НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

# Алгоритмы компьютерного зрения

Савченко А.В.
Д.т.н., проф. Каф. Информационные системы и технологии
avsavchenko@hse.ru

# План лекции

1. Основные задачи компьютерного зрения
2. Метод Виолы-Джонса
3. Модели формы
4. Deep learning
5. Сверточные нейронные сети
6. Проблемы ConvNet
7. Распознавание объектов на видео
8. Детектирование объектов

# Основные задачи компьютерного зрения

# Очистка изображений от шума

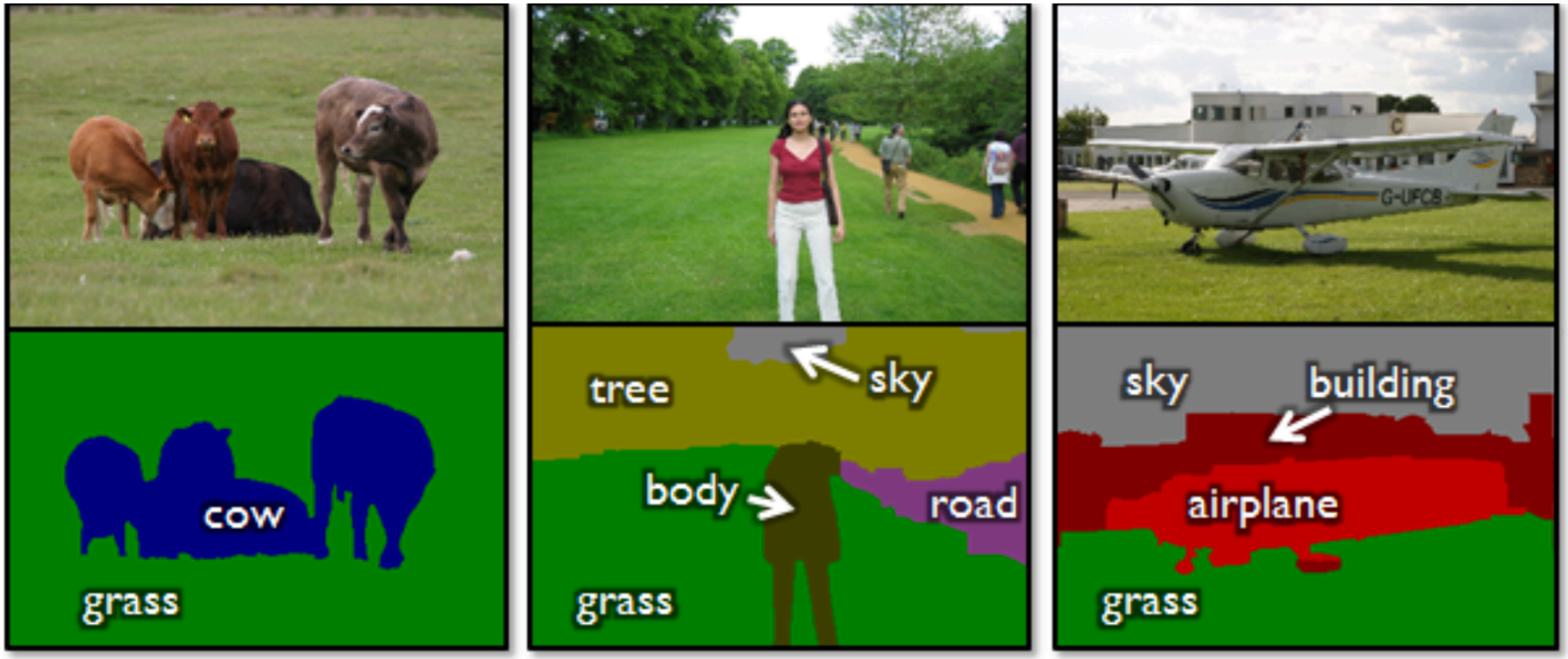
https://www.mathworks.com/help/wavelet/examples/denoising-signals-and-images.html

# Распознавание изображений

(a) Siberian husky

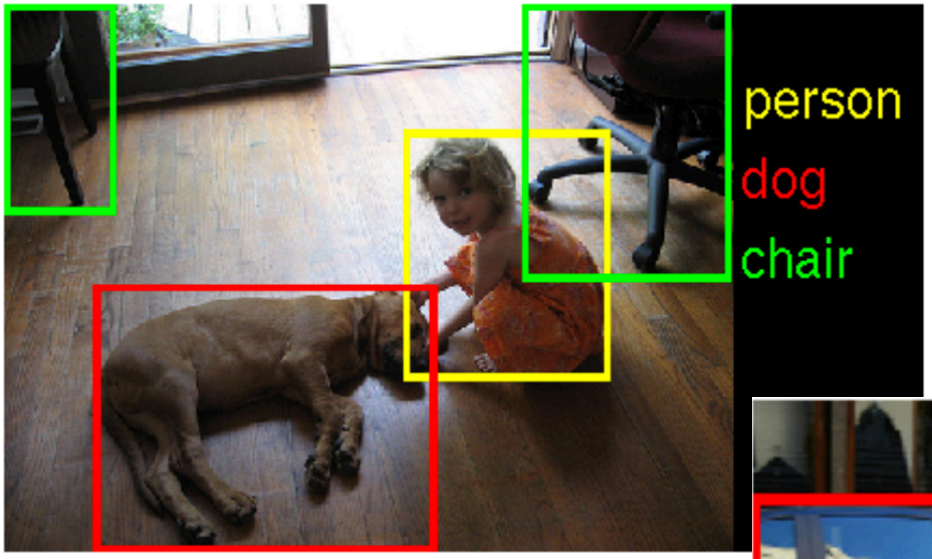
(b) Eskimo dog

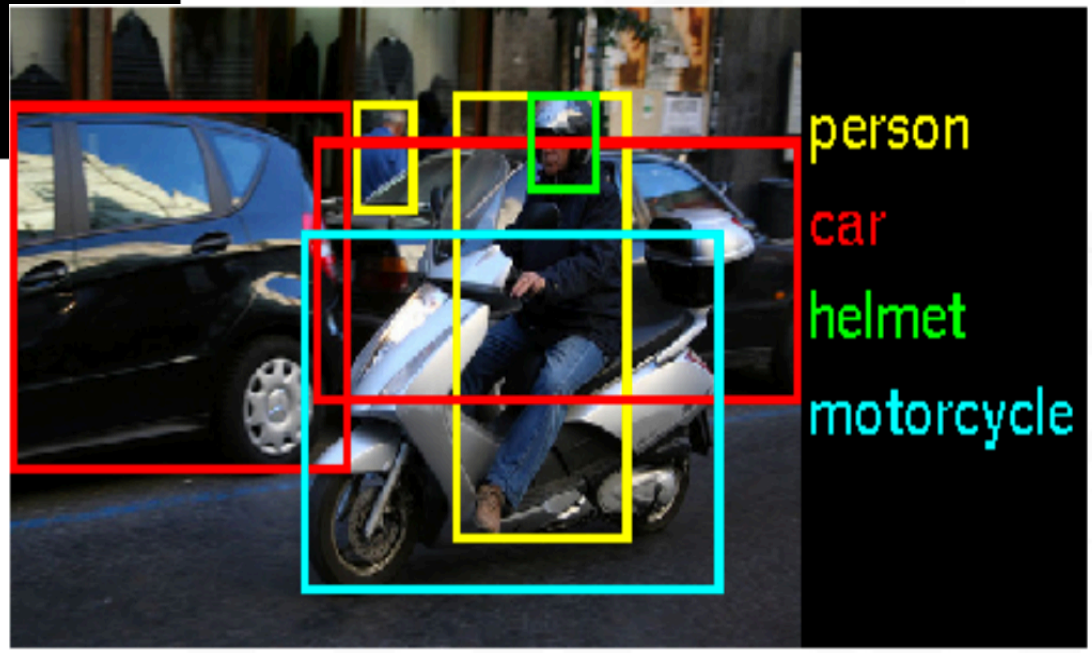
# Семантическая сегментация




| object classes | building | grass | tree | cow | sheep | sky | airplane | water | face | car |
|---|---|---|---|---|---|---|---|---|---|---|
| bicycle | flower | sign | bird | book | chair | road | cat | dog | body | boat |

