# Глубокие нейронные сети

К. В. Воронцов
vokov@forecsys.ru

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

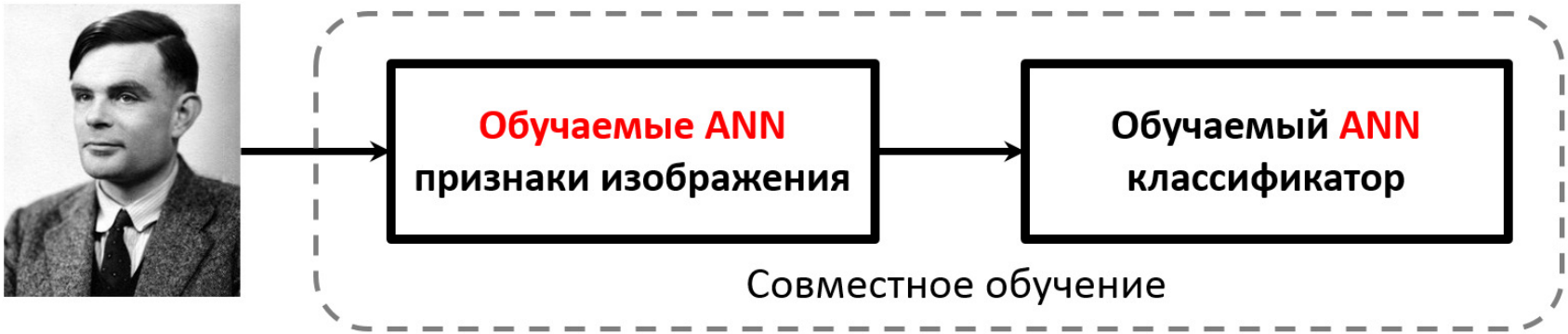
17 сентября 2019 • ШАД Яндекс

# Содержание

# Классификация изображений

Классический подход к распознаванию изображений:


Современный подход — end-to-end deep learning:

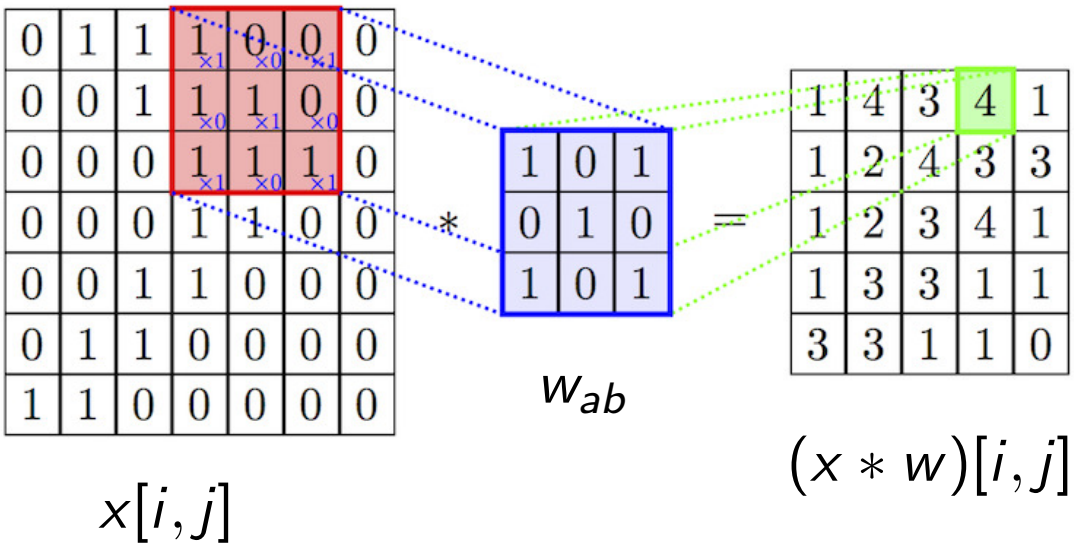


# Свёрточный слой нейронов (convolution layer)

$x[i,j]$ — 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:

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

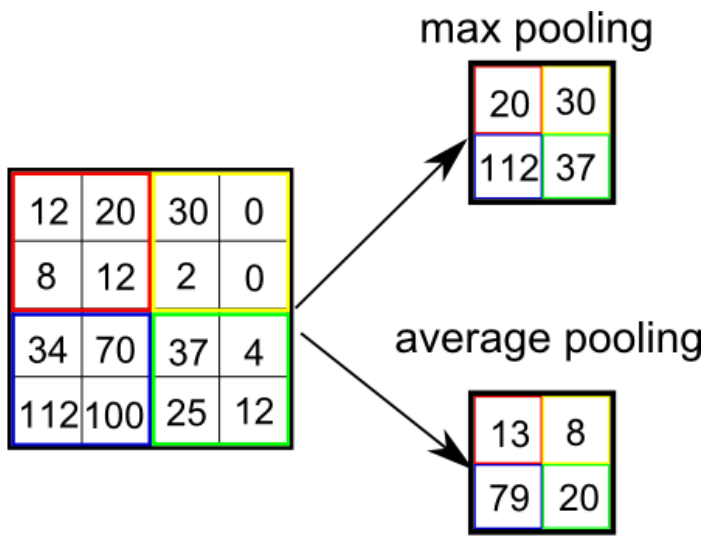


# Объединяющий слой нейронов (pooling layer)

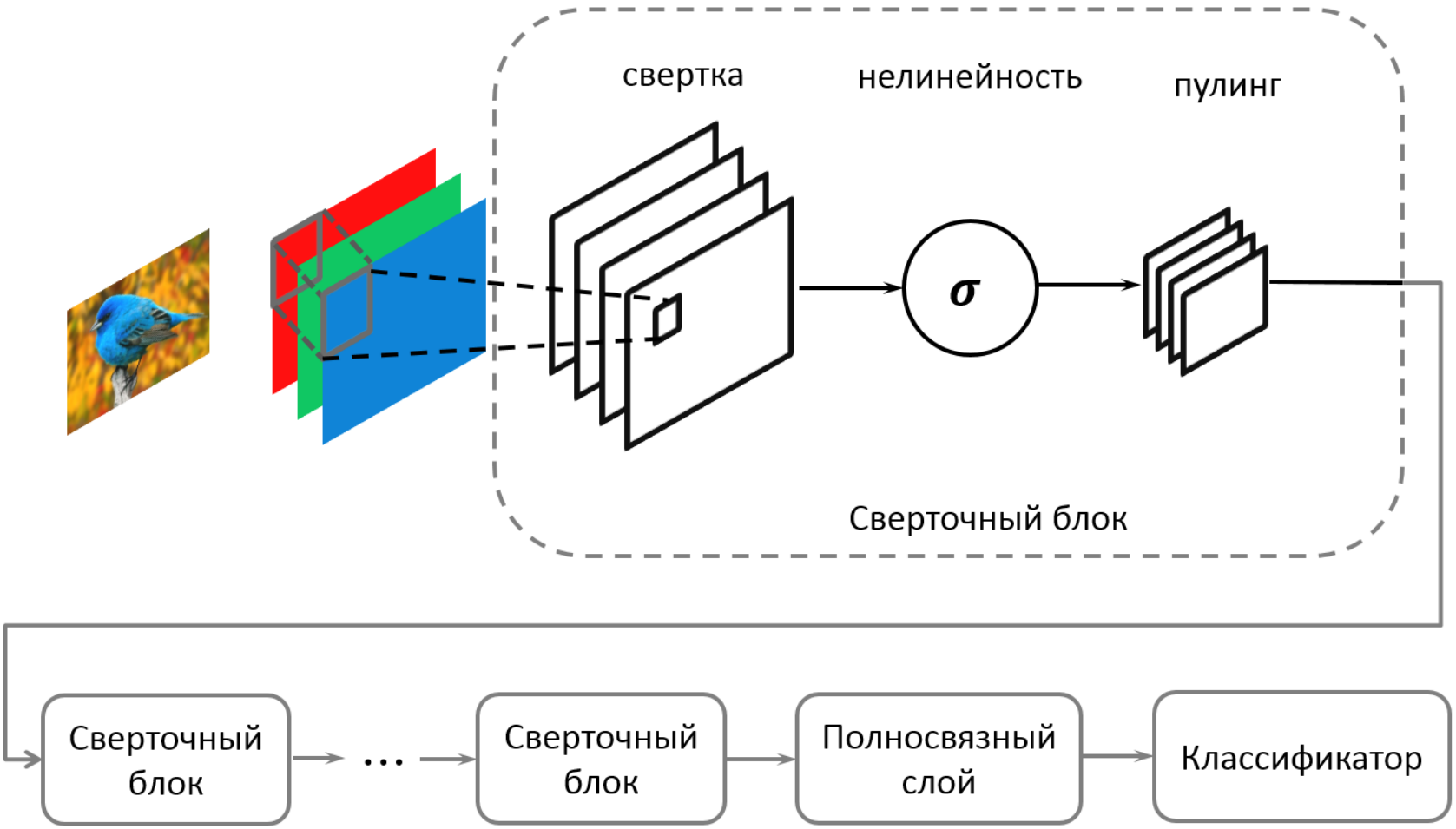
Объединяющий нейрон — это необучаемая свёртка с шагом $h > 1$, агрегирующая данные прямоугольной области $h \times h$:

