Школа Анализа Данных
Яндекса

Курс «Анализ изображений и видео»

Лекция №5
«Свёрточные нейросети»

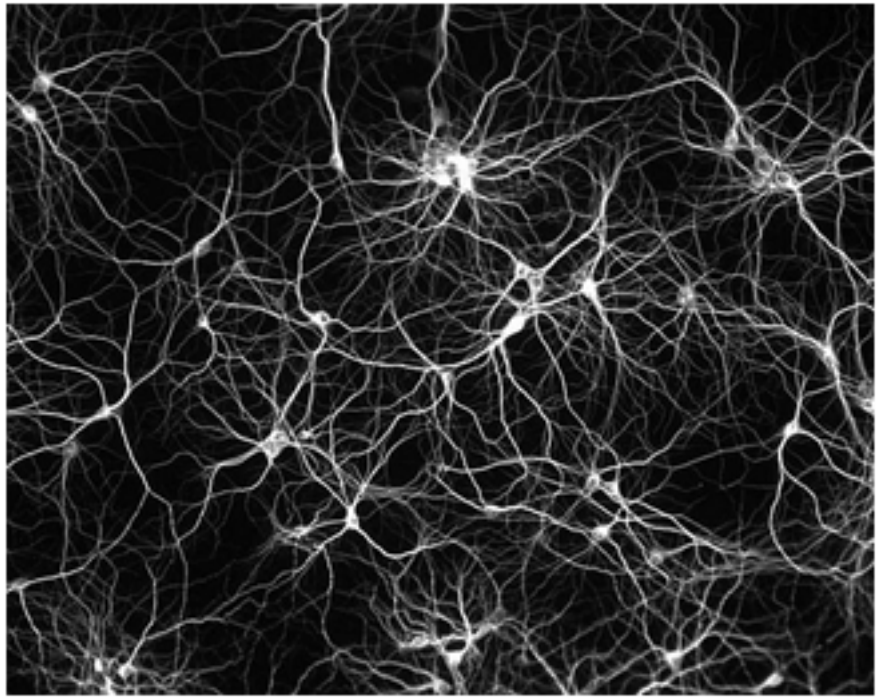
Антон Конушин
Заведующий лабораторией компьютерной графики и мультимедиа
ВМК МГУ

7 октября 2016 года

Благодарность

В лекции использованы материалы курса «Машинное обучение» К.В. Воронцова

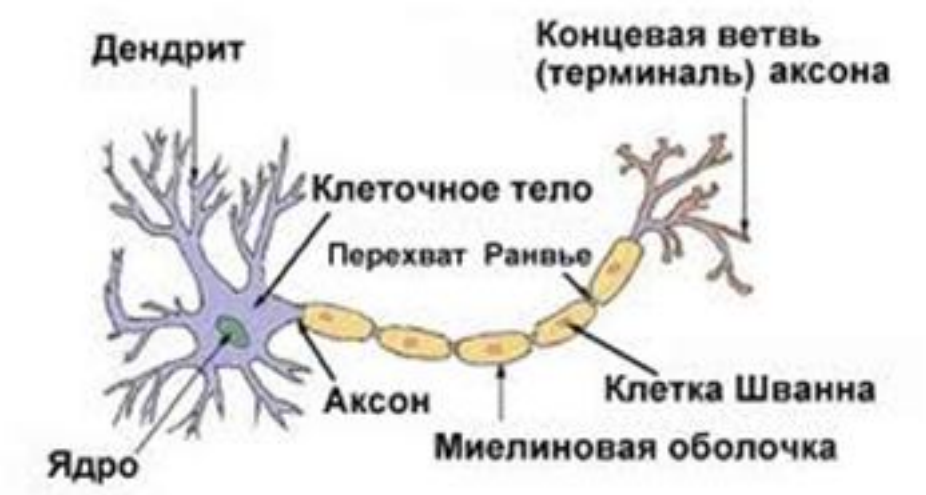
Структура мозга

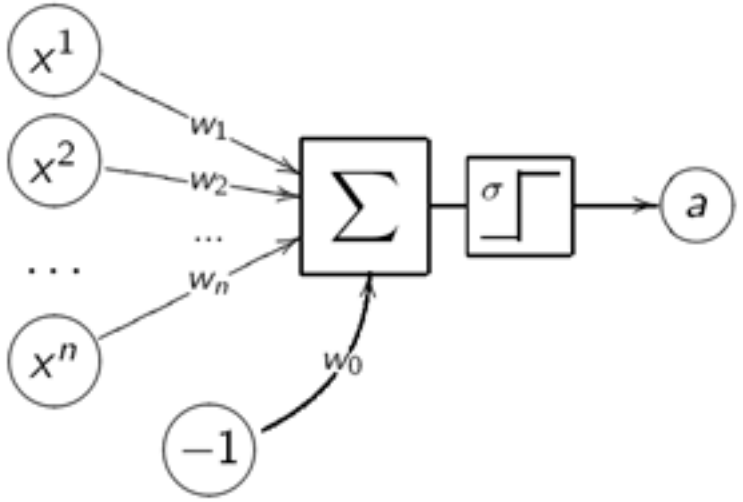


Нейросеть

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 $f_0(x) \equiv -1$

Градиентный метод обучения

Минимизация аппроксимированного эмпирического риска:

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

Численная минимизация методом *градиентного спуска*:

$$w^{(0)} := \text{начальное приближение;}$$

$$w^{(t+1)} := w^{(t)} - \eta \cdot \nabla Q(w^{(t)}), \qquad \nabla Q(w) = \left(\frac{\partial Q(w)}{\partial w_j}\right)_{j=0}^{n},$$

где η — *градиентный шаг*, называемый также *темпом обучения*.

$$w^{(t+1)} := w^{(t)} - \eta \sum_{i=1}^{\ell} \mathscr{L}'\big(\langle w^{(t)}, x_i\rangle y_i\big)\, x_i\, y_i.$$

Идея ускорения сходимости:
брать (x_i, y_i) по одному и сразу обновлять вектор весов.

SG: Стохастический градиентный спуск

Вход: выборка X^ℓ, темп обучения h, темп забывания λ;
Выход: вектор весов w;

1 инициализировать веса w_j, $j = 0, \ldots, n$;
2 инициализировать оценку функционала: $\bar{Q} := \frac{1}{\ell} \sum_{i=1}^{\ell} \mathscr{L}_i(w)$;
3 **повторять**
4 | выбрать объект x_i из X^ℓ случайным образом;
5 | вычислить потерю: $\varepsilon_i := \mathscr{L}_i(w)$;
6 | сделать градиентный шаг: $w := w - h\nabla \mathscr{L}_i(w)$;
7 | оценить функционал: $\bar{Q} := (1 - \lambda)\bar{Q} + \lambda\varepsilon_i$;
8 **пока** значение $\bar{Q}$ и/или веса w не сойдутся;

Robbins, H., Monro S. A stochastic approximation method // Annals of Mathematical Statistics, 1951, 22 (3), p. 400–407.

Спуск с правилом Хэбба

Задача классификации: $x_i \in \mathbb{R}^{n+1}$, $y_i \in \{-1, +1\}$,

$$a(x, w) = \text{sign}\langle w, x\rangle, \qquad \mathscr{L}_i(w) = \big(-\langle w, x_i\rangle y_i\big)_+.$$

Градиентный шаг SG — правило Хэбба [1949]:

$$\textbf{если } \langle w, x_i\rangle y_i < 0 \textbf{ то } w := w + h x_i y_i,$$

То же самое для случая $y_i \in \{0, 1\}$,

$$a(x, w) = \big[\langle w, x\rangle > 0\big], \qquad \mathscr{L}_i(w) = \big(a(x_i, w) - y_i\big)\langle w, x_i\rangle,$$

Градиентный шаг SG — персептрон Розенблатта [1957]:

$$w := w - h\big(a(x_i, w) - y_i\big)x_i.$$

Обоснование

Задача классификации: $X = \mathbb{R}^{n+1}$, $Y = \{-1, 1\}$.

Теорема (Новиков, 1962)

Пусть выборка X^ℓ линейно разделима:

$$\exists \tilde{w},\ \exists \delta > 0\colon \langle \tilde{w}, x_i \rangle y_i > \delta \ \text{ для всех } \ i = 1, \ldots, \ell.$$

Тогда Алгоритм SG с правилом Хэбба находит вектор весов w,

- разделяющий обучающую выборку без ошибок;
- при любом начальном положении $w^{(0)}$;
- при любом темпе обучения $\eta > 0$;
- независимо от порядка предъявления объектов x_i;
- за конечное число исправлений вектора w;
- если $w^{(0)} = 0$, то число исправлений $t_{\max} \leqslant \frac{1}{\delta^2} \max \|x_i\|$.

Плюсы и минусы

Достоинства:

1. легко реализуется;
2. легко обобщается на любые $g(x, w)$, $\mathscr{L}(a, y)$;
3. возможно динамическое (потоковое) обучение;
4. на сверхбольших выборках можно получить неплохое решение, даже не обработав все (x_i, y_i);
5. подходит для задач с большими данными

Недостатки:

1. возможна расходимость или медленная сходимость;
2. застревание в локальных минимумах;
3. подбор комплекса эвристик является искусством;
4. проблема переобучения;

Масштаб задач

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

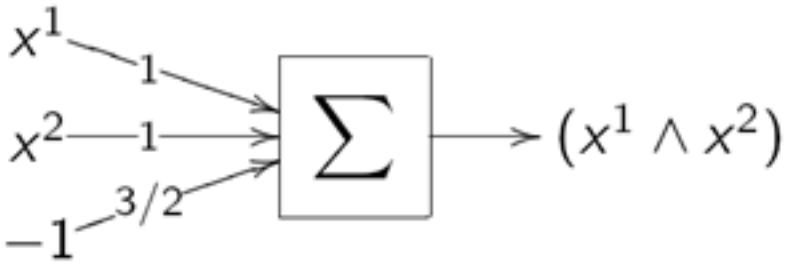
$$Q(w; X^\ell) = \sum_{i=1}^{\ell} \big(\sigma(\langle w, x_i\rangle) - y_i\big)^2 \to \min_w;$$

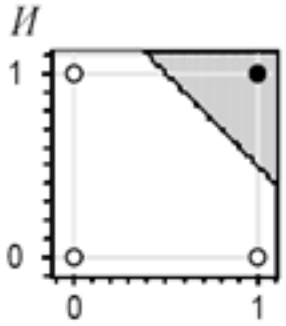
Насколько богатый класс функций реализуется нейроном?
А сетью (суперпозицией) нейронов?

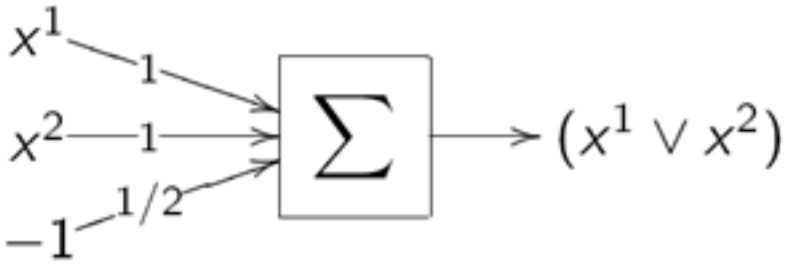
Логические элементы

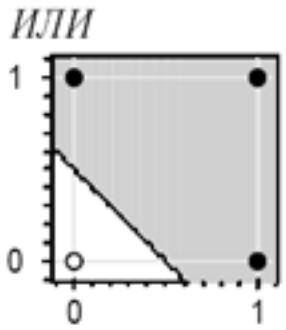
Функции И, ИЛИ, НЕ от бинарных переменных x^1 и x^2:

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







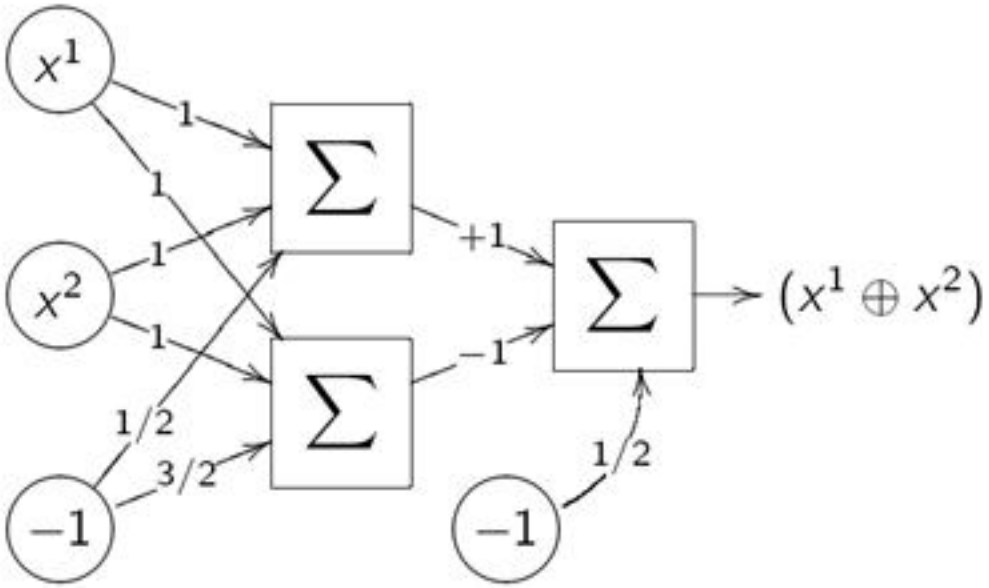


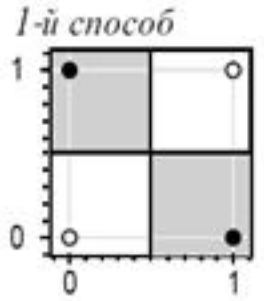
Исключающее ИЛИ (XOR)

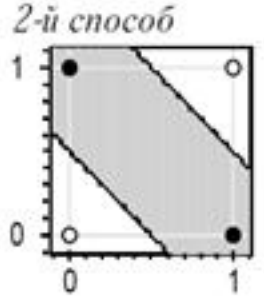
Функция $x^1 \oplus x^2 = [x^1 \neq x^2]$ не реализуема одним нейроном.

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:

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

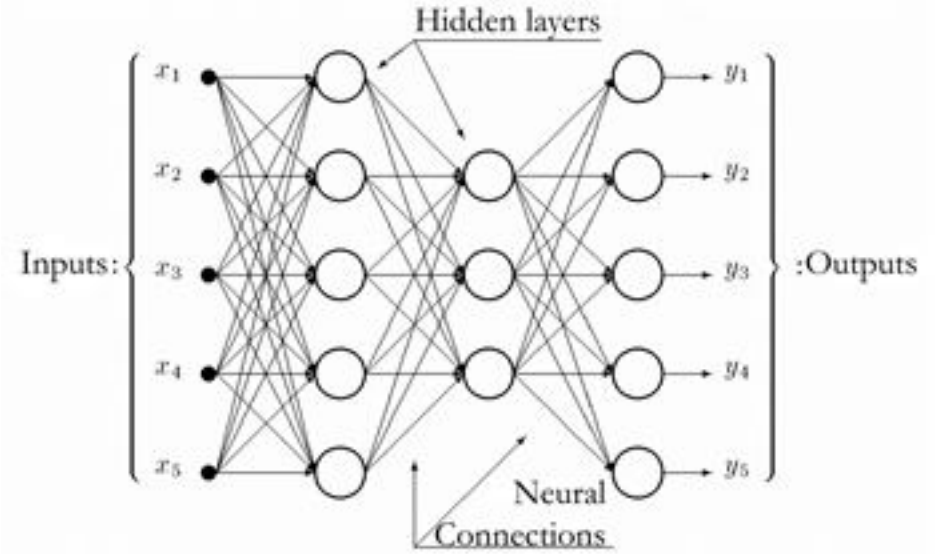
где h_k, φ_{ik} — непрерывные функции, и φ_{ik} не зависят от f.

