

Глубокое обучение и вообще

Ульянкин Филипп

23 марта 2022 г.

Посиделка 5: Нейросети — конструктор LEGO

Agenda

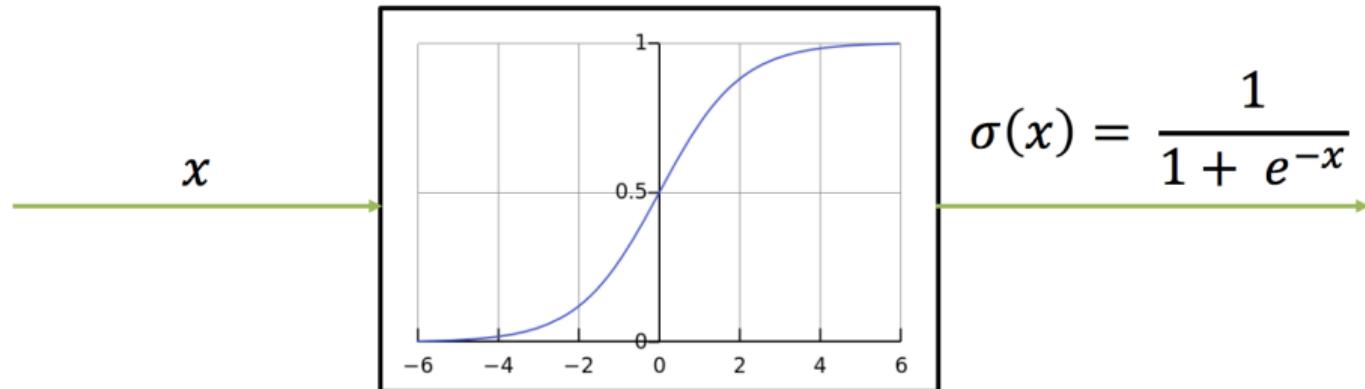
- Какими бывают функции активации
- Паралич нейронной сети и взрыв градиентов
- Инициализация весов
- Переобучение нейронных сетей
- Регуляризация: l_1 , l_2 , дропаут, ранняя остановка, их взаимосвязь
- Первая порция советов по обучению нейросетей

Какими бывают функции активации

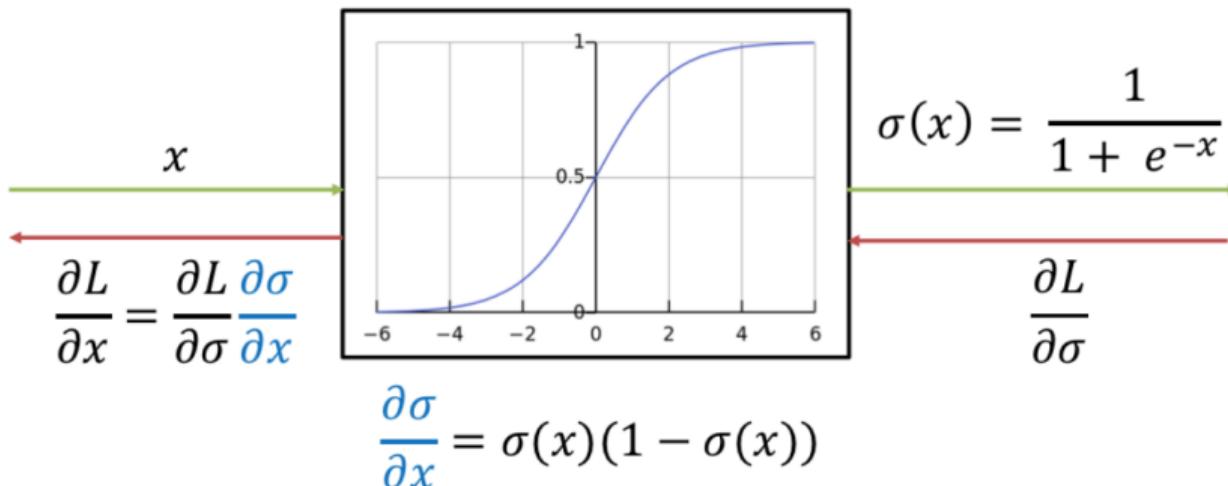


$$y = ?$$

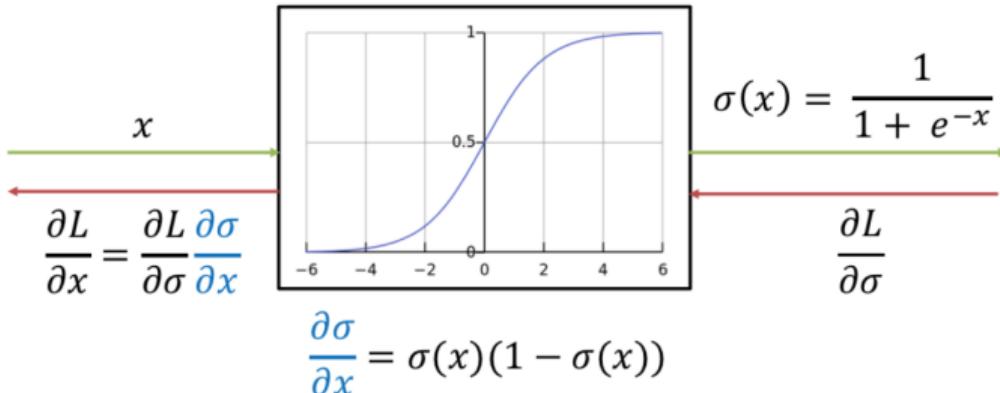
Сигмоида (sigmoid activation)



Сигмоида (sigmoid activation)



Сигмоида (sigmoid activation)



- Сигмоида была популярна как классическая функция активации, но у неё есть ряд проблем
- **Насыщенные нейроны зануляют градиенты, в глубоких сетях возможен паралич**

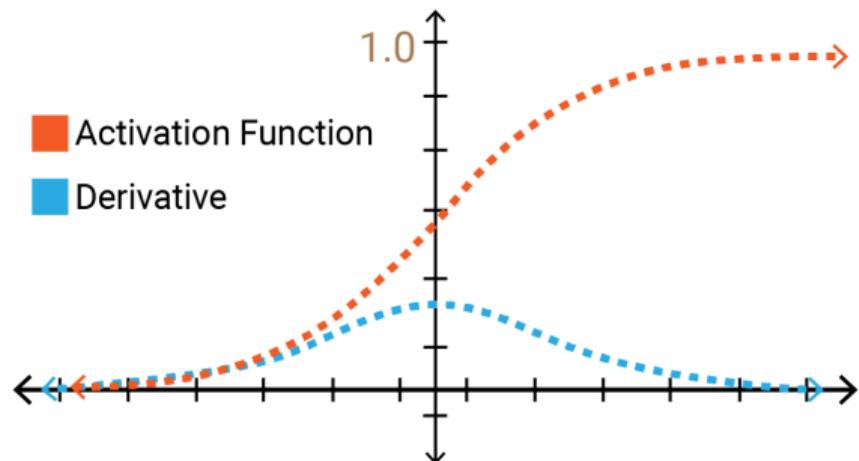
Затухание градиента (vanishing gradient problem)

- Изменение параметров в ходе обучения происходит на величину, которая не влияет на выход сети
- Проблема связана с очень маленькими градиентами при обновлении весов:

$$w_t = w_{t-1} - \gamma \cdot \nabla L(w_{t-1})$$

- При насыщении сети сигмоида убивает градиенты

$$f(t) = \frac{1}{1 + e^{-t}}$$

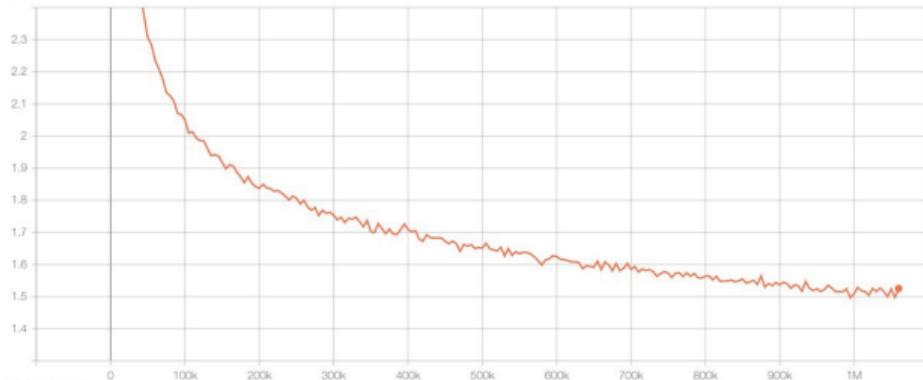


Затухание градиента (vanishing gradient problem)

- Изменение параметров в ходе обучения происходит на величину, которая не влияет на выход сети
- Проблема связана с очень маленькими градиентами при обновлении весов:

$$w_t = w_{t-1} - \gamma \cdot \nabla L(w_{t-1})$$

- При насыщении сети сигмоида убивает градиенты



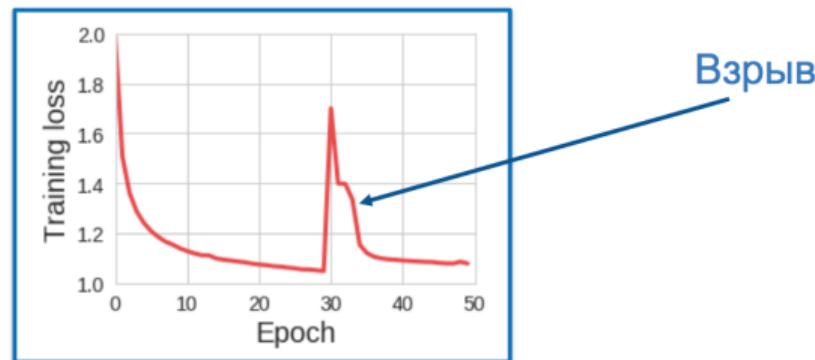
Толи обучение сошлося, толи веса просто больше не обновляются ...

Паралич сети

- В случае сигмоиды $\sigma'(x) = \sigma(x) \cdot (1 - \sigma(x))$
- Сигмода принимает значения на отрезке $[0; 1]$, значит максимальное значение её производной это $1/4$
- Если сеть очень глубокая, происходит **затухание градиента**
- Градиент затухает экспоненциально \Rightarrow сходимость замедляется, более ранние веса обновляются дольше, более глубокие веса быстрее \Rightarrow значение градиента становится ещё меньше \Rightarrow наступает **паралич сети**
- В сетях с небольшим числом слоёв этот эффект незаметен

Взрыв градиента (exploding gradient problem)

- Изменение параметров в ходе обучения происходит на очень большую величину и обучение деградирует
- Проблема связана с очень большими градиентами при обновлении весов:

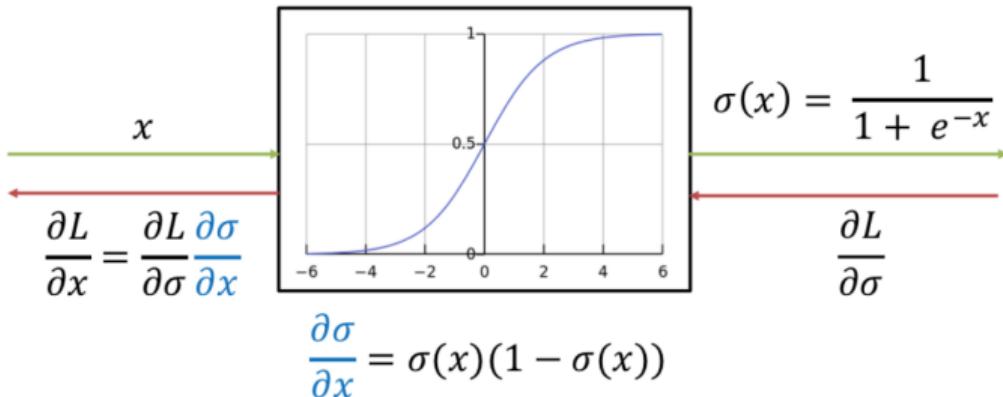


$$w_t = w_{t-1} - \gamma \cdot \nabla L(w_{t-1})$$

Взрыв и затухание градиента

- Затухание градиента связано не только с сигмоидой
- Взрыв градиента также встречается довольно часто
- ⇒ грамотные инициализация и оптимизация
- ⇒ разработка новых архитектур и специальных слоёв