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

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

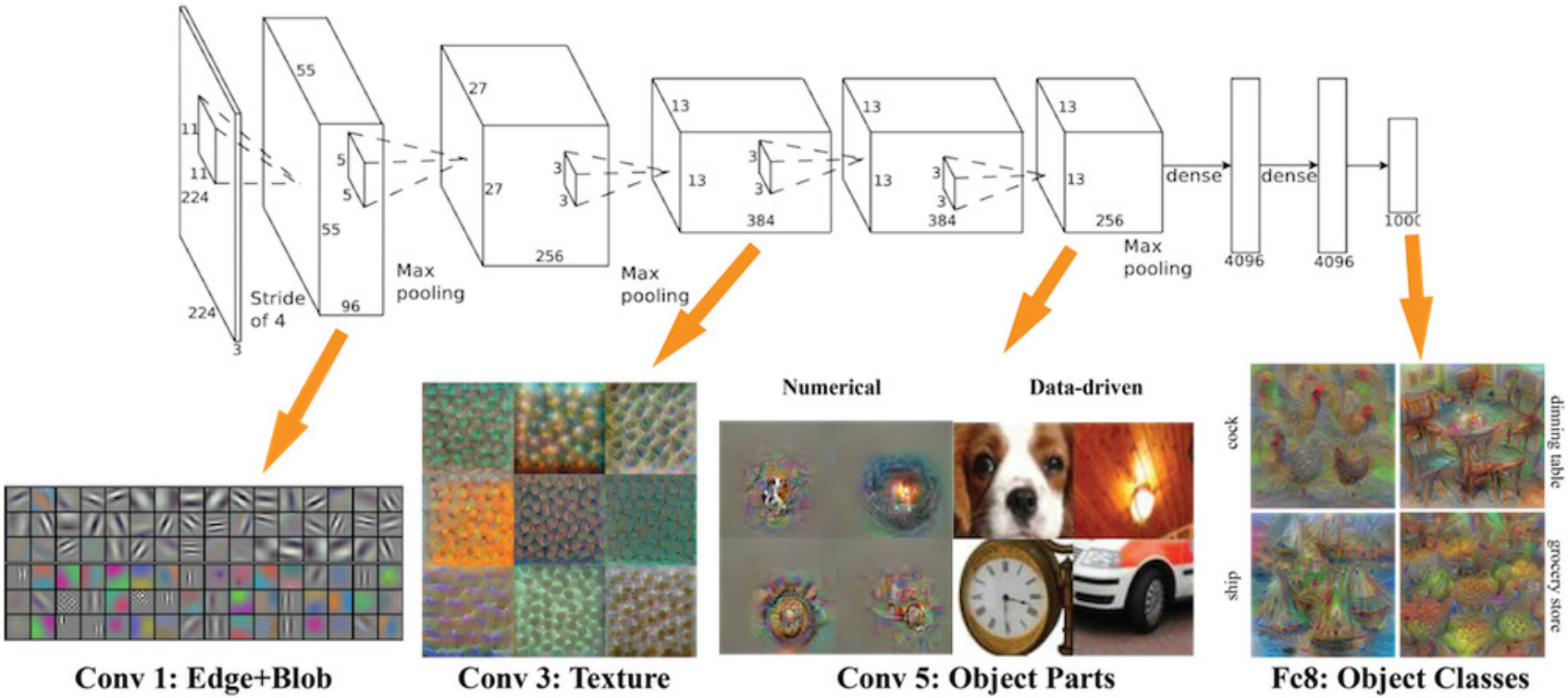


# Стандартная схема сверточной сети (Convolutional NN)



# Свёрточная сеть обучается извлечению признаков

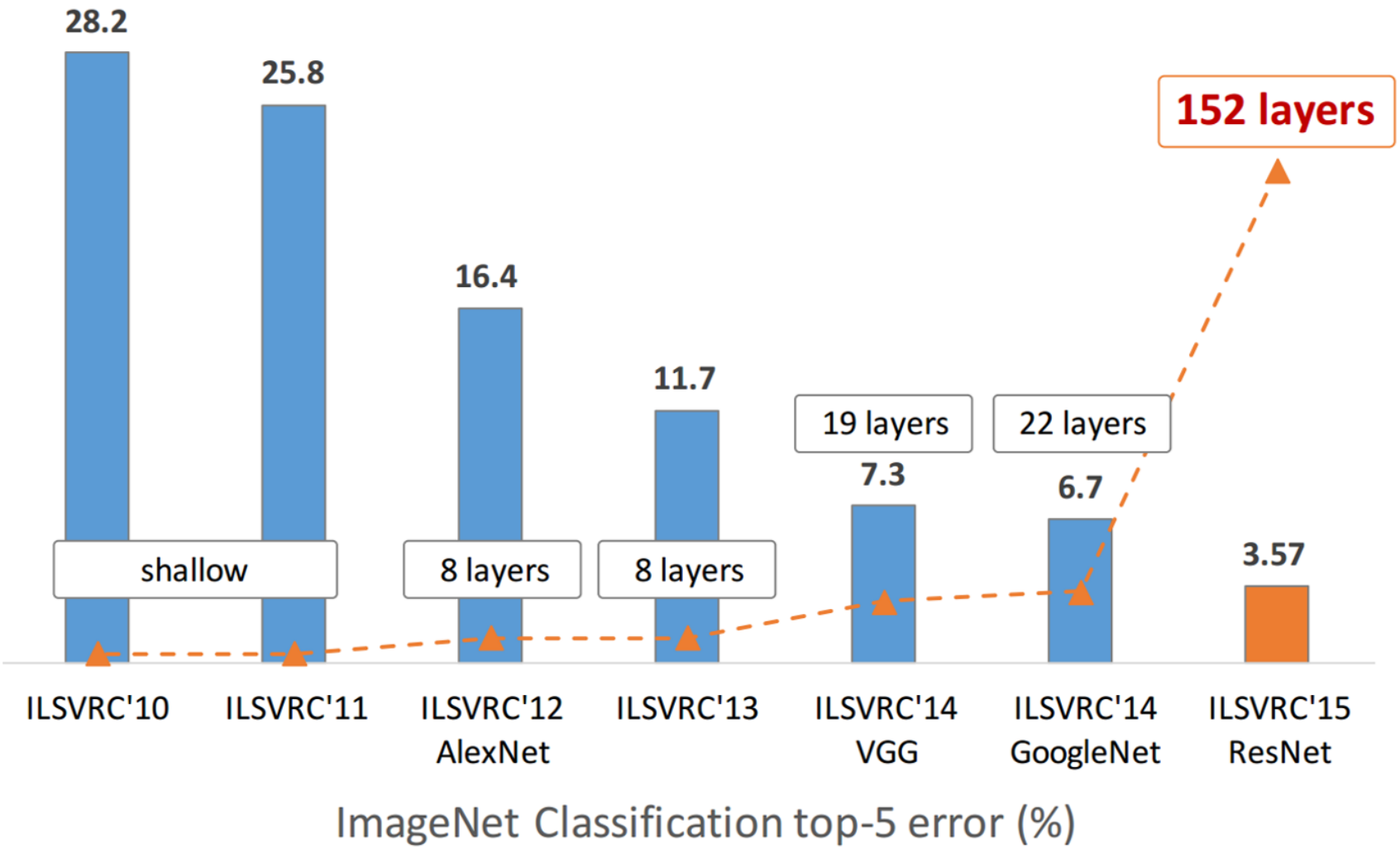
Чем выше слой, тем более крупные и сложные элементы изображений он способен распознавать



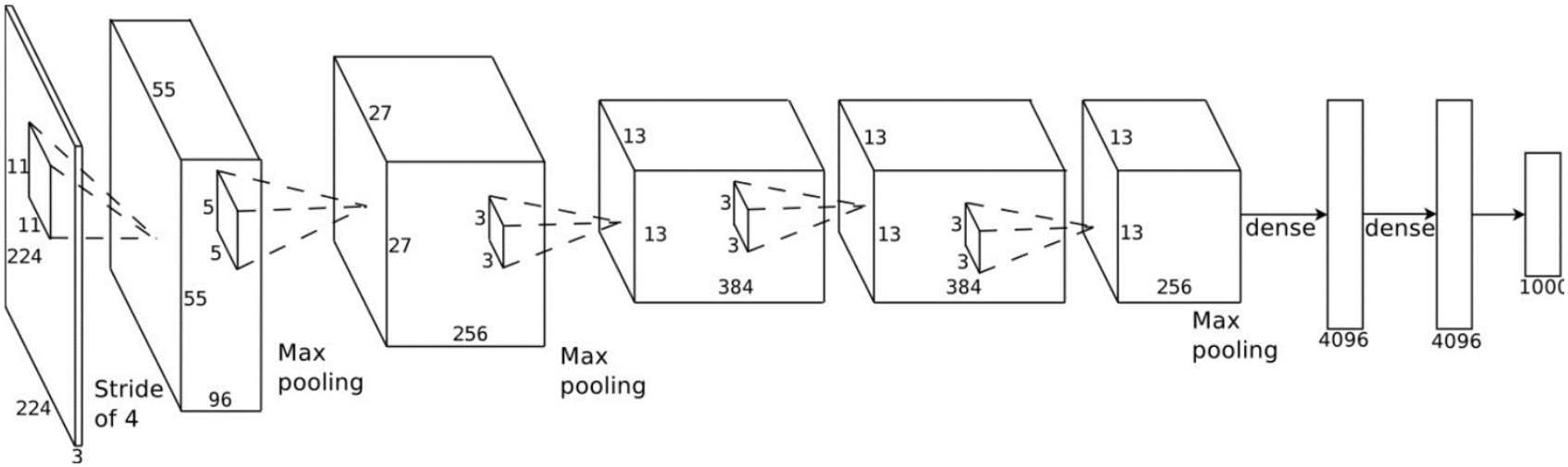

*Krizhevsky A., Sutskever I., Hinton G.* ImageNet Classification with Deep Convolutional Neural Networks. 2012.

