



УНИВЕРСИТЕТ ИТМО

# Лекция 7. Свёрточные сети

Олег А. Евстафьев<sup>1</sup> Михаил А. Каканов<sup>1</sup>

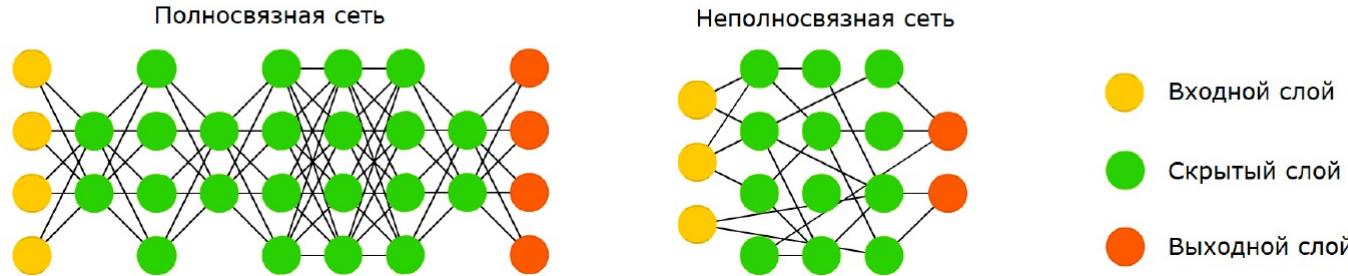
<sup>1</sup> Факультет систем управления и робототехники, Университет ИТМО  
[{oaevstafev, makakanov}@itmo.ru](mailto:{oaevstafev, makakanov}@itmo.ru)

- Свёрточная нейронная сеть – Convolutional Neural Network (CNN)
- Слой свёртки
- Слой подвыборки (pooling layer)
- Принципы CNN
- Локальное восприятие
- Уменьшение размерности
- Разделяемые веса
- Признаки в CNN
- Обучение CNN
- Основные особенности и виды архитектур CNN

# Глубокие нейронные сети (Deep Neural Network, DNN)

1965: первые глубокие нейронные сети

2012: свёрточная сеть для классификации изображений AlexNet



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

# Глубокие нейронные сети (Deep Neural Network, DNN)

Классический подход к распознаванию изображений:



Конструируемые  
признаки изображения

Любой обучаемый  
классификатор

Современный подход — end-to-end deep learning:



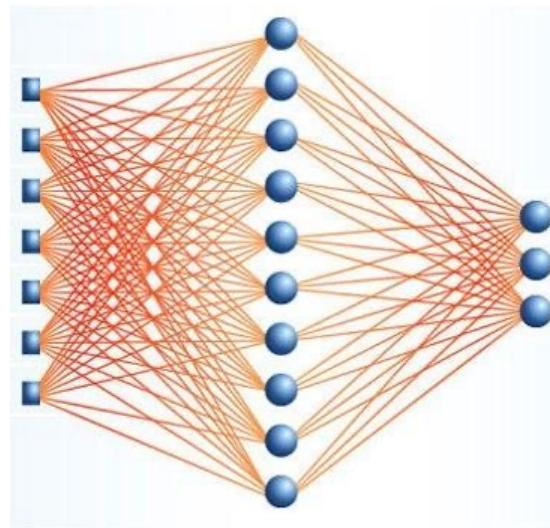
Обучаемые ANN  
признаки изображения

Обучаемый ANN  
классификатор

Совместное обучение

# Полносвязные нейронные сеть

Иногда называют «обычными» или «классическими» нейросетям.



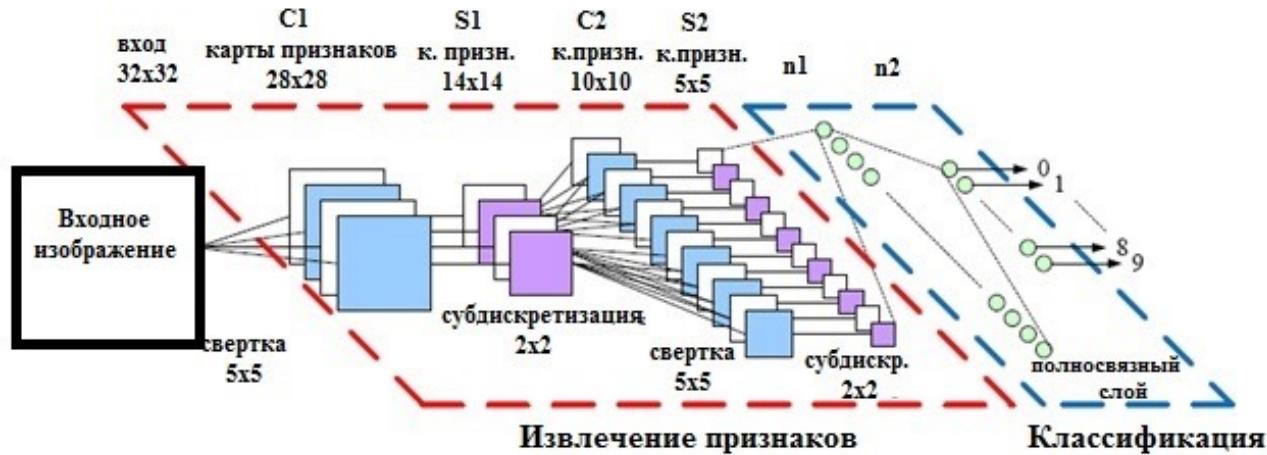
### Недостатки полносвязной многослойной сети для анализа изображений:

- Большое количество весов для обучения (изображения  $28 \times 28 - 784$  входа, 800 и 10 нейронов на 2 слоях – 635200 весов)
- Изображение представляется в виде плоского массива – теряется информация о топологии

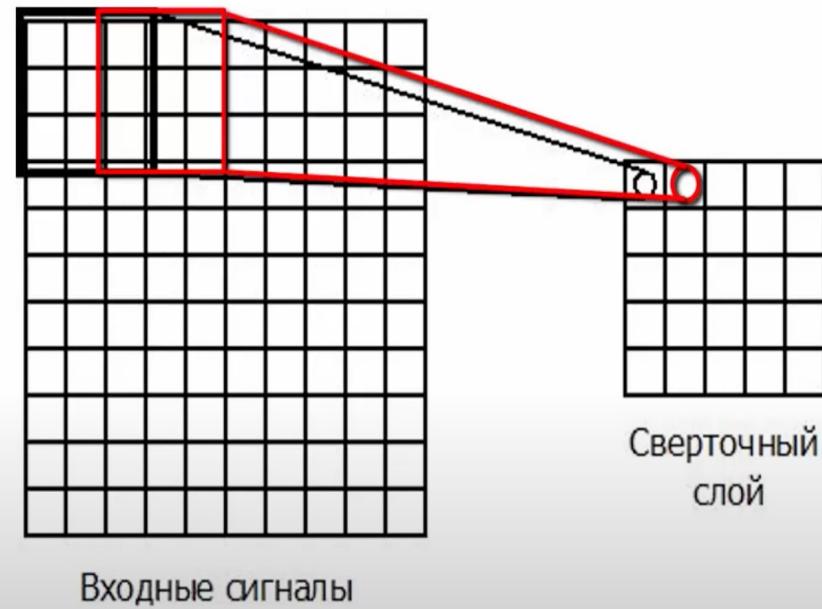
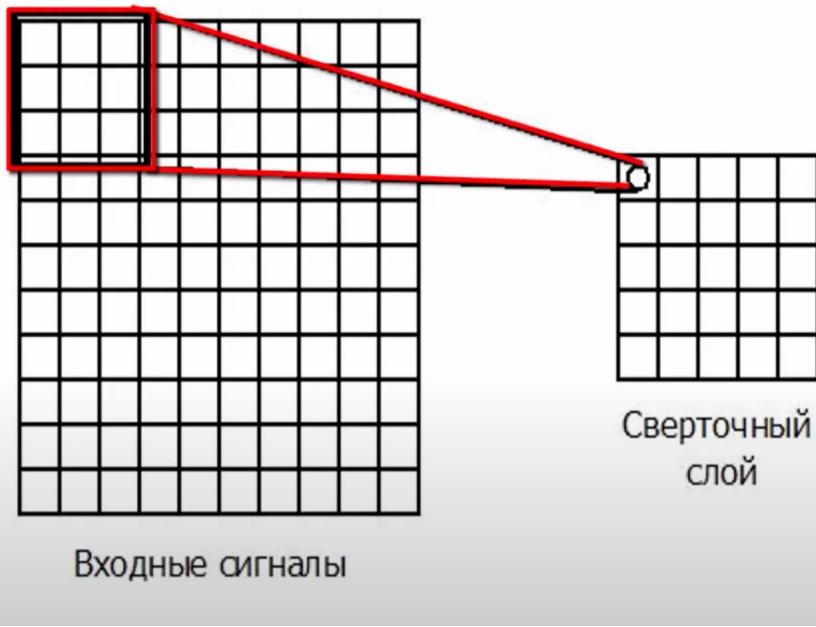
### Принципы сверточных нейронных сетей (convolutional neural networks):

- Локальное восприятие
- Разделяемые веса
- Уменьшение размерности

# Общая структура СИНС



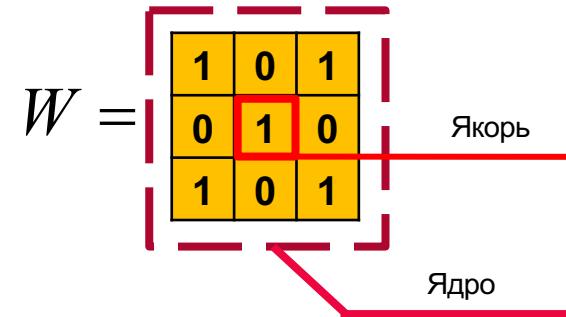
## Локальное восприятие



# АЛГОРИТМ СВЕРТКИ ИЗОБРАЖЕНИЯ

Пусть дано изображение в формате матрицы чисел  $X$  и квадратная матрица  $W$ :

$$X = \begin{array}{|c|c|c|c|c|} \hline 1 & 1 & 1 & 0 & 0 \\ \hline 0 & 1 & 1 & 1 & 0 \\ \hline 0 & 0 & 1 & 1 & 1 \\ \hline 0 & 0 & 1 & 1 & 0 \\ \hline 0 & 1 & 1 & 0 & 0 \\ \hline \end{array}$$



Тогда операция свертки определяется выражением

$$y_{i,j} = \sum_{s=1}^K \sum_{t=1}^K (w_{s,t} x_{((i-1)+s, (j-1)+t)}), \quad (1)$$

где  $w_{s,t}$  – это значение элемента ядра свертки в позиции  $(s,t)$ ,  $y_{i,j}$  – значение пикселя выходного изображения,  $x_{((i-1)+s, (j-1)+t)}$  – значение пикселя исходного изображения,  $K$  – размер ядра свертки.

