

Методы машинного обучения. Глубокие нейронные сети

Воронцов Константин Вячеславович
www.MachineLearning.ru/wiki?title=User:Vokov
вопросы к лектору: voron@forecsys.ru

материалы курса:
github.com/MSU-ML-COURSE/ML-COURSE-21-22
орг.вопросы по курсу: mlcmc@mail.ru

1 Глубокие нейронные сети и их обоснования

- Глубокие нейронные сети: глубина важнее ширины
- Польза избыточной параметризации
- Векторизация сложных объектов

2 Свёрточные нейронные сети

- Свёртки и пулинги для обработки изображений
- Приложения: изображения, тексты, речь, игры
- Обобщение: данные с локальными структурами

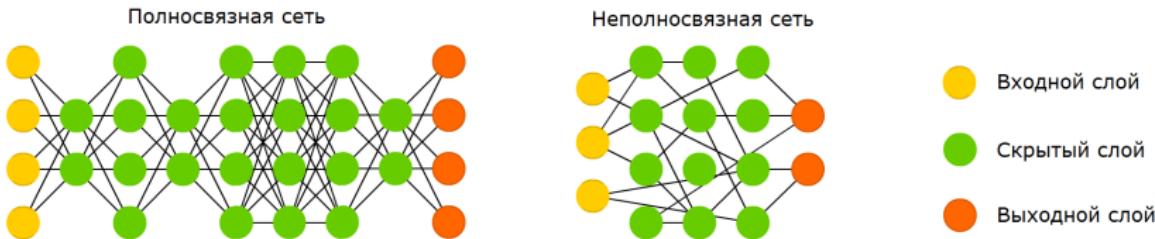
3 Рекуррентные нейронные сети

- Нейронные сети для обработки последовательностей
- Сети долгой кратковременной памяти LSTM
- Варианты LSTM, сети GRU и SRU

Глубокие нейронные сети (Deep Neural Network, DNN)

1965: первые глубокие нейронные сети

2012: свёрточная сеть для классификации изображений AlexNet



- Архитектура сети — структура слоёв и связей между ними, позволяющая наделять DNN нужными свойствами
- DNN позволяют принимать на входе и генерировать на выходе *сложно структурированные данные*

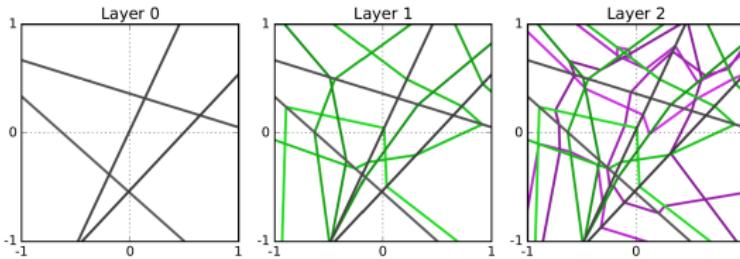
Ивахненко А. Г., Лапа В. Г. Кибернетические предсказывающие устройства. 1965.
Krizhevsky A. et al. ImageNet classification with deep convolutional neural networks. 2012.

Глубина важнее ширины

A_{LH}^n — семейство полносвязных многослойных сетей $f(x, W)$:
 L слоёв, H нейронов в каждом слое, $x \in \mathbb{R}^n$, функции
активации кусочно-линейные (ReLU, hard-tanh и т.п.).

Мера разнообразия семейства A_{LH}^n — максимальное число
участков линейности $f(x, W)$ — выпуклых многогранников в \mathbb{R}^n .

Пример. Участки линейности, $n = 2$, $L = 3$, $H = 4$:



Теорема. Разнообразие семейства A_{LH}^n растёт как $O(H^{nL})$.

Избыточная параметризация может ускорять сходимость

Рассмотрим t -й шаг SGD: $\mathcal{L}(x_i w) \rightarrow \min_w, \quad x_i, w \in \mathbb{R}^n, \quad i \equiv i(t)$:

$$w^{t+1} := w^t - \eta x_i \mathcal{L}'(x_i w^t)$$

Пример избыточной параметризации: $\mathcal{L}(x_i w_1 v) \rightarrow \min_{w_1, v}, \quad v \in \mathbb{R}$:

$$w_1^{t+1} := w_1^t - \eta x_i v^t \mathcal{L}'(x_i w_1^t v^t)$$

$$v^{t+1} := v^t - \eta (x_i w_1^t) \mathcal{L}'(x_i w_1^t v^t)$$

Рекуррентная формула для $w^t = w_1^t v^t$:

$$w^{t+1} := w_1^{t+1} v^{t+1} = w^t - \eta^t x_i \mathcal{L}'(x_i w^t) - \sum_{\tau=1}^{t-1} \eta^{t,\tau} x_{i(\tau)} \mathcal{L}'(x_{i(\tau)} w^\tau)$$

Это (неожиданно!) метод Momentum с адаптивным шагом η^t и адаптивными коэффициентами сглаживания $\eta^{t,\tau}$.

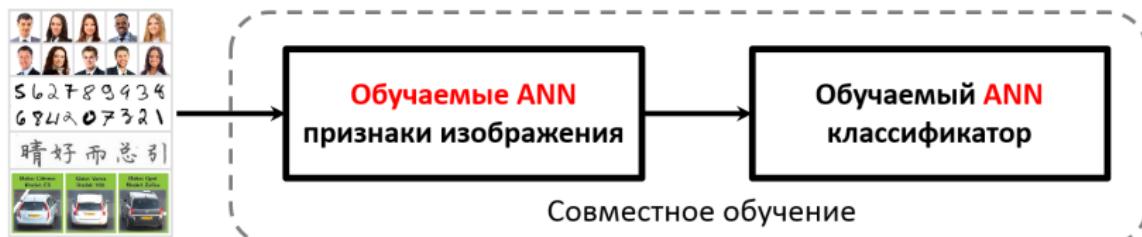
Sanjeev Arora, Nadav Cohen, Elad Hazan. On the Optimization of Deep Networks: Implicit Acceleration by Overparameterization. 2018

