

# Нейронные сети

## Лекция 3. Несвёрточные слои

**Иванов И.Е.**

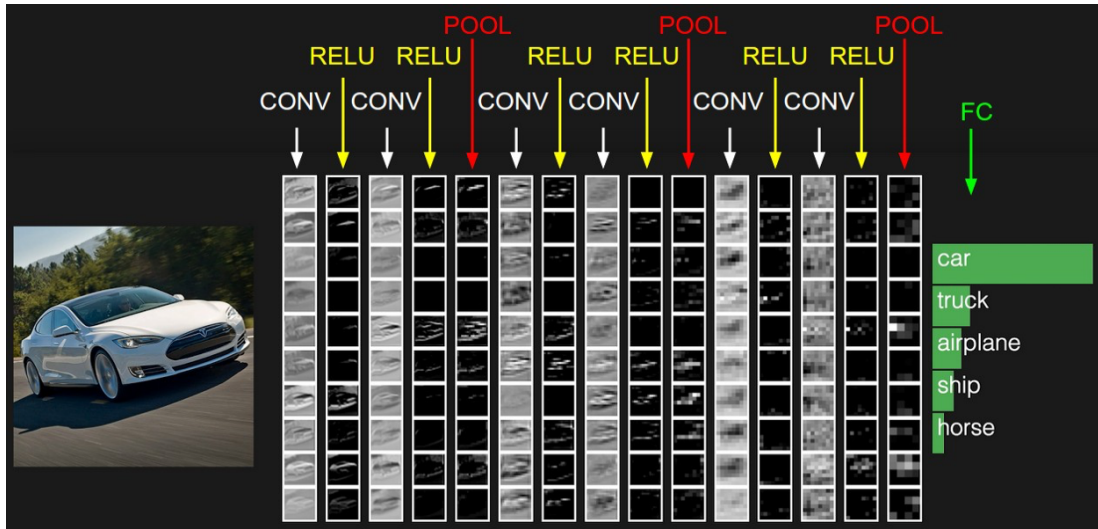
MaTIS

10 октября 2025г.



- 1 Субдискретизация
- 2 Нелинейность
- 3 Полносвязный и Softmax слои
- 4 Дропаут
- 5 Пакетная нормализация

# Визуализация работы сверточной сети<sup>1</sup>



<sup>1</sup><https://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html>

# Основные типы слоев в СНС

С прошлой лекции:

## Входной слой INPUT

Необработанные пиксельные значения входной картинки. Это — первый слой в СНС.

## Сверточный слой CONV

Скалярное произведение между элементами фильтра (также называемого **ядром** свертки) и ограниченной областью (обычно гораздо меньше всей площади  $H \times W$ ) входного слоя, с которой имеются связи, с помощью скользящего окна (слева направо сверху вниз).



# Основные типы слоев в CHC

Сегодня:

## Нелинейность ReLU

Нелинейность вида  $ReLU(x) = \max(0, x)$ , применяемая ко всем нейронам слоя поточечно.

## Слой субдискретизации POOL

Уменьшение размерности по пространственным измерениям  $w, h$ . Могут использоваться разные подходы: усреднение, взятие максимума по подобласти и т.п.

## Полносвязный слой FC (Fully connected)

Матричное умножение — в данном случае каждый нейрон выходного слоя связан со всеми нейронами входного слоя (в отличие от сверточного слоя).



1	3	2	9
7	4	1	5
8	5	2	3
4	2	1	4

7	9
8	

Слой субдискретизации решает две проблемы:

- Снижает пространственную размерность
- Помогает не переобучаться

# Параметры слоя субдискретизации

## Размер фильтра

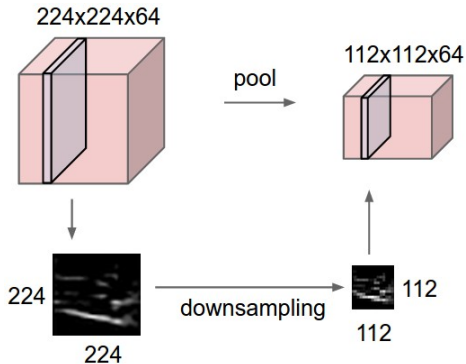
Пространственная размерность области (по горизонтали и вертикали), внутри которой применяется функция уменьшения размерности (max, avg).