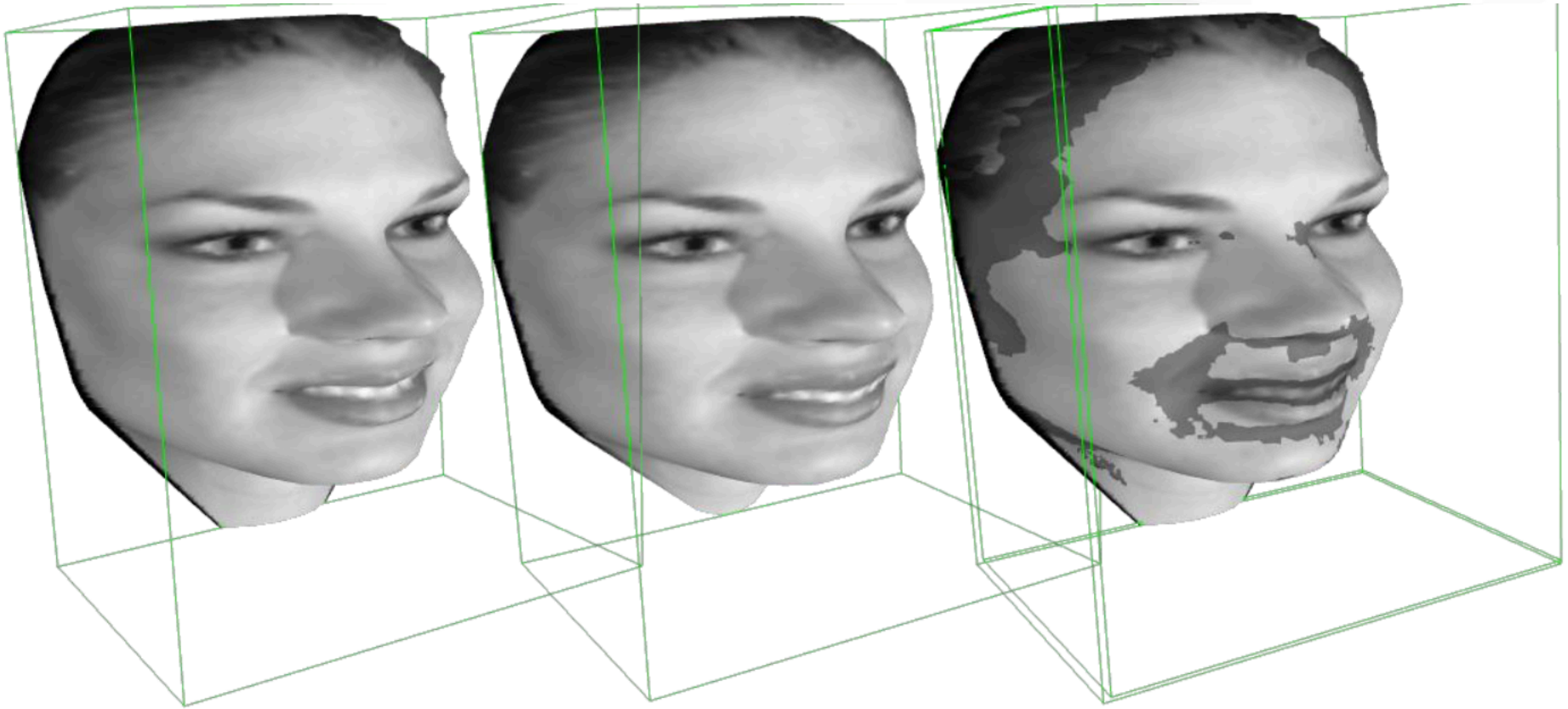
# Детектирование объектов на изображениях



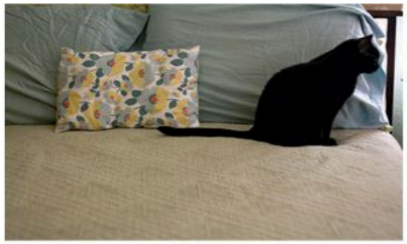
# Восстановление 3D модели по изображению

**Предсказание нейросетевой модели** | **Оригинальная 3D модель** | **Ошибка**



[Рассадин, 2017]

# Текстовое описание изображений

| Describes without errors | Describes with minor errors | Somewhat related to the image | Unrelated to the image |
|---|---|---|---|
|  A person riding a motorcycle on a dirt road. |  Two dogs play in the grass. |  A skateboarder does a trick on a ramp. |  A dog is jumping to catch a frisbee. |
|  A group of young people playing a game of frisbee. |  Two hockey players are fighting over the puck. |  A little girl in a pink hat is blowing bubbles. |  A refrigerator filled with lots of food and drinks. |
|  A herd of elephants walking across a dry grass field. |  A close up of a cat laying on a couch. |  A red motorcycle parked on the side of the road. |  A yellow school bus parked in a parking lot. |

http://arxiv.org/abs/1411.4555 “Show and Tell: A Neural Image Caption Generator”

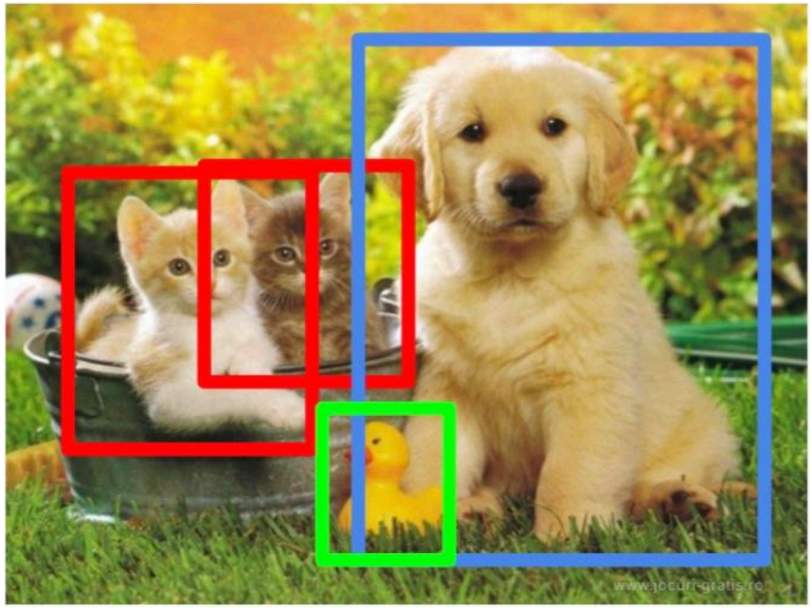
# Перенос стиля


https://arxiv.org/abs/1508.06576 “A Neural Algorithm of Artistic Style”

# Метод Виолы-Джонса

## (по материалам В. Писаревского, руководителя проекта OpenCV, Intel)

# Детектирование объектов. Постановка задачи



**Определить положение (прямоугольник) и метку для каждого объекта (из определенного множества классов) на изображении**

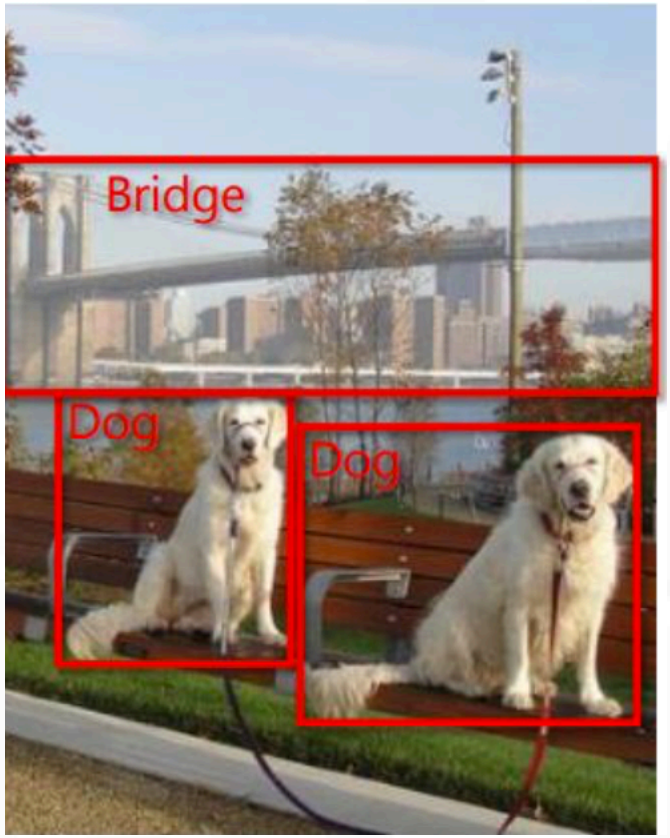
Кошка

Собака

Утка

https://telecombcn-dl.github.io/2017-dlcv/slides/D3L4-objects.pdf

# Распознавание (классификация) и детектирование


http://tutorial.caffe.berkeleyvision.org/caffe-cvpr15-detection.pdf

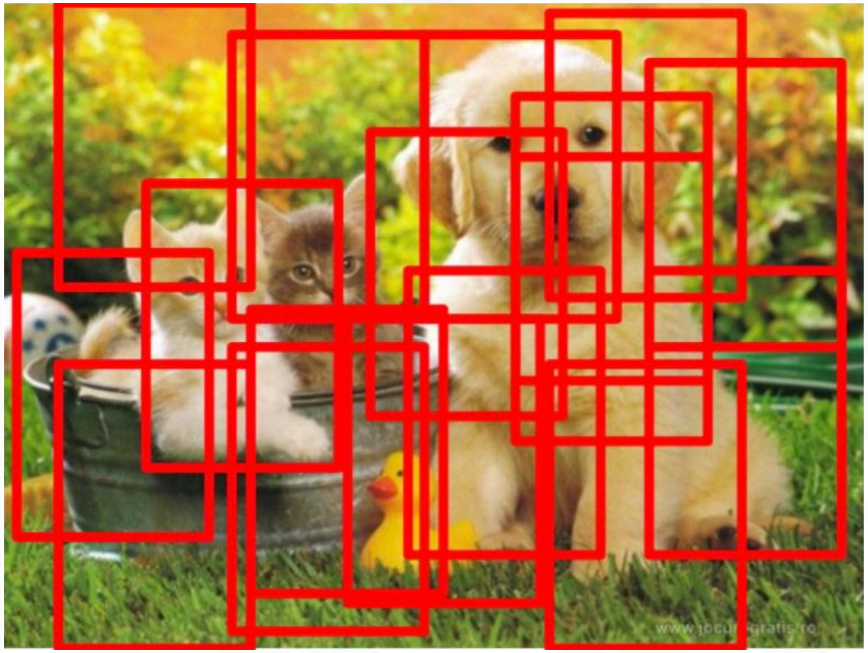
# Детектирование объектов как задача классификации

Классы = [Кошка, Собака, Утка]