Генерация признаков для распознавания изображений

Классический подход к распознаванию изображений:



Современный подход — end-to-end deep learning:



Sanjeev Arora. Toward theoretical understanding of deep learning. ICML-2018 Tutorial
<https://unsupervised.cs.princeton.edu/deeplearningtutorial.html>

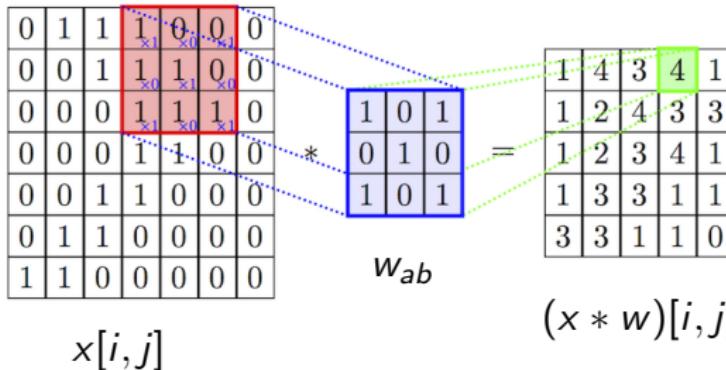
Свёрточный слой нейронов (convolution layer)

$x[i, j]$ — исходные признаки, пиксели $n \times m$ -изображения

w_{ab} — ядро свёртки, $a = -A, \dots, +A$, $b = -B, \dots, +B$

Неполносвязный свёрточный нейрон с $(2A+1)(2B+1)$ весами:

$$(x * w)[i, j] = \sum_{a=-A}^A \sum_{b=-B}^B w_{ab} x[i+a, j+b]$$



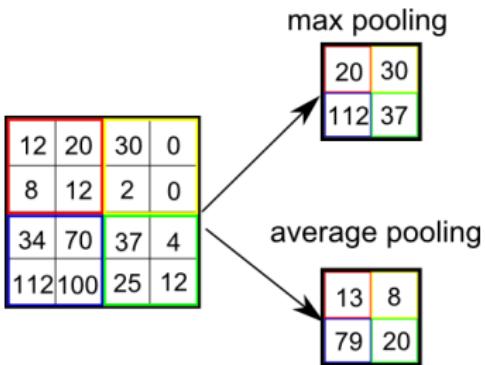
Объединяющий слой нейронов (pooling layer)

Объединяющий нейрон — это необучаемая свёртка с шагом $h > 1$, агрегирующая данные прямоугольной области $h \times h$:

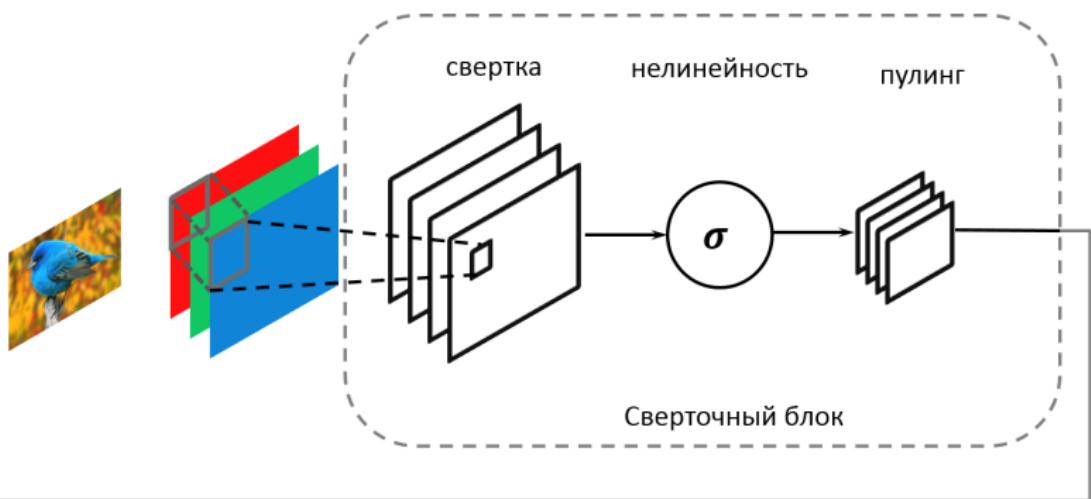
$$y[i, j] = F(x[hi, hj], \dots, x[hi + h - 1, hj + h - 1]),$$

где F — агрегирующая функция: max, average и т.п.

max-pooling позволяет обнаружить элемент в любой из ячеек

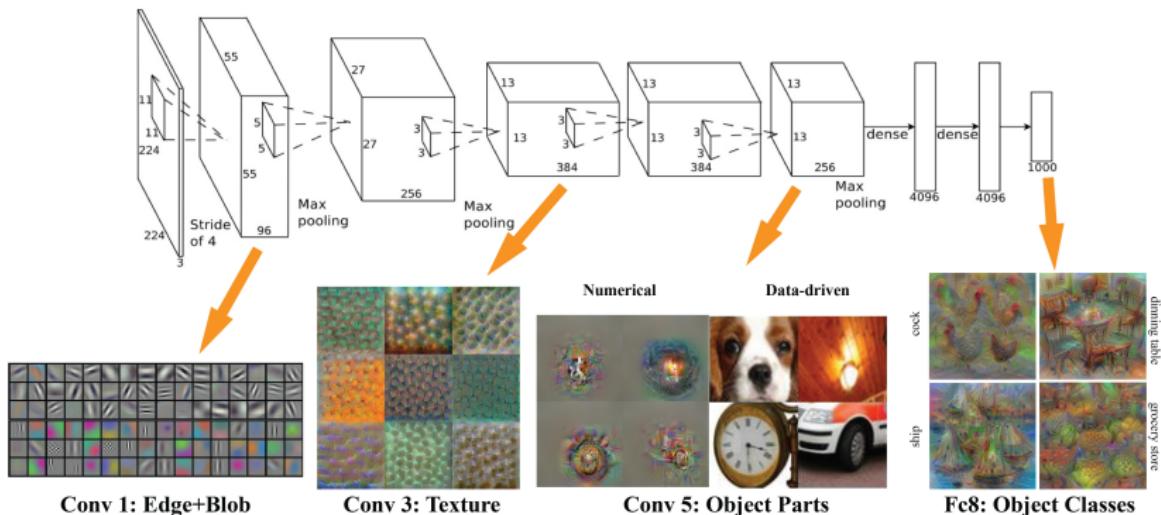


Стандартная схема сверточной сети (Convolutional NN)



