Без насыщения
- Вычислительно эффективно
- Сходится быстрее сигмоиды! (в ~6 раз)
- **не «умирает»**

Leaky ReLU

$$f(x) = \max(0.01x, x)$$

Функции активации

[Mass et al., 2013]
[He et al., 2015]



- Без насыщения
- Вычислительно эффективно
- Сходится быстрее сигмоиды! (в ~6 раз)
- **не «умирает»**

Leaky ReLU

$$f(x) = \max(0.01x, x)$$

Parametric Rectifier (PReLU)

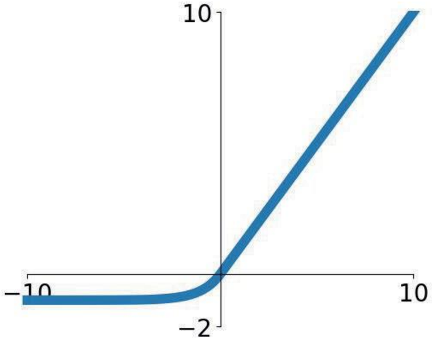
$$f(x) = \max(\alpha x, x)$$

Обучаемый параметр альфа

Функции активации

[Clevert et al., 2015]

Exponential Linear Units (ELU)



$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha\left(\exp(x) - 1\right) & \text{if } x \le 0 \end{cases}$$

(по умолчанию Alpha = 1)

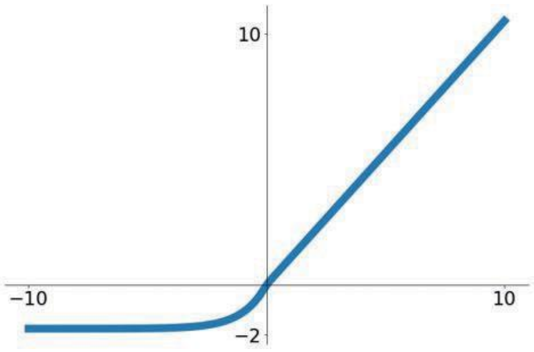
- Все плюсы ReLU
- Среднее достаточно близко к нулю
- Отрицательное насыщение дает устойчивость к шуму, в сравнении с Leaky ReLU

- Надо считать exp()

Функции активации

[Klambauer et al. ICLR 2017]

Scaled Exponential Linear Units (SELU)



$$f(x) = \begin{cases} \lambda x & \text{if } x > 0 \\ \lambda\alpha(e^x - 1) & \text{otherwise} \end{cases}$$

$\alpha = 1.6733$, $\lambda = 1.0507$

- Масштабируемая версия ELU лучше для глубоких сетей
- “Самонормализация”
- SELU сети могут работать без BatchNorm
  - Кое-что еще позже

Функции активации

Maxout “Нейрон”

[Goodfellow et al., 2013]

- Реализует нелинейность не являясь скалярным произведением
- Обобщает ReLU and Leaky ReLU
- Почти линеен! Без насыщения! Не умирает!

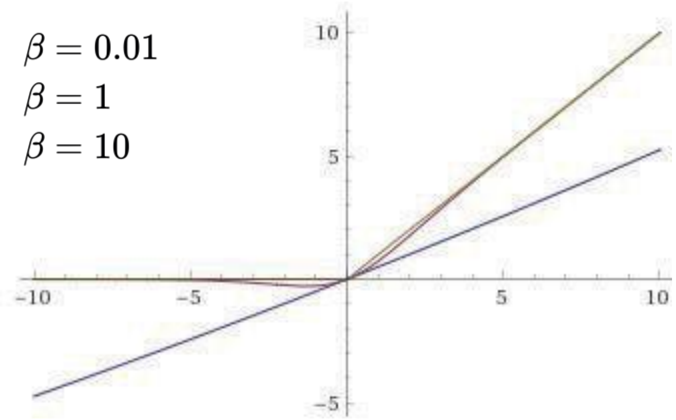
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

Проблема: удваивает количество весов :(

Функции активации

[Ramachandran et al. 2018]

Swish



$$f(x) = x\sigma(\beta x)$$

- Оптимальный выбор из класса нелинейностей.
- Swish дает лучшую точность на CIFAR-10

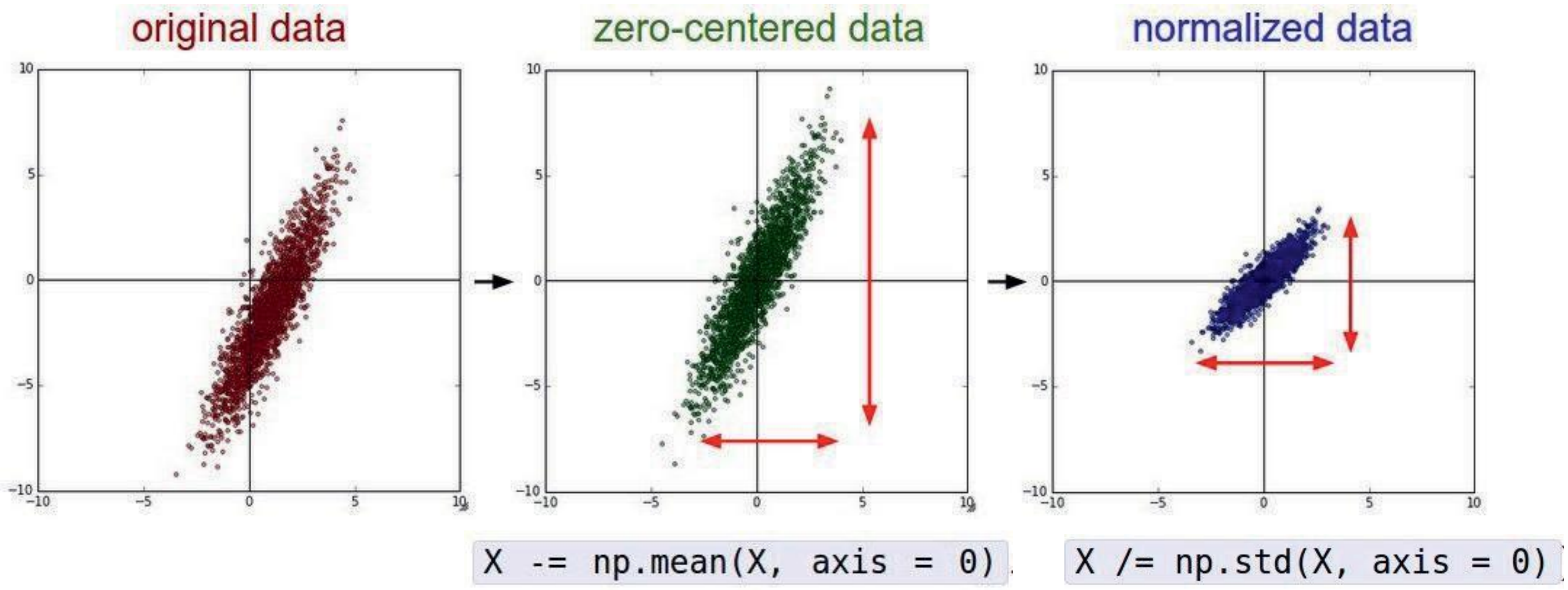
TLDR: на практике:

- Пользуйтесь ReLU. Если обучение идет плохо, уменьшайте learning rate
- Пробуйте Leaky ReLU / ELU / SELU / Maxout
- Пробуйте PReLU с маленьким learning rate
- Не используйте sigmoid или tanh