Кошка (окно(0,0,w,h))? **Нет**

Собака (окно(0,0,w,h))? **Нет**

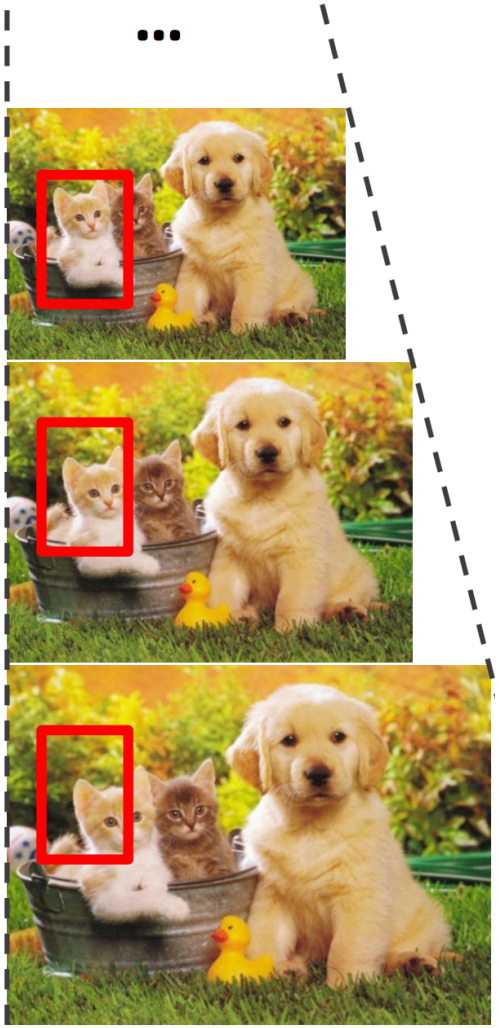
Утка(окно(0,0,w,h))? **Нет**



**Перебираем разные положения и размеры окон**

**Классификатор должен быть очень быстрым!**

# Multi-scale sliding window approach



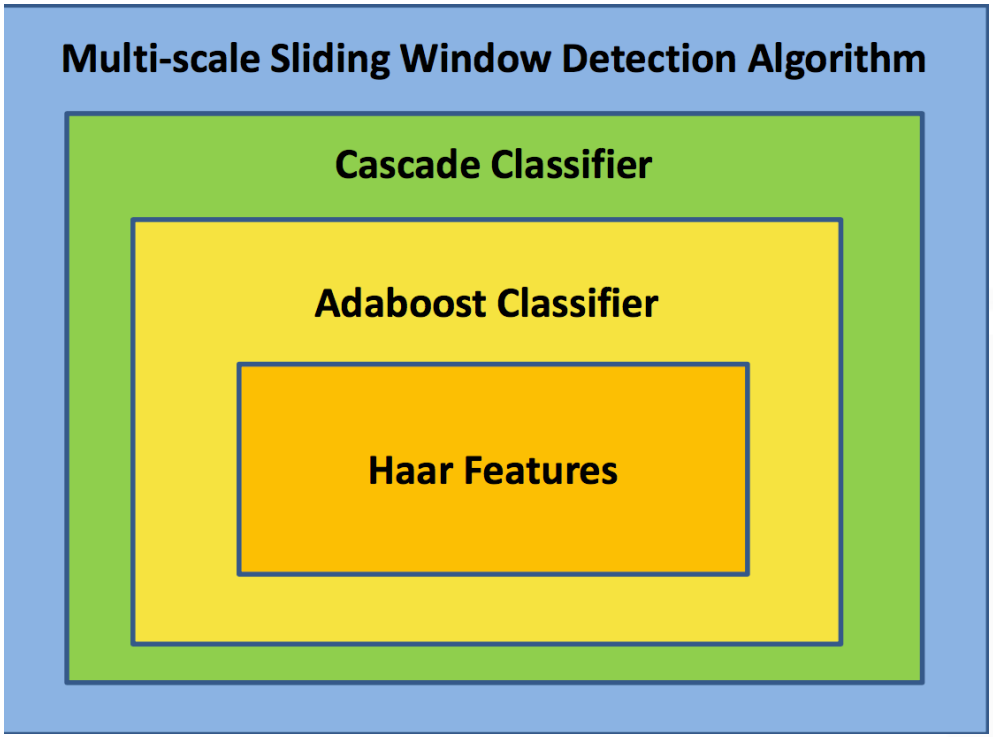
**Пирамида + фиксированный размер окна:**

1. **Нужно натренировать только один классификатор на каждый класс.**
2. **На верхних слоях пирамиды меньше позиций окна – экономим вычисления**
3. **Фиксированное соотношение сторон: ok для лиц, почти ok для пешеходов, не ok – для машин**

# Детектирование лиц (1)

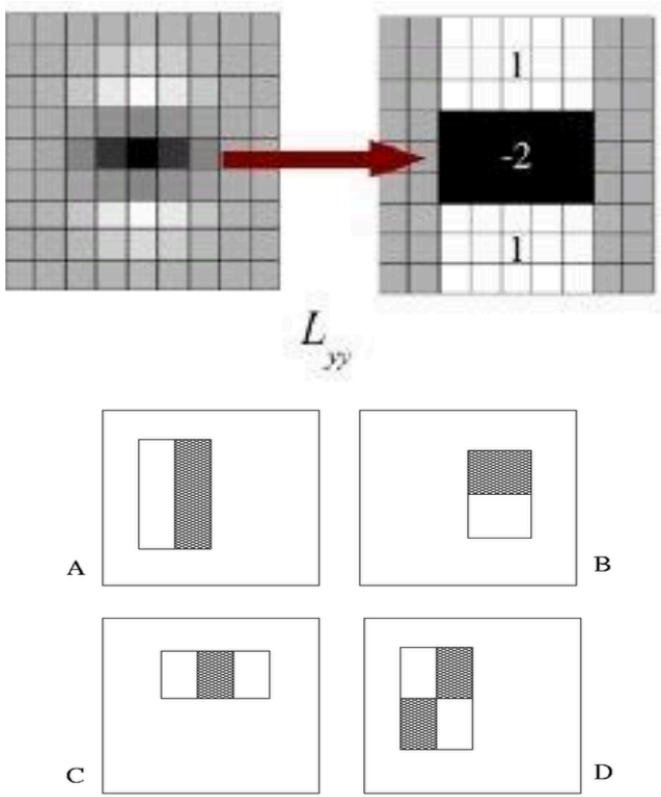
**Метод Виолы-Джонса – один из первых детекторов лиц в режиме реального времени – 15 FPS на Pentium III**

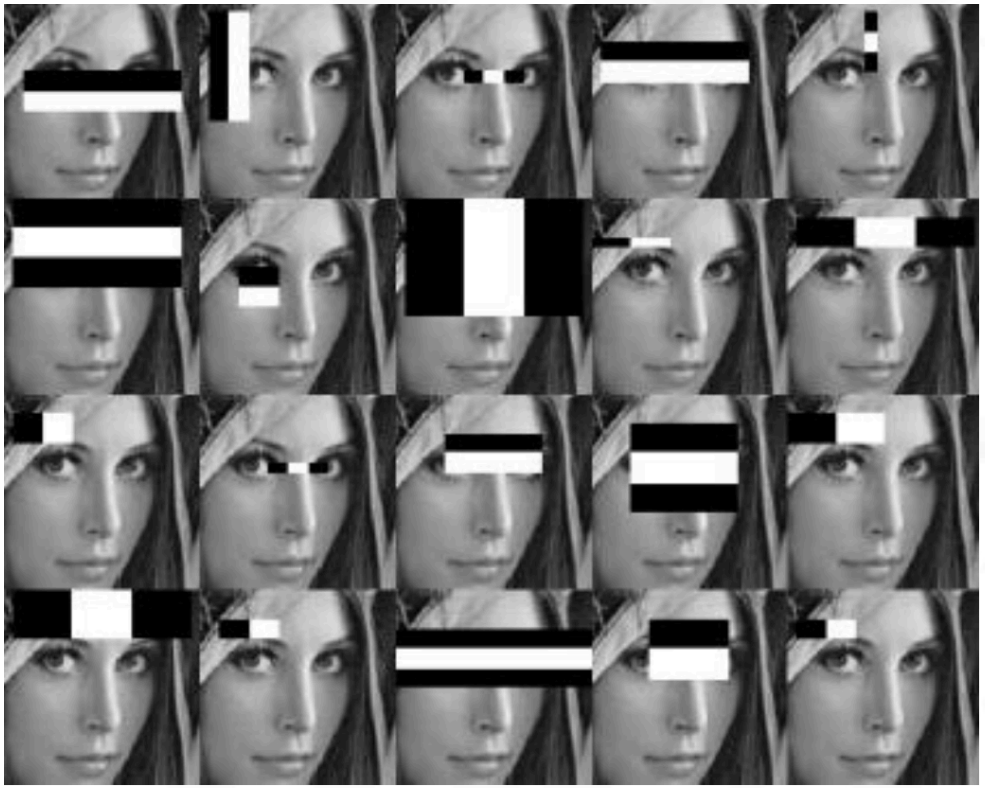
**P. Viola, M. Jones. Rapid object detection using a boosted cascade of simple features. CVPR 2001.**