Свёрточная сеть обучается извлечению признаков

Чем выше слой, тем более крупные и сложные элементы изображений он способен распознавать



Krizhevsky A., Sutskever I., Hinton G. ImageNet Classification with Deep Convolutional Neural Networks. 2012.

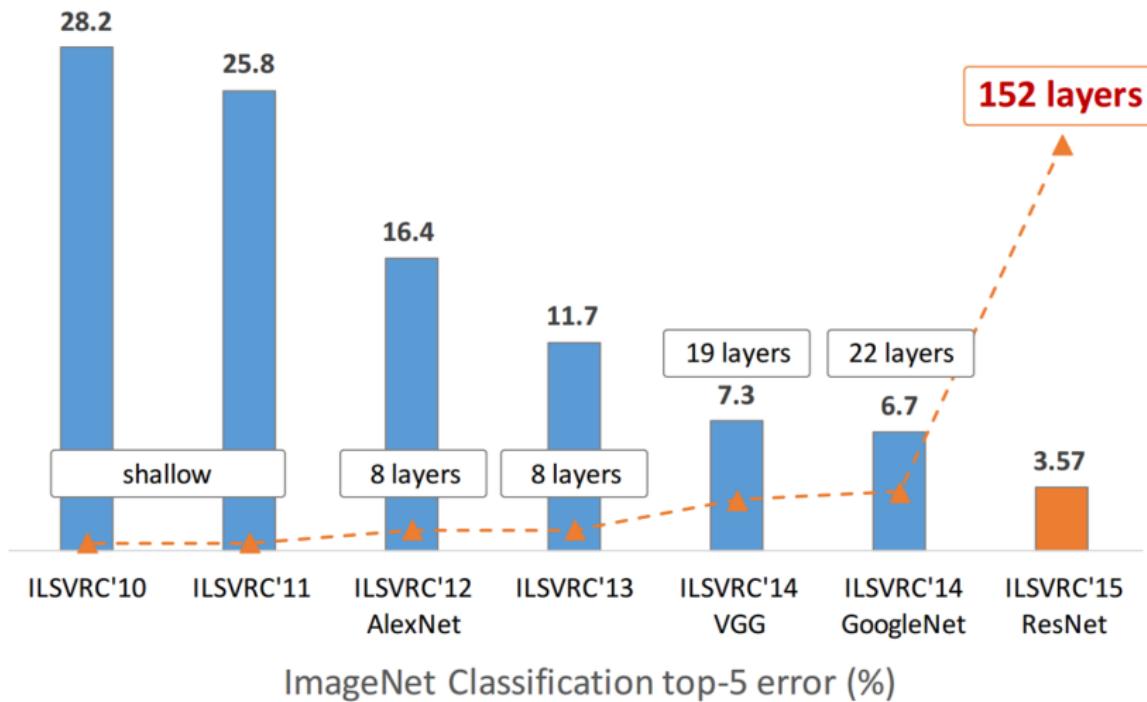
ImageNet — большая выборка размеченных изображений



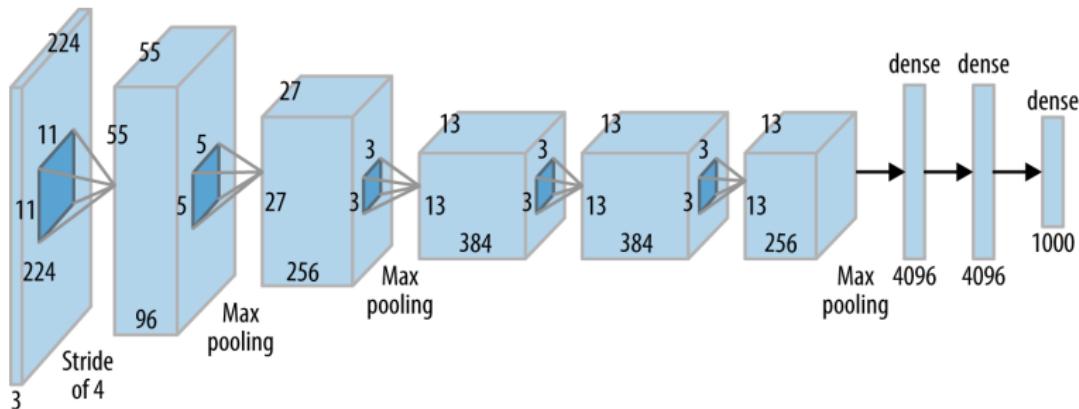
Li Fei-Fei et al. ImageNet: A large-scale hierarchical image database. 2009.

Li Fei-Fei et al. Construction and analysis of a large scale image ontology. 2009.

Развитие свёрточных сетей (краткая история ImageNet)



AlexNet: первый глубокий прорыв на ImageNet



- ReLU + Dropout + пополнение выборки
- 60 млн параметров (в основном в полносвязных слоях)
- Подбор размеров фильтров и пулинга
- GPU

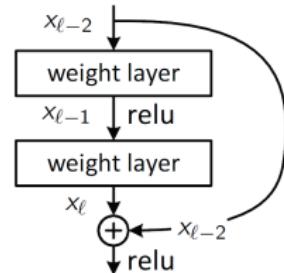
Krizhevsky A., Sutskever I., Hinton G. ImageNet Classification with Deep Convolutional Neural Networks. 2012.

