• Прикладные модели машинного обучения •
Глубокие нейронные сети

Воронцов Константин Вячеславович
k.v.vorontsov@phystech.edu
http://www.MachineLearning.ru/wiki?title=User:Vokov

Этот курс доступен на странице вики-ресурса
http://www.MachineLearning.ru/wiki
«Машинное обучение (курс лекций, К.В.Воронцов)»

МФТИ • 7 февраля 2025

Содержание

Напоминание: линейные модели классификации и регрессии

Обучающая выборка: $X^\ell = (x_i, y_i)_{i=1}^\ell$, объекты $x_i \in \mathbb{R}^n$, ответы y_i

Задача регрессии, $Y = \mathbb{R}$:
$a(x, w) = \langle w, x_i \rangle$ — линейная модель регрессии

$$Q(w; X^\ell) = \sum_{i=1}^{\ell} \big(\langle w, x_i \rangle - y_i\big)^2 \to \min_w$$

Задача классификации, $Y = \{\pm 1\}$:
$a(x, w) = \text{sign}\langle w, x_i \rangle$ — линейная модель классификации

$$Q(w; X^\ell) = \sum_{i=1}^{\ell} L\big(\underbrace{\langle w, x_i \rangle y_i}_{M_i(w)}\big) \to \min_w$$

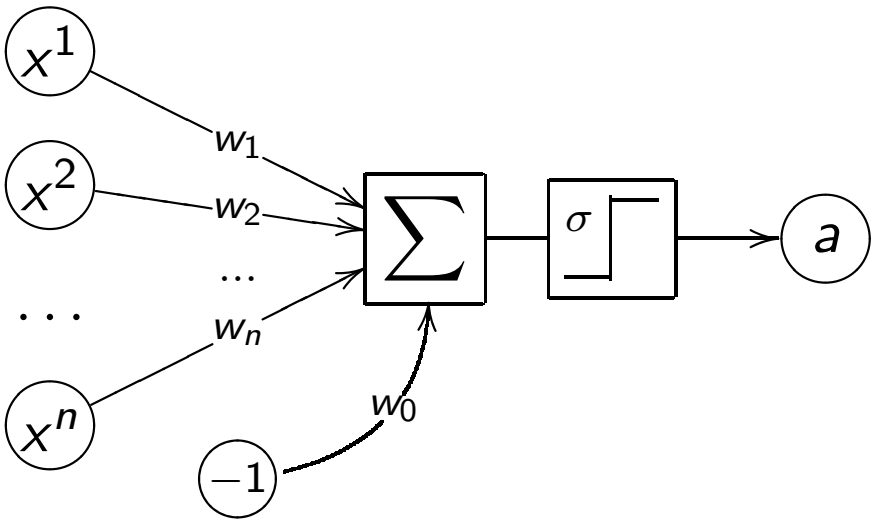
$L(M)$ — невозрастающая функция отступа, например,
$L(M) = \ln(1 + e^{-M})$, $(1 - M)_+$, e^{-M}, $\frac{1}{1+e^M}$, и др.

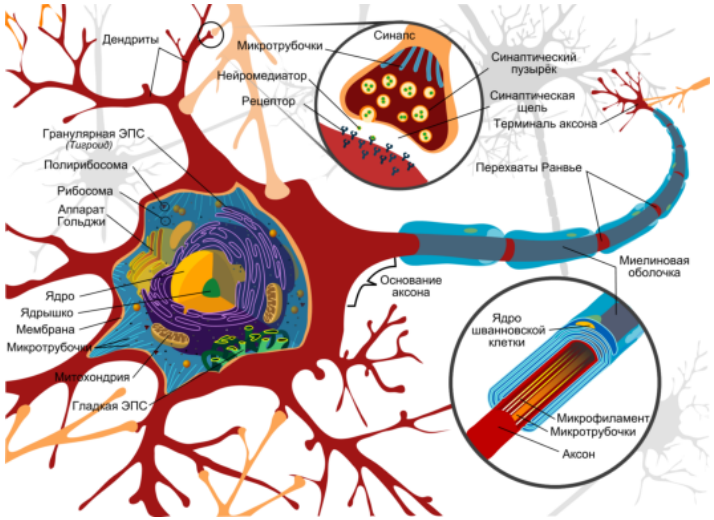
Напоминание: математическая модель нейрона

Линейная модель нейрона МакКаллока-Питтса [1943]:

$$a(x,w) = \sigma\left(\sum_{j=1}^{n} w_j f_j(x) - w_0\right) = \sigma\big(\langle w, x\rangle\big), \quad w, x \in \mathbb{R}^{n+1},$$

$\sigma(z)$ — функция активации (например, sign или th),
w_j — вес признака $f_j(x)$,
w_0 — вес признака $f_0(x) \equiv -1$, он же порог активации.





Напоминание: полносвязная нейронная сеть с L слоями

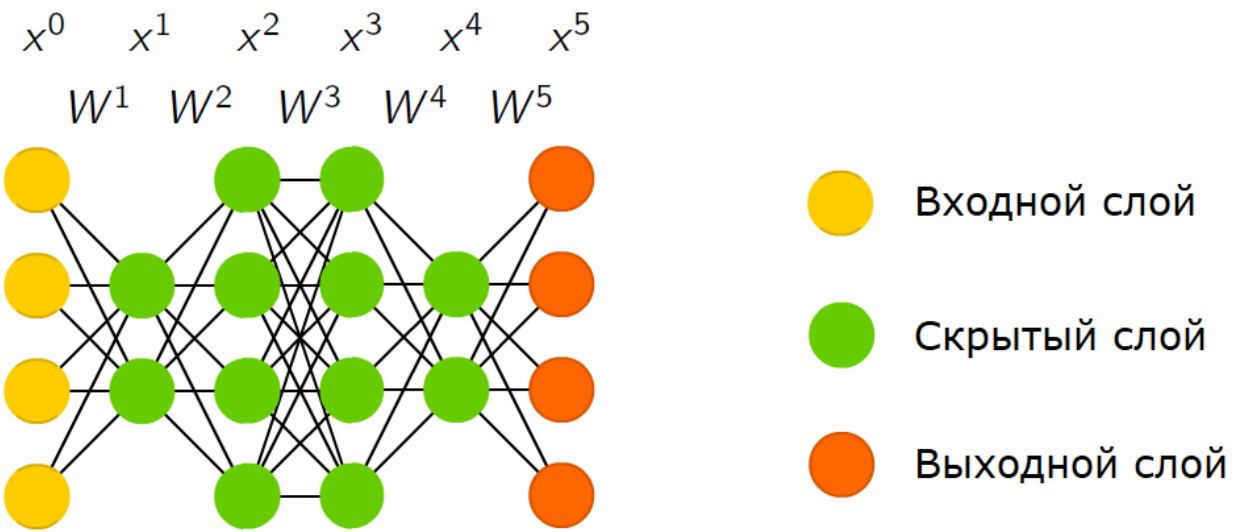
Архитектура сети: H_l — число нейронов в l-м слое, $l = 1, \ldots, L$

$x^0 = x = \big(f_j(x)\big)_{j=0}^{n}$ — вектор признаков на входе сети, $H_0 = n$

$x^l = \big(x_h^l\big)_{h=0}^{H_l}$ — вектор признаков на выходе l-го слоя, $x_0^l = -1$

x^L — выходной вектор сети размерности H_L

$W^l = (w_{kh}^l)$ — матрица весов l-го слоя, размера $(H_{l-1}+1) \times H_l$



Напоминание: алгоритм SG (Stochastic Gradient)

Минимизация средних потерь на обучающей выборке:

$$Q(w) = \frac{1}{\ell}\sum_{i=1}^{\ell}\mathscr{L}(w, x_i) \to \min_w.$$

Вход: выборка $(x_i, y_i)_{i=1}^{\ell}$; темп обучения η; параметр λ;
Выход: вектор весов всех слоёв $w = (W^1, \ldots, W^L)$;

инициализировать веса w и текущую оценку $Q(w)$;
повторять
  выбрать объект x_i из X^ℓ (например, случайно);
  вычислить потерю $\mathscr{L}_i := \mathscr{L}(w, x_i)$;
  градиентный шаг: $w := w - \eta\nabla\mathscr{L}(w, x_i)$;
  оценить значение функционала: $Q := (1-\lambda)Q + \lambda\mathscr{L}_i$;
пока значение Q и/или веса w не стабилизируются;

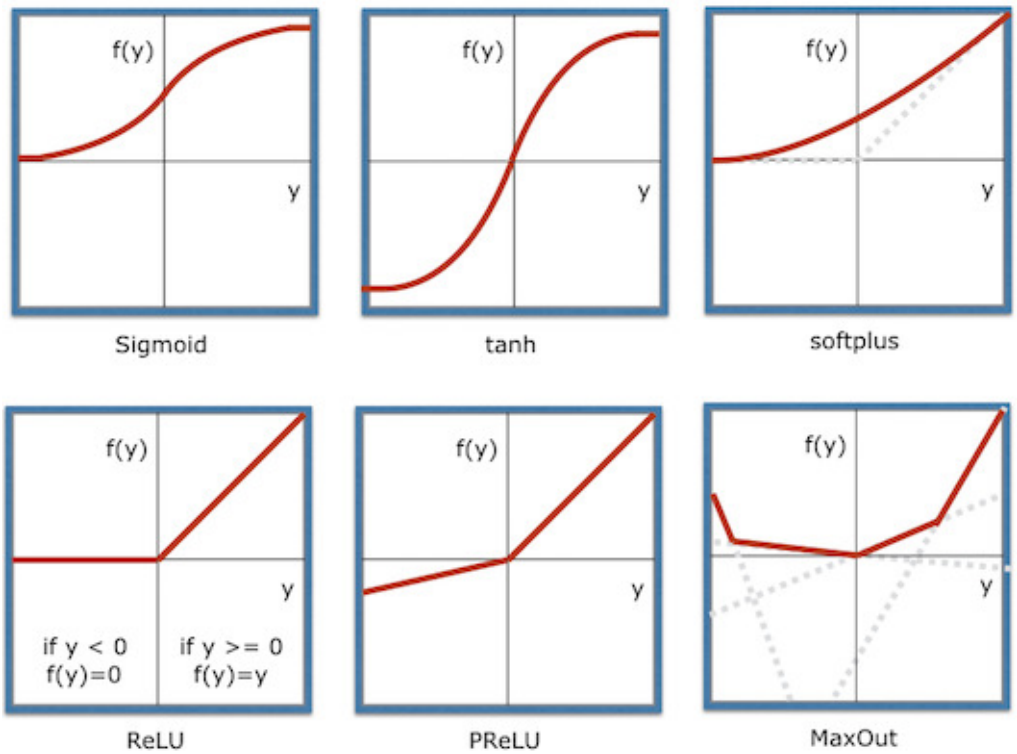
H.Robbins, S.Monro. A stochastic approximation method // Annals of Math. Stat., 1951.