# Детектирование лиц (2). Фильтры Хаара

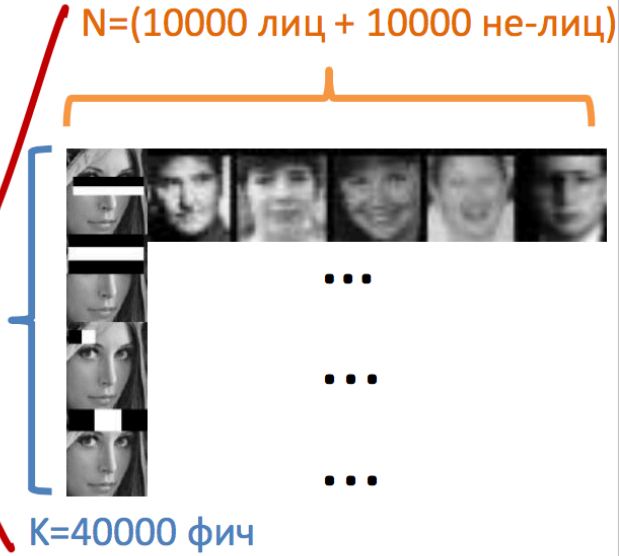
**Тренировочная выборка: 10000 лиц приведенных к размеру 20x20, примерно столько же не-лиц 20x20.**
**Для окна 20x20 всего порядка 40000 признаков Хаара**



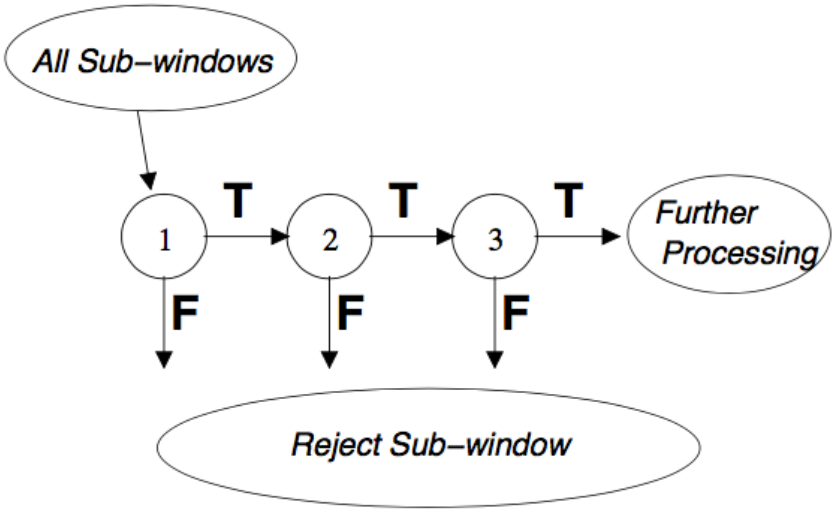
# Детектирование лиц (3). AdaBoost

- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where $m$ and $l$ are the number of negatives and positives respectively.
- For $t = 1, \ldots, T$:
  1. Normalize the weights,
  $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$
  so that $w_t$ is a probability distribution.
  2. For each feature, $j$, train a classifier $h_j$ which is restricted to using a single feature. The error is evaluated with respect to $w_t$, $\epsilon_j = \sum_i w_i |h_j(x_i) - y_i|$.
  3. Choose the classifier, $h_t$, with the lowest error $\epsilon_t$.
  4. Update the weights:
  $$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$
  where $e_i = 0$ if example $x_i$ is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- The final strong classifier is:
$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$
where $\alpha_t = \log \frac{1}{\beta_t}$



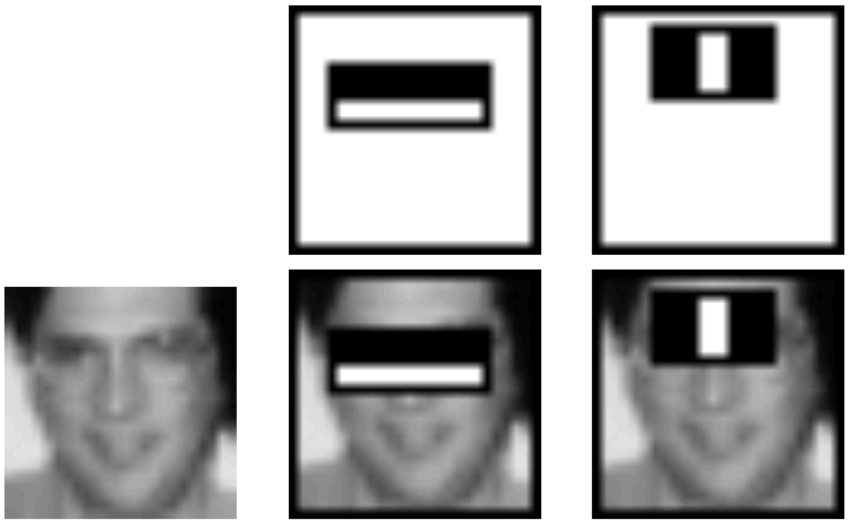

Шаг 2 выполняется за K*O(N) !!!

# Детектирование лиц (4). Каскад классификаторов

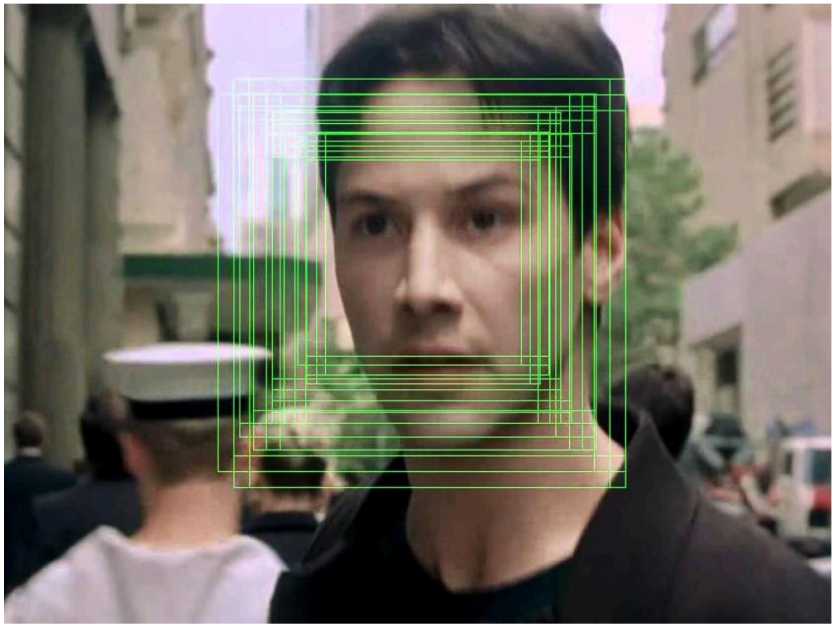



Viola, Jones, 2001

## Признаки в первом каскаде



- Каждый fi натренирован с ~1 (0.999) hit-rate и 0.5 false positive rate.
- Для тренировки каждого fi отбираем только сэмплы (позитивные и негативные), которые прошли все предыдущие f1...fi-1.
- Результирующий классификатор из 20 стадий очень точный: hit-rate $0.999^{20} \approx 0.98$ и false alarm rate $0.5^{20} \approx 10^{-6}$!
- Очень быстрый: откидывает ~90% кандидатов на первых 3 стадиях
- Сложно набрать достаточно негативных примеров для тренировки каждой стадии классификатора

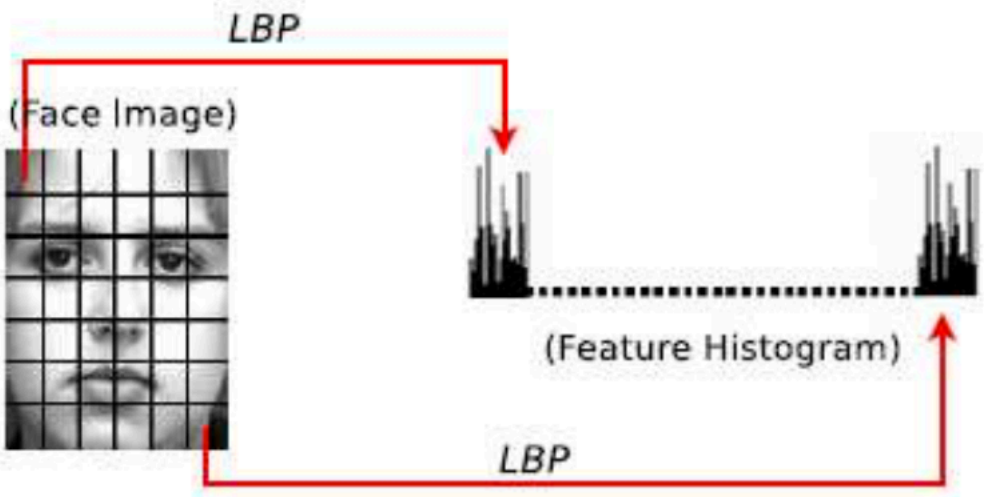


**Коэффициент сходства Жаккара (Jaccard)**

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

- **A и B считаются близкими если JD(A, B) ≤ eps (eps ≈ 0.6..0.9)**
- **Можно построить классы эквивалентности (связные компоненты в графе близких кандидатов). Из каждого класса сформировать один прямоугольник (median, avg, ...)**
- **Можно использовать non-maxima suppression: рассмотреть все пары близких прямоугольников (A,B), выбрать прямоугольник с большим значением confidence**

# Детектирование лиц (6). LBP признаки




http://cs229.stanford.edu/proj2008/Jo-FaceDetectionUsingLBPfeatures.pdf

**«Слабый» классификатор определяется признаком $f_j(x)$ – номером элемента гистограммы, порогом и операцией (меньше/ больше)**

$$h_j(x) = \begin{cases} 1 & \text{if } p_j f_j(x) \leq p_j \theta_j \\ 0 & \text{otherwise} \end{cases}$$

- **Все вычисления целочисленные**
- **Всего для окна 20x20 имеется ≈ 4000 LBP фич вместо 40000 фич Хаара**
- **На iphone каскад на LBP фичах работает в 10 раз быстрее Хаара, на ноутбуке – в 2-3 раза быстрее, занимая в 20 раз меньше памяти**
- **Точность LBP по сравнению с признаками Хаара обычно ниже**