# ImageNet — большая выборка размеченных изображений

# 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

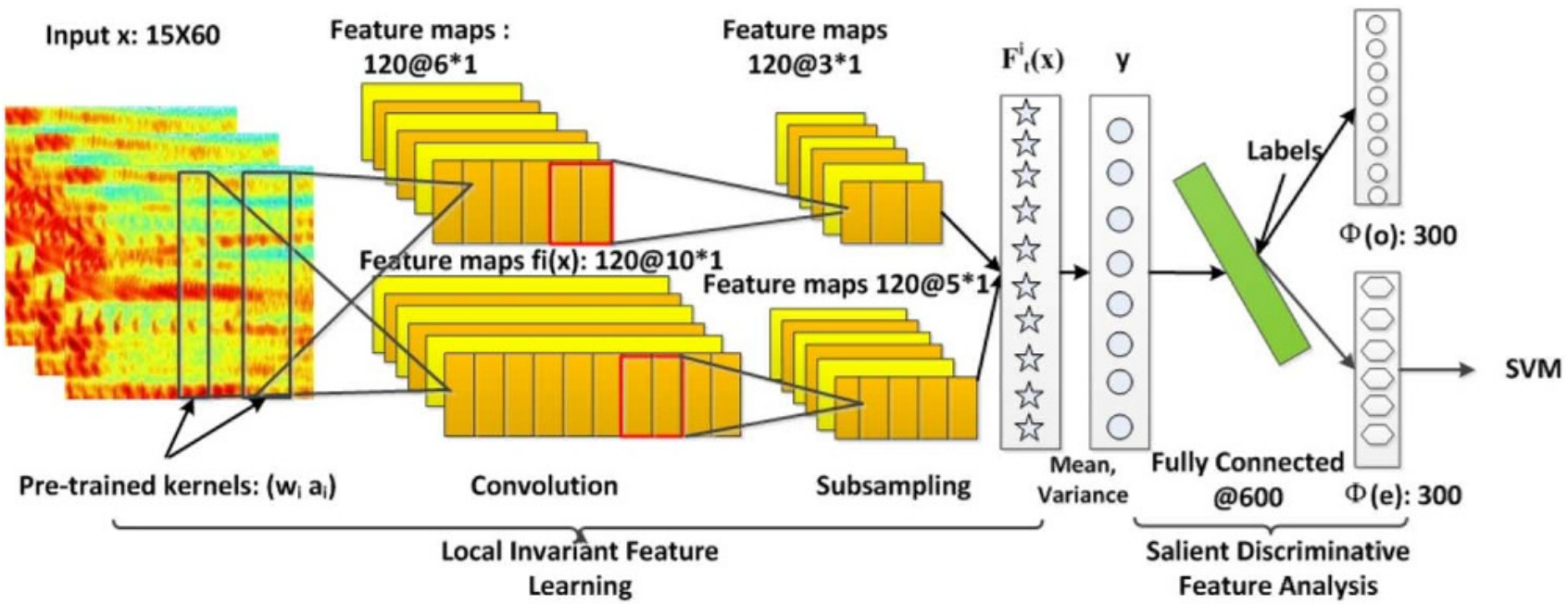
*Krizhevsky A., Sutskever I., Hinton G.* ImageNet Classification with Deep Convolutional Neural Networks. 2012.

# Часто используемые приёмы в CNN

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

# Приложение: распознавание речевых сигналов

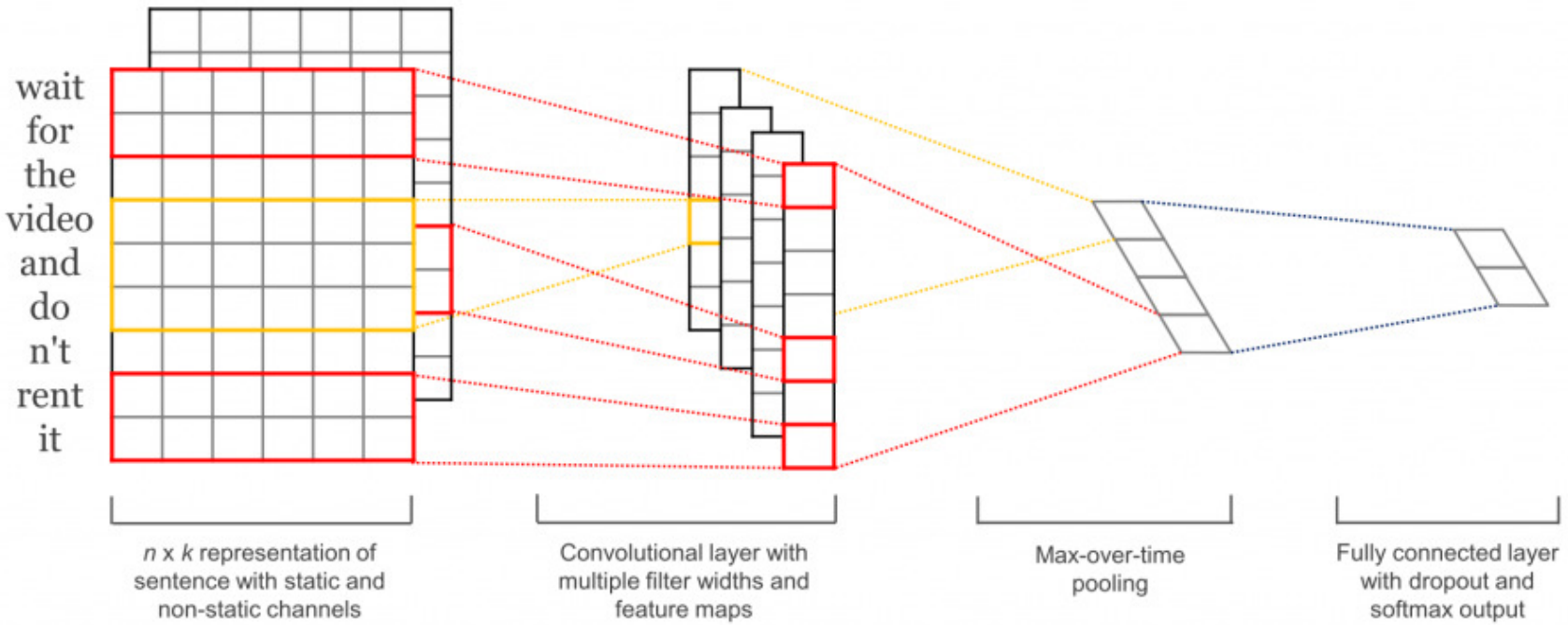
Последовательные фрагменты сигнала представляются векторами спектрального разложения




