Лаборатория компьютерной
графики и мультимедиа
ВМК МГУ имени М.В. Ломоносова

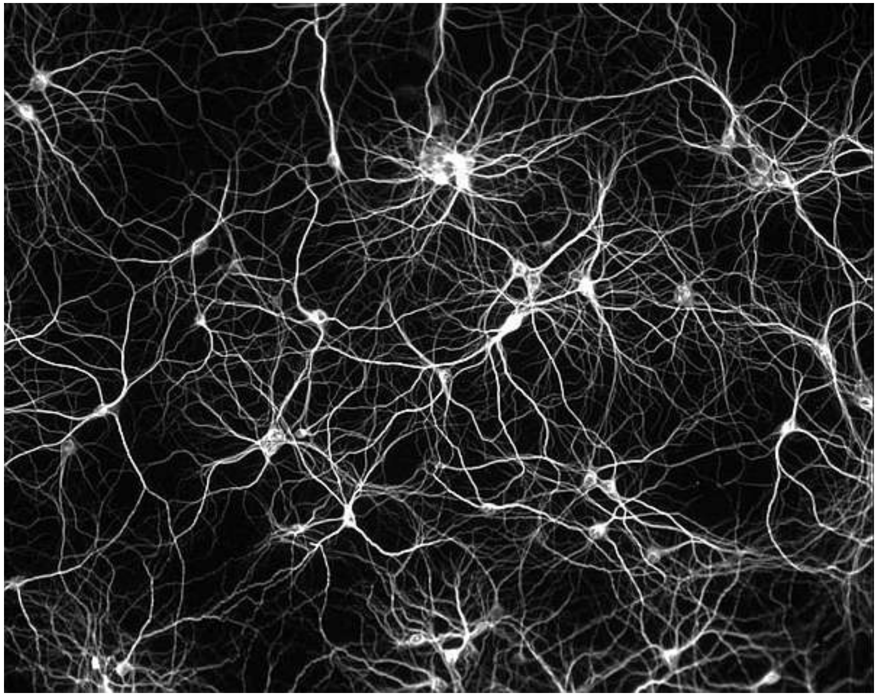
Курс «Введение в компьютерное зрение
и глубокое обучение»

Лекция №5
«Свёрточные нейросети, ч.1»

Антон Конушин
Заведующий лабораторией компьютерной графики и мультимедиа
ВМК МГУ

18 марта 2019 года

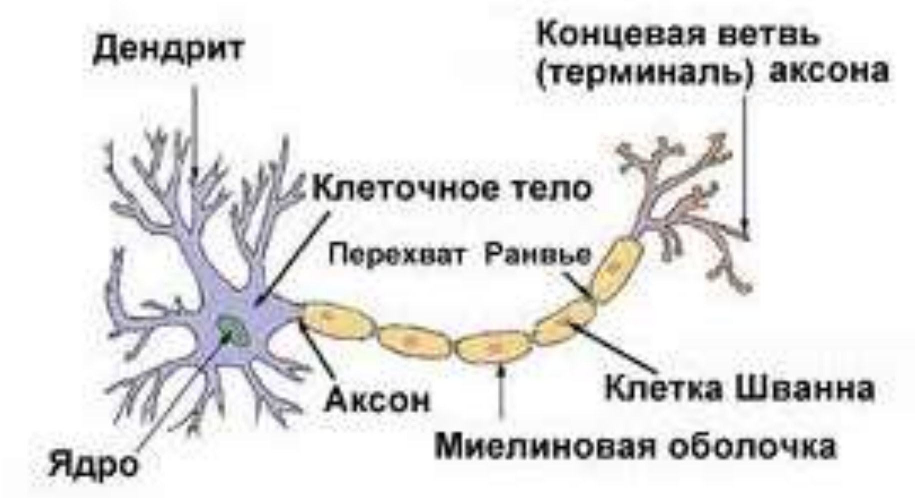
Структура мозга

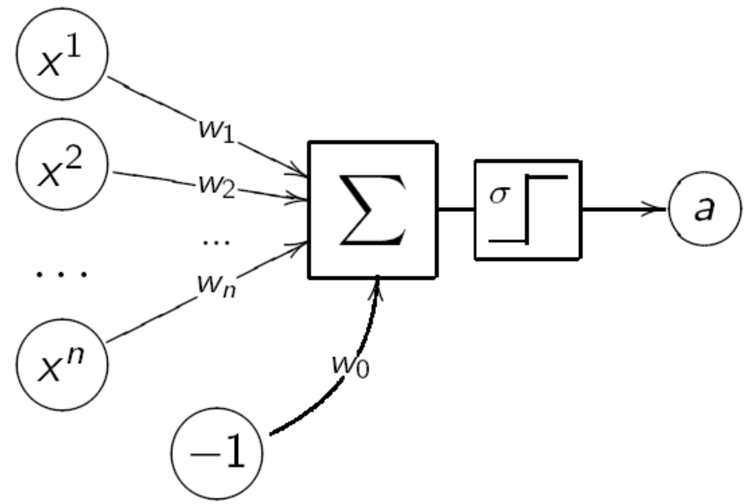


Нейросеть

Отдельный нейрон

Линейная модель МакКаллока-Питтса





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

$\sigma(z)$ — функция активации (например, sign),

w_j — весовые коэффициенты синаптических связей,

w_0 — порог активации,

$w, x \in \mathbb{R}^{n+1}$, если ввести константный признак $f_0(x) \equiv -1$

Задачи для нейросетей

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

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

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

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

- Важно: для разных задач мы будем пользоваться разными функциями активации.
- **sign** подойдет для классификации
- Для регрессии подойдет непрерывная функция активации

Важный вопрос

Какие функции представимы с помощью нейросетей?

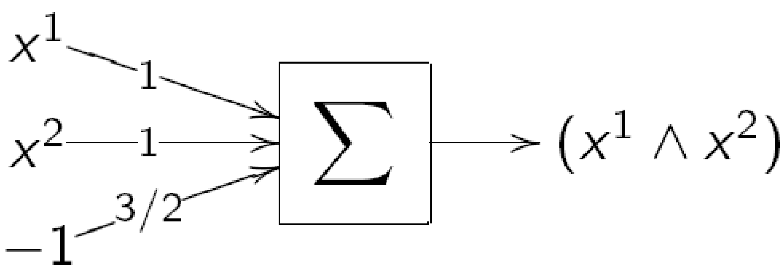
Логические элементы

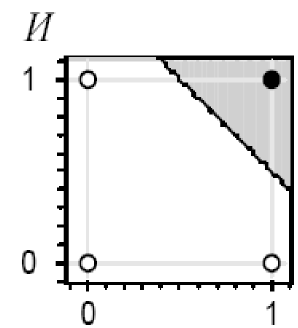
Функции И, ИЛИ, НЕ от бинарных переменных x^1 и x^2:

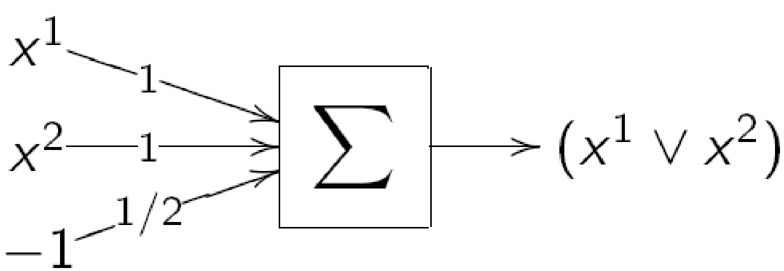
$$x^1 \wedge x^2 = \left[x^1 + x^2 - \tfrac{3}{2} > 0\right];$$

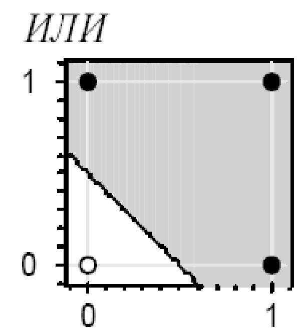
$$x^1 \vee x^2 = \left[x^1 + x^2 - \tfrac{1}{2} > 0\right];$$

$$\neg x^1 = \left[-x^1 + \tfrac{1}{2} > 0\right];$$





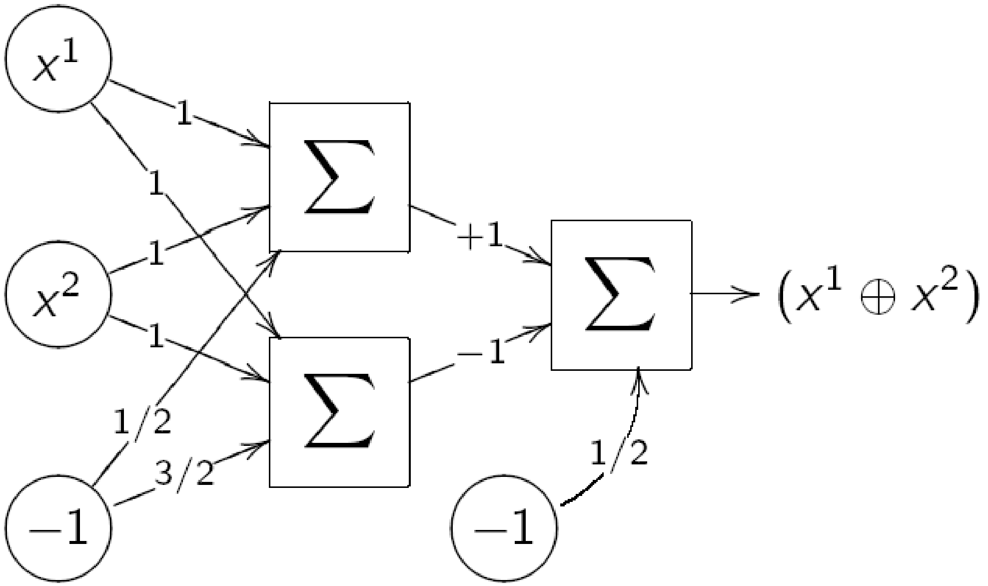


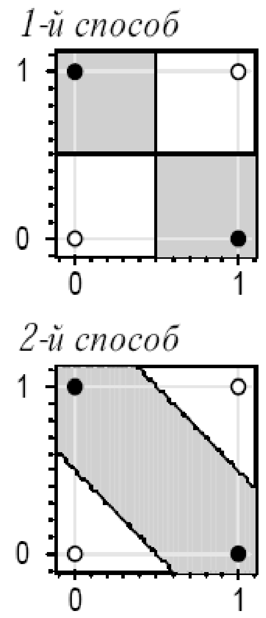


Исключающее ИЛИ (XOR)

Функция $x^1 \oplus x^2 = [x^1 \neq x^2]$ не реализуема одним нейроном.
Два способа реализации:

- Добавлением нелинейного признака:
  $x^1 \oplus x^2 = \left[x^1 + x^2 - 2x^1x^2 - \frac{1}{2} > 0\right]$;
- Сетью (двухслойной суперпозицией) функций И, ИЛИ, НЕ:
  $x^1 \oplus x^2 = \left[(x^1 \vee x^2) - (x^1 \wedge x^2) - \frac{1}{2} > 0\right]$.





Приближение функций нейросетью

Утверждение

Любая булева функция представима в виде ДНФ, следовательно, и в виде двухслойной сети.

Решение тринадцатой проблемы Гильберта:

Теорема (Колмогоров, 1957)

Любая непрерывная функция n аргументов на единичном кубе $[0,1]^n$ представима в виде суперпозиции непрерывных функций одного аргумента и операции сложения:

$$f(x^1, x^2, \dots, x^n) = \sum_{k=1}^{2n+1} h_k\left(\sum_{i=1}^{n} \varphi_{ik}(x^i)\right),$$

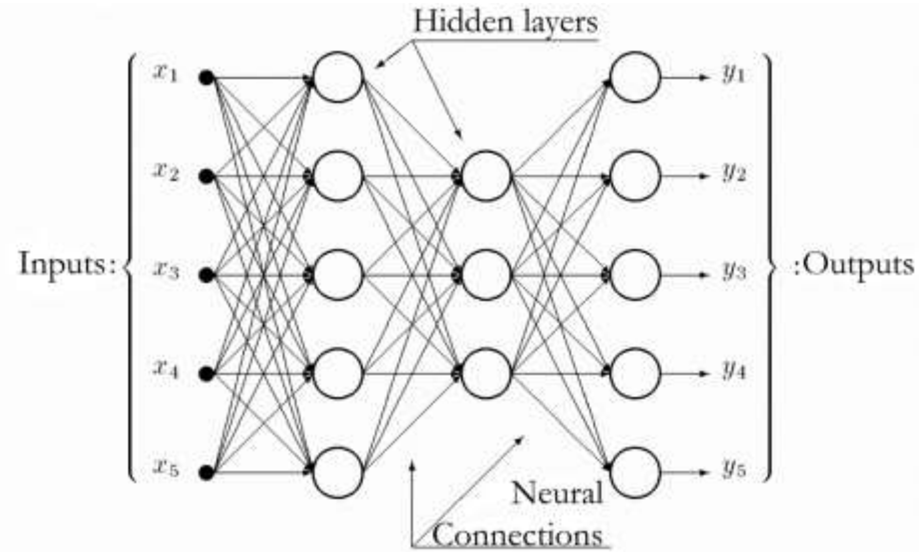
где h_k, φ_{ik} — непрерывные функции, и φ_{ik} не зависят от f.

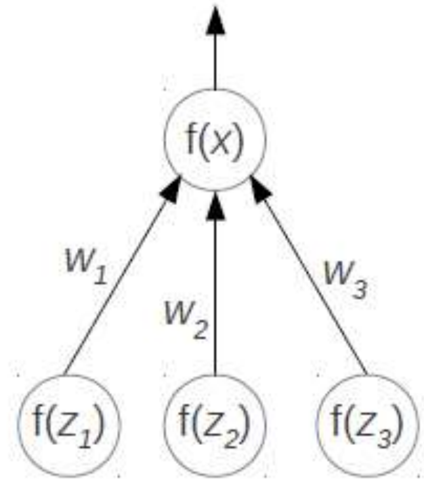
Представимость функций

- Итого, теоретически доказано, что с помощью линейных операций и одной нелинейной функции активации можно вычислить любую непрерывную функцию с любой желаемой точностью

- Однако из доказательств не следует, как должна быть устроена сеть, сколько в ней должно быть нейронов, какие у них должны быть веса

Задание нейросети





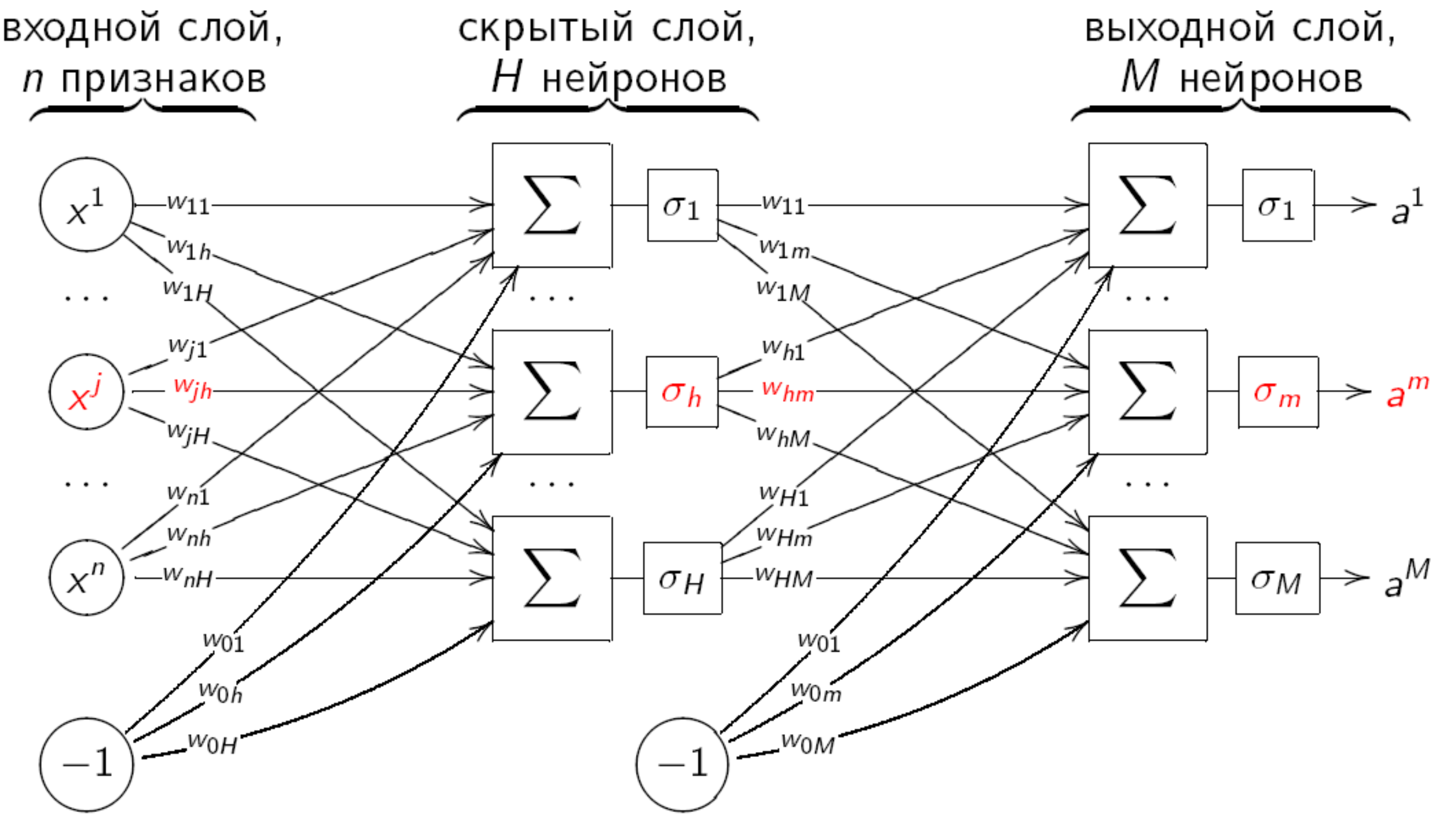
$x = w_1 f(z_1) + w_2 f(z_2) + w_3 f(z_3)$

x is called the total input to the neuron, and $f(x)$ is its output

- **Архитектура нейросети** – взвешенный ориентированный граф, в котором вершины – нейроны, ребра – связи
- **Веса нейросети** – совокупность весов всех рёбер (веса каждого нейрона)
- Обучение нейросети = подбор оптимальных весов
- Подбор архитектуры:
  - Разработчиком, исходя из опыта и «лучших примеров»
  - Есть работы по «автоподбору» архитектуры
  - Есть методы «упрощения» архитектуры

Многослойная нейросеть

Пусть для общности $Y = \mathbb{R}^M$, для простоты слоёв только два.

