Нейронные сети

Лекция 3. Несвёрточные слои

Иванов И.Е.

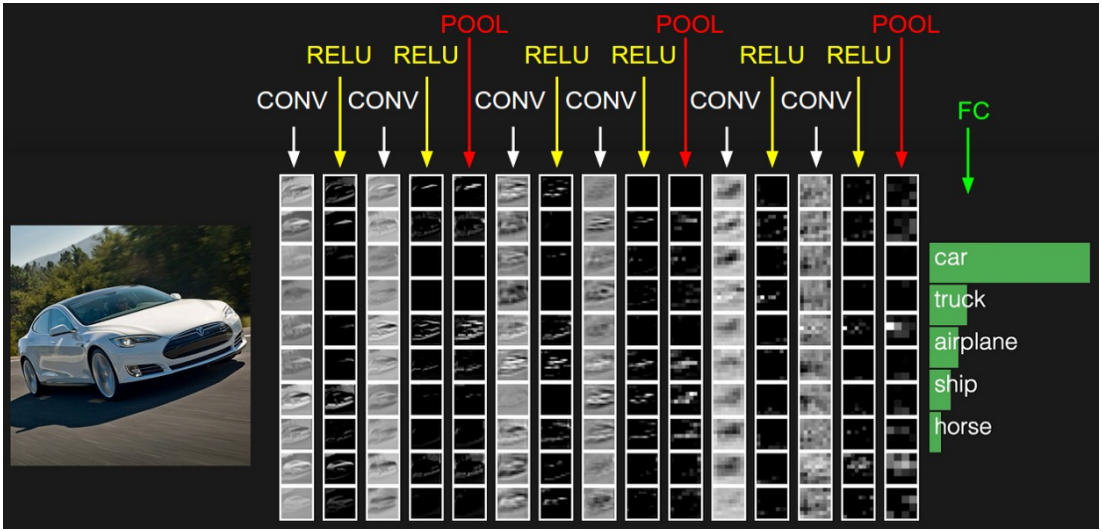
МаТИС

04 октября 2024г.

План лекции

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

Визуализация работы сверточной сети[1]



[1] https://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html

Основные типы слоев в CHC

С прошлой лекции:

Входной слой INPUT

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

Сверточный слой CONV

Скалярное произведение между элементами фильтра (также называемого **ядром** свертки) и ограниченой 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.

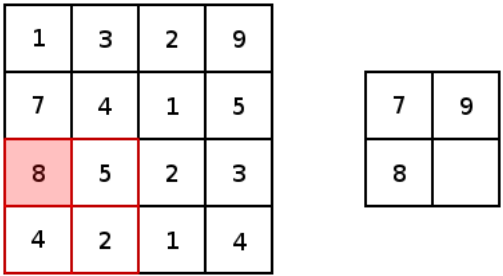
Слой субдискретизации POOL

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).

Субдискретидация



Слой субдискретизации решает две проблемы:

- Снижает пространственную размерность
- Помогает не переобучаться

Параметры слоя субдискретизации

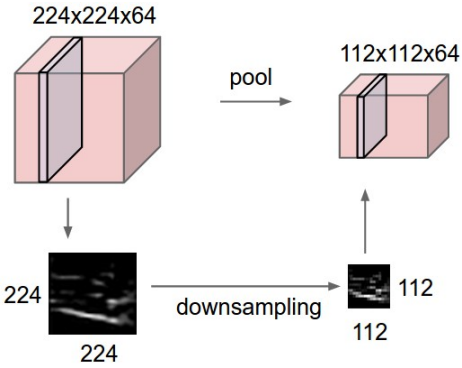
Размер фильтра

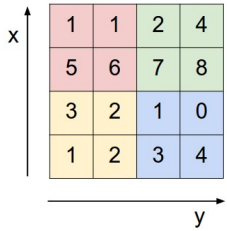
Пространственная размерность области (по горизонтали и вертикали), внутри которой применяется функция уменьшения размерности (max, avg).