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

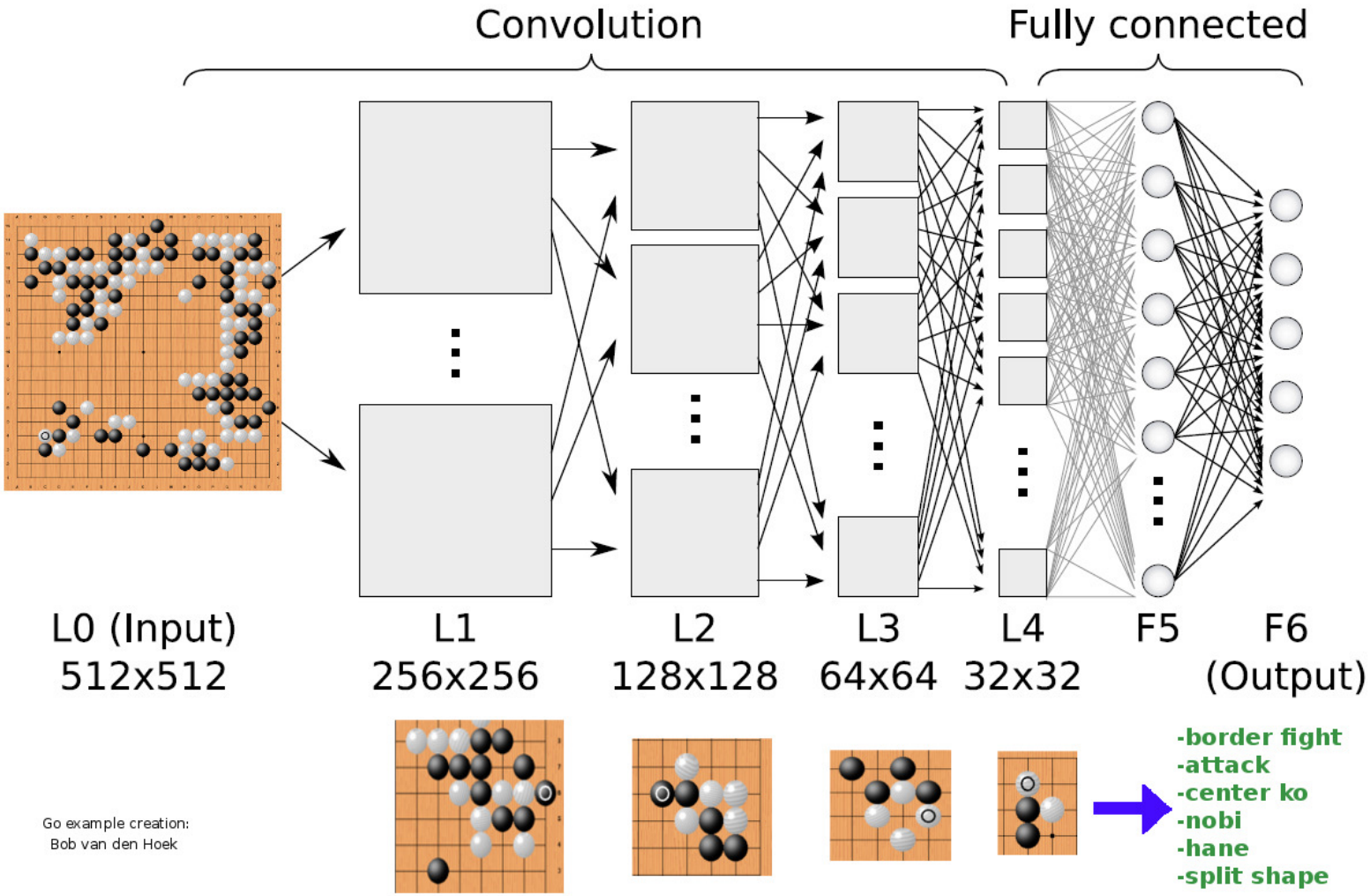
# Приложение: классификация предложений в тексте

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



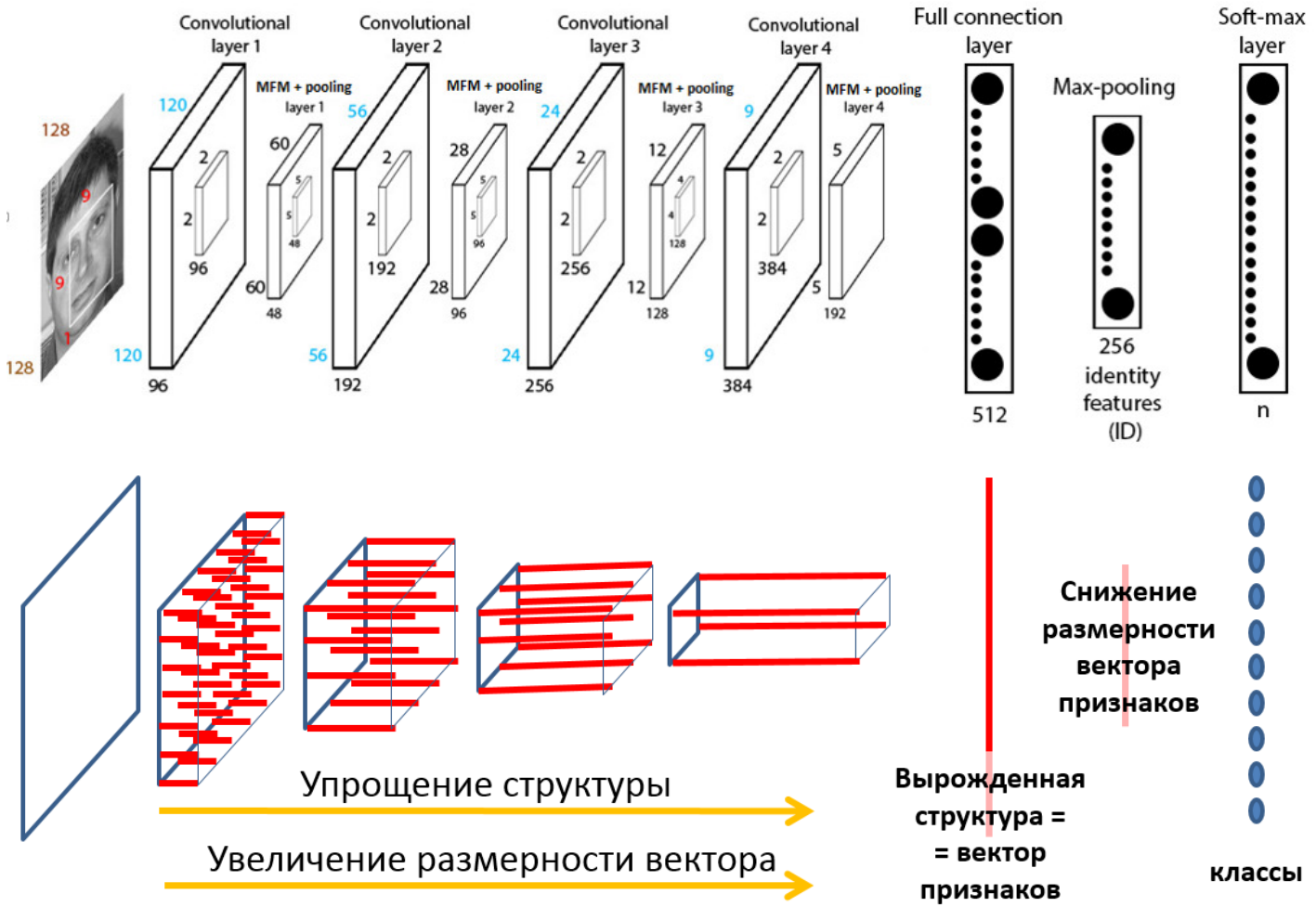

*Yoon Kim.* Convolutional neural networks for sentence classification. 2014

# Приложение: принятие решений в логических играх




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

# Идея обобщения CNN на любые структурированные данные



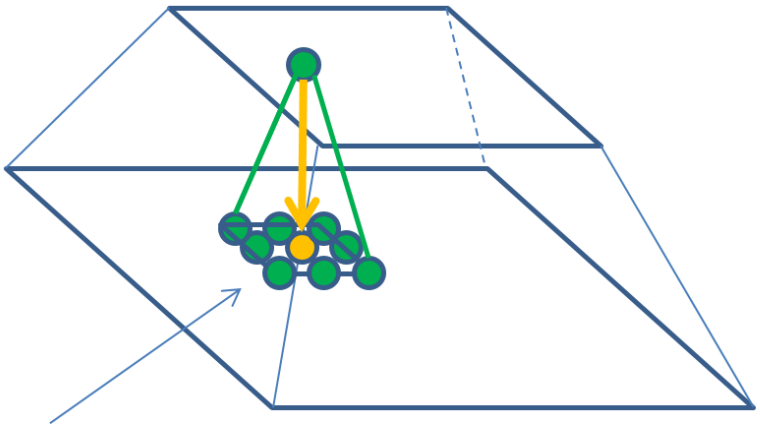

*Визильтер Ю.В., Горбацевич В.С.* Структурно-функциональный анализ и синтез глубоких конволюционных нейронных сетей. ММРО-2017.

# Идея обобщения CNN на любые структурированные данные

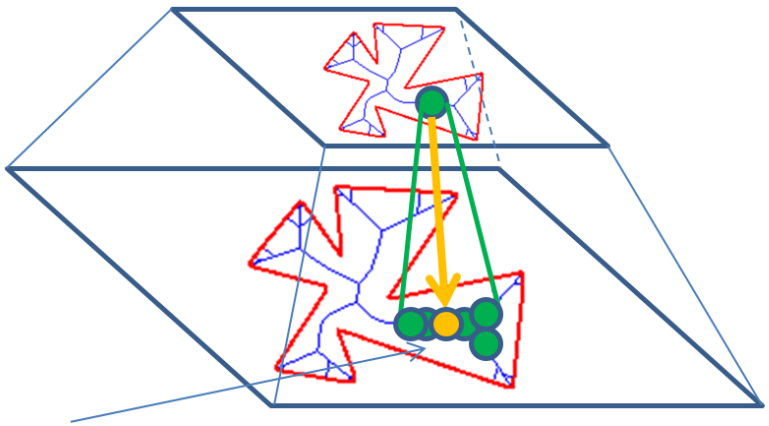
Допустим, каждый объект имеет структуру, заданную графом

**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



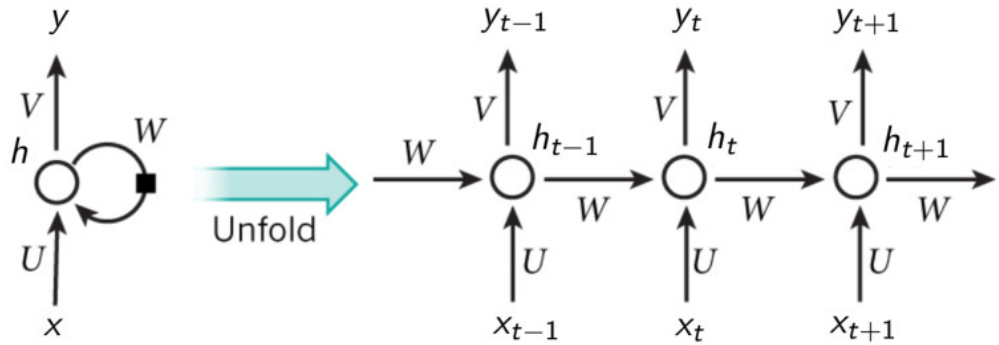
Локальная окрестность, определяемая для любой вершины графа +
+ операция свёртки по окрестности