## Шаг (stride)

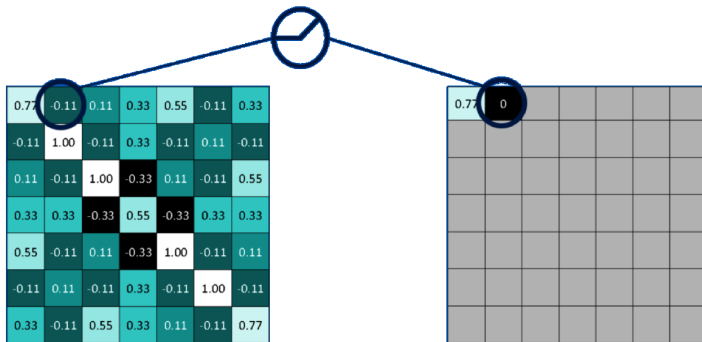
Количество элементов по горизонтали или вертикали, на которое перемещается фильтр для получения результирующей карты признаков.



# Иллюстрация уменьшения размерности







- Применение нелинейной функции (например,  $ReLU(x) = \max(0, x)$ )
- Цель: выделение наиболее значимой информации

- Также называется активацией
- Нужен для увеличения эффективной глубины СНС.
- Применяется поэлементно для нейронов всего слоя.
- Обычно не имеет обучаемых параметров (за редким исключением, например, PReLU)

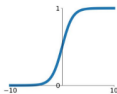
## Примеры активаций

- Rectified Linear Unit  $ReLU(x) = \max(0, x)$
- Сигмоида  $\sigma(x) = \frac{1}{1+\exp(-x)}$
- Гиперболический тангенс  $\tanh(x) = 2\sigma(2x) - 1$
- ReLU с утечкой (Leaky ReLU)  $LReLU(x) = (x < 0) * \alpha x + (x \geq 0) * x$
- Maxout( $x$ ) =  $\max(a_1x + b_1, a_2x + b_2)$
- Экспоненциальный Linear Unit  $ELU(x) = (x < 0) * \alpha(\exp(x) - 1) + (x \geq 0) * x$

## Activation Functions

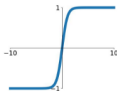
### Sigmoid

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



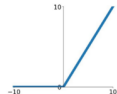
### tanh

$$\tanh(x)$$



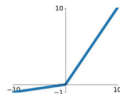
### ReLU

$$\max(0, x)$$



### Leaky ReLU

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

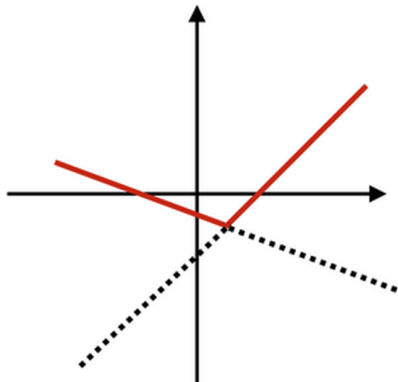
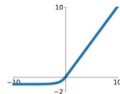


### Maxout

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

### ELU

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



- 1 Для классификации на  $N$  классов обычно определяют *вероятность*  $p_i$  принадлежности к каждому из классов
- 2 Для этого сначала вычисляют  $N$  т.н. *логитов*  $l_i$  — скалярных значений из  $\mathbb{R}$
- 3 При этом на выходе последней операции СНС (например, свертки) может оказаться тензор  $X'$  произвольного размера  $M = d * w * h$ , который может быть преобразован для упрощения вычислений в вектор  $X$  размера  $M \times 1$
- 4 Как раз для преобразования  $M$  входов в  $N$  выходов-логитов и применяется полносвязный слой, или умножение на матрицу  $A$  размера  $N \times M$ :  $Y = A * X$ ,  $Y_i = l_i$
- 5 Иногда к результату умножения на матрицу добавляют одномерный тензор сдвига  $b^k$  длины  $N$

**Замечание.** Обычно полносвязные слои — самые большие по объему и не очень быстрые, поэтому нужно стараться их избегать (average pooling) либо оптимизировать