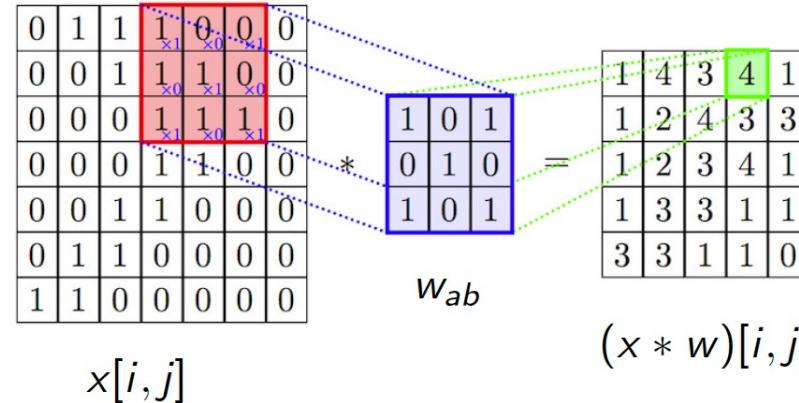
# АЛГОРИТМ СВЕРТКИ ИЗОБРАЖЕНИЯ

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

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

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

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



# АЛГОРИТМ СВЕРТКИ ИЗОБРАЖЕНИЯ

Пусть дано изображение в формате матрицы чисел  $X$  и квадратная матрица  $W$ :

|   |   |   |   |   |
|---|---|---|---|---|
| 1<br><small><math>\times 1</math></small> | 1<br><small><math>\times 0</math></small> | 1<br><small><math>\times 1</math></small> | 0 | 0 |
| 0<br><small><math>\times 0</math></small> | 1<br><small><math>\times 1</math></small> | 1<br><small><math>\times 0</math></small> | 1 | 0 |
| 0<br><small><math>\times 1</math></small> | 0<br><small><math>\times 0</math></small> | 1<br><small><math>\times 1</math></small> | 1 | 1 |
| 0   | 0   | 1   | 1 | 0 |
| 0   | 1   | 1   | 0 | 0 |

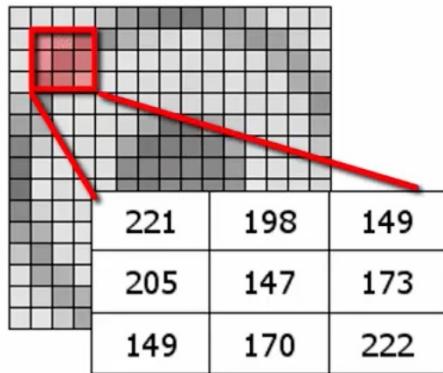
Изображение

|   |  |  |
|---|--|--|
| 4 |  |  |
|   |  |  |
|   |  |  |
|   |  |  |

Карта  
признаков

$$y_{i,j} = \sum_{s=1}^K \sum_{t=1}^K (w_{s,t} x_{((i-1)+s, (j-1)+t)}), \quad (1)$$

где  $w_{s,t}$  – это значение элемента ядра свертки в позиции  $(s,t)$ ,  $y_{i,j}$  – значение пикселя выходного изображения,  $x_{((i-1)+s, (j-1)+t)}$  – значение пикселя исходного изображения,  $K$  – размер ядра свертки.



Ядро  
свертки

|    |   |   |
|----|---|---|
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |

$$N(x,y) = 221 * (-1) + \\ 198 * 0 + \\ 149 * 1 + \\ 205 * (-2) + \\ 147 * 0 + \\ 173 * 2 + \\ 149 * (-1) + \\ 170 * 0 + \\ 222 * 1 = -63$$

## Размытие

|     |     |     |
|-----|-----|-----|
| 1/9 | 1/9 | 1/9 |
| 1/9 | 1/9 | 1/9 |
| 1/9 | 1/9 | 1/9 |

## Выделение границ

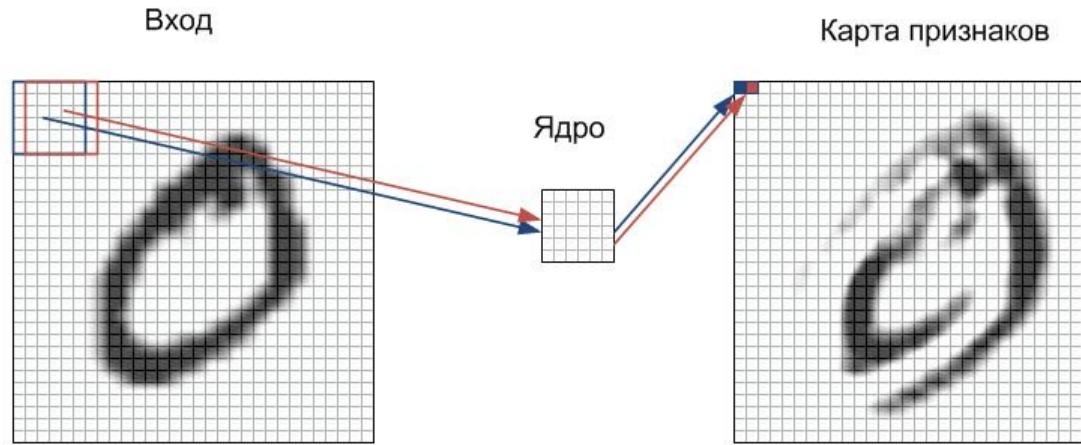
|    |    |    |
|----|----|----|
| 0  | -1 | 0  |
| -1 | 4  | -1 |
| 0  | -1 | 0  |

## Повышение четкости

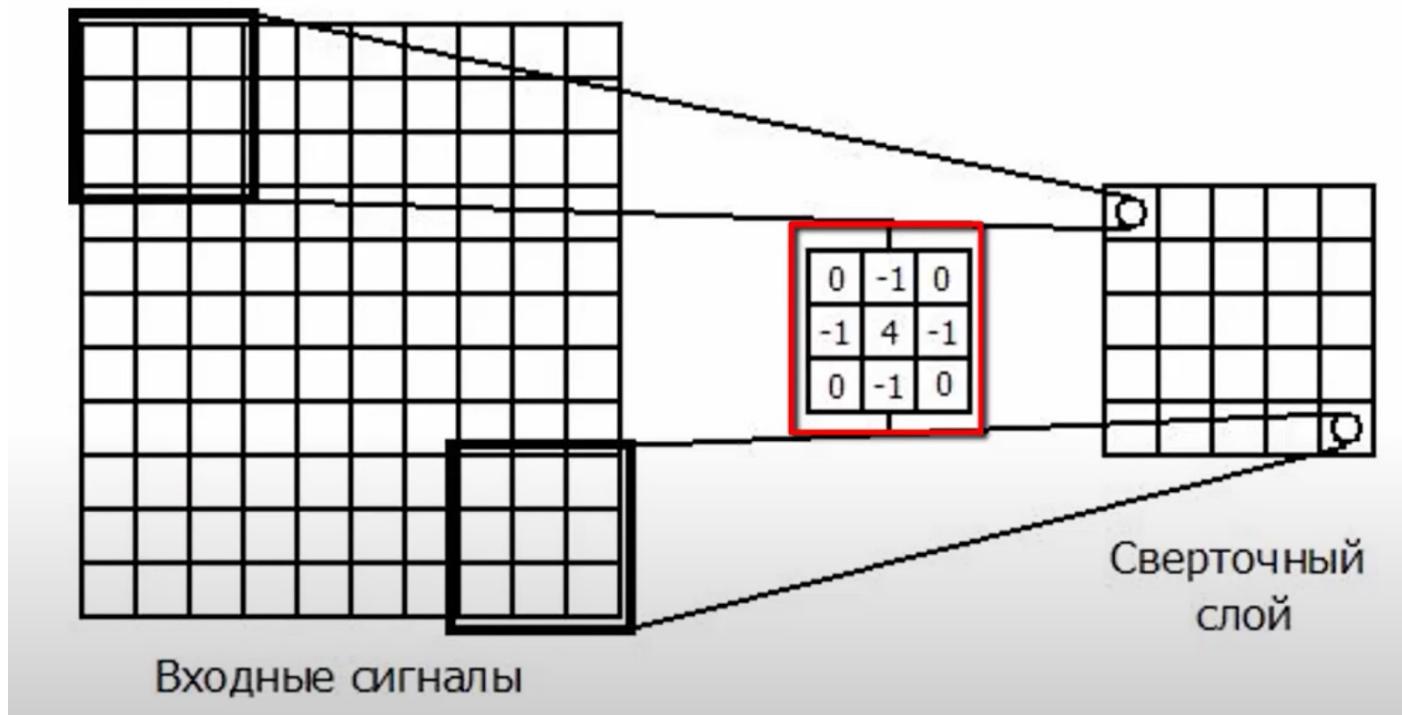
|    |    |    |
|----|----|----|
| 0  | -1 | 0  |
| -1 | 5  | -1 |
| 0  | -1 | 0  |