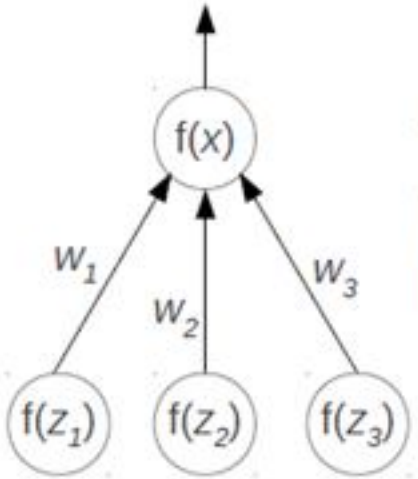
Представимость функций

- 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



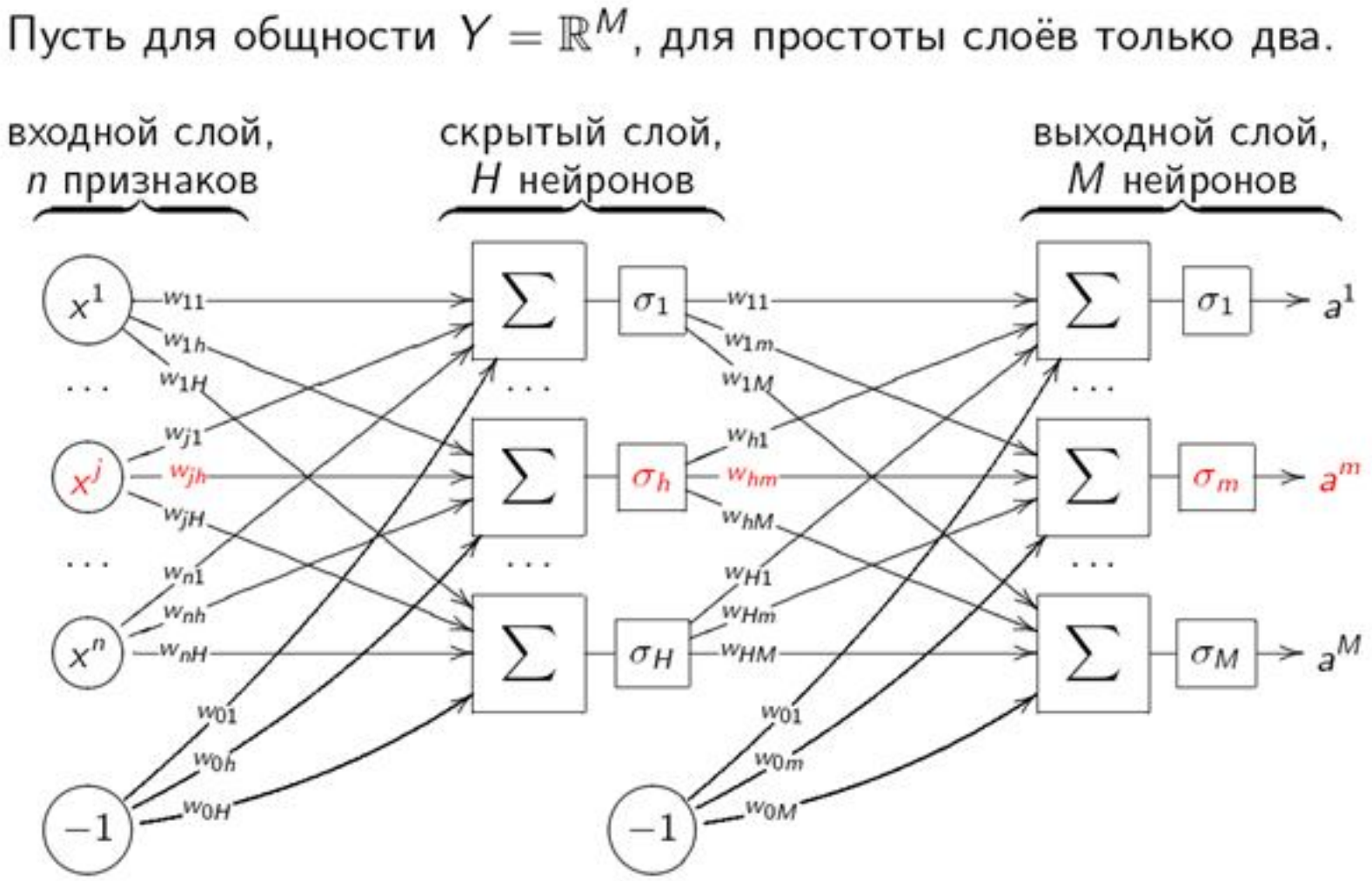


$x = w_1 f(z_1) + w_2 f(z_2) + w_3 f(z_3)$

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
  - Архитектуру пока не умеют «учить», её задаёт разработчик, исходя из опыта и «лучших примеров»
  - Есть методы «упрощения» архитектуры
- **Веса нейросети** – совокупность весов всех рёбер (веса каждого нейрона)
  - Настройка весов – «обучение» нейросети, для этого предложено несколько подходов

Многослойная нейросеть

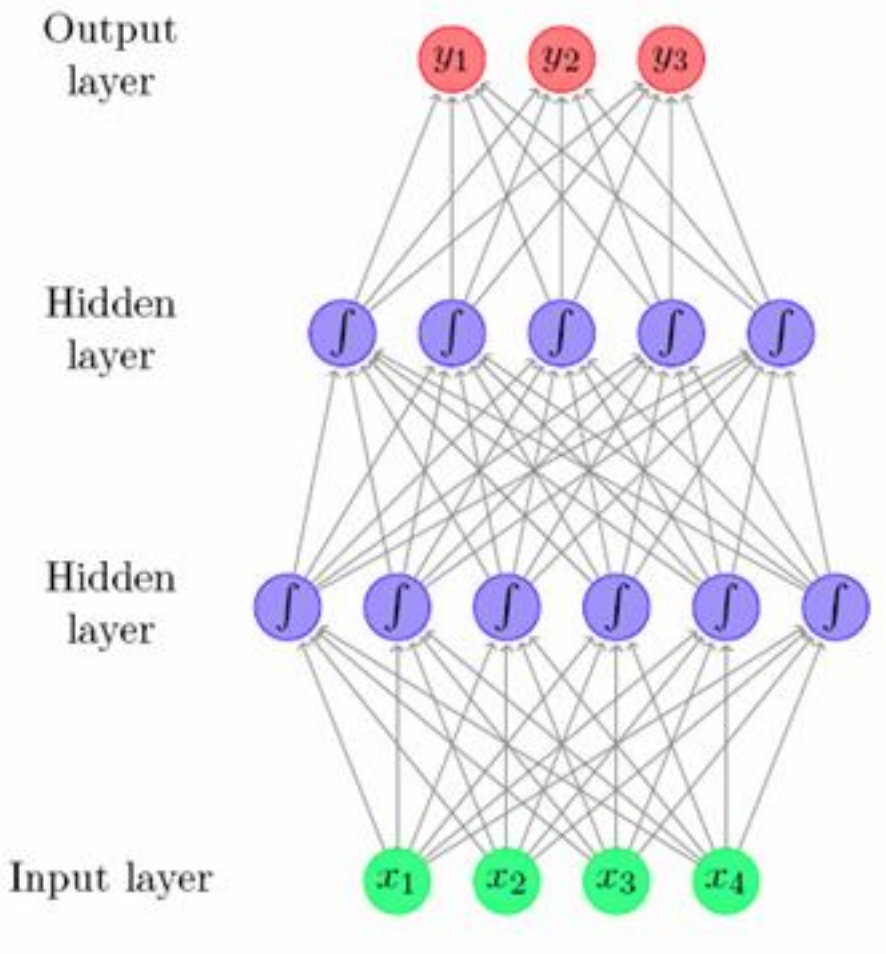


- Передача сигналов идёт в одном направлении (feed-forward)
- Сеть можно разделить на «слои», по числу предшествующих нейронов на пути сигнала

Архитектура нейросети

- Мы будем рассматривать архитектуры, применяемые для анализа изображений
- Вход – обычно изображение $I \in \mathbb{R}^{width \times height \times c}$ (тензор размерности 3)
- 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)

Слои

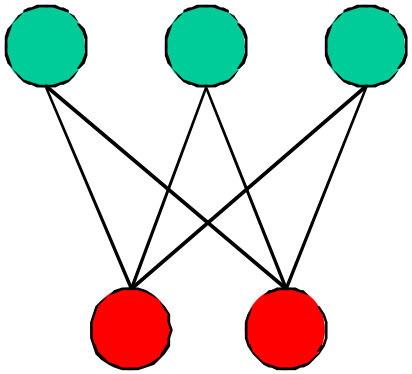


- Выход каждого слоя можно обозначить через x
- x может быть одномерным вектором (как в примере)
- Но при обработке изображений его часто удобно представлять в виде тензора

$$X \in \mathbb{R}^{m \times n \times k}$$

- Под «слоями» будем понимать нейроны
- Слой получает на вход тензор и выдает тензор данных

Линейный перспептрон



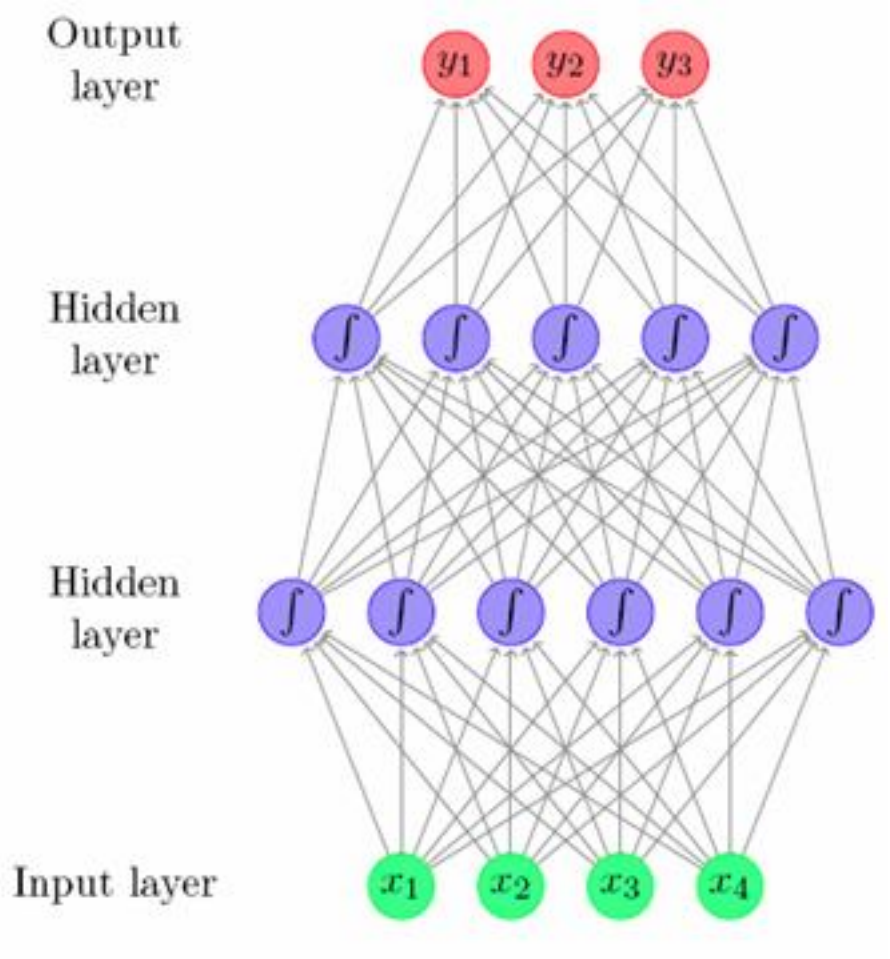
Простейшая нейронная сеть с входом и выходом, реализующая линейную функцию:

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

$$X \in \mathbb{R}^{d_{in}} \quad W \in \mathbb{R}^{d_{in} x d_{out}} \quad b \in \mathbb{R}^{d_{out}}$$

Вход и выход мы видим, поэтому это видимые слои

Персептрон



- Промежуточные слои – «скрытые»
- Если каждый нейрон слоя связан с каждым нейроном предыдущего слоя, то такой слой – полносвязанный (fully connected)
- Нейроны скрытых слоёв обычно имеют функцию активации (нелинейную)

Функция перспетрона

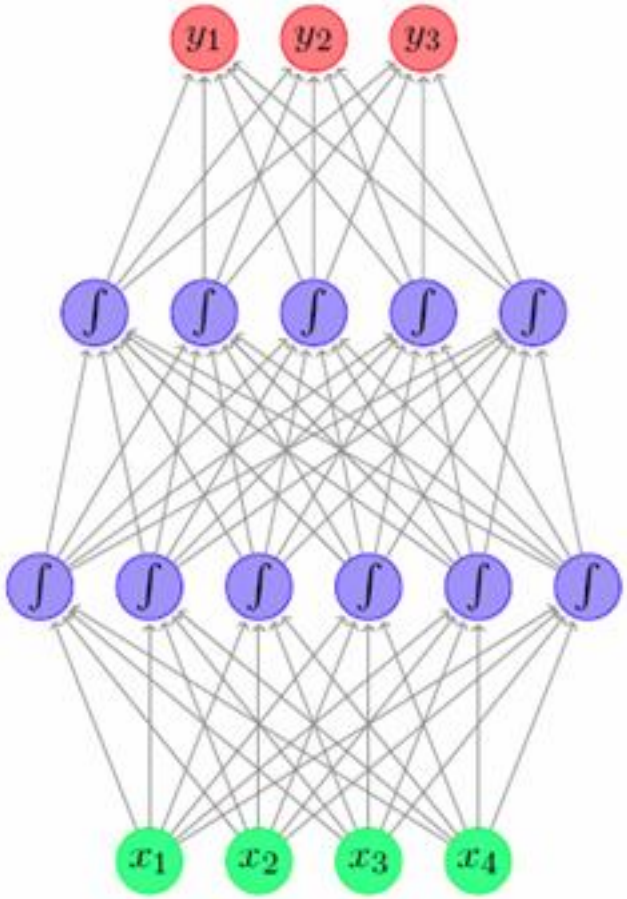
Варианты записей:

- Одна функция:

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

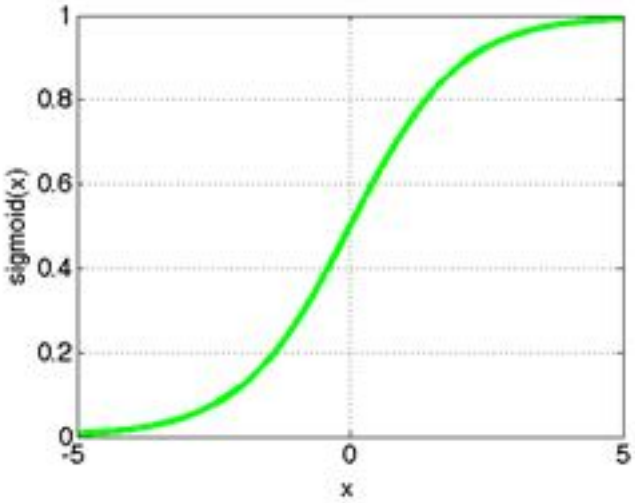
- Как послойные преобразования:

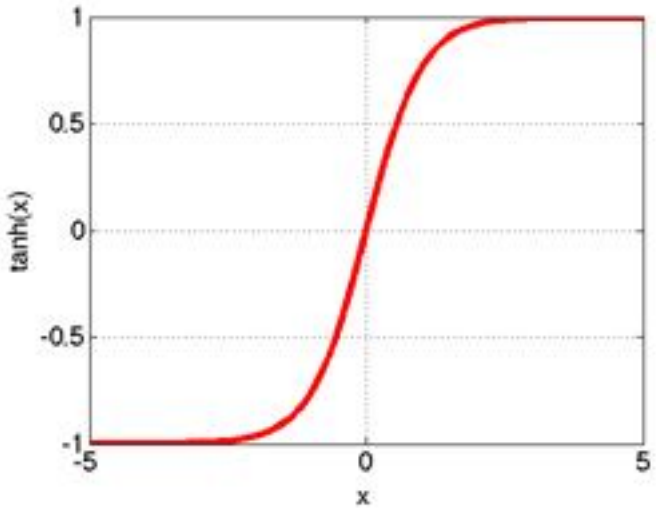
$$\begin{aligned} NN_{MLP2}(\mathbf{x}) &= \mathbf{y} \\ \mathbf{h^1} &= g^1(\mathbf{x}\mathbf{W^1} + \mathbf{b^1}) \\ \mathbf{h^2} &= g^2(\mathbf{h^1}\mathbf{W^2} + \mathbf{b^2}) \\ \mathbf{y} &= \mathbf{h^2}\mathbf{W^3} \end{aligned}$$