Напоминание. Функции активации

Функции $\sigma(y) = \frac{1}{1+e^{-y}}$ и $\text{th}(y) = \frac{e^{y}-e^{-y}}{e^{y}+e^{-y}}$ могут приводить к затуханию градиентов или «параличу сети»

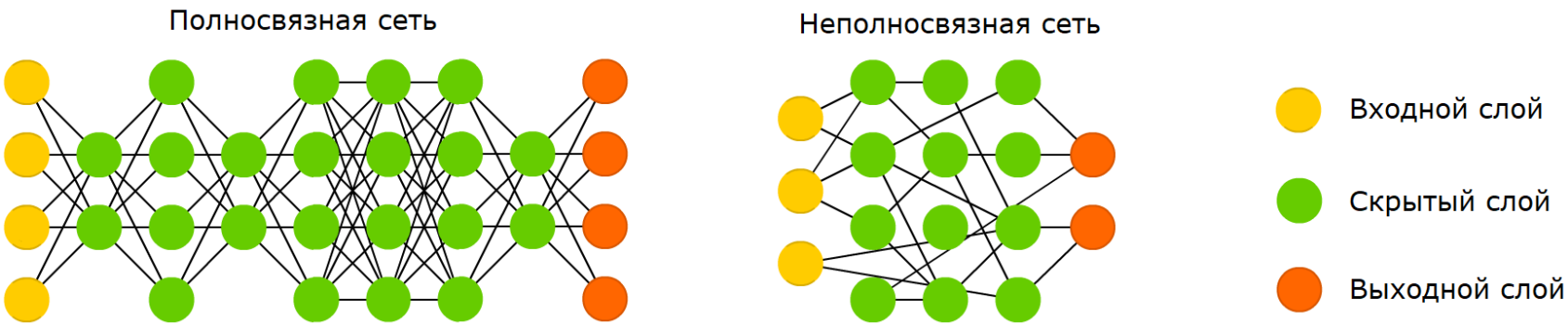
Функция положительной срезки (Rectified Linear Unit, ReLU)

$$\text{ReLU}(y) = \max\{0, y\}; \qquad \text{PReLU}(y) = \max\{0, y\} + \alpha \min\{0, y\}$$



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*

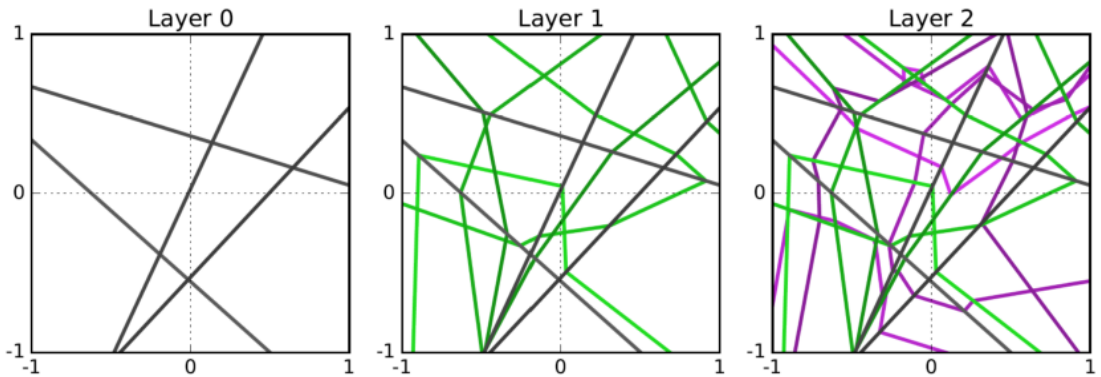
Ива́хненко А. Г., Лапа В. Г. Кибернетические предсказывающие устройства. 1965.
Krizhevsky A. et al. ImageNet classification with deep convolutional neural networks. 2012.

Глубина важнее ширины

A^n_{LH} — семейство полносвязных многослойных сетей $a(x, w)$:
n признаков, 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^n_{LH} — максимальное число участков линейности $a(x, w)$ — выпуклых многогранников в $\mathbb{R}^n$.

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



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

M.Raghu et al. On the Expressive Power of Deep Neural Networks, 2016.

Избыточная параметризация может ускорять сходимость

Рассмотрим t-й шаг SGD: $\mathscr{L}(x_i w) \to \min\limits_{w}$, $x_i, w \in \mathbb{R}^n$, $i \equiv i(t)$:

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

Пример избыточной параметризации: $\mathscr{L}(x_i w_1 v) \to \min\limits_{w_1, v}$, $v \in \mathbb{R}$:

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

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

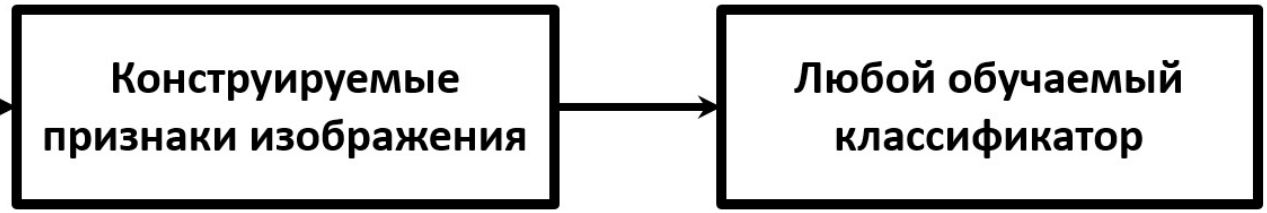
$$w^{t+1} := w_1^{t+1} v^{t+1} = w^t - \eta^t x_i \mathscr{L}'(x_i w^t) - \sum_{\tau=1}^{t-1} \eta^{t,\tau} x_{i(\tau)} \mathscr{L}'(x_{i(\tau)} w^\tau)$$

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

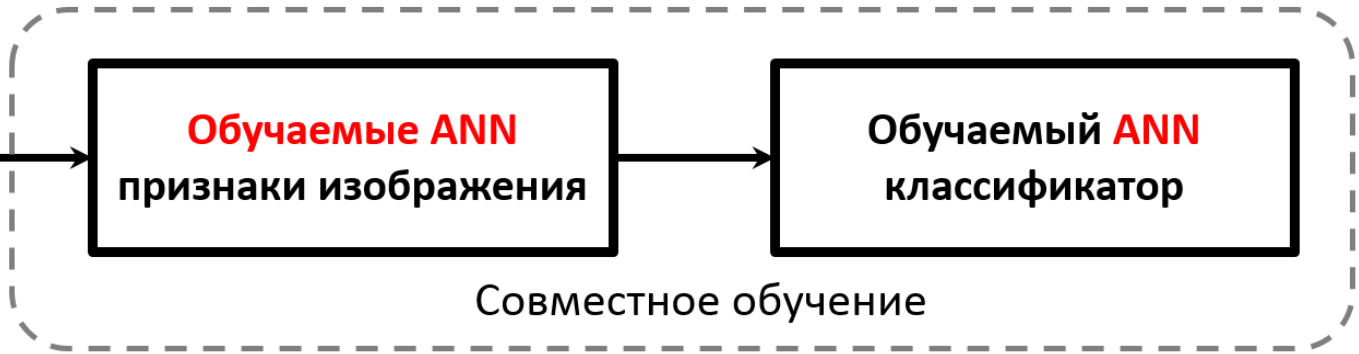
Sanjeev Arora, Nadav Cohen, Elad Hazan. On the Optimization of Deep Networks: Implicit Acceleration by Overparameterization. 2018

Генерация признаков для распознавания изображений

Классический подход к распознаванию изображений:



Современный подход — end-to-end deep learning:



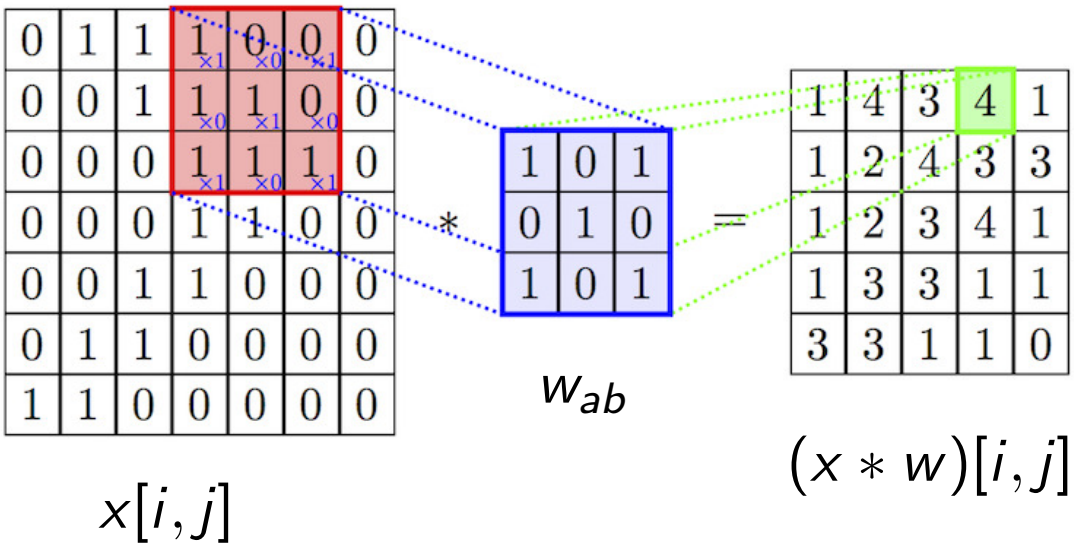
Sanjeev Arora. Toward theoretical understanding of deep learning. ICML-2018 Tutorial
https://unsupervised.cs.princeton.edu/deeplearningtutorial.html