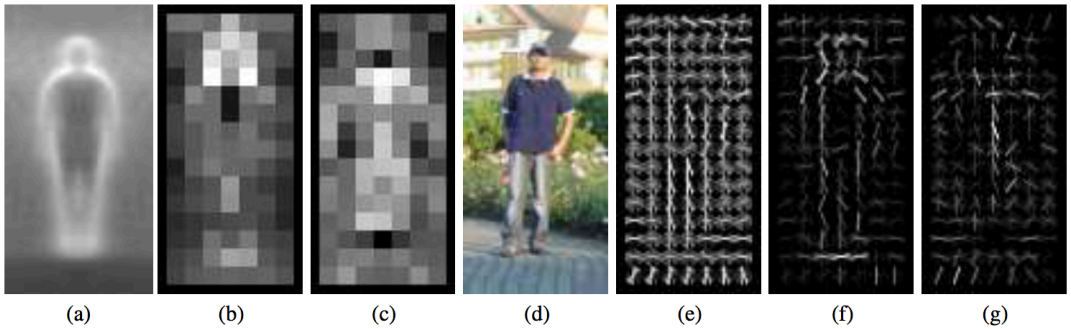
# Детектирование лиц (7). Пример программного кода в OpenCV

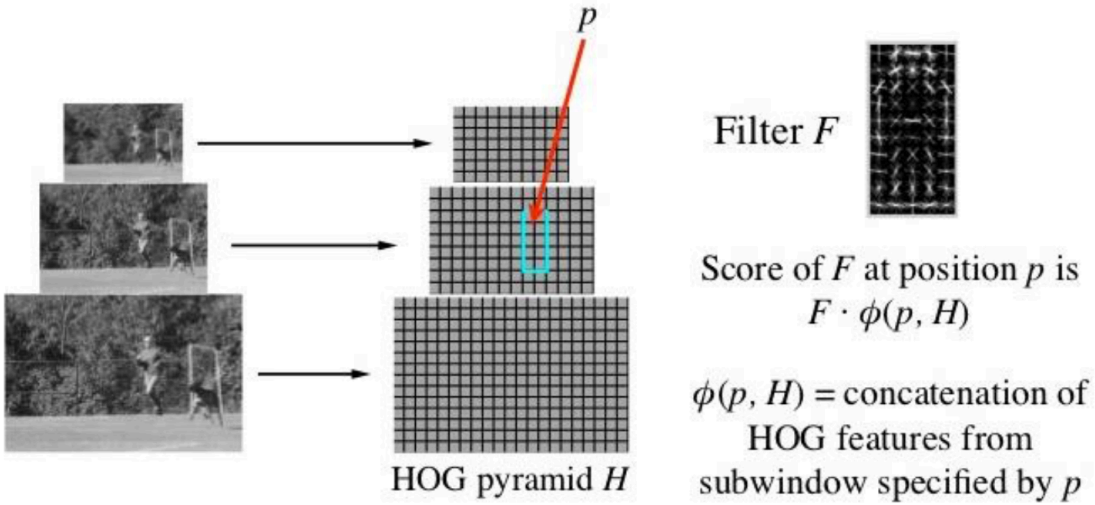
**Для повышения точности можно детектировать область глаз внутри лица**

```
Mat img_gray;
cvtColor( matImage, img_gray, CV_BGR2GRAY );
equalizeHist(img_gray, img_gray);
std::vector<Rect> faces;

face_cascade.detectMultiScale(img_gray, faces,1.3, 2,
     CASCADE_DO_ROUGH_SEARCH|CASCADE_SCALE_IMAGE, Size(30, 30));

for(int i = 0; i < faces.size(); ++i ){
    Rect& r = faces[i];
    std::vector<Rect> eyes;
    cv::Mat faceROI = img_gray (r);
    eye_cascade. detectMultiScale(faceROI, eyes,1.1, 2,
      CASCADE_DO_ROUGH_SEARCH|CASCADE_SCALE_IMAGE, Size(30, 30));
    if(!eyes.empty()){
        //do something with face r
    }
 }
```

- **Haar/LBP признаки не подходят для разнотекстурных объектов**
- **Более точные результаты получаются с помощью HOG**

## Детектирование пешеходов




**(a) The average gradient image over the training examples. (b) Each "pixel" shows the maximum positive SVM weight in the block centred on the pixel. (c) Likewise for the negative SVM weights. (d) A test image. (e) It's computed R-HOG descriptor. (f,g) The R-HOG descriptor weighted by respectively the positive and the negative SVM weights.**

Dalal, Triggs. Histograms of oriented gradients for human detection (2005)

**Используются пирамиды и скользящее окно. Окно разбивается на блоки, шаг сдвига окна кратен блоку.**

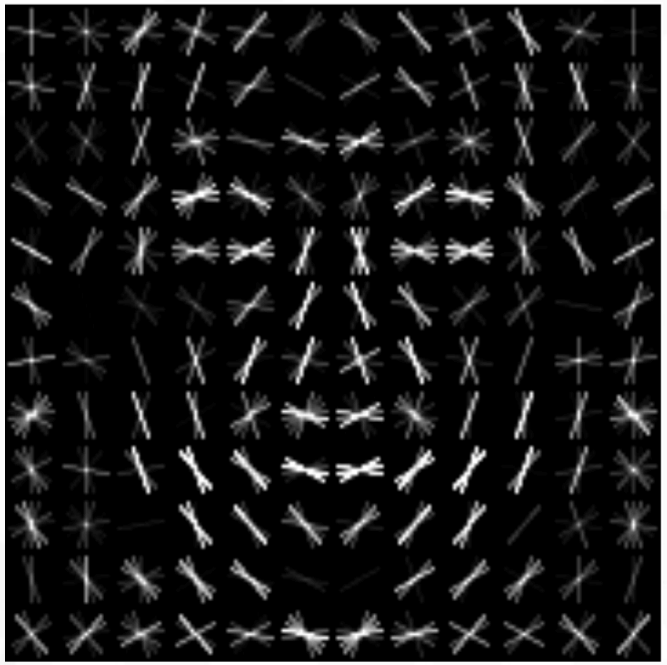
# И вновь детектирование лиц

Dlib C++ Library

```
dets, scores, idx = detector.run(img, 1, -1)
for i, d in enumerate(dets):
    print("Detection {}, score: {}, face_type:{}".format(d,
scores[i], idx[i]))
```

**Детектор лиц в Dlib намного точнее OpenCV, но в несколько раз медленнее**

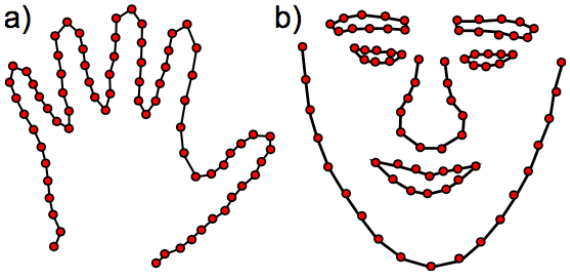
**HOG работают и для лиц!**



https://www.youtube.com/watch?v=LsK0hzcEyHI

# Модели формы

## Представление формы. Landmark points $W=[w_1,\ldots,w_N]$




## Точки близки к краям (edge)

$$Pr(\mathbf{x}|\mathbf{W}) \propto \prod_{n=1}^{N} \exp\left[-(\text{dist}\left[\mathbf{x}, \mathbf{w}_n\right])^2\right]$$

$$\hat{\mathbf{W}} = \underset{\mathbf{W}}{\text{argmax}}\left[Pr(\mathbf{W}|\mathbf{x})\right] = \underset{\mathbf{W}}{\text{argmax}}\left[Pr(\mathbf{x}|\mathbf{W})Pr(\mathbf{W})\right]$$

$$Pr(\mathbf{W}) \propto \prod_{n}^{N} \exp\left[\alpha\,\text{space}[\mathbf{w}, n] + \beta\,\text{curve}[\mathbf{w}, n]\right]$$

$$\text{space}[\mathbf{w}, n] = -\left(\frac{\sum_{n=1}^{N}\sqrt{(\mathbf{w}_n - \mathbf{w}_{n-1})^T(\mathbf{w}_n - \mathbf{w}_{n-1})}}{N} - \sqrt{(\mathbf{w}_n - \mathbf{w}_{n-1})^T(\mathbf{w}_n - \mathbf{w}_{n-1})}\right)^2, \quad (17.5)$$