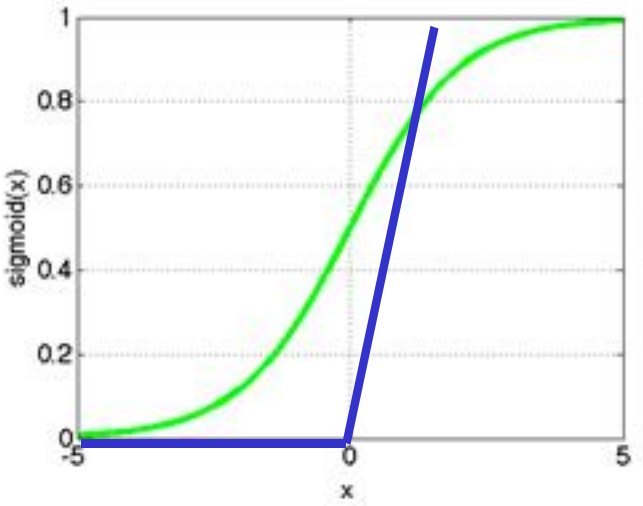


Функции активации

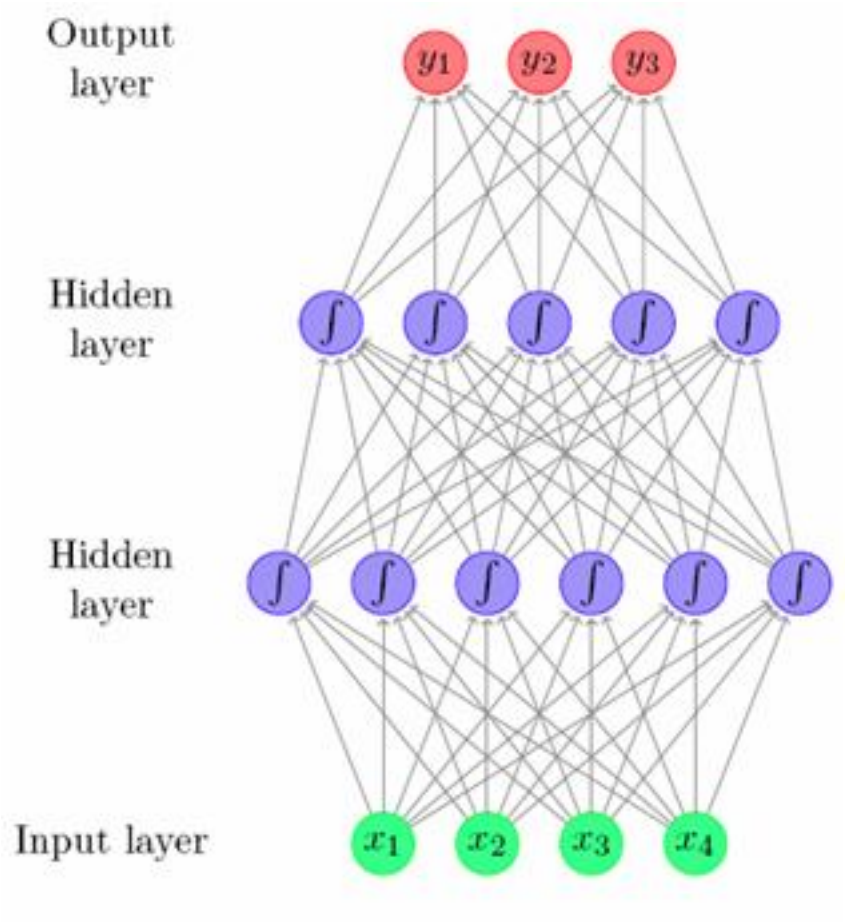
- Tanh
  - $\tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$
- Sigmoid:
  - $\mathrm{sigm}(x) = \frac{1}{1+e^{-x}}$ - преобразует все значения в интервал [0,1]
- Rectified linear
  - ReLU(x) = max(0,x)







Персептрон



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

Функции потерь

- Бинарная классификация (1 нейрон)
  - y метка из {-1,1}, $\hat{y}$ - выход классификатора (вещественное число)
  - Ответ классификатора даётся в виде $\text{sign}(\hat{y})$
  - Ответ правильный, если $y\hat{y}>0$ (одинаковый знак)
  - Ошибка **hinge loss**:

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

  - Потери 0, если ответ правильный и $|\hat{y}| \geq 1$ (уверенный)
- Многоклассовая классификация (2+ нейрона)
  - Categorical cross-entropy loss

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

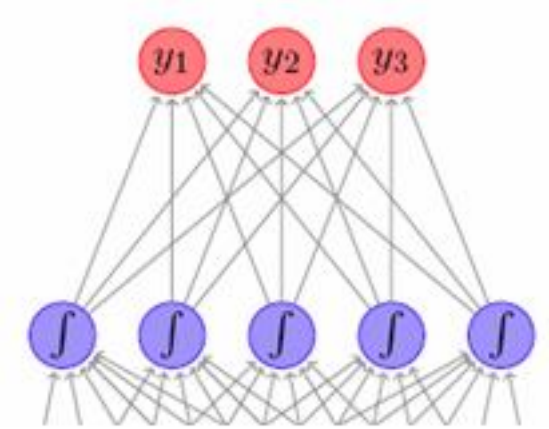
  - Измеряет близость истинного и оцененного распределения меток (обычно после softmax)

Softmax-преобразование

- Функция из многоклассовой логистической регрессии:

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

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



- «Сжимаем» произвольный вектор длины N в вектор со значениями (0,1) и суммой 1
- Полученный вектор можем интерпретировать как вектор вероятностей меток классов

Как обучать?

- **Отрицание проблемы (denial)**
  - Однослойные сети
  - Эвристическое задание параметров
- **Эволюционная стратегия**
  - Случайно меняем (jitter) веса и оставляем, если сеть улучшилась
- **Прокрастинация**
  - Откладываем главную задачу «на потом», обучаем веса на каждом слое так, чтобы они извлекали какую-то полезную информацию
- **Оптимизация всех параметров в совокупности за счёт матана** (calculus)
  - Градиентный спуск – обратное распространение ошибки (backpropagation)

Geofrey Hinton «To Recognize Shapes, First Learn to Generate Images», TR, 2006

Стохастический градиентный спуск

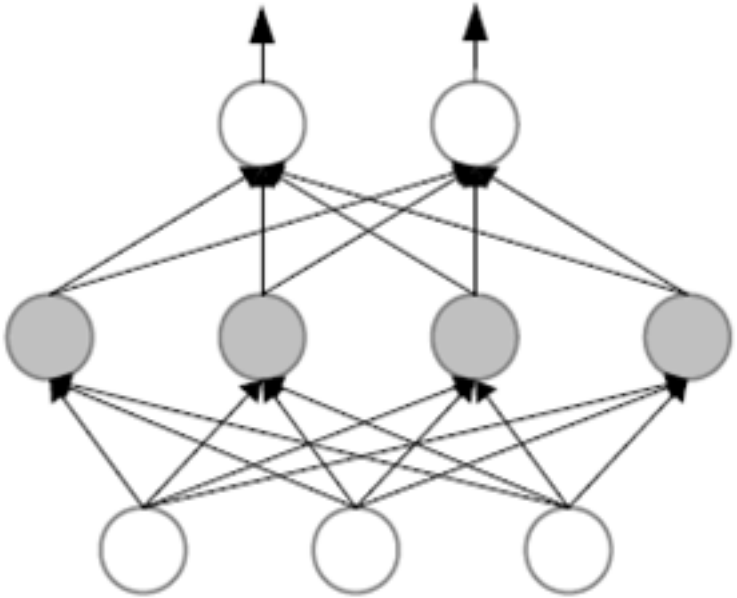
Точно также, как при обучении одного нейрона (линейного классификатора), можем обучать все параметры нейросети

Algorithm 1 Online 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 training example $\mathbf{x_i}, \mathbf{y_i}$
    Compute the loss $L(f(\mathbf{x_i};\theta), \mathbf{y_i})$
    $\hat{\mathbf{g}} \leftarrow$ gradients of $L(f(\mathbf{x_i};\theta), \mathbf{y_i})$ w.r.t θ
    $\theta \leftarrow \theta + \eta_k \hat{\mathbf{g}}$
return θ

Как посчитать градиент для всех параметров нейросети?

Расчёт градиента



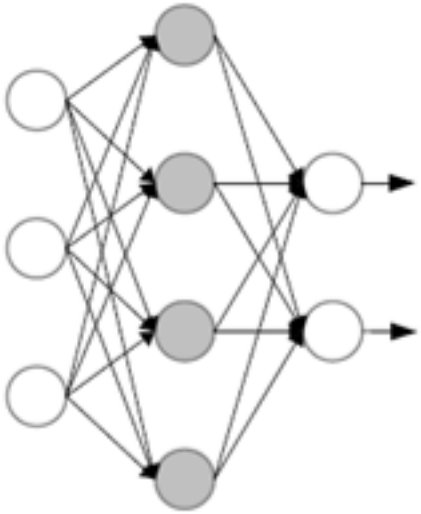
- Нейросеть вычисляет дифференцируемую функцию от своих входов
- Можем последовательно применять правило дифференцирования сложных функций для вычисления производных по каждому параметру нейросети
- Метод получил название «обратное распространение ошибки» и имеет длинную историю

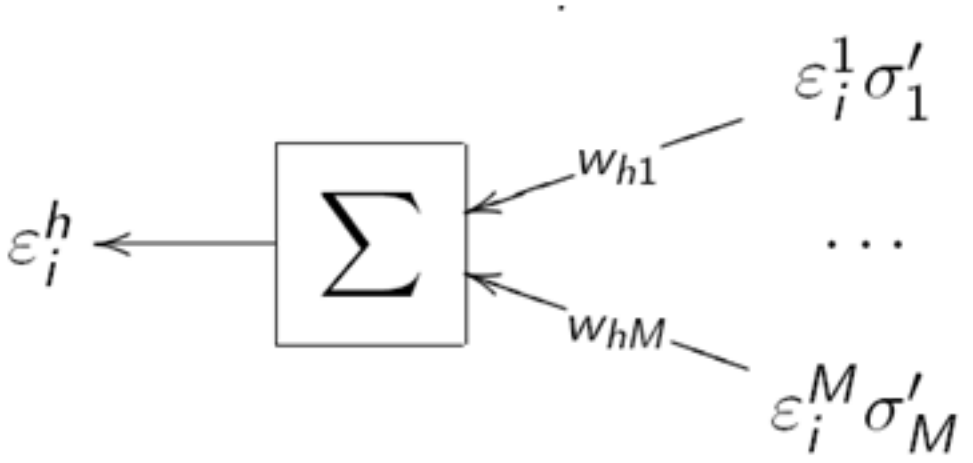
•↑ *Галушкин А. И.* Синтез многослойных систем распознавания образов. — М.: «Энергия», 1974.

•↑ *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)

Инициализация сети

- Пусть веса слоя $\mathbf{W} \in \mathbb{R}^{d_{in} \times d_{out}}$

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

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

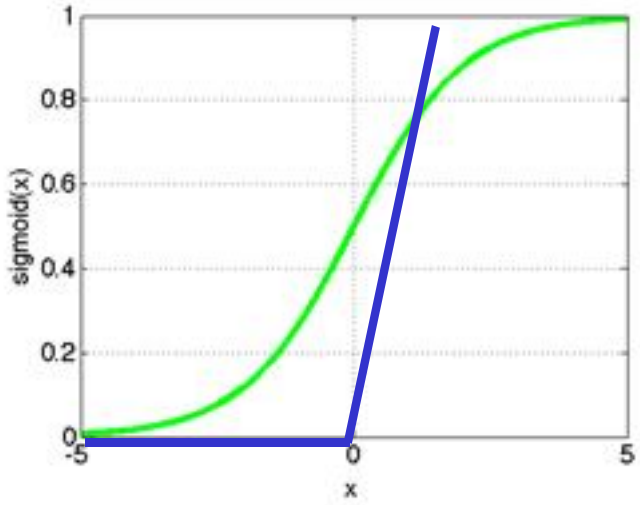
Развитие SGD

- 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

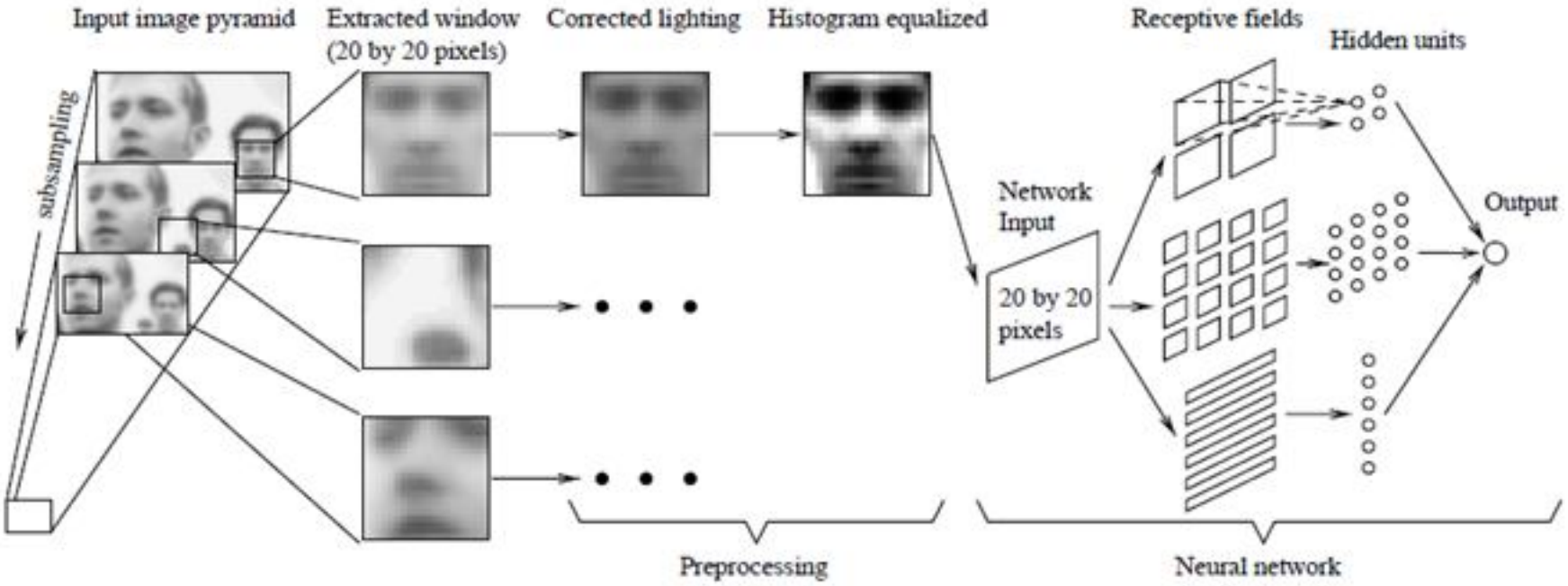
Проблемы обучения

- «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



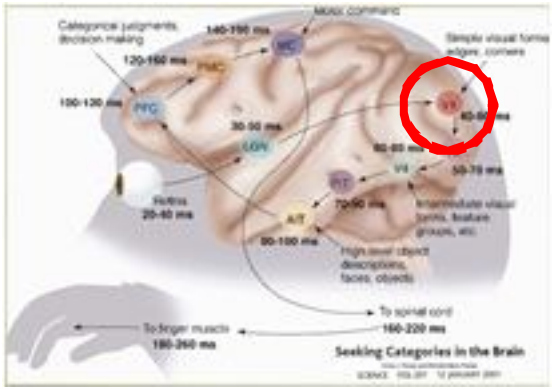
B. Rowley, T. Kanade. Neural Network-Based Face Detection. PAMI, 1998.

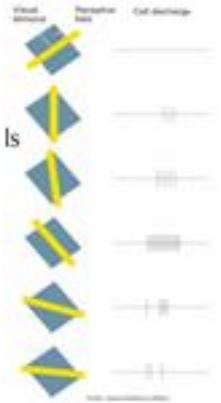
Использование некоторых знаний о зрении человека в архитектурах нейросетей

Знания о зрении человека

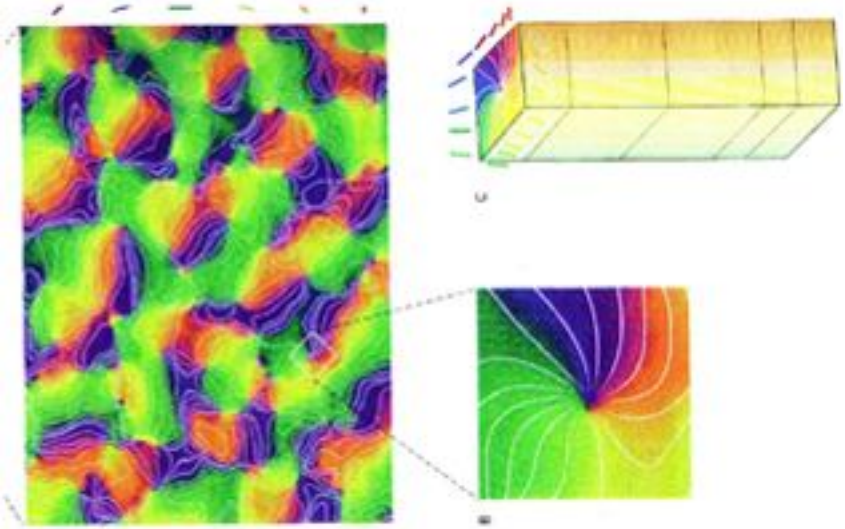
Воспользуемся знаниями об обработке визуальной информации в мозге человека.