- ❶ Операция Softmax — это обобщение сигмоиды на случай  $N$  входов:

$$\text{Softmax}(Y)_i = \frac{e^{l_i}}{\sum_{k=1}^N e^{l_k}} = p_i$$

- ❷ Теперь  $p_i$  — корректный вектор вероятностей:

$$\sum_{k=1}^N p_k = 1, \quad 0 \leq p_i \leq 1 \quad \forall i = 1 \dots N$$



## Шаблон глубокой СНС

$\text{INPUT} \rightarrow [[\text{CONV} \rightarrow \text{RELU}] * N \rightarrow \text{POOL?}] * M \rightarrow [\text{FC} \rightarrow \text{RELU}] * K \rightarrow \text{Softmax}$

**Замечание.** Современные СНС зачастую имеют немного более сложную структуру

- 1 Сети типа ResNet имеют т.н. остаточные (residual) связи
- 2 Сети типа Inception предлагают конкатенацию слоев + разделение одной 2D свертки на две 1D свертки
- 3 Слой BatchNormalization выполняет послойную нормализацию
- 4 Dropout борется с переобучением



# Откуда берутся размерности $\geq 4$

## Размерность 4

Обычно это размерность т.н. пакета (batch) входных данных, над которыми все операции выполняются совершенно идентично и параллельно (в рамках используемой архитектуры). Например, размер пакета из 32 входных картинок

## Размерность 5

Дополнительная размерность необходима для обработки видео и задает количество кадров, при этом она будет четвертой размерностью, а на пятую сдвинется размер пакета (он всегда либо первый, либо последний — в зависимости от реализации).



# Дропаут<sup>2</sup> (выброс)

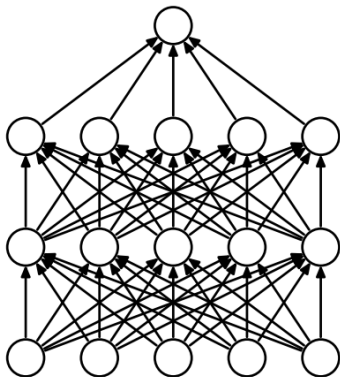
- Для уменьшения переобучения, во время обучения нейроны “выключают” с вероятностью  $0 \leq 1 - p \leq 1$
- Это можно сделать, зануляя “выключенные” нейроны
- На тесте нейроны не выключаются; при этом выход нейрона умножается на  $p$ 
  - Матожидание выхода нейрона при обучении  $px + (1 - p)0 = px$  (т.к. мы либо пропускаем нейрон без изменений, либо зануляем)
  - Поэтому при тестировании, когда все нейроны включены, их выходы нужно шкалировать для такого же матожидания
- Либо при обучении делим выход на  $p$ : тогда на тесте ничего домножать не надо

---

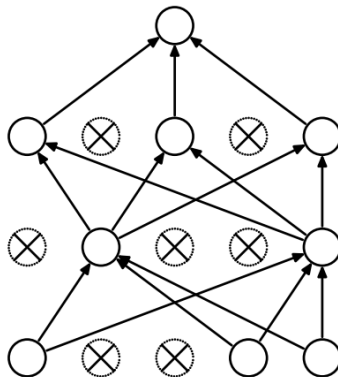
<sup>2</sup>Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting” 2014



# Схема дропаута



(a) Standard Neural Net



(b) After applying dropout.

# Внутренний ковариационный сдвиг

## Проблема

- Внутренний ковариационный сдвиг (Internal Covariate Shift, ICS) — изменение распределения значений нейронов вследствие изменения параметров нейросети во время обучения
- Более глубокая нейросеть  $\Rightarrow$  больший сдвиг

## Очевидные пути решения для глубоких нейросетей (следующая лекция)

- Очень аккуратная инициализация параметров нейросети
- Маленький коэффициент скорости обучения (и, как следствие, очень медленное обучение)
- Нормализация входа<sup>3</sup> (помогает слабо)

<sup>3</sup>LeCun Y. A. et al. "Efficient backprop". 1998

## Решение

- **Нормализация по пакету** (Batch Normalization, BN) — та самая 4 размерность
- Можно нормализовать каждый слой (а не только вход)
- Нужно нормализовать на каждом пакете данных (mini-batch)
- Дальше для обучения параметров нейросети будут подаваться уже нормализованные значения (и т.о. уменьшаем ICS)

## Преимущества BN

- За счет большего learning rate скорость обучения возрастает в разы
- Не так чувствительна к инициализации
- Не нужен дропаут

<sup>4</sup>Ioffe S., Szegedy C. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". 2015