Сигмоида (sigmoid activation)



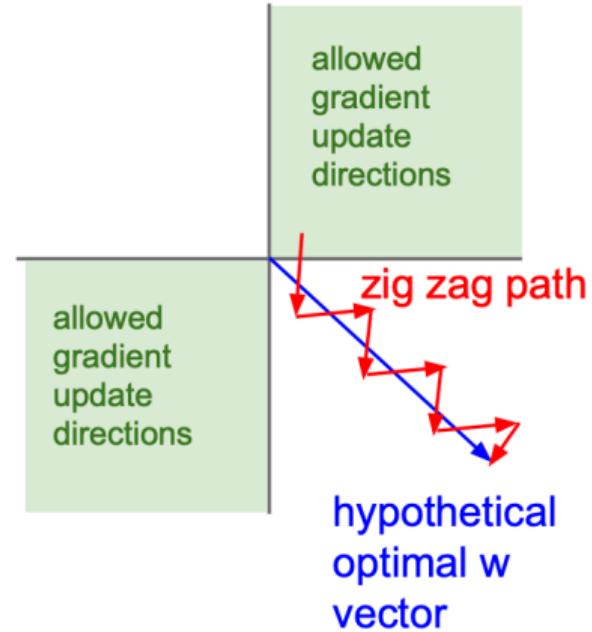
- Насыщенные нейроны зануляют градиенты, в глубоких сетях возможен паралич
- Не центрирована относительно нуля
- Что мы можем сказать о градиентах нейрона с сигмоидой?

Центрирование относительно нуля

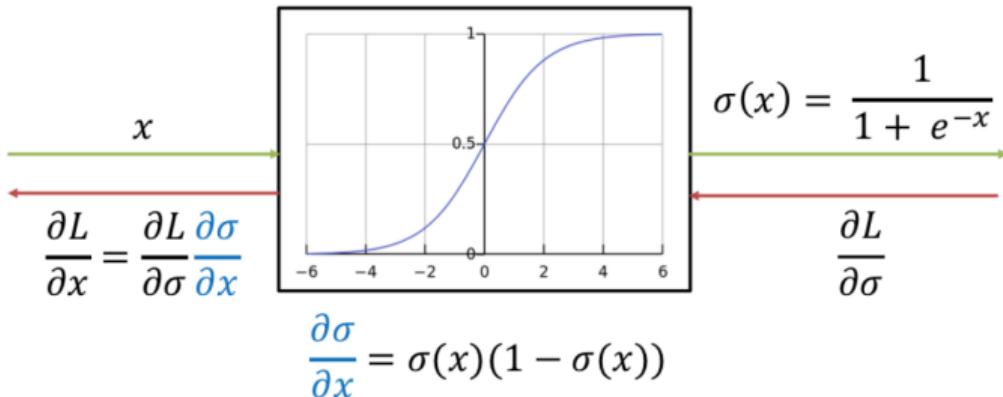
- Выход слоя мы обычно находим как

$$o_i = \sigma(h_i)$$

- он всегда положительный, значит градиент по весам, идущим на вход в текущий нейрон тоже положительные
- \Rightarrow все веса обновляются в одинаковом направлении
- \Rightarrow сходимость идёт медленнее, причём зиг-загами

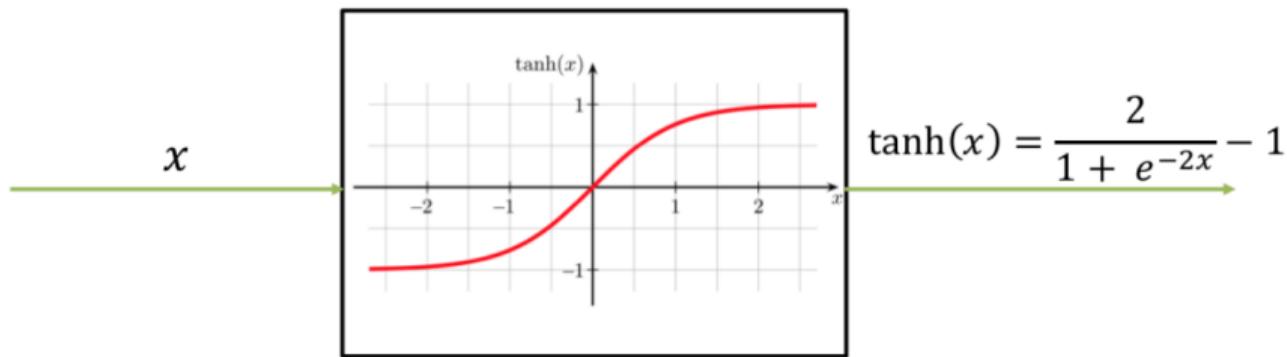


Сигмоида (sigmoid activation)



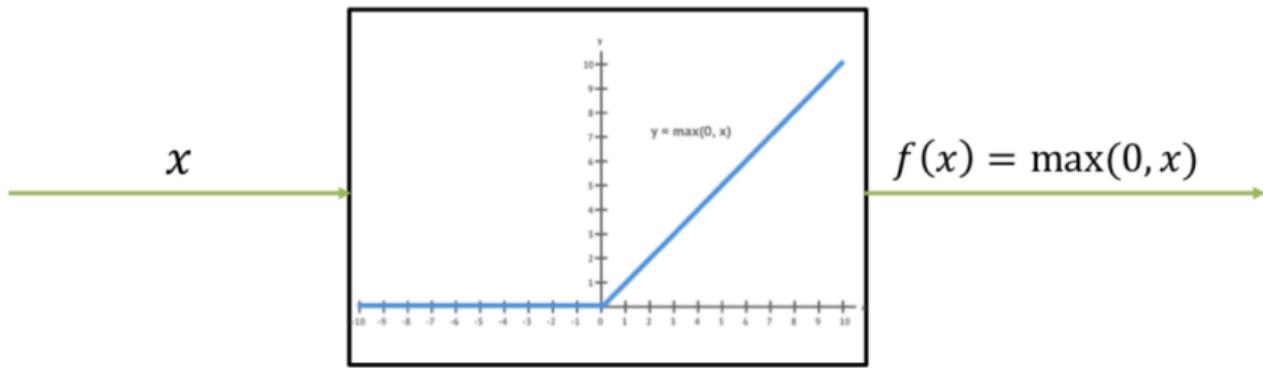
- Насыщенные нейроны зануляют градиенты, в глубоких сетях возможен паралич
- Не центрирована относительно нуля
- Вычислять e^x дорого

Гиперболический тангенс (tanh activation)



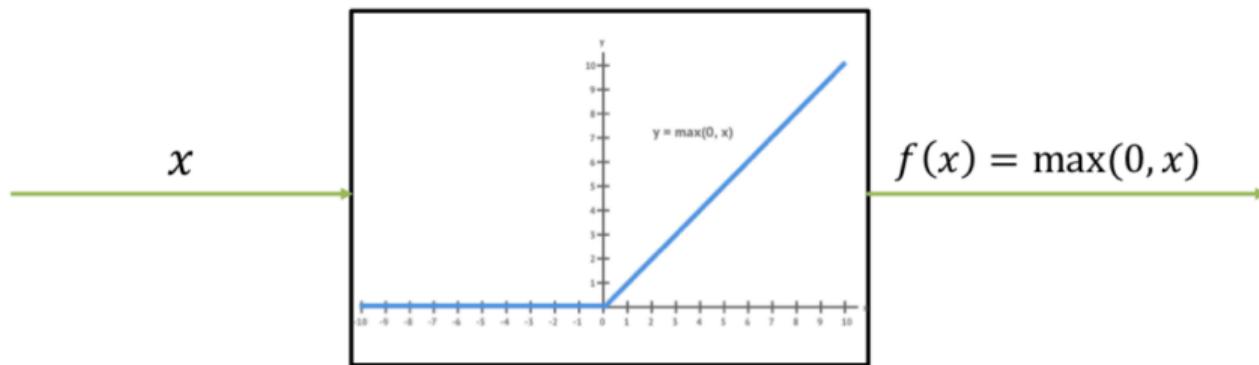
- Центрирован относительно нуля
- Всё ещё похож на сигмоиду
- $f'(x) = 1 - f(x)^2 \Rightarrow$ затухание градиента

Rectifier Linear Unit activation, ReLU (2012)



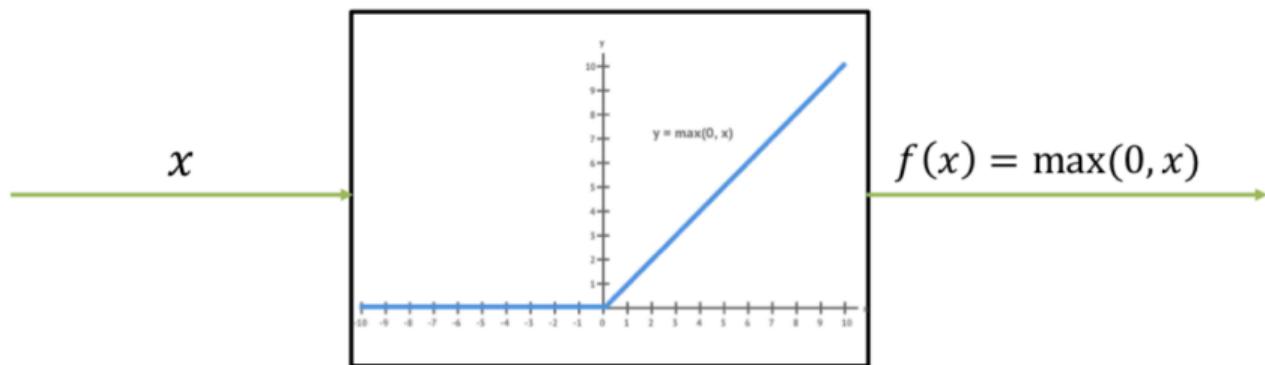
- Быстро вычисляется
- Градиенты не угасают при $x > 0$
- На практике ускоряет сходимость

ReLU (2012)



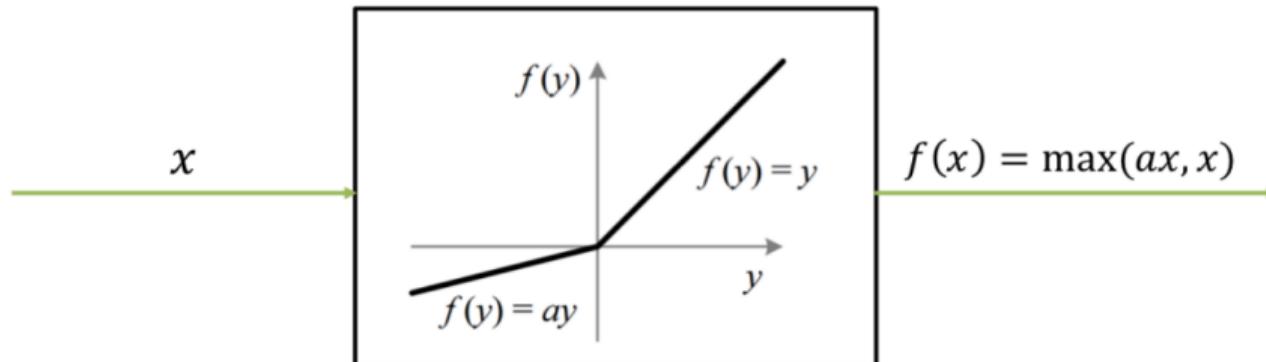
- Сетка может умереть, если активация занулится на всех нейронах
- Не центрирован относительно нуля