Свёрточный слой нейронов (convolution layer)

$x[i,j]$ — исходные признаки, пиксели $n \times m$-изображения
w_{ab} — ядро свёртки, $a = -A, \ldots, +A$, $b = -B, \ldots, +B$

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

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

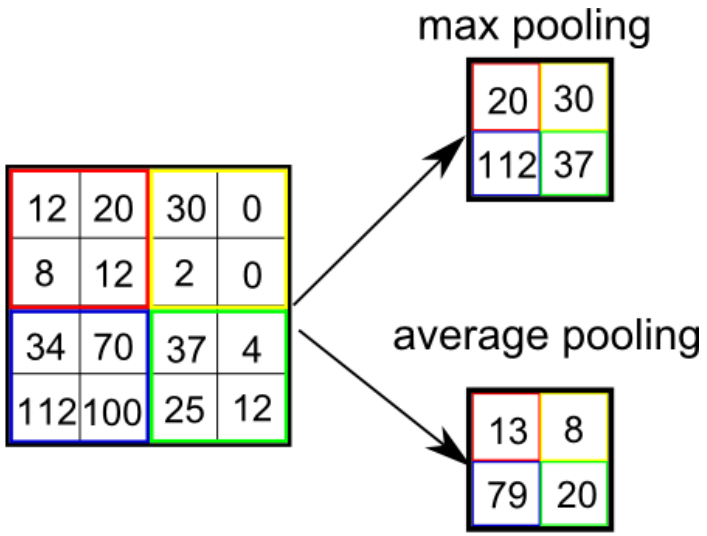


Объединяющий слой нейронов (pooling layer)

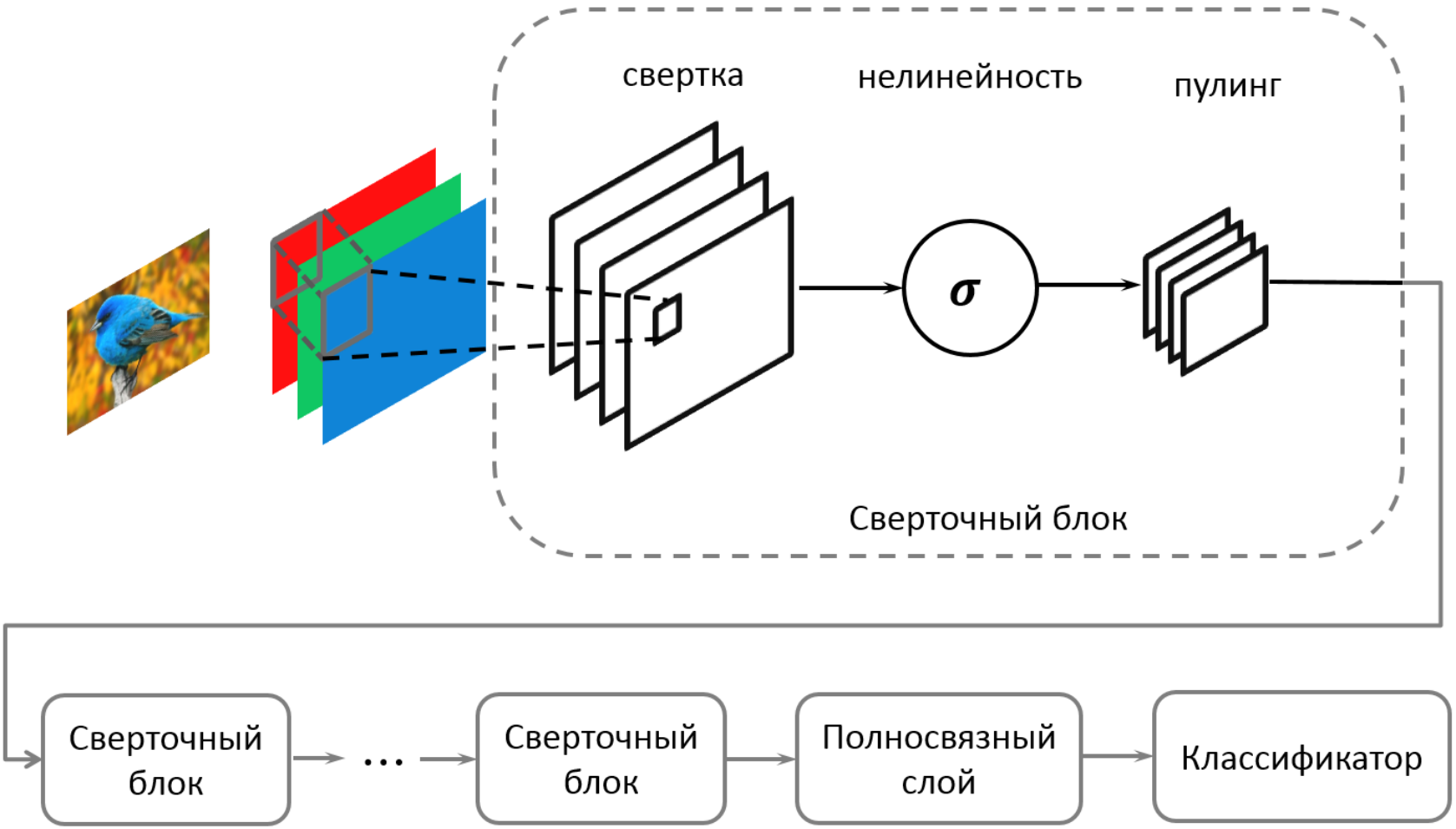
Объединяющий нейрон — это необучаемая свёртка с шагом $h > 1$, агрегирующая данные прямоугольной области $h \times h$:

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

где F — агрегирующая функция: max, average и т.п.
max-pooling позволяет обнаружить элемент в любой из ячеек



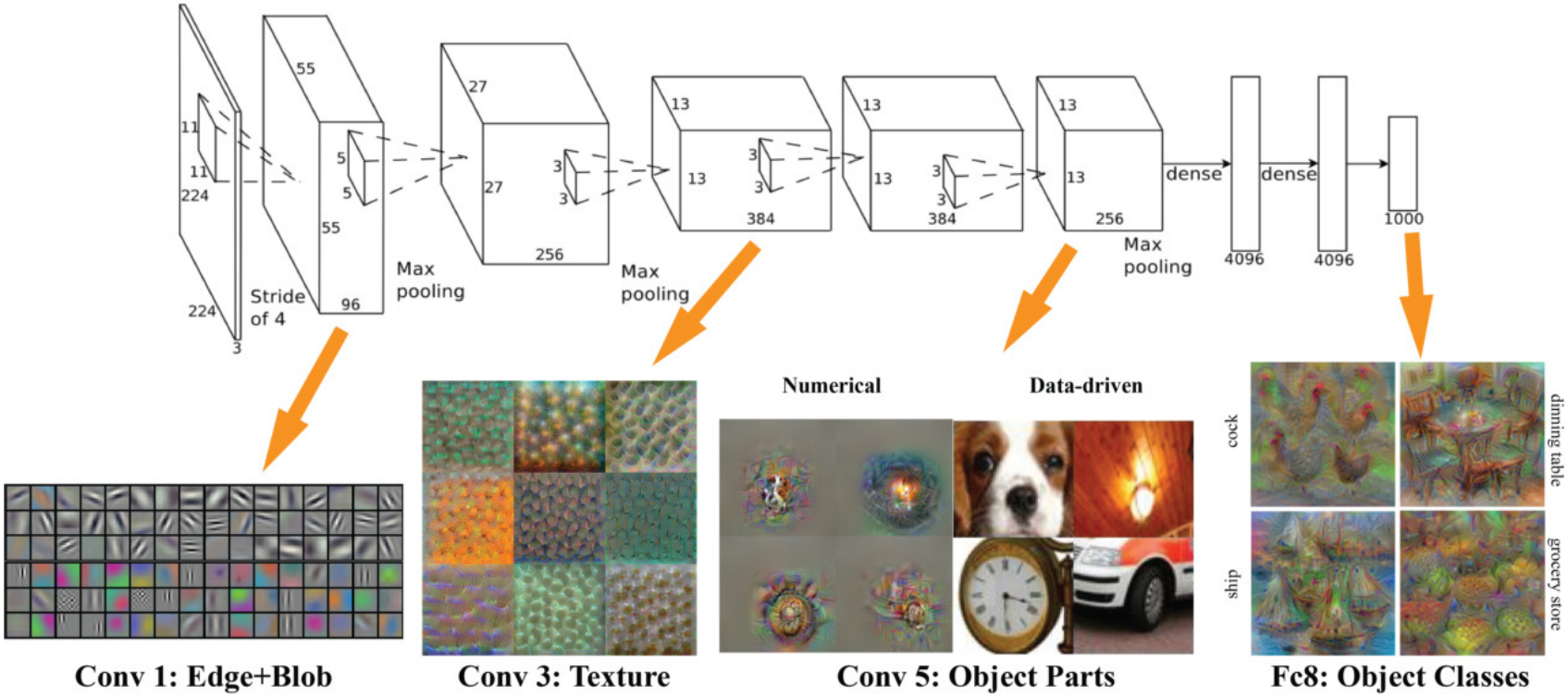
Стандартная схема сверточной сети (Convolutional NN)



Yann LeCun et al. Learning algorithms for classification: A comparison on handwritten digit recognition. 1995