  Tanh может быть иногда полезен на сигналах

Подготовка данных

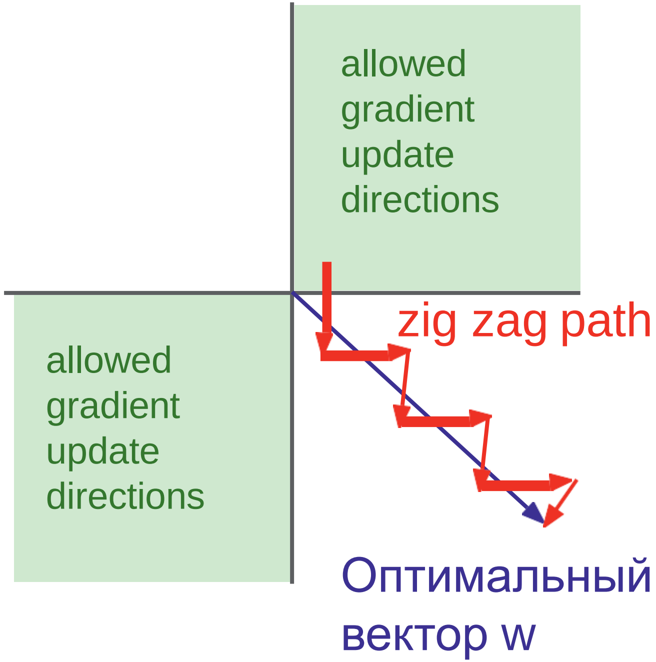
Подготовка данных



Полагаем X [NxD] матрица данных,
наблюдения по строкам

Вспоминаемм: Рассмотрим случай положительных входов…

$$f\left(\sum_i w_i x_i + b\right)$$

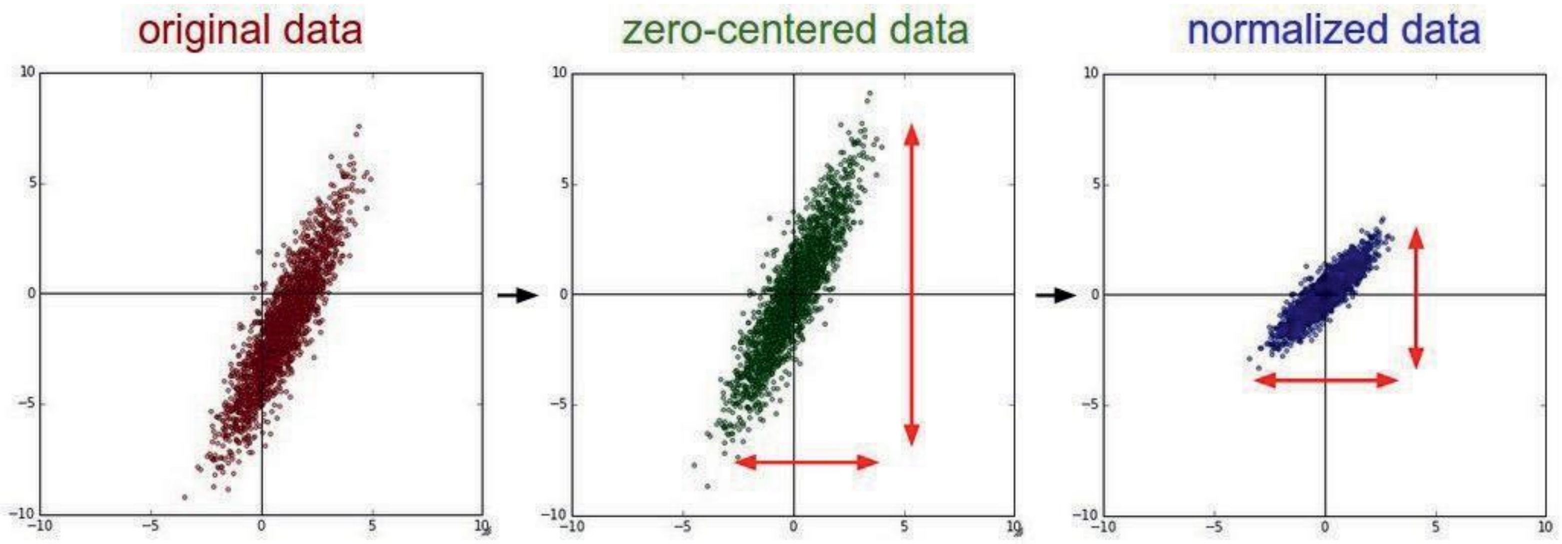


Что будет с градиентом по **w**?

Всегда либо положителен либо отрицателен :(

Нулевое среднее данных нам поможет!

Подготовка данных



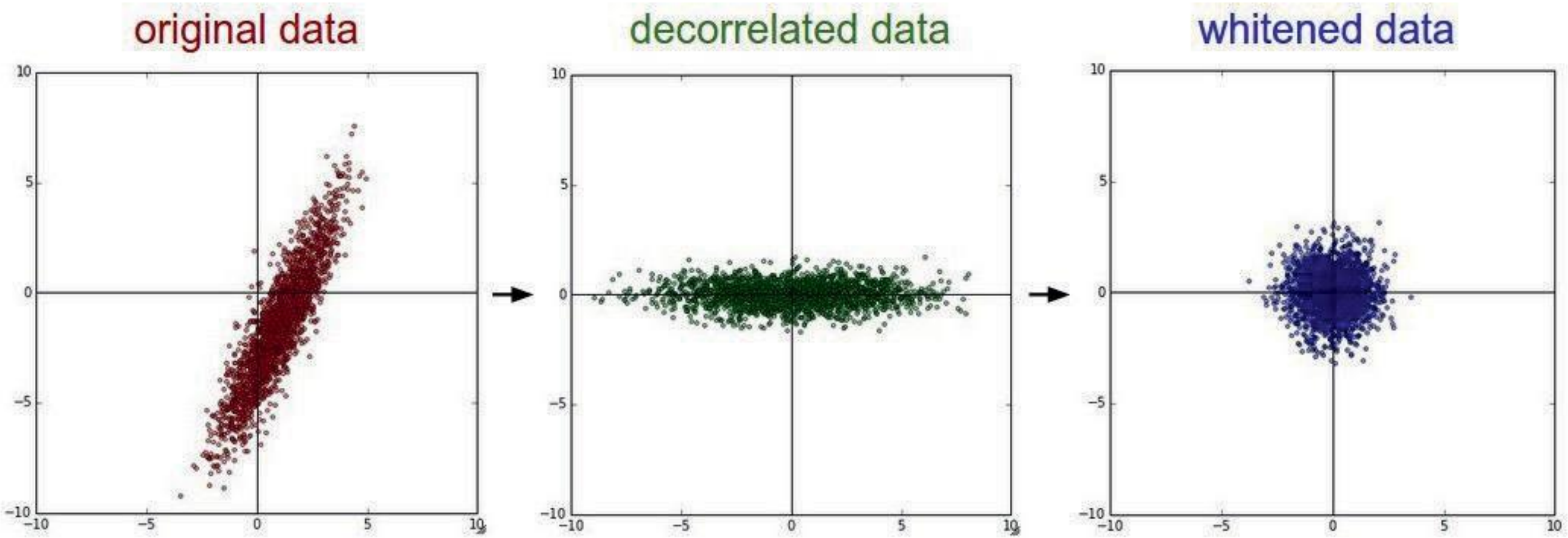

```
X -= np.mean(X, axis = 0)
```

```
X /= np.std(X, axis = 0)
```

(Assume X [NxD] is data matrix, each example in a row)

Подготовка данных

На практике полезно: **PCA** и **Whitening**

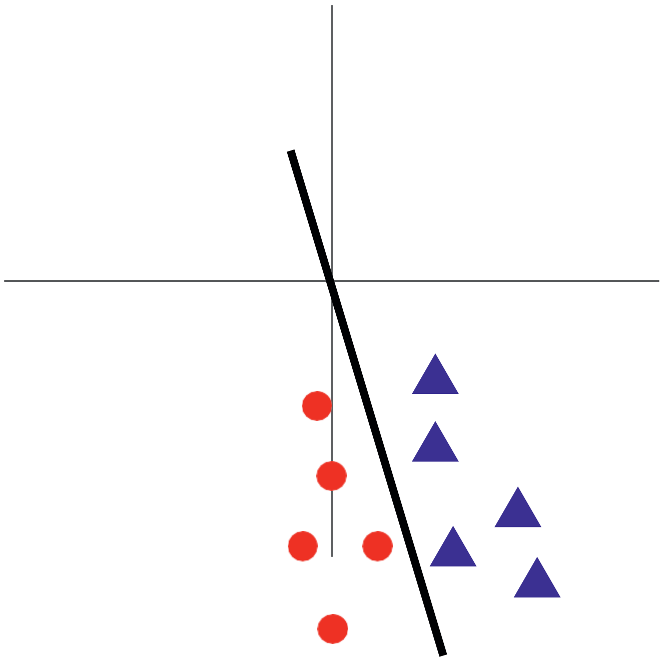


(ковариационная матрица данных диагональная)

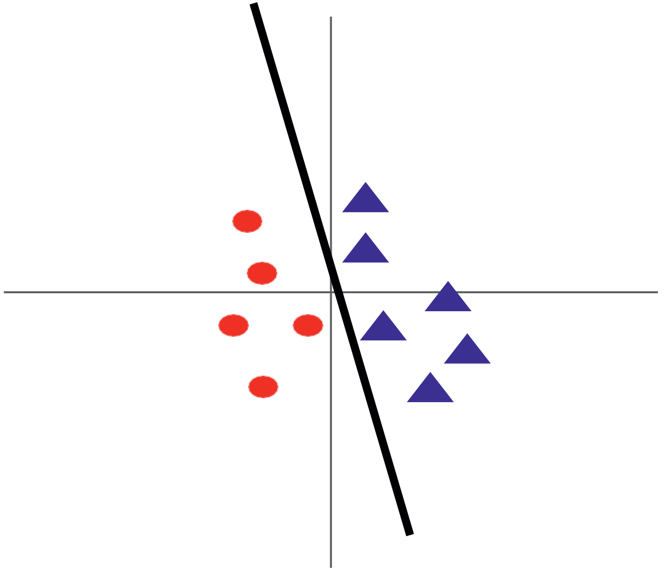
(ковариационная матрица данных единичная)