В нейронных сетях ядра свертки определяются автоматически в процессе обучения

# Процесс распространения сигнала в С-слое



# РАЗДЕЛЯЕМЫЕ ВЕСА

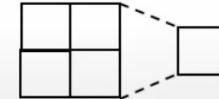


Распознавание объектов вне зависимости от масштаба

Факт наличия признака важнее знания места его точного положения на изображении

Слои подвыборки (subsampling):

- Усреднение
- Выбор максимального значения



# АЛГОРИТМ СУБДИСКРЕТИЗАЦИИ ИЗОБРАЖЕНИЯ

Пусть дано изображение в формате матрицы чисел  $X$  и установлено, что размеры окна субдискретизации –  $k \times k$ :

$$X = \begin{array}{|c|c|c|c|} \hline 1 & 3 & 2 & 0 \\ \hline 0 & 5 & 1 & 1 \\ \hline 3 & 0 & 1 & 4 \\ \hline 0 & 0 & 7 & 1 \\ \hline \end{array}$$

Тогда операция субдискретизации определяется выражением

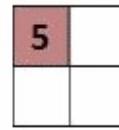
$$y_{(i,j)} = \max(x_{(ik+s, jk+t)}), \quad (2)$$

где  $y_{(i,j)}$  – это значение пикселя выходного изображения,  $x_{(ik+s, jk+t)}$  – значение пикселя исходного изображения.

# АЛГОРИТМ СУБДИСКРЕТИЗАЦИИ ИЗОБРАЖЕНИЯ

Пусть дано изображение в формате матрицы чисел  $X$  и установлено, что размеры окна субдискретизации –  $k \times k$ :

|   |   |   |   |
|---|---|---|---|
| 1 | 3 | 2 | 0 |
| 0 | 5 | 1 | 1 |
| 3 | 0 | 1 | 4 |
| 0 | 0 | 7 | 1 |



Тогда операция субдискретизации определяется выражением

$$y_{(i,j)} = \max(x_{(ik+s, jk+t)}), \quad (2)$$

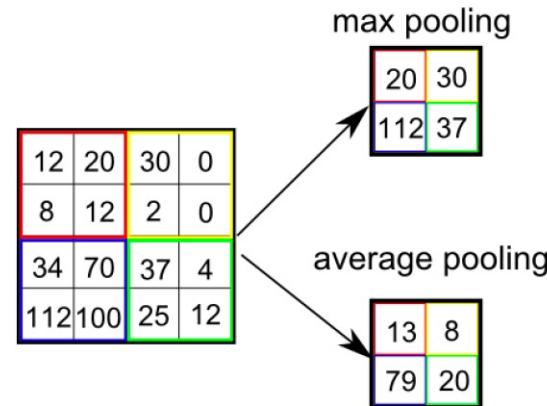
где  $y_{(i,j)}$  – это значение пикселя выходного изображения,  $x_{(ik+s, jk+t)}$  – значение пикселя исходного изображения.

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

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

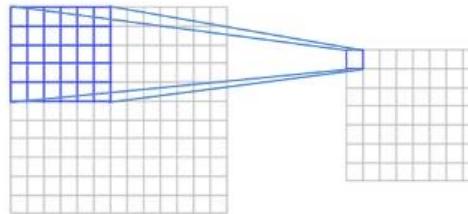
где  $F$  — агрегирующая функция: max, average и т.п.

max-pooling позволяет обнаружить элемент в любой из ячеек

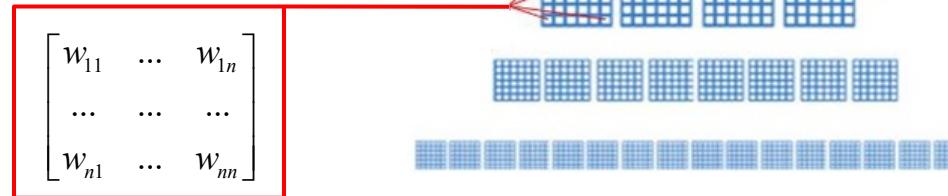


Такая архитектура заключает в себе 3 основных парадигмы:

1. *Локальное восприятие:*



2. *Разделяемые веса:*



2. *Субдискретизация:*

За счет этого повышается устойчивость входного сигнала  
к смещениям и незначительным деформациям

# ПАРАДИГМЫ СИНС

Каждый фрагмент изображения поэлементно умножается на небольшую матрицу весов (ядро), результат суммируется. Эта сумма является пикселом выходного изображения, которое называется картой признаков. Здесь я опустил тот факт, что взвешенная сумма входов еще пропускается через функцию активации (как в любой другой нейросети). На самом деле это может происходить и в S-слое, принципиальной разницы нет. Следует сказать, что в идеале не разные фрагменты проходят последовательно через ядро, а параллельно все изображение проходит через идентичные ядра. Кроме того, количество ядер (наборов весов) определяется разработчиком и зависит от того какое количество признаков необходимо выделить. Еще одна особенность сверточного слоя в том, что он немного уменьшает изображение за счет краевых эффектов.

Суть субдискретизации и S-слоев заключается в уменьшении пространственной размерности изображения. Т.е. входное изображение грубо (усреднением) уменьшается в заданное количество раз. Чаще всего в 2 раза, хотя может быть и не равномерное изменение, например, 2 по вертикали и 3 по горизонтали. Субдискретизация нужна для обеспечения инвариантности к масштабу.

Чередование слоев позволяет составлять карты признаков из карт признаков, что на практике означает способность распознавания сложных иерархий признаков.

Обычно после прохождения нескольких слоев карта признаков вырождается в вектор или даже скаляр, но таких карт признаков становится сотни. В таком виде они подаются на один-два слоя полно связной сети. Выходной слой такой сети может иметь различные функции активации. В простейшем случае это может быть тангенциальная функция, также успешно используются радиальные базисные функции.

# МЕТОД ОБРАТНОГО РАСПРОСТРАНЕНИЯ ОШИБКИ

При реализации обратного распространения ошибки выходного слоя выбрана следующим образом

$$E_n^p = \frac{1}{2} \sum_{k=1}^M (x_k - d_k)^2 \rightarrow \min, \quad (3)$$

где  $M$  — количество нейронов выходного слоя,  $k$  — номер выходного нейрона,  $x_k$  — реальное значение выходного сигнала нейрона,  $d_k$  — ожидаемое значение.

Для обучения сети был выбран стандартный алгоритм градиентного спуска. Коррекция весов осуществляется по формуле

$$(\omega_n^{ij})_{new} = (\omega_n^{ij})_{old} - \eta \cdot \left( \frac{\partial E_n^p}{\partial \omega_n^{ij}} \right), \quad (4)$$

где  $(\omega_n^{ij})_{new}$  — значение весов после коррекции,  $(\omega_n^{ij})_{old}$  — значение весов до коррекции,  $\eta$  характеризует скорость обучения.

$$\frac{\partial E_n^p}{\partial \omega_n^{ij}} = x_{n-1}^j \cdot \frac{\partial E_n^p}{\partial y_n^i}, \quad (5)$$

где  $x_{n-1}^j$  — выход  $j$ -го нейрона  $(n-1)$ -го слоя,  $y_n^i$  — скалярное произведение всех выходов нейронов  $(n-1)$ -го слоя и соответствующих весовых коэффициентов.

$$\frac{\partial E_n^p}{\partial y_n^i} = G(x_n^j) \cdot \frac{\partial E_n^p}{\partial x_n^i}, \quad (6)$$

где  $G(x_n^i)$  — производная функции активации, а  $\frac{\partial E_n^p}{\partial x_n^i} = x_n^i - d_n^i$ .

Ошибка необходимо распространить на предыдущий слой.  
Это делается по следующей формуле

$$\frac{\partial E_{n-1}^p}{\partial x_{n-1}^k} = \sum_i \omega_n^{ik} \cdot \frac{\partial E_n^p}{\partial y_n^i}. \quad (7)$$

Для того чтобы можно было начать обучение нашей сети нужно определиться с тем, как измерять качество распознавания. В нашем случае для этого будем использовать самую распространенную в теории нейронных сетей функцию среднеквадратической ошибки (СКО, MSE) [3]:

$$E^p = \frac{1}{2} (D^p - O(I^p, W))^2$$

В этой формуле  $E^p$  — это ошибка распознавания для  $p$ -ой обучающей пары,  $D^p$  — желаемый выход сети,  $O(I^p, W)$  — выход сети, зависящий от  $p$ -го входа и весовых коэффициентов  $W$ , куда входят ядра свертки, смещения, весовые коэффициенты  $S$ - и  $F$ -слоев. Задача обучения так настроить веса  $W$ , чтобы они для любой обучающей пары  $(I^p, D^p)$  давали минимальную ошибку  $E^p$ . Чтобы посчитать ошибку для всей обучающей выборки просто берется среднее арифметическое по ошибкам для всех обучающих пар. Такую усредненную ошибку обозначим как  $E$ .

Для минимизации функции ошибки  $E^p$  самыми эффективными являются градиентные методы. Рассмотрим суть градиентных методов на примере простейшего одномерного случая (т.е. когда у нас всего один вес). Если мы разложим в ряд Тейлора функцию ошибки  $E^p$ , то получим следующее выражение:

$$E(W) = E(W_c) + (W - W_c) \frac{dE(W_c)}{dW} + \frac{1}{2}(W - W_c)^2 \frac{d^2E(W_c)}{dW^2} + \dots$$

Здесь  $E$  — все та же функция ошибки,  $W_c$  — некоторое начальное значение веса.