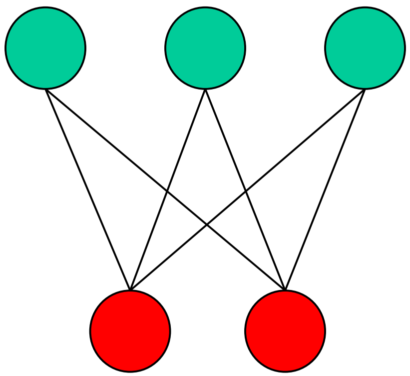


- Передача сигналов идёт в одном направлении (feed-forward)
- Сеть можно разделить на «слои», по числу предшествующих нейронов на пути сигнала

Архитектура нейросети

- Мы будем рассматривать архитектуры, применяемые для анализа изображений
- Вначале нужно определить вход и выход
- Вход – обычно изображение $I \in \mathbb{R}^{width \times height \times c}$ (трёхмерная матрица)
- Выход – ответы на задачу, которую мы поставили
  - Бинарная классификация
    - Один нейрон (+1, -1)
    - Два нейрона – один для +1 класса, второй для -1 класса
  - Многоклассовая классификация с K-классами
    - K нейронов
  - Регрессия
    - Одно число – один нейрон на выходе
    - K чисел – k выходных нейронов (пример - (x1, y1, x2, y2) для bbox объекта при решении задачи детекции)

Линейный перспептрон



Простейшая нейронная сеть с входом и выходом, реализующая линейную функцию:

$$NN_{perceptron}(x) = xW + b$$

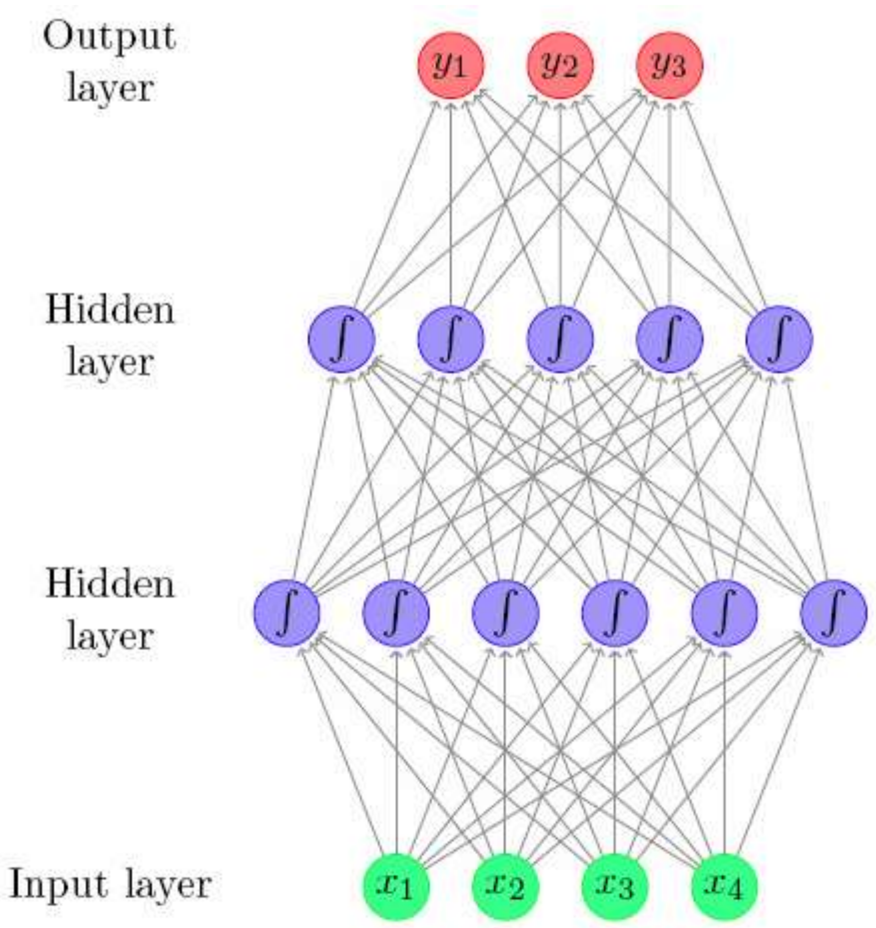
$X \in \mathbb{R}^{d_{in}}$ - вектор входа

$W \in \mathbb{R}^{d_{in} x d_{out}}$ - матрица весов всех нейроннов выходного слоя

$b \in \mathbb{R}^{d_{out}}$ - вектор сдвигов

Вход и выход мы видим, поэтому это видимые слои

Многослойный персептрон



- Добавим промежуточные слои.
- Они будут «скрытые»
- В перспептроне в каждом слое каждый нейрон связан с каждым нейроном предыдущего слоя
- Такие слои называются *полносвязанными (fully connected)*
- Нейроны скрытых слоёв обычно имеют функцию активации (нелинейную)

Функция перспетрона

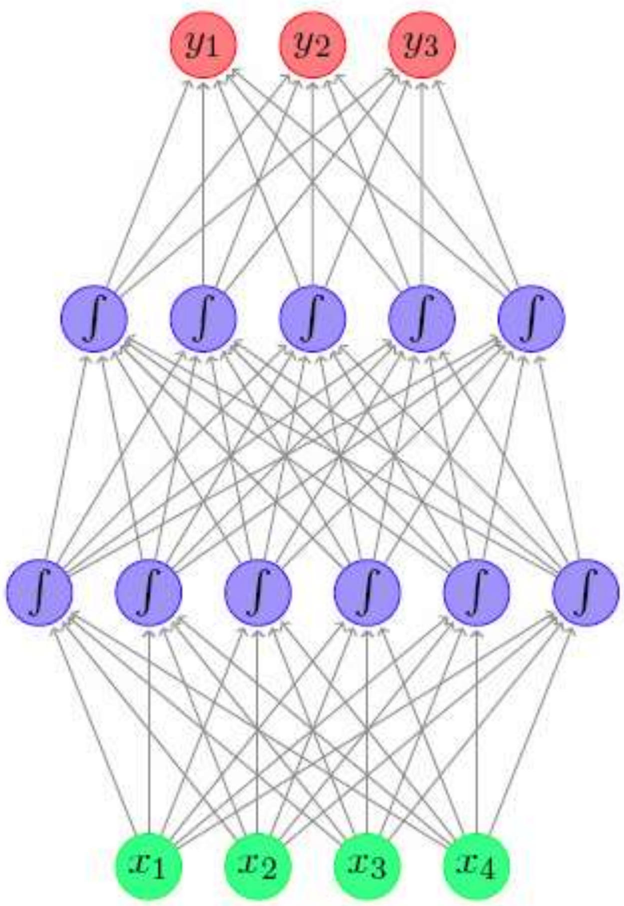
Варианты записей:

- Как послойные преобразования:

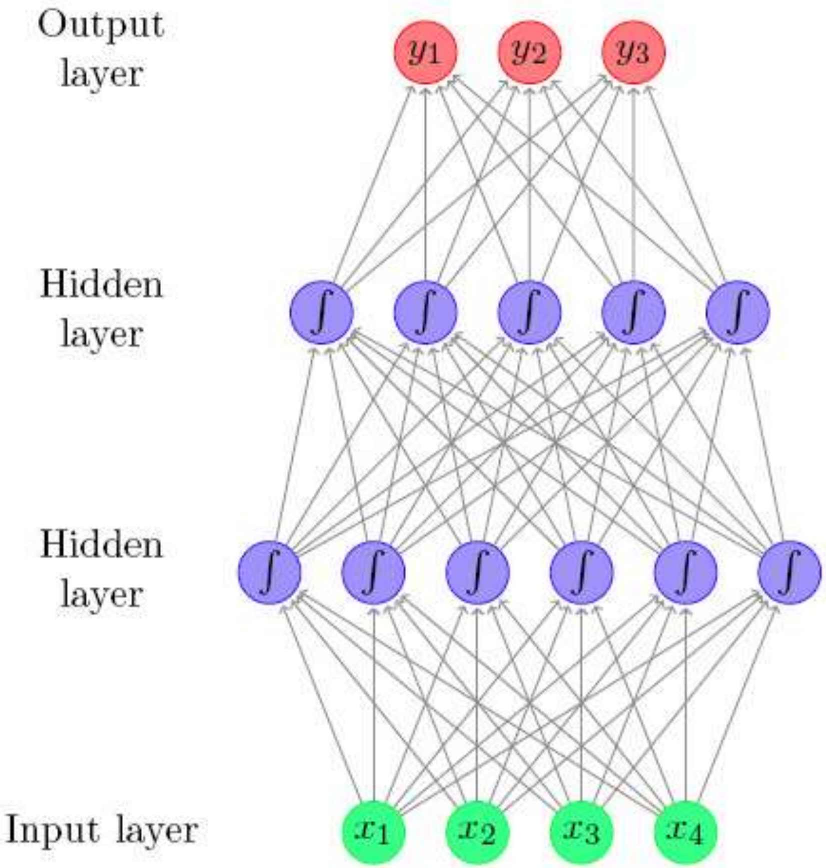
$$
\begin{aligned}
NN_{MLP2}(\mathbf{x}) =& \mathbf{y} \\
\mathbf{h}^1 =& g^1(\mathbf{x}\mathbf{W}^1 + \mathbf{b}^1) \\
\mathbf{h}^2 =& g^2(\mathbf{h}^1\mathbf{W}^2 + \mathbf{b}^2) \\
\mathbf{y} =& \mathbf{h}^2\mathbf{W}^3
\end{aligned}
$$

- Одна функция:

$$NN_{MLP2}(\mathbf{x}) = (g^2(g^1(\mathbf{x}\mathbf{W}^1 + \mathbf{b}^1)\mathbf{W}^2 + \mathbf{b}^2))\mathbf{W}^3$$



Персептрон



- С помощью персептрона (даже только с 1 скрытым слоем) мы можем задавать произвольные функции
- Как обучать (настраивать веса)?

Обучение нейросети

- Умеем обучать классификаторы с помощью градиентного спуска
- Нам потребуется:
  - Задать функцию потерь (целевой функционал)
  - Инициализировать веса сети
  - Применить метод градиентного спуска к нейросети
- Пока посмотрим только классификацию

Бинарная классификация

- Пусть у нейросети один выход y
- y метка из {-1,1}
- $\hat{y}$ - выход классификатора (вещественное число)
- Ответ классификатора даётся в виде $\mathrm{sign}(\hat{y})$
- Ответ правильный, если $y\hat{y}$>0 (одинаковый знак)
- Ошибка **hinge loss** (как в SVM):

$$L_{hinge(binary)}(\hat{y}, y) = \max(0, 1 - y \cdot \hat{y})$$

- Потери 0, если ответ правильный и $|\hat{y}| \geq 1$ (уверенный)

Многоклассовая классификация

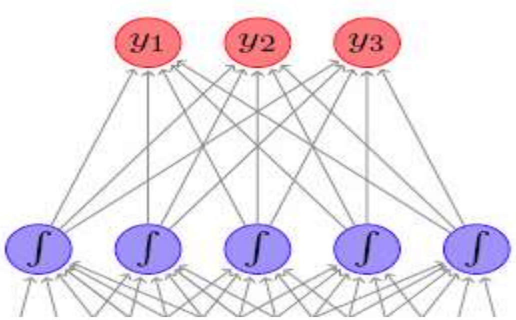
- У каждого класса i свой выходной нейрон y_i
- Можем воспользоваться кросс-энтропией как функцией потерь

$$L_{cross-entropy}(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_i y_i \log(\hat{y}_i)$$

- Измеряет близость истинного y и оцененного $\hat{y}$ распределения меток
- Чтобы $\hat{y}$ можно было трактовать как распределение, добавим в конце слой softmax

$$\mathrm{x} = x_1, \dots, x_k$$

$$softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{k} e^{x_j}}$$



- «Сжимаем» произвольный вектор длины N в вектор со значениями (0,1) и суммой 1

Инициализация весов сети

- Пусть веса слоя $\mathbf{W} \in \mathbb{R}^{d_{in} \times d_{out}}$
- Есть несколько методов инициализации весов
- Инициализация Xavier (2010):

$$\mathbf{W} \sim U\left[-\frac{\sqrt{6}}{\sqrt{d_{in} + d_{out}}}, +\frac{\sqrt{6}}{\sqrt{d_{in} + d_{out}}}\right]$$

где U[a,b] – равномерное распределение в интервале

- Инициализация He (2015):

Нормальное распределение с матожиданием 0 и стандартным отклонением

$$\sqrt{\frac{2}{d_{in}}}.$$

Есть и более новые!

Стохастический градиентный спуск

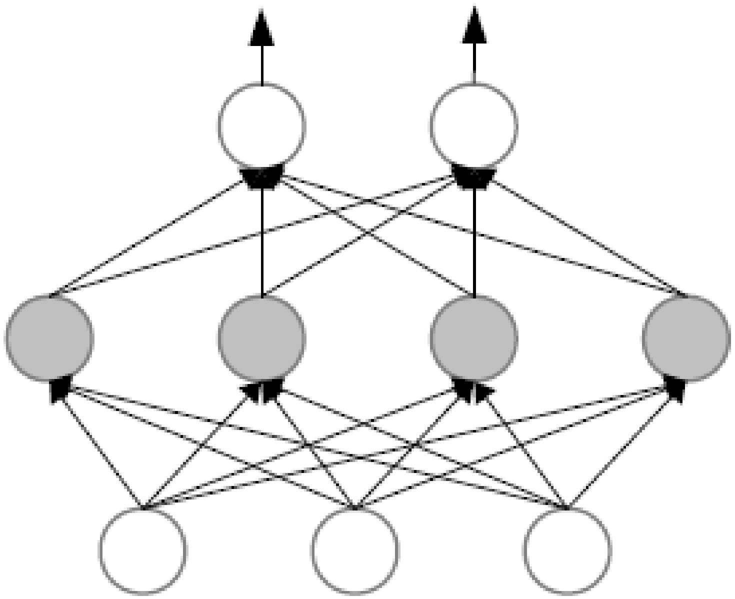
Точно также, как при обучении одного нейрона (линейного классификатора), можем обучать все параметры нейросети

Algorithm 1 Online Stochastic Gradient Descent Training

1: **Input:** Function $f(\mathbf{x};\theta)$ parameterized with parameters θ.
2: **Input:** Training set of inputs $\mathbf{x_1},\ldots,\mathbf{x_n}$ and outputs $\mathbf{y_1},\ldots,\mathbf{y_n}$.
3: **Input:** Loss function L.
4: **while** stopping criteria not met **do**
5: Sample a training example $\mathbf{x_i},\mathbf{y_i}$
6: Compute the loss $L(f(\mathbf{x_i};\theta),\mathbf{y_i})$
7: $\hat{\mathbf{g}} \leftarrow$ gradients of $L(f(\mathbf{x_i};\theta),\mathbf{y_i})$ w.r.t θ
8: $\theta \leftarrow \theta + \eta_k \hat{\mathbf{g}}$
9: **return** θ

Как посчитать градиент для всех параметров нейросети?

Расчёт градиента



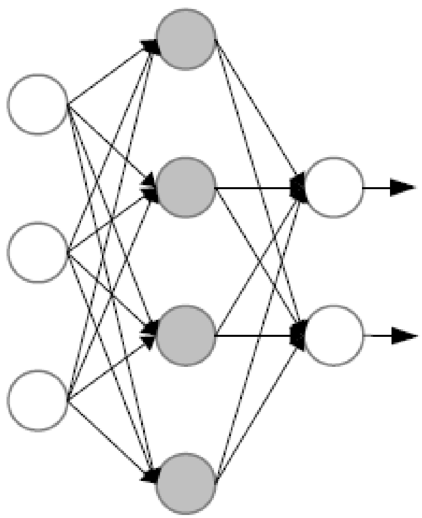
- Нейросеть вычисляет дифференцируемую функцию от своих входов
- Можем последовательно применять правило дифференцирования сложных функций для вычисления производных по каждому параметру нейросети
- Метод получил название «обратное распространение ошибки» и имеет длинную историю

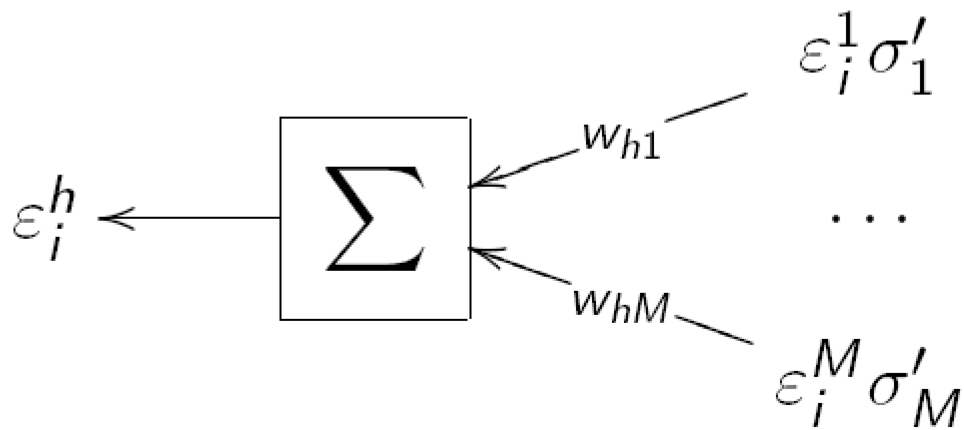
•↑ *Галушкин А. И.* Синтез многослойных систем распознавания образов. — М.: «Энергия», 1974.