Подготовка данных

До нормализации: функция потерь чувствительна к малым изменениям обучение неустойчиво



После нормализации: функция потерь не столь чувствительна к изменениям весов, оптимизация более устойчива



TLDR: На практике для изображений:

только центрируем

Например для CIFAR-10 с изображениями [32,32,3]

- Вычитаем среднее изображение

  (для AlexNet) (mean image = [32,32,3] array)
- Вычитаем поканальное среднее

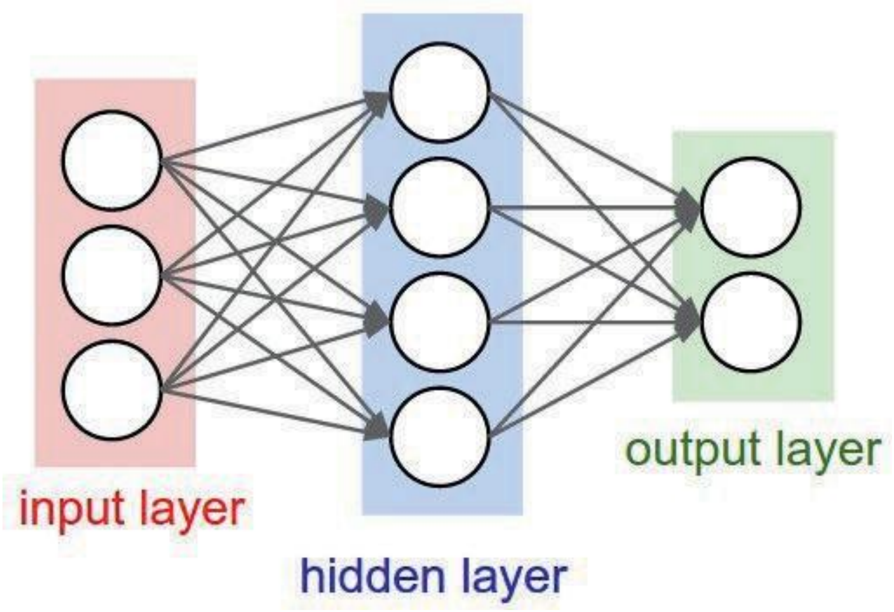
  (для VGGNet) (поканальное среднее = 3 числа)
- Вычитаем поканальное среднее
  Делим на поканальное std

  (для ResNet) (поканальное среднее = 3 числа)

Для изображений PCA или whitening не делают

Инициализация весов

- Как инициализировать веса?



- Первая идея: **малые случайные величины**

гауссов шум с нулевым средним и 0.01 STD

```
W = 0.01 * np.random.randn(Din, Dout)
```

Работает для малых сетей, не работает для глубоких.

Инициализация весов: статистика

```
dims = [4096] * 7
hs = []
x = np.random.randn(16, dims[0])
for Din, Dout in zip(dims[:-1], dims[1:]):
    W = 0.01 * np.random.randn(Din, Dout)
    x = np.tanh(x.dot(W))
    hs.append(x)
```

Прямой проход для 6-ти слойной сети с 4096 скрытыми нейронами

Что случится с активациями на последнем слое?

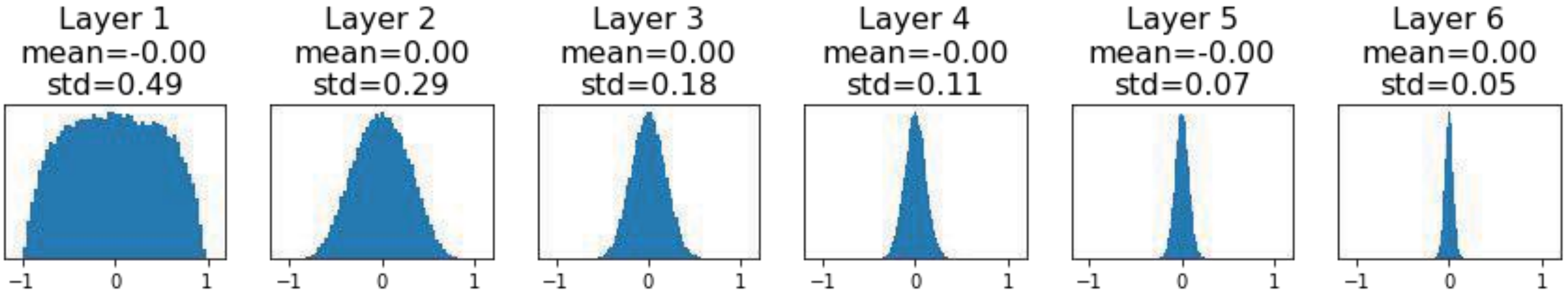
Инициализация весов: статистика

```
dims = [4096] * 7
hs = []
x = np.random.randn(16, dims[0])
for Din, Dout in zip(dims[:-1], dims[1:]):
    W = 0.01 * np.random.randn(Din, Dout)
    x = np.tanh(x.dot(W))
    hs.append(x)
```

Прямой проход для 6-ти слойной сети с 4096 скрытыми нейронами

Все активации уходят в ноль для глубоких слоев

Q: Как будут выглядеть градиенты dL/dW?



Инициализация весов: статистика

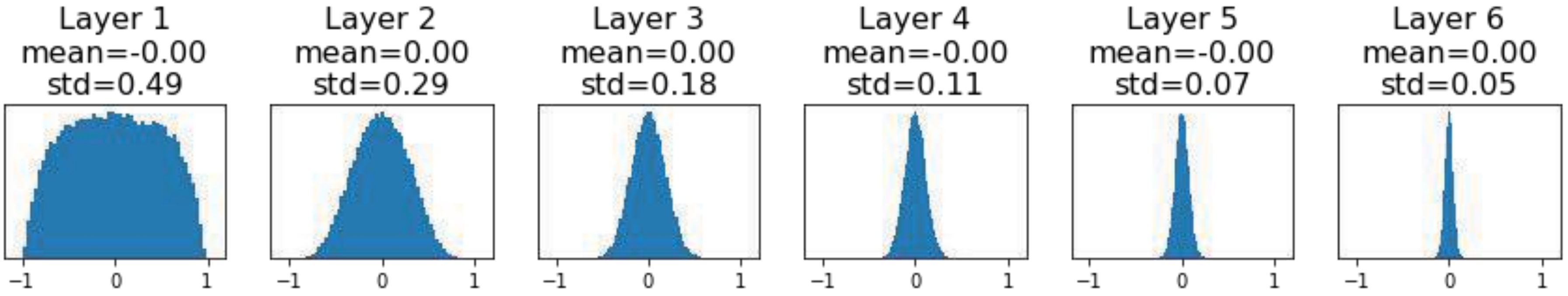
```
dims = [4096] * 7
hs = []
x = np.random.randn(16, dims[0])
for Din, Dout in zip(dims[:-1], dims[1:]):
    W = 0.01 * np.random.randn(Din, Dout)
    x = np.tanh(x.dot(W))
    hs.append(x)
```

Прямой проход для 6-ти слойной сети с 4096 скрытыми нейронами

Все активации уходят в ноль для глубоких слоев

Q: Как будут выглядеть градиенты dL/dW?