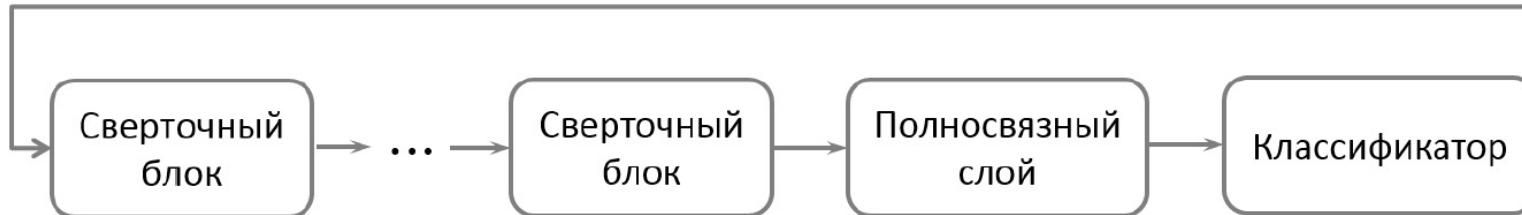
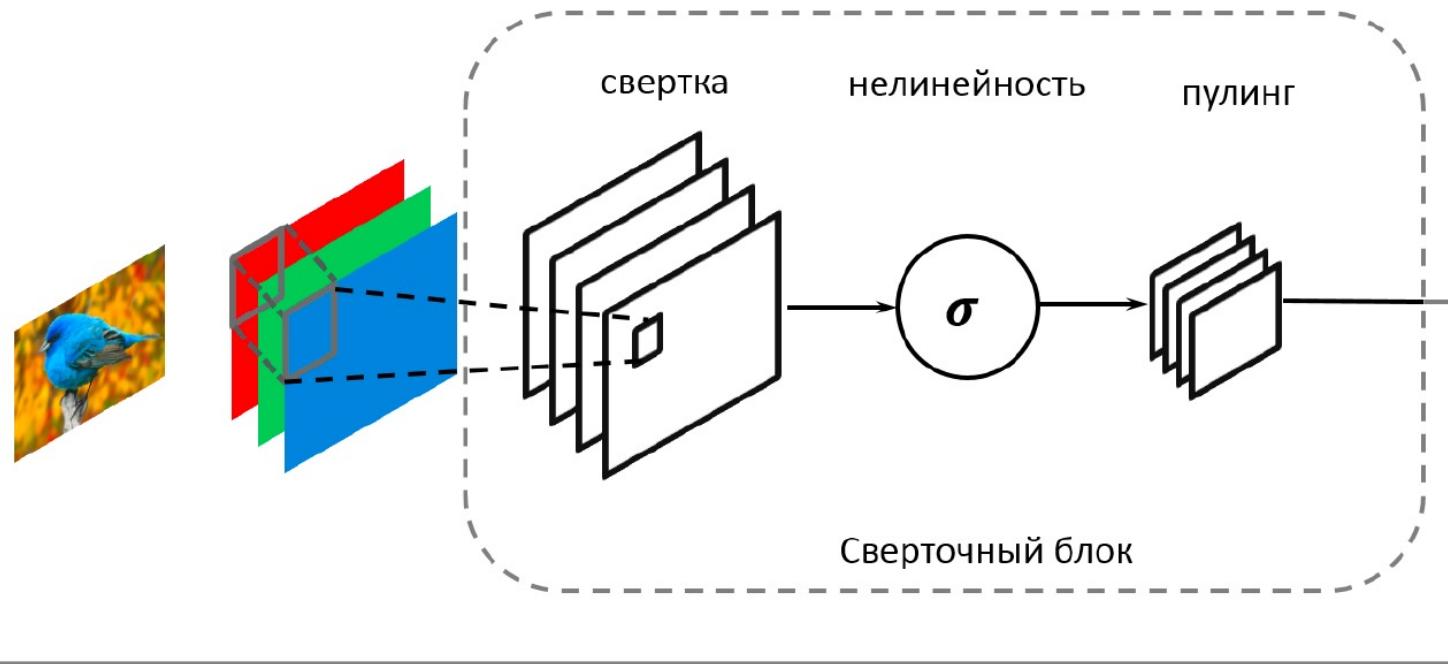
Из школьной математики мы помним, что для нахождения экстремума функции необходимо взять ее производную и приравнять нулю. Так и поступим, возьмем производную функции ошибки по весам, отбросив члены выше 2го порядка:

$$\frac{dE(W)}{dW} = \frac{dE(W_c)}{dW} + (W - W_c) \frac{d^2E(W_c)}{dW^2}$$

из этого выражения следует, что вес, при котором значение функции ошибки будет минимальным можно вычислить из следующего выражения:

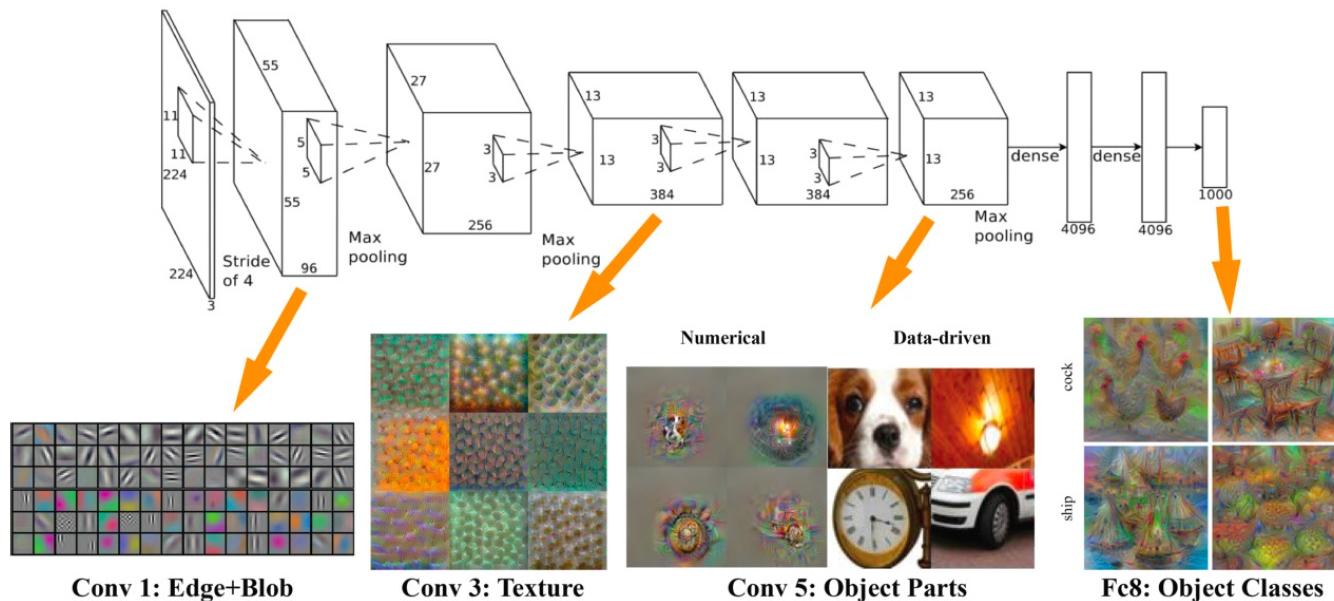
$$W_{min} = W_c - \left( \frac{d^2E(W_c)}{dW^2} \right)^{-1} \frac{dE(W_c)}{dW}$$

# Стандартная схема сверточной сети

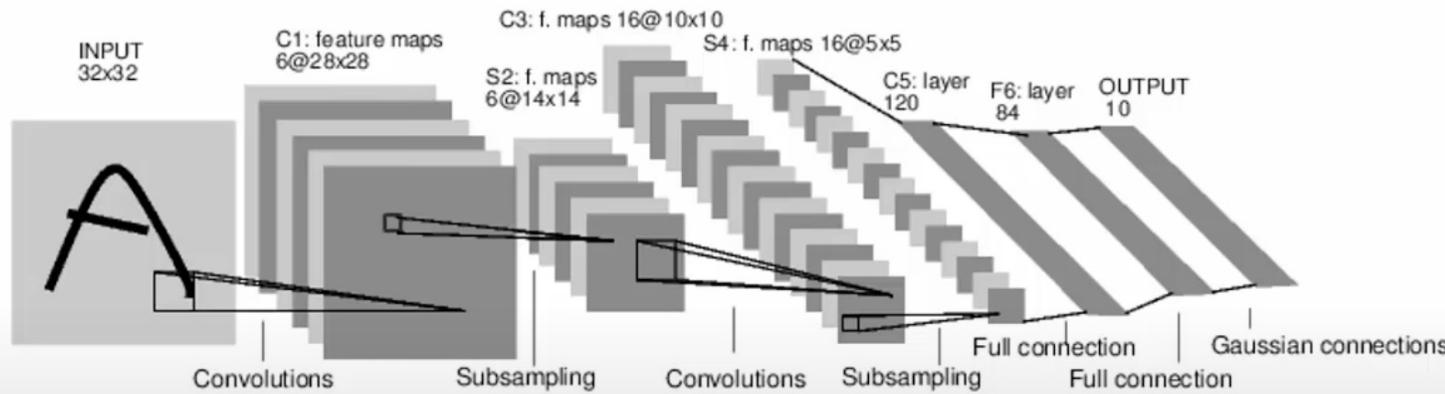


# Сверточная сеть извлечение признаков

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

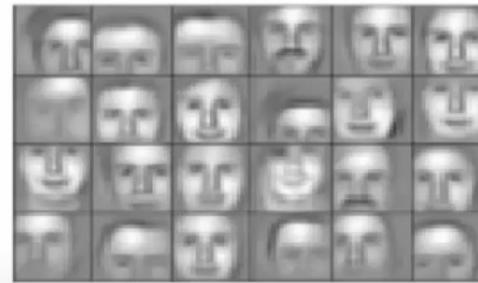
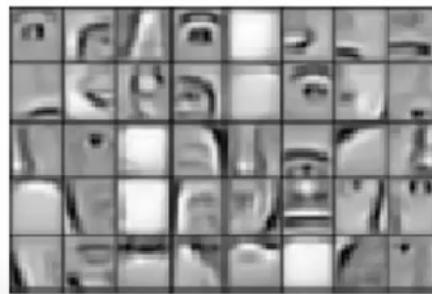
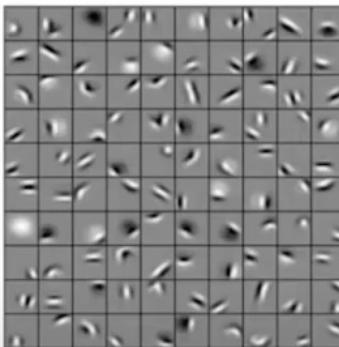


# Сверточная сеть LeNet-5



Back-Propagation Applied to Handwritten Zip Code Recognition / Y. LeCun,  
B. Boser, J. S. Denker et al. 1989

# Сверточная сеть для распознавания лиц



Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y. Ng.  
Unsupervised Learning of Hierarchical Representations with  
Convolutional Deep Belief Networks (2011)

# ImageNet – большая выборка размеченных изображений



flamingo



cock



ruffed grouse



quail



partridge

..