Шаг (stride)

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

Иллюстрация уменьшения размерности

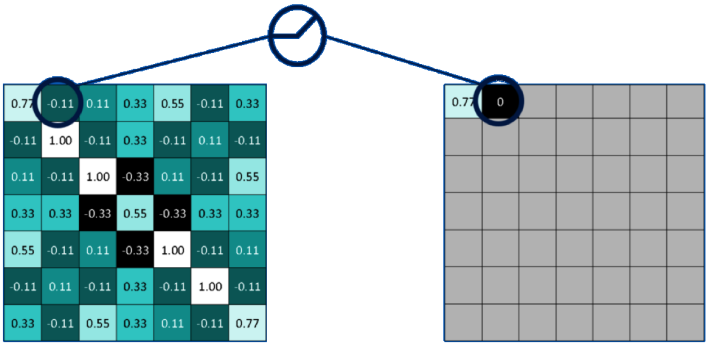




Maxpool с шагом 2
и фильтром 2x2

6	8
3	4

Активация



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

Слой нелинейности

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

Примеры активаций

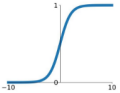
- Rectified Linear Unit $ReLU(x) = \max(0, x)$
- Сигмоида $\sigma(x) = \frac{1}{1+\exp(-x)}$
- Гиперболический тангенс $\tanh(x) = 2\sigma(2x) - 1$
- ReLU с утечкой (Leaky ReLU) $LReLU(x) = (x < 0) * \alpha x + (x \geq 0) * x$
- $Maxout(x) = \max(a_1 x + b_1, a_2 x + b_2)$
- Экспоненциальный Linear Unit $ELU(x) = (x < 0) * \alpha(\exp(x) - 1) + (x \geq 0) * x$

Иллюстрация активаций

Activation Functions

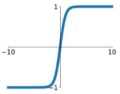
Sigmoid

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



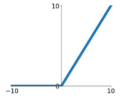
tanh

$\tanh(x)$



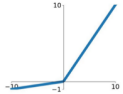
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

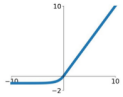


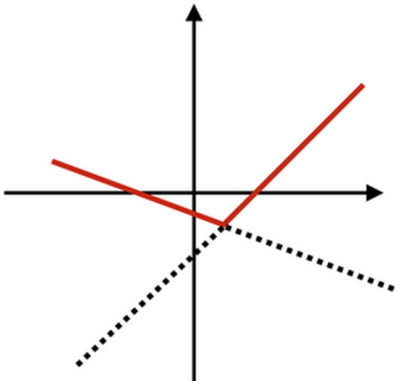
Maxout

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

ELU

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





Полносвязный слой

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

Слой Softmax

1. Операция Softmax — это обобщение сигмоиды на случай N входов:

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

2. Теперь p_i — корректный вектор вероятностей:

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

Шаблоны построения CHC

Шаблон глубокой CHC

INPUT→[[CONV→RELU]*N→POOL?]*M→[FC→RELU]*K→Softmax

Замечание. Современные CHC 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 ≥ 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

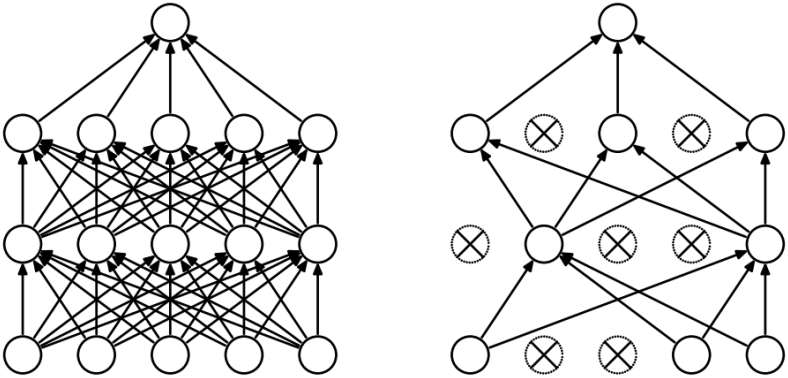
Дополнительная размерность необходима для обработки видео и задает количество кадров, при этом она будет четвертой размерностью, а на пятую сдвинется размер пакета (он всегда либо первый, либо последний — в зависимости от реализации).