A: Близко к нулю…



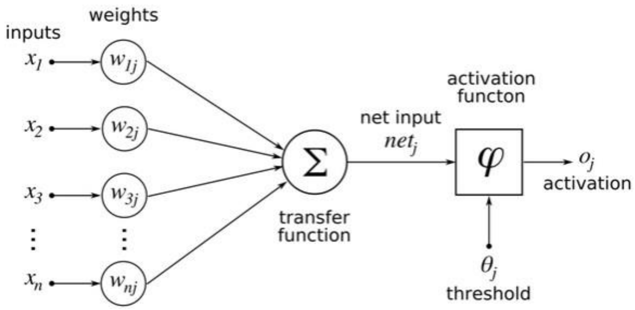
Инициализация весов: “Xavier”

```
dims = [4096] * 7
hs = []
x = np.random.randn(16, dims[0])
for Din, Dout in zip(dims[:-1], dims[1:]):
    W = np.random.randn(Din, Dout) / np.sqrt(Din)
    x = np.tanh(x.dot(W))
    hs.append(x)
```

“Xavier” initialization:
std = 1/sqrt(Din)

Glorot and Bengio, “Understanding the difficulty of training deep feedforward neural networks”, AISTAT 2010

Обоснование инициализации Ксавьера



$$s = \sum_i^n w_i x_i$$

$$\mathrm{var}(s) = E(s - E(s))^2$$

$$\mathrm{var}(s) = var\Big(\sum_i^N \mathbf{x}_i \mathbf{w}_i\Big) = \sum_i^N \mathrm{var}(\mathbf{x}_i \mathbf{w}_i) =$$

$$= \sum_i^N E\big((\mathbf{x}_i\mathbf{w}_i) - E(\mathbf{x}_i\mathbf{w}_i)\big)^2 = \sum_i^N E\Big((\mathbf{x}_i\mathbf{w}_i)^2 - 2(\mathbf{x}_i\mathbf{w}_i)E(\mathbf{x}_i\mathbf{w}_i) + E(\mathbf{x}_i\mathbf{w}_i)^2\Big)$$

нулевые средние: $E(\mathbf{x}_i) = 0, \qquad E(\mathbf{w}_i) = 0$

$$\sum_i^N E\Big((\mathbf{x}_i\mathbf{w}_i)^2 - 2(\mathbf{x}_i\mathbf{w}_i)\cancelto{0}{E(\mathbf{x}_i\mathbf{w}_i)} + (\cancelto{0}{E(\mathbf{x}_i\mathbf{w}_i)})^2\Big) = \sum_i^N E\Big((\mathbf{x}_i\mathbf{w}_i)^2\Big)$$

$$\sum_i^N E\Big((\mathbf{x}_i\mathbf{w}_i)^2\Big) = \sum_i^N \Big(\mathrm{var}(\mathbf{x}_i)\,\mathrm{var}(\mathbf{w}_i)\Big)$$

считаем $\mathbf{x}_i$, $\mathbf{w}_i$ одинаково распределенными:

$$\sum_i^N \Big(\mathrm{var}(\mathbf{w}_i)\,\mathrm{var}(\mathbf{x}_i)\Big) = \boxed{n\,\mathrm{var}(\mathbf{w}_i)}\,\mathrm{var}(\mathbf{x}_i)$$

$$\mathrm{var}(ax) = a^2\,\mathrm{var}(x)$$

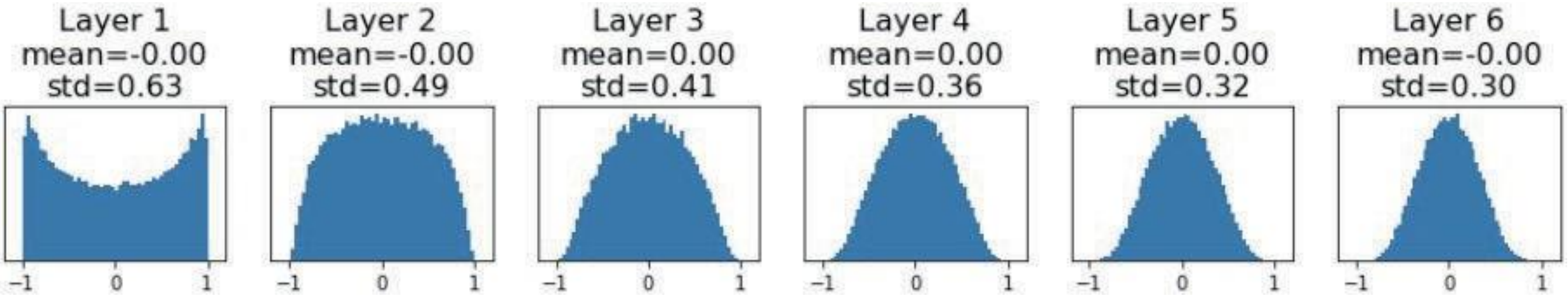
```
w = np.random.randn(n) / sqrt(n)
```

Инициализация весов: “Xavier”

```
dims = [4096] * 7
hs = []
x = np.random.randn(16, dims[0])
for Din, Dout in zip(dims[:-1], dims[1:]):
    W = np.random.randn(Din, Dout) / np.sqrt(Din)
    x = np.tanh(x.dot(W))
    hs.append(x)
```

“Xavier” initialization:
std = 1/sqrt(Din)

“Почти хорошо”: сравнимый масштаб активаций по всем слоям



Glorot and Bengio, “Understanding the difficulty of training deep feedforward neural networks”, AISTAT 2010

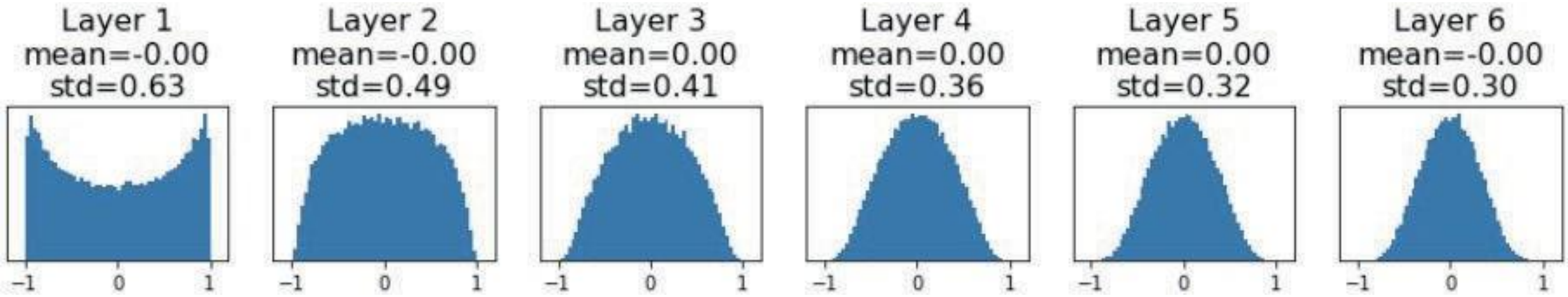
Инициализация весов: “Xavier”

```
dims = [4096] * 7
hs = []
x = np.random.randn(16, dims[0])
for Din, Dout in zip(dims[:-1], dims[1:]):
    W = np.random.randn(Din, Dout) / np.sqrt(Din)
    x = np.tanh(x.dot(W))
    hs.append(x)
```

“Xavier” initialization:
std = 1/sqrt(Din)

“Just right”: Activations are nicely scaled for all layers!

Для сверток,
Din = filter_size^2 * input_channels



Glorot and Bengio, “Understanding the difficulty of training deep feedforward neural networks”, AISTAT 2010

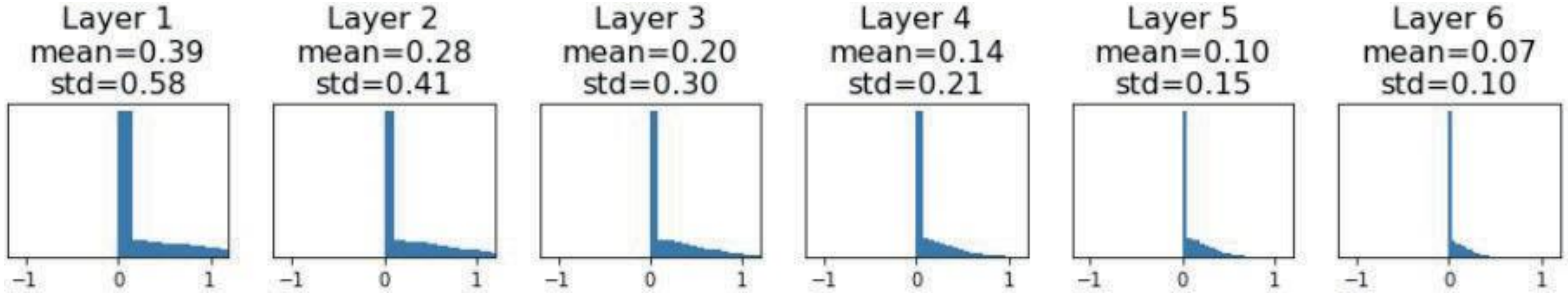
Инициализация весов: ReLU

```
dims = [4096] * 7
hs = []
x = np.random.randn(16, dims[0])
for Din, Dout in zip(dims[:-1], dims[1:]):
    W = np.random.randn(Din, Dout) / np.sqrt(Din)
    x = np.maximum(0, x.dot(W))
    hs.append(x)
```

Change from tanh to ReLU

Xavier предполагает активацию с нулевым средним

Для ReLU не работает!