# 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

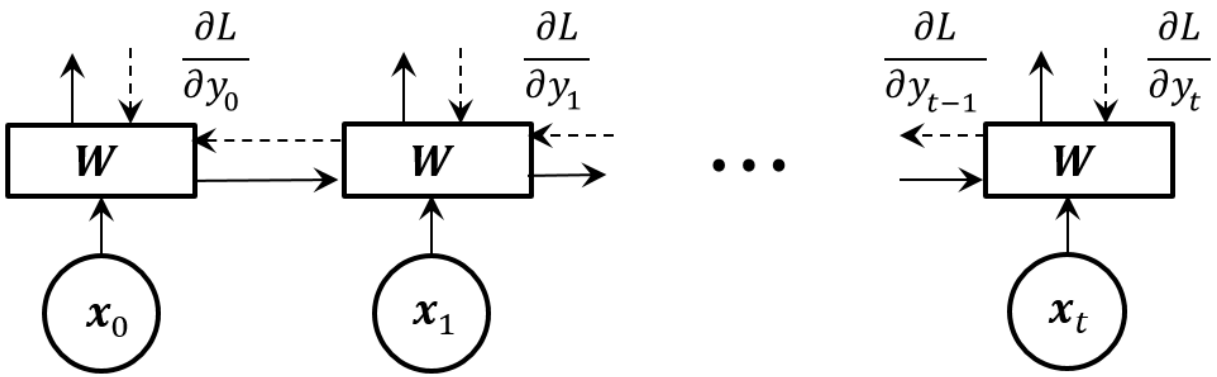
---

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

# Обучение рекуррентных сетей

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

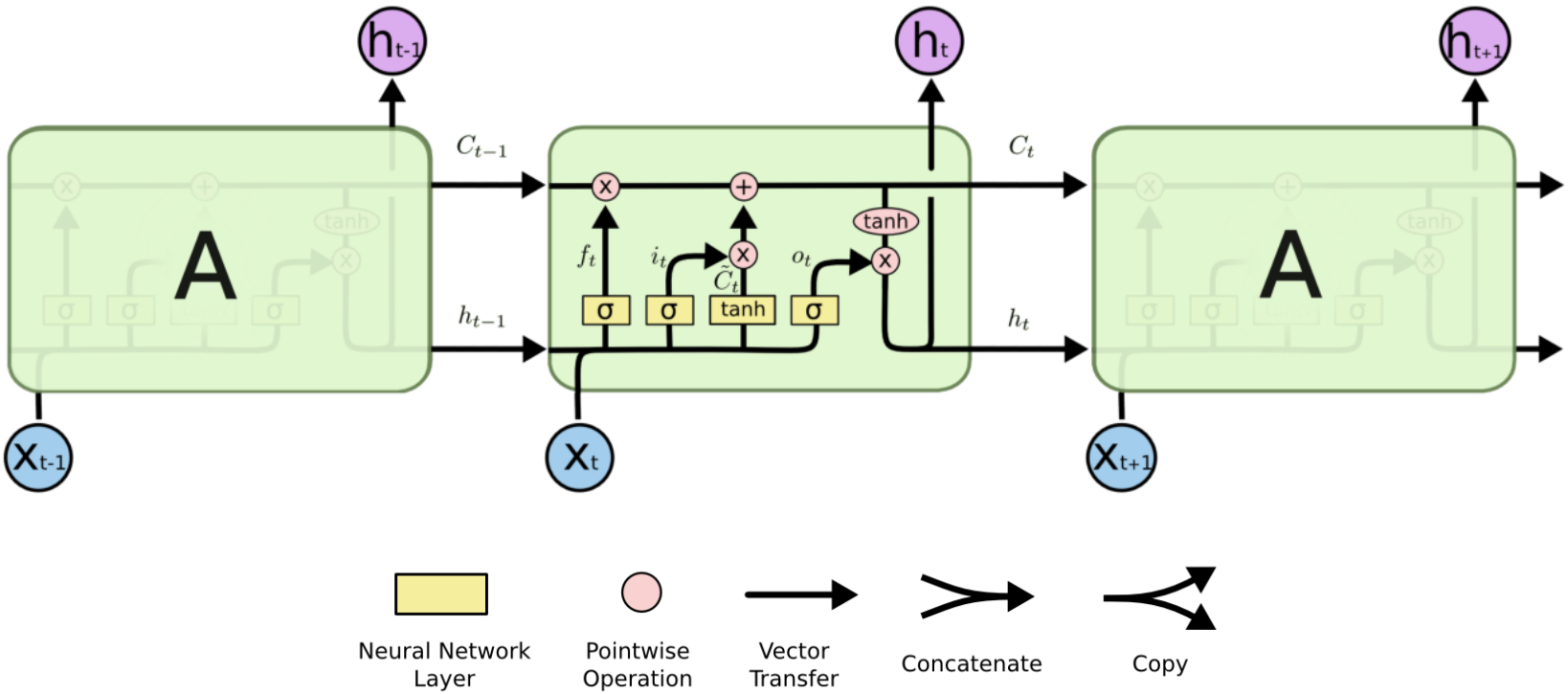
$$\frac{\partial \mathscr{L}_t}{\partial W} = \frac{\partial \mathscr{L}_t}{\partial y_t}\frac{\partial y_t}{\partial h_t}\sum_{k=0}^{t}\left(\prod_{i=k+1}^{t}\frac{\partial h_i}{\partial h_{i-1}}\right)\frac{\partial h_k}{\partial W}$$




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

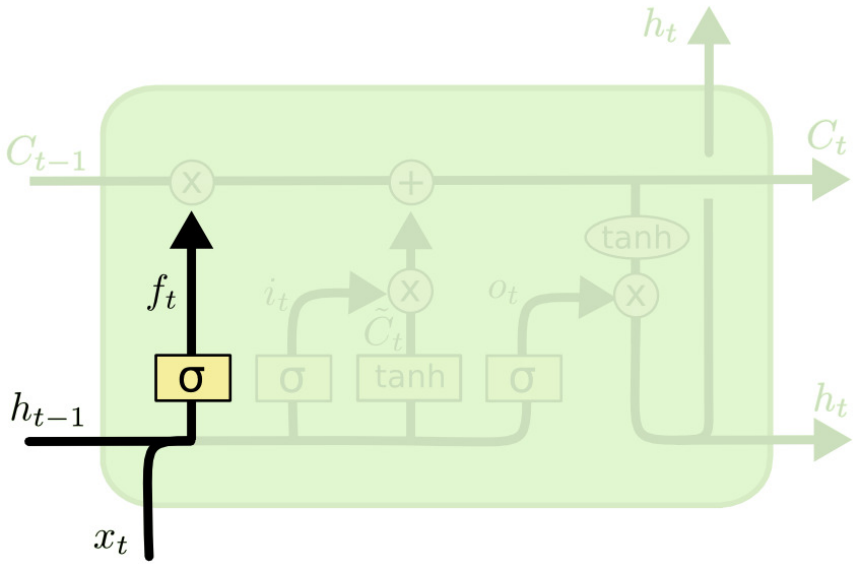
# Сети долгой кратковременной памяти (long short-term memory)

**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)




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

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

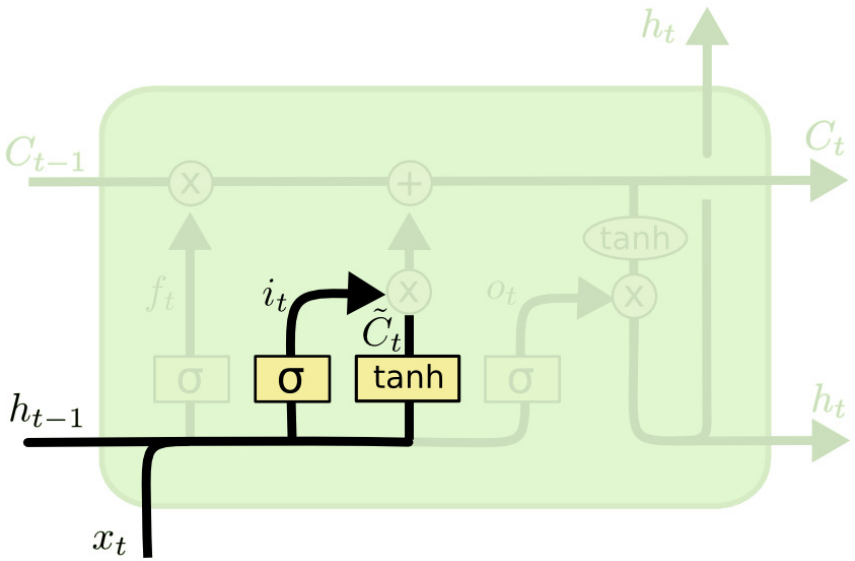
$\odot$ — операция покомпонентного перемножения векторов.