Зануление ReLU



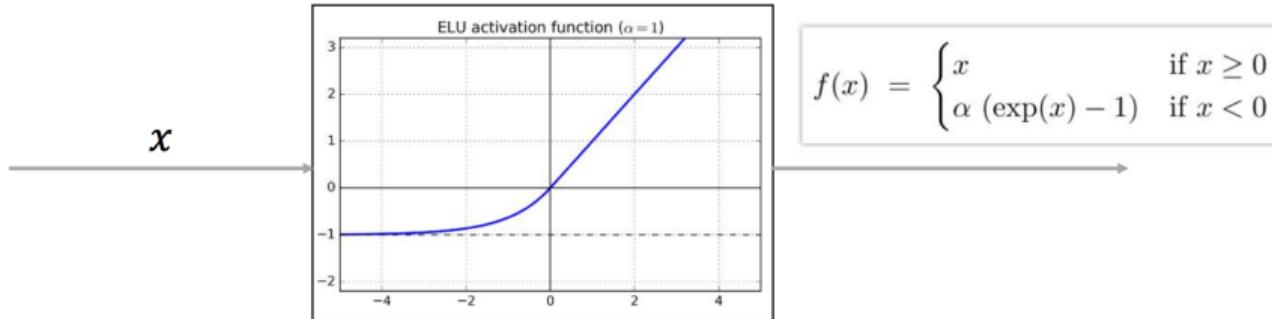
- $f(x) = \max(0, w_0 + w_1 \cdot h_1 + \dots + w_k \cdot h_k)$
- Если w_0 инициализировано большим отрицательным числом, нейрон сразу умирает \Rightarrow надо аккуратно инициализировать веса

Leaky ReLU activation (2013)



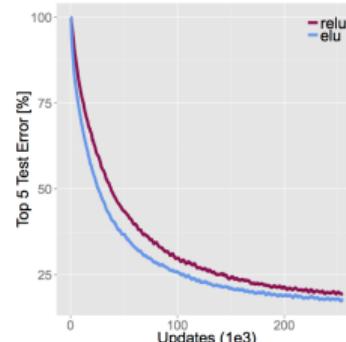
- Как ReLU, но не умирает, всё ещё легко считается
- Производная может быть любого знака
- Важно, чтобы $a \neq 1$, иначе линейность
- Не центрирован относительно нуля

Exponential Linear Units activation, ELU (2015)



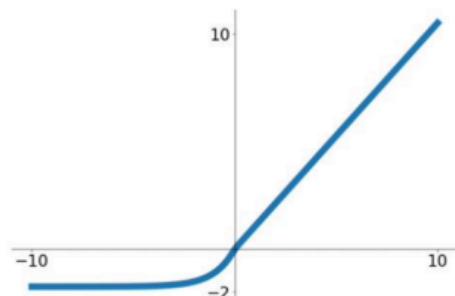
- Примерно центрирован в нуле (в контексте математического ожидания)
- Сходимость быстрее ReLU
- На практике ускоряет сходимость

<https://arxiv.org/pdf/1511.07289.pdf>



(b) Top-5 test error

Scaled Exponential Linear Units activation, SELU (2017)



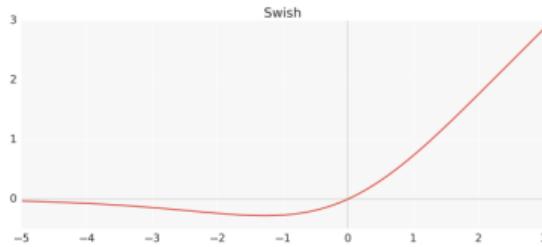
$$f(x) = \begin{cases} \lambda x & \text{if } x > 0 \\ \lambda\alpha(e^x - 1) & \text{otherwise} \end{cases}$$

$\alpha = 1.6733, \lambda = 1.0507$

- Нормализованная версия ELU, лучше работает для глубоких сетей
- Обладает свойством самонормализации
- Можно учить сети без нормализации по батчам (будем обсуждать позже)

<https://arxiv.org/abs/1706.02515>

Swish (2017)



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

параметр β обучается

- В 2017 году Google Brain хитрым автоматическим поиском на основе RNN нашла функцию активации Swish, работающую лучше ReLU

Function	RN	WRN	DN
ReLU [$\max(x, 0)$]	93.8	95.3	94.8
$x \cdot \sigma(\beta x)$	94.5	95.5	94.9
$\max(x, \sigma(x))$	94.3	95.3	94.8
$\cos(x) - x$	94.1	94.8	94.6
$\min(x, \sin(x))$	94.0	95.1	94.4
$(\tan^{-1}(x))^2 - x$	93.9	94.7	94.9
$\max(x, \tanh(x))$	93.9	94.2	94.5
$\text{sinc}(x) + x$	91.5	92.1	92.0
$x \cdot (\sinh^{-1}(x))^2$	85.1	92.1	91.1

Table 1: CIFAR-10 accuracy.

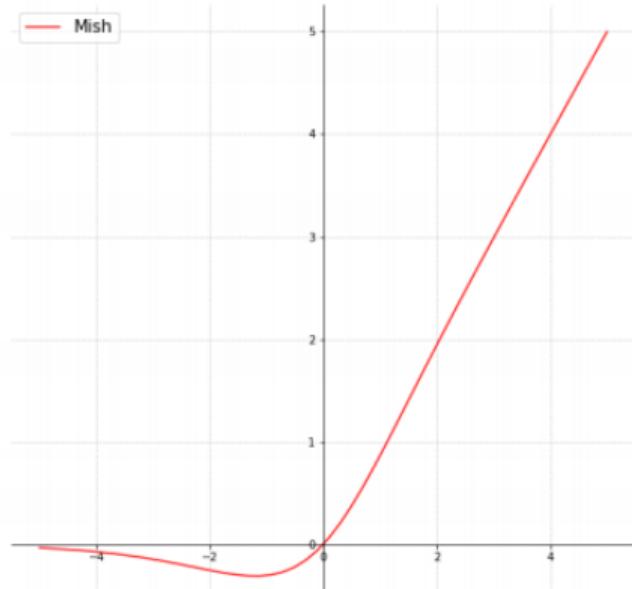
Function	RN	WRN	DN
ReLU [$\max(x, 0)$]	74.2	77.8	83.7
$x \cdot \sigma(\beta x)$	75.1	78.0	83.9
$\max(x, \sigma(x))$	74.8	78.6	84.2
$\cos(x) - x$	75.2	76.6	81.8
$\min(x, \sin(x))$	73.4	77.1	74.3
$(\tan^{-1}(x))^2 - x$	75.2	76.7	83.1
$\max(x, \tanh(x))$	74.8	76.0	78.6
$\text{sinc}(x) + x$	66.1	68.3	67.9
$x \cdot (\sinh^{-1}(x))^2$	52.8	70.6	68.1

Table 2: CIFAR-100 accuracy.

<https://arxiv.org/abs/1710.05941>

<https://krutikabapat.github.io/Swish-Vs-Mish-Latest-Activation-Functions/>

Mish (2019)



Похожим образом получили функции
активации Mish

$$f(x) = x \cdot \tanh(\ln(1 + e^x))$$

<https://arxiv.org/pdf/1908.08681.pdf>

<https://krutikabapat.github.io/Swish-Vs-Mish-Latest-Activation-Functions/>

Activate or Not, ACON (2020)

- Swish -- гладкий вариант ReLU с гейтом
- Функция сама решает, нужна слою активация или нет
- Можно обобщить этот подход и придумать более интересные функции потерь, которые дадут улучшение

<https://arxiv.org/pdf/2009.04759.pdf>

TLDR: что на практике

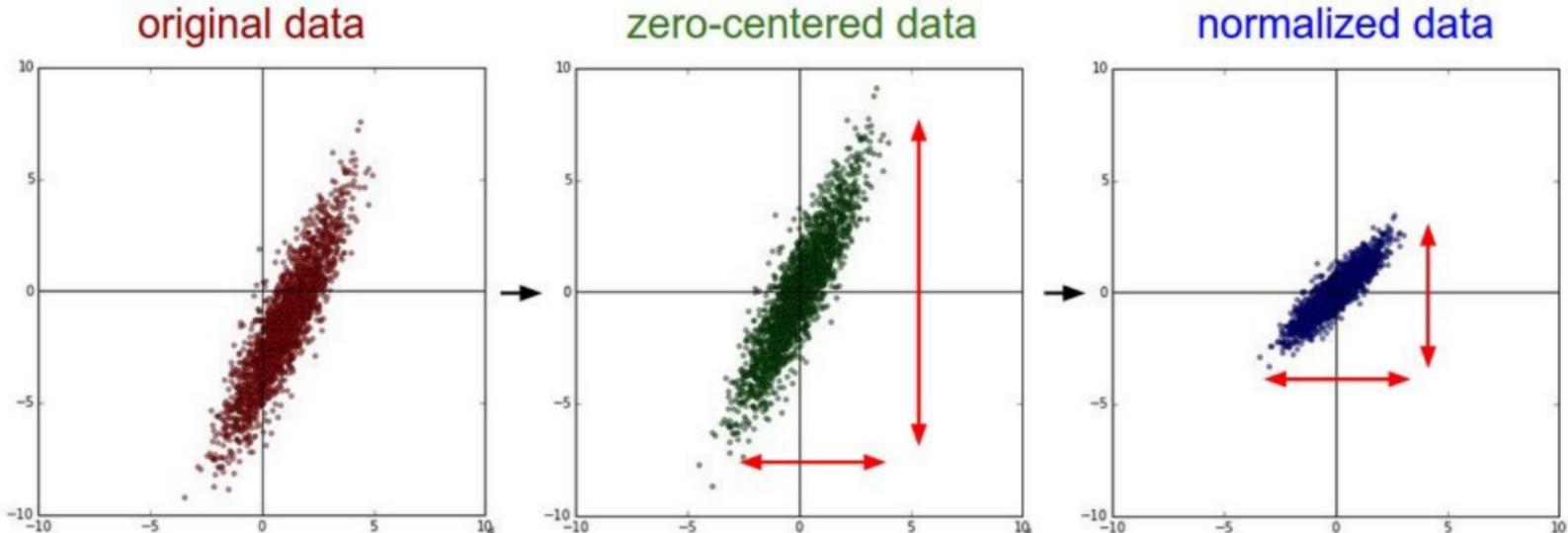
- Начните с ReLU, аккуратно инициализируйте веса и настраивайте скорость обучения
- Попробуйте Leaky ReLU / ELU / SELU / Swish / Mish, если есть время на эксперименты, чтобы выжать максимум
- Не используйте сигмоиду и tanh