•↑ *Werbos P. J.*, Beyond regression: New tools for prediction and analysis in the behavioral sciences. Ph.D. thesis, Harvard University, Cambridge, MA, 1974.

•↑ *1* *2* *Rumelhart D.E., Hinton G.E., Williams R.J.*, Learning Internal Representations by Error Propagation. In: Parallel Distributed Processing, vol. 1, pp. 318—362. Cambridge, MA, MIT Press. 1986.

Процедура обратного распространения ошибки





- Градиент для каждого параметра нейрона можем рассчитать как взвешенное произведение ошибок всех связанных с ним последующий нейронов на производную функции активации нейрона
- Вначале посчитаем градиент нейронов выходного слоя, затем предыдущего и т.д.
- Для всей сети мы можем это сделать за один обратный проход, как-бы запустив сеть «задом наперёд»

Minibatch SGD

Algorithm 2 Minibatch Stochastic Gradient Descent Training

Input: Function $f(\mathbf{x};\theta)$ parameterized with parameters θ.
Input: Training set of inputs $\mathbf{x_1},\ldots,\mathbf{x_n}$ and outputs $\mathbf{y_1},\ldots,\mathbf{y_n}$.
Input: Loss function L.
while stopping criteria not met **do**
  Sample a minibatch of m examples $\{(\mathbf{x_1},\mathbf{y_1}),\ldots,(\mathbf{x_m},\mathbf{y_m})\}$
  $\hat{\mathbf{g}} \leftarrow 0$
  for $i=1$ to m **do**
    Compute the loss $L(f(\mathbf{x_i};\theta),\mathbf{y_i})$
    $\hat{\mathbf{g}} \leftarrow \hat{\mathbf{g}} +$ gradients of $\frac{1}{m}L(f(\mathbf{x_i};\theta),\mathbf{y_i})$ w.r.t θ
  $\theta \leftarrow \theta + \eta_k \hat{\mathbf{g}}$
return θ

- Формируем случайную группу примеров (minibatch) и усредняем градиенты по всем примерам
- Много плюсов:
  - Менее шумный градиент (по сравнению с одним примеров)
  - Быстрее сходимся (меняем параметры после расчёта части примеров, а не всех)
  - Вычислительно эффективнее (в RAM держим minibatch)

Заметки по процедуре обучения

- Темп обучения выбирают таким образом, чтобы веса не слишком колебались и сходились
- С течением времени темп обучения уменьшают, чтобы обеспечить сходимость весов
- «Эпоха» – этап обучения на всех примерах из обучающей выборки
- Обычно обучение состоит из нескольких эпох, между которыми меняют параметры обучения
    - Обычно экспоненциально снижает скорость обучения

Расписание

- Очень важен процесс формирования mini-batch
- Он должен хорошо представлять обучающую выборку
- В нём должны присутствовать и положительные, и отрицательные примеры (часто в равных долях)
- Можно управлять составлением minibatch, составив расписание

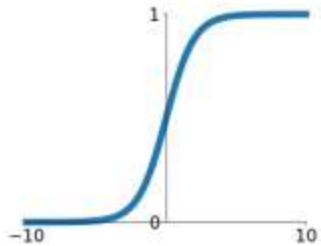
Развитие SGD

- Сохранение и использование градиентов с предыдущих минибатчей
  - SGD + Momentum (Поляк, 1964)
  - Nesterov momentum
- Адаптивный выбор темпа обучения для каждого минибатча
  - AdaGrad
  - AdaDelta
  - Adam
  - И т.д.
- Обычно все реализованы в библиотеках и можно экспериментировать с ними

Функции активации

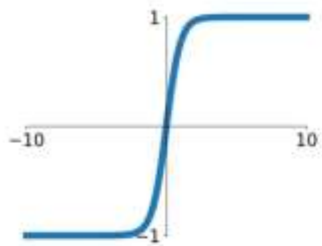
Sigmoid

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



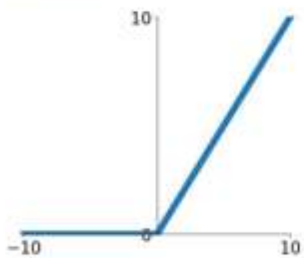
tanh

$\tanh(x)$



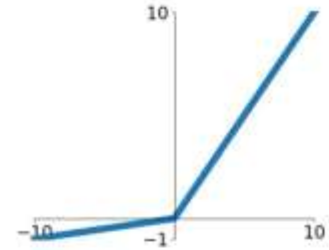
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

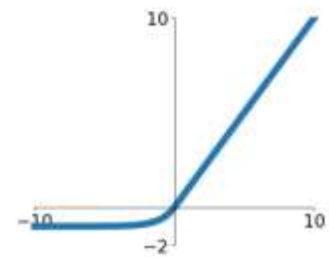


Maxout

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

ELU

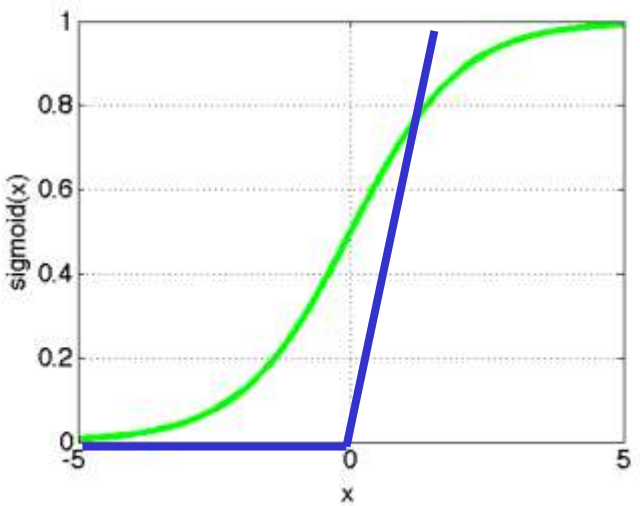
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Source: http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture06.pdf

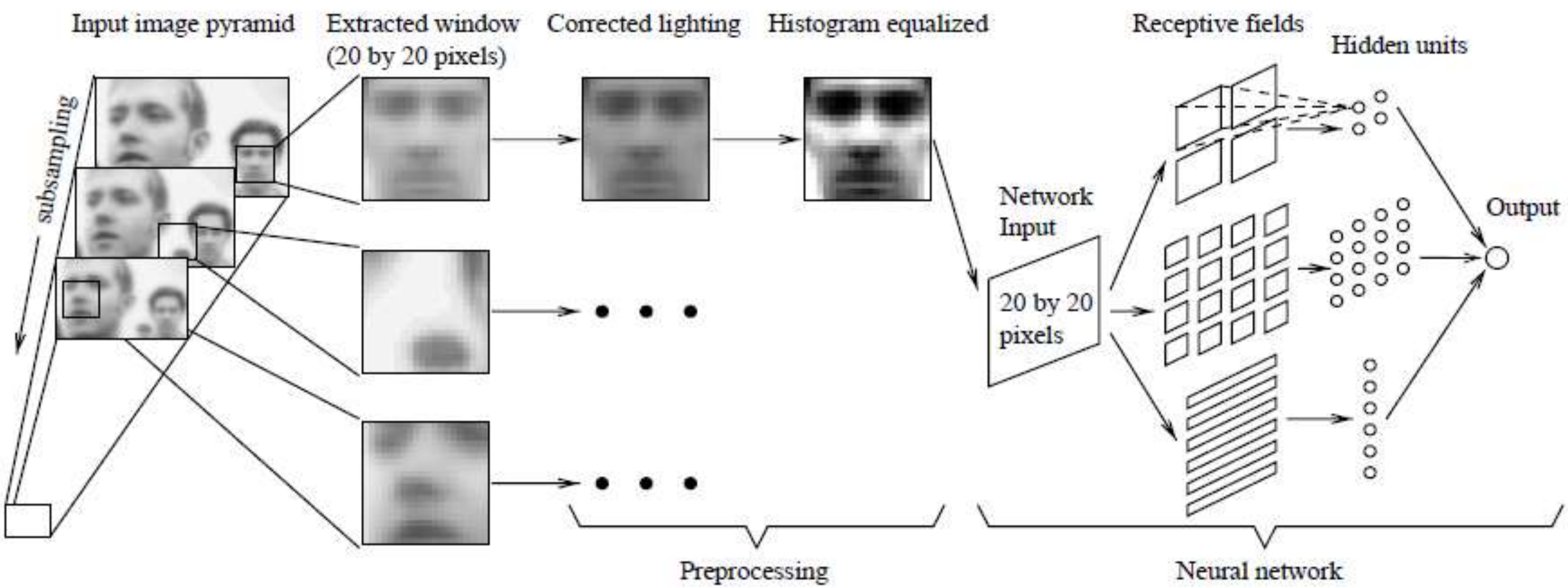
Проблемы обучения

- «Затухание» (saturation) градиентов, если попадаем на область низкого градиента функции активации
  - Sigmoid очень страдает от этого
  - ReLU не страдает
- «Мёртвые» (dead) нейроны, которые, так получилось, получают только отрицательные или нулевые входы
- Исчезающие или взрывные градиенты (vanishing or exploding gradients) в глубоких сетях



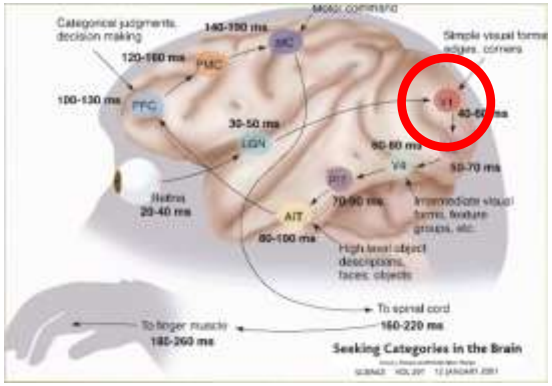
Rowley face detector (1998)

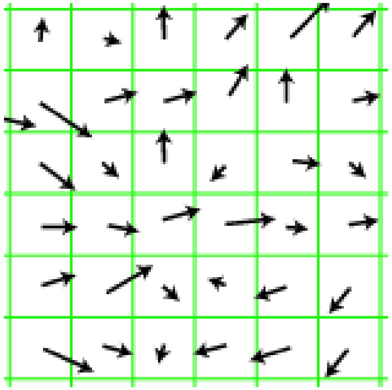
- Метод обратного распространения ошибки оказался очень эффективным
- Пример – детектор лица, лучший до Viola-Jones

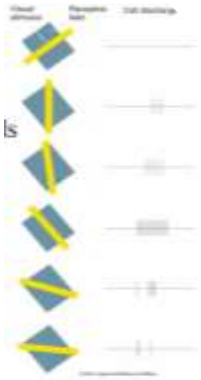


B. Rowley, T. Kanade. Neural Network-Based Face Detection. PAMI, 1998.

Использование знаний о зрении



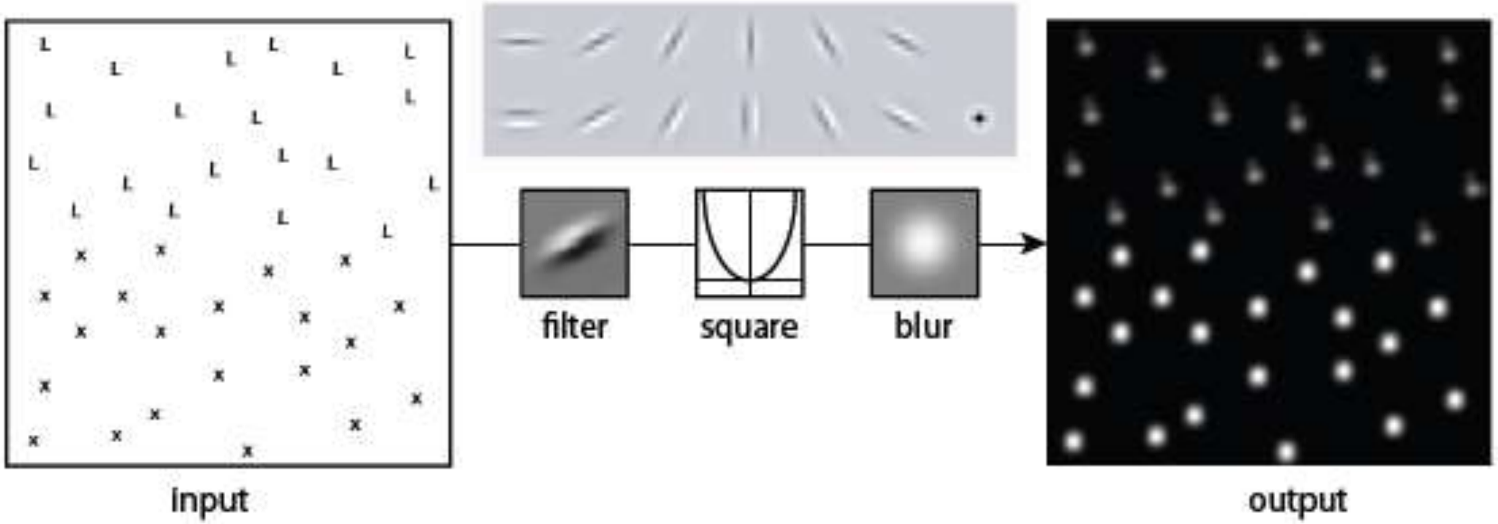




- Ранее мы не раз упоминали о важности информации о краях и градиентах для распознавания изображений
- На этой идее основаны гистограммы градиентов
- Что нам нужно для расчёта гистограмм градиентов?

Pietro Perona and Jitendra Malik «Detecting and Localizing edges composed of steps, peaks and roofs», ICCV 1990

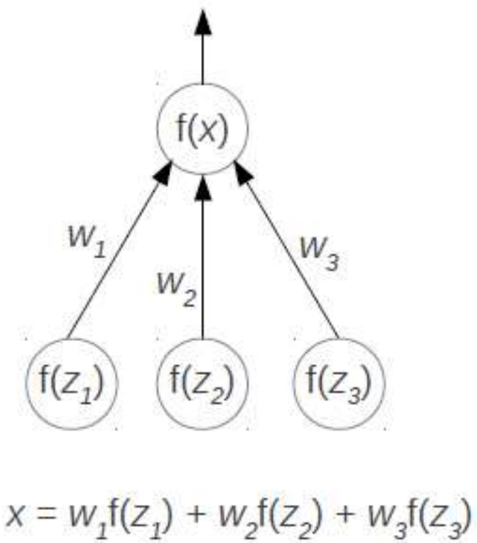
Банки текстурных фильтров

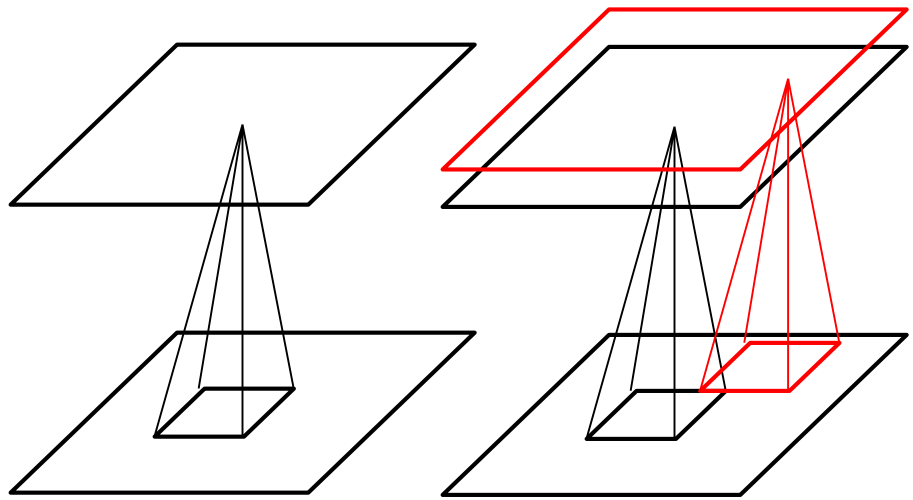


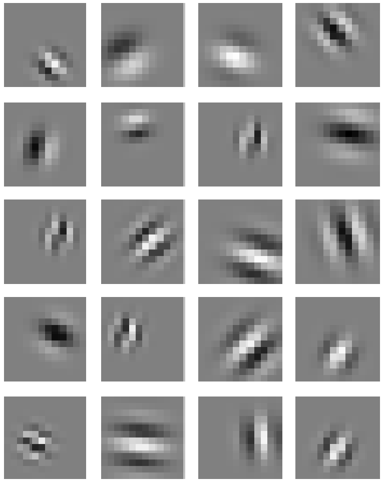
- Выберем набор (банк) фильтров, каждый из которых чувствителен к краю определенной ориентации и размера
- Каждый пиксель изображения после обработки банком фильтров даёт вектор признаков
- Этот вектор признаков эффективно описывает локальную текстуру окрестности пикселя

Pietro Perona and Jitendra Malik «Detecting and Localizing edges composed of steps, peaks and roofs», ICCV 1990