Свёрточная сеть обучается извлечению признаков

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

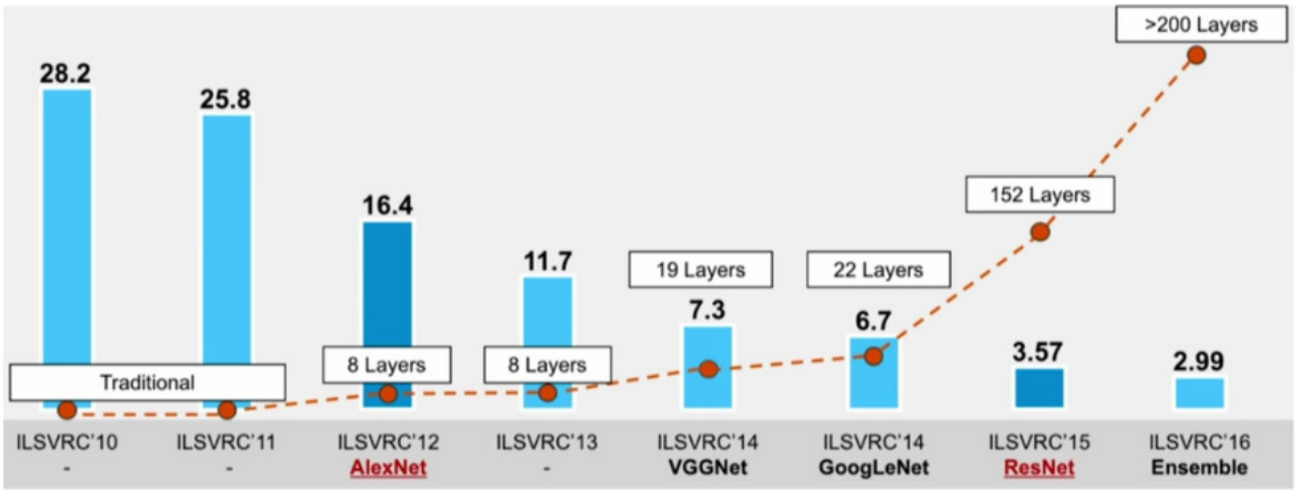


Krizhevsky A., Sutskever I., Hinton G. ImageNet classification with deep convolutional neural networks. 2012.

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.

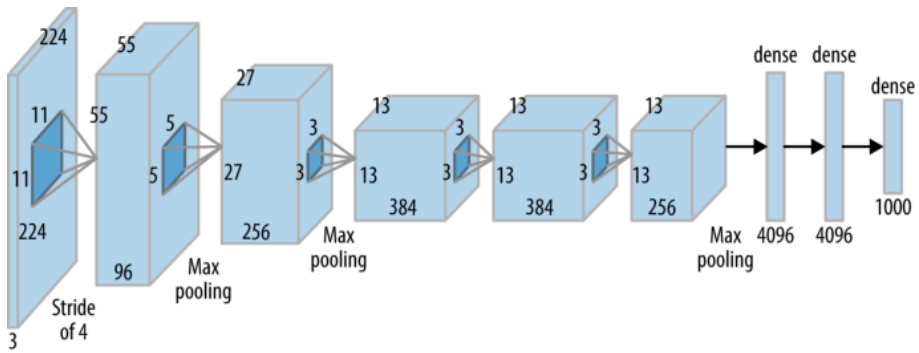
Глубокие свёрточные сети для классификации изображений



Старт в 2009. Человеческий уровень ошибок 5% пройден в 2015

Свёрточная сеть **AlexNet**:

+ ReLU + Dropout
+ 60M параметров
+ пополнение выборки
+ подбор размеров слоёв
+ GPU

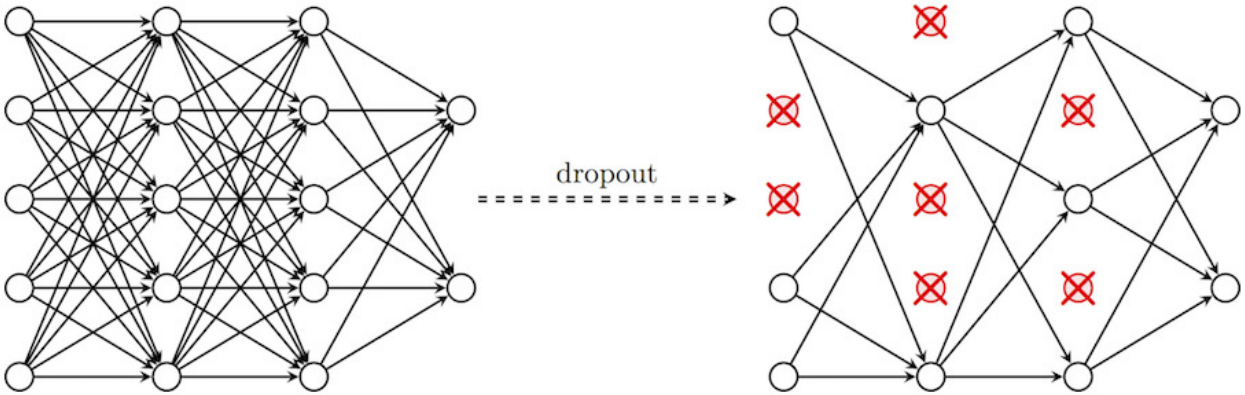


Krizhevsky A. et al. ImageNet classification with deep convolutional neural networks. 2012.

Напоминание. Dropout — случайные отключения нейронов

Этап обучения: делая градиентный шаг $\mathscr{L}(w, x_i) \to \min_w$,
отключаем h-ый нейрон l-го слоя с вероятностью p_l:

$$x_{ih}^{l+1} = \xi_h^l \, \sigma_h\Big(\sum_j w_{jh} x_{ij}^l\Big), \qquad \mathsf{P}(\xi_h^l = 0) = p_l$$



Этап применения: включаем все нейроны, но с поправкой:

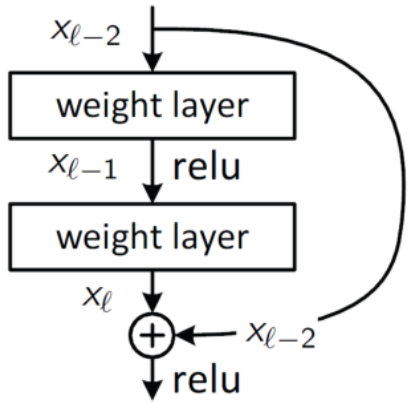
$$x_{ih}^{l+1} = (1 - p_l)\sigma_h\Big(\sum_j w_{jh} x_{ij}^l\Big)$$

N.Srivastava, G.Hinton, A.Krizhevsky, I.Sutskever, R.Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. 2014

ResNet: остаточная нейронная сеть (Residual NN)

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

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



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

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

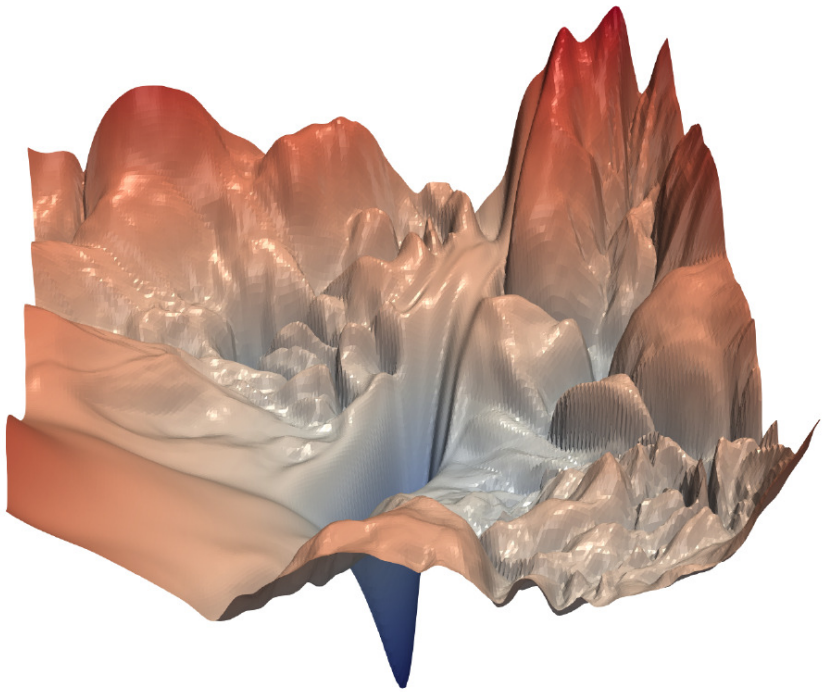
$$x_l = \sigma(Wx_{l-1})\underbrace{\tau(W'x_{l-1})}_{\text{transform gate}} + x_{l-d}\underbrace{\big(1-\tau(W'x_{l-1})\big)}_{\text{carry gate}}$$

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

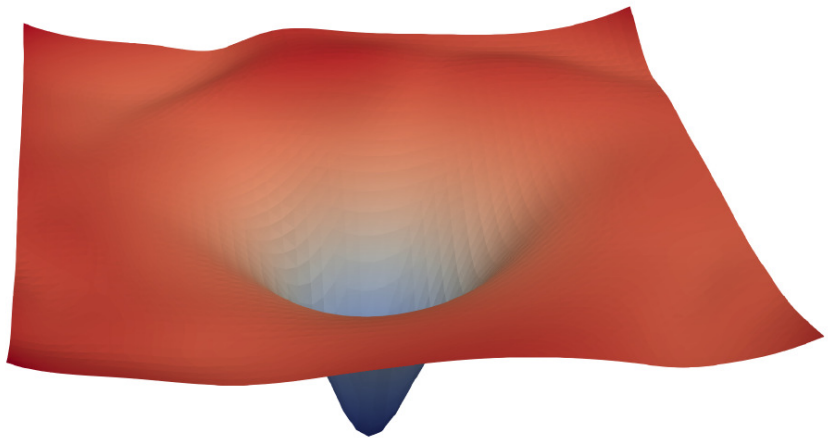
R.K.Srivastava, K.Greff, J.Schmidhuber. Highway Networks. 2015