Egyptian cat



Persian cat



Siamese cat



tabby



lynx

..



dalmatian



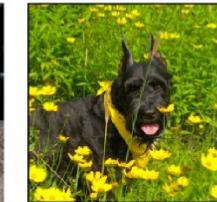
keeshond



miniature schnauzer standard schnauzer giant schnauzer



standard schnauzer

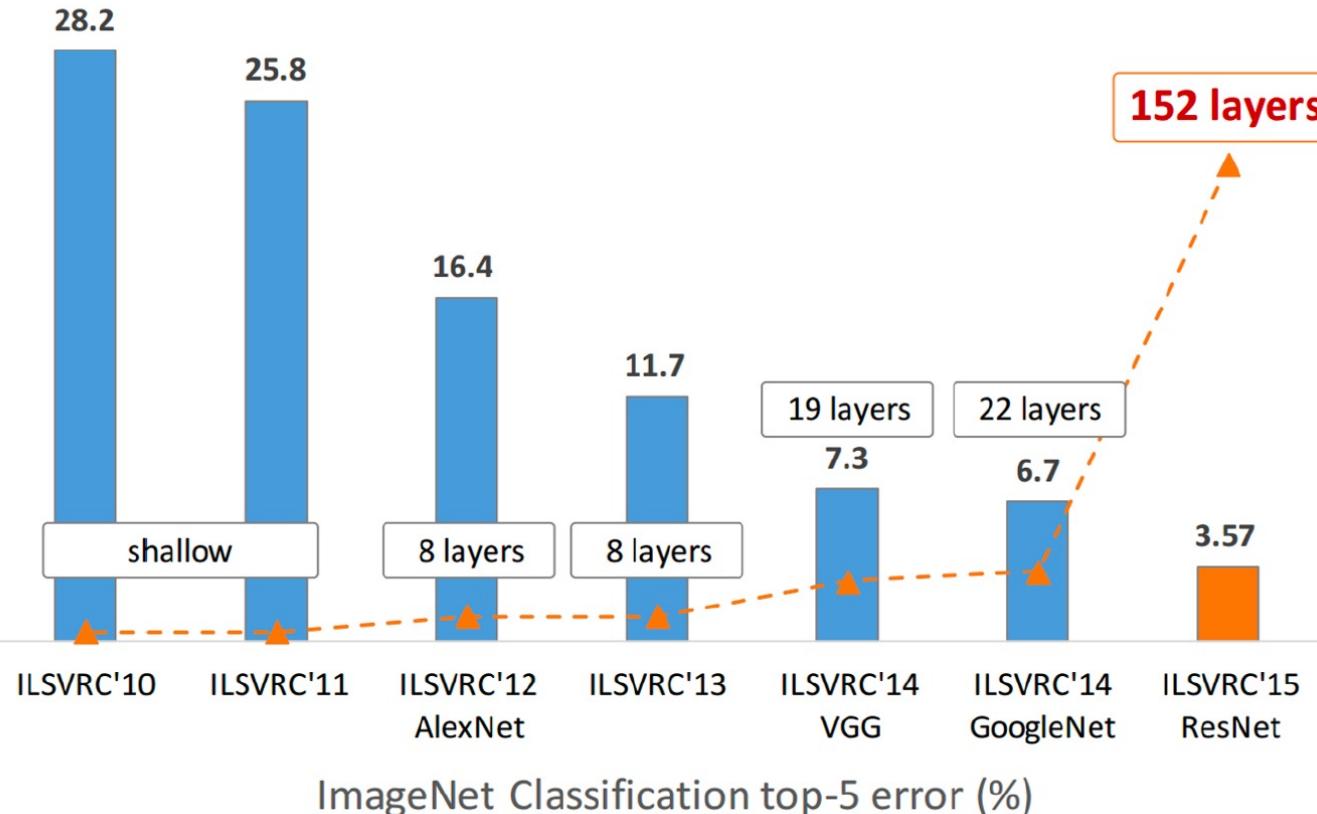


giant schnauzer

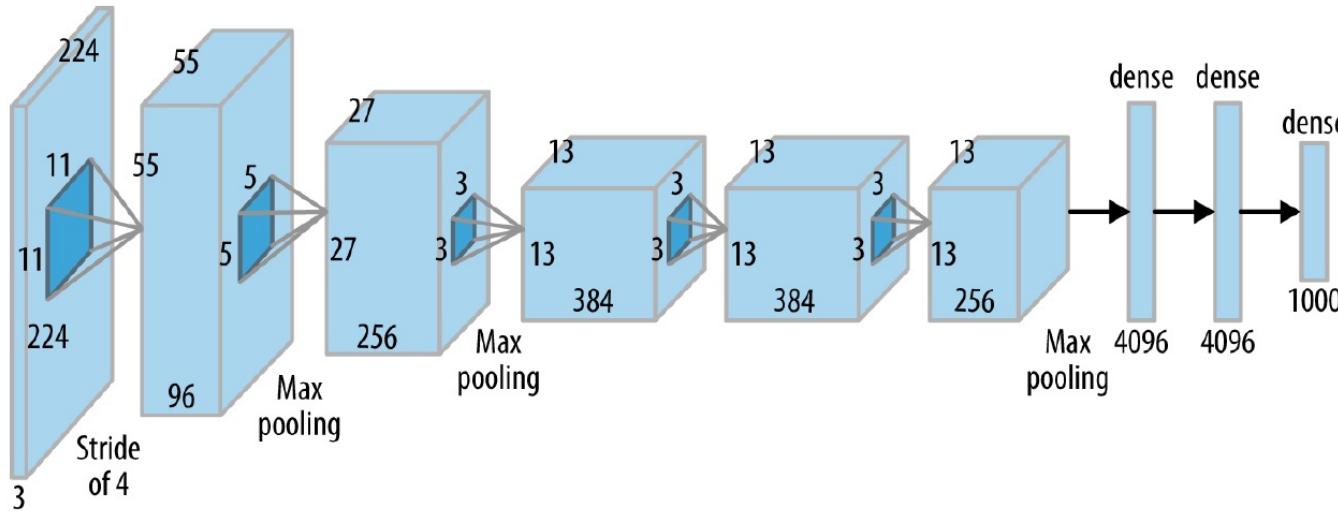
*Li Fei-Fei et al. ImageNet: A large-scale hierarchical image database. 2009.*

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

# Развитие свёрточных сетей



# AlexNet: первый прорыв на ImageNet



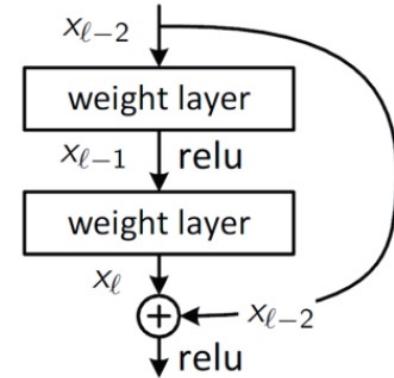
- ReLU + Dropout + пополнение выборки
- 60 млн параметров (в основном в полно связанных слоях)
- Подбор размеров фильтров и пулинга
- GPU

# ResNet: остаточная нейронная сеть (Residual NN)

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

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

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

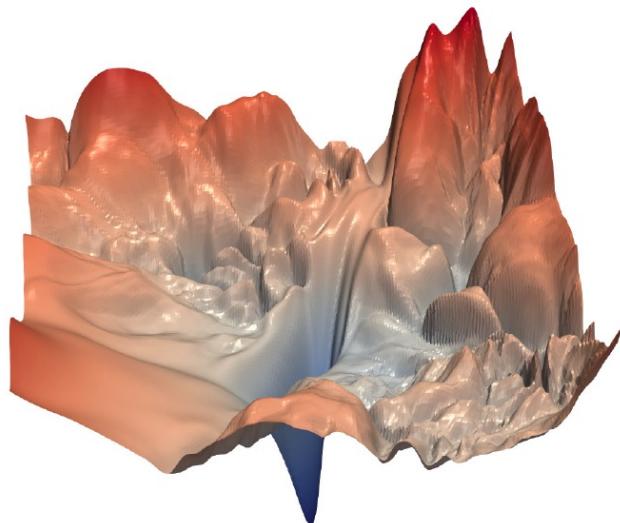


- Приращения более устойчивы  $\Rightarrow$  улучшается сходимость
- Появляется возможность увеличивать число слоёв
- Обобщение — Highway Networks:

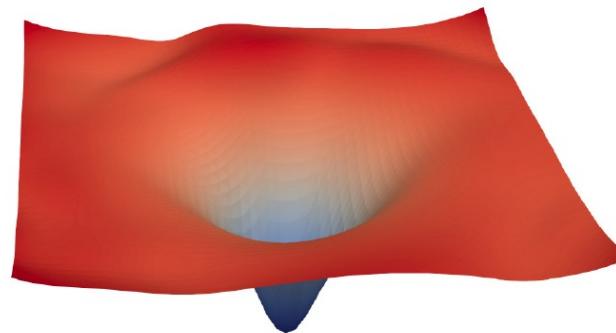
$$x_\ell = \sigma(Wx_{\ell-1}) \underbrace{\tau(W'x_{\ell-1})}_{\text{transform gate}} + x_{\ell-d} \underbrace{(1 - \tau(W'x_{\ell-1}))}_{\text{carry gate}}$$

# ResNet: визуализация оптимального критерия

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



without skip connections



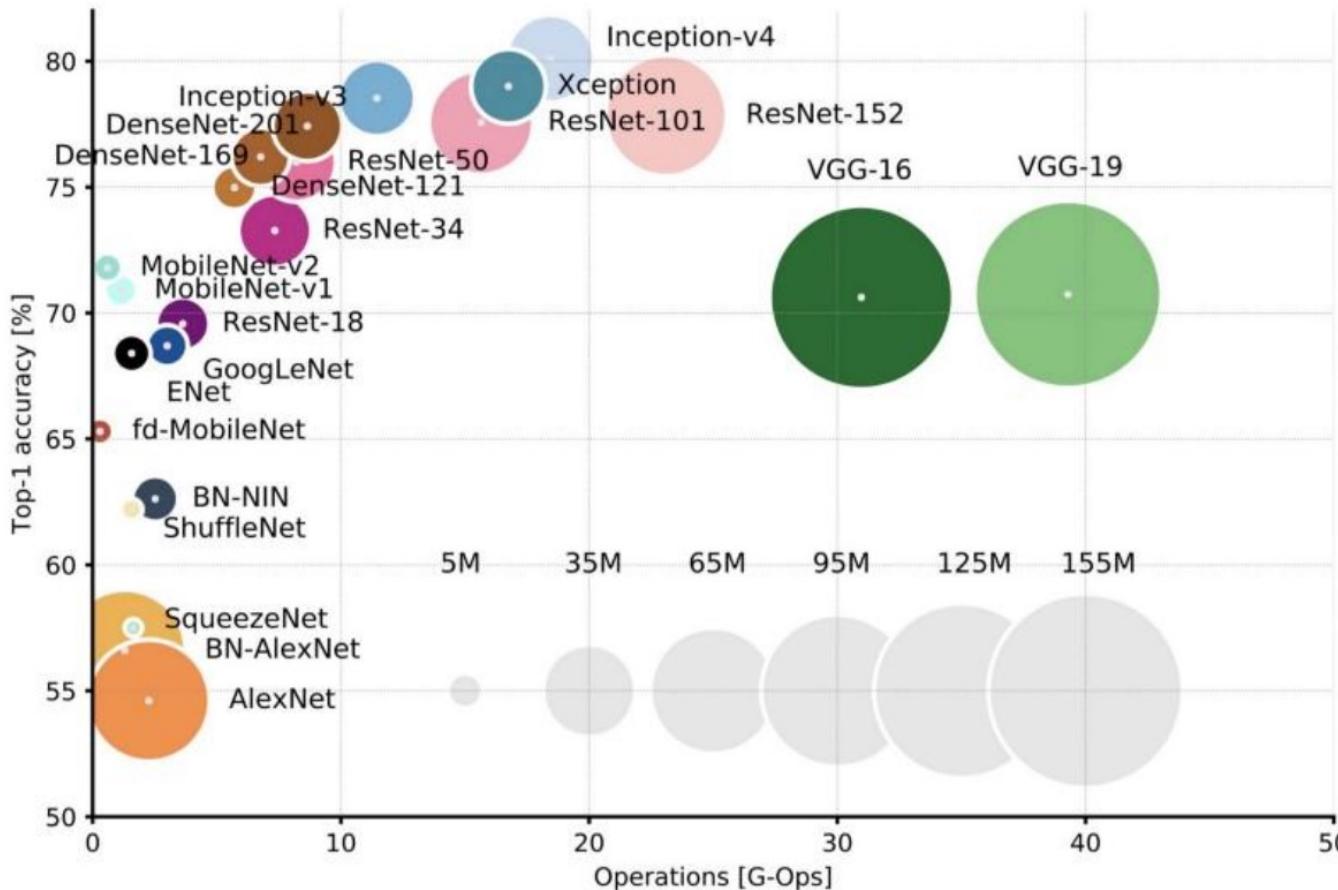
with skip connections

# Часто используемые приёмы в CNN

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



# CNN – Виды архитектур



## LeNet5

Особенности LeNet5:

Свёрточная нейросеть, использующая последовательность из трёх слоёв: слои свёртки (convolution), слои группирования (pooling) и слои нелинейности (non-linearity) → с момента публикации работы Лекуна это, пожалуй, одна из главных особенностей глубокого обучения применительно к изображениям.

- Использует свёртку для извлечения пространственных свойств.
- Подвыборка с использованием пространственного усреднения карт.
- Нелинейность в виде гиперболического тангенса или сигмоид.
- Финальный классификатор в виде многослойной нейросети (MLP).
- Разреженная матрица связности между слоями позволяет уменьшить объём вычислений.

## Dan Ciresan Net

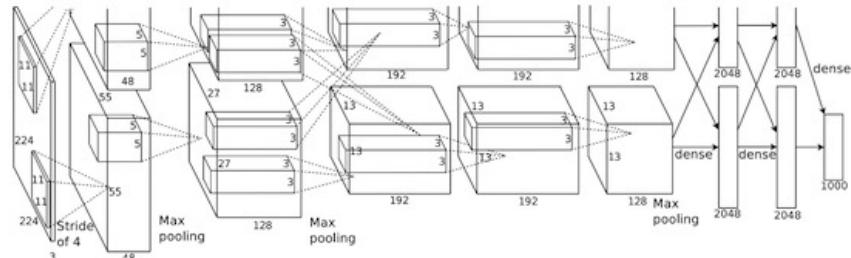
В 2010-м Дан Кирешан (Dan Claudiu Ciresan) и Йорген Шмидхубер (Jurgen Schmidhuber) опубликовали одно из первых описаний реализации GPU-нейросетей. Их работа содержала прямую и обратную реализацию 9-слойной нейросети на NVIDIA GTX 280.

## AlexNet

- Особенности этого решения:

Использование блоков линейной ректификации (ReLU) в качестве нелинейностей.

- Использование методики отбрасывания для выборочного игнорирования отдельных нейронов в ходе обучения, что позволяет избежать переобучения модели.
- Перекрытие max pooling, что позволяет избежать эффектов усреднения average pooling.
- Использование NVIDIA GTX 580 для ускорения обучения.



## Overfeat

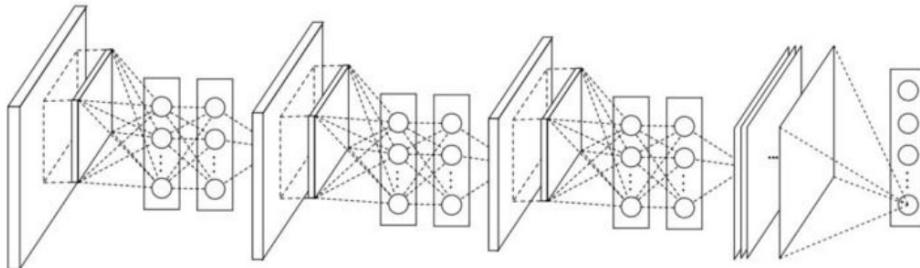
В декабре 2013-го лаборатория NYU Яна Лекуна опубликовала описание Overfeat, разновидности AlexNet. Также в статье описывались обучаемые bounding boxes, и впоследствии вышло много других работ по этой тематике. Мы считаем, что лучше научиться сегментировать объекты, а не использовать искусственные bounding boxes.

## VGG

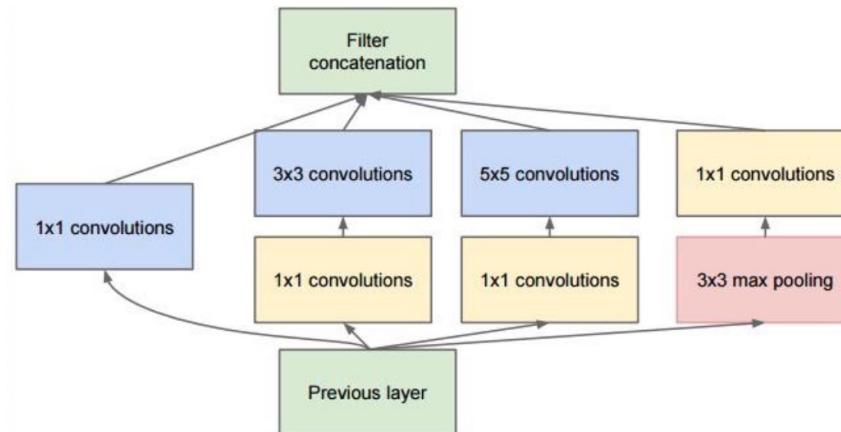
| ConvNet Configuration               |                               |                             |  |  |   |
|-------------------------------------|-------------------------------|-----------------------------|--|--|---|
| A                                   | A-LRN                         | B                           | C  | D  | E   |
| 11 weight layers                    | 11 weight layers              | 13 weight layers            | 16 weight layers                           | 16 weight layers                           | 19 weight layers  |
| input ( $224 \times 224$ RGB image) |                               |                             |  |  |   |
| conv3-64                            | conv3-64<br><b>LRN</b>        | conv3-64<br><b>conv3-64</b> | conv3-64<br>conv3-64                       | conv3-64<br>conv3-64                       | conv3-64<br>conv3-64                                    |
| maxpool                             |                               |                             |  |  |   |
| conv3-128                           | conv3-128<br><b>conv3-128</b> | conv3-128<br>conv3-128      | conv3-128<br>conv3-128                     | conv3-128<br>conv3-128                     | conv3-128<br>conv3-128                                  |
| maxpool                             |                               |                             |  |  |   |
| conv3-256<br>conv3-256              | conv3-256<br>conv3-256        | conv3-256<br>conv3-256      | conv3-256<br>conv3-256<br><b>conv1-256</b> | conv3-256<br>conv3-256<br><b>conv3-256</b> | conv3-256<br>conv3-256<br>conv3-256<br><b>conv3-256</b> |
| maxpool                             |                               |                             |  |  |   |
| conv3-512<br>conv3-512              | conv3-512<br>conv3-512        | conv3-512<br>conv3-512      | conv3-512<br>conv3-512<br><b>conv1-512</b> | conv3-512<br>conv3-512<br><b>conv3-512</b> | conv3-512<br>conv3-512<br>conv3-512<br><b>conv3-512</b> |
| maxpool                             |                               |                             |  |  |   |
| conv3-512<br>conv3-512              | conv3-512<br>conv3-512        | conv3-512<br>conv3-512      | conv3-512<br>conv3-512<br><b>conv1-512</b> | conv3-512<br>conv3-512<br><b>conv3-512</b> | conv3-512<br>conv3-512<br>conv3-512<br><b>conv3-512</b> |
| maxpool                             |                               |                             |  |  |   |
| FC-4096                             |                               |                             |  |  |   |
| FC-4096                             |                               |                             |  |  |   |
| FC-1000                             |                               |                             |  |  |   |
| soft-max                            |                               |                             |  |  |   |