Краткий обзор функций активаций: <https://arxiv.org/pdf/1804.02763.pdf>

Предобработка данных



Предобработка данных

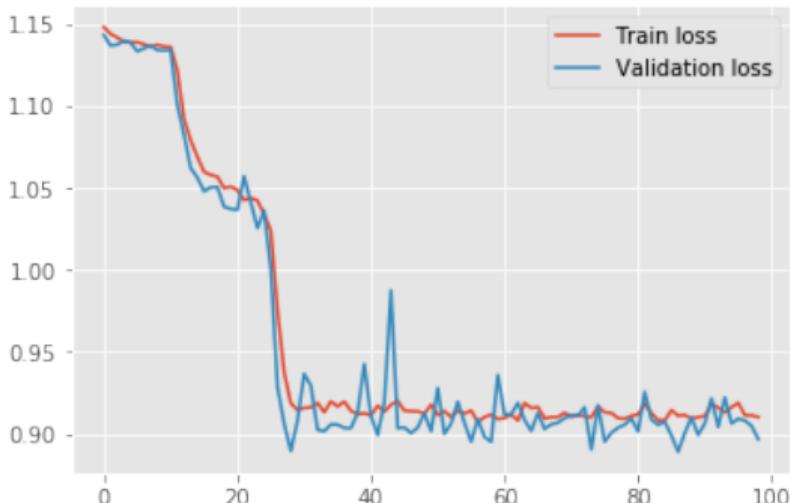


```
X -= np.mean(X, axis = 0)
```

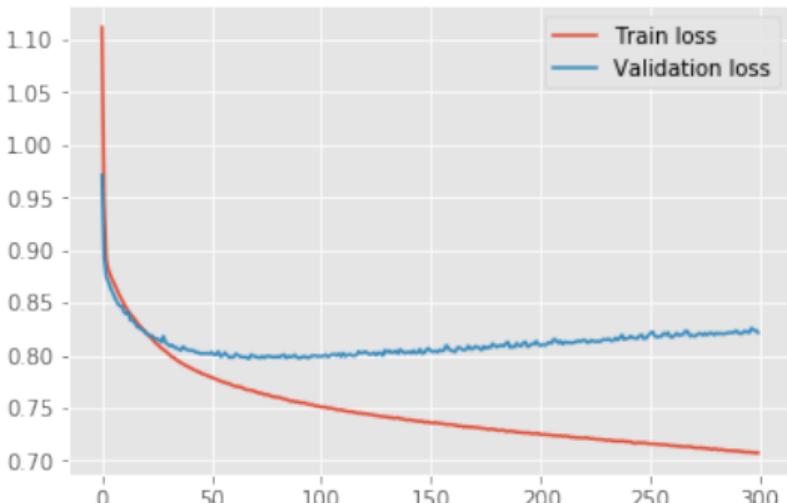
```
X /= np.std(X, axis = 0)
```

Предобработка табличных данных

Без нормализации

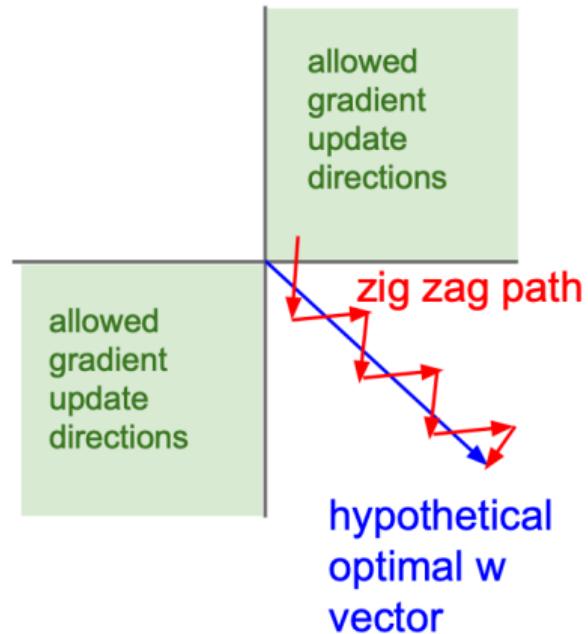


С нормализацией

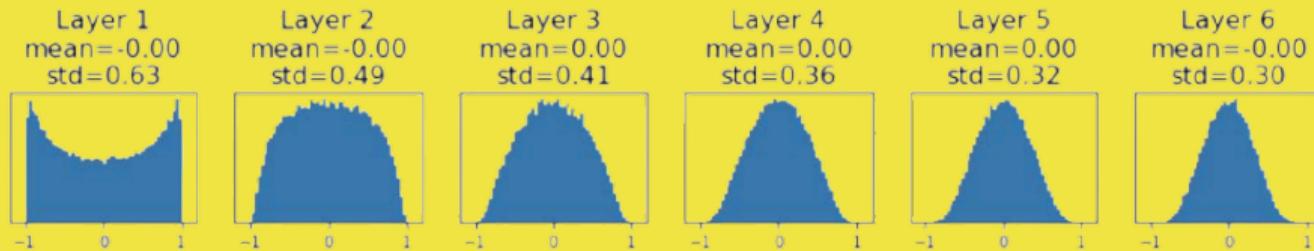


Зачем нормализация картинкам?

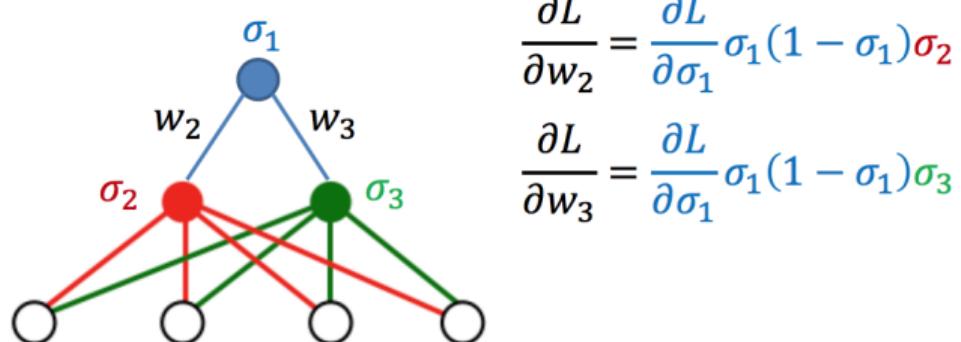
- Что происходит, когда все входы в нейрон положительные?
- Все градиенты либо положительные либо отрицательные
- Картинки центрируют для того, чтобы были отрицательные входы
- Картинки нормируют, чтобы инициализированным в окрестности нуля весам легче было сходиться



Инициализация весов



Инициализация весов

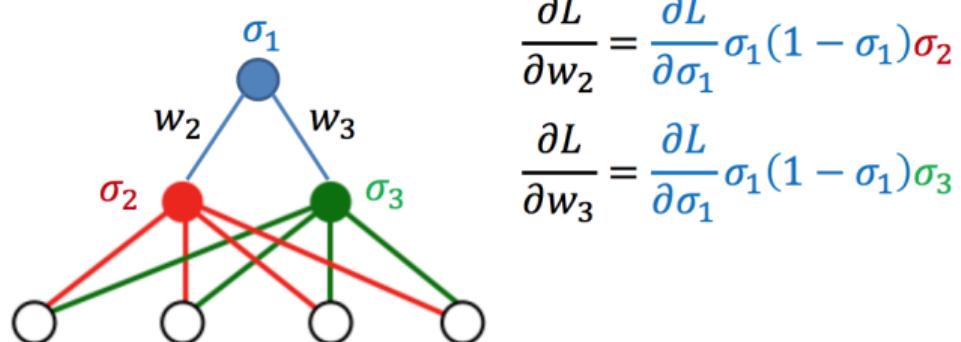


$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial \sigma_1} \sigma_1 (1 - \sigma_1) \color{red}{\sigma_2}$$

$$\frac{\partial L}{\partial w_3} = \frac{\partial L}{\partial \sigma_1} \sigma_1 (1 - \sigma_1) \color{red}{\sigma_3}$$

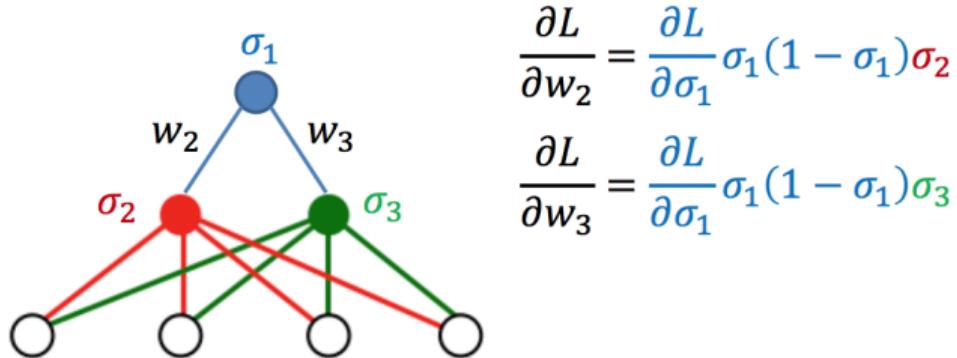
- Что будет, если инициализировать веса нулями?

Инициализация весов



- Что будет, если инициализировать веса нулями?
- σ_2 и σ_3 будут обновляться одинаково

Инициализация весов



- Хочется уничтожить симметрию
- Можно инициализировать маленькими рандомными числами из какого-то распределения (нормальное, равномерное)
- Это будет плохо работать для глубоких сетей

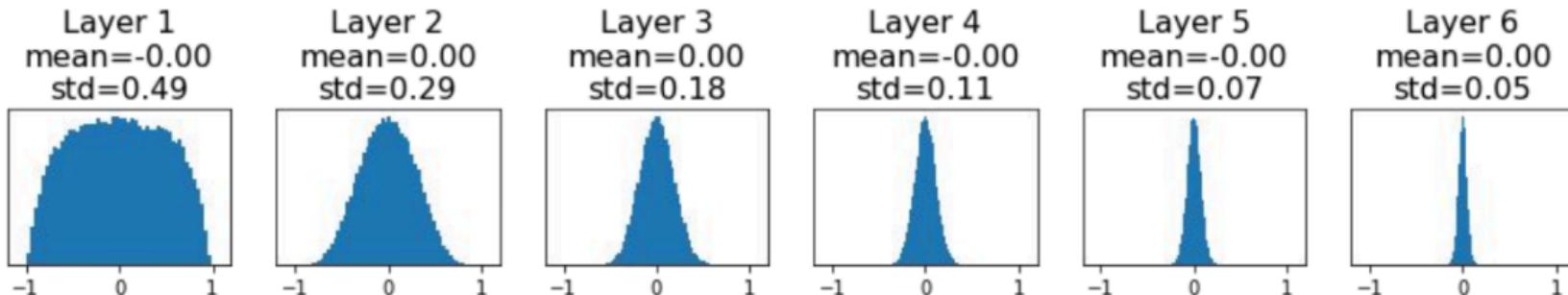
Инициализация весов

- Прямой проход через 6 слоёв нейросети
- Что произойдёт с активацией к последнему слою?

```
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)
```

Инициализация весов

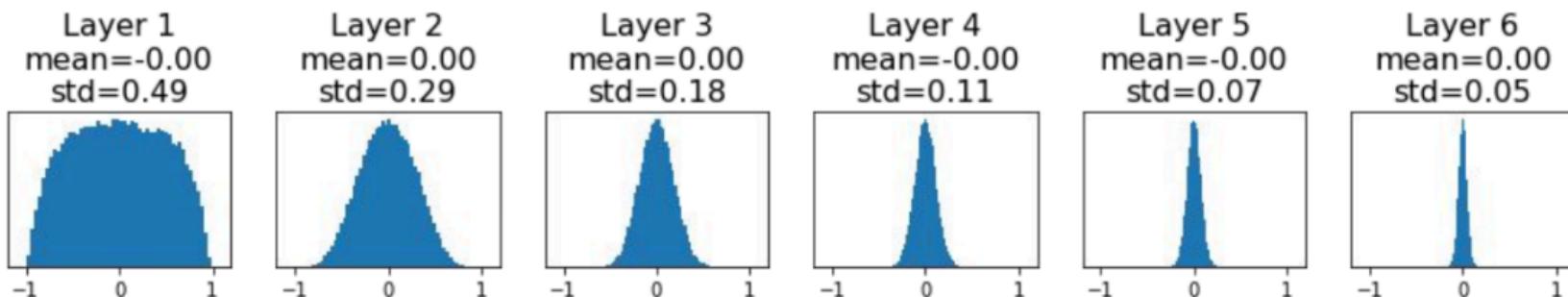
- Прямой проход через 6 слоёв нейросети
 - Что произойдёт с активацией к последнему слою?
 - Активация постепенно стягивается к нулю
- ```
dims = [4096] * 7
hs = []
x = np.random.randn(16, dims[0])
for Din, Dout in zip(dims[: -1], dims[1:]):
 W = 0.01 * np.random.randn(Din, Dout)
 x = np.tanh(x.dot(W))
 hs.append(x)
```



# Инициализация весов

- Активация постепенно стягивается к нулю
- А что будет происходить с градиентами?