ResNet: визуализация оптимизационного критерия

Сквозные связи (skip connection) упрощают оптимизируемый критерий, устраняя локальные экстремумы и седловые точки:



without skip connections



with skip connections

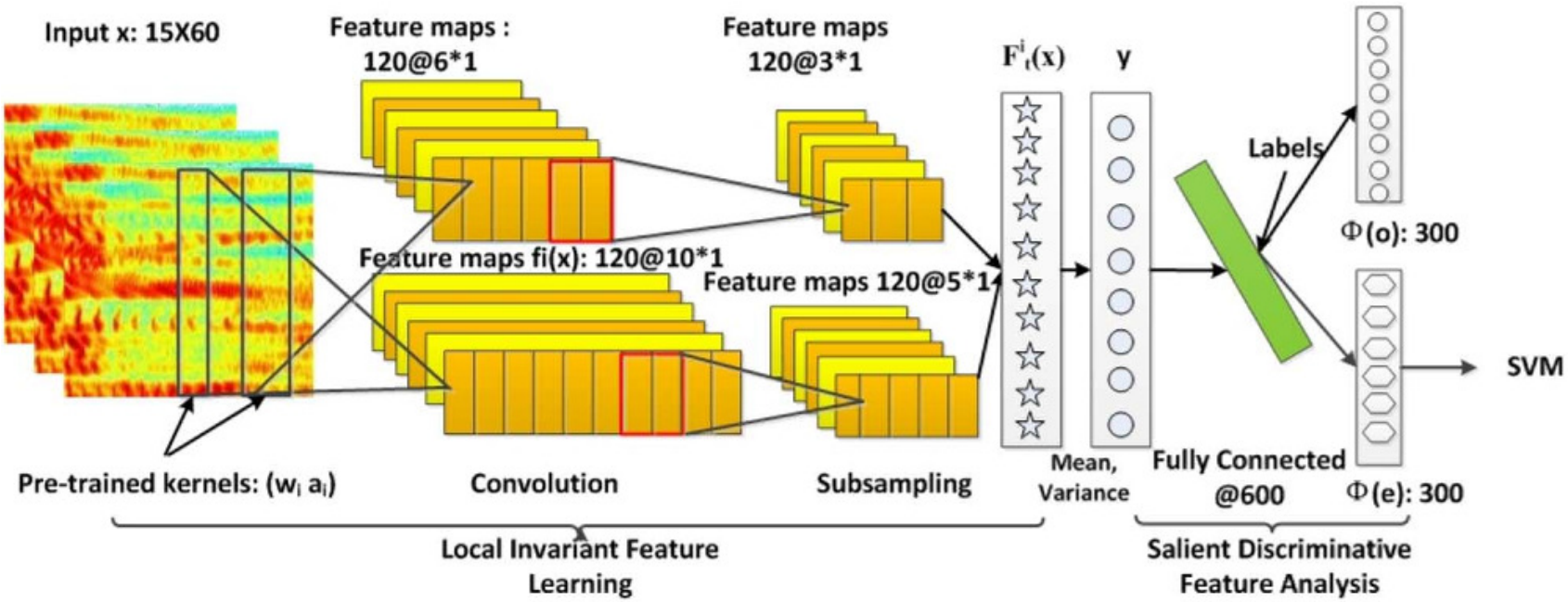
Hao Li et al. Visualizing the Loss Landscape of Neural Nets. 2018

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

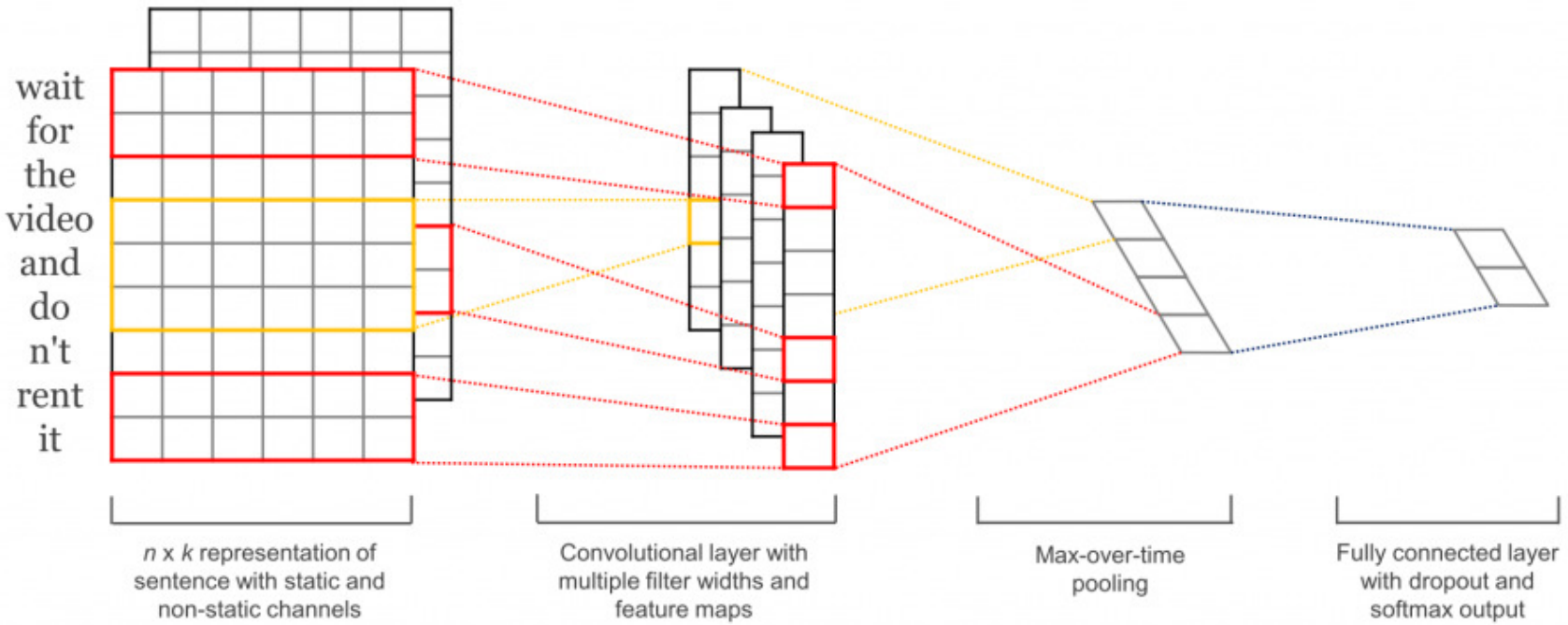
Последовательные фрагменты сигнала представляются векторами спектрального разложения



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

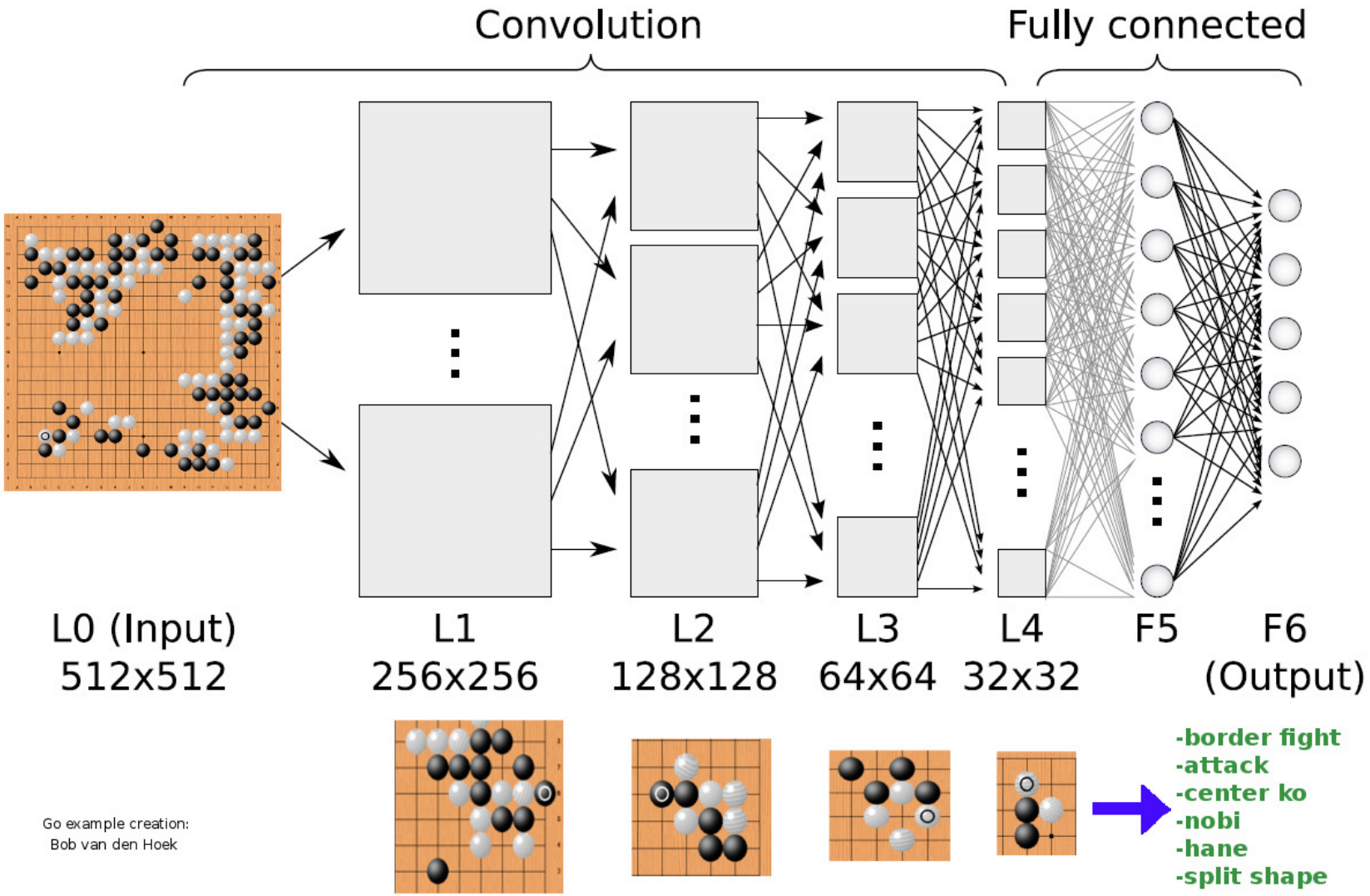
Приложение: классификация предложений в тексте