# CNN – Виды архитектур

## Network-in-network



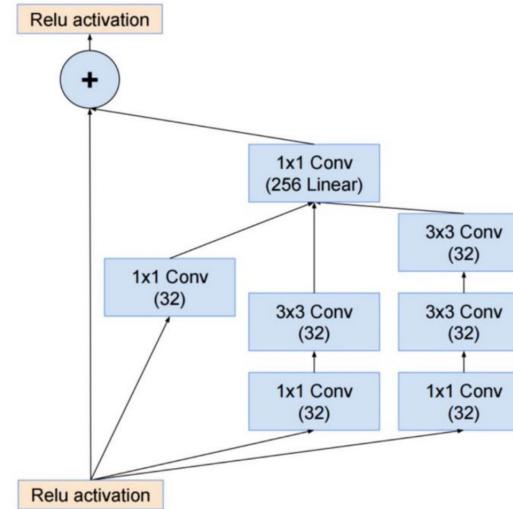
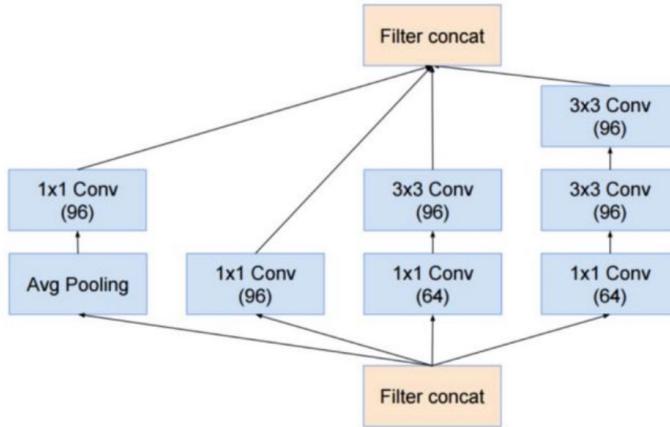
## GoogLeNet и Inception



## Inception V3 (V2)

- Максимизация потока информации в сети за счёт аккуратного баланса между её глубиной и шириной. Перед каждым pooling-ом увеличиваются карты свойств.
- С увеличением глубины также систематически увеличивается количество свойств или ширина слоя.
- Ширина каждого слоя увеличивается ради увеличения комбинации свойств перед следующим слоем.
- По мере возможности используются только свёртки 3x3. Учитывая, что фильтры 5x5 и 7x7 можно декомпозировать с помощью нескольких 3x3

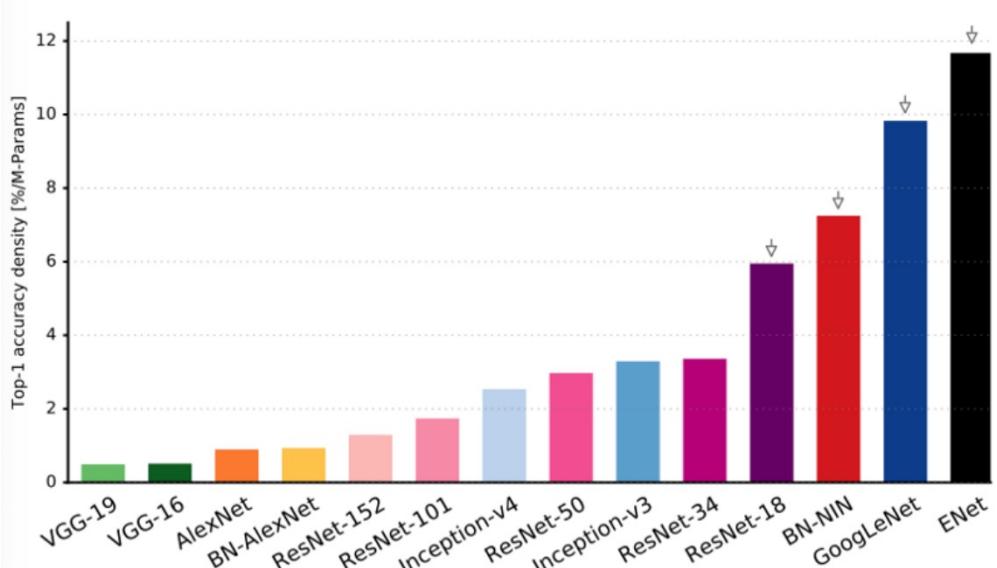
## Inception V4



## SqueezeNet

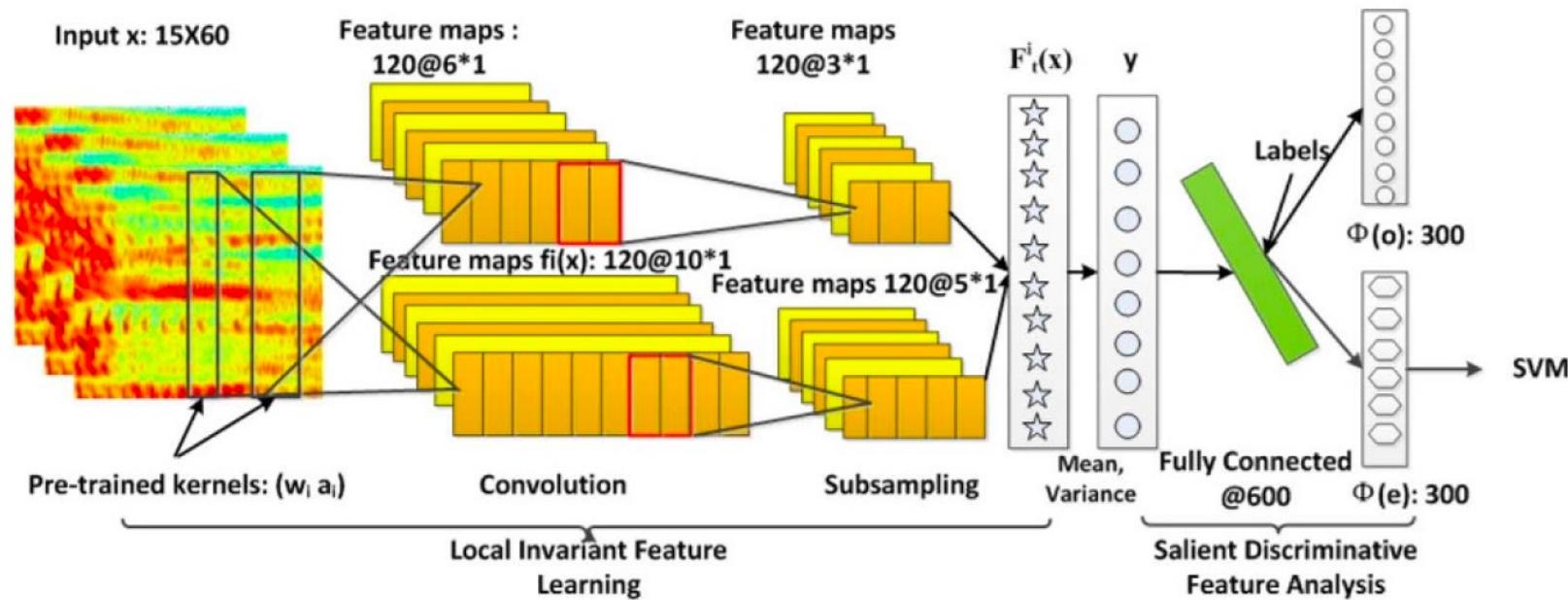
Переделка на новый лад многих концепций из ResNet и Inception. Авторы продемонстрировали, что улучшение архитектуры позволяет уменьшить размеры сетей и количество параметров без сложных алгоритмов сжатия.