Интуитивно кажется, что основная информация в картинке содержится в краях (границах)

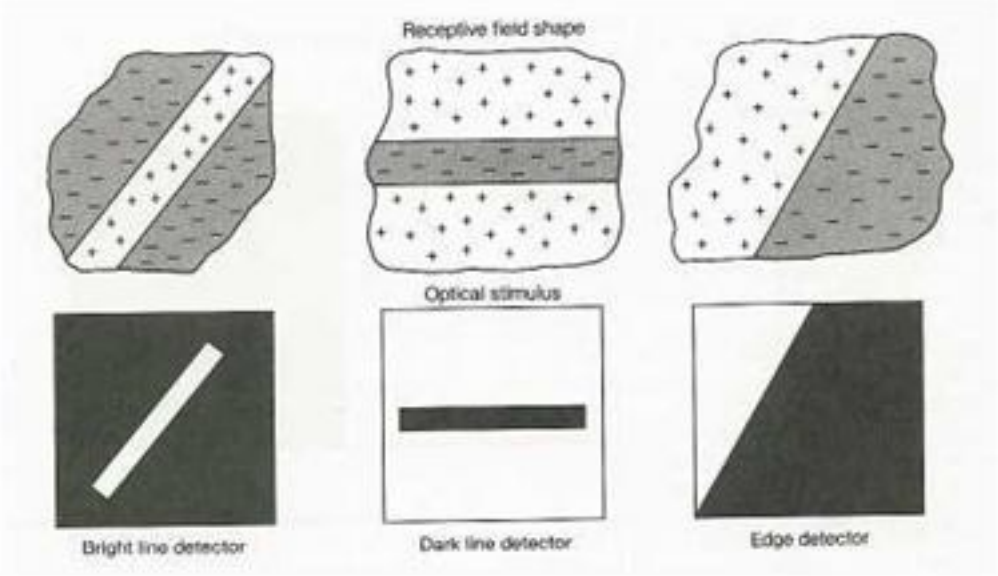




- В первичной визуальной коре головного мозга есть клетки, чувствительные к краям определенной ориентации
- Для каждой области есть набор таких клеток, чувствительные к краям разной ориентации

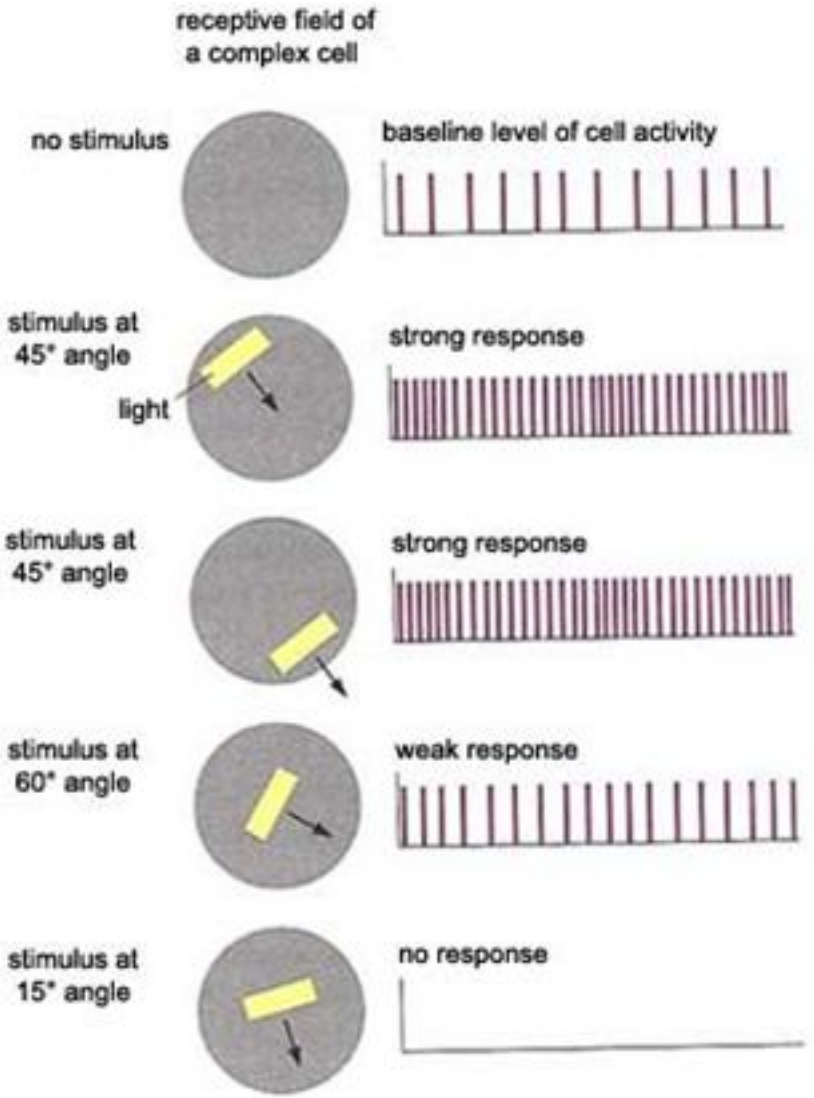


Простые (S) и сложные (C) клетки



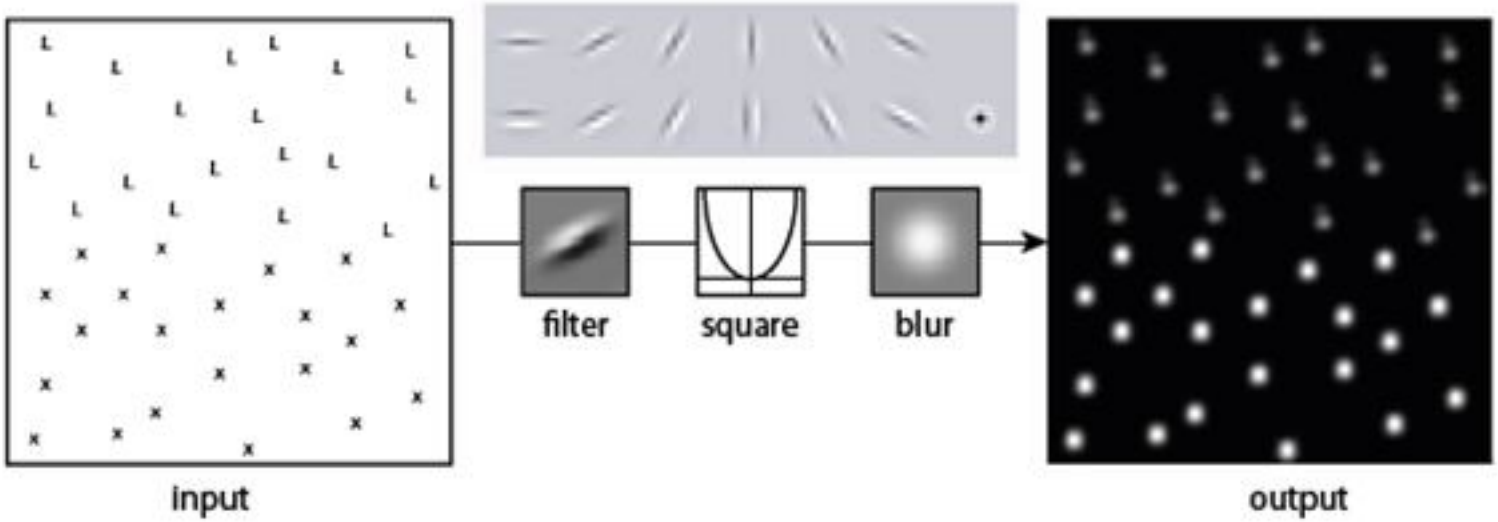
Простые клетки

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



Сложные клетки

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

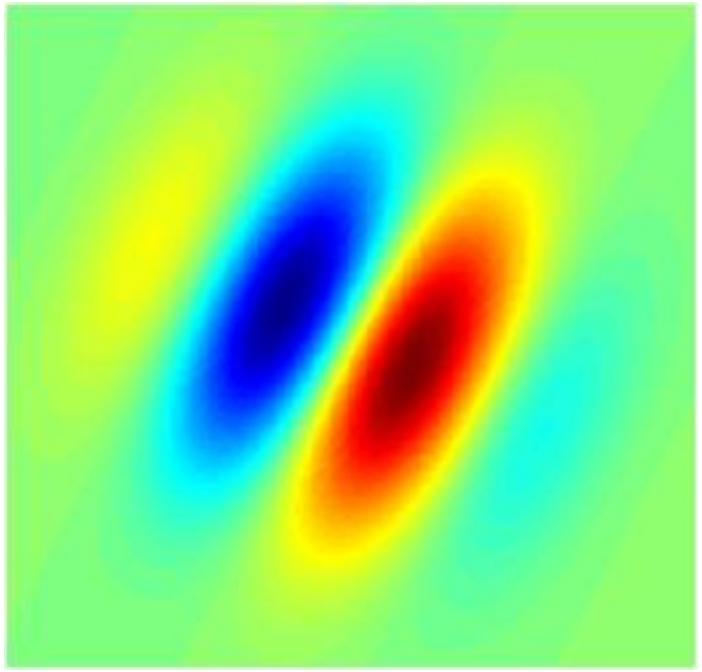
Pietro Perona and Jitendra Malik «Detecting and Localizing edges composed of steps, peaks and roofs», ICCV 1990

Фильтры Габора

$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right)\cos\left(2\pi\frac{x'}{\lambda} + \psi\right)$$

$$x' = x\cos(\theta) + y\sin(\theta)$$

$$y' = -x\sin(\theta) + y\cos(\theta)$$



θ - ориентация

λ - длина волны

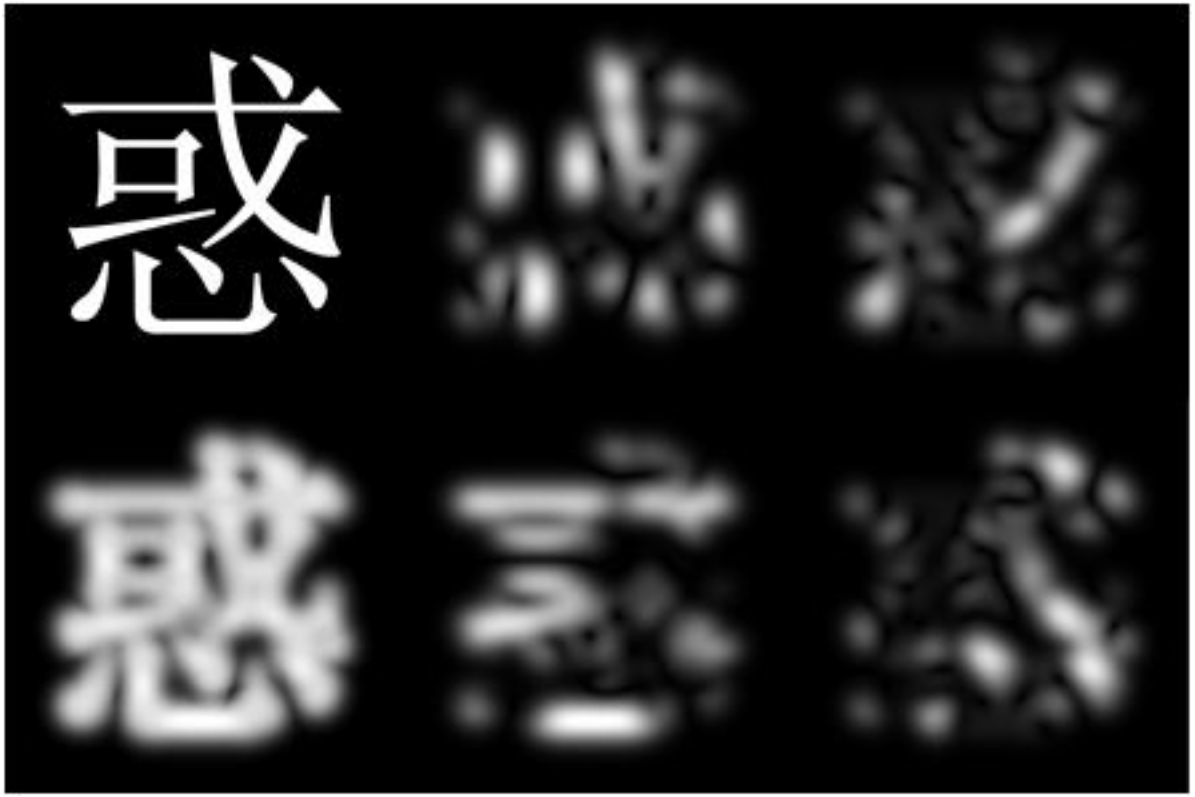
σ - сигма гауссиана

γ - соотношение размеров (aspect ratio), «эллиптичность фильтра»

ψ - сдвиг фазы

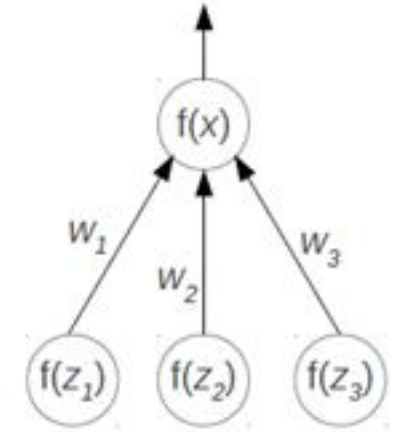
- 2D фильтр Габора – ядро гауссиана, домноженное на синусоиду
- Предложены в 1947 Денисом Габором (нобелевским лауреатом), независимо переоткрыты в 1980 году
- Позволяет сделать банк фильтров, для выделения краёв разной ориентации, масштаба и положения в окрестности

Поиск краёв с помощью Габора

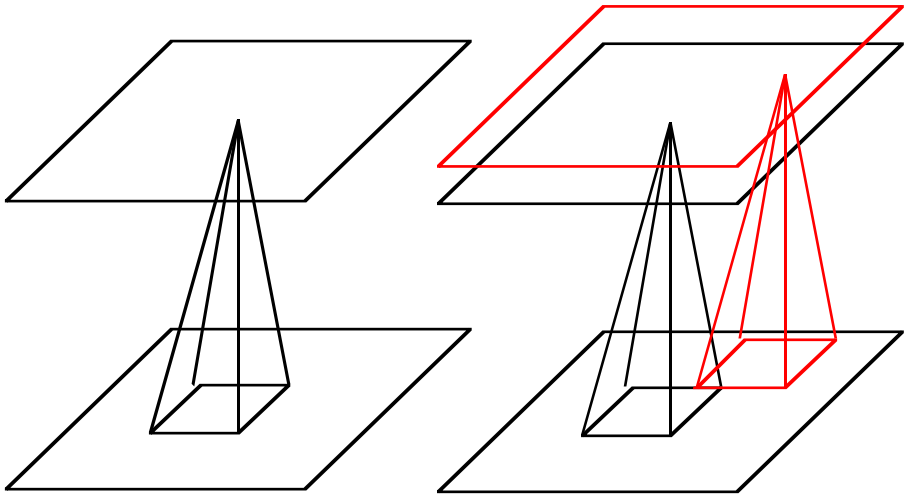


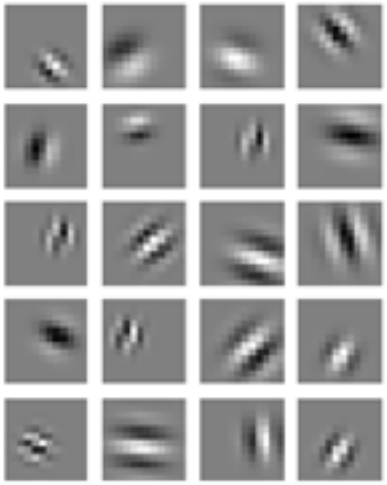
- Слева вверху – иероглиф
- 4 справа – применение фильтров Габора с ориентациями 0, 45, 90 и 135 градусов
- Слева внизу – совмещение результатов фильтрации

Нейрон как линейный фильтр



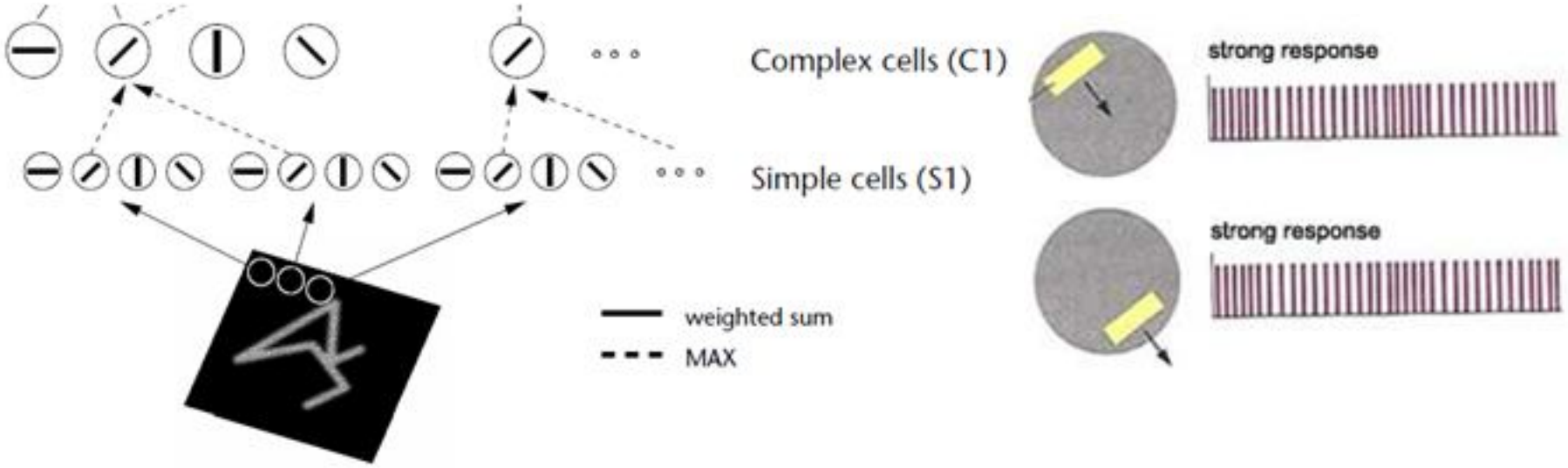
$$x = w_1 f(z_1) + w_2 f(z_2) + w_3 f(z_3)$$