ResNet: остаточная нейронная сеть (Residual NN)

Сквозная связь (skip connection) слоя ℓ с предшествующим слоем $\ell - d$:

$$x_\ell = \sigma(Wx_{\ell-1}) + x_{\ell-d}$$

Слой ℓ выучивает не новое векторное представление x_ℓ , а его приращение $x_\ell - x_{\ell-d}$



- Приращения более устойчивы \Rightarrow улучшается сходимость
- Появляется возможность увеличивать число слоёв
- Обобщение — Highway Networks:

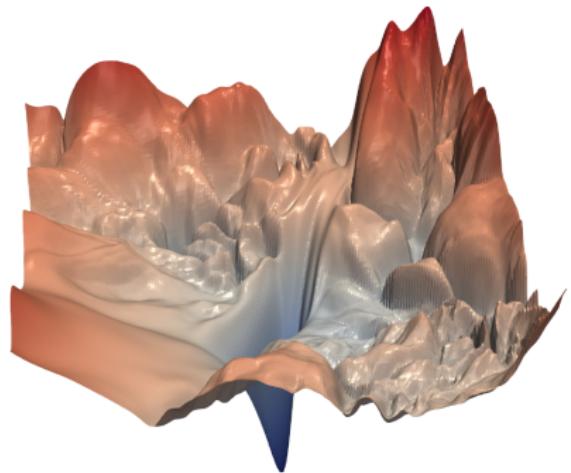
$$x_\ell = \sigma(Wx_{\ell-1}) \underbrace{\tau(W'x_{\ell-1})}_{\text{transform gate}} + x_{\ell-d} \underbrace{(1 - \tau(W'x_{\ell-1}))}_{\text{carry gate}}$$

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition. 2015

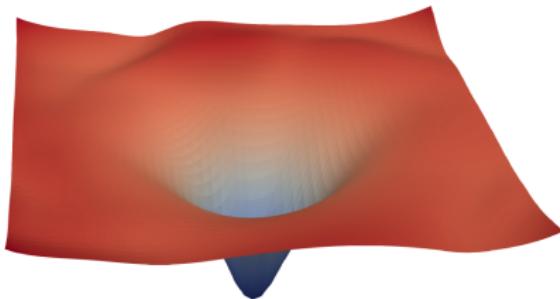
R.K.Srivastava, K.Greff, J.Schmidhuber. Highway Networks. 2015

ResNet: визуализация оптимизационного критерия

Сквозные связи упрощают оптимизируемый критерий, устранивая локальные экстремумы и седловые точки:



without skip connections



with skip connections

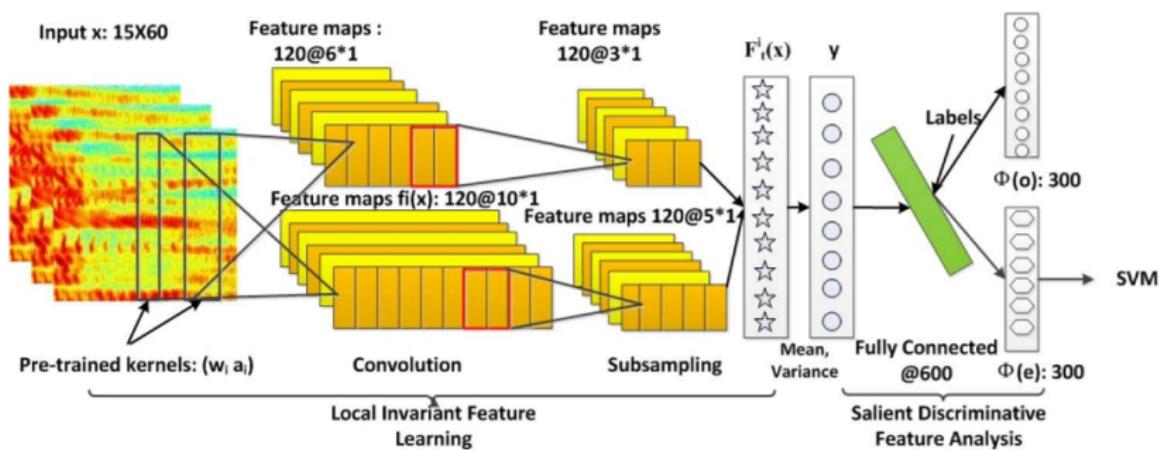
Часто используемые приёмы в CNN

- функции активации без горизонтальных асимптот, типа ReLU
- адаптивные градиентные методы
- dropout
- batch normalization
- остаточные нейронные сети (Residual NN)
- подбор числа слоёв и их размеров
- dataset augmentation — пополнение выборки с помощью преобразований, сохраняющих класс объекта