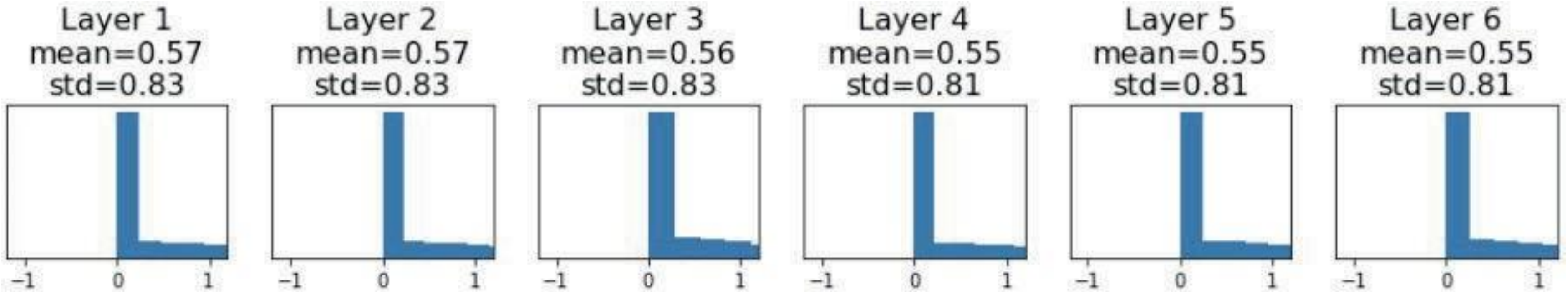


Инициализация весов: Kaiming / MSRA инициализация

```
dims = [4096] * 7
hs = []
x = np.random.randn(16, dims[0])
for Din, Dout in zip(dims[:-1], dims[1:]):
    W = np.random.randn(Din, Dout) * np.sqrt(2/Din)
    x = np.maximum(0, x.dot(W))
    hs.append(x)
```

ReLU correction: std = sqrt(2 / Din)

“Почти работает”:
Активации опять стали сохранять масштаб!



He et al, “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification”, ICCV 2015

Публикации по инициализации…

Understanding the difficulty of training deep feedforward neural networks
by Glorot and Bengio, 2010

Exact solutions to the nonlinear dynamics of learning in deep linear neural networks by Saxe et al, 2013

Random walk initialization for training very deep feedforward networks by Sussillo and Abbott, 2014

Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification by He et al., 2015

Data-dependent Initializations of Convolutional Neural Networks by Krähenbühl et al., 2015

All you need is a good init, Mishkin and Matas, 2015

Fixup Initialization: Residual Learning Without Normalization, Zhang et al, 2019

The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks, Frankle and Carbin, 2019

Batch Normalization

Batch Normalization

[Ioffe and Szegedy, 2015]

Давайте сделаем нулевое среднее и единичную дисперсию для активаций!

Чтобы это сделать на некотором слое:

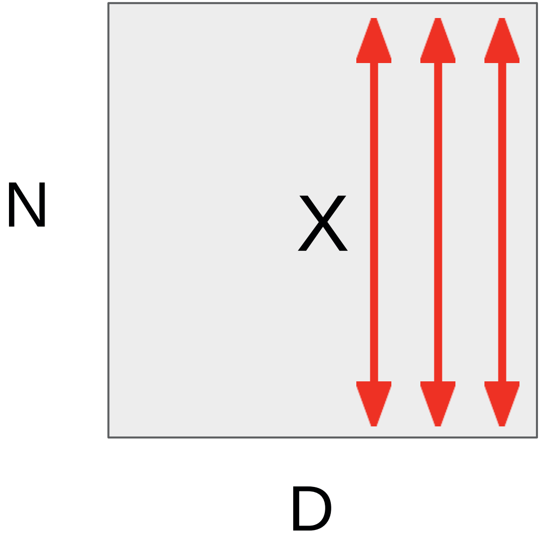
$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

Это дифференцируемая функция...

Batch Normalization

[Ioffe and Szegedy, 2015]

Input: $x : N \times D$



$$\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{i,j}$$

Поканальное среднее, размерности D

$$\sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N}(x_{i,j} - \mu_j)^2$$

Поканальная дисперсия, размерности D

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

Нормализованный x, Размерности N x D

Проблема: что если нулевое среднее и единичная дисперсия слишком сильное требование?

Batch Normalization

[Ioffe and Szegedy, 2015]

Вход: $x : N \times D$

Обучаемые параметры сдвига и масштаба:

$\gamma, \beta : D$

Если $\gamma = \sigma$, $\beta = \mu$ получим нулевое среднее и единичную дисперсию!

$$\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{i,j}$$

Поканальное среднее, размерности D

$$\sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N}(x_{i,j} - \mu_j)^2$$

Поканальная дисперсия, размерности D

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

Нормализованный x, размерности N x D

$$y_{i,j} = \gamma_j \hat{x}_{i,j} + \beta_j$$

Новый выход, размерности N x D

Batch Normalization: Тест

Оценки определены на минибатче; что делать на тесте?

Вход: $x : N \times D$

Обучаемые параметры сдвига и масштаба:

$$\gamma, \beta : D$$

Если $\gamma = \sigma$, $\beta = \mu$ получим нулевое среднее и единичную дисперсию!

$$\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{i,j}$$

Поканальное среднее, размерности D

$$\sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N}(x_{i,j} - \mu_j)^2$$

Поканальная дисперсия, размерности D

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

Нормализованный x, размерности N x D

$$y_{i,j} = \gamma_j \hat{x}_{i,j} + \beta_j$$

Новый выход, размерности N x D

Batch Normalization: Тест

Вход: $x : N \times D$

Обучаемые параметры сдвига и масштаба:

$\gamma, \beta : D$

На этапе теста батчнорм это линейный оператор! Может быть объединен с conv/FC

$\mu_j =$ Скользящее среднее величин с этапа обучения

Поканальное среднее, размерности D

$\sigma_j^2 =$ Скользящее среднее величин с этапа обучения

Поканальная дисперсия, размерности D

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$

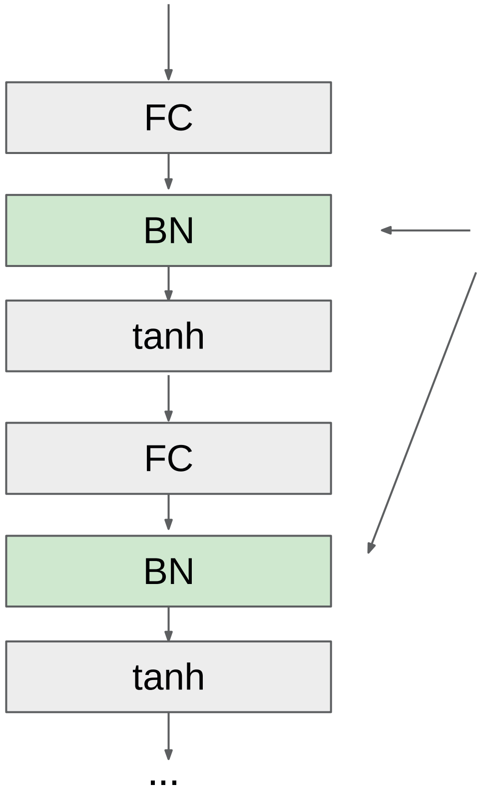
Нормализованный x, размерности N x D

$$y_{i,j} = \gamma_j \hat{x}_{i,j} + \beta_j$$

Новый выход, размерности N x D

Batch Normalization

[Ioffe and Szegedy, 2015]

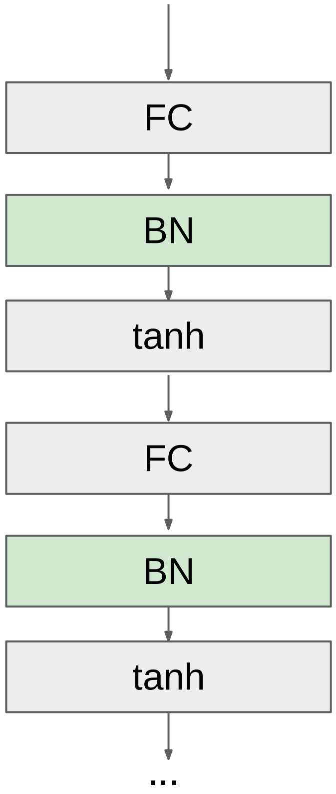


Обычно ставят **после** полносвязных или сверточных слоев и **перед** нелинейностю.