$\sigma$ — сигмоидная функция.

---

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

# Сети долгой кратковременной памяти (long short-term memory)




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

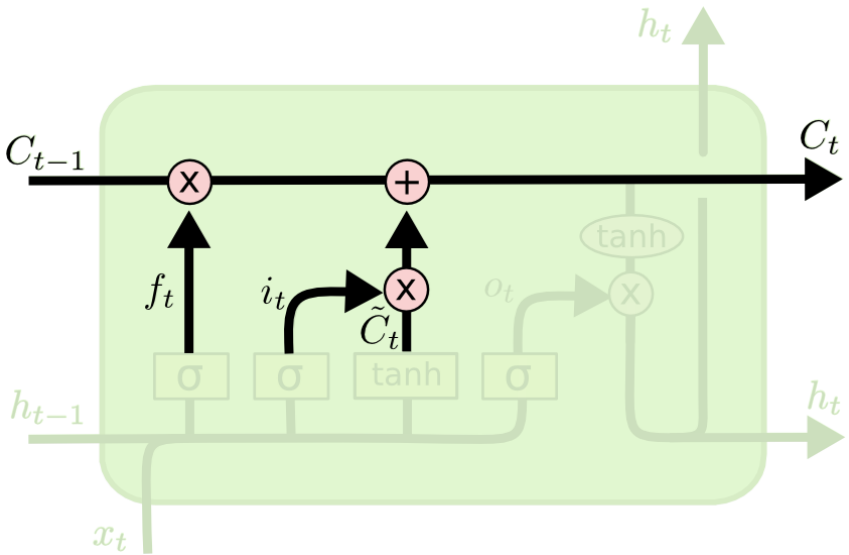
*Фильтр входных данных* (input gate) с параметрами $W_i$, $b_i$ решает, какие координаты вектора состояния надо обновить.

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

---

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

# Сети долгой кратковременной памяти (long short-term memory)




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

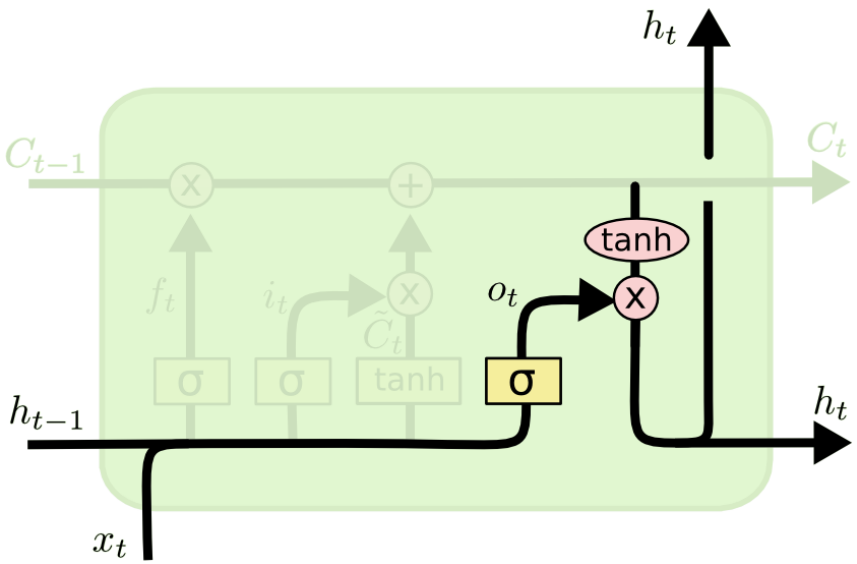
Новое состояние $C_t$ формируется как смесь старого состояния $C_{t-1}$ с фильтром $f_t$ и вектора значений-кандидатов $\tilde{C}_t$ с фильтром $i_t$.

Настраиваемых параметров нет.

---

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

# Сети долгой кратковременной памяти (long short-term memory)




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

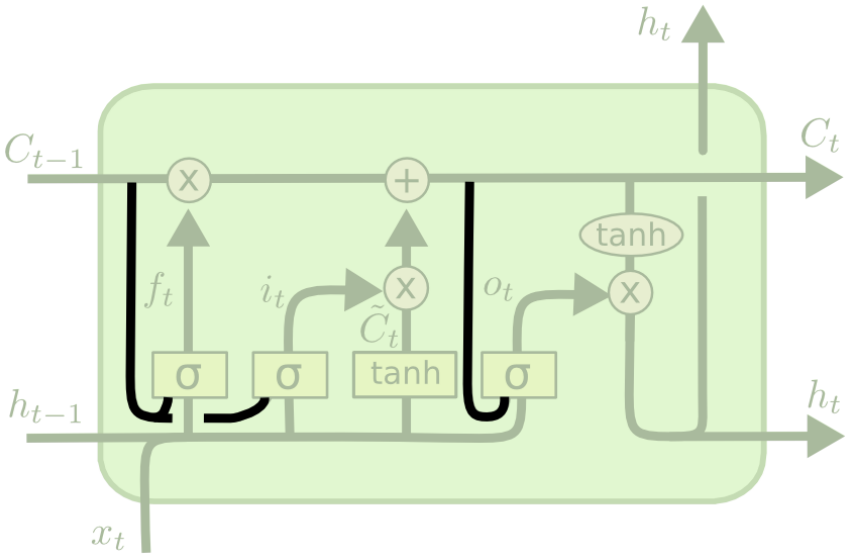
*Фильтр выходных данных* (output gate) с параметрами $W_o$, $b_o$ решает, какие координаты вектора состояния $C_t$ надо выдать.

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

---

*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 состояния $C_{t-1}$ или $C_t$.

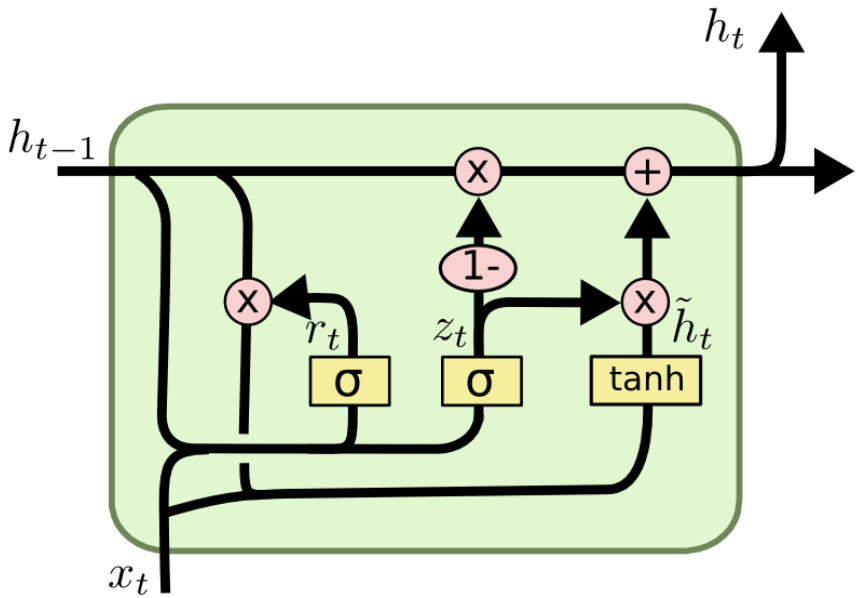
Увеличивается число параметров модели.

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)




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

Используется только состояние $h_t$, вектор $C_t$ не вводится.

*Фильтр обновления* (update gate) вместо входного и забывающего.

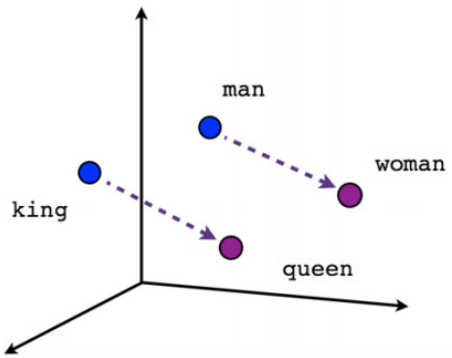
*Фильтр перезагрузки* (reset gate) решает, какую часть памяти нужно перенести дальше с прошлого шага.

---

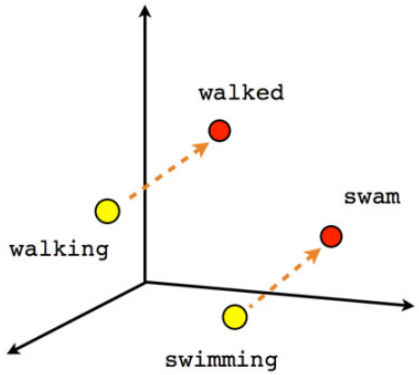
*Cho K.* On the properties of neural machine translation: encoder-decoder approaches. 2014.

# Задача семантического векторного представления слов

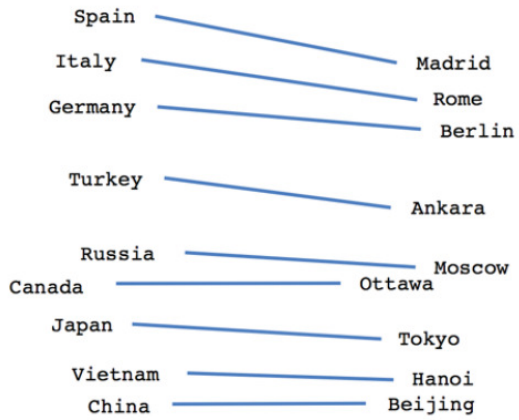
Найти для каждого слова $w$ вектор $x_w \in \mathbb{R}^T$, чтобы близкие по смыслу слова имели близкие векторы.