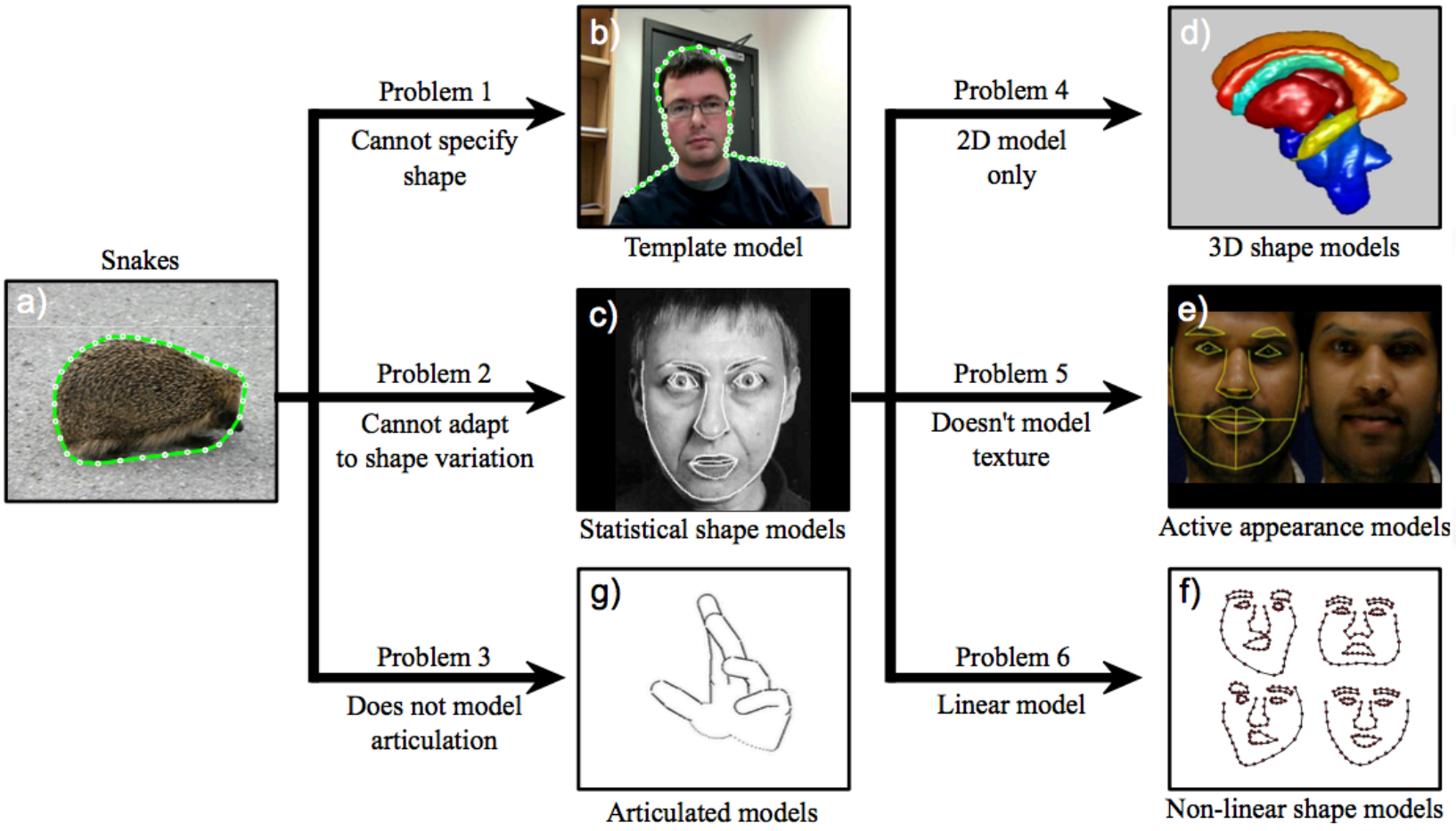
**– одинаковое расстояние между точками**

$$\text{curve}[\mathbf{w}, n] = -(\mathbf{w}_{n-1} - 2\mathbf{w}_n + \mathbf{w}_{n+1})^T(\mathbf{w}_{n-1} - 2\mathbf{w}_n + \mathbf{w}_{n+1})$$

**– контур должен быть гладким**

Ponce «Computer vision: models, learning and inference»

# Параметрические контурные модели формы

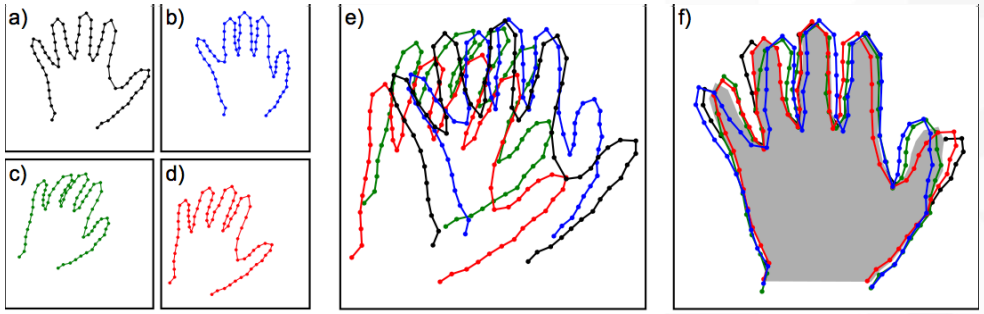



Ponce «Computer vision: models, learning and inference»

## Subspace shape model

$$\mathbf{w}_i = \boldsymbol{\mu} + \boldsymbol{\Phi}\mathbf{h}_i + \epsilon_i \qquad Pr(\mathbf{w}_i) = \int Pr(\mathbf{w}_i|\mathbf{h}_i)Pr(\mathbf{h}_i)d\mathbf{h}_i = \text{Norm}_{\mathbf{w}_i}[\boldsymbol{\mu}, \boldsymbol{\Phi}\boldsymbol{\Phi}^T + \sigma^2\mathbf{I}].$$

## Выравнивание контуров в обучающем множестве (Procrustes analysis)




## Iterative closest point – итеративный поиск ближайших точек контура $y_n$

$$\hat{\mathbf{h}} = \underset{\mathbf{h}}{\text{argmax}} \left[ \max_{\boldsymbol{\Psi}} \left[ \sum_{n=1}^{N} \left( -\frac{(\text{dist}\left[\mathbf{x}_i, \mathbf{trans}[\boldsymbol{\mu}_n + \boldsymbol{\Phi}_n\mathbf{h}, \boldsymbol{\Psi}]\right])^2}{\sigma^2} \right) + \log[\text{Norm}_{\mathbf{h}}[\mathbf{0}, \mathbf{I}]] \right] \right]$$

$$\hat{\mathbf{h}} = \underset{\mathbf{h}}{\text{argmax}} \left[ \sum_{n=1}^{N} \log[Pr(\mathbf{y}_n|\mathbf{h}), \boldsymbol{\Psi}] + \log[Pr(\mathbf{h})] \right]$$

$$= \underset{\mathbf{h}}{\text{argmax}} \left[ \sum_{n=1}^{N} -\left(\mathbf{y}_n - \text{trans}[\boldsymbol{\mu}_n + \boldsymbol{\Phi}_n\mathbf{h}, \boldsymbol{\Psi}]\right)^2 / \sigma^2 - \log[\mathbf{h}^T\mathbf{h}] \right]$$


Ponce «Computer vision: models, learning and inference» & [Cootes, 1995]

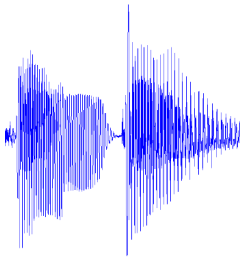
# Deep learning

# Традиционный подход к распознаванию образов

## Специально спроектированные признаки + обучаемый классификатор

Hand-crafted Feature Extractor → "Simple" Trainable Classifier →

### Распознавание речи



MFCC → Mix of Gaussians → Classifier →

fixed — unsupervised — supervised

### Распознавание изображений

SIFT HoG → K-means Sparse Coding → Pooling → Classifier →

fixed — unsupervised — supervised

Low-level Features — Mid-level Features

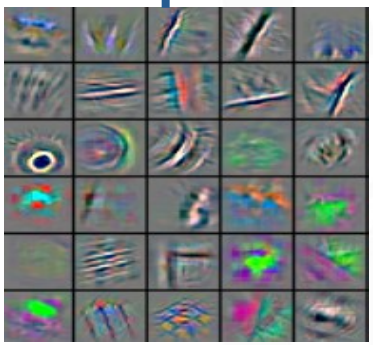
DLI-Teaching Kit

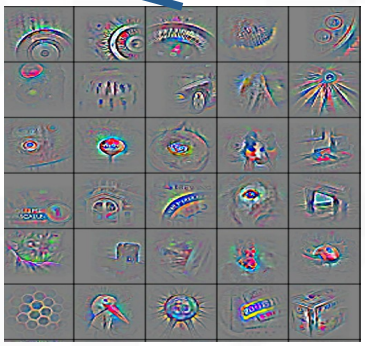
# Deep learning=Learning representations/features

**– End-to-end learning / Feature learning / Deep learning**
**Обучаемые признаки + обучаемый классификатор**

Trainable Feature Extractor → Trainable Classifier

Low-level feature → Mid-level feature → High-level feature → Trainable classifier