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



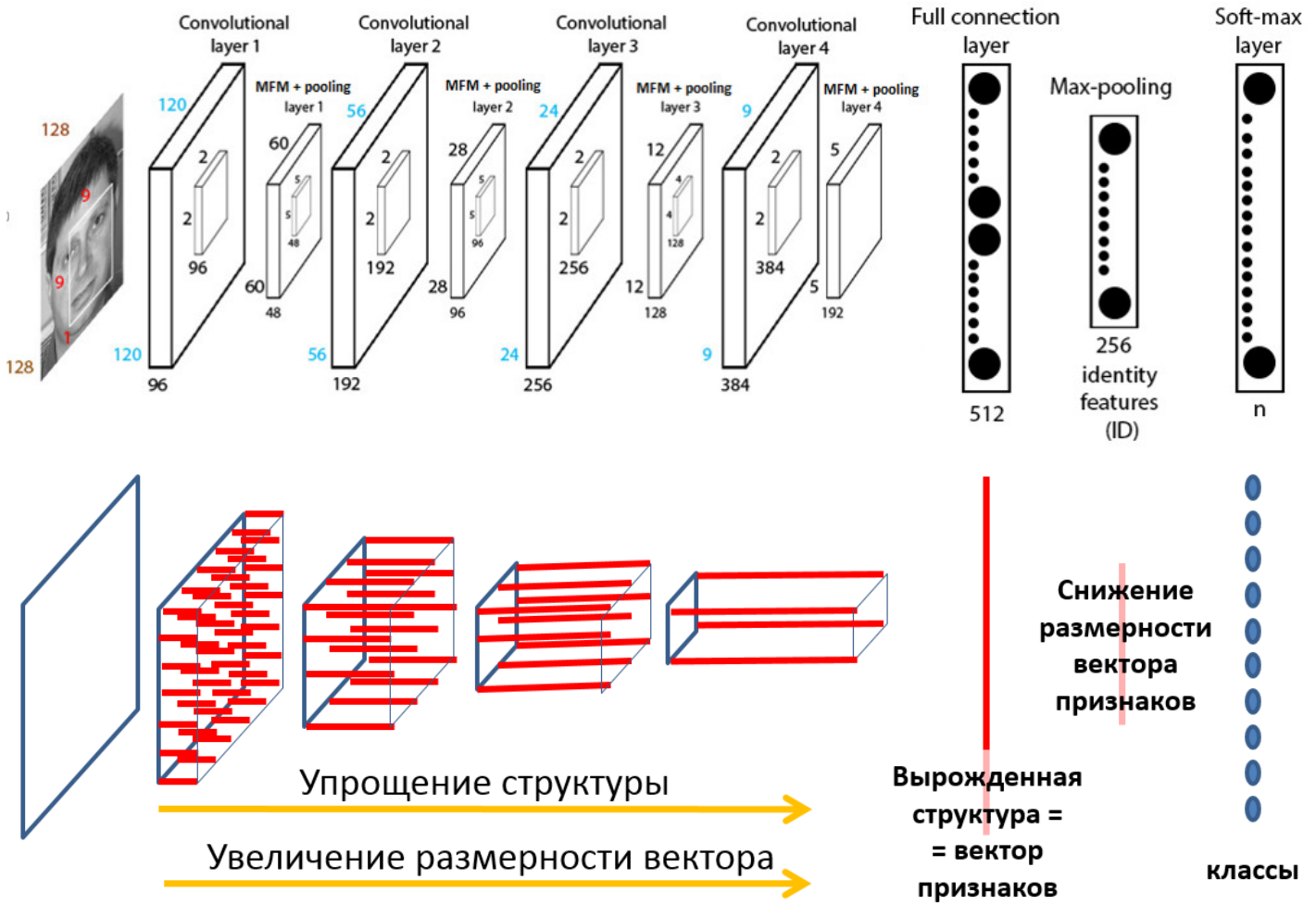
Yoon Kim. Convolutional neural networks for sentence classification. 2014

Приложение: принятие решений в логических играх



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

Глубокая свёрточная сеть как способ векторизации изображений



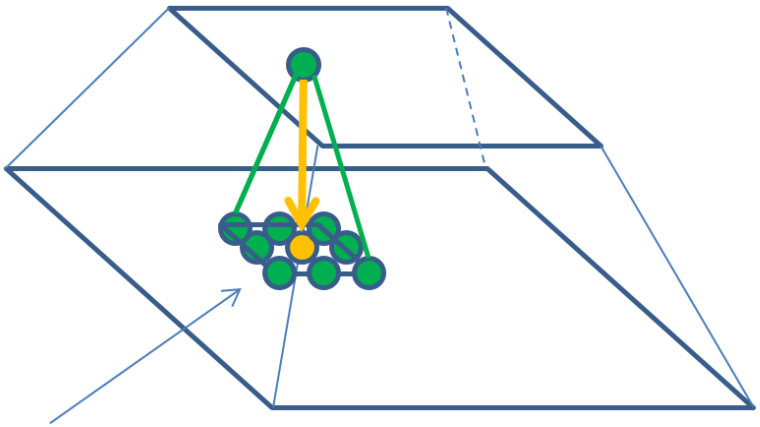
Визильтер Ю.В., Горбацевич В.С. Структурно-функциональный анализ и синтез глубоких конволюционных нейронных сетей. ММРО-2017.

Идея обобщения CNN на любые структурированные данные

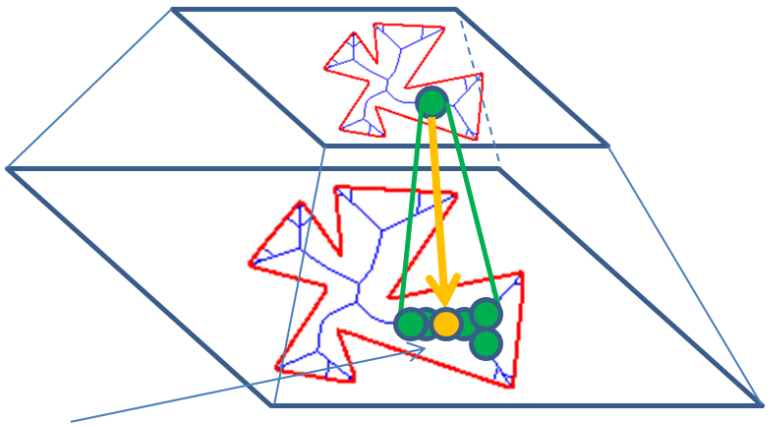
Допустим, каждый объект имеет структуру, заданную графом

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

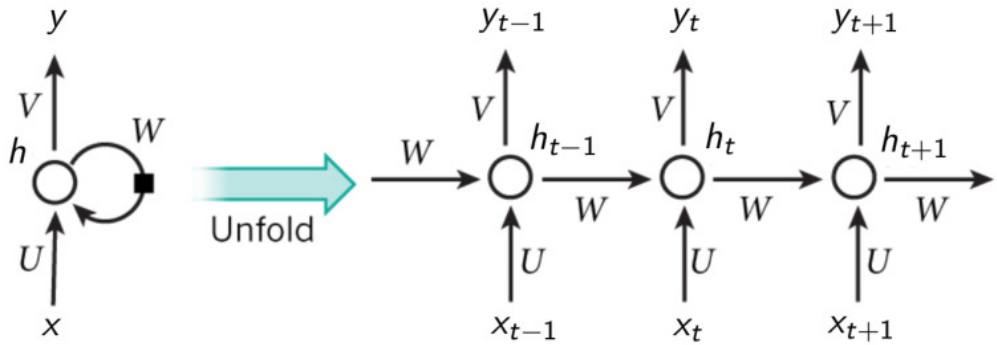
x_t — входной вектор в момент t

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:

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

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

Приложения рекуррентных нейронных сетей

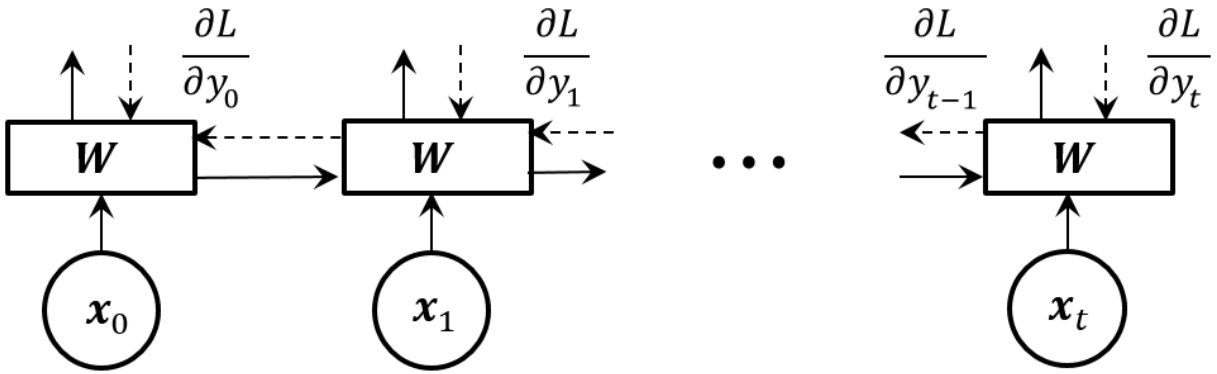
- Прогнозирование временных рядов
- Управление технологическими процессами
- Классификация текстов или их фрагментов
- Анализ тональности документа / предложений / слов
- Машинный перевод
- Распознавание речи
- Синтез речи
- Синтез ответов на вопросы, разговорный интеллект
- Генерация подписей к изображениям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

Andrej Karpathy. The unreasonable effectiveness of recurrent neural networks. 2015.