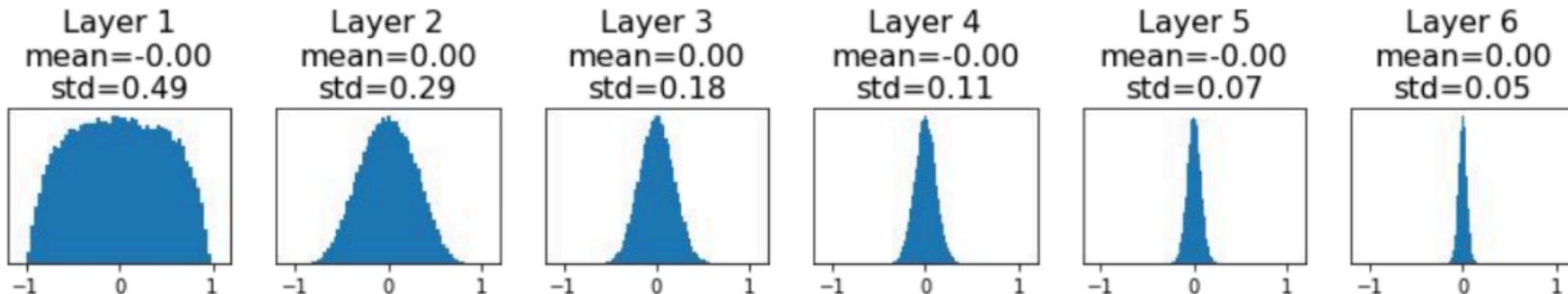
```
dims = [4096] * 7
hs = []
x = np.random.randn(16, dims[0])
for Din, Dout in zip(dims[: -1], dims[1 :]):
 W = 0.01 * np.random.randn(Din, Dout)
 x = np.tanh(x.dot(W))
 hs.append(x)
```



# Инициализация весов

- Активация постепенно стягивается к нулю
- А что будет происходить с градиентами?
- Все будут нулевыми :(

```
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)
```



# Инициализация весов

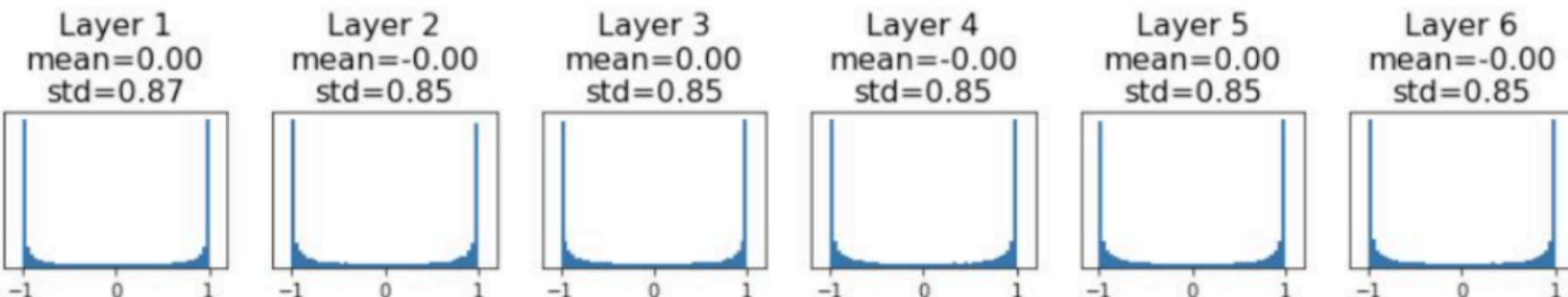
- Увеличим стандартное отклонение в инициализации с 0.01 до 0.05
- Что произойдёт с активацией к последнему слою?

```
dims = [4096] * 7
hs = []
x = np.random.randn(16, dims[0])
for Din, Dout in zip(dims[: -1], dims[1 :]):
 W = 0.05 * np.random.randn(Din, Dout)
 x = np.tanh(x.dot(W))
 hs.append(x)
```

# Инициализация весов

- Постепенно происходит насыщение, градиенты опять занулятся

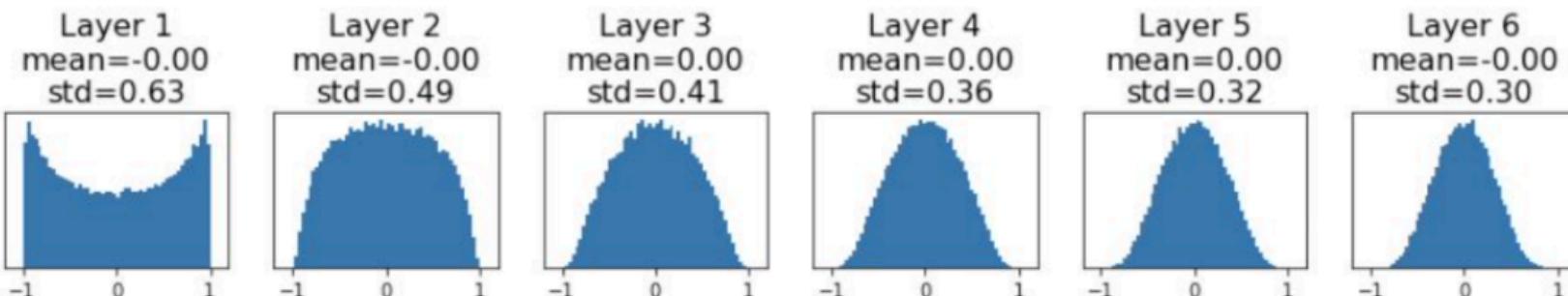
```
dims = [4096] * 7
hs = []
x = np.random.randn(16, dims[0])
for Din, Dout in zip(dims[: -1], dims[1 :]):
 W = 0.05 * np.random.randn(Din, Dout)
 x = np.tanh(x.dot(W))
 hs.append(x)
```



# Инициализация весов

- Распределение активаций не меняется, с градиентами всё хорошо

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



# Инициализация весов

- Посмотрим на выход нейрона перед активацией:

$$h_i = \sum_{i=1}^{n_{in}} w_i x_i$$

- Дисперсия  $h_i$  выражается через дисперсии  $x$  и  $w$
- Будем считать, что веса  $w_1, \dots, w_k \sim iid$ , наблюдения  $x_1, \dots, x_n \sim iid$ , а ещё  $x_i$  и  $w_i$  независимы между собой

## Инициализация весов (симметричная активация)

$$\begin{aligned}\text{Var}(h_i) &= \text{Var}\left(\sum_{i=1}^{n_{in}} w_i x_i\right) = \sum_{i=1}^{n_{in}} \text{Var}(w_i x_i) = \\ &= \sum_{i=1}^{n_{in}} [\mathbb{E}(x_i)]^2 \cdot \text{Var}(w_i) + [\mathbb{E}(w_i)]^2 \cdot \text{Var}(x_i) + \text{Var}(x_i) \cdot \text{Var}(w_i) =\end{aligned}$$

[https://en.wikipedia.org/wiki/Variance#Product\\_of\\_independent\\_variables](https://en.wikipedia.org/wiki/Variance#Product_of_independent_variables)

## Инициализация весов (симметричная активация)

$$\begin{aligned}\text{Var}(h_i) &= \text{Var}\left(\sum_{i=1}^{n_{in}} w_i x_i\right) = \sum_{i=1}^{n_{in}} \text{Var}(w_i x_i) = \\ &= \sum_{i=1}^{n_{in}} [\mathbb{E}(x_i)]^2 \cdot \text{Var}(w_i) + [\mathbb{E}(w_i)]^2 \cdot \text{Var}(x_i) + \text{Var}(x_i) \cdot \text{Var}(w_i) =\end{aligned}$$

- Если функция активации симметричная, тогда  $\mathbb{E}(x_i) = 0$ . Будем инициализировать веса с нулевым средним, тогда  $\mathbb{E}(w_i) = 0$ .

## Инициализация весов (симметричная активация)

$$\begin{aligned}\text{Var}(h_i) &= \text{Var}\left(\sum_{i=1}^{n_{in}} w_i x_i\right) = \sum_{i=1}^{n_{in}} \text{Var}(w_i x_i) = \\ &= \sum_{i=1}^{n_{in}} [\mathbb{E}(x_i)]^2 \cdot \text{Var}(w_i) + [\mathbb{E}(w_i)]^2 \cdot \text{Var}(x_i) + \text{Var}(x_i) \cdot \text{Var}(w_i) = \\ &= \sum_{i=1}^{n_{in}} \text{Var}(x_i) \cdot \text{Var}(w_i)\end{aligned}$$

- Предполагаем, что  $x_i$  инициализируются из одного распределения,  $w_i$  сами инициализируем из одного распределения.

## Инициализация весов (симметричная активация)

$$\begin{aligned}\text{Var}(h_i) &= \text{Var}\left(\sum_{i=1}^{n_{in}} w_i x_i\right) = \sum_{i=1}^{n_{in}} \text{Var}(w_i x_i) = \\ &= \sum_{i=1}^{n_{in}} [\mathbb{E}(x_i)]^2 \cdot \text{Var}(w_i) + [\mathbb{E}(w_i)]^2 \cdot \text{Var}(x_i) + \text{Var}(x_i) \cdot \text{Var}(w_i) = \\ &= \sum_{i=1}^{n_{in}} \text{Var}(x_i) \cdot \text{Var}(w_i) = \text{Var}(x) \cdot [n_{in} \cdot \text{Var}(w)]\end{aligned}$$

- Хотим, чтобы зелёная штука была равна единице, тогда у потока будет всегда постоянная дисперсия, совпадающая с  $\text{Var}(x)$ .

## Инициализация весов (симметричная активация)

$$\begin{aligned}\mathbf{Var}(h_i) &= \mathbf{Var}\left(\sum_{i=1}^{n_{in}} w_i x_i\right) = \sum_{i=1}^{n_{in}} \mathbf{Var}(w_i x_i) = \\ &= \sum_{i=1}^{n_{in}} [\mathbf{E}(x_i)]^2 \cdot \mathbf{Var}(w_i) + [\mathbf{E}(w_i)]^2 \cdot \mathbf{Var}(x_i) + \mathbf{Var}(x_i) \cdot \mathbf{Var}(w_i) = \\ &= \sum_{i=1}^{n_{in}} \mathbf{Var}(x_i) \cdot \mathbf{Var}(w_i) = \mathbf{Var}(x) \cdot \underbrace{[n_{in} \cdot \mathbf{Var}(w)]}_{=1}\end{aligned}$$

# Плохая инициализация весов

Пусть

$$w_i \sim U \left[ -\frac{1}{\sqrt{n_{in}}}; \frac{1}{\sqrt{n_{in}}} \right],$$

тогда

$$\text{Var}(w_i) = \frac{1}{12} \cdot \left( \frac{1}{\sqrt{n_{in}}} + \frac{1}{\sqrt{n_{in}}} \right)^2 = \frac{1}{3n_{in}} \Rightarrow \text{Var}(h_i) = \frac{1}{3}$$

Получаем затухание!