Приложение: распознавание речевых сигналов

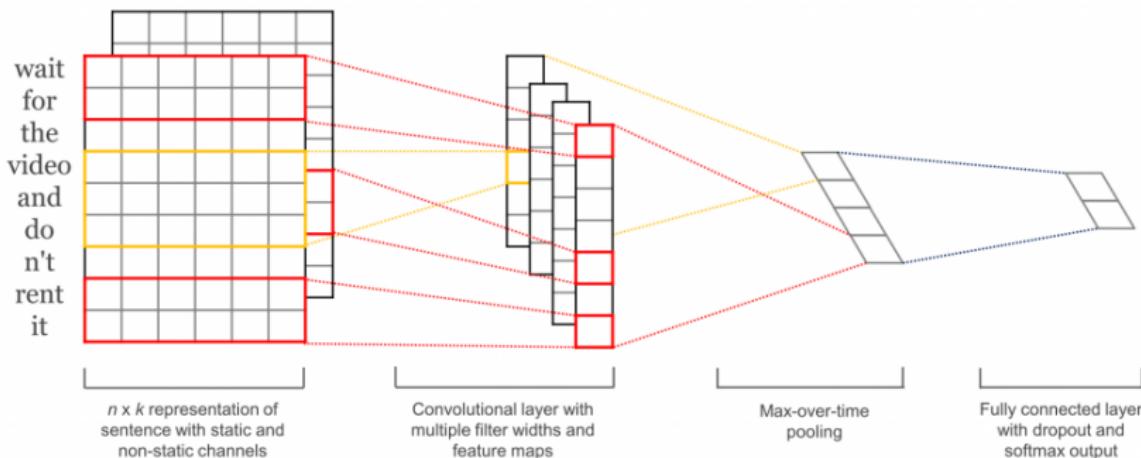
Последовательные фрагменты сигнала представляются векторами спектрального разложения



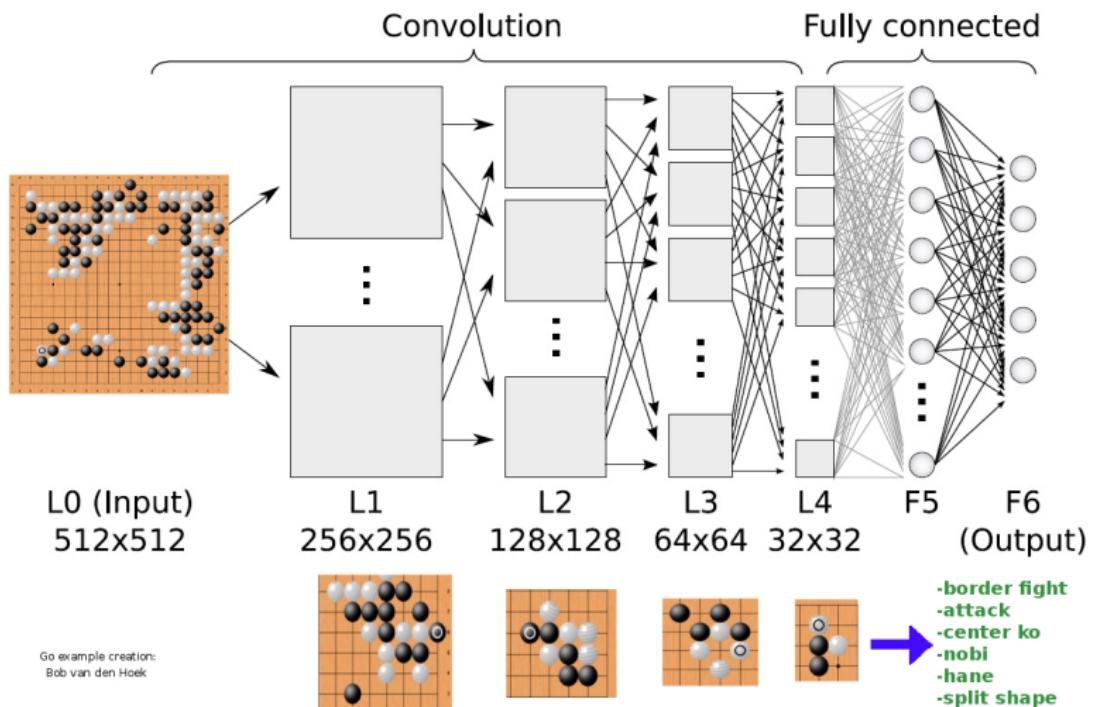
Qirong Mao, Ming Dong, Zhengwei Huang, Yongzhao Zhan. Learning salient features for speech emotion recognition using convolutional neural networks. 2014.

Приложение: классификация предложений в тексте

Последовательные слова в тексте представляются векторами с помощью векторных представлений (word2vec и др.)

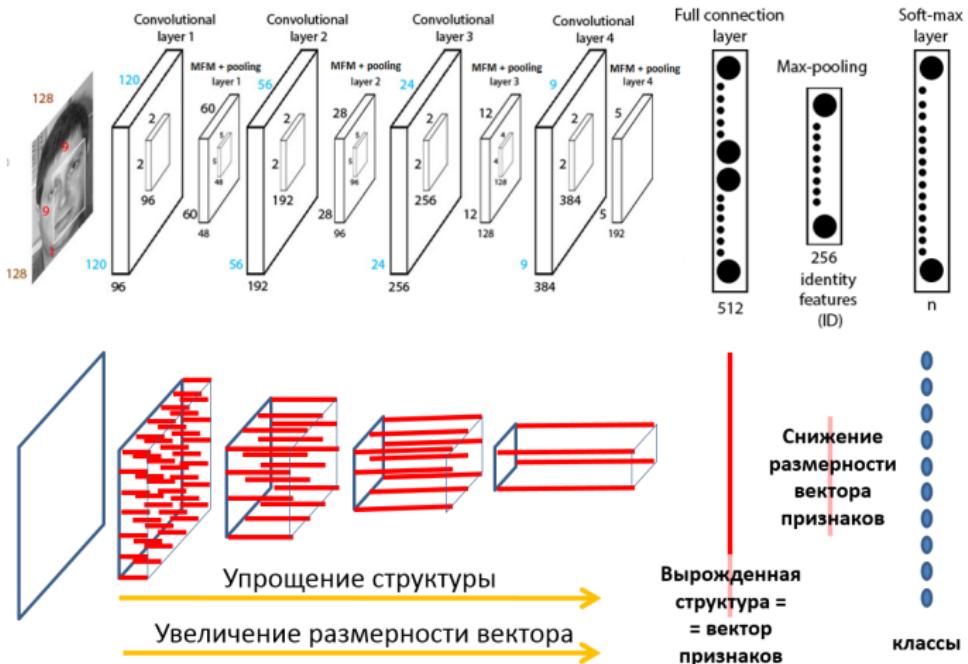


Приложение: принятие решений в логических играх



D.Silver et al. (DeepMind) Mastering the game of Go without human knowledge. 2017.