Нейрон как линейный фильтр

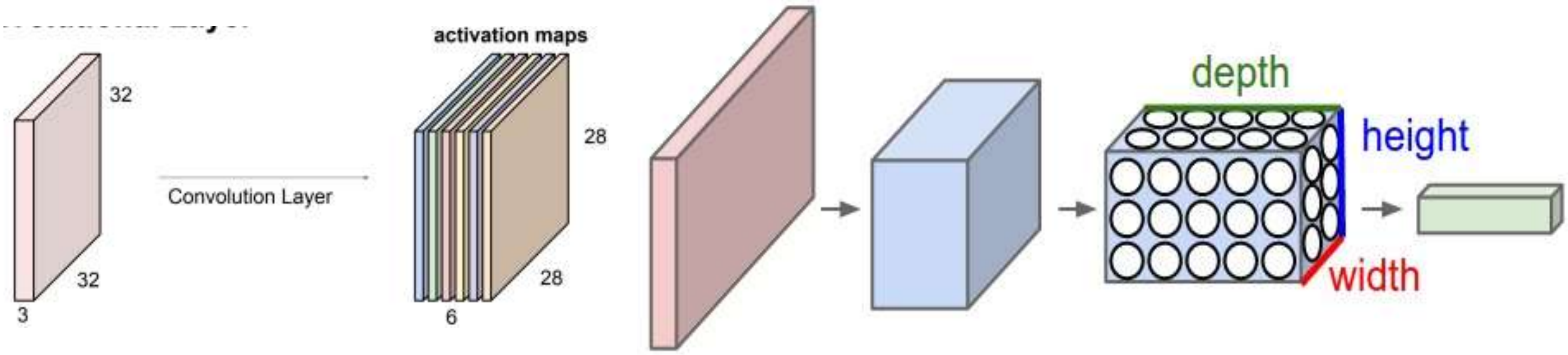






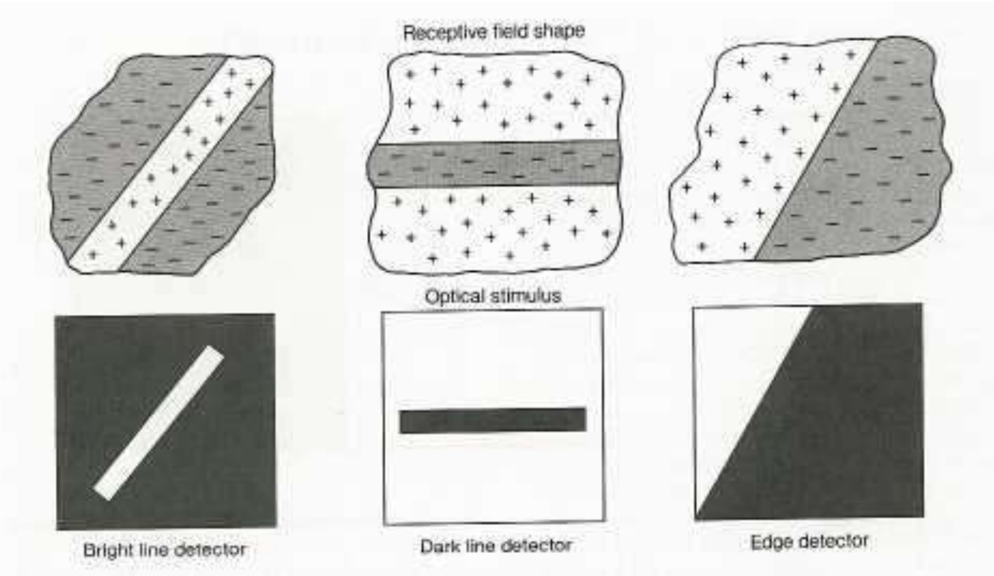
- Операцию линейной фильтрации (свёртки) для одного пикселя можно реализовать одним нейроном
- Свёртку изображения целиком можно реализовать как «слой» нейронов, веса которых одинаковы
- Набор свёрток, которые применяются к одному и тому же входу, мы объединяем в «свёрточный слой»
- Свёрточный слой реализует операцию обработки входа банком фильтров

Пояснение свёрточного слоя

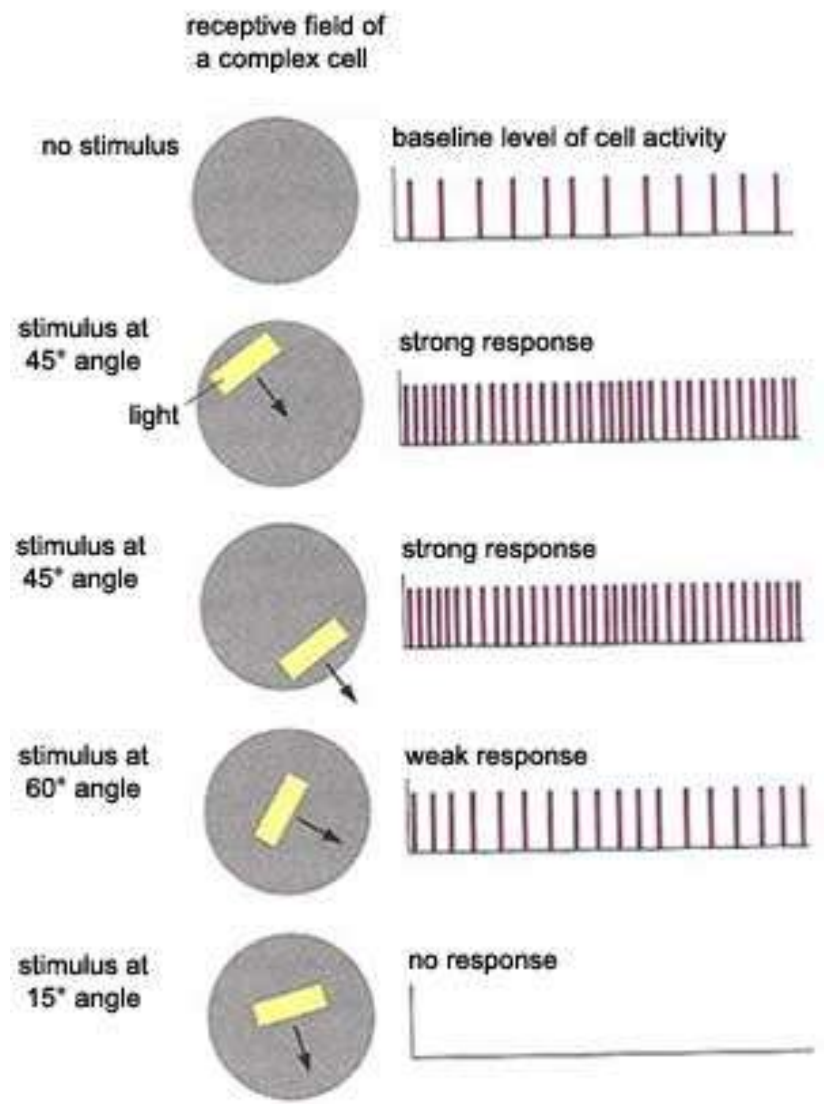


- Каждая свёртка даёт на выходе изображение nxn пикселей, называемое картой активации (activation map)
- Мы объединяем карты активаций в в одну трёхмерную матрицу (многоканальное изображение)
- Глубина (depth) получившейся матрицы равна числу свёрток в свёрточном слое

Простые (S) и сложные (C) клетки



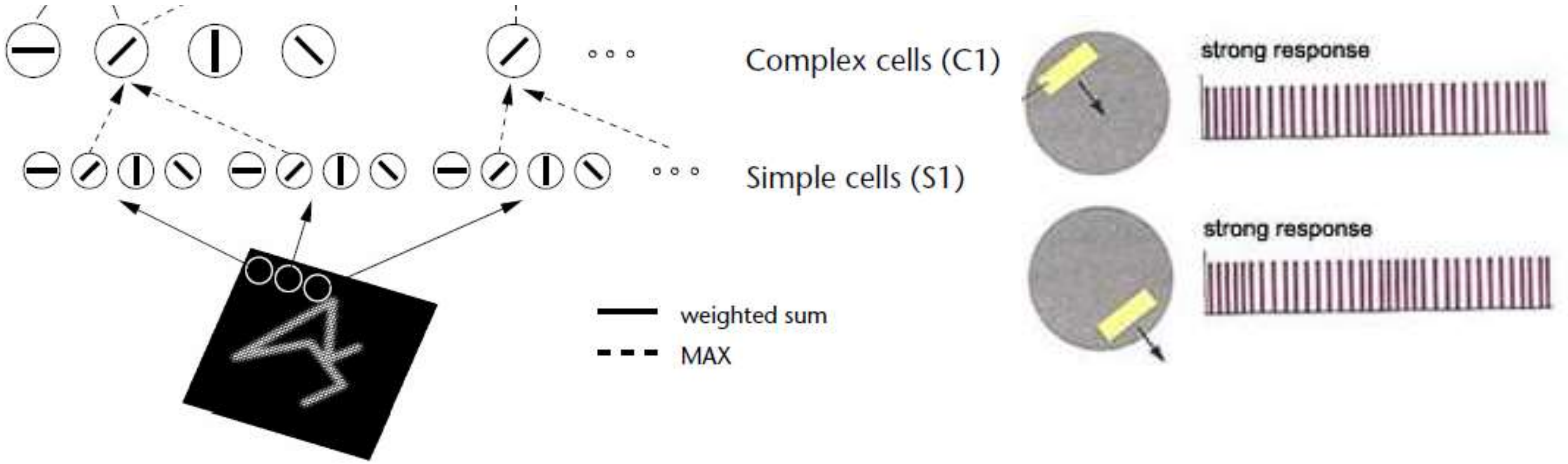
Простые клетки



- Простые клетки чувствительны к контрастным объектам определённого размера, ориентации и положения
- Сложные клетки ***инвариантны*** к сдвигам в небольшой окрестности
- Как их смоделировать?

Сложные клетки

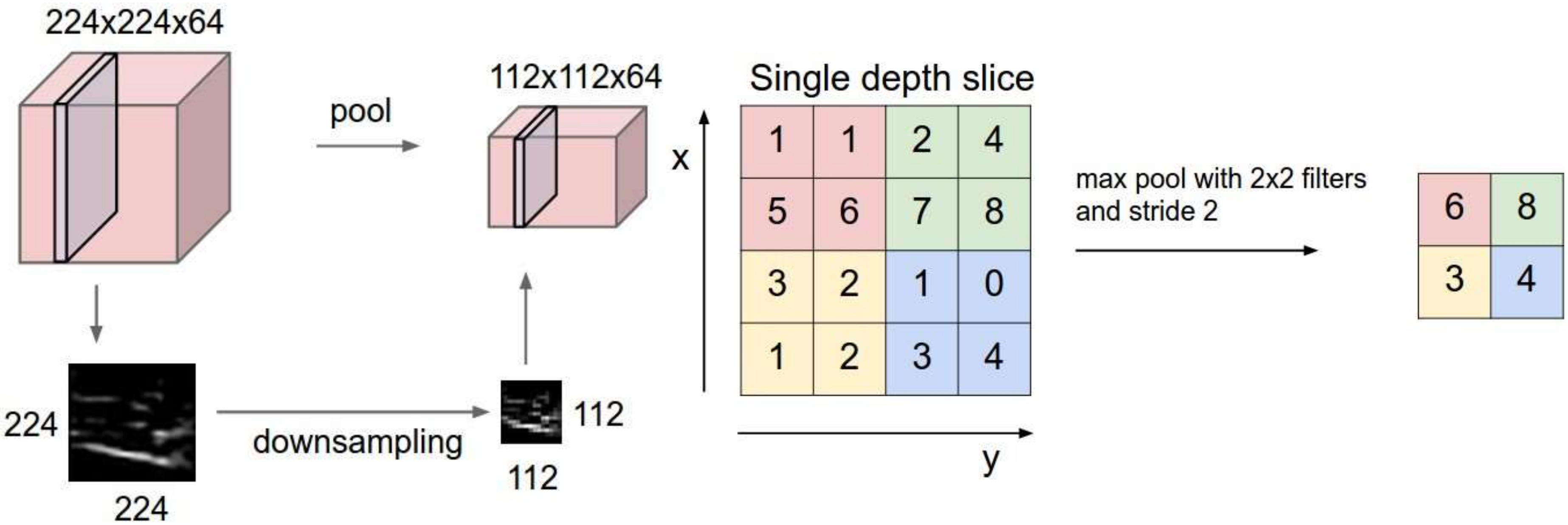
Инвариантность через MAX-pooling



Инвариантность можно обеспечить за счёт применения оператора MAX на выходах набора «простых» клеток, чувствительных к краю одной ориентации, но в разных точках одной области

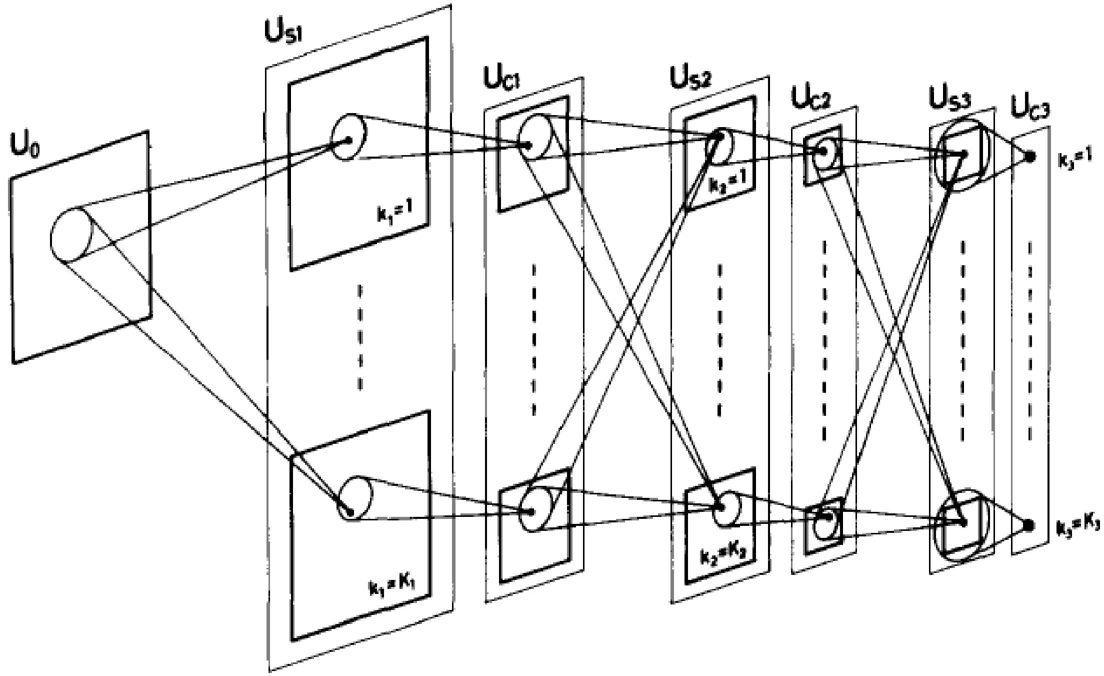
Riesenhuber, M. & Poggio, T. (1999). [Hierarchical Models of Object Recognition in Cortex](). *Nature Neuroscience* **2**: 1019-1025.

Пояснение работы



Source: http://cs231n.github.io/convolutional-networks/

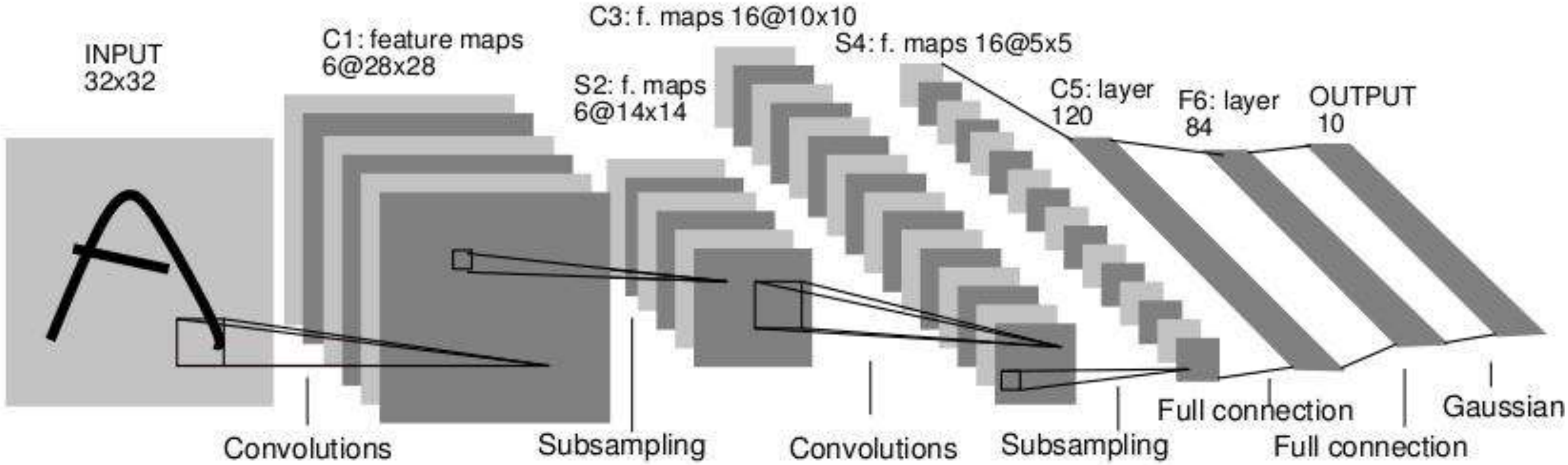
Neocognitron (1980)



- Многослойная нейросеть с чередующимимся S и C слоями
  - S-слои – линейные фильтры изображения («свёрточный слой»)
  - C-слои – MAX операторы, дающие инвариантность
- На верхнем уровне обеспечивается инвариантность по положению по всему изображению

K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics, 36(4): 93-202, 1980.