## Инициализация Ксавье/Глорота (2010)

Пусть

$$w_i \sim U \left[ -\frac{\sqrt{3}}{\sqrt{n_{in}}}, \frac{\sqrt{3}}{\sqrt{n_{in}}} \right],$$

тогда

$$\text{Var}(w_i) = \frac{1}{12} \cdot \left( \frac{\sqrt{3}}{\sqrt{n_{in}}} + \frac{\sqrt{3}}{\sqrt{n_{in}}} \right)^2 = \frac{1}{n_{in}} \Rightarrow \text{Var}(h_i) = 1$$

## Инициализация Ксавье/Глорота (2010)

Пусть

$$w_i \sim U \left[ -\frac{\sqrt{3}}{\sqrt{n_{in}}}; \frac{\sqrt{3}}{\sqrt{n_{in}}} \right],$$

тогда

$$\text{Var}(w_i) = \frac{1}{12} \cdot \left( \frac{\sqrt{3}}{\sqrt{n_{in}}} + \frac{\sqrt{3}}{\sqrt{n_{in}}} \right)^2 = \frac{1}{n_{in}} \Rightarrow \text{Var}(h_i) = 1$$

При forward pass на вход идёт  $n_{in}$  наблюдений, при backward pass на вход идёт  $n_{out}$  градиентов  $\Rightarrow$  канал с дисперсией может быть непостоянным, если число весов от слоя к слою сильно колеблется

# Инициализация Ксавье/Глорота (2010)

- Для неодинаковых размеров слоёв невозможно удовлетворить обоим условиям, поэтому обычно усредняют и пытаются инициализировать веса с дисперсией

$$\frac{2}{n_{in} + n_{out}}$$

- Такая инициализация называется **инициализацией Ксавье (или Глорота)**

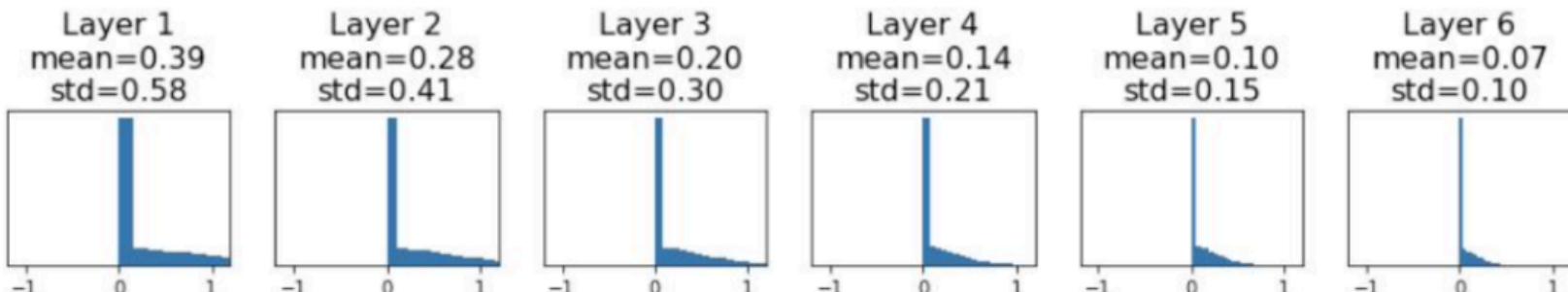
$$w_i \sim U \left[ -\frac{\sqrt{6}}{\sqrt{n_{out} + n_{in}}}; \frac{\sqrt{6}}{\sqrt{n_{out} + n_{in}}} \right],$$

<http://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf>

# А что с ReLU?

Инициализация  
Ксавье предполагает  
симметричную  
функцию активации

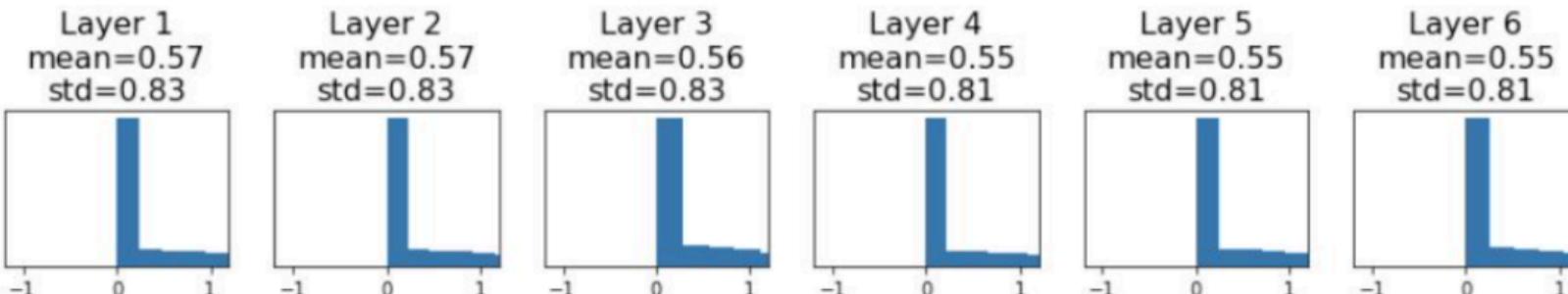
```
dims = [4096] * 7
hs = []
x = np.random.randn(16, dims[0])
for Din, Dout in zip(dims[: -1], dims[1 :]):
 W = 1/np.sqrt(Din) * np.random.randn(Din, Dout)
 x = np.max(0, x.dot(W))
 hs.append(x)
```



# А что с ReLU?

Инициализация  
Ксавье предполагает  
симметричную  
функцию активации

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



# Инициализация Xe (2015)

$$\begin{aligned}\text{Var}(h_i) &= \text{Var}\left(\sum_{i=1}^{n_{in}} w_i x_i\right) = \sum_{i=1}^{n_{in}} \text{Var}(w_i x_i) = \\ &= \sum_{i=1}^{n_{in}} [\mathbb{E}(x_i)]^2 \cdot \text{Var}(w_i) + [\mathbb{E}(w_i)]^2 \cdot \text{Var}(x_i) + \text{Var}(x_i) \cdot \text{Var}(w_i)\end{aligned}$$

- Когда нет симметрии, можно занулить только второе слагаемое

# Инициализация Xe (2015)

$$\begin{aligned}\text{Var}(h_i) &= \text{Var}\left(\sum_{i=1}^{n_{in}} w_i x_i\right) = \sum_{i=1}^{n_{in}} \text{Var}(w_i x_i) = \\ &= \sum_{i=1}^{n_{in}} [\mathbb{E}(x_i)]^2 \cdot \text{Var}(w_i) + [\mathbb{E}(w_i)]^2 \cdot \text{Var}(x_i) + \text{Var}(x_i) \cdot \text{Var}(w_i) = \\ &= \sum_{i=1}^{n_{in}} [\mathbb{E}(x_i)]^2 \cdot \text{Var}(w_i) + \text{Var}(x_i) \cdot \text{Var}(w_i) = \sum_{i=1}^{n_{in}} \text{Var}(w_i) \cdot E(x_i^2)\end{aligned}$$

- Когда нет симметрии, можно занулить только второе слагаемое

# Инициализация Xe (2015)

$$\begin{aligned}\mathbf{Var}(h_i) &= \mathbf{Var}\left(\sum_{i=1}^{n_{in}} w_i x_i\right) = \sum_{i=1}^{n_{in}} \mathbf{Var}(w_i x_i) = \\ &= \sum_{i=1}^{n_{in}} [\mathbf{E}(x_i)]^2 \cdot \mathbf{Var}(w_i) + [\mathbf{E}(w_i)]^2 \cdot \mathbf{Var}(x_i) + \mathbf{Var}(x_i) \cdot \mathbf{Var}(w_i) = \\ &= \sum_{i=1}^{n_{in}} [\mathbf{E}(x_i)]^2 \cdot \mathbf{Var}(w_i) + \mathbf{Var}(x_i) \cdot \mathbf{Var}(w_i) = \sum_{i=1}^{n_{in}} \mathbf{Var}(w_i) \cdot E(x_i^2) = \\ &= E(x^2) \cdot [n_{in} \cdot \mathbf{Var}(w)]\end{aligned}$$

# Инициализация Xe (2015)

$$\begin{aligned}\mathsf{Var}(h_i) &= E(x_i^2) \cdot [n_{in} \cdot \mathsf{Var}(w)] \\ x_i &= \max(0; h_{i-1})\end{aligned}$$

# Инициализация Xe (2015)

$$\begin{aligned}\text{Var}(h_i) &= E(x_i^2) \cdot [n_{in} \cdot \text{Var}(w)] \\ x_i &= \max(0; h_{i-1})\end{aligned}$$

Пусть  $w_{i-1}$  распределены симметрично относительно нуля, тогда  $h_{i-1}$  тоже будут симметрично распределены, тогда:

# Инициализация Xe (2015)

$$\begin{aligned}\text{Var}(h_i) &= E(x_i^2) \cdot [n_{in} \cdot \text{Var}(w)] \\ x_i &= \max(0; h_{i-1})\end{aligned}$$

Пусть  $w_{i-1}$  распределены симметрично относительно нуля, тогда  $h_{i-1}$  тоже будут симметрично распределены, тогда:

$$E(x_i^2) = \frac{1}{2} \cdot \text{Var}(h_{i-1})$$

# Инициализация Xe (2015)

$$\begin{aligned}\text{Var}(h_i) &= E(x_i^2) \cdot [n_{in} \cdot \text{Var}(w)] \\ x_i &= \max(0; h_{i-1})\end{aligned}$$

Пусть  $w_{i-1}$  распределены симметрично относительно нуля, тогда  $h_{i-1}$  тоже будут симметрично распределены, тогда:

$$E(x_i^2) = \frac{1}{2} \cdot \text{Var}(h_{i-1})$$

$$\text{Var}(h_i) = \frac{1}{2} \cdot \text{Var}(h_{i-1}) \cdot [n_{in} \cdot \text{Var}(w)] \Rightarrow \text{Var}(w_i) = \frac{2}{n_{in}}$$

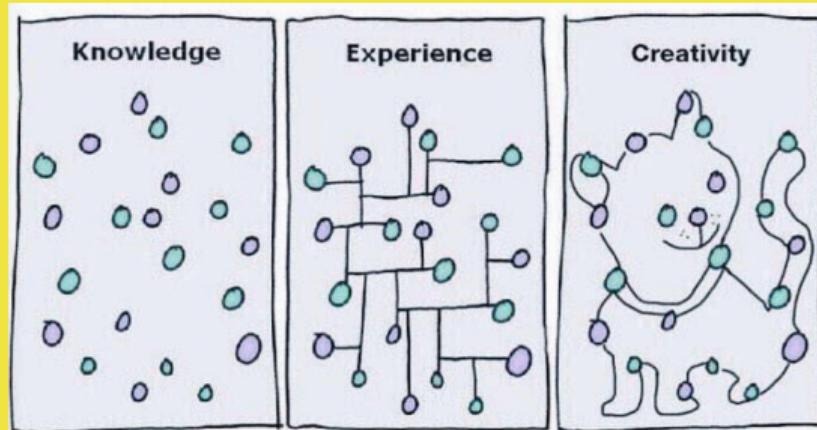
## TLDR: что на практике

- Для симметричных функций с нулевым средним используйте инициализацию Ксавье (Глорота) `init="glorot_uniform"`
- Для ReLU и им подобным инициализацию Хе `init="he_uniform"` или `init="he_normal"`
- Эти две инициализации корректируют параметры распределений в зависимости от входа и выхода слоя так, чтобы поддерживать дисперсию равной единице

# Тема инициализации сегодня активно исследуется

- 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

# Переобучение нейронных сетей



# Что такое асимптотика?

- $n$  — число наблюдений
- Обучение почти любой ML-модели — максимизация правдоподобия
- При  $n \rightarrow \infty$  модель работает хорошо

# Что такое асимптотика?

- $n$  — число наблюдений
- Обучение почти любой ML-модели — максимизация правдоподобия
- При  $n \rightarrow \infty$  модель работает хорошо