Обучение рекуррентных сетей

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

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

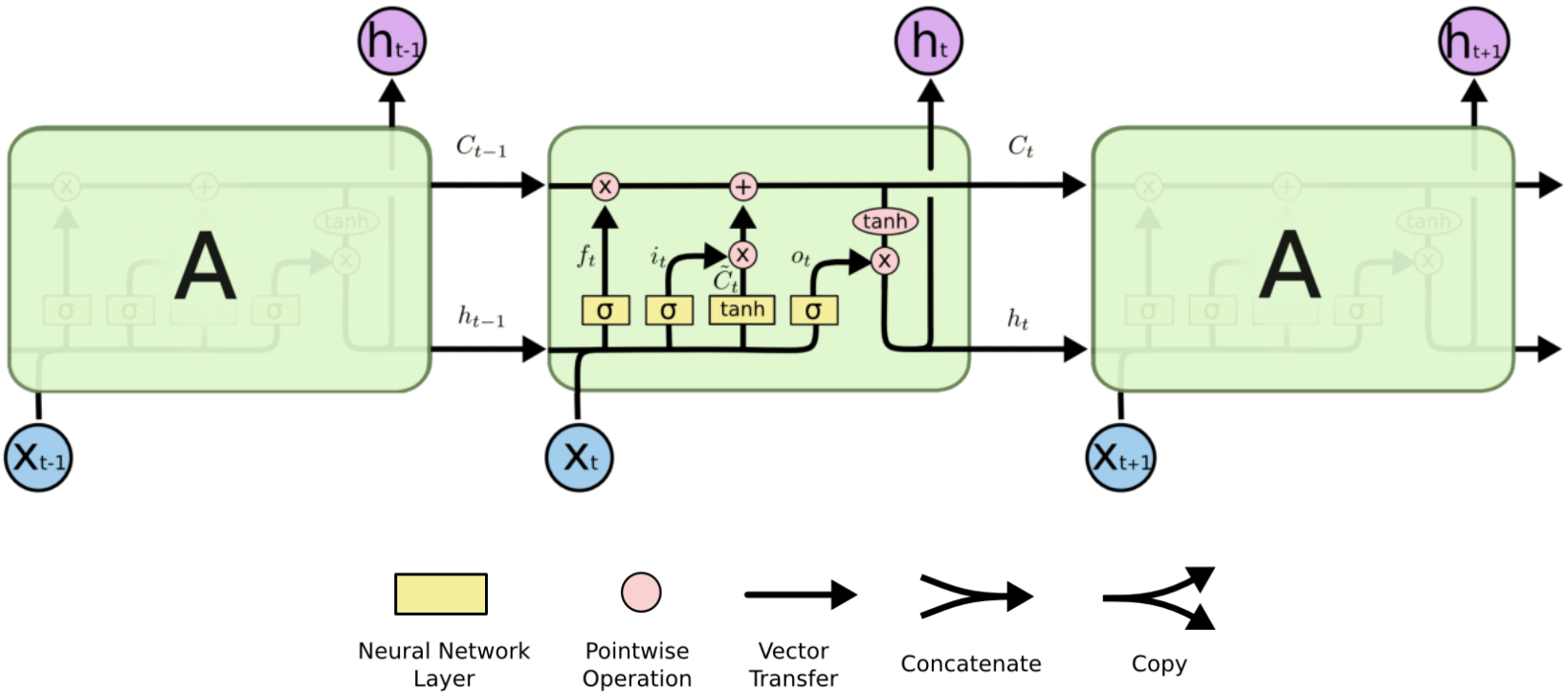


Для предотвращения затухания и взрыва градиентов: $\frac{\partial h_i}{\partial h_{i-1}} \to 1$

D.Rumelhart, G.Hinton, R.Williams. Learning internal representations by error propagation, 1985.

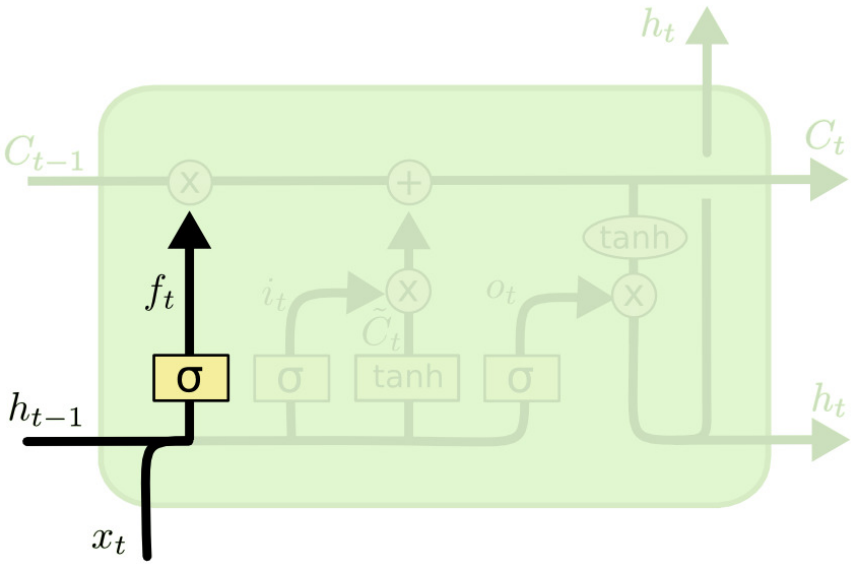
Сети долгой кратковременной памяти (long short-term memory)

Мотивация LSTM: сеть должна долго помнить предысторию, т.е. контекст, какой именно — сеть должна выучить сама.
Вводится C_t — вектор долгого контекста 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)



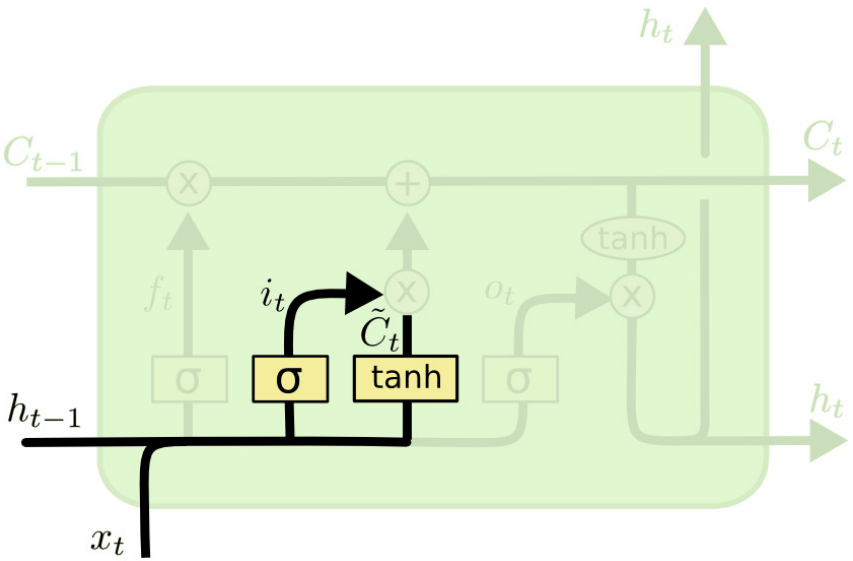
$$f_t = \sigma\big(W_f \cdot [h_{t-1}, x_t] + b_f\big)$$
$$i_t = \sigma\big(W_i \cdot [h_{t-1}, x_t] + b_i\big)$$
$$o_t = \sigma\big(W_o \cdot [h_{t-1}, x_t] + b_o\big)$$
$$\tilde{C}_t = \text{th}\big(W_c \cdot [h_{t-1}, x_t] + b_c\big)$$
$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$
$$h_t = o_t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

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

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

Christopher Olah. http://colah.github.io/posts/2015-08-Understanding-LSTMs

Сети долгой кратковременной памяти (long short-term memory)



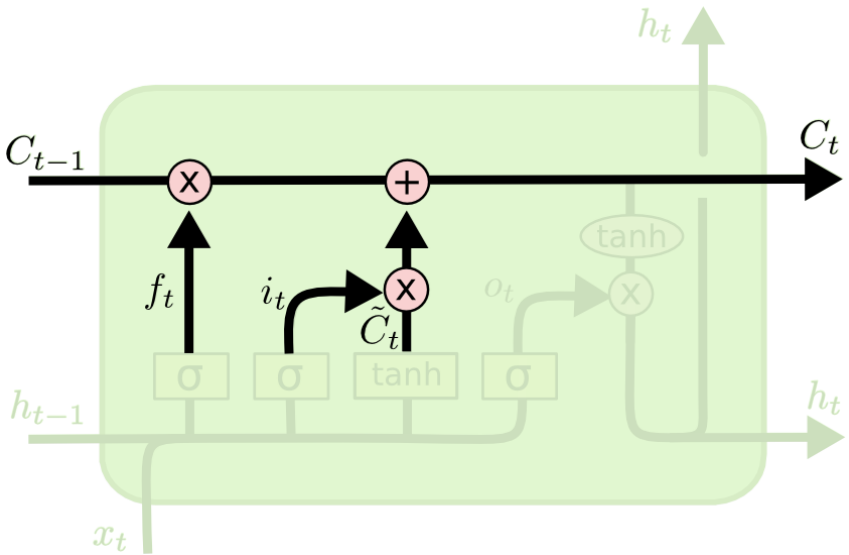
$$f_t = \sigma\big(W_f \cdot [h_{t-1}, x_t] + b_f\big)$$
$$i_t = \sigma\big(W_i \cdot [h_{t-1}, x_t] + b_i\big)$$
$$o_t = \sigma\big(W_o \cdot [h_{t-1}, x_t] + b_o\big)$$
$$\tilde{C}_t = \text{th}\big(W_c \cdot [h_{t-1}, x_t] + b_c\big)$$
$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$
$$h_t = o_t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

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

Модель нового контекста с параметрами W_c, b_c формирует вектор $\tilde{C}_t$ с информацией о новом контексте.