Дропаут[2] (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

[2] 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[3] (помогает слабо)

[3] LeCun Y. A. et al. "Efficient backprop". 1998

Пакетная нормализация[4]

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

[4] 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)

Режимы работы BN

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

Обучение

- Подсчитываем μ_B и σ_B на пакете B
- Обновляем глобальные значения (соотв. всему обучающему множеству) μ_{avg} и σ_{avg}

Тестирование

- Используем значения μ_{avg} и σ_{avg} вне зависимости от μ_B и σ_B на текущем пакете

BN: статистика на пакете

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

Статистика на пакете

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

BN: режим обучения

Гиперпараметры

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

Шаг обучения k

- $\mu^m_{avg,k} = \alpha\mu^m_{avg,k-1} + (1-\alpha)\mu^m_B$ (инициализация $\mu^m_{avg,0} = 0$)
- $\sigma^{2m}_{avg,k} = \alpha\sigma^{2m}_{avg,k-1} + (1-\alpha)\sigma^{2m}_B$ (инициализация $\sigma^{2m}_{avg,0} = 1$)
- Выход нормализованного слоя: $Y^{mt}_{ij} = \gamma^m \frac{X^{mt}_{ij} - \mu^m_B}{\sqrt{\sigma^{2m}_B + \epsilon}} + \beta^m$
- Параметры γ^m (масштаб, scale) и β^m (сдвиг, shift) — обучаемые

Замечание. В случае $\gamma^m = \sqrt{\sigma^{2m}_B + \epsilon}, \beta^m = \mu^m_B$ получим $Y^{mt}_{ij} = X^{mt}_{ij}$ и BN в принципе может обучиться ничего не делать (ничего не портить).

BN: режим теста

1. Используем уже обученные параметры масштаба γ^m и сдвига β^m
2. Несмотря на то, что в тесте данные тоже могут подаваться пакетами, не обращаем внимание на статистику пакета μ_B^m и σ_B^{2m}
3. Не обновляем глобальные параметры μ_{avg}^m и σ_{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

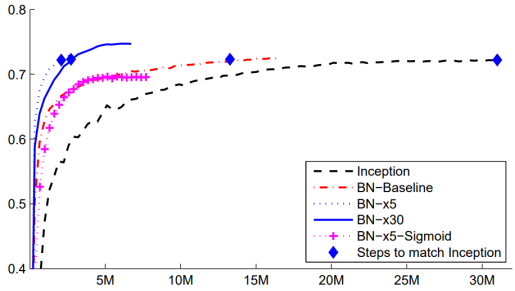
- Для каждой карты признаков нужно хранить 4 числа: 2 — глобальные статистики, и 2 — параметры сдвига и масштаба
- Если L слоев по M карт каждый, то число BN параметров составляет $N_{BN} = 4LM$
- $N_{BN} \ll N_{CONV}$

BN: эффект

Использование BN позволило достичь двух целей:

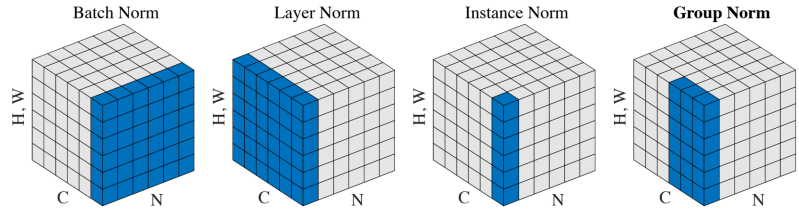
- Ускорить обучение до одинакового качества (вплоть до 15 раз)
- Улучшить качество (на 2.6%)

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[5] (layer normalization)
- Нормализация по одной карте признаков[6] (instance normalization)
- Нормализация по части слоя[7] (group normalization)



[5] Ba J. L., Kiros J. R., Hinton G. E. “Layer normalization”. 2016

[6] Ulyanov D., Vedaldi A., Lempitsky V. “Instance normalization: The missing ingredient for fast stylization”. 2016

[7] Wu Y., He K. “Group normalization”. 2018

LN: Нормализация по слою

- 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

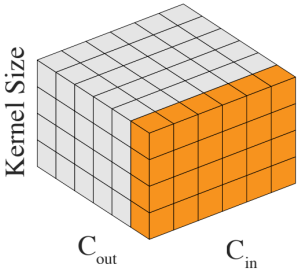
- Предположим, что мы используем пакет размера T
- X_{ij}^{mt} — четырехмерный тензор значений для некоторого слоя, где
  - $1 \le i \le H, 1 \le j \le W$ — пространственные координаты (ширина и высота),
  - $m = 1 \dots M$ — номер карты признаков,
  - $t = 1 \dots T$ — номер внутри пакета.

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[8]

- 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

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

[8] Qiao S. et al. "Weight standardization." 2019

Спасибо за внимание!