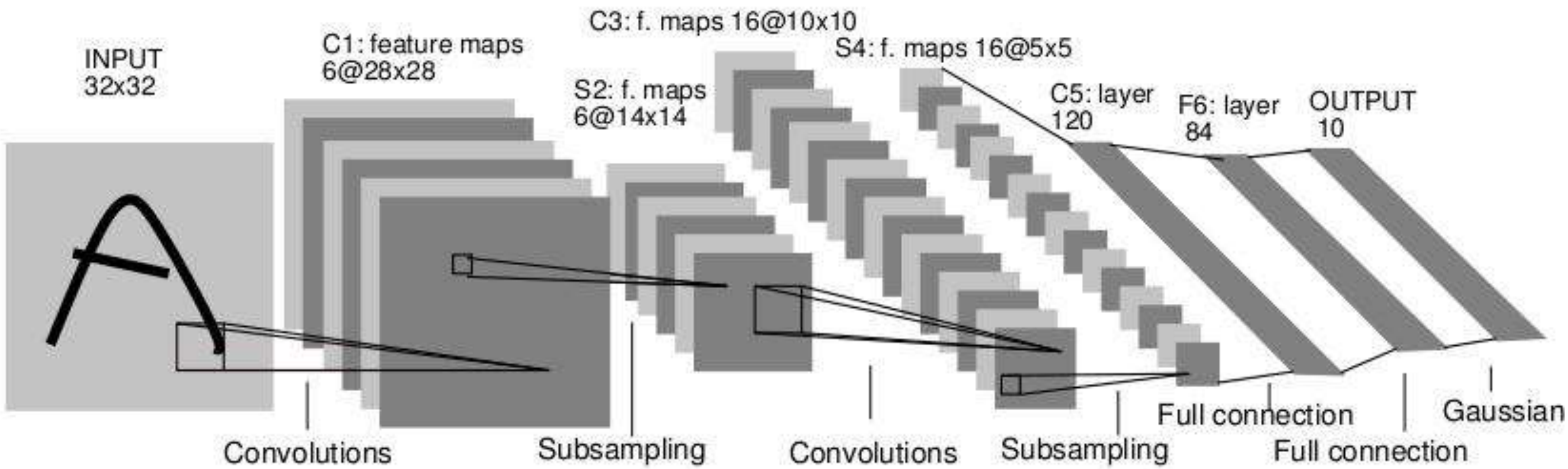
Свёрточные сети



- Неокогнитрон + обратное распространение ошибки = свёрточная сеть (Convolutional Neural Network, CNN)
- Поскольку для сверточного слоя нужно задать параметры только всех свёрток, что число параметров заметно меньше общего числа весов слоя
- Очень эффективная архитектура для распознавания изображений

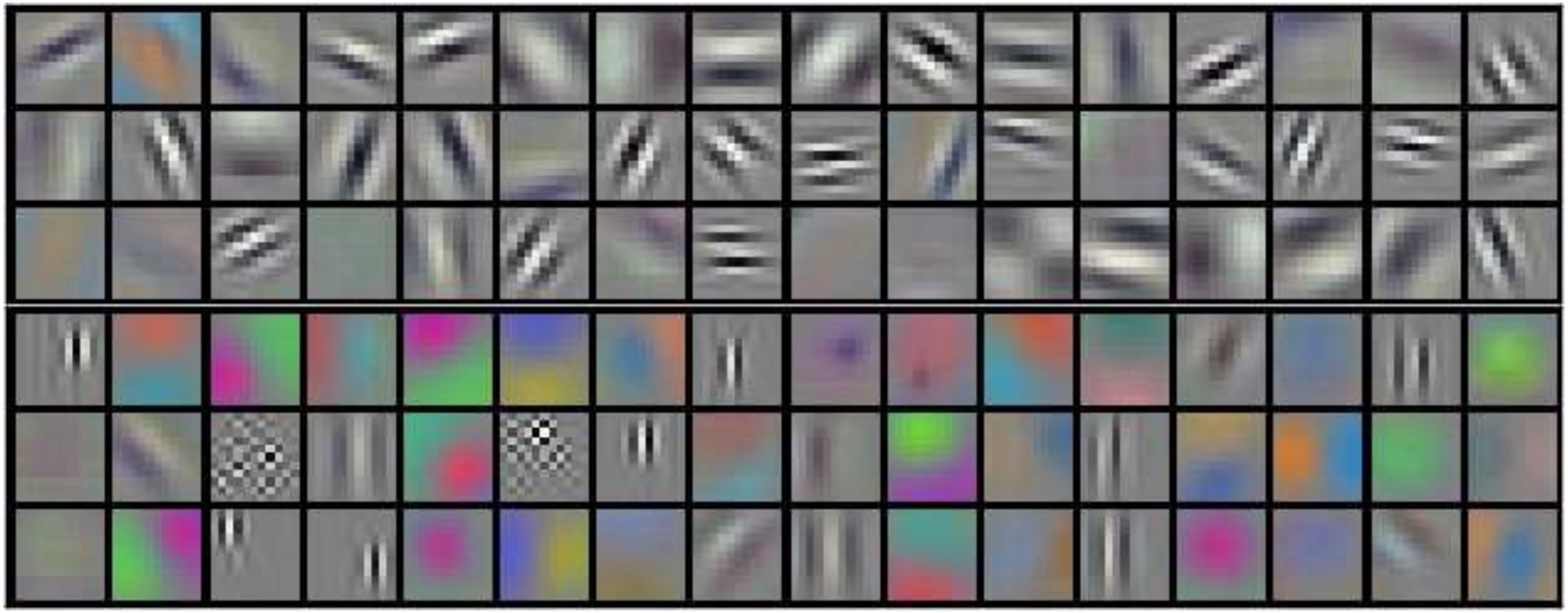
LeCun, Y., Bottou, L., Bengio, Y., and Haner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE

Размеры свёрток и число параметров



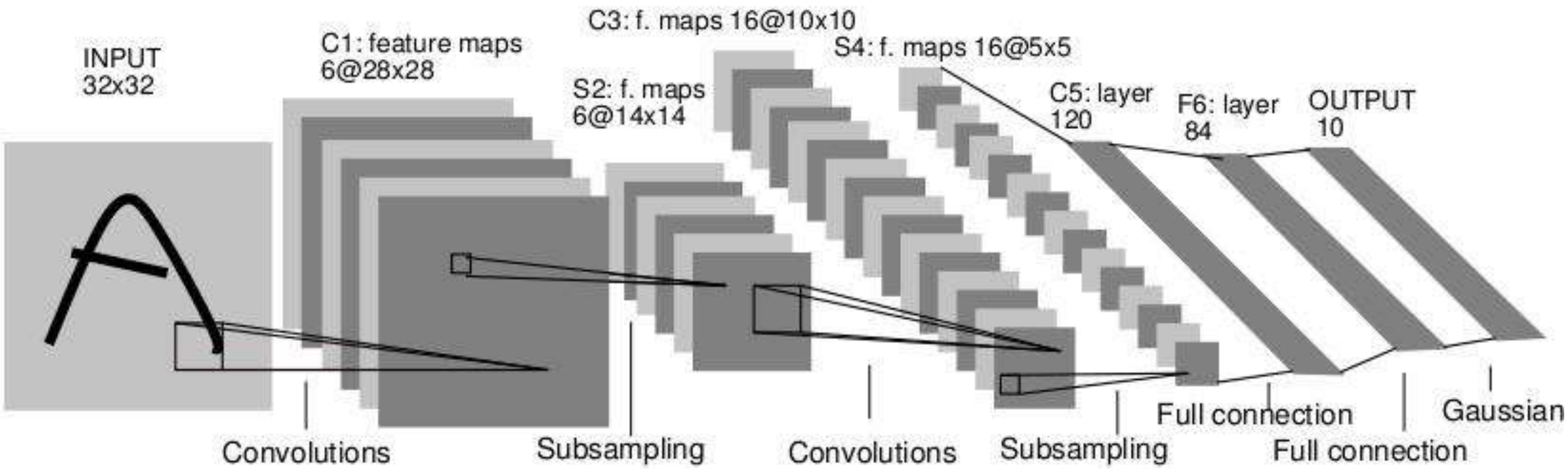
- Каковы размеры фильтров свёрток на разных слоях?
  - 5*5*1 на первом слое
  - 5*5*6 на втором свёрточном слое
  - «Глубина» тензора на выходе свёрточного слоя равна числу свёрток в свёрточном слое
  - 3е измерение свёртки равно «толщине» входного тензора
- Число весов и параметров второго слоя:
  - Параметров = 16 свёрток 5x5x6 = 150*16 = 2400
  - Весов = Параметров x 10 x 10 = 240000

Фильтры первого уровня

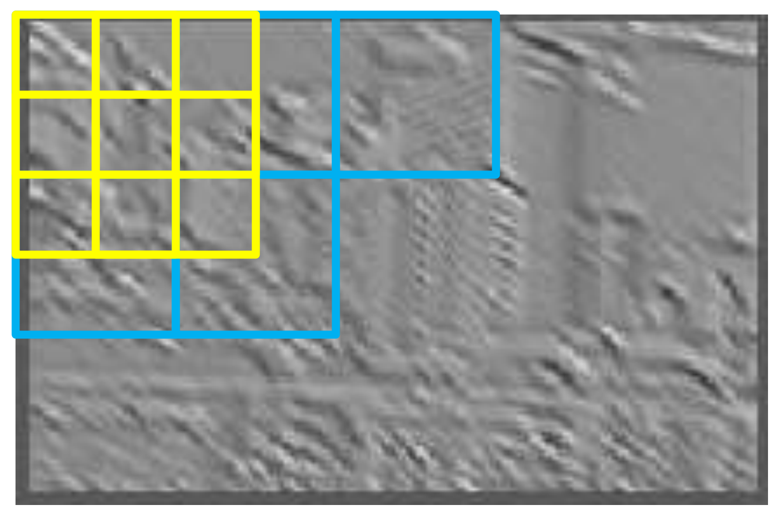


- Визуализируем веса фильтров
- Поскольку сворачиваем RGB изображение, то визуализация весов в RGB
- Обратите внимание на вычисление градиентов цветов

Subsampling/Pooling слои

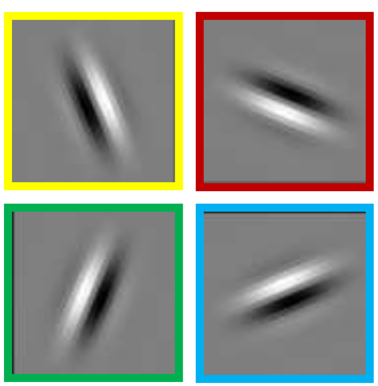


- Subsampling слой – уменьшение разрешения
- Обычно **Sum** или **Max** по областям с некоторым шагом
  - Sum-pooling
  - Max-pooling
- Области берутся с перекрытием или без перекрытия



Шаг свёртки

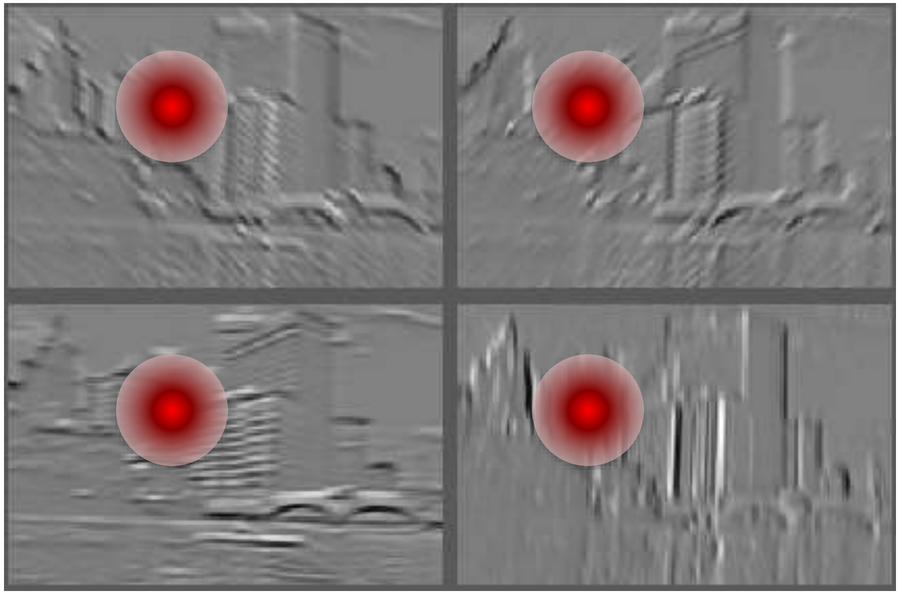
- Свёртка для каждого пикселя может быть слишком долго или избыточно
- Можем повторять фильтры со сдвигом на n пикселей
- Тензор на выхода имеет меньшую размерность по ширине и высоте при шаге >=2, по сравнению с обычной свёрткой
- Можем и не делать отдельный subsampling слой



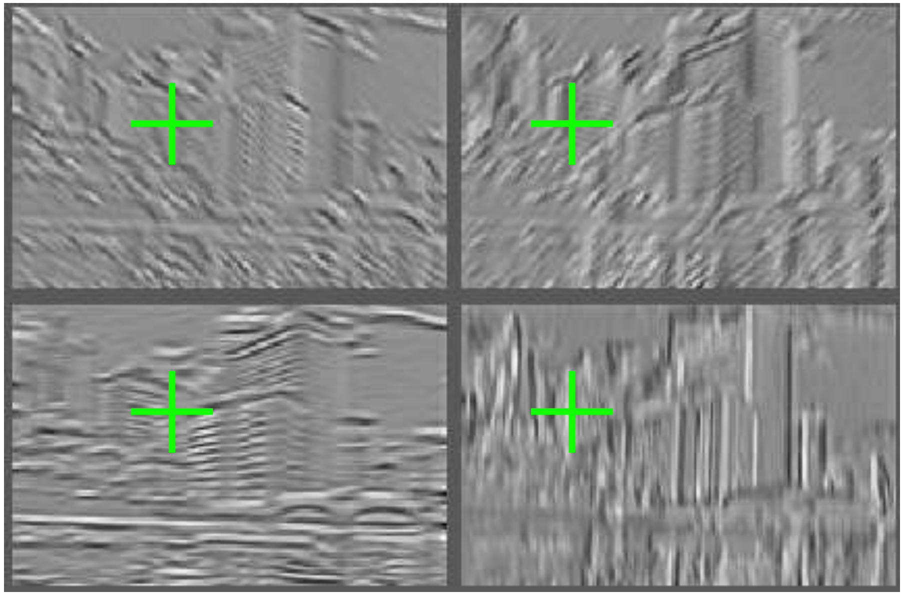
Нормализующие слои

Локальная нормализация:

- Local mean = 0, local std. = 1, “Local” → 7x7 Gaussian



Карты признаков



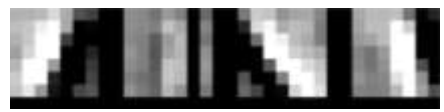
После нормализации контраста

Вспомним построение признаков

Признаки → Кодирование → Учёт по области

Пикселы / Признаки →

Фильтрация по словарю



\+ Нелинейность

Каждый этап – свой слой сети

Нормализация

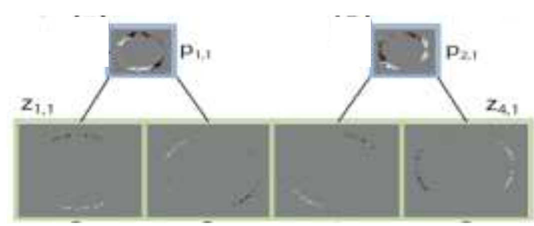
Группировка

Разреженность

Max / Softmax

Нормализация локального контраста

Учёт по области (Sum или Max)

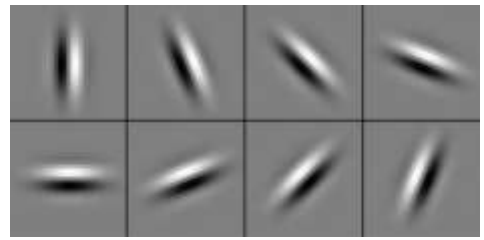


→ Признаки

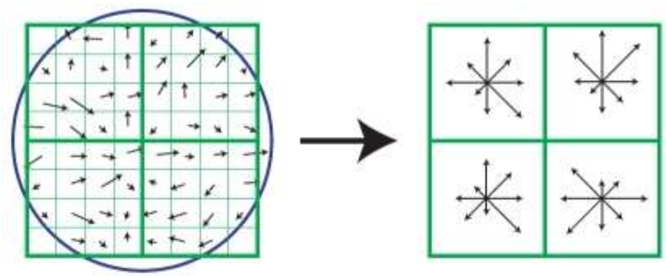
Дескриптор HOG/SIFT

Пикселы

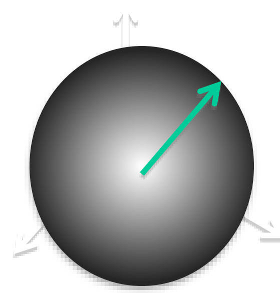
Фильтры Габора



Суммирование по ячейкам



Нормализация вектора до единичной длины

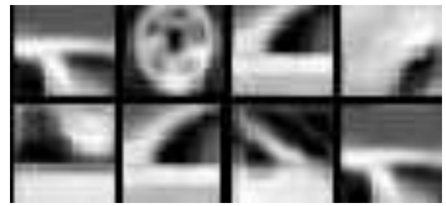


Вектор-признак

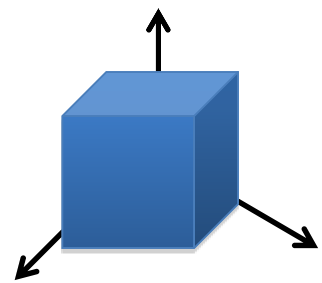
Мешок визуальных слов

HOG

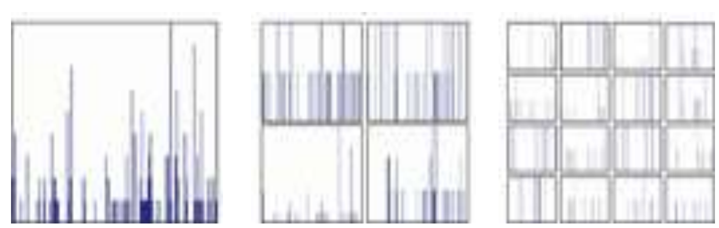
Свёртка с словарём



Max
(Находим самое близкое слово из словаря – квантуем)