Christopher Olah. http://colah.github.io/posts/2015-08-Understanding-LSTMs

Сети долгой кратковременной памяти (long short-term memory)



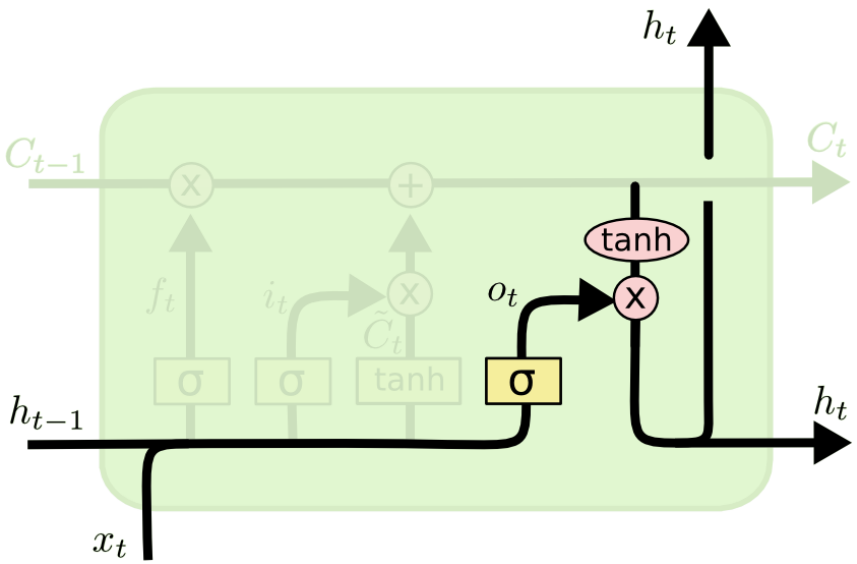
$$f_t = \sigma\big(W_f \cdot [h_{t-1}, x_t] + b_f\big)$$
$$i_t = \sigma\big(W_i \cdot [h_{t-1}, x_t] + b_i\big)$$
$$o_t = \sigma\big(W_o \cdot [h_{t-1}, x_t] + b_o\big)$$
$$\tilde{C}_t = \text{th}\big(W_c \cdot [h_{t-1}, x_t] + b_c\big)$$
$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$
$$h_t = o_t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

Новый вектор контекста C_t формируется как смесь старого вектора контекста C_{t-1} с фильтром f_t и нового вектора контекста $\tilde{C}_t$ с фильтром i_t.

Настраиваемых параметров нет.

Christopher Olah. http://colah.github.io/posts/2015-08-Understanding-LSTMs

Сети долгой кратковременной памяти (long short-term memory)



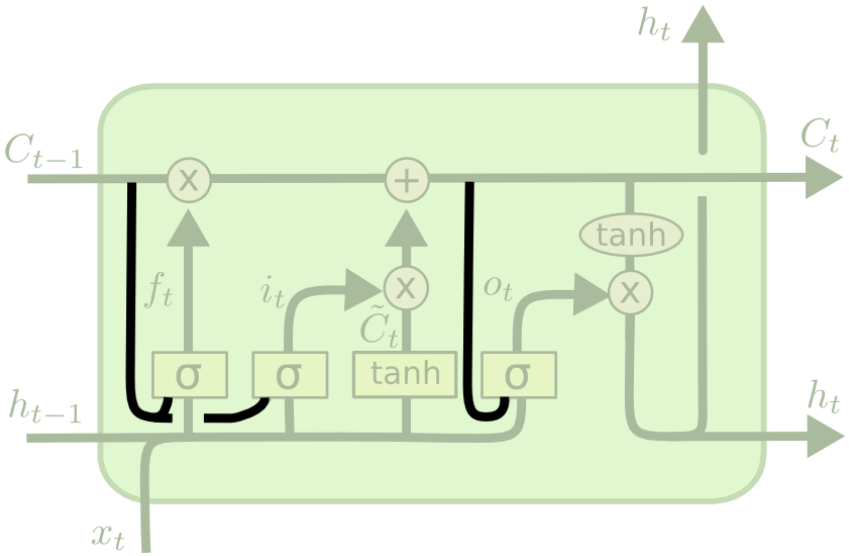
$$f_t = \sigma\big(W_f \cdot [h_{t-1}, x_t] + b_f\big)$$
$$i_t = \sigma\big(W_i \cdot [h_{t-1}, x_t] + b_i\big)$$
$$o_t = \sigma\big(W_o \cdot [h_{t-1}, x_t] + b_o\big)$$
$$\tilde{C}_t = \text{th}\big(W_c \cdot [h_{t-1}, x_t] + b_c\big)$$
$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$
$$h_t = o_t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

Фильтр выходных данных (output gate) с параметрами W_o, b_o решает, какие координаты вектора контекста C_t пойдут на выход.

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

Christopher Olah. http://colah.github.io/posts/2015-08-Understanding-LSTMs

Вариант LSTM с «замочными скважинами» (peepholes)



$$f_t = \sigma\big(W_f \cdot [C_{t-1}, h_{t-1}, x_t] + b_f\big)$$
$$i_t = \sigma\big(W_i \cdot [C_{t-1}, h_{t-1}, x_t] + b_i\big)$$
$$\tilde{C}_t = \mathrm{th}\big(W_c \cdot [h_{t-1}, x_t] + b_c\big)$$
$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$
$$o_t = \sigma\big(W_o \cdot [C_t, h_{t-1}, x_t] + b_o\big)$$
$$h_t = o_t \odot \mathrm{th}(C_t)$$

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

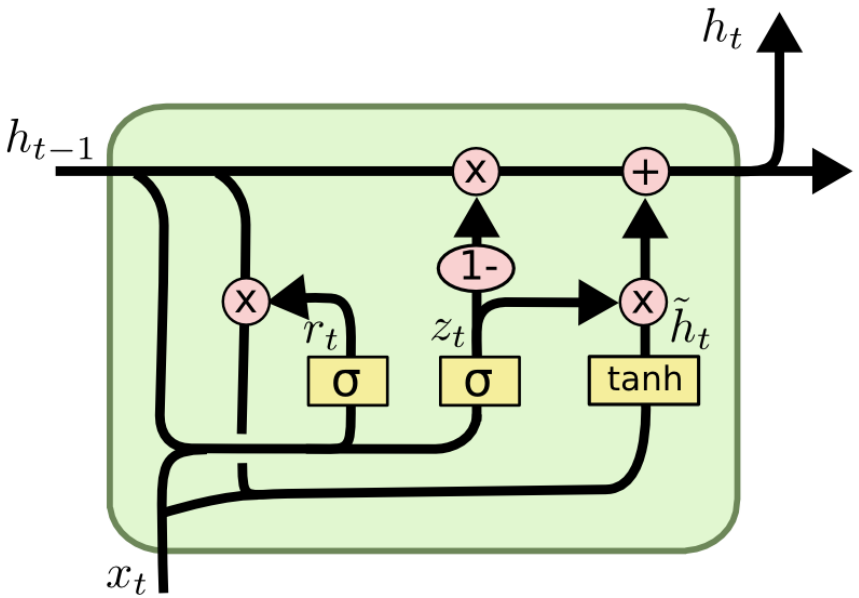
Увеличивается число параметров модели.

Увеличивается число слоёв — с трёх до четырёх.

Замочную скважину можно использовать не для всех фильтров.

Gers F. A., Schmidhuber J. Recurrent Nets that Time and Count. 2000.

Упрощение LSTM: Gated Recurrent Unit (GRU)



$$z_t = \sigma\big(W_z \cdot [h_{t-1}, x_t] + b_z\big)$$
$$r_t = \sigma\big(W_r \cdot [h_{t-1}, x_t] + b_r\big)$$
$$\tilde{h}_t = \text{th}\big(W_h \cdot [r_t \odot h_{t-1}, x_t] + b_h\big)$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$$

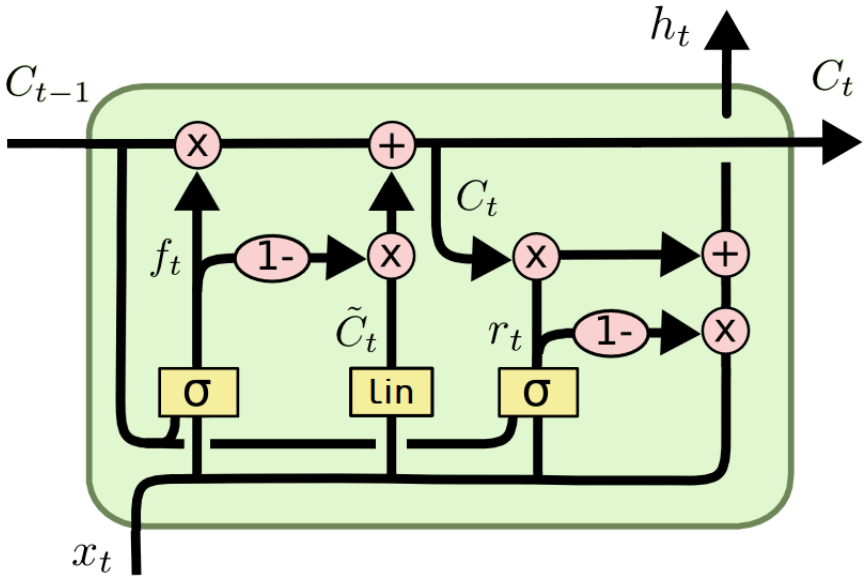
Используется только состояние h_t, вектор C_t не вводится.

Фильтр обновления (update gate) вместо входного и забывающего.

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

Cho K. On the properties of neural machine translation: encoder-decoder approaches. 2014.

Упрощение LSTM: Simple Recurrent Unit (SRU)



$$f_t = \sigma\big(W_f x_t + v_f \odot C_{t-1} + b_f\big)$$
$$r_t = \sigma\big(W_r x_t + v_r \odot C_{t-1} + b_r\big)$$
$$\tilde{C}_t = W_C x_t$$
$$C_t = f_t \odot C_{t-1} + (1 - f_t) \odot \tilde{C}_t$$
$$h_t = r_t \odot C_t + (1 - r_t) \odot x_t$$

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.

Резюме

- *Свёрточные сети:* постепенная векторизация сложно структурированных данных, обучаемая совместно с основной предсказательной 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 к искажениям
  - 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