- Операцию линейной фильтрации (свёртки) для одного пикселя можно реализовать одним нейроном
- Свёртку изображения целиком можно реализовать как «слой» нейронов, веса которых одинаковы
  - *Свёрточный слой*
- Набор свёрточных слоёв реализует свёртку с банком фильтров
- Меньшее число параметров, чем у полносвязного слоя с тем же числом нейронов

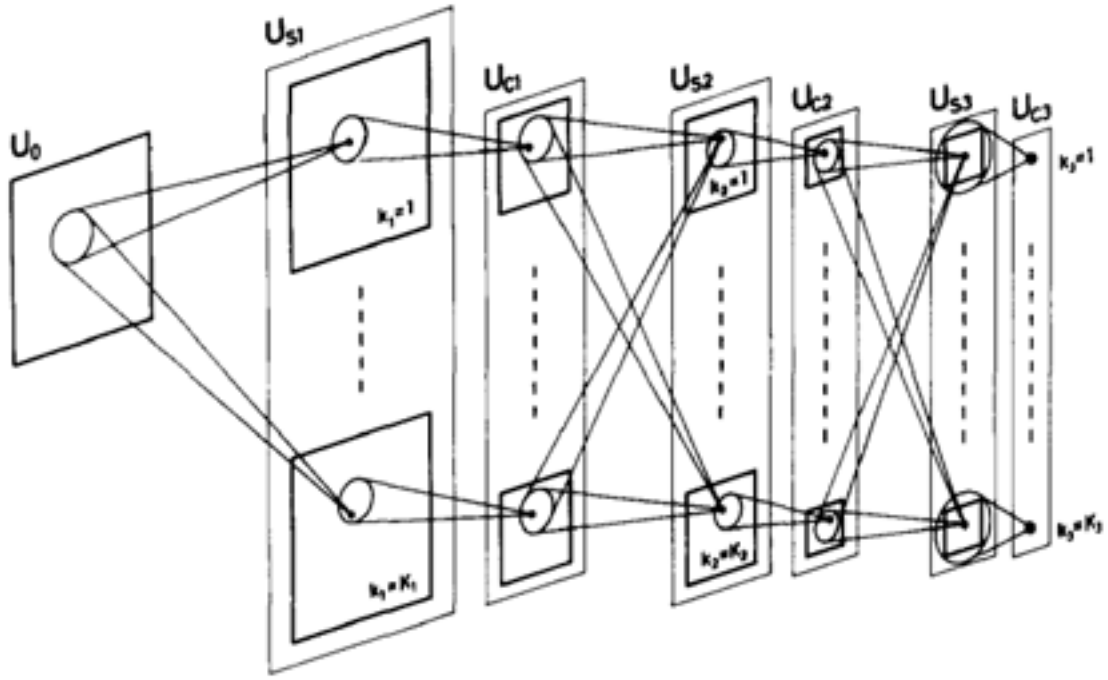
Инвариантность через MAX-pooling



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.

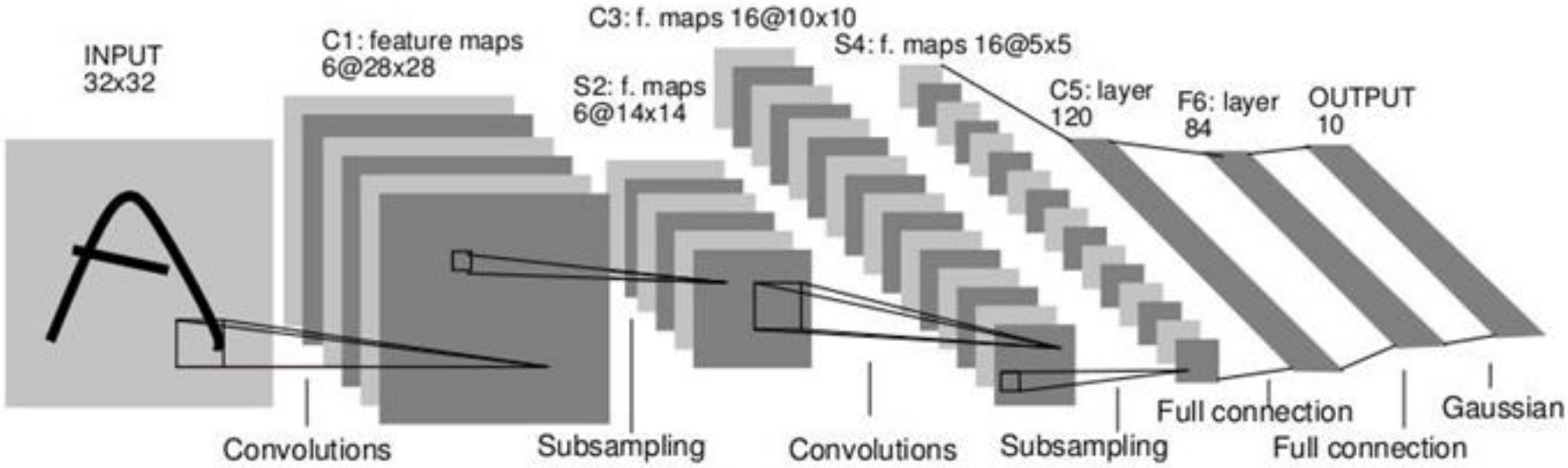
Neocognitron (1980)



- 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.

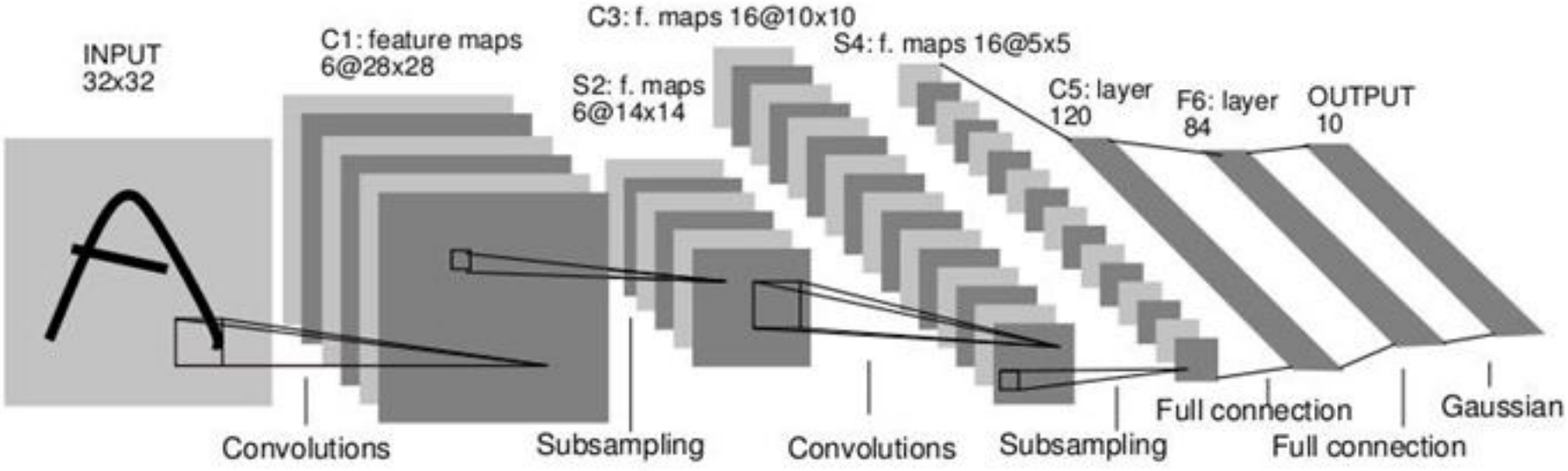
Свёрточные сети



- 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

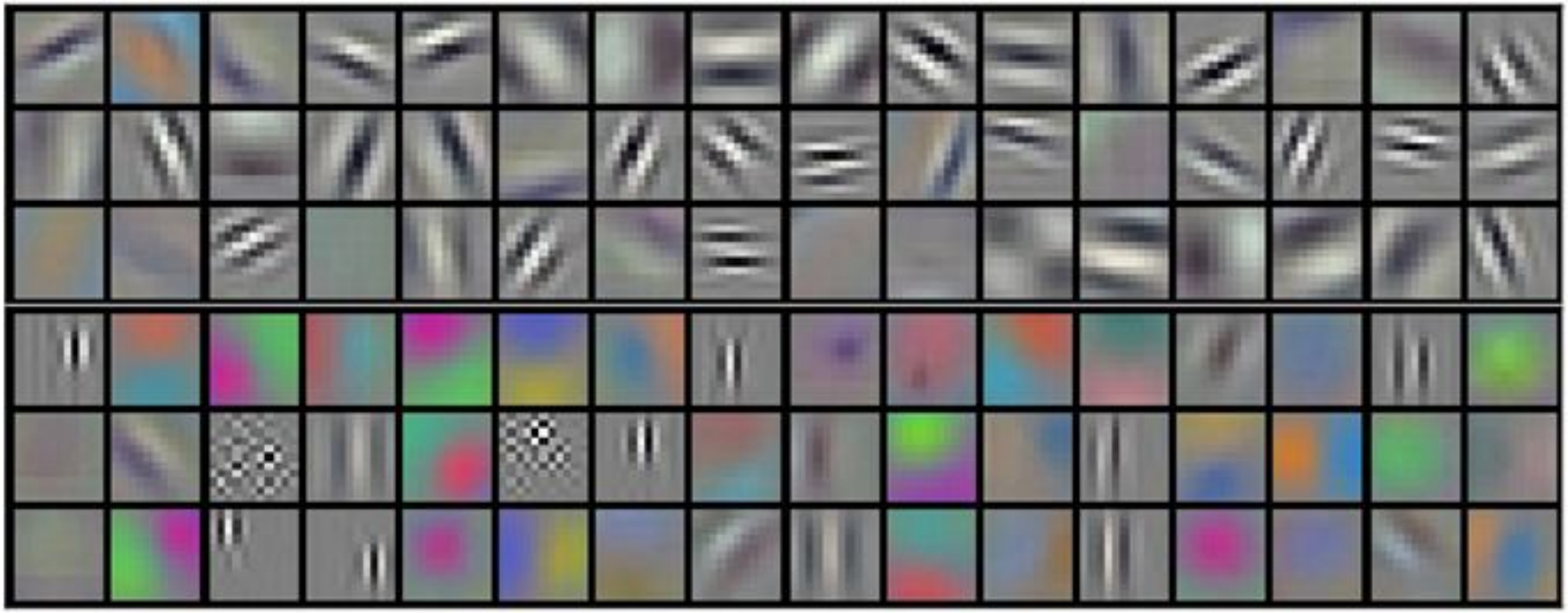
LeCun, Y., Bottou, L., Bengio, Y., and Haner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE

Свёрточные слои



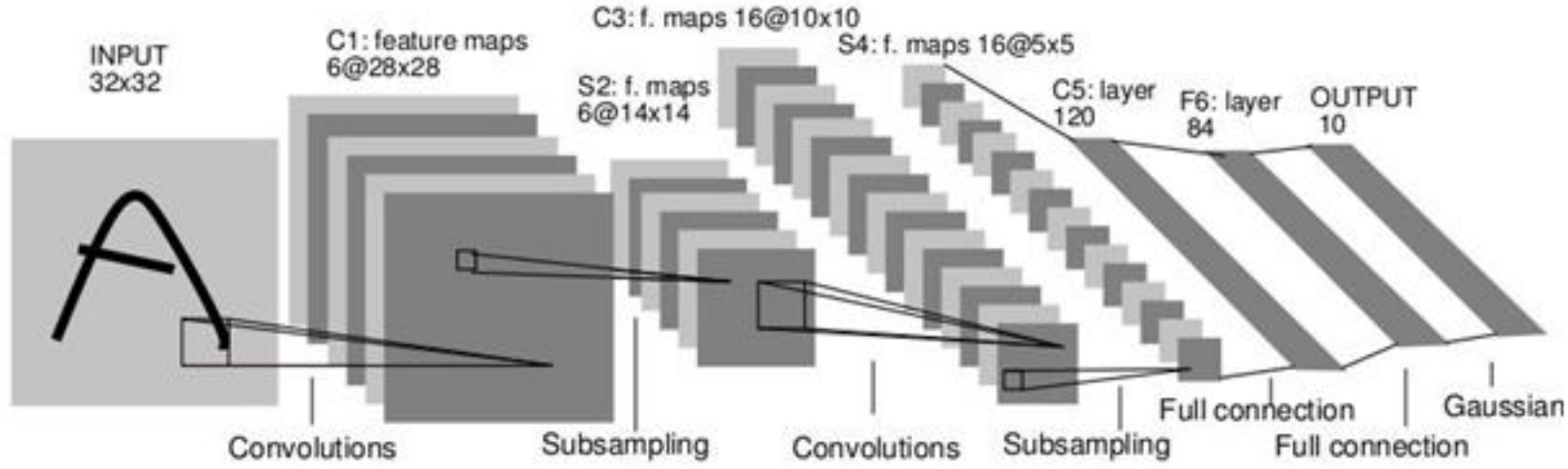
- Каковы размеры фильтров свёрток на разных слоях?
  - 5*5*1 на первом слое
  - 5*5*6 на втором свёрточном слое
  - «Глубина» тензора на выходе свёрточного слоя равна числу свёрток в свёрточном слое
  - 3е измерение свёртки равно «толщине» входного тензора
- Число весов и параметров второго слоя:
  -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

Фильтры первого уровня

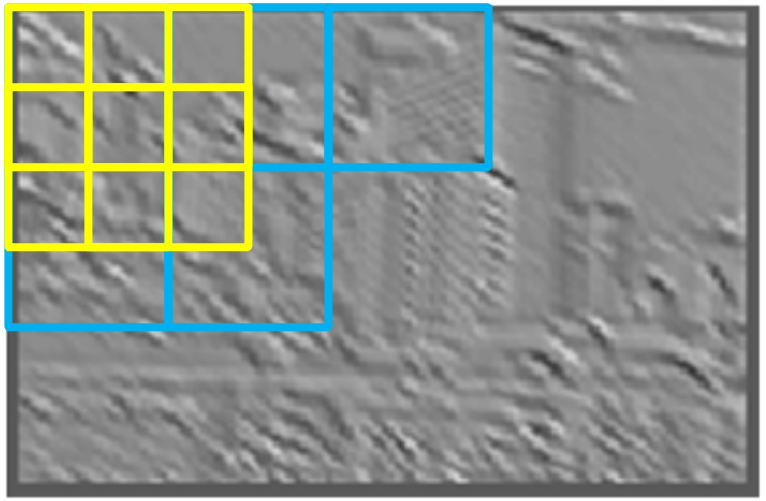


- Визуализируем веса фильтров
- Поскольку сворачиваем RGB изображение, то визуализация весов в RGB
- Обратите внимание на вычисление градиентов цветов

Subsampling/Pooling слои



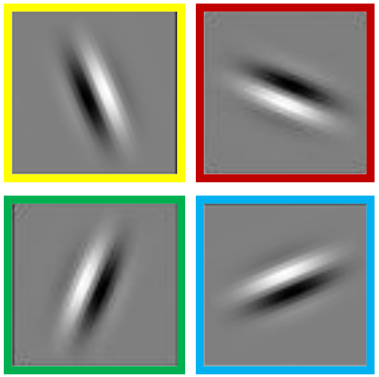
- Subsampling слой – уменьшение разрешения
- Обычно **Sum** или **Max** по областям с некоторым шагом
  - Sum-pooling
  - Max-pooling
- Области берутся с перекрытием или без перекрытия



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

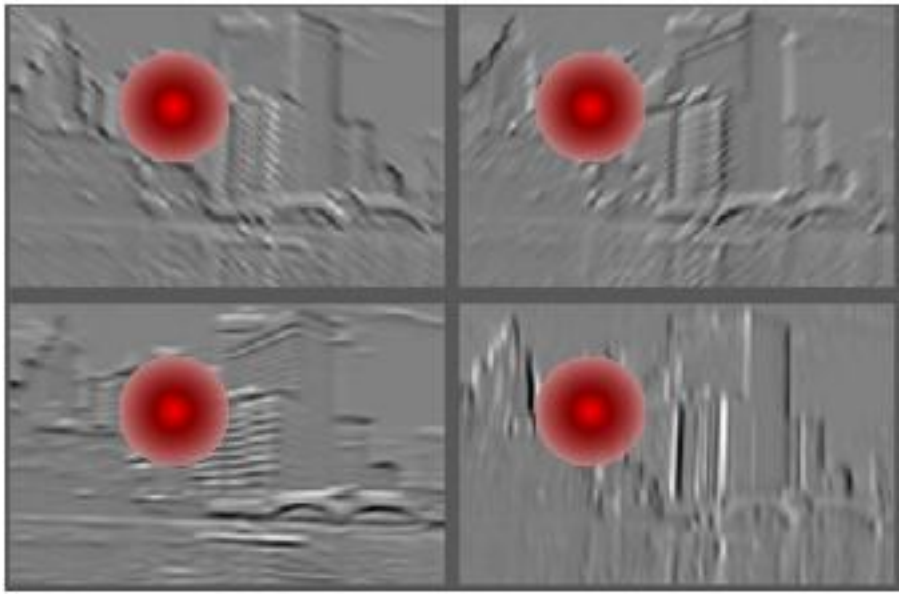
- Можем и не делать отдельный subsampling слой