Глубокая свёрточная сеть как способ векторизации изображений



Визильтер Ю.В., Горбацевич В.С. Структурно-функциональный анализ и синтез глубоких конволюционных нейронных сетей. ММРО-2017.

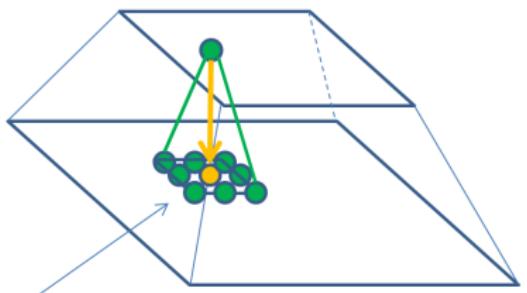
Идея обобщения CNN на любые структурированные данные

Допустим, каждый объект имеет структуру, заданную графом

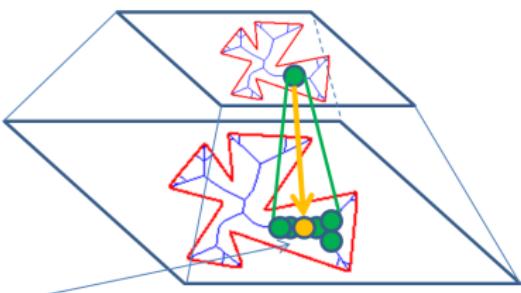
Свёртка определяется по локальной окрестности вершины

Пулинг агрегирует векторы вершин локальной окрестности

Такая сеть обучается находить и классифицировать подграфы



Прямоугольное окно заданного размера с центром в заданной точке +
+ операция свёртки по окну



Локальная окрестность, определяемая для любой вершины графа +
+ операция свёртки по окрестности

Задачи обработки последовательностей

x_t — входной вектор в момент t

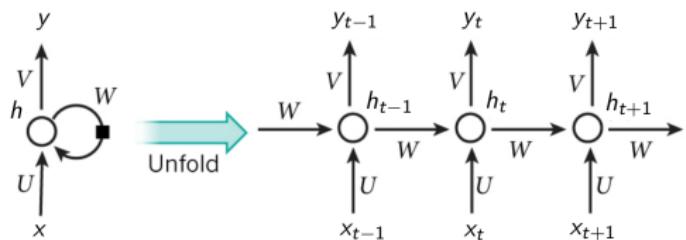
h_t — вектор скрытого состояния в момент t

y_t — выходной вектор (в некоторых приложениях $y_t \equiv h_t$)

Разворачивание (unfolding) рекуррентной сети

$$h_t = \sigma_h(Ux_t + Wh_{t-1})$$

$$y_t = \sigma_y(Vh_t)$$



Обучение рекуррентной сети:

$$\sum_{t=0}^T \mathcal{L}_t(U, V, W) \rightarrow \min_{U, V, W}$$

$\mathcal{L}_t(U, V, W) = \mathcal{L}(y_t(U, V, W))$ — потеря от предсказания y_t

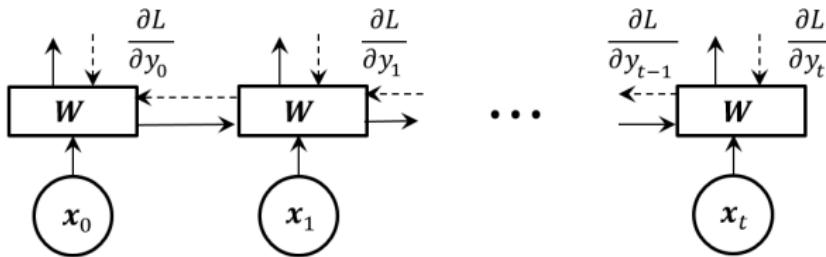
Приложения рекуррентных нейронных сетей

- Прогнозирование временных рядов
- Управление технологическими процессами
- Классификация текстов или их фрагментов
- Анализ тональности документа / предложений / слов
- Машинный перевод
- Распознавание речи
- Синтез речи
- Синтез ответов на вопросы, разговорный интеллект
- Генерация подписей к изображениям
- Генерация рукописного текста
- Интерпретация генома и другие задачи биоинформатики

Обучение рекуррентных сетей

Специальный вариант обратного распространения ошибок,
Backpropagation Through Time (BPTT)

$$\frac{\partial \mathcal{L}_t}{\partial W} = \frac{\partial \mathcal{L}_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \sum_{k=0}^t \left(\prod_{i=k+1}^t \frac{\partial h_i}{\partial h_{i-1}} \right) \frac{\partial h_k}{\partial W}$$

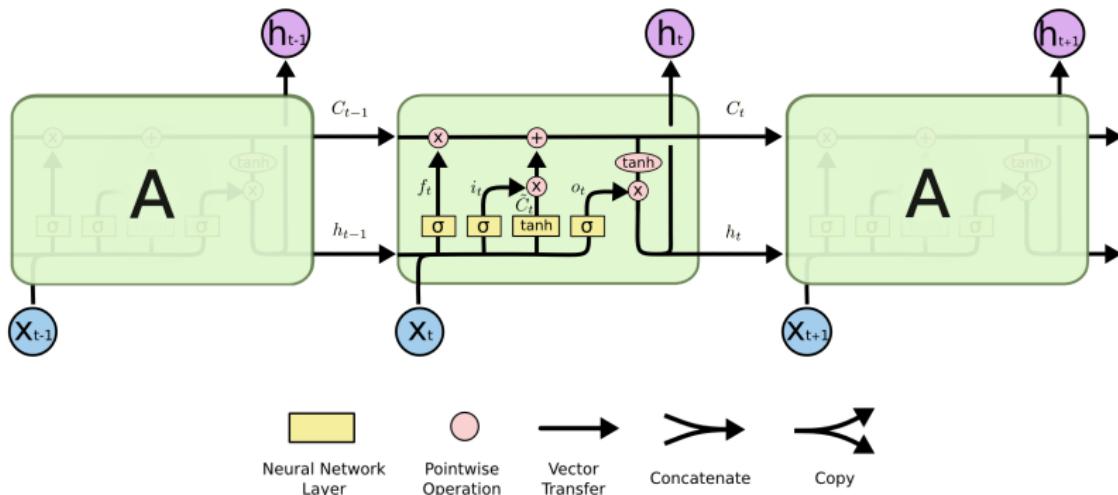


Для предотвращения затухания и взрыва градиентов: $\frac{\partial h_i}{\partial h_{i-1}} \rightarrow 1$