# Что такое асимптотика?

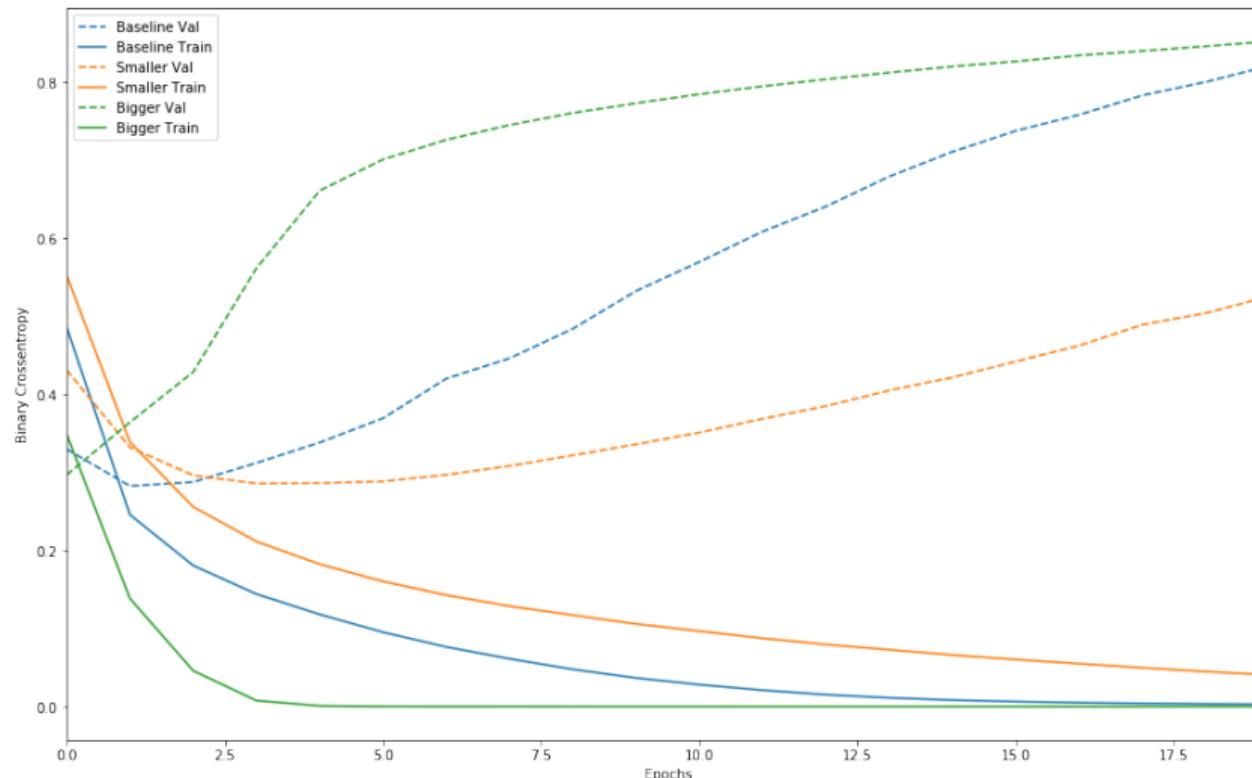
- $n$  — число наблюдений  $k$  — число параметров
- Обучение почти любой ML-модели — максимизация правдоподобия
- При  $n \rightarrow \infty$  модель работает хорошо
- При  $\frac{n}{k} \rightarrow \infty$  модель работает хорошо

# Что такое асимптотика?

- $n$  — число наблюдений  $k$  — число параметров
- Обучение почти любой ML-модели — максимизация правдоподобия
- При  $n \rightarrow \infty$  модель работает хорошо
- При  $\frac{n}{k} \rightarrow \infty$  модель работает хорошо
- Мы живём в эпоху перепараметризованных моделей,  $k \gg n$
- У таких моделей вскрывается много интересных свойств



# Размеры сеток и переобучение



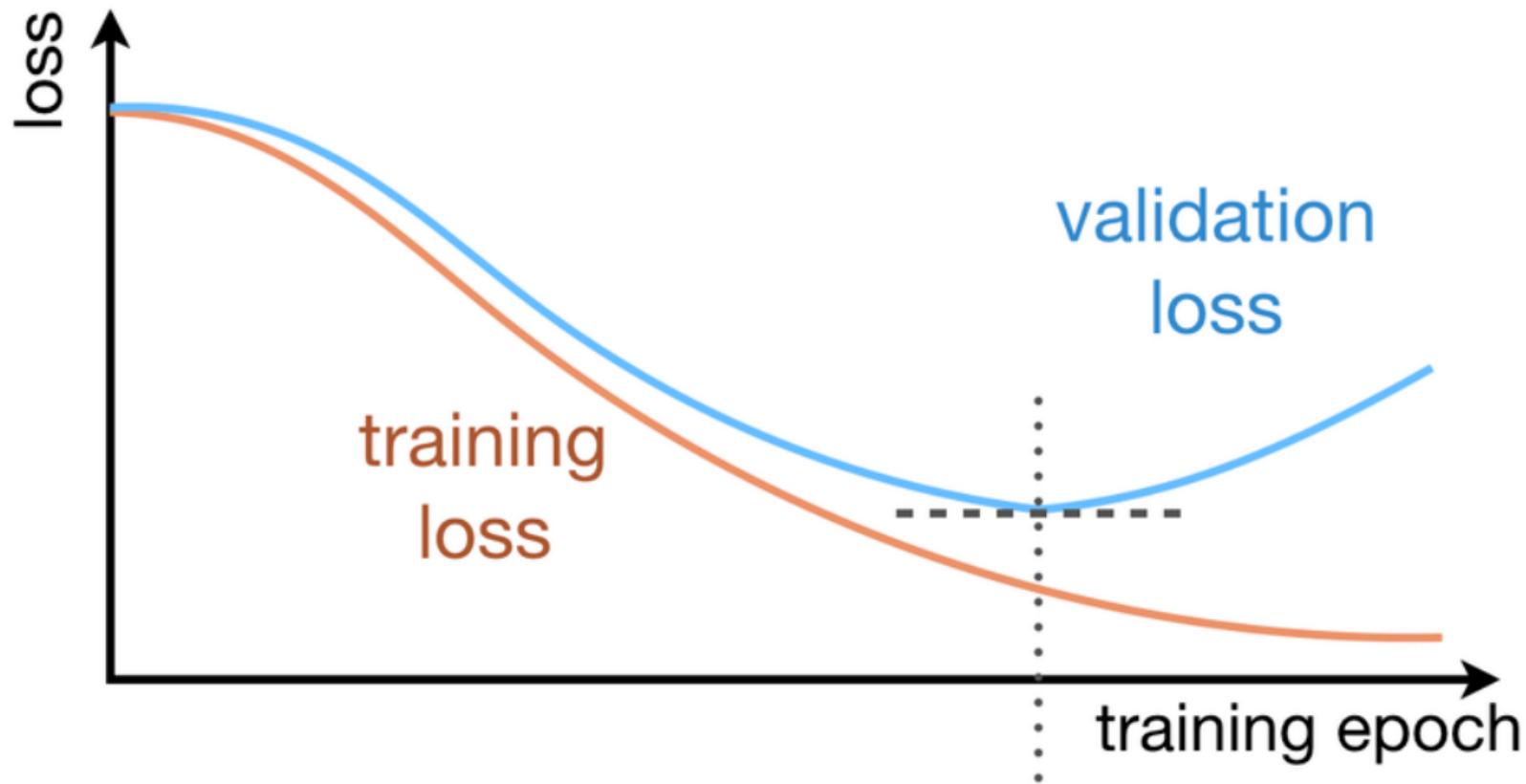
# Борьба с переобучением

- Граница сильно размыта
- Сокращение числа параметров, изобретение более эффективных архитектур
- Регуляризация, разные способы мешать модели подгоняться под обучающую выборку
- Более эффективные методы оптимизации

# Регуляризация

- $L_2$ : приплюсовываем к функции потерь  $\lambda \cdot \sum w_i^2$
- $L_1$ : приплюсовываем к функции потерь  $\lambda \cdot \sum |w_i|$
- Можно регуляризовать не всю сетку, а отдельный нейрон или слой
- Можно наложить регуляризацию на выход из слоя
- Не даёт нейрону сфокусироваться на слишком выделяющемся входе
- Любой дополнительный шум в данных — регуляризация

## Ранняя остановка (early stopping)



## Ранняя остановка (early stopping)

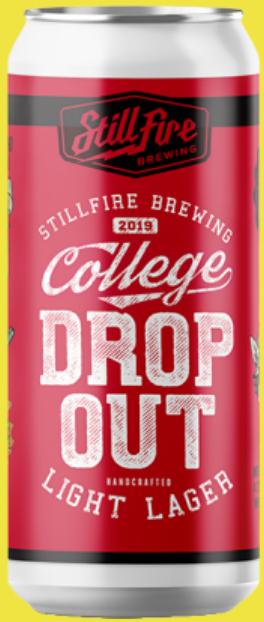
- Будем останавливать обучение, когда качество на валидации начинает падать
- Ранняя остановка действует примерно также, как регуляризаторы
- Например, в книге Гудфеллоу "Глубокое обучение" на стр. 218 можно найти, что ранняя остановка для линейных моделей эквивалентна  $l_2$  регуляризации с MSE, обучаемой SGD

# Регуляризация и Dropout

- На практике обычно используют Dropout
- Он работает похожим на регуляризаторы образом
- Например, в [1] написано:

«We show that the dropout regularizer is first-order equivalent to an L2 regularizer applied after scaling the features by an estimate of the inverse diagonal Fisher information matrix»

[1] <https://arxiv.org/abs/1307.1493>



## Хинтон и банк

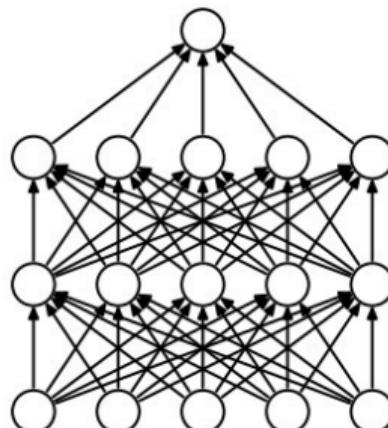
Посещая свой банк я заметил, что операционисты, обслуживающие меня, часто меняются. Я спросил одного из них, почему так происходит. Он сказал, что не знает, но им часто приходится переходить с места на место. Я предположил, что это делается для исключения мошенническогоговора с сотрудником банка.

Это навело меня на мысль, что удаление случайно выбранного подмножества нейронов из каждого примера может помочь предотвратить заговор модели с исходными данными и ослабить эффект переобучения.

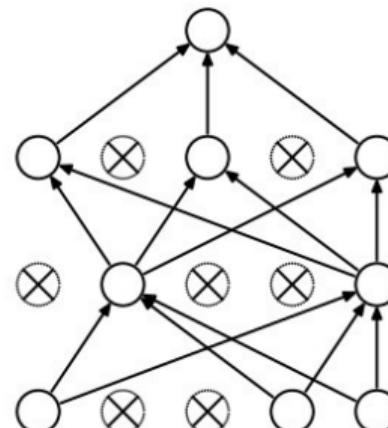
[https://www.reddit.com/r/MachineLearning/comments/4w6tsv/ama\\_we\\_are\\_the\\_google\\_brain\\_team\\_wed\\_love\\_to](https://www.reddit.com/r/MachineLearning/comments/4w6tsv/ama_we_are_the_google_brain_team_wed_love_to)

# Dropout

- При каждом прямом проходе мы зануляем выход нейрона с вероятностью  $p$
- Делает нейроны более устойчивыми к случайным возмущениям



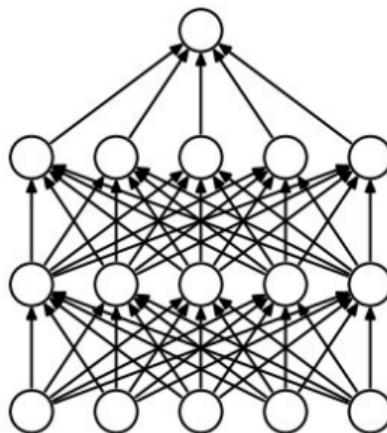
(a) Standard Neural Net



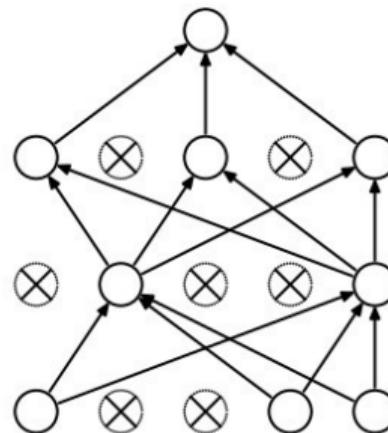
(b) After applying dropout.