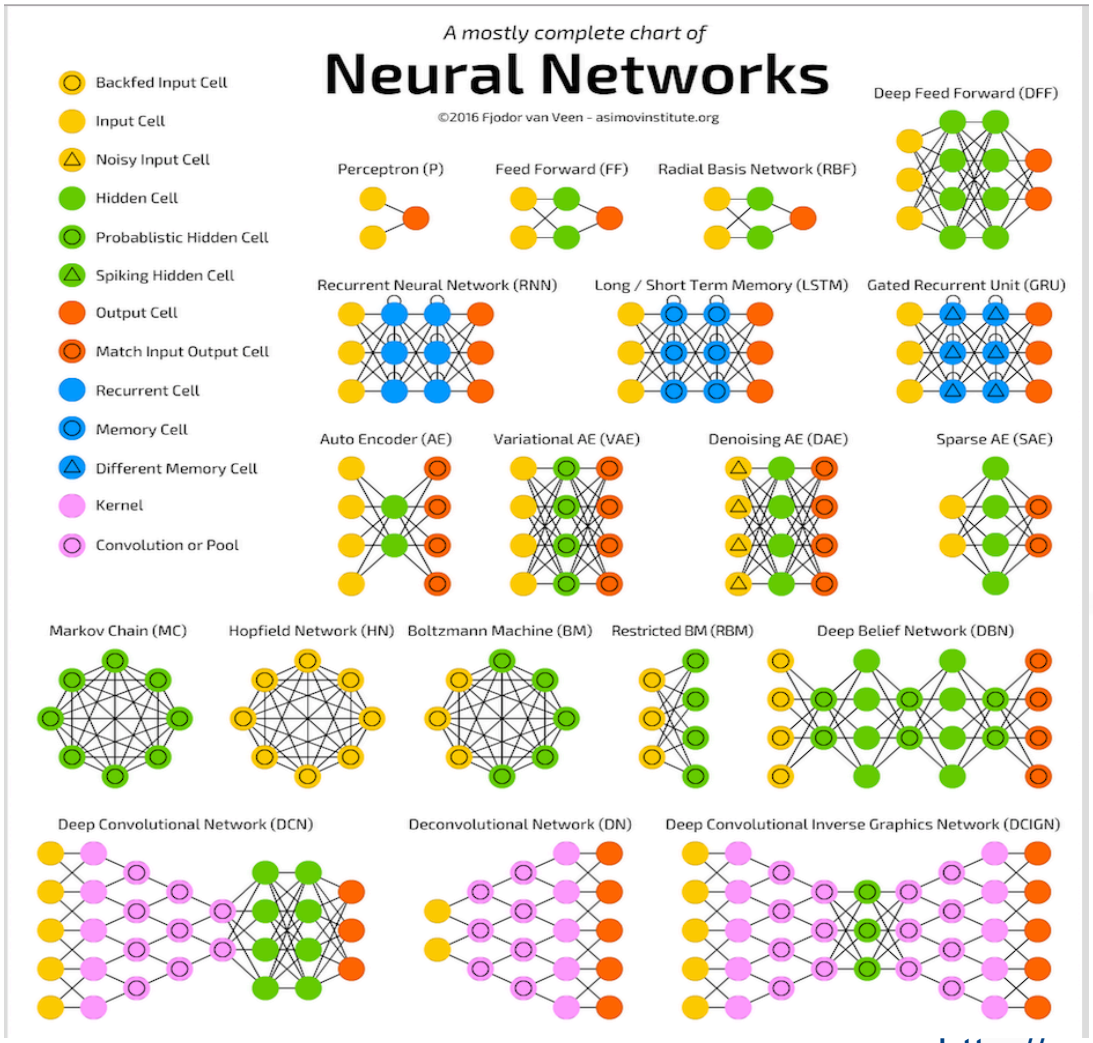


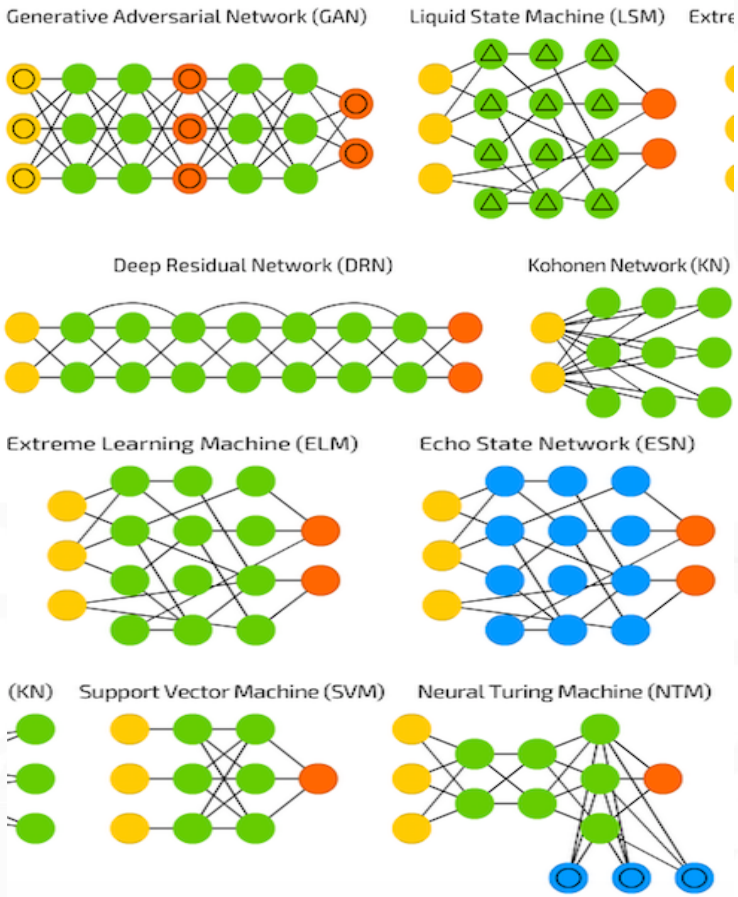
DLI-Teaching Kit, Zeiler & Fergus 2013

# Инструменты

| **Инструмент** | **Язык программирования** |
|---|---|
| Tensorflow | Python, C++ |
| Keras | Python |
| Theano | Python |
| Caffe | C++, Python, Matlab |
| Torch | Lua, C, Python (PyTorch) |
| Caffe2 | Python, C++ |
| CNTK | Python, C++, C#, Java |
| Deeplearning4j | Java, Python |
| Misc (deeplearning.net/software_links/) | |

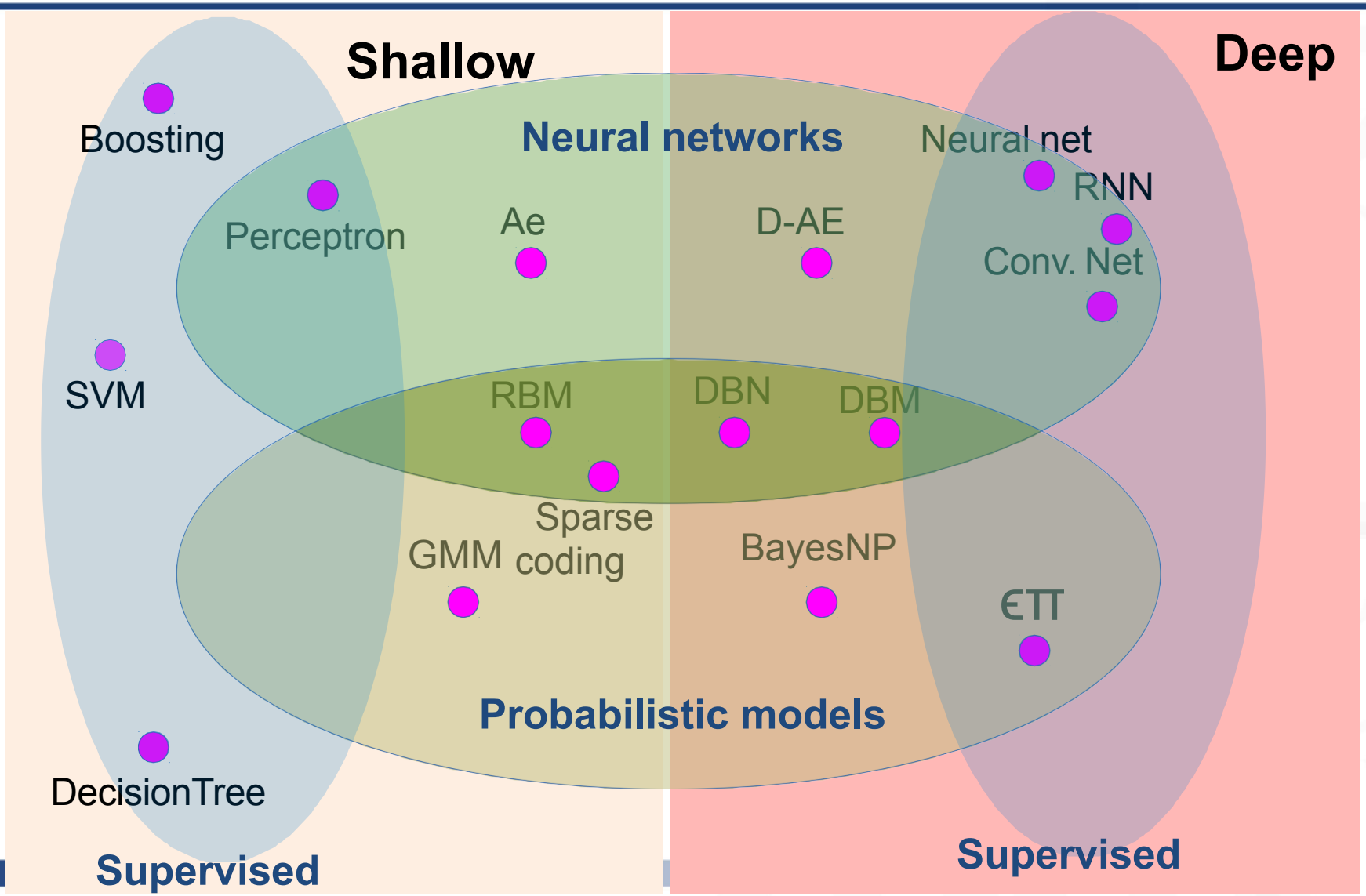
# Многообразие архитектур (1)