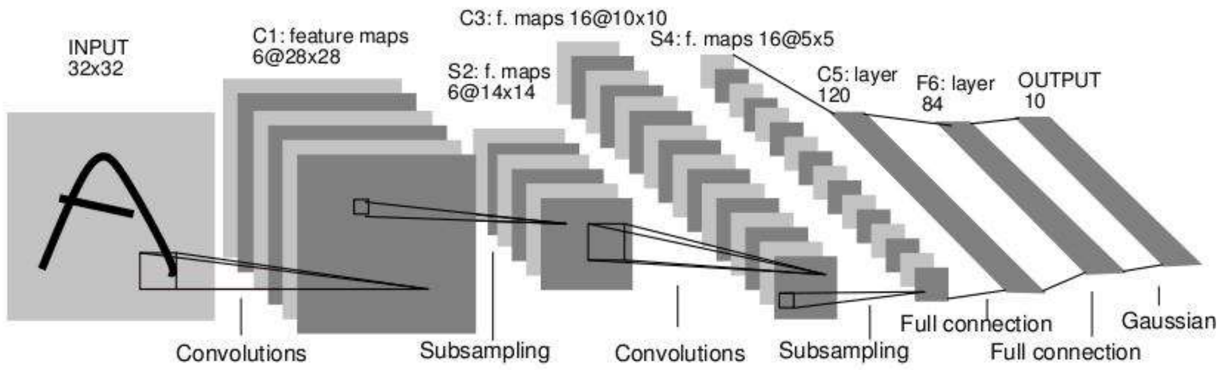


Суммирование по области
(Sum)



На классификатор

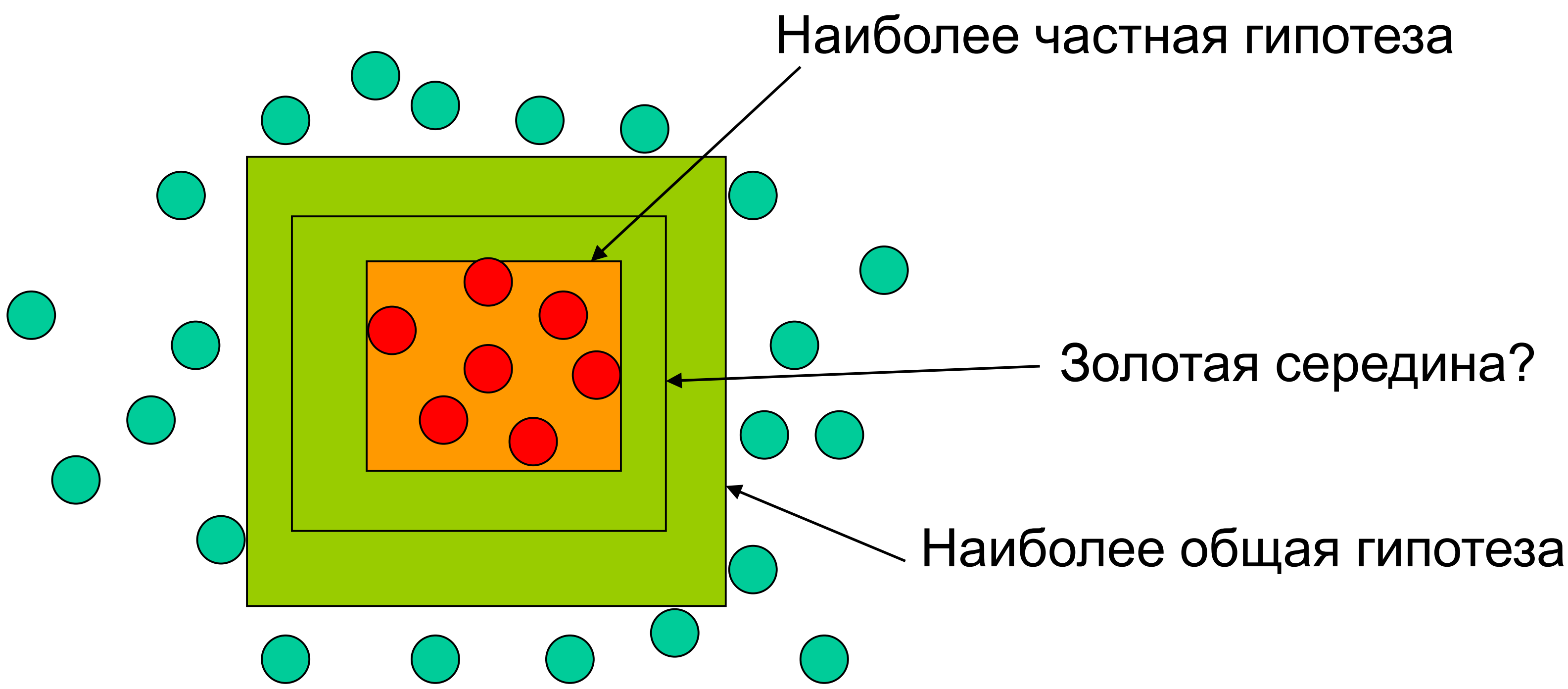
Важный вывод



- С помощью свёрточной нейросети из 2х свёрточных слоёв можно реализовать большинство эвристических методов вычисления признаков изображения (гистограммы цветов, HOG, мешки визуальных слов)
- Последующие слои реализуют какие-то признаки «более высокого уровня»
  - Какие именно – хороший вопрос, активно исследуется
- При обучении свёрточной сети эти признаки *обучаются* под решение поставленной задачи, а не задаются пользователем

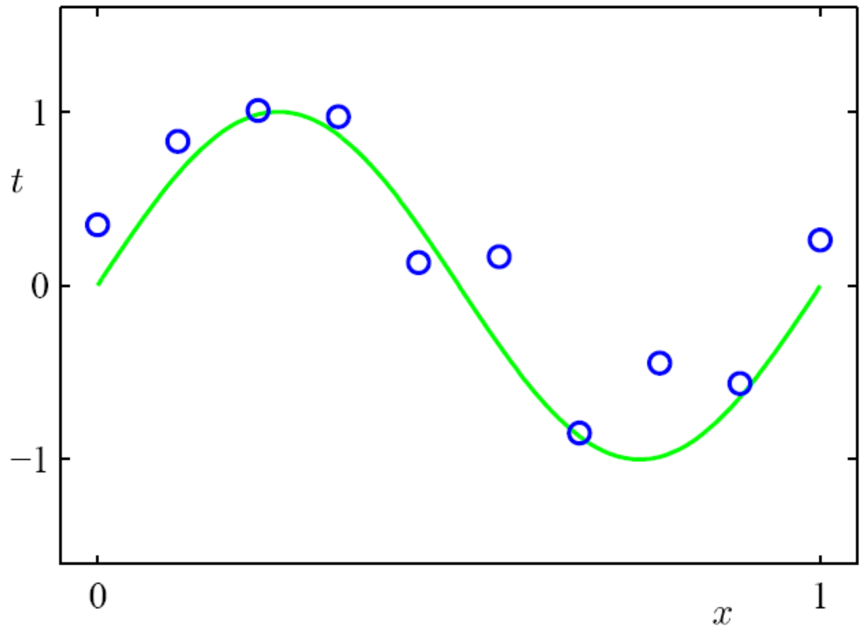
Явление переобучения

- Моделей, имеющих нулевой эмпирический риск (ошибку обучения) может существовать неограниченное количество:



Простой эксперимент

- Искусственный пример: задача регрессии
  - На самом деле $t = \sin(2\pi x) + \epsilon$, ϵ - нормально распределенный шум
  - Но мы этого не знаем
  - Есть обучающая выборка, требуется восстановить зависимость



Простой эксперимент

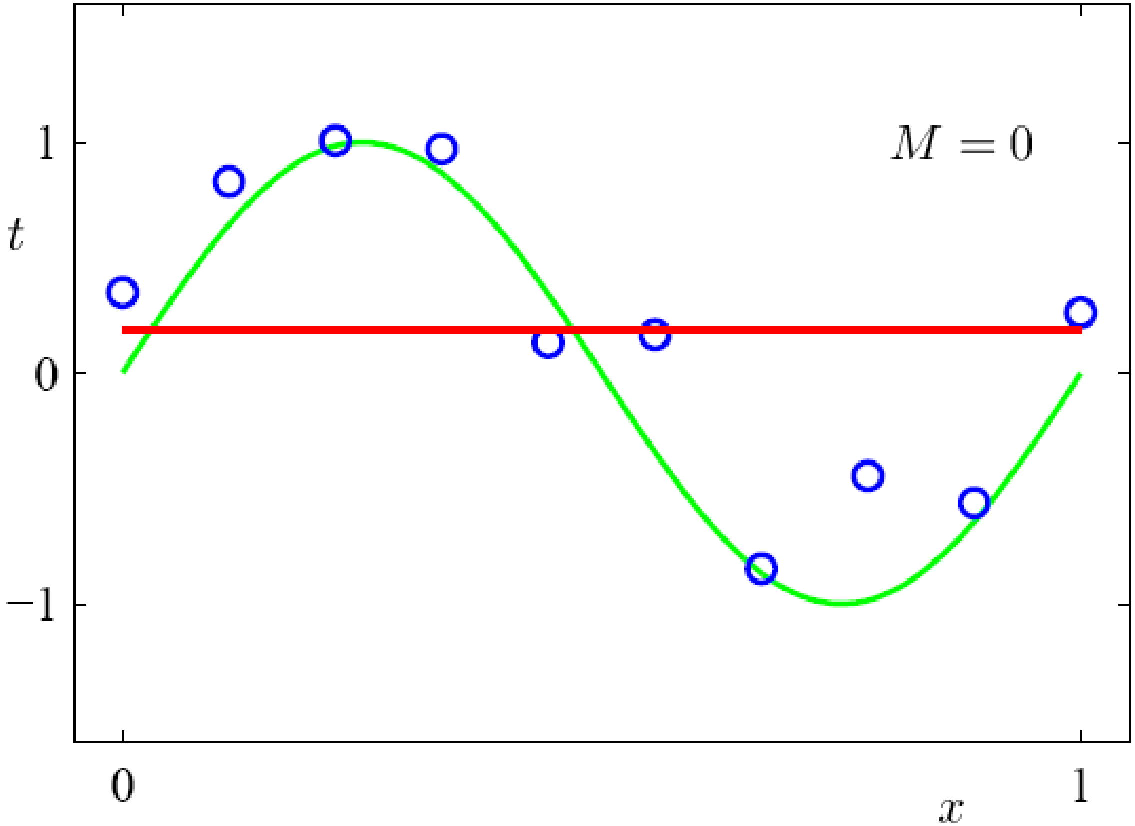
- Будем выбирать целевую зависимость среди параметризованного множества - полиномов порядка M

$$y(x, \mathbf{x}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \mathbf{w}^T \phi_M(x)$$

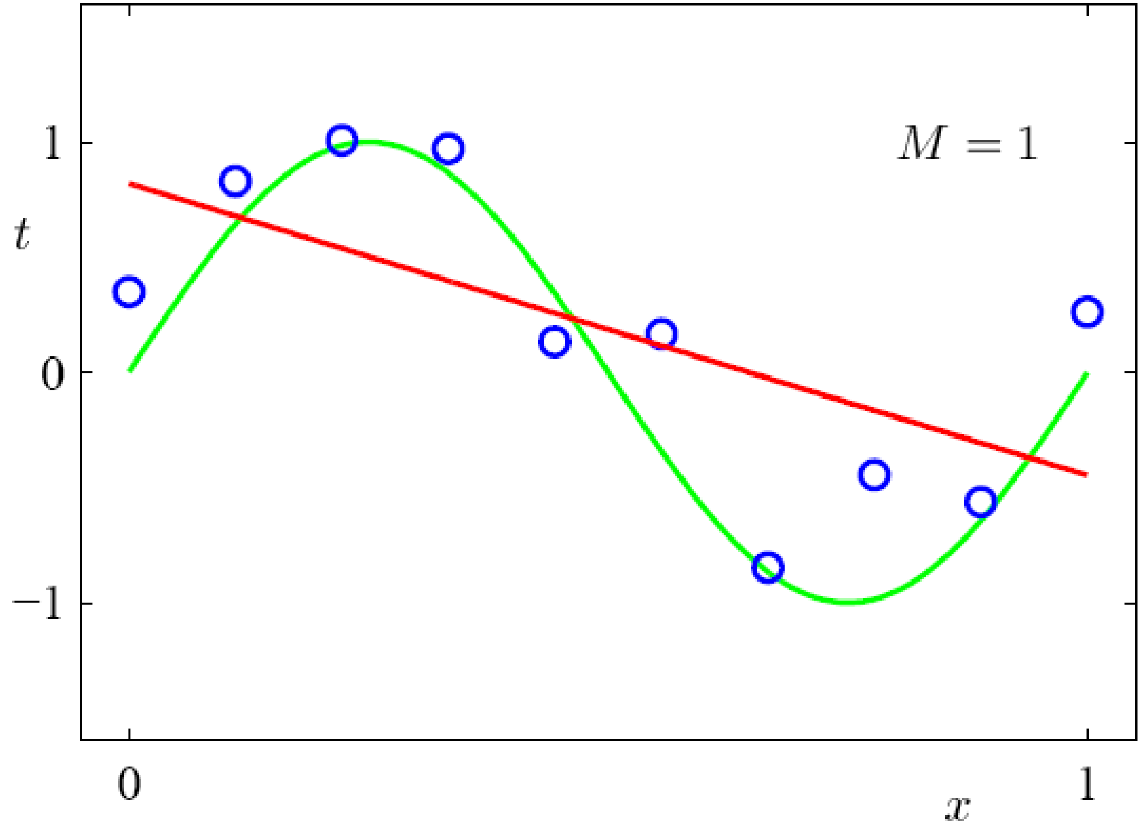
- Введем функция потерь $L((x,t),y) = \frac{1}{2}(y(x,\mathbf{w}) - t)^2$

- Среди множества полиномов будем выбирать тот, который приносит наименьшие суммарные потери на обучающей выборке (*минимальный эмпирический риск*)