$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

Batch Normalization

[Ioffe and Szegedy, 2015]



- Сильно упрощает обучение глубоких сеетй!
- Стабилизирует расчет градиентов
- Делает возможным более высокий learning rates, и более быструю сходимость
- Сети более устойчивы к начальным значениям весов
- Регуляризирует обучение
- Нулевая стоимость на инференсе, объединяется со свертками!
- Ведет себя по разному на обучении и инференсе, причина многих ошибок!

Batch Normalization для СНС

Batch Normalization для **полносвязных** сетей

$$x: N \times D$$

Normalize

$$\mu,\sigma: 1 \times D$$

$$\gamma,\beta: 1 \times D$$

$$y = \gamma(x-\mu)/\sigma+\beta$$

Batch Normalization для **сверточных** сетей
(Spatial Batchnorm, BatchNorm2D)

$$x: N\times C\times H\times W$$

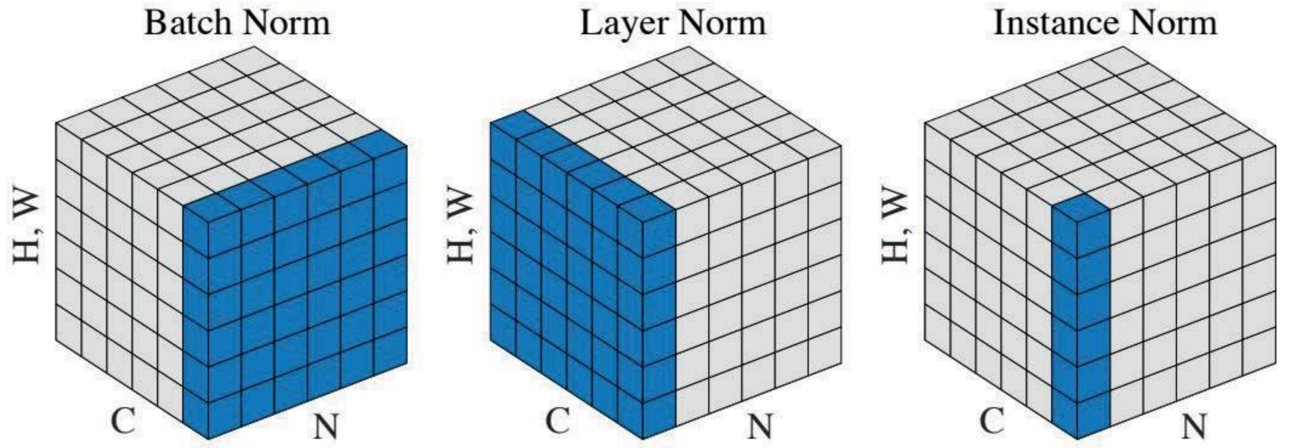
Normalize

$$\mu,\sigma: 1\times C\times 1\times 1$$

$$\gamma,\beta: 1\times C\times 1\times 1$$

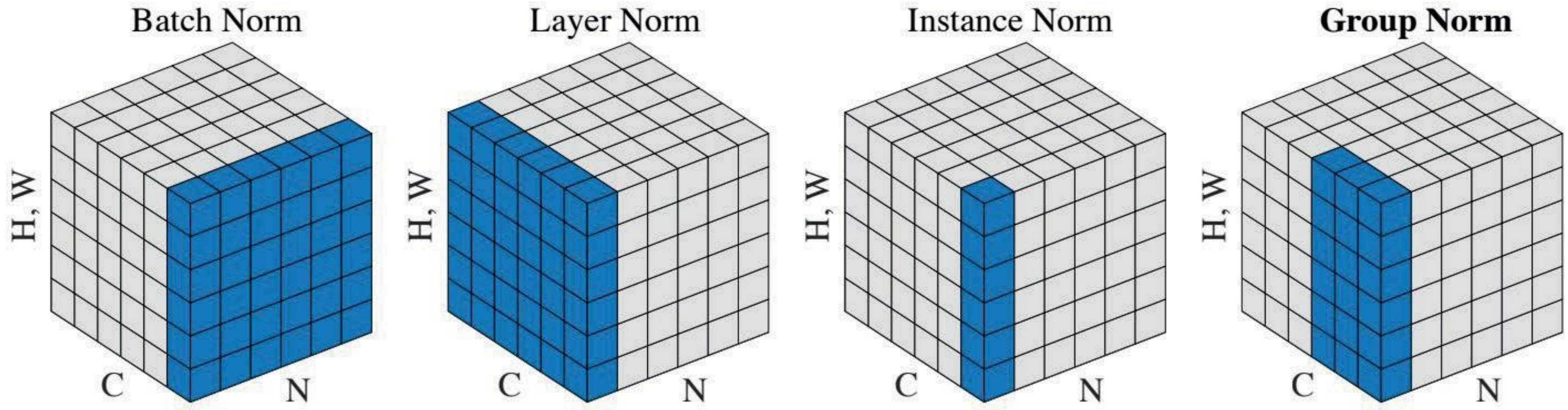
$$y = \gamma(x-\mu)/\sigma+\beta$$

Comparison of Normalization Layers



Wu and He, “Group Normalization”, ECCV 2018

Group Normalization



Wu and He, “Group Normalization”, ECCV 2018

Transfer learning

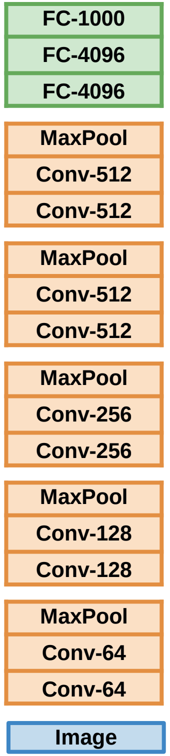
“Вам надо много данных, чтобы обучить вашу сеть”

BUSTED

Donahue et al, “DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition”, ICML 2014
Razavian et al, “CNN Features Off-the-Shelf: An Astounding Baseline for Recognition”, CVPR Workshops 2014

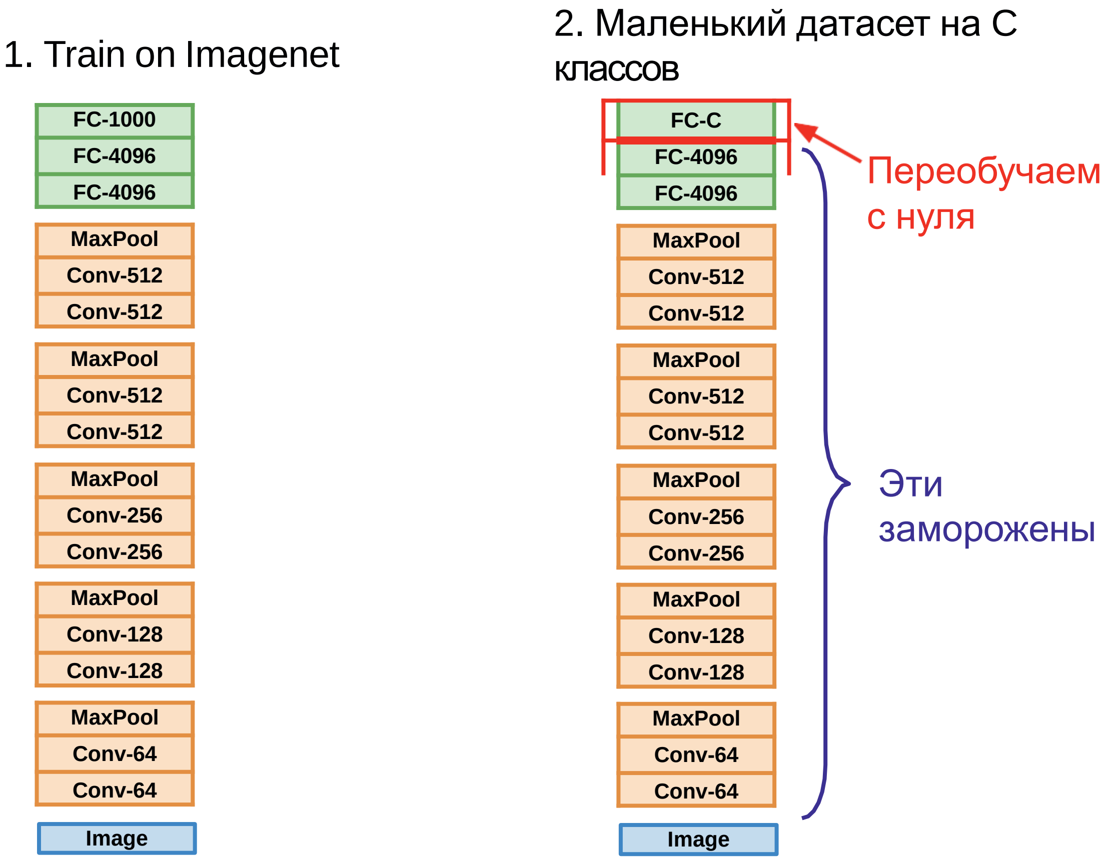
Transfer Learning для СНС

1. Учим на Imagenet



Transfer Learning для СНС

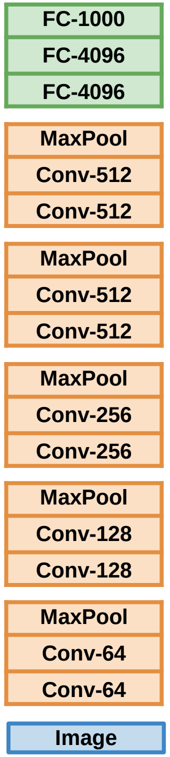
Donahue et al, “DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition”, ICML 2014
Razavian et al, “CNN Features Off-the-Shelf: An Astounding Baseline for Recognition”, CVPR Workshops 2014