Male-Female



Verb tense



Country-Capital

## Дистрибутивная гипотеза

- Words that occur in the same contexts tend to have similar meanings [Harris, 1954].
- You shall know a word by the company it keeps [Firth, 1957].

# Формализация дистрибутивной гипотезы в программе word2vec

**Дано:** $n_{uw}$ — совстречаемость слов $u, w$ в окне $\pm h$ слов
**Найти:** семантические векторные представления слов $x_w$
**Модель:** чем ближе векторы $x_u$ и $x_w$, тем больше вероятность слова $w$ в контексте слова $u$:

$$p(w|u) = \underset{w \in W}{\mathrm{SoftMax}} \langle x_w, x_u \rangle = \frac{\exp \langle x_w, x_u \rangle}{\sum_v \exp \langle x_v, x_u \rangle}$$

**Критерий** максимума log-правдоподобия и его аппроксимация:

$$\sum_{w,u \in W} n_{wu} \ln p(w|u) \;\to\; \max_{\{x_w\}}$$

$$\sum_{w,u \in W} n_{wu} \left( \ln \sigma \langle x_w, x_u \rangle + \sum_{i=1}^{k} \ln \sigma \left( -\langle x_{v_i}, x_u \rangle \right) \right) \;\to\; \max_{\{x_w\}}$$

где $v_1, \ldots, v_k$ — случайные $k$ слов не из контекста $u$.

*T.Mikolov et al.* Efficient estimation of word representations in vector space, 2013.

# Модели векторных представлений текстов и графов

**word2vec**: векторных представлений (эмбеддинги) слов

*T.Mikolov et al.* Efficient estimation of word representations in vector space. 2013.

**paragraph2vec**: эмбеддинги текстовых фрагментов

*Q.Le, T.Mikolov.* Distributed representations of sentences and documents. 2014.

**sent2vec**: эмбеддинги предложений

*M.Pagliardini et al.* Unsupervised learning of sentence embeddings using compositional n-gram features. 2017.

**FastText**: эмбеддинги символьных *n*-грамм

`https://github.com/facebookresearch/fastText`

**node2vec**: эмбеддинги узлов графов

*A.Grover, J.Leskovec.* Node2vec: scalable feature learning for networks. 2016.

**graph2vec**: более общие эмбеддинги графов

*A.Narayanan et al.* Graph2vec: learning distributed representations of graphs. 2017.

**StarSpace**: эмбеддинги чего угодно от Facebook AI Research

*L.Wu, A.Fisch, S.Chopra, K.Adams, A.B.J.Weston.* StarSpace: embed all the things! 2018.

Обзор нейросетевых моделей векторных представлений графов

*Zonghan Wu et.al.* A Comprehensive Survey on Graph Neural Networks. 2019.

# Перенос обучения (transfer learning)

$f(x_i, \alpha)$ — часть модели, универсальная для всех задач
$g(x_i, \beta)$ — часть модели, специфичная для каждой задачи

*Базовая задача* на выборке $\{x_i\}_{i=1}^{\ell}$ с функцией потерь $\mathscr{L}_i$:

$$\sum_{i=1}^{\ell} \mathscr{L}_i\big(f(x_i, \alpha), g(x_i, \beta)\big) \;\to\; \max_{\alpha,\beta}$$

*Целевая задача* на другой выборке $\{x'_i\}_{i=1}^{m}$, с другими $\mathscr{L}'_i$, $g'$:

$$\sum_{i=1}^{m} \mathscr{L}'_i\big(f(x'_i, \alpha), g'(x'_i, \beta')\big) \;\to\; \max_{\beta'}$$

при $m \ll \ell$ это может быть намного лучше, чем

$$\sum_{i=1}^{m} \mathscr{L}'_i\big(f(x'_i, \alpha), g'(x'_i, \beta')\big) \;\to\; \max_{\alpha,\beta'}$$

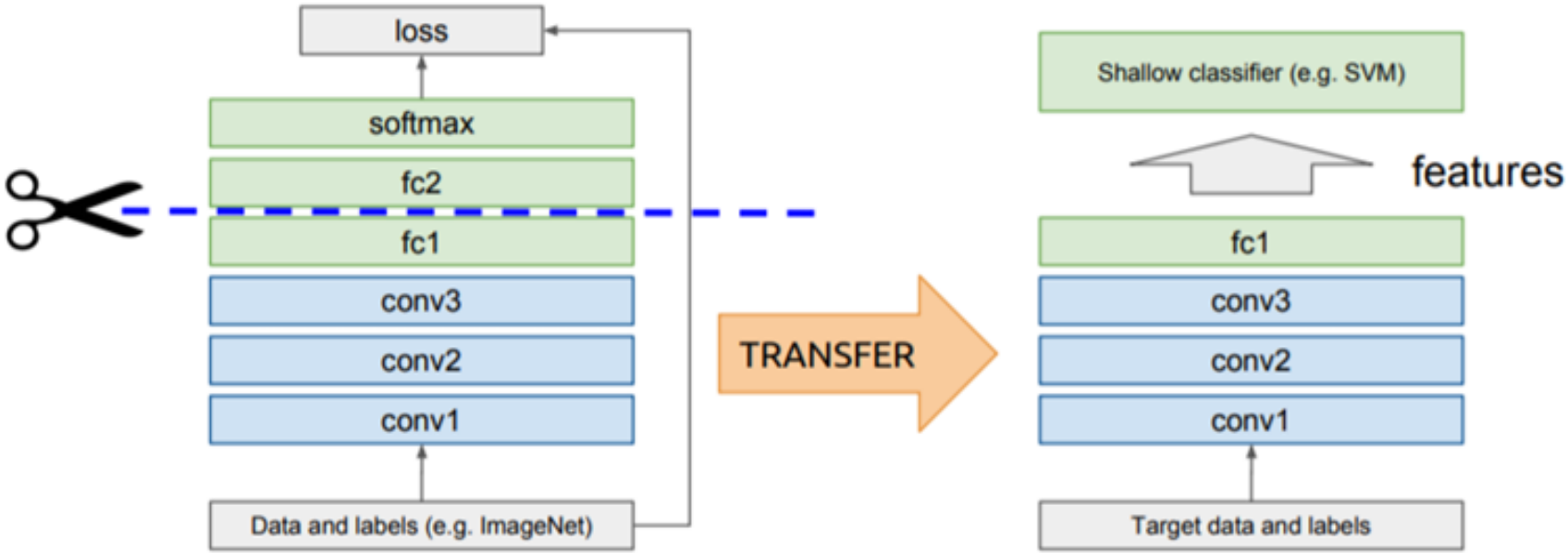
---

*Sinno Jialin Pan, Qiang Yang.* A Survey on Transfer Learning. 2009

# Пред-обученные (pre-trained) нейронные сети

Свёрточная сеть для обработки изображений:

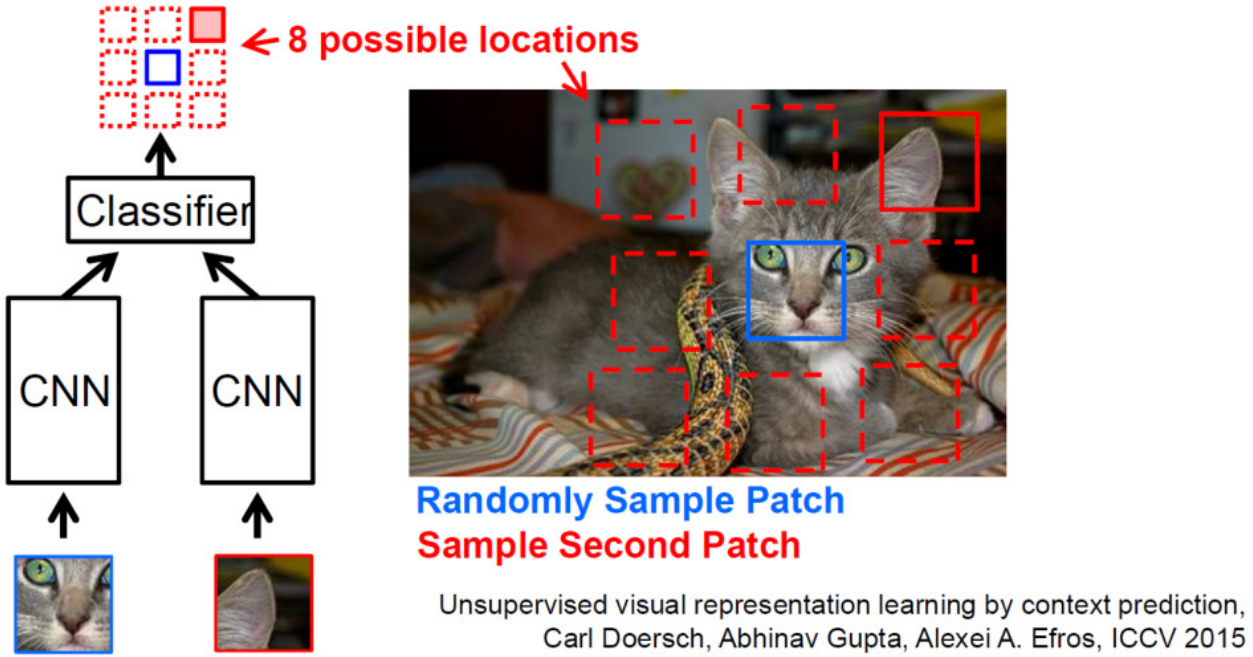
- $f(x, \alpha)$ — свёрточные слои для векторизации объектов
- $g(x, \beta)$ — полносвязные слои под конкретную задачу




*Jason Yosinski, Jeff Clune, Yoshua Bengio, Hod Lipson.* How transferable are features in deep neural networks? 2014.