# Когда и где применять BN

## Когда

- В глубоких нейросетях
- Нужно ускорить скорость обучения

## Где

- После операции свертки или других матричных операций
- До применения функции активации (до ReLU) — т.к. функция активации сама по себе сильно меняет распределение
- Тем не менее, есть свидетельства, что порой можно применить BN и после активации (хотя и не всегда это работает)



BN работает по-разному во время тестирования (т.н. inference mode) и во время обучения

## Обучение

- Подсчитываем  $\mu_B$  и  $\sigma_B$  на пакете  $B$
- Обновляем глобальные значения (соотв. всему обучающему множеству)  $\mu_{avg}$  и  $\sigma_{avg}$

## Тестирование

- Используем значения  $\mu_{avg}$  и  $\sigma_{avg}$  вне зависимости от  $\mu_B$  и  $\sigma_B$  на текущем пакете



- Предположим, что мы используем пакет размера  $T$
- $X_{ij}^{mt}$  — четырехмерный тензор значений для некоторого слоя, где
  - $1 \leq i \leq H, 1 \leq j \leq W$  — пространственные координаты (ширина и высота),
  - $m = 1 \dots M$  — номер карты признаков,
  - $t = 1 \dots T$  — номер внутри пакета.

### Статистика на пакете

- $\mu_B^m = \frac{1}{HWT} \sum_t \sum_{i,j} X_{ij}^{mt}$
- $\sigma_B^{2m} = \frac{1}{HWT} \sum_t \sum_{i,j} (X_{ij}^{mt} - \mu_B^m)^2$



## Гиперпараметры

- $\alpha \in [0, 1]$ : параметр сглаживания для обновления глобальных параметров
- $\epsilon > 0$  — регуляризатор (маленькое число)

## Шаг обучения $k$

- $\mu_{avg,k}^m = \alpha \mu_{avg,k-1}^m + (1 - \alpha) \mu_B^m$  (инициализация  $\mu_{avg,0}^m = 0$ )
- $\sigma_{avg,k}^{2m} = \alpha \sigma_{avg,k-1}^{2m} + (1 - \alpha) \sigma_B^{2m}$  (инициализация  $\sigma_{avg,0}^{2m} = 1$ )
- Выход нормализованного слоя:  $Y_{ij}^{mt} = \gamma^m \frac{X_{ij}^{mt} - \mu_B^m}{\sqrt{\sigma_B^{2m} + \epsilon}} + \beta^m$
- Параметры  $\gamma^m$  (масштаб, scale) и  $\beta^m$  (сдвиг, shift) — обучаемые

**Замечание.** В случае  $\gamma^m = \sqrt{\sigma_B^{2m} + \epsilon}$ ,  $\beta^m = \mu_B^m$  получим  $Y_{ij}^{mt} = X_{ij}^{mt}$  и BN в принципе может обучиться ничего не делать (ничего не портить).



- 1 Используем уже обученные параметры масштаба  $\gamma^m$  и сдвига  $\beta^m$
- 2 Несмотря на то, что в тесте данные тоже могут подаваться пакетами, не обращаем внимание на статистику пакета  $\mu_B^m$  и  $\sigma_B^{2m}$
- 3 Не обновляем глобальные параметры  $\mu_{avg}^m$  и  $\sigma_{avg}^{2m}$
- 4 Выход нормализованного слоя: 
$$Y_{ij}^{mt} = \gamma^m \frac{X_{ij}^{mt} - \mu_{avg}^m}{\sqrt{\sigma_{avg}^{2m} + \epsilon}} + \beta^m$$

## Число параметров для BN

- Для каждой карты признаков нужно хранить 4 числа: 2 — глобальные статистики, и 2 — параметры сдвига и масштаба
- Если  $L$  слоев по  $M$  карт каждый, то число BN параметров составляет  $N_{BN} = 4LM$
- $N_{BN} \ll N_{CONV}$

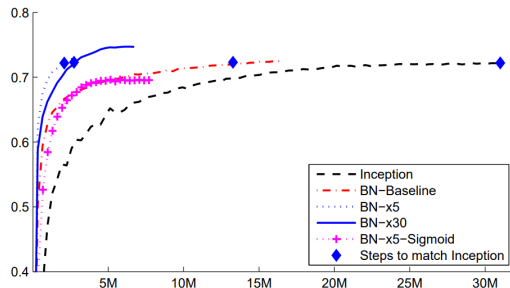




Использование BN позволило достичь двух целей:

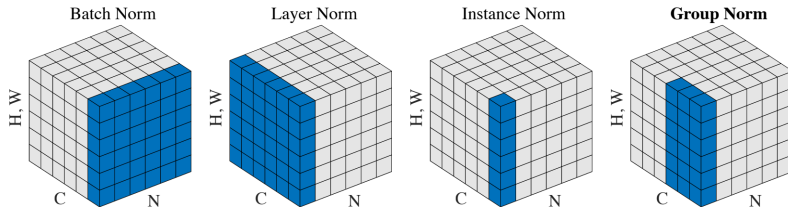
- Ускорить обучение до одинакового качества (вплоть до 15 раз)
- Улучшить качество (на 2.6%)

	Model	Steps to 72.2%	Max accuracy
	Inception	$31.0 \cdot 10^6$	72.2%
	BN-Baseline	$13.3 \cdot 10^6$	72.7%
LR = LR x 5	BN-x5	$2.1 \cdot 10^6$	73.0%
LR = LR x 30	BN-x30	$2.7 \cdot 10^6$	74.8%



# Другие виды нормализаций

- Нормализация по слою, а не по пакету<sup>5</sup> (layer normalization)
- Нормализация по одной карте признаков<sup>6</sup> (instance normalization)
- Нормализация по части слоя<sup>7</sup> (group normalization)



<sup>5</sup>Ba J. L., Kiros J. R., Hinton G. E. "Layer normalization". 2016

<sup>6</sup>Ulyanov D., Vedaldi A., Lempitsky V. "Instance normalization: The missing ingredient for fast stylization". 2016

<sup>7</sup>Wu Y., He K. "Group normalization". 2018

- Предположим, что мы используем пакет размера  $T$
- $X_{ij}^{mt}$  — четырехмерный тензор значений для некоторого слоя, где
  - $1 \leq i \leq H, 1 \leq j \leq W$  — пространственные координаты (ширина и высота),
  - $m = 1 \dots M$  — номер карты признаков,
  - $t = 1 \dots T$  — номер внутри пакета.

## Статистика по слою

- $\mu_B^t = \frac{1}{HWM} \sum_m \sum_{i,j} X_{ij}^{mt}$
- $\sigma_B^{2t} = \frac{1}{HWM} \sum_m \sum_{i,j} (X_{ij}^{mt} - \mu_B^t)^2$



# IN: Нормализация по одной карте признаков

- Предположим, что мы используем пакет размера  $T$
- $X_{ij}^{mt}$  — четырехмерный тензор значений для некоторого слоя, где
  - $1 \leq i \leq H, 1 \leq j \leq W$  — пространственные координаты (ширина и высота),
  - $m = 1 \dots M$  — номер карты признаков,
  - $t = 1 \dots T$  — номер внутри пакета.

## Статистика по одной карте признаков

- $\mu_B^{mt} = \frac{1}{HW} \sum_{i,j} X_{ij}^{mt}$
- $\sigma_B^{2mt} = \frac{1}{HW} \sum_{i,j} (X_{ij}^{mt} - \mu_B^{mt})^2$

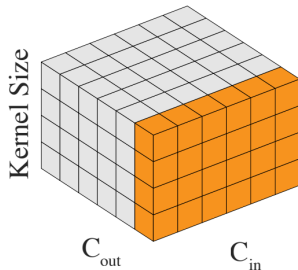


# WS: стандартизация весов<sup>8</sup>

- $W_{uv}^{mk}$  — четырехмерный тензор значений для некоторого фильтра, где
  - $1 \leq u \leq p, 1 \leq v \leq q$  — пространственные координаты (ширина и высота),
  - $m = 1 \dots M$  — количество карт входного слоя,
  - $k = 1 \dots K$  — количество карт выходного слоя.

## Стандартизация весов

- $\mu_W^k = \frac{1}{pqM} \sum_m \sum_{u,v} W_{uv}^{mk}$
- $\sigma_W^{2k} = \frac{1}{pqM} \sum_m \sum_{u,v} (W_{uv}^{mk} - \mu_W^k)^2$
- $\hat{W}_{uv}^{mk} = \frac{W_{uv}^{mk} - \mu_W^k}{\sqrt{\sigma_W^{2k} + \epsilon}}$



<sup>8</sup>Qiao S. et al. "Weight standardization." 2019

Спасибо за внимание!