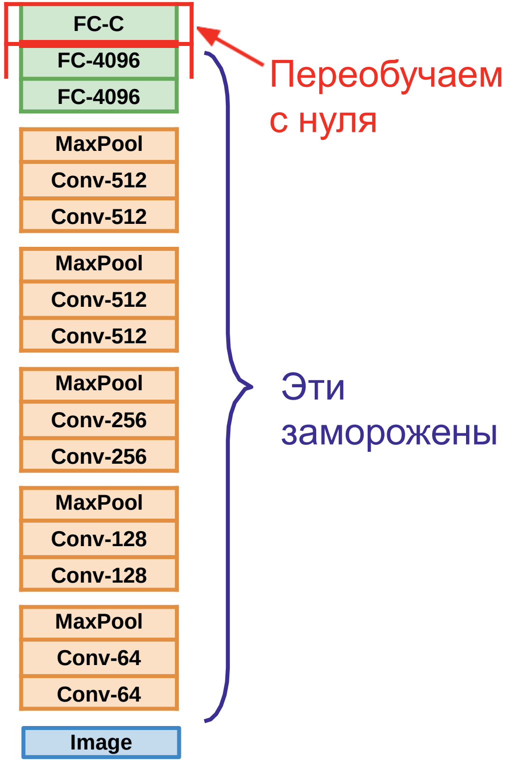
Transfer Learning для СНС

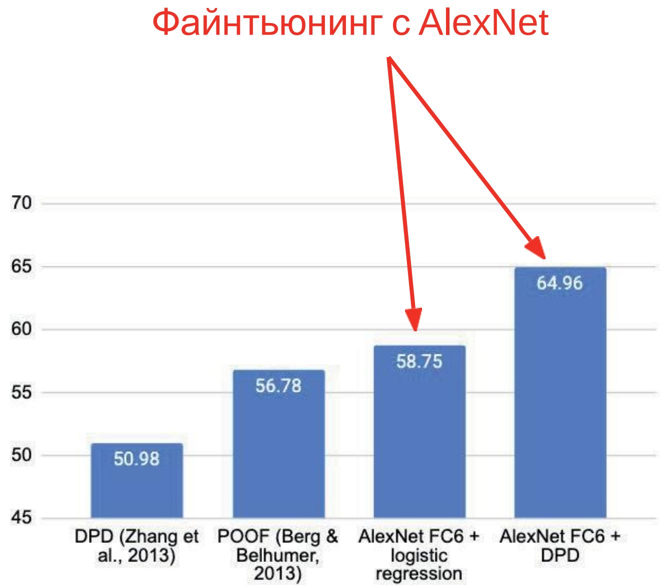
Donahue et al, “DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition”, ICML 2014
Razavian et al, “CNN Features Off-the-Shelf: An Astounding Baseline for Recognition”, CVPR Workshops 2014

1. Train on Imagenet



2. Маленький датасет на С классов





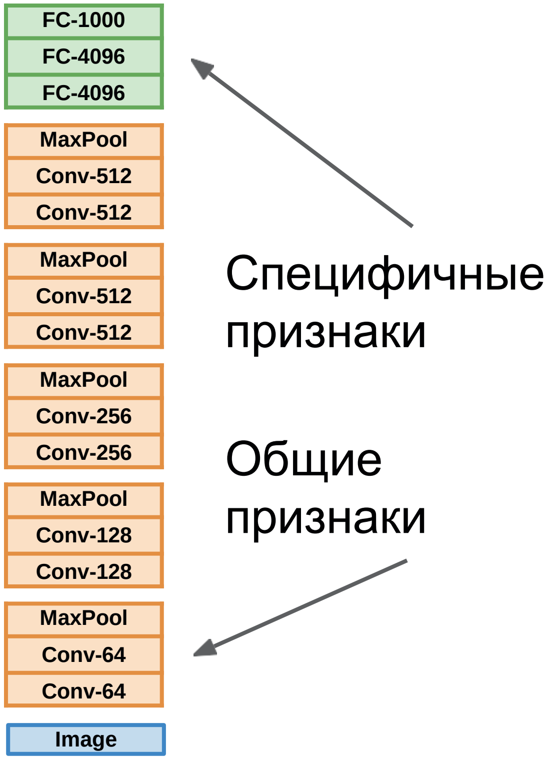
Donahue et al, “DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition”, ICML 2014

Transfer Learning для CHC

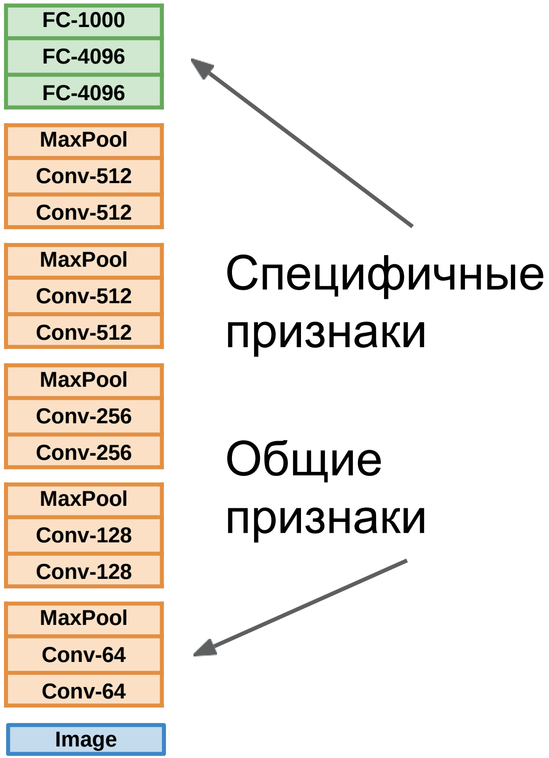
Donahue et al, “DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition”, ICML 2014
Razavian et al, “CNN Features Off-the-Shelf: An Astounding Baseline for Recognition”, CVPR Workshops 2014

1. Train on Imagenet

2. Маленький датасет на C классов



	Близкий датасет	**Отличный датасет**
Мало данных	?	?
Много данных	?	?

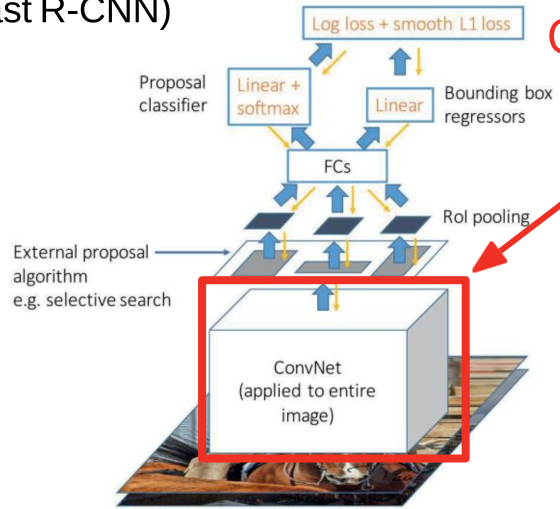


	Близкий датасет	**Отличный датасет**
Мало данных	Линейный классификатор поверх сети	Проблема… Попробовать линейный классификатор на ранних слоях
Много данных	Файнтьюн нескольких слоев	Файнтьюн большого количества слоев