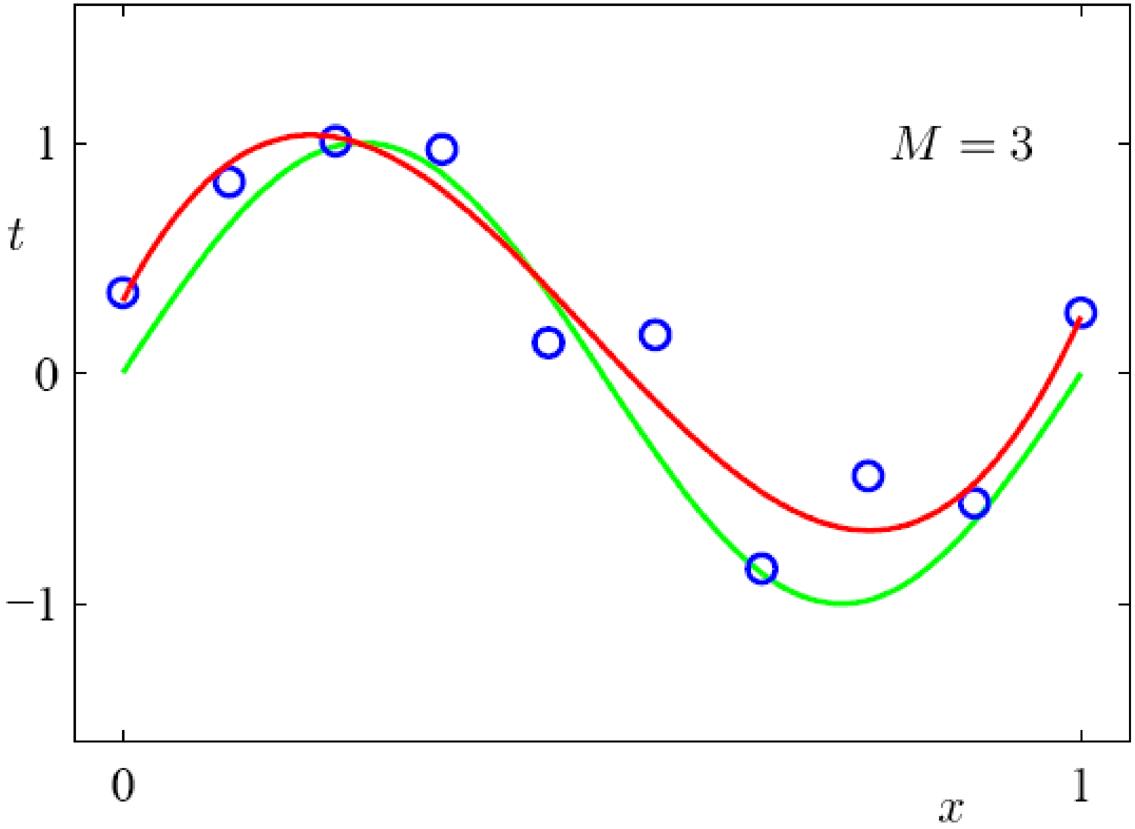
Простой эксперимент



Простой эксперимент

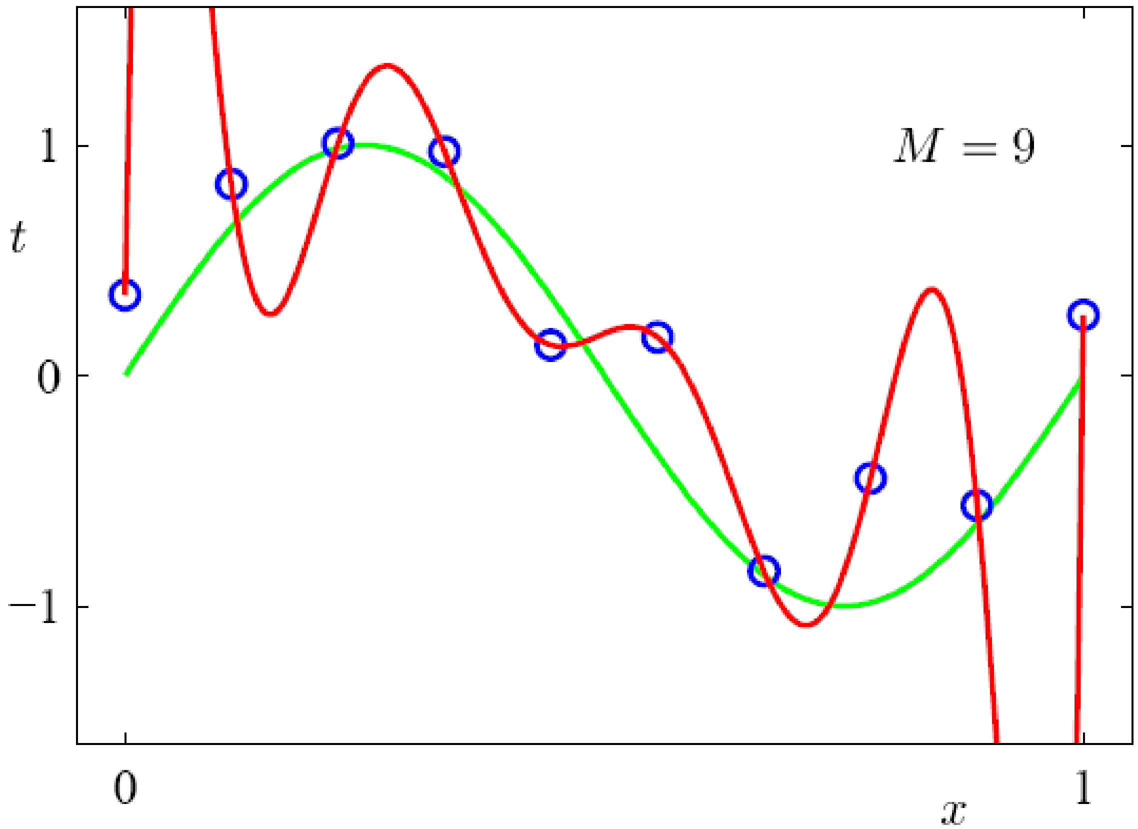


Простой эксперимент



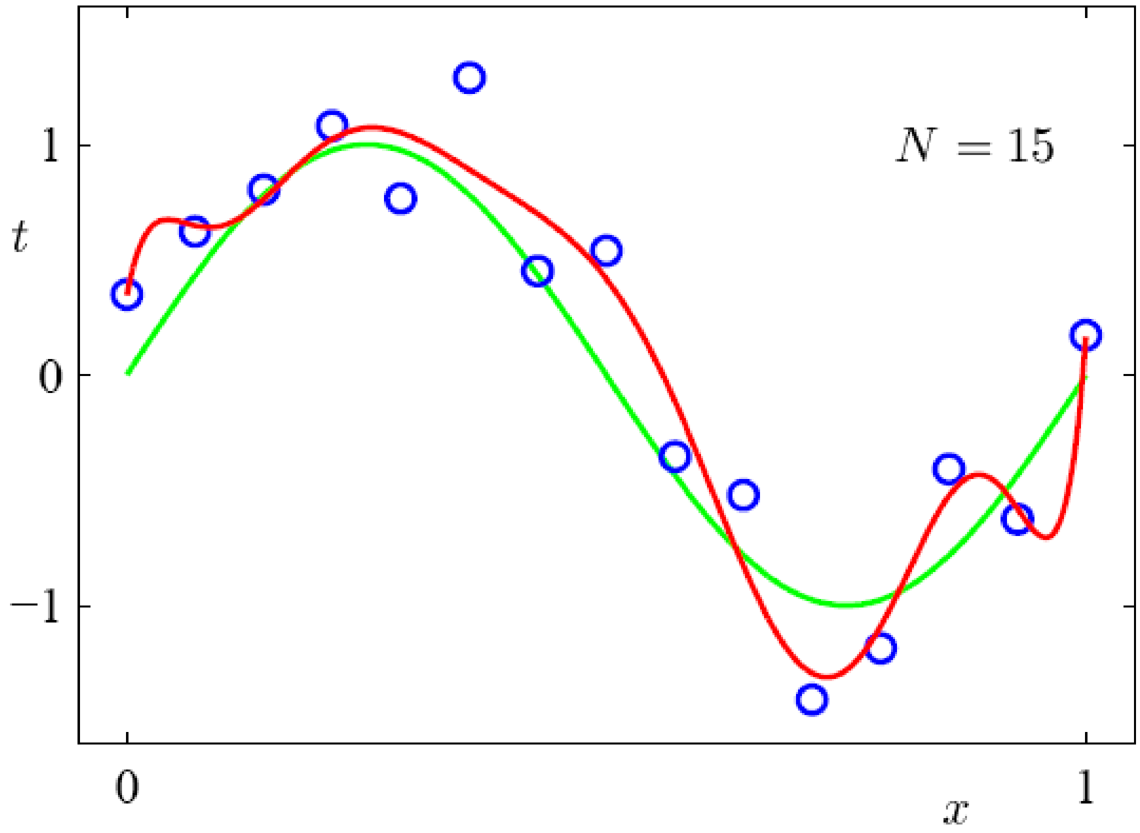
Что будет, если мы воспользуемся полиномом 9ой степени?

Простой эксперимент

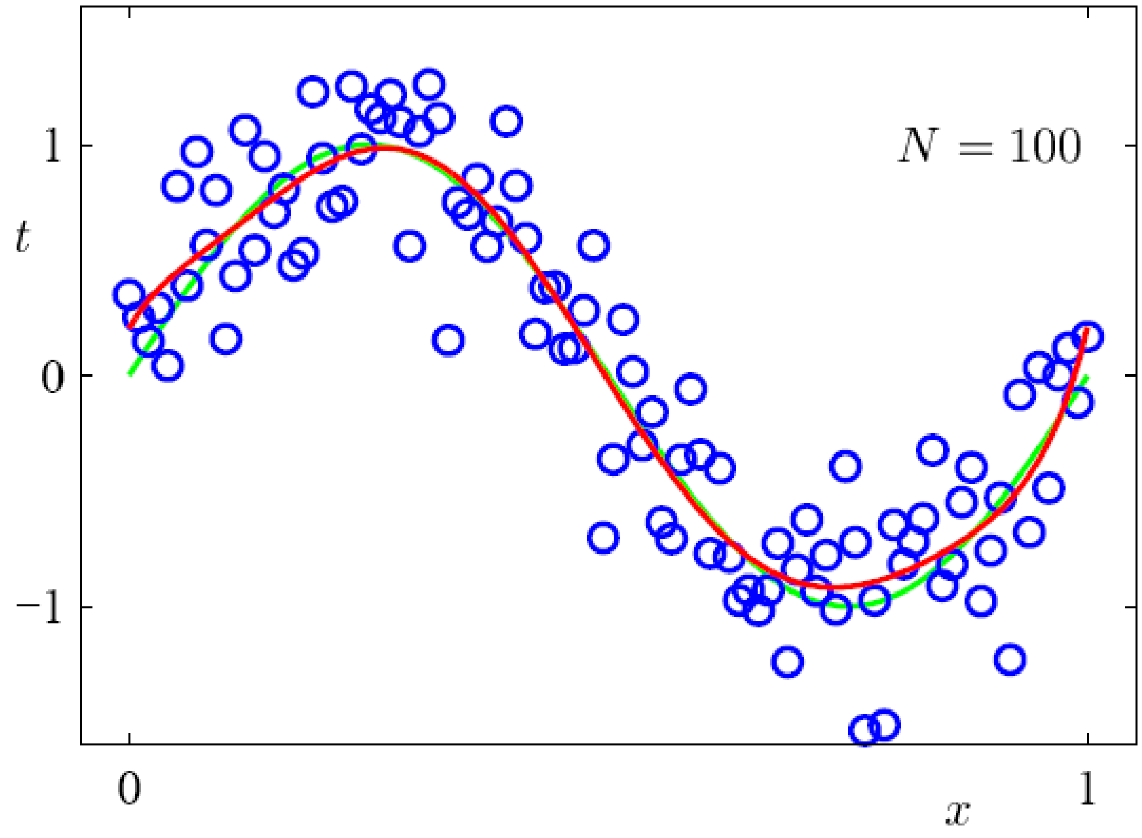


Какой эмпирический риск и какой общий риск?

Простой эксперимент



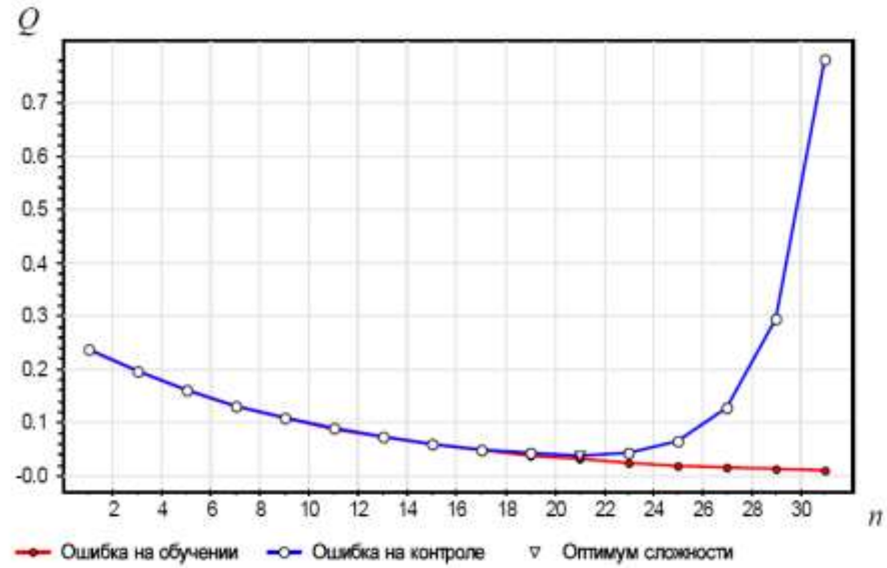
Простой эксперимент



Видим, что есть некоторая связь между сложностью параметрического семейства (количеством параметров) и размером обучающей выборки

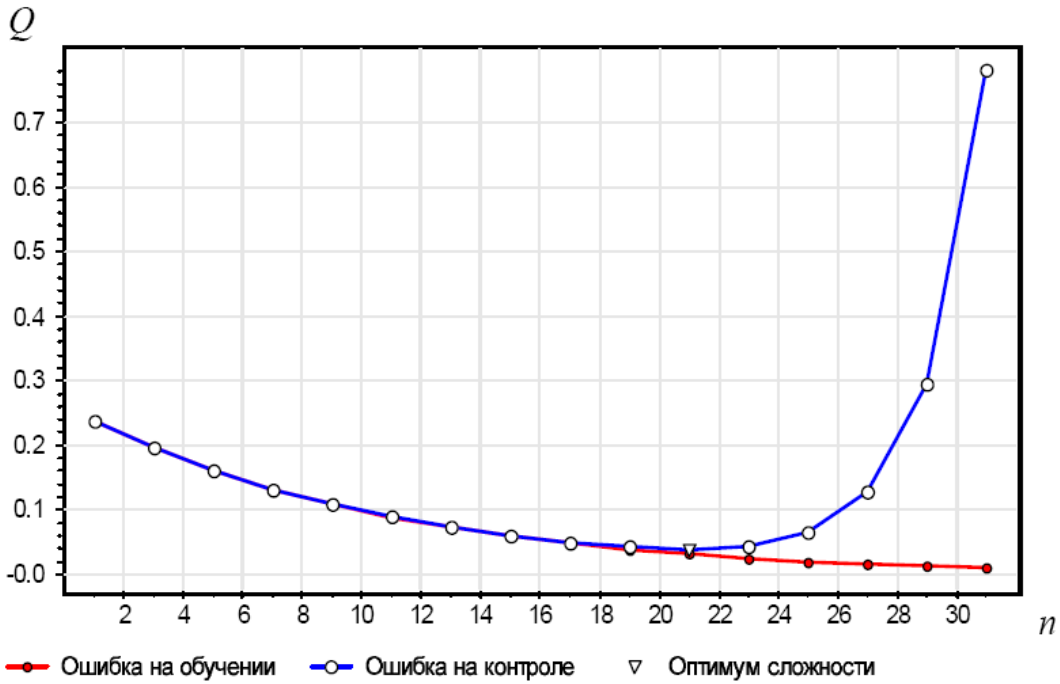
Явление переобучения

- При уменьшении ошибки обучения определенного момента может наблюдаться увеличение общего риска (уменьшение предсказательной способности)



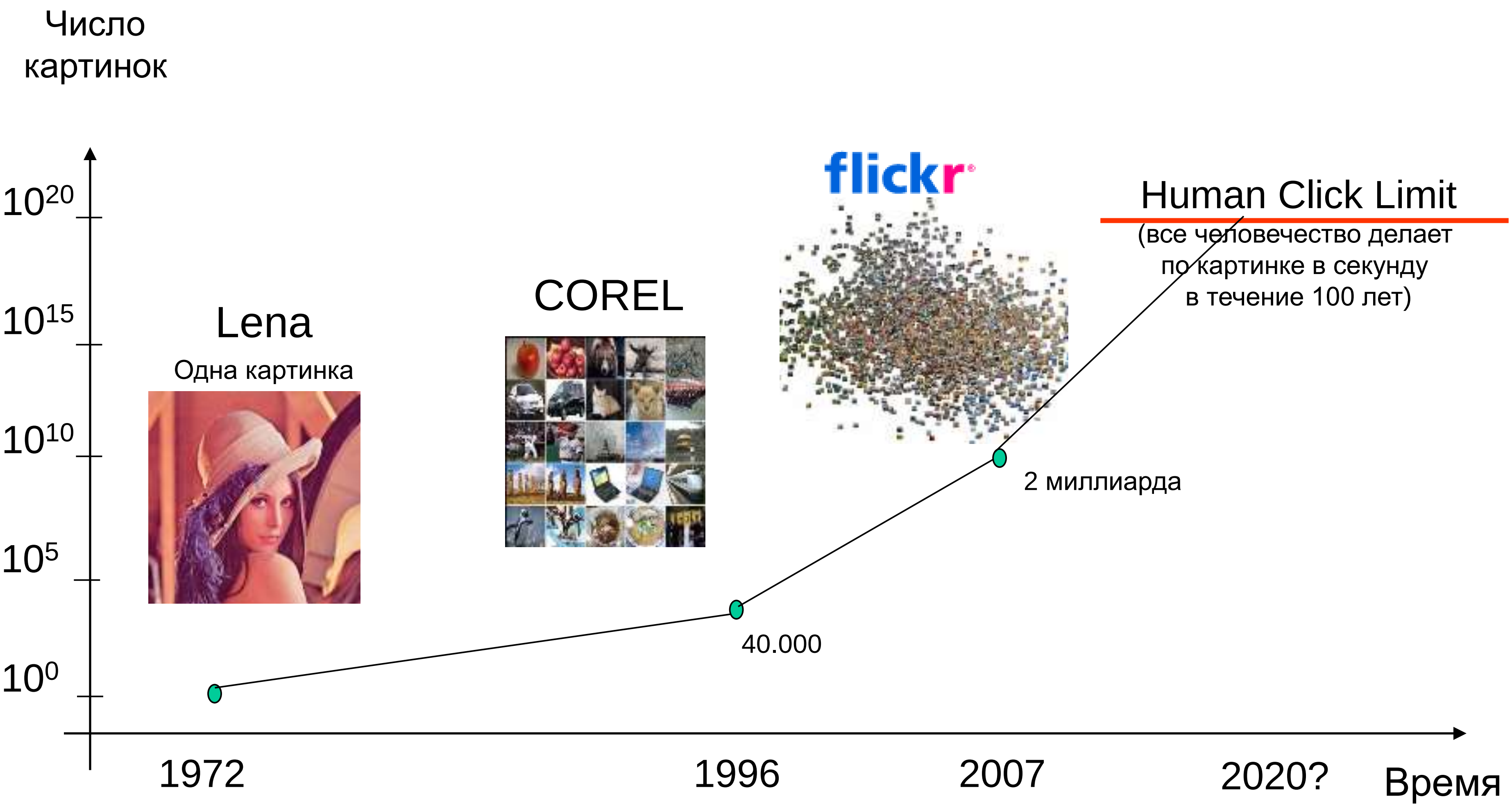
- Причина – гипотеза хорошо описывает свойства не объектов в целом, но только лишь объектов из обучающей выборки.
- Чаще всего это означает, что модель слишком сложная, и она «запоминает» обучающую выборку

Переобучение в нейросетях



- Чем сложнее задача – тем более сложная нейросеть нужна
- Но параметрой нейросети очень много, и нейросеть легко «переобучается»

«Интернет-бум» + «Закон Мура»



Slide by Antonio Torralba

Нейросети – победители LSVR 2012

IMAGENET

Large-scale visual recognition (LSVR)

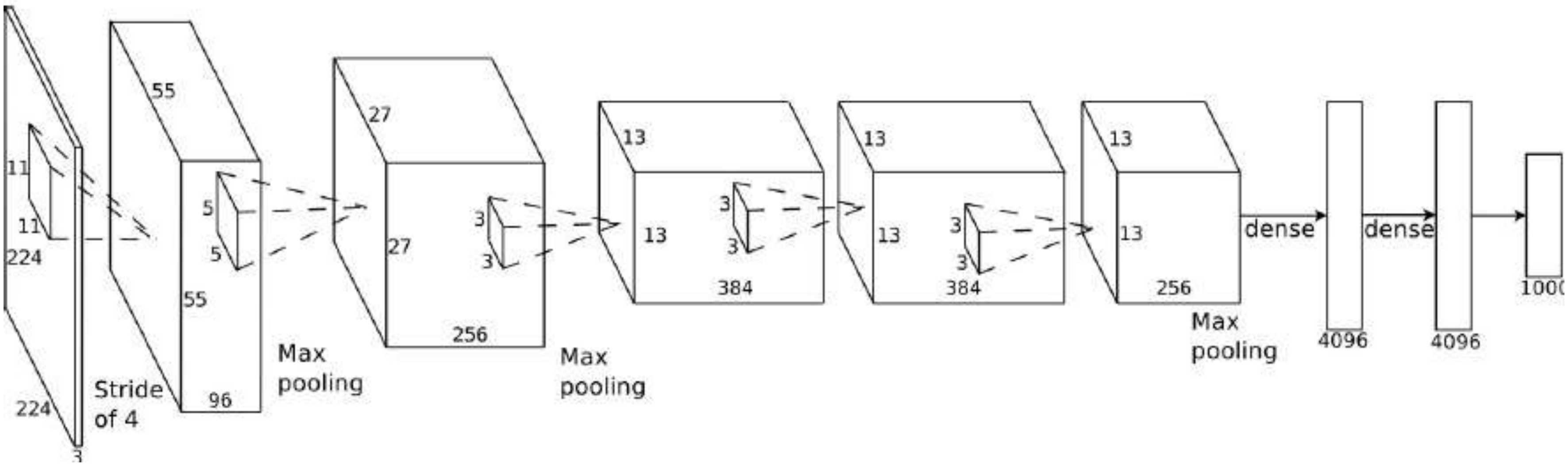
Task 1: Classification

- Predict a class label

- 5 predictions / image
- 1000 classes
- 1,200 images per class for training
- Bounding boxes for 50% of training.