Сети долгой кратковременной памяти (long short-term memory)

Мотивация LSTM: сеть должна долго помнить контекст, какой именно — сеть должна выучить сама.

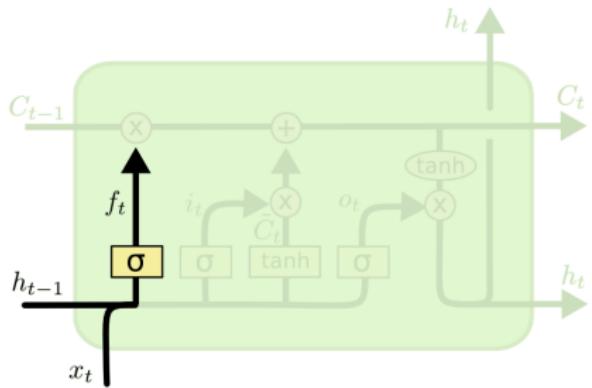
Вводится C_t — вектор состояния сети в момент t .



Hochreiter S., Schmidhuber J. Neural Computation, 9(8), 1997

Greff K., Schmidhuber J. <http://arxiv.org/pdf/1503.04069.pdf>, 2015

Сети долгой кратковременной памяти (long short-term memory)



$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \text{th}(W_C \cdot [h_{t-1}, x_t] + b_C)$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

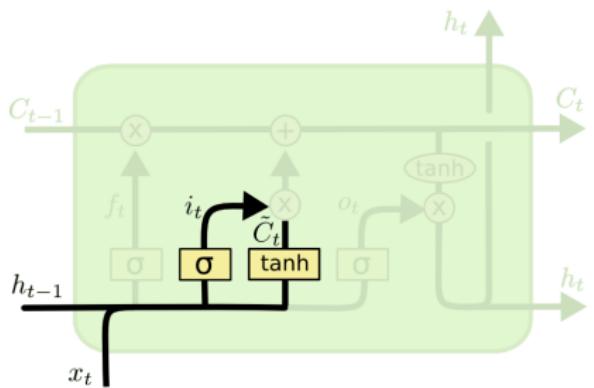
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t \odot \text{th}(C_t)$$

Фильтр забывания (forget gate) с параметрами W_f , b_f решает, какие координаты вектора состояния C_{t-1} надо запомнить.

- — операция покомпонентного перемножения векторов,
- $[h_{t-1}, x_t]$ — конкатенация векторов,
- σ — сигмоидная функция.

Сети долгой кратковременной памяти (long short-term memory)



$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \text{th}(W_C \cdot [h_{t-1}, x_t] + b_C)$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

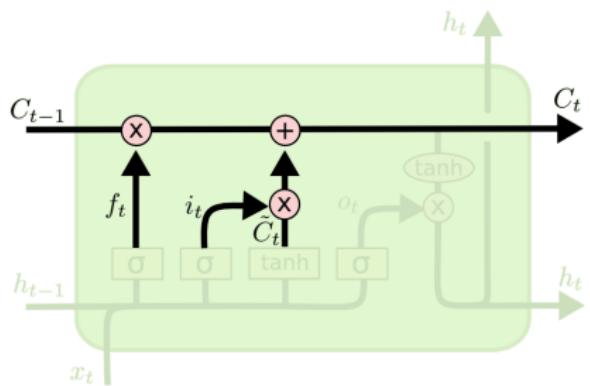
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t \odot \text{th}(C_t)$$

Фильтр входных данных (input gate) с параметрами W_i, b_i решает, какие координаты вектора состояния надо обновить.

Модель нового состояния с параметрами W_C, b_C формирует вектор \tilde{C}_t значений-кандидатов нового состояния.

Сети долгой кратковременной памяти (long short-term memory)

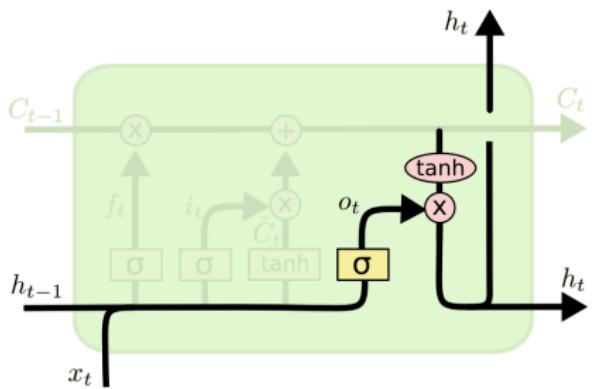


$$\begin{aligned}
 f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\
 i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\
 \tilde{C}_t &= \text{th}(W_C \cdot [h_{t-1}, x_t] + b_C) \\
 C_t &= f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \\
 o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\
 h_t &= o_t \odot \text{th}(C_t)
 \end{aligned}$$

Новое состояние C_t формируется как смесь старого состояния C_{t-1} с фильтром f_t и вектора значений-кандидатов \tilde{C}_t с фильтром i_t .

Настраиваемых параметров нет.