Transfer learning везде…

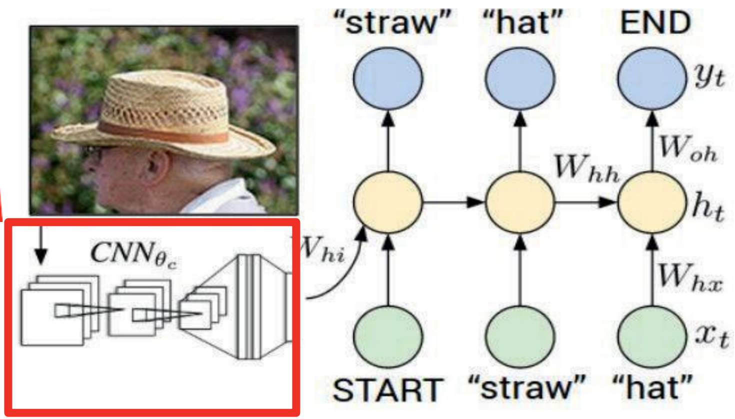
(Это норма, а не исключение)

Object **Detection**
(Fast R-CNN)



Предобученная СНС (ImageNet)

Image Captioning: CNN + RNN



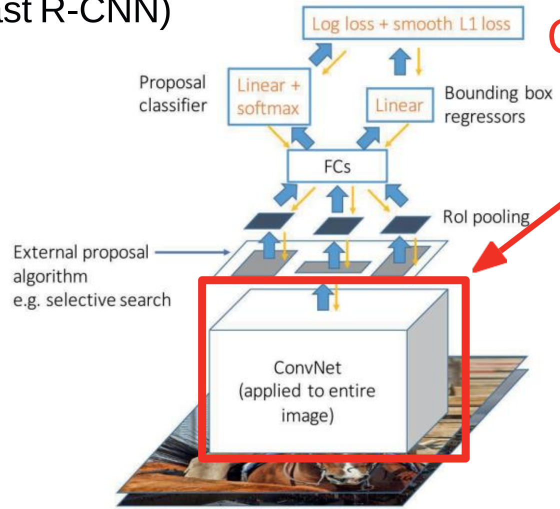
Girshick, "Fast R-CNN", ICCV 2015
Figure copyright Ross Girshick, 2015. Reproduced with permission.

Karpathy and Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions", CVPR 2015
Figure copyright IEEE, 2015. Reproduced for educational purposes.

Transfer learning везде…

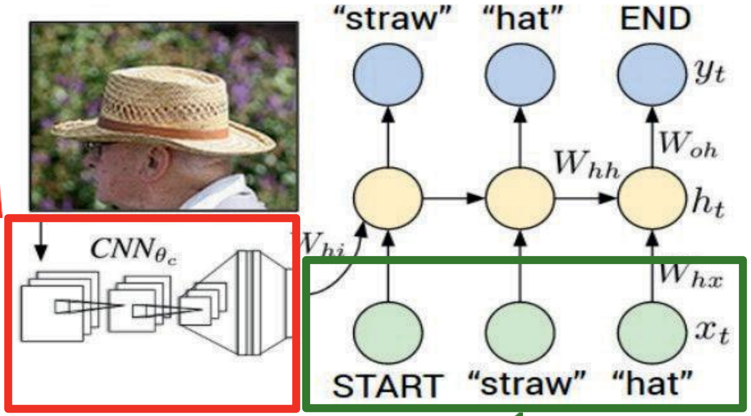
(Это норма, а не исключение)

Object **Detection**
(Fast R-CNN)



Предобученная
СНС (ImageNet)

Image Captioning: CNN + RNN

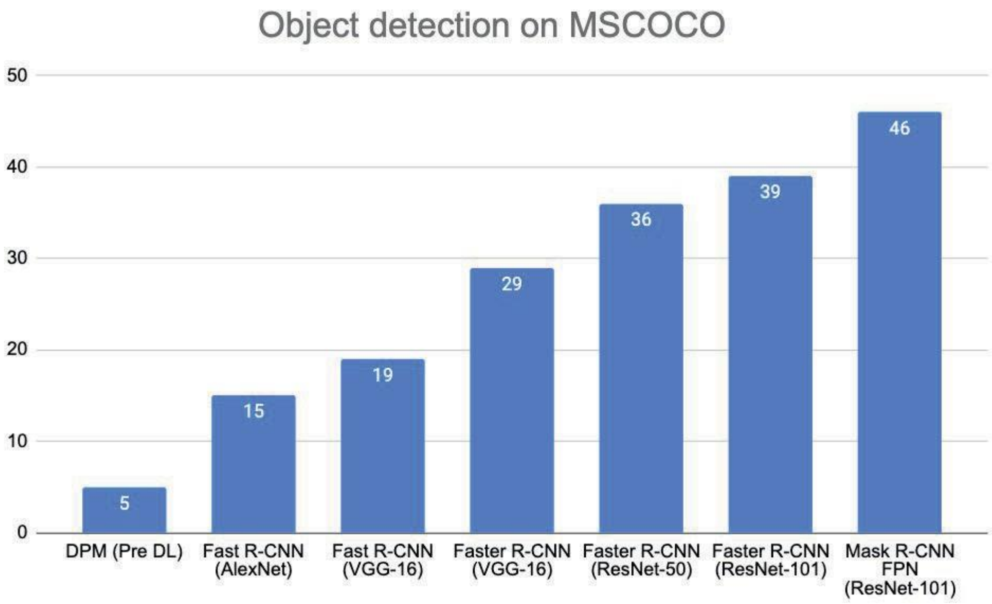


Обработка слов на основе
предобученного word2vec

Girshick, "Fast R-CNN", ICCV 2015

Karpathy and Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions", CVPR 2015

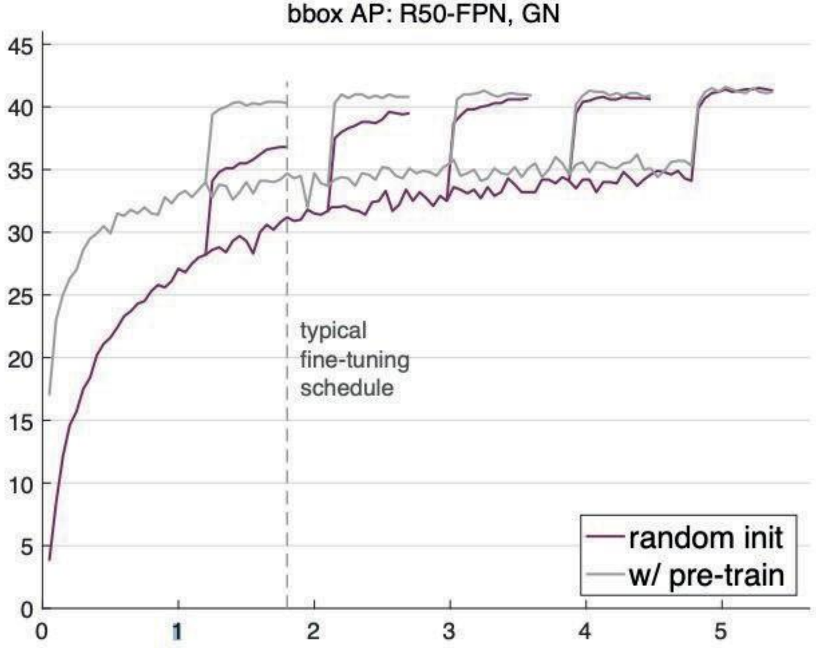
Transfer learning - Архитектура имеет значение



Girshick, “The Generalized R-CNN Framework for Object Detection”, ICCV 2017 Tutorial on Instance-Level Visual Recognition

Архитектуры разберем позже

Transfer learning везде…
Но возможно это не столь важно!



He et al, “Rethinking ImageNet Pre-training”, ICCV 2019
Figure copyright Kaiming He, 2019. Reproduced with permission.

Обучение с нуля (from scratch) работает практически также, как использование предобученной на ImageNet модели

Но в 2-3 раза дольше

Собрать побольше данных и обучить с нуля лучше, чем дообучить

На практике:

Transfer learning be like

Source: AI & Deep Learning Memes For Back-propagated Poets

На практике:

Что если у вас есть < ~1 млн. картинок?

1. Найти большой похожий датасет, научить на нем Большую Модель
2. Transfer learn на своем датасете

Используйте зоопарки моделей “Model Zoo” ищите нужную предобученную модель и используйте

TensorFlow: https://github.com/tensorflow/models
PyTorch: https://github.com/pytorch/vision

Итого

TLDRs

Мы разобрали:

- Функции активации (ReLU это хорошо)
- Подготовку данных (для картинок: вычитаем среднее)
- Инициализацию весов (Xavier/He)
- Batch Normalization (используйте!)
- Transfer learning (используйте при возможности!)

Далее:
Обучение НС, часть 2

- Политики обновления весов
- Learning rate политики
- Проверка градиентов
- Регуляризация (Dropout и т.д.)
- Контроль обучения
- Тестирование (Ансамбли и т.д.)
- Подбор гиперпараметров
- Transfer learning / fine-tuning – еще раз