Winner
SuperVision
Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton
University of Toronto

SuperVision

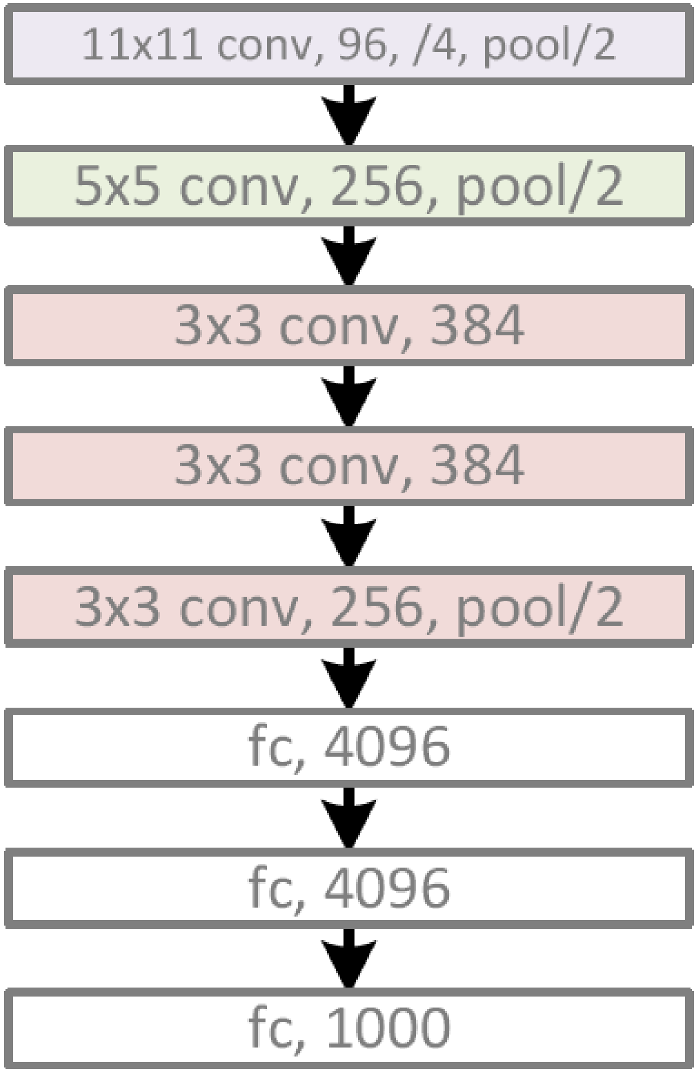


- 650,000 neurons
- 60,000,000 parameters
- 630,000,000 connections
- 1 машина, 2 GPU по 2Gb, 5GB Ram, 27Gb HDD, 1 неделя на обучение

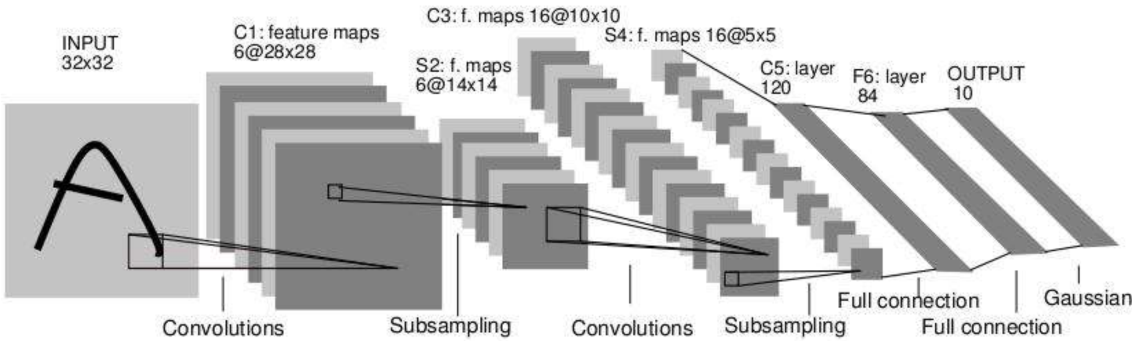
Krizhevsky A., Sutskever I., Hinton, G. E. ImageNet Classification with Deep Convolutional Neural Networks // NIPS 2012

AlexNet

AlexNet, 8 layers
(ILSVRC 2012)

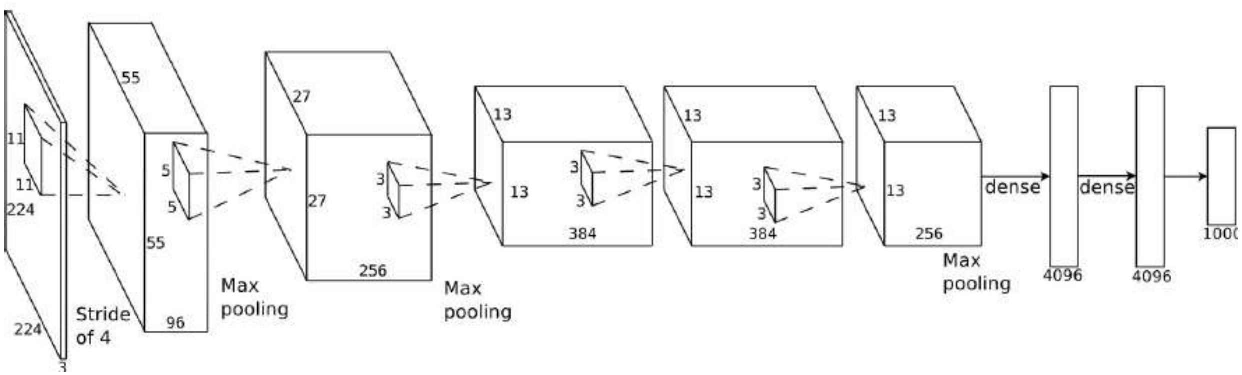


CNN раньше и сейчас



1998 год

- 2 свёрточных слоя (6 и 6 фильтров)
- 2 полносвязанных (120 и 84 нейрона)



2012 год

- 5 свёрточных слоёв (96, 256, 384, 384, 256 фильтров)
- 2 полносвязанных (4096 и 4096 нейрона)

- Больше слоёв, фильтров, нейронов
- За счёт большого объёма данных, вычислительной мощности и некоторых улучшений (ReLU и т.д.) смогли обучить такую большую сеть

Krizhevsky A., Sutskever I., Hinton, G. E. (2012) ImageNet Classification with Deep Convolutional Neural Networks // NIPS 2012: Neural Information Processing Systems. Lake Tahoe, Nevada.

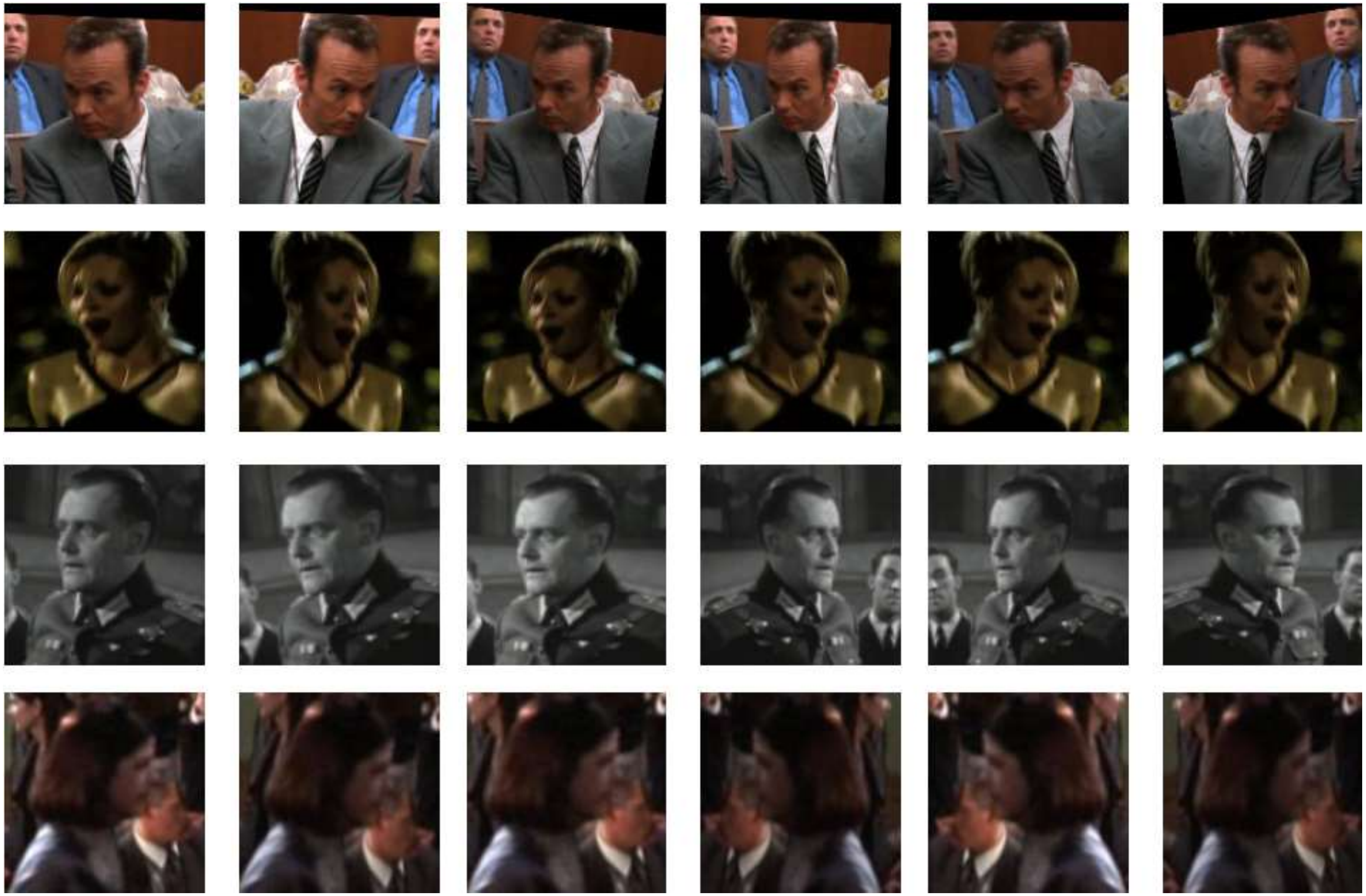
Примеры работы

Заметки про обучение

- Размножение данных (Data augmentation)

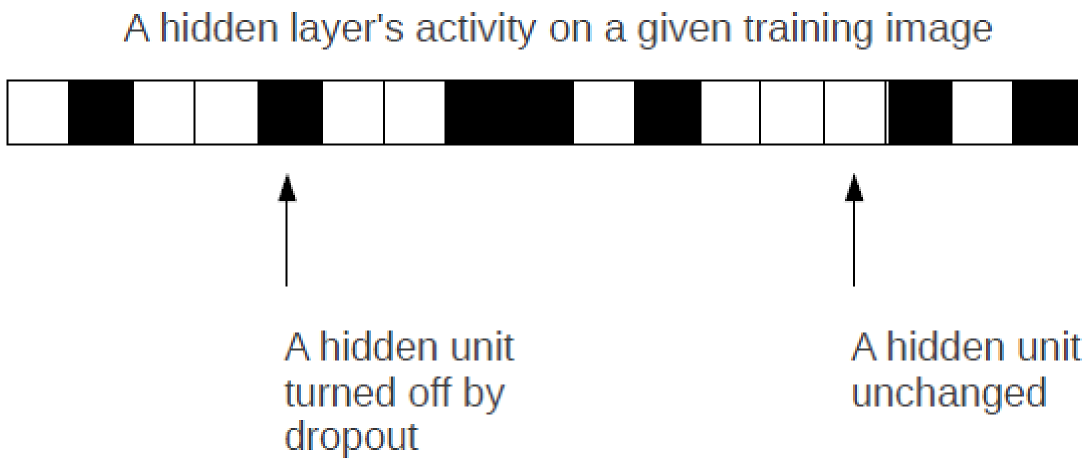
- Борьба с переобучением
- Из 256x256 случайно выбираем фрагменты 224x224 и их отражения
- Добавляем цветовые искажения

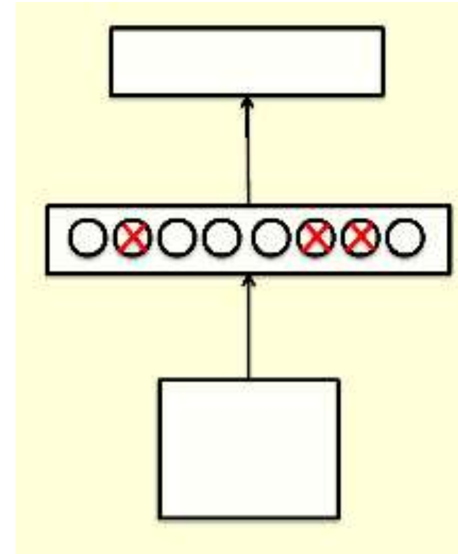
Варианты размножения данных



- Небольшие сдвиги, отображения, повороты, изменения масштаба
- Лучше использовать библиотеку!

Dropout

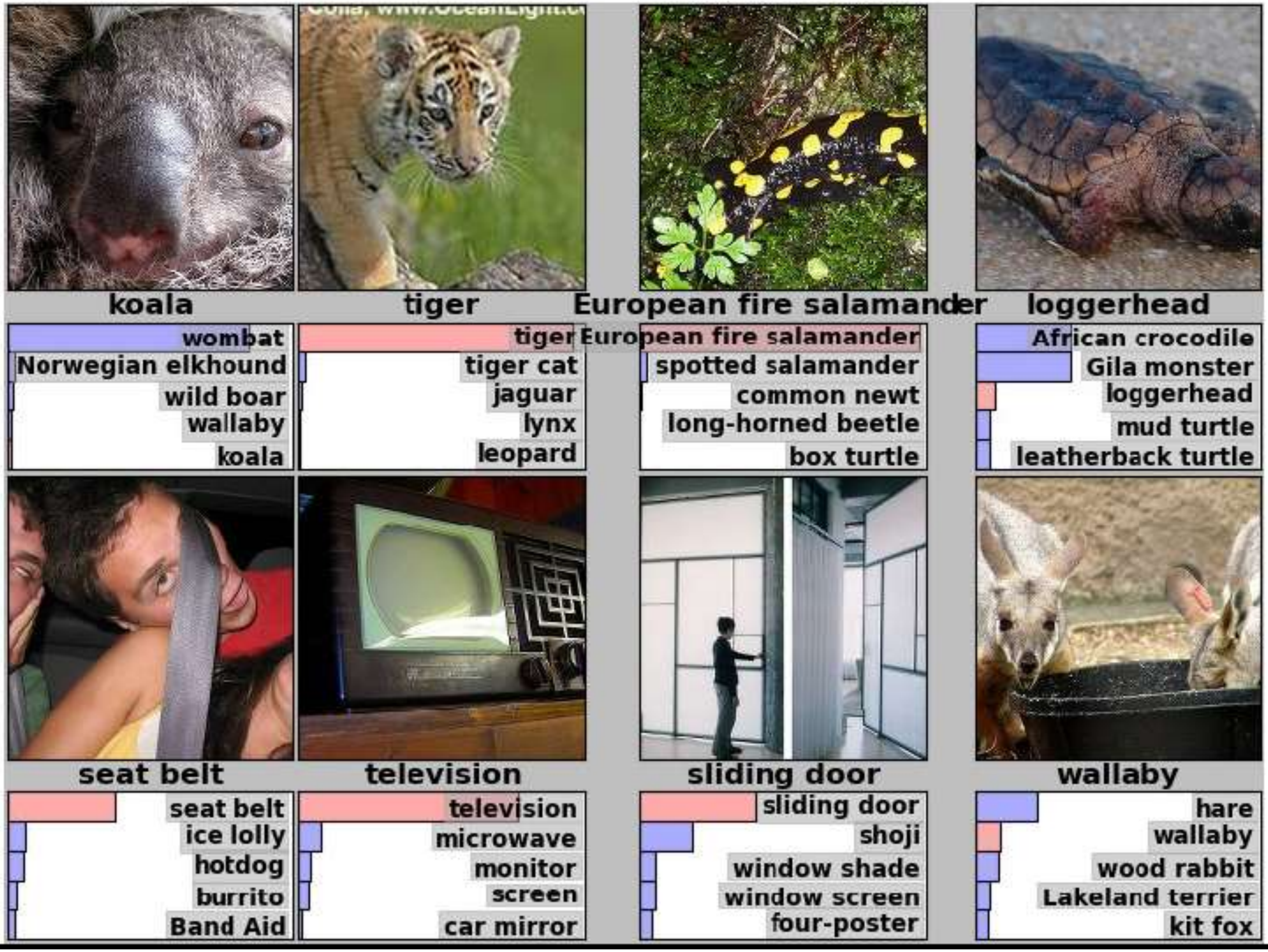




- Отключаем половину нейронов в каждом слое
- Получаем случайную выборку из множества сетей
- Во время тестирования используем «среднюю» сеть с уполовиненными весами

Nitish Srivastava Improving Neural Networks with Dropout. Master Thesis, 2013

Примеры работы



Резюме

- Концептуально нейросети остались такими же, как в 1990х, но было предложено множество относительно небольших изменений, которые в совокупности с ростом доступных данных позволили сети эффективно обучать
- Есть целый ряд библиотек и уже обученных моделей
- Возможность взять обученную модель и настроить её на другие данные позволяет расширять круг решаемых задач на те, где данных не так много
- Нейросетевые модели стали применять очень широко