Сети долгой кратковременной памяти (long short-term memory)

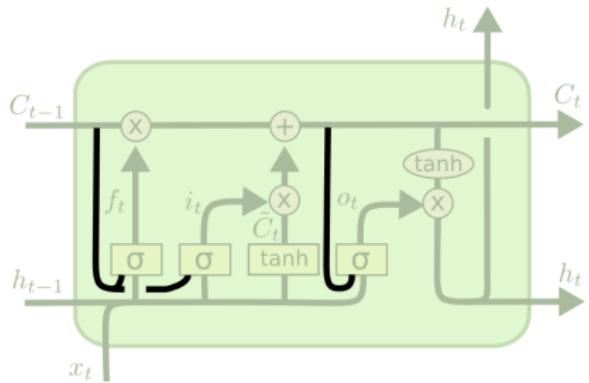


$$\begin{aligned}f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ \tilde{C}_t &= \text{th}(W_C \cdot [h_{t-1}, x_t] + b_C) \\ C_t &= f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \\ o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ h_t &= o_t \odot \text{th}(C_t)\end{aligned}$$

Фильтр выходных данных (output gate) с параметрами W_o , b_o решает, какие координаты вектора состояния C_t надо выдать.

Выходной сигнал h_t формируется из вектора состояния C_t с помощью нелинейного преобразования th и фильтра o_t .

Вариант LSTM с «замочными скважинами» (peepholes)



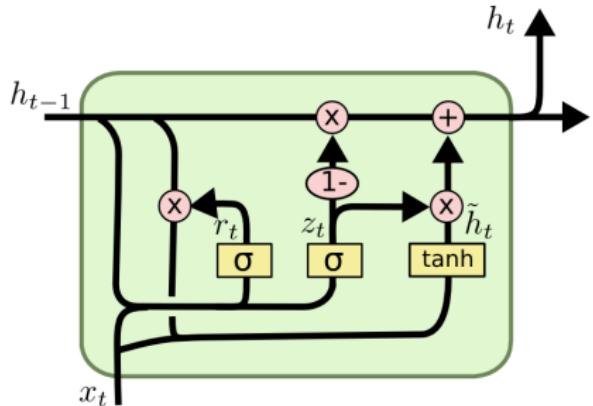
$$\begin{aligned}
 f_t &= \sigma(W_f \cdot [C_{t-1}, h_{t-1}, x_t] + b_f) \\
 i_t &= \sigma(W_i \cdot [C_{t-1}, h_{t-1}, x_t] + b_i) \\
 \tilde{C}_t &= \text{th}(W_C \cdot [h_{t-1}, x_t] + b_C) \\
 C_t &= f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \\
 o_t &= \sigma(W_o \cdot [C_t, h_{t-1}, x_t] + b_o) \\
 h_t &= o_t \odot \text{th}(C_t)
 \end{aligned}$$

Все фильтры «подглядывают» вектор состояния C_{t-1} или C_t .

Увеличивается число параметров модели.

Замочную скважину можно использовать не для всех фильтров.

Упрощение LSTM: Gated Recurrent Unit (GRU)



$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$

$$\tilde{h}_t = \text{th}(W_h \cdot [r_t \odot h_{t-1}, x_t])$$

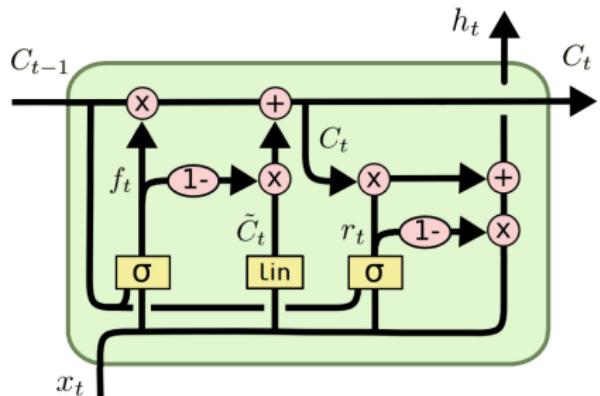
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$$

Используется только состояние h_t , вектор C_t не вводится.

Фильтр обновления (update gate) вместо входного и забывающего.

Фильтр перезагрузки (reset gate) r_t решает, какую часть памяти нужно перенести дальше с прошлого шага.

Упрощение LSTM: Simple Recurrent Unit (SRU)



$$f_t = \sigma(W_f x_t + v_f \odot C_{t-1} + b_f)$$

$$\tilde{C}_t = W_C x_t$$

$$C_t = f_t \odot C_{t-1} + (1 - f_t) \odot \tilde{C}_t$$

$$r_t = \sigma(W_r x_t + v_r \odot C_{t-1} + b_r)$$

$$h_t = r_t \odot C_t + (1 - r_t) \odot x_t$$

С предыдущего шага передаётся только вектор C_{t-1} .

Два фильтра: забывания (forget gate) и перезагрузки (reset gate).

Сквозные связи (skip connections): x_t передаётся на все слои.

Облегчённая рекуррентность: $v_f \odot C_{t-1}$ вместо $W_f C_{t-1}$,

позволяет вычислять координаты векторов параллельно.

Tao Lei et al. Simple recurrent units for highly parallelizable recurrence. 2018.

- *Свёрточные сети*: векторизация сложно структурированных данных, обучаемая совместно с основной моделью
- *Рекуррентные сети*: обучаемые преобразования входной последовательности в выходную (seq2seq)
- Приёмы, сделавшие возможным глубокое обучение:
 - продвинутые градиентные методы ускоряют сходимость
 - регуляризации и dropout предотвращают переобучение
 - batch norm сокращает вычислительные погрешности
 - augmentation обеспечивает устойчивость кискажениям
 - ReLU предотвращает затухание и взрыв градиентов
 - свёртки и разреживание сокращают число параметров
 - skip connections позволяют увеличивать глубину
- Переход от feature engineering к architecture engineering
- Подбор архитектуры и гиперпараметров всё ещё искусство