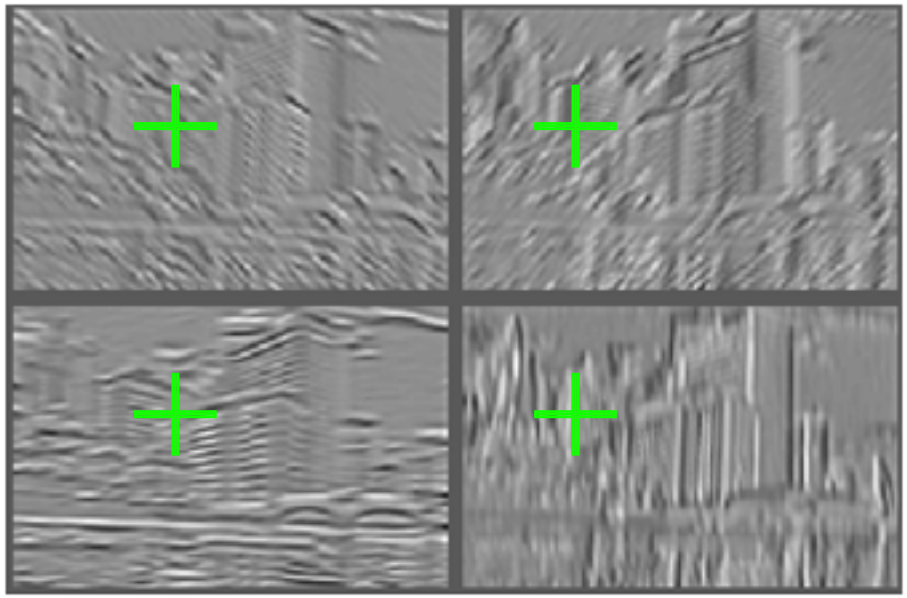
Нормализующие слои

Локальная нормализация:

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



Карты признаков



После нормализации контраста

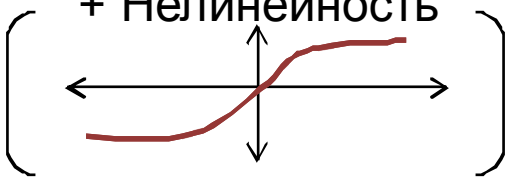
Вспомним построение признаков

Признаки → Кодирование → Учёт по области

Пикселы / Признаки →

Фильтрация по словарю

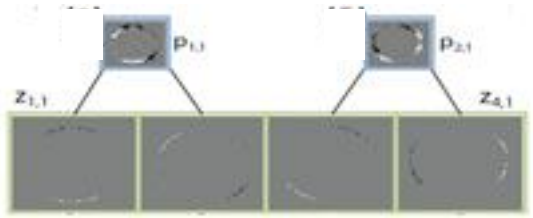
+ Нелинейность



Каждый этап – свой слой сети

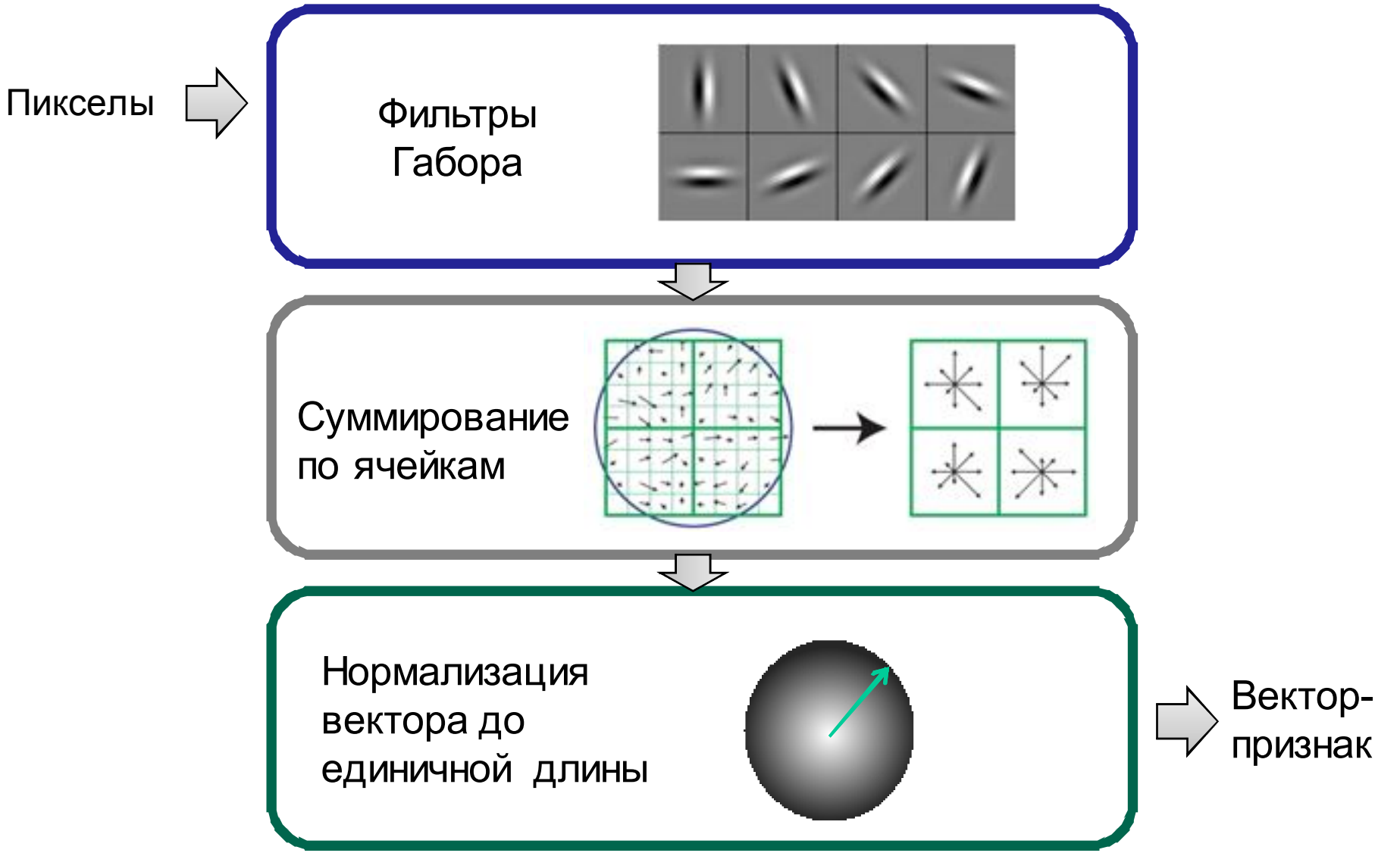
Нормализация | Группировка Разреженность | Max / Softmax | Нормализация локального контраста

Учёт по области (Sum или Max)



→ Признаки

Дескриптор HOG/SIFT



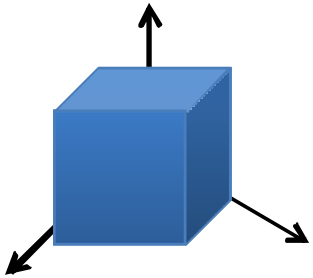
Мешок визуальных слов

HOG

Свёртка с словарём

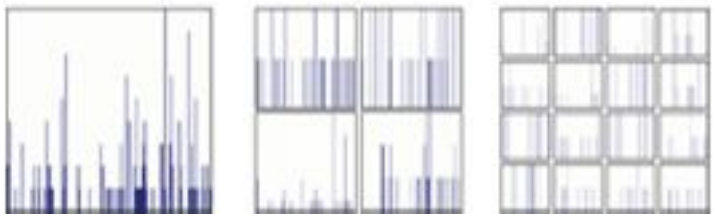
Max

(Находим самое близкое слово из словаря – квантуем)



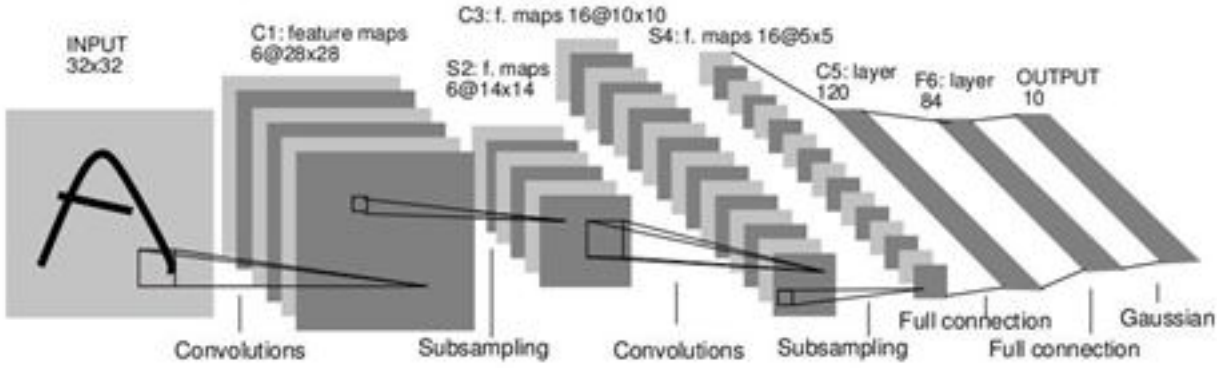
Суммирование по области

(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

Этапный результат

IMAGENET

14M изображений

Будет 1000 на каждую категорию

http://www.image-net.org

Large-scale visual recognition (LSVR)

Task 1: Classification

Car

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

Task 2: Detection
(Classification + Localization)

classification Car

- Predict a class label and a bounding box
- 5 predictions / image
- 1000 classes
- 1,200 images per class for training
- Bounding boxes for 40% of training.

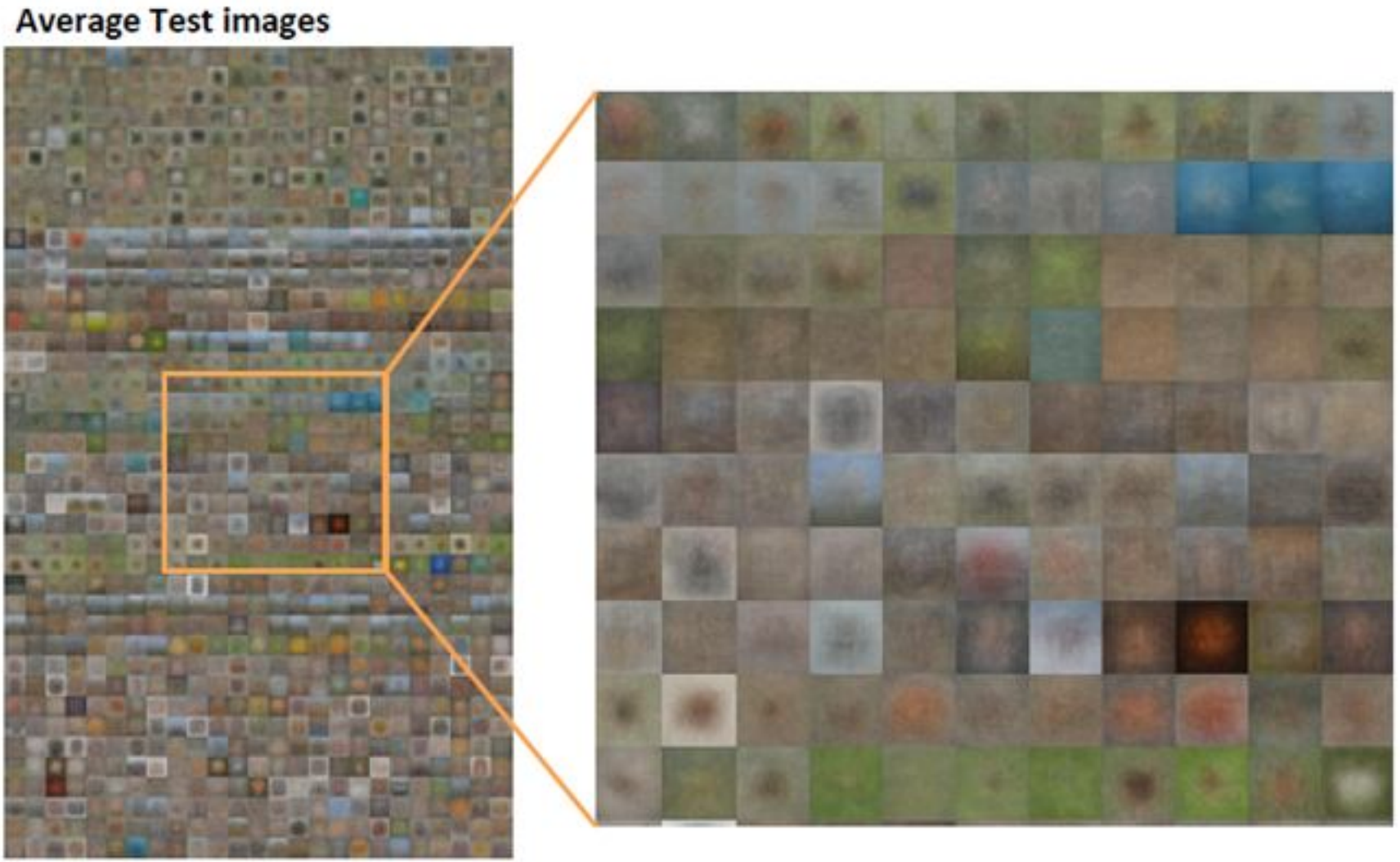
Task 3: Fine-grained classification

classification Walker hound

- Predict a class label given a bounding box in test
- 1 prediction / image
- 120 dog classes (subset)
- ~200 images per class for training (subset)
- Bounding boxes for 100% of training

1.2M изображений для обучения

Изображения в среднем



Нейросети – победители 2012 года

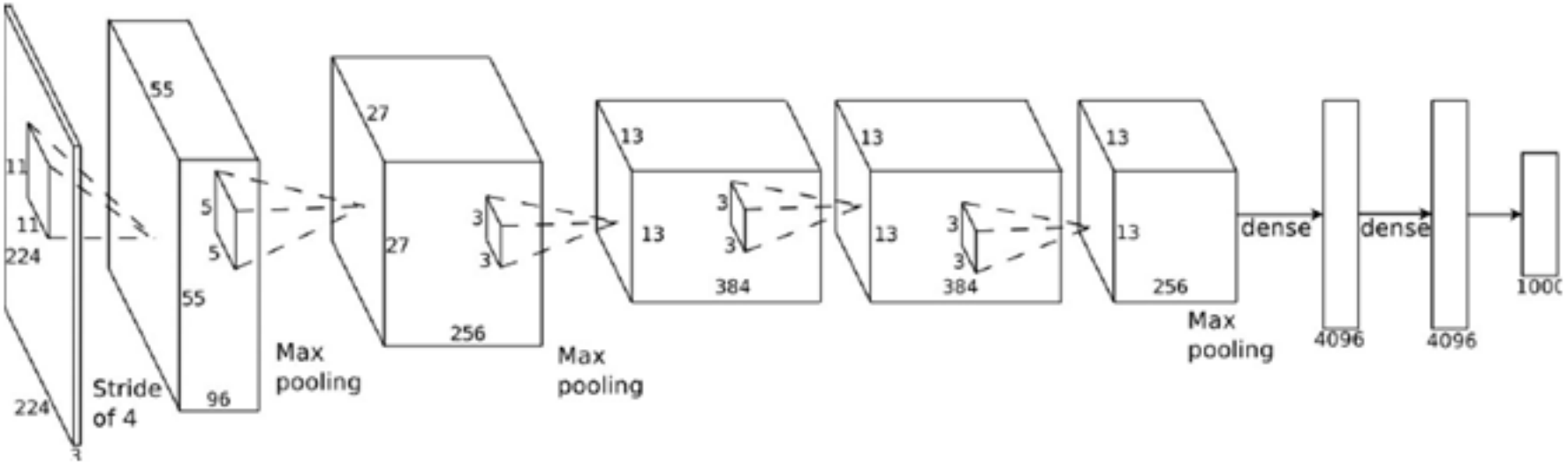
Winner

SuperVision

Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton

University of Toronto

SuperVision

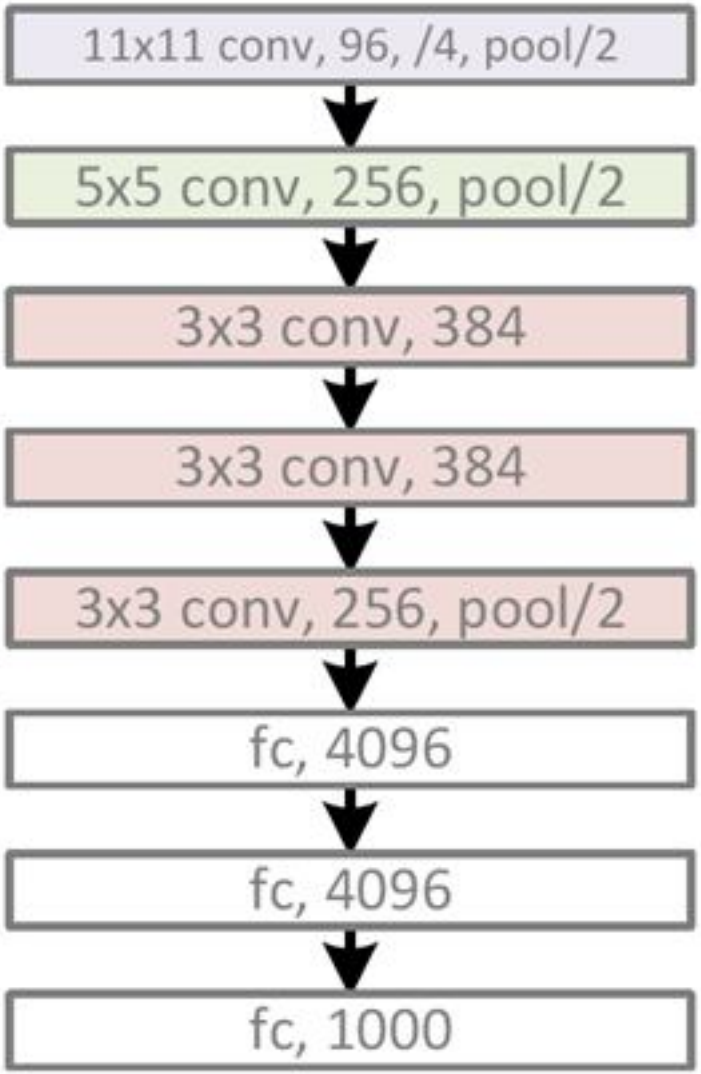


- 650,000 neurons
- 60,000,000 parameters
- 630,000,000 connections
- 1 машина, 2 GPU по 2Gb, 5GB Ram, 27Gb HDD, 1 неделя на обучение