# Самообучение (self-supervised learning)

В компьютерном зрении сеть учится предсказывать взаимное расположение двух фрагментов на одном изображении

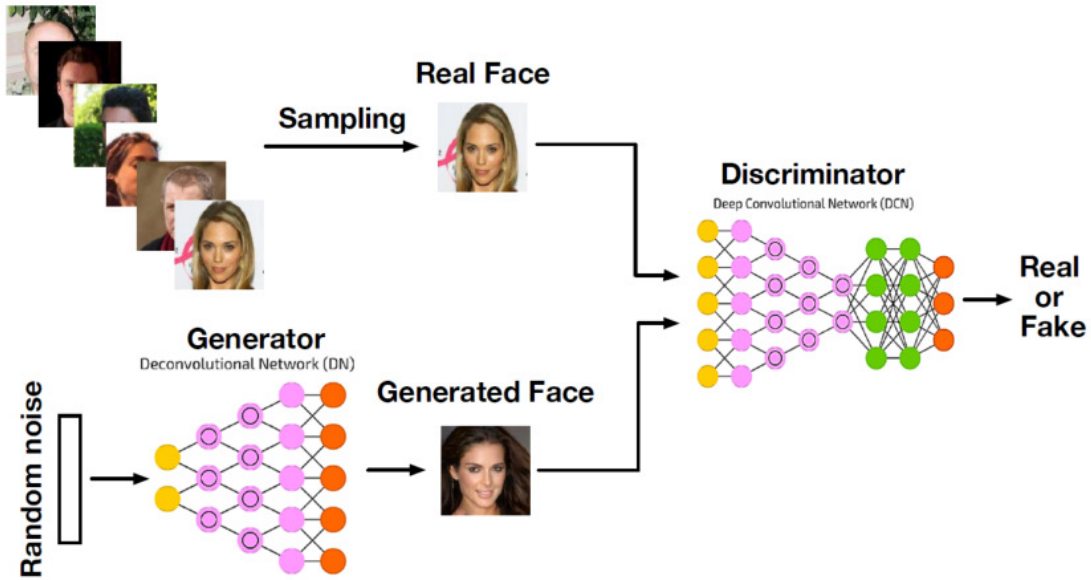



Unsupervised visual representation learning by context prediction,
Carl Doersch, Abhinav Gupta, Alexei A. Efros, ICCV 2015

**Преимущество:** сеть выучивает векторные представления объектов без размеченной обучающей выборки.

# Генеративная состязательная сеть (Generative Adversarial Net)

*Генератор* $G(z)$ учится порождать объекты $x$ из шума $z$
*Дискриминатор* $D(x)$ учится отличать их от реальных объектов




*Antonia Creswell et al.* Generative Adversarial Networks: an overview. 2017.
*Zhengwei Wang, Qi She, Tomas Ward.* Generative Adversarial Networks: a survey and taxonomy. 2019.
*Chris Nicholson.* A Beginner's Guide to Generative Adversarial Networks. https://pathmind.com/wiki/generative-adversarial-network-gan. 2019.

# Постановка задачи GAN

**Дано:** выборка объектов $\{x_i\}_{i=1}^m$ из $X$

**Найти:**

вероятностную генеративную модель $G(z,\alpha)$: $x \sim p(x|z,\alpha)$

вероятностную дискриминативную модель $D(x,\beta) = p(1|x,\beta)$

**Критерий:**

обучение дискриминативной модели $D$:

$$\sum_{i=1}^{m} \ln D(x_i,\beta) + \ln\big(1 - D\big(G(z_i,\alpha),\beta\big)\big) \;\to\; \max_{\beta}$$

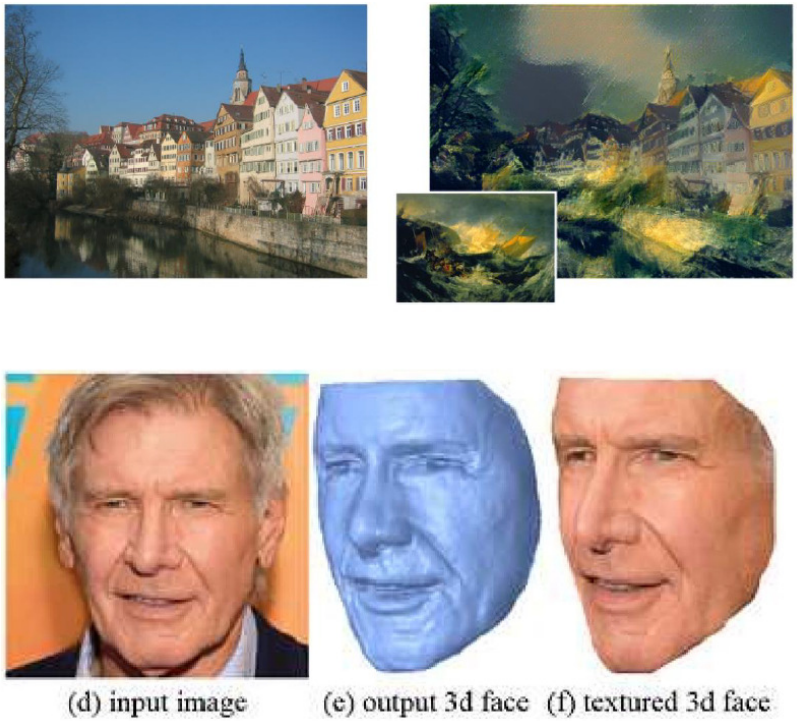
обучение генеративной модели $G$:

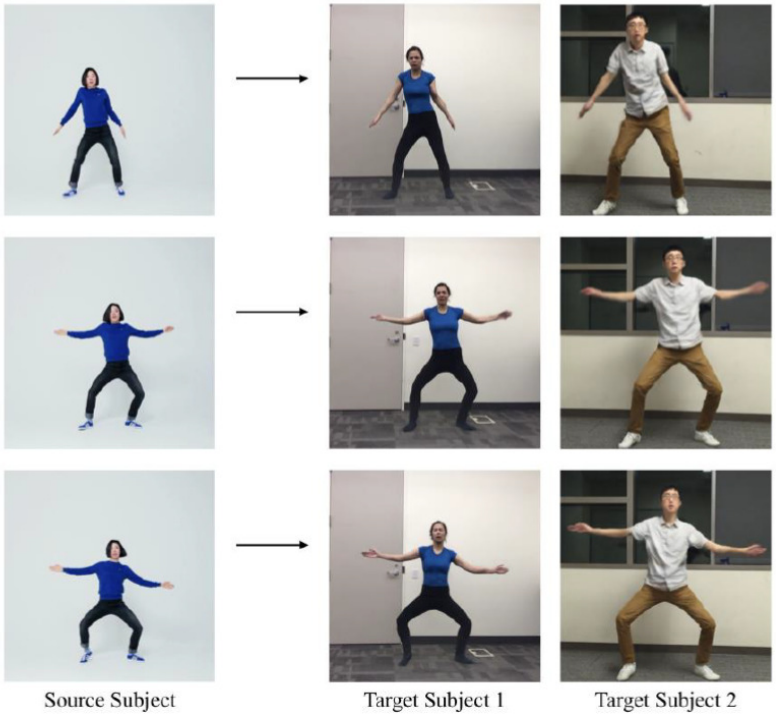
$$\sum_{i=1}^{m} \ln\big(1 - D\big(G(z_i,\alpha),\beta\big)\big) \;\to\; \min_{\alpha}$$

---

*Ian Goodfellow et al.* Generative Adversarial Nets. 2014

# Примеры GAN для синтеза изображений и видео




*Chuan Li, Michael Wand.* Precomputed Real-Time Texture Synthesis with Markovian Generative Adversarial Networks. 2016.

*Xiaoxing Zeng, Xiaojiang Peng, Yu Qiao.* DF2Net: A Dense Fine Finer Network for Detailed 3D Face Reconstruction. ICCV-2019.

*Caroline Chan, Shiry Ginosar, Tinghui Zhou, Alexei A. Efros.* Everybody Dance Now. ICCV-2109.

# Резюме

- *Свёрточные сети* переводят сложно структурированные данные в вектор фиксированной размерности
- *Рекуррентные сети* позволяют обрабатывать последовательности векторов
- Векторные представления дискретных объектов: word2vec, graph2vec, StarSpace и другие
- *Состязательные сети* способны генерировать сложные реалистичные объекты
- Подбор архитектуры и параметров сети — это искусство