# Dropout

- Борьба с ко-адаптацией, не все соседи похожи друг на друга, не все дети похожи на родителей
- Dropout эквивалентен обучению ансамбля из  $2^n$  сетей



(a) Standard Neural Net



(b) After applying dropout.

## Dropout в формулах

- Можно определить как слой  $d(h)$
- Параметров у слоя нет, есть только гиперпараметр  $p$

## Dropout в формулах

- Можно определить как слой  $d(h)$
- Параметров у слоя нет, есть только гиперпараметр  $p$
- forward pass без dropout:

$$o = f(XW)$$

- forward pass с dropout:

$$o = d \cdot f(XW), \quad d = (d_1, \dots, d_k) \sim iidBern(p)$$

# Dropout в формулах

- Можно определить как слой  $d(h)$
- Параметров у слоя нет, есть только гиперпараметр  $p$
- forward pass без dropout:

$$o = f(XW)$$

- forward pass с dropout:

$$o = d \cdot f(XW), \quad d = (d_1, \dots, d_k) \sim iidBern(p)$$

$$o_i = d_i \cdot f(wx_i^T) = \begin{cases} f(wx_i^T), & 1 - p \\ 0, & p \end{cases}$$

Дропаут — это просто небольшая модификация функции активации

# Dropout в формулах

- forward pass:

$$o = f(X \cdot W)$$

$$o = d \cdot f(X \cdot W), \quad d = (d_1, \dots, d_k) \sim iidBern(p)$$

- backward pass:

$$grad = f'(h) \cdot W \cdot grad$$

$$grad = d \cdot f'(h) \cdot W \cdot grad$$

# Dropout

- При обучении мы домножаем часть выходов на  $d_i$ , тем самым мы изменяем только часть параметров и нейроны учатся более независимо  $\Rightarrow$  регуляризация
- Дропаут выплёвывает случайные выходы, что делать на стадии тестирования?

# Dropout

- При обучении мы домножаем часть выходов на  $d_i$ , тем самым мы изменяем только часть параметров и нейроны учатся более независимо  $\Rightarrow$  **регуляризация**
- Дропаут выплёвывает случайные выходы, **что делать на стадии тестирования?**
- Нам надо сымитировать работу такого ансамбля: можно отключать по очереди все возможные комбинации нейронов, получить  $2^n$  прогнозов и усреднить их

# Dropout

- При обучении мы домножаем часть выходов на  $d_i$ , тем самым мы изменяем только часть параметров и нейроны учатся более независимо  $\Rightarrow$  регуляризация
- Дропаут выплёвывает случайные выходы, что делать на стадии тестирования?
- Нам надо сымитировать работу такого ансамбля: можно отключать по очереди все возможные комбинации нейронов, получить  $2^n$  прогнозов и усреднить их
- Но лучше просто брать по дропауту математическое ожидание:

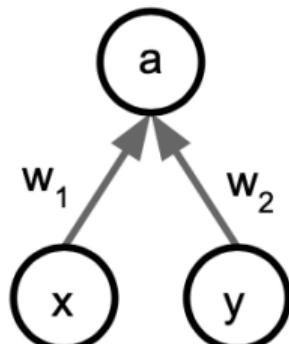
$$o = p \cdot f(X \cdot W)$$

# Dropout на тестовой выборке

- Но лучше просто брать по дропауту математическое ожидание:

$$o = p \cdot f(X \cdot W)$$

**Пример:** используем дропаут с вероятностью 0.5:



$$\begin{aligned}\mathbb{E}(a) &= \frac{1}{4} \cdot (w_1 x + w_2 y) + \frac{1}{4} \cdot (w_1 x + 0y) + \\ &\quad + \frac{1}{4} \cdot (0x + w_2 y) + \frac{1}{4} \cdot (0x + 0y) = \frac{1}{2} \cdot (w_1 \cdot x + w_2 \cdot y)\end{aligned}$$

Достаточно домножить выход на 0.5

# Обратный Dropout

- На тесте ищем математическое ожидание:

$$o = p \cdot f(X \cdot W)$$

- Это неудобно! Надо переписывать функцию для прогнозов!

# Обратный Dropout

- На тесте ищем математическое ожидание:

$$o = p \cdot f(X \cdot W)$$

- Это неудобно! Надо переписывать функцию для прогнозов!
- Давайте лучше будем домножать на  $\frac{1}{p}$  на этапе обучения:

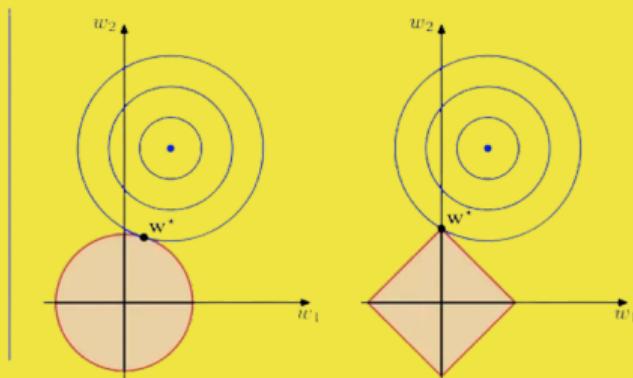
train: 
$$o = \frac{1}{p} \cdot d \cdot f(X \cdot W)$$

test: 
$$o = f(X \cdot W)$$

# Интерпретация

- Мы обучаем все возможные архитектуры нейросетей, которые получаются из исходной выбрасыванием отдельных нейронов
- У всех этих архитектур общие веса
- На этапе применения усредняем прогнозы всех этих архитектур

# Регуляризация нейронных сетей

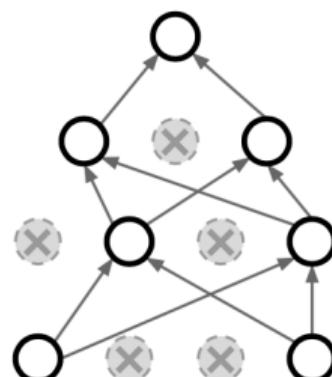
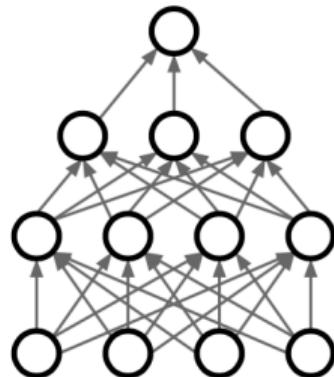


# Регуляризация: общий паттерн

- На обучающей выборке мы вносим какую-то случайность
- На тестовой выборке мы усредняем эту случайность
- Есть много подходов и приёмов, в этой секции мы рассмотрим несколько примеров

# Dropout

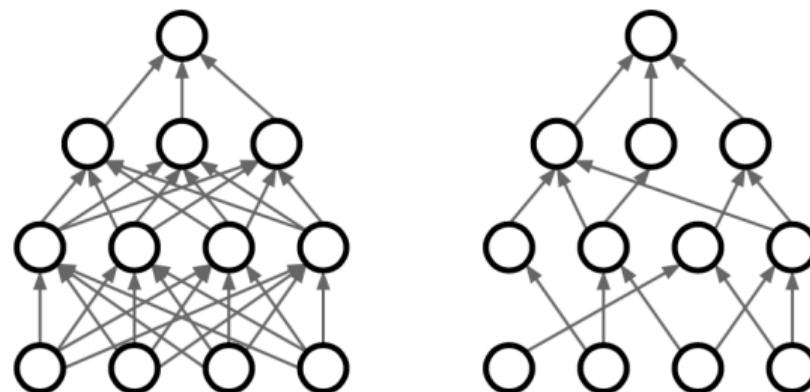
- **Обучение:** случайно зануляем выходы нейронов
- **Тест:** усредняем выход на вероятность



<http://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf>

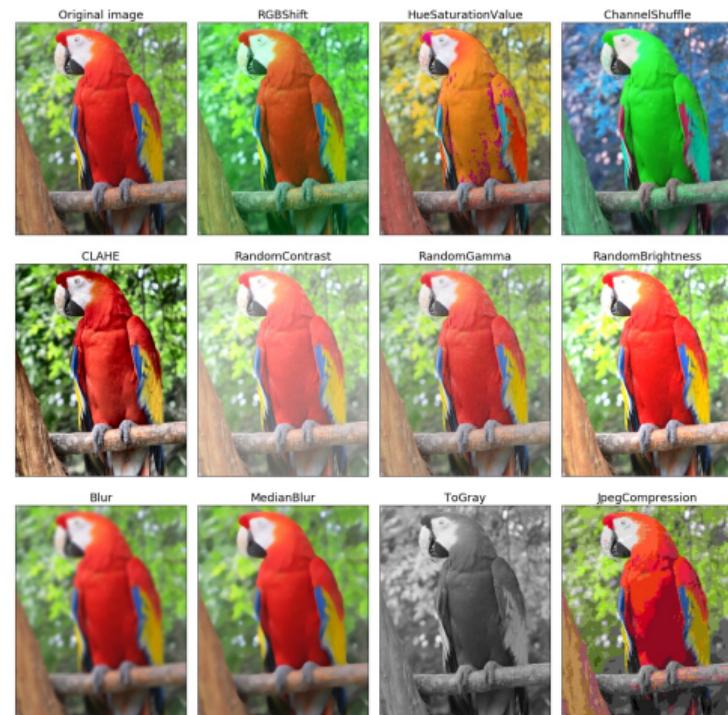
# DropConnect

- **Обучение:** случайно удаляем связи между нейронами (зануляем соответствующий вес)
- **Тест:** используем все связи
- На практике используется редко



# Аугментация (Data augmentation)

- Увеличение, уменьшение
- Повороты, искажения, приближения
- Новые цвета, затемнения
- Смена стиля
- Нужно ли добавлять сдвиги?



[https://www.tensorflow.org/tutorials/images/data\\_augmentation](https://www.tensorflow.org/tutorials/images/data_augmentation)

# Аугментация (Data augmentation)

- Увеличение, уменьшение
- Повороты, искажения, приближения
- Новые цвета, затемнения
- Смена стиля
- Нужно ли добавлять сдвиги?  
**(нет, вместо них лучше пулинг)**



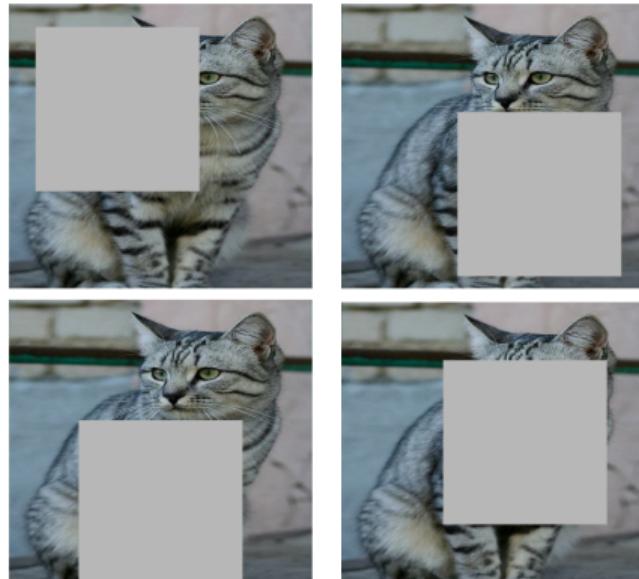
[https://www.tensorflow.org/tutorials/images/data\\_augmentation](https://www.tensorflow.org/tutorials/images/data_augmentation)

# Аугментация (Data augmentation)

- Много разных вариантов
- «Бесплатное» расширение обучающей выборки
- **Обучение:** случайно применяем к картинкам из текущего батча
- **Тест:** делаем несколько аугментаций картинки, применяем сеть к каждой и усредняем предсказания
- Для тестовых данных набор аугментаций всегда фиксирован

# Вырезание (Cutout/Random crop)

- **Обучение:** Зануляем рандомные куски картинки
- **Тест:** используем всю картинку
- Неплохо работает для маленьких датасетов как CIFAR
- На больших датасетах не очень полезно



<https://arxiv.org/abs/1708.04552>