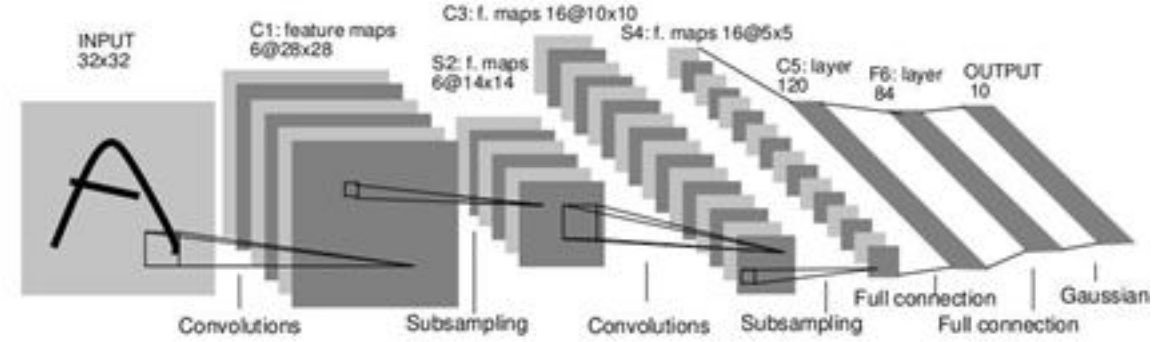
Krizhevsky A., Sutskever I., Hinton, G. E. ImageNet Classification with Deep Convolutional Neural Networks // NIPS 2012

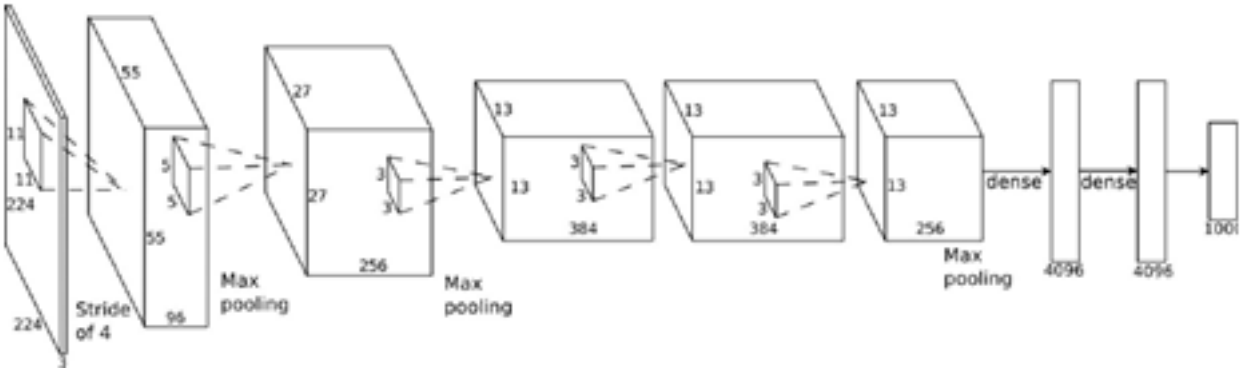
AlexNet

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)

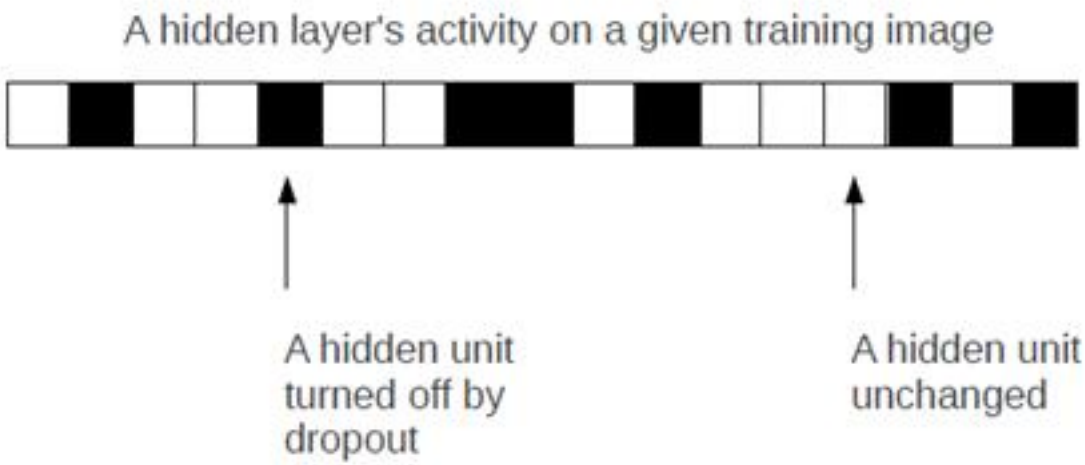
- Больше слоёв, фильтров, нейронов
- За счёт большого объёма данных, вычислительной мощности и некоторых улучшений (ReLU и т.д.) смогли обучить такую большую сеть

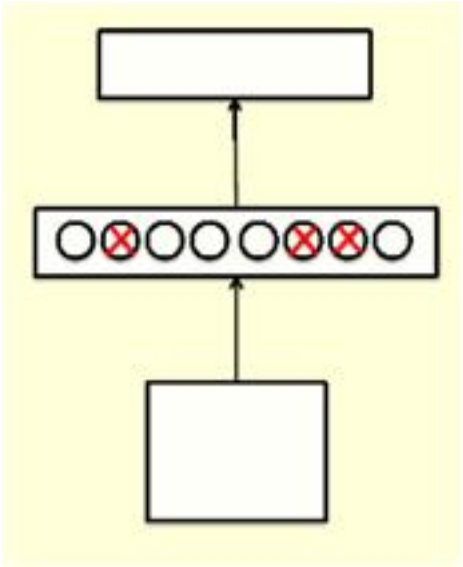
Krizhevsky A., Sutskever I., Hinton, G. E. (2012) ImageNet Classification with Deep Convolutional Neural Networks // NIPS 2012: Neural Information Processing Systems. Lake Tahoe, Nevada.

Размножение данных

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

Dropout





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

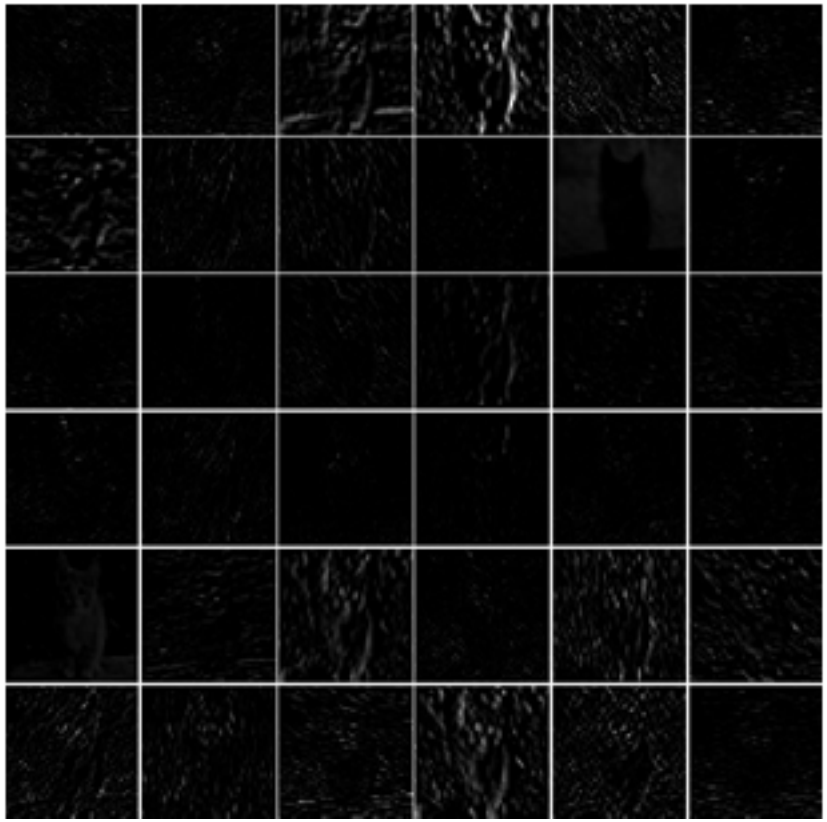
Nitish Srivastava Improving Neural Networks with Dropout. Master Thesis, 2013

Примеры работы

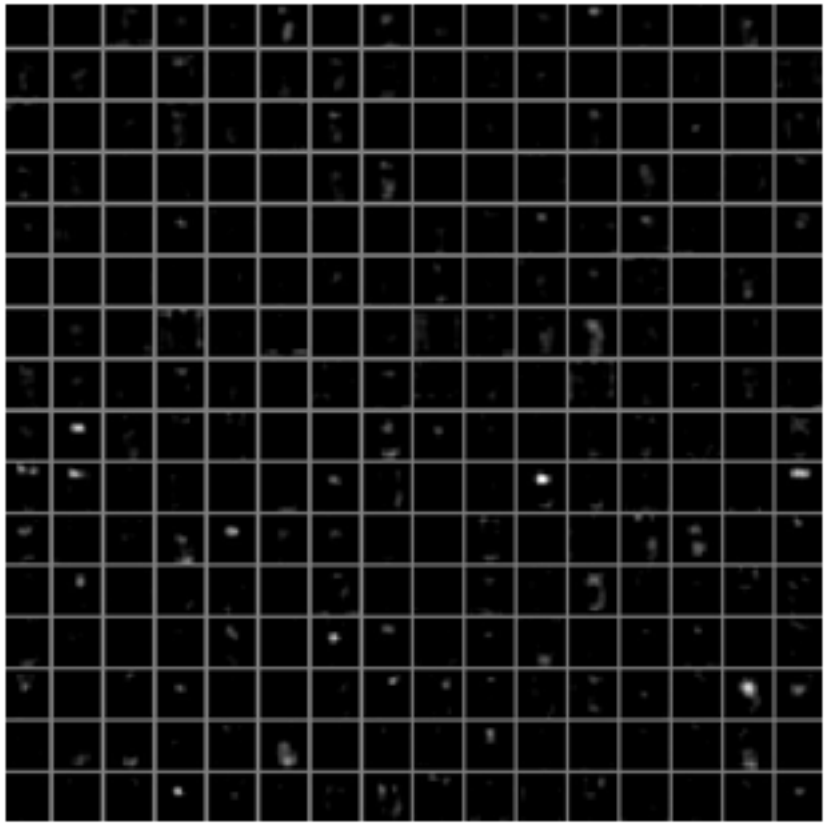
Примеры работы

Визуализация работы нейросети

Визуализация активаций (тензоров)



Слой conv1



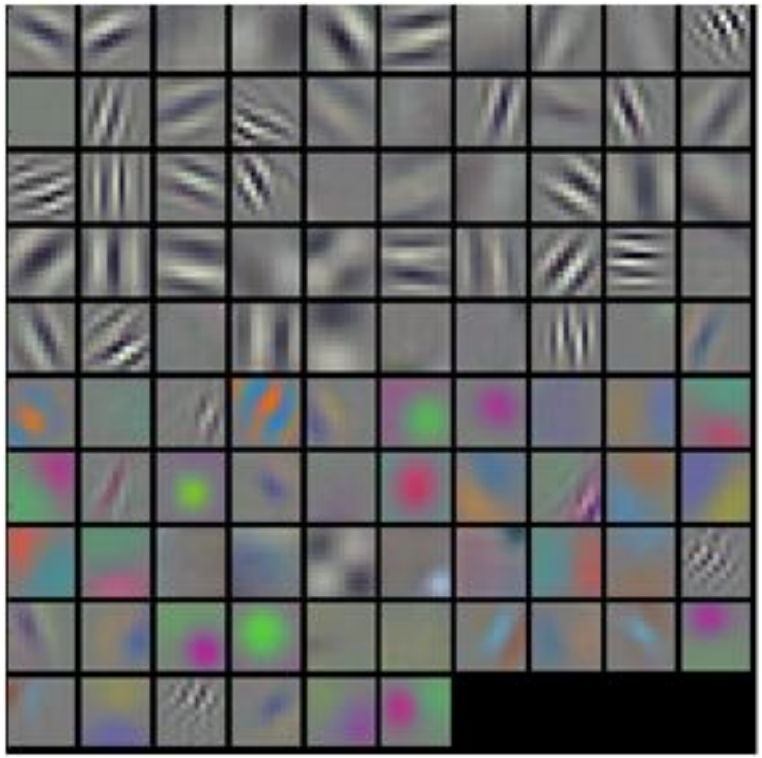
Слой conv5

Обратите внимание на «разреженность» значений

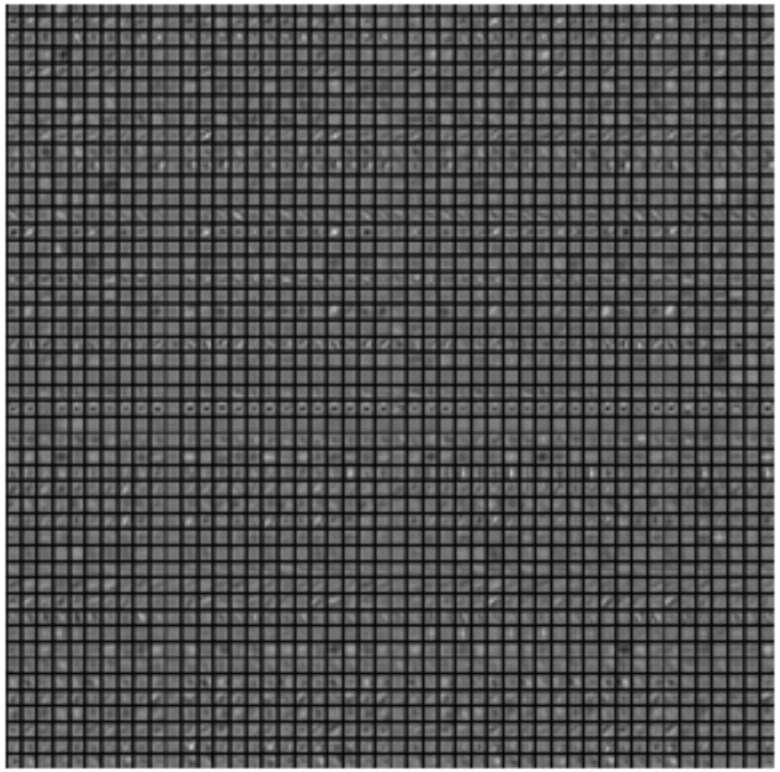
http://cs231n.github.io/understanding-cnn/