## ENet



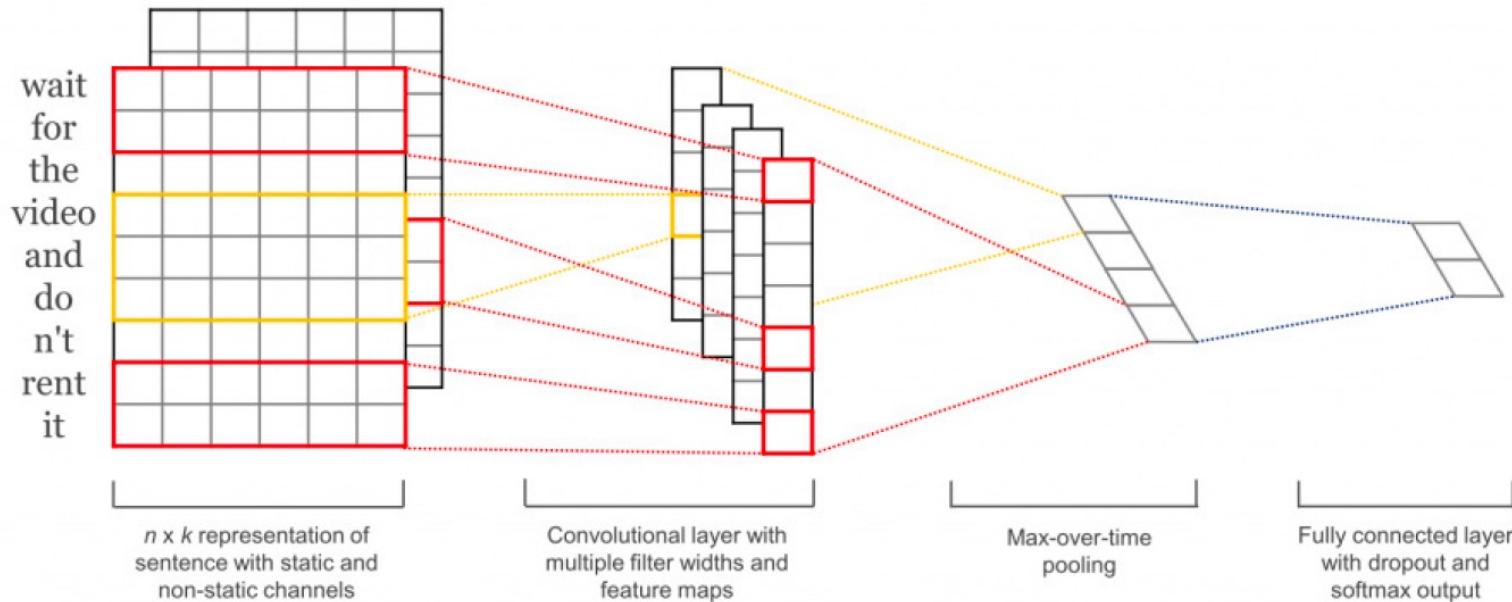
# CNN – Распознавание речевых сигналов

Последовательные фрагменты сигнала представляются векторами спектрального разложения

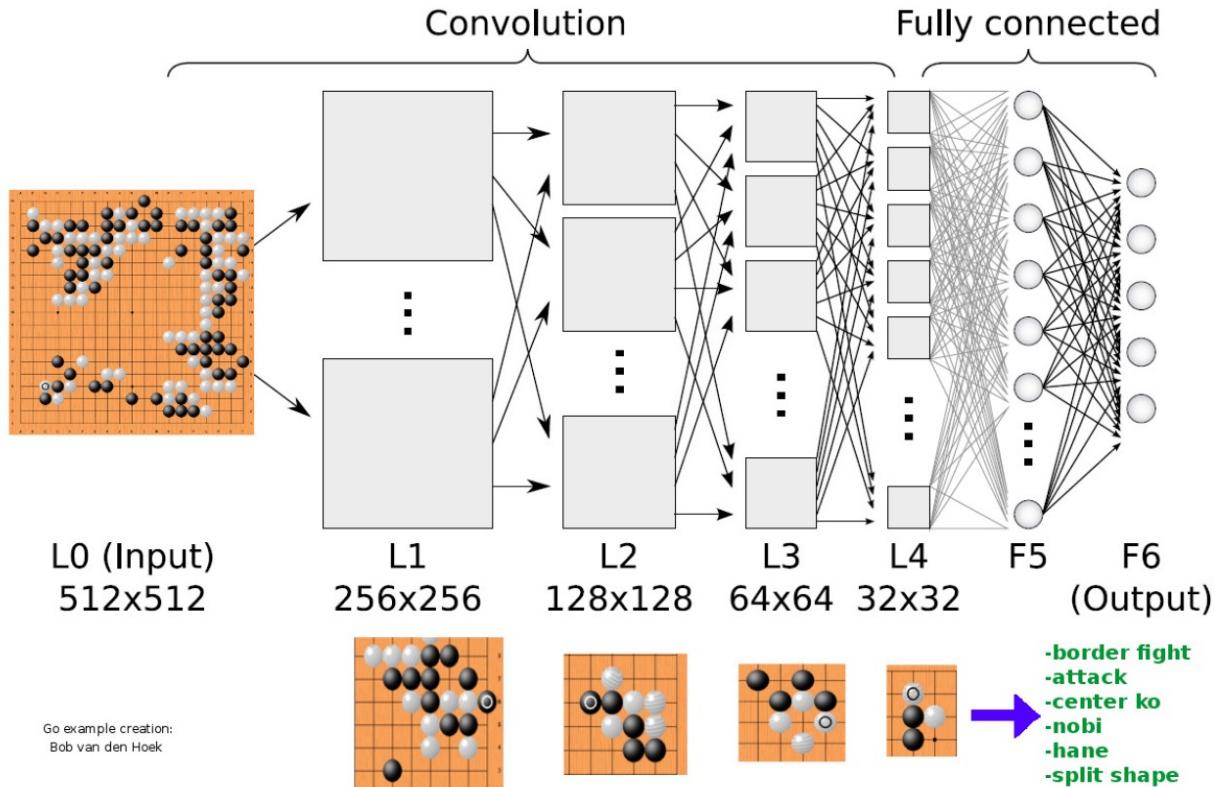


# CNN – Классификация предложений в тексте

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

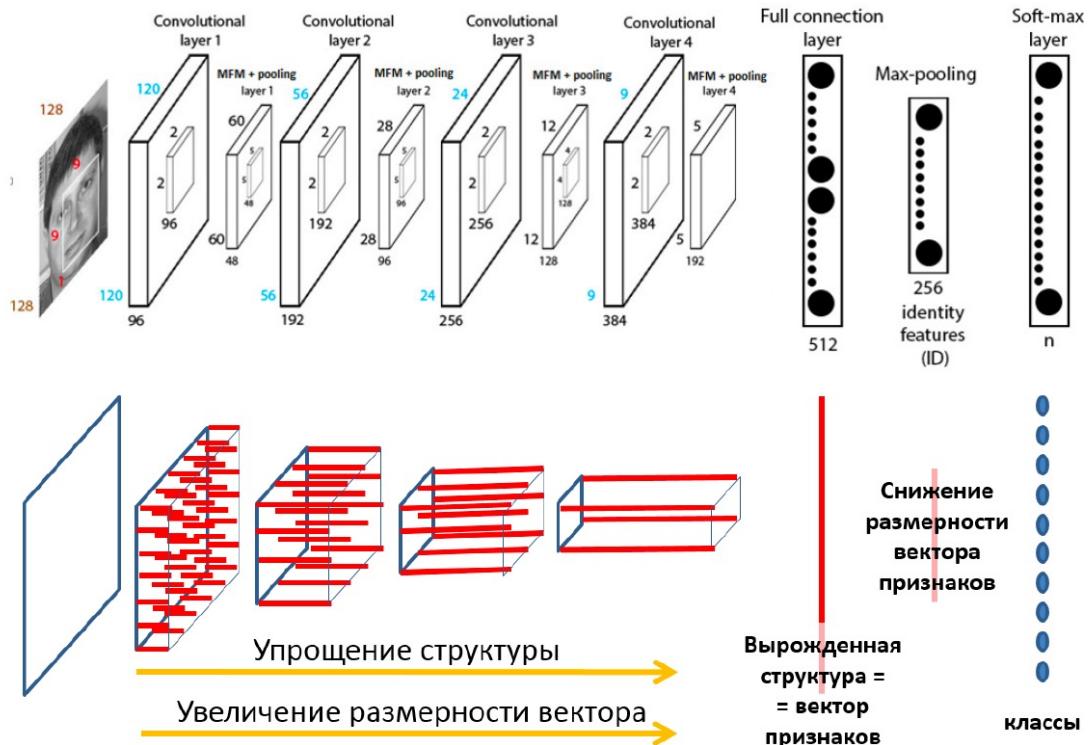


# CNN – принятие решений в логических играх



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

# CNN – как способ векторизации изображений



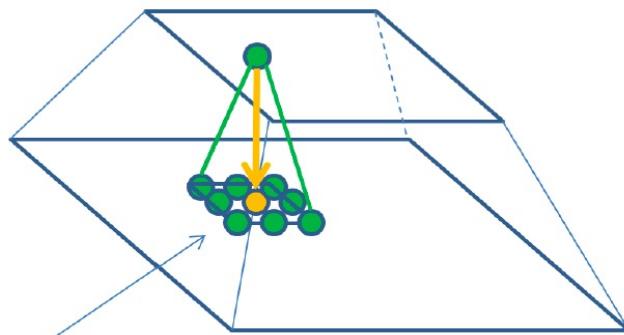
# CNN – идея обобщения на любые структурированные данные

Допустим, каждый объект имеет структуру, заданную графом

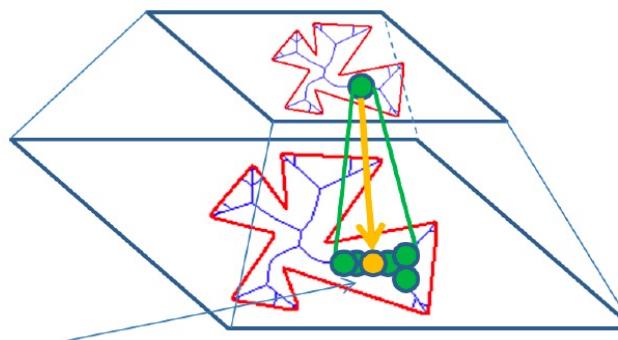
Свёртка определяется по локальной окрестности вершины

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