Визуализация работы нейросети

Визуализация фильтров



Слой conv1



Слой conv2

http://cs231n.github.io/understanding-cnn/

Визуализация работы нейросети

Изображения, на которых достигается максимальный отклик фильтра

Фильтры слоя pool5

http://cs231n.github.io/understanding-cnn/

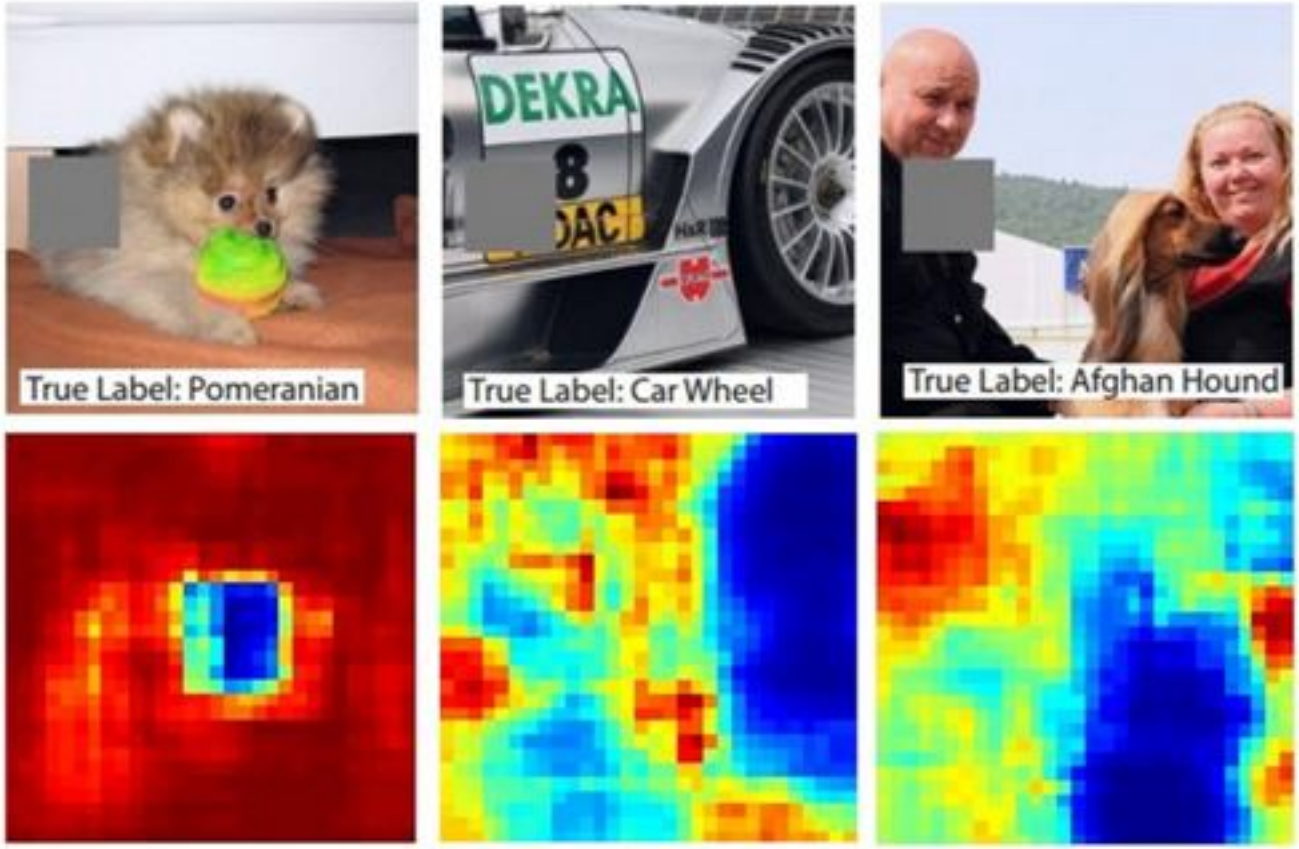
t-SNE

- Можем вычислить L2 расстояние между выходами full6 или full7 слоёв
- Воспользуемся отображением точек из 4096-мерного пространства на 2х мерное, сохраняющее L2 расстояния (приближенно)
- Визуализируем изображения
- Видим, что близкие по смыслу изображения оказываются близки друг к другу

http://cs.stanford.edu/people/karpathy/cnnembed/

Визуализация работы нейросети

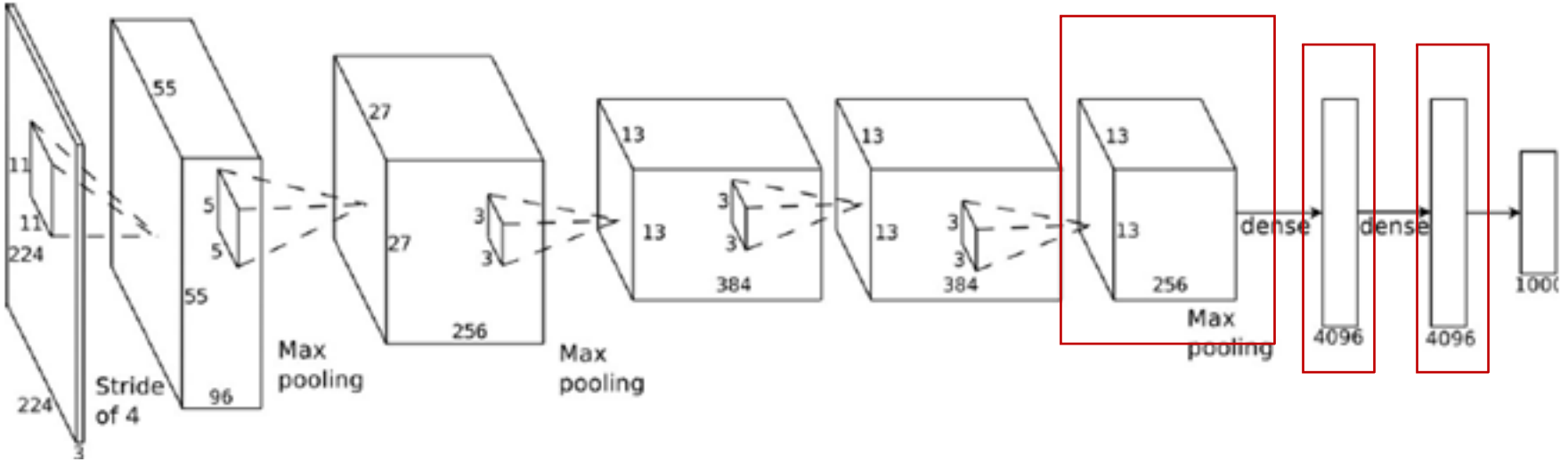
Какой объект на изображении определяет метку?



- Закрываем фрагмент изображения, вычисляем вероятность целевого класса
- Сканируем изображение и строим «heatmap» вероятности объекта целевого класса

http://cs231n.github.io/understanding-cnn/

Высокоуровневые признаки



- Эксперименты с визуализацией показали, что выходы всех слоёв, особенно верхних, можно использовать как хорошие признаки изображений
- Можно обучить сеть на одних данных (ImageNet) и применять её на других для вычисления признаков
- Выходы полносвязных слоёв содержат много «семантической» информации, но только о тех объектах, которые были в обучающей выборке

Donahue et. al.DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition, 2013

Распознавание на других базах

	$DeCAF_5$	$DeCAF_6$	$DeCAF_7$
LogReg	63.29 ± 6.6	84.30 ± 1.6	84.87 ± 0.6
LogReg with Dropout	-	86.08 ± 0.8	85.68 ± 0.6
SVM	77.12 ± 1.1	84.77 ± 1.2	83.24 ± 1.2
SVM with Dropout	-	$\mathbf{86.91 \pm 0.7}$	85.51 ± 0.9
Yang et al. (2009)		84.3	
Jarrett et al. (2009)		65.5	

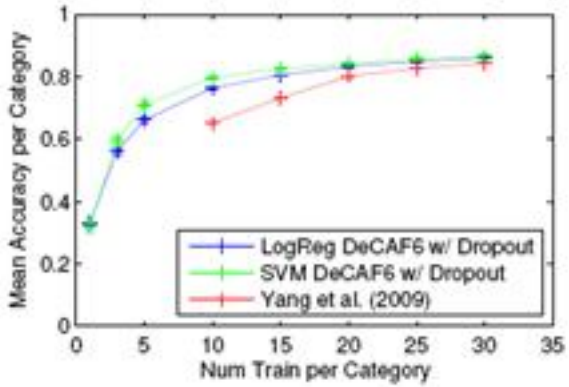


Figure 4. Left: average accuracy per class on Caltech-101 with 30 training samples per class across three hidden layers of the network and two classifiers. Our result from the training protocol/classifier combination with the best validation accuracy – SVM with Layer 6 (+ dropout) features – is shown in **bold**. Right: average accuracy per class on Caltech-101 at varying training set sizes.

Обучили нейросеть для извлечения признаков на ImageNet и применили для Caltech 101

Donahue et. al.DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition, 2013

Классификация близких объектов

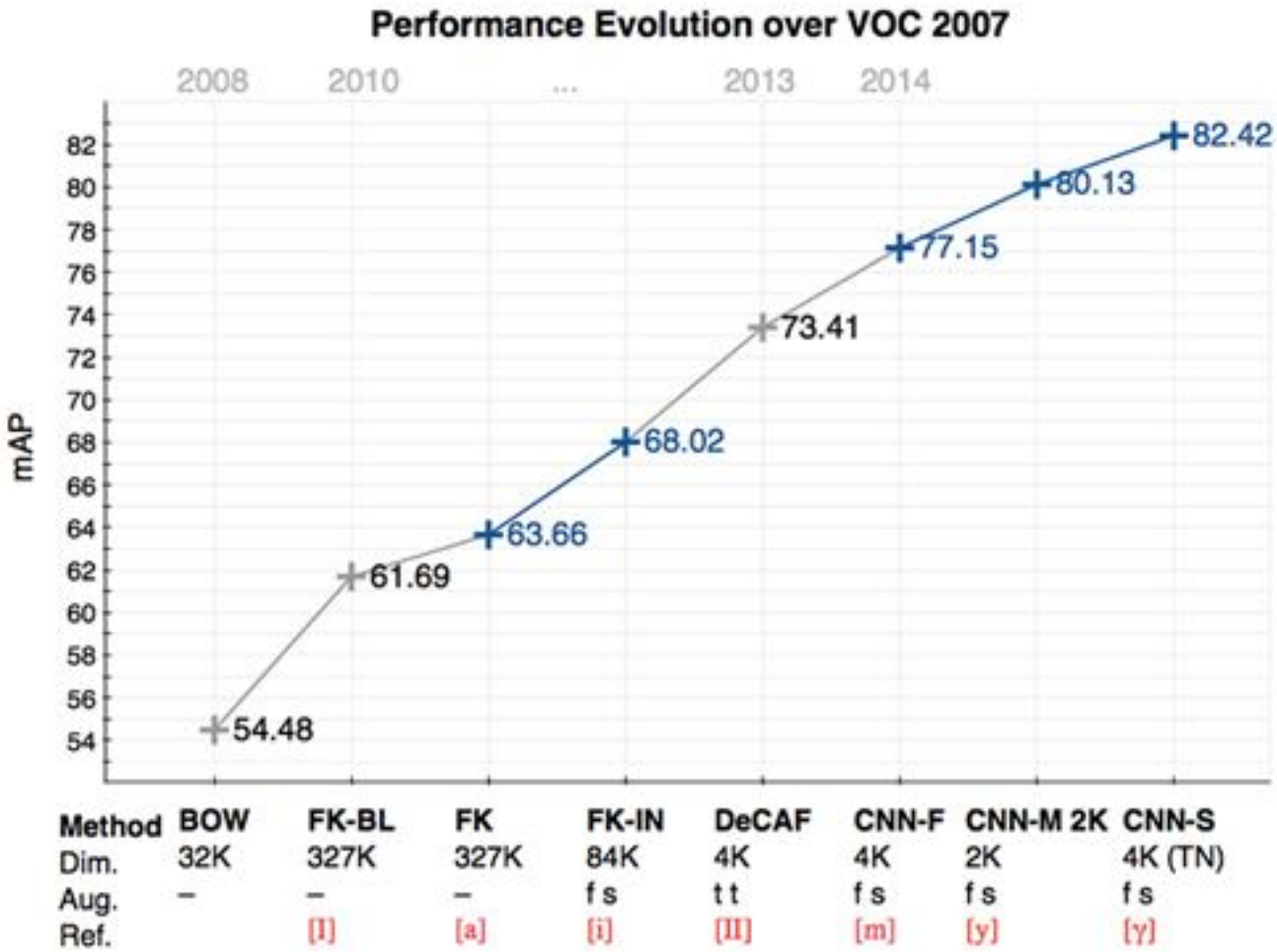
Method	Part info	mean Accuracy
Sift+Color+SVM[45]	✗	17.3
Pose pooling kernel[49]	✓	28.2
RF[47]	✓	19.2
DPD[50]	✓	51.0
Poof[5]	✓	56.8
CNN-SVM	✗	53.3
CNNaug-SVM	✗	**61.8**
DPD+CNN(DeCaf)+LogReg[10]	✓	**65.0**

Table 3: **Results on CUB 200-2011 Bird dataset.** The table distinguishes between methods which use part annotations for training and sometimes for evaluation as well and those that do not. [10] generates a pose-normalized CNN representation using DPD [50] detectors which significantly boosts the results to **64.96**.

- Fine-graded classification – например, определение видов птиц
- Возьмём нейросеть, обученную для классификации ImageNet
- Применим её для получения вектор-признаков изображений
  Обучаем классификатор поверх этих признаков
- Profit!

Ali Sharif Razavian Hossein Azizpour Josephine Sullivan Stefan Carlsson
CNN Features off-the-shelf: an Astounding Baseline for Recognition. 2014

Рост качества по методам

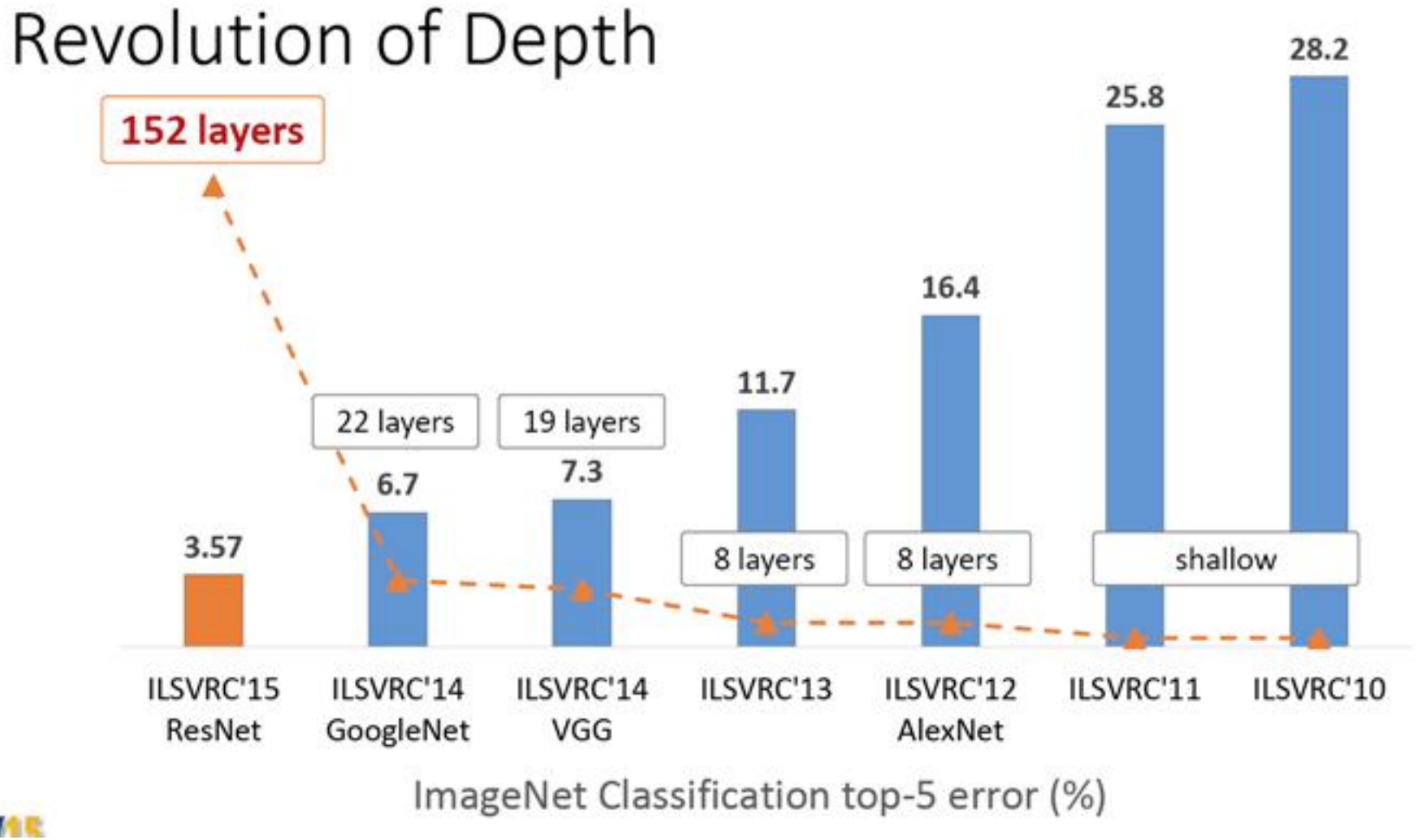


Активно ищут «правила» задания архитектур для повышения качества

Пока прогресс очень быстрый и предлагают много довольно простых решений

K. Chatfield et. Al. Return of the Devil in the Details: Delving Deep into Convolutional Nets, BMVC 2014

Рост глубины



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