http://www.asimovinstitute.org/neural-network-zoo/

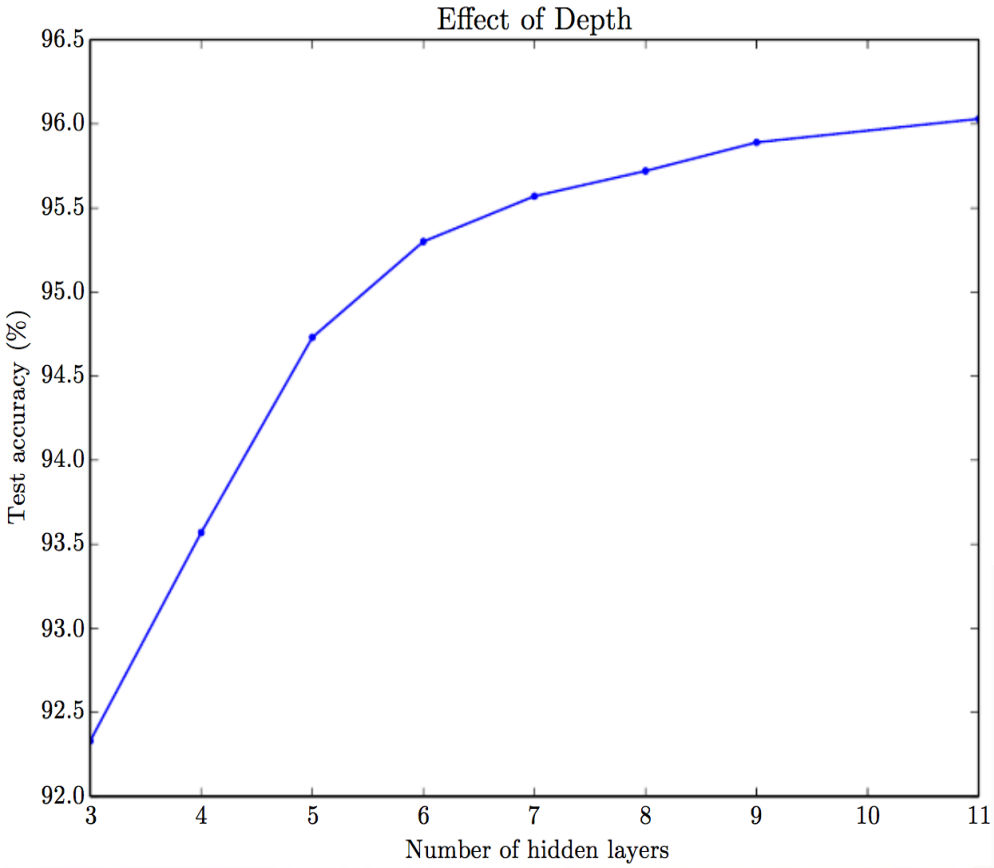
# Многообразие архитектур (2)

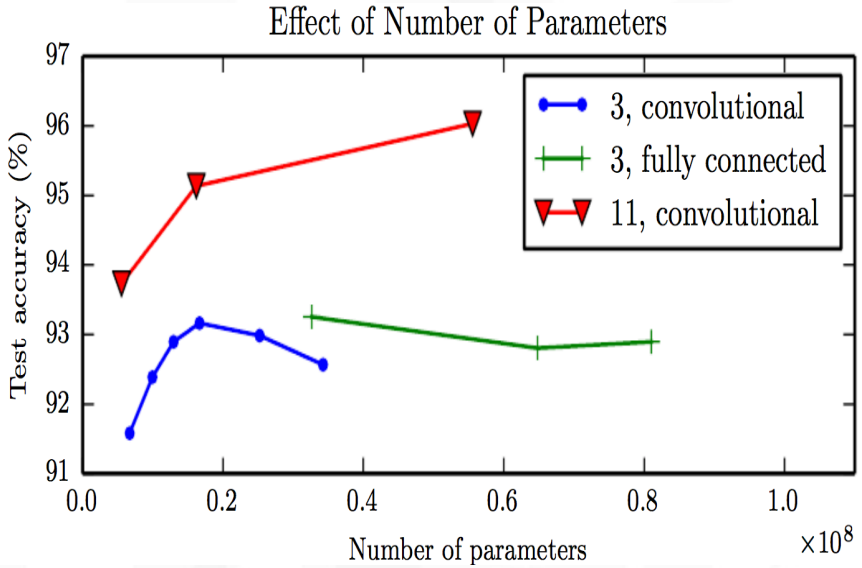


# Эффект глубины

**Для теоремы универсальной аппроксимации может потребоваться экспоненциально большое число нейронов!**

**Пример. Классификация изображений номеров домов**




Goodfellow et al 2014,
Goodfellow I., Bengio Y., Courville A.,
Deep learning

# Функции потерь (loss)

| Задача | Функция активации последнего слоя | Функция потерь |
|---|---|---|
| Регрессия | Линейная | Квадратичная (MSE) |
| Бинарная классификация | Сигмоида $\hat{y} = \sigma(z)$ | $P(D\|\theta) = \prod_{i=1}^{n} f(x_i,\theta)^{y_i} \cdot (1 - f(x_i,\theta))^{(1-y_i)}$ Negative log likelihood (NLL) – бинарная кросс-энтропия $-\sum_{i=1}^{n} y_i \log f(x_i,\theta) + (1-y_i)\log(1-f(x_i,\theta))$ |
| Многоклассовая классификация | Softmax $\text{softmax}(\boldsymbol{z})_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$ | Категориальная кросс-энтропия: $-\sum_{j=1}^{n} \text{logsoftmax}(\mathbf{z}_j)_{y_j} = -\sum_{j=1}^{n} \mathbf{y}_j \log \mathbf{f}(\mathbf{x}_j,\theta)$ |

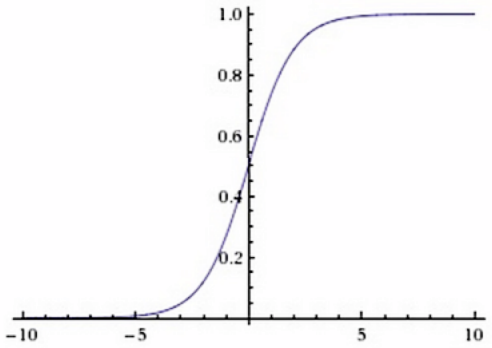
**Производная функции потерь по z пропорциональна разности выхода сети и идеального ответа y.**

**MSE loss для задач классификации не подходит – производная затухает!**

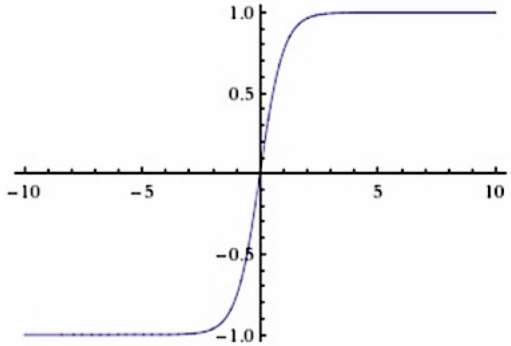
# Функции активации скрытых слоев

## Традиционные

### Логистическая сигмоида

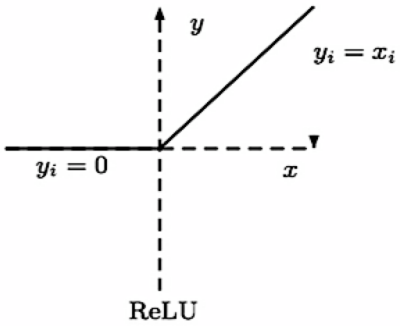



### tanh

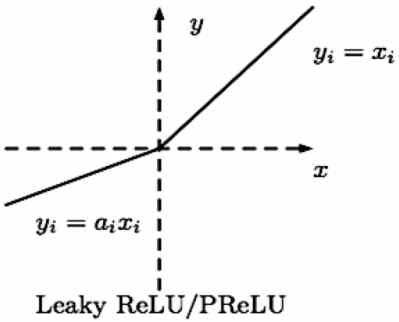



## Современные ReLU-based

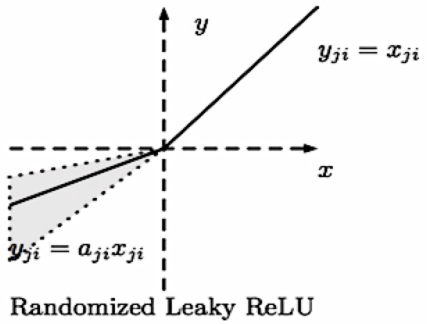
### ReLU(Rectified Linear Unit)




### Parametrized ReLU




### Randomized ReLU




http://lamda.nju.edu.cn/weixs/slide/CNNTricks_slide.pdf
PReLU - http://arxiv.org/abs/1502.01852

# Режим обучения: Batch vs Stochastic

- **Batch – функция потерь считается для всех примеров**

– Медленно
– Наиболее точная оценка градиента функции потерь

- **Stochastic – функция потерь считается для выбранного наугад примера**

– Очень быстро
– Оценка градиента функции потерь неточная

- **Mini-Batch – функция потерь считается для нескольких выбранных наугад примеров**

– Относительно быстро
– Оценка градиента функции потерь достаточно точная, особенно для наборов данных с повышенной избыточностью (redundancy)

**В теории:**
**– точность оценки градиента увеличивается пропорционально квадратному корню из числа примеров**
**– время вычислений растет линейно**

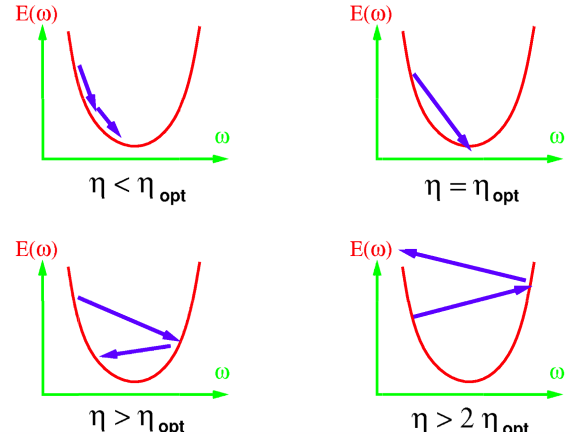
# Обучение сети (1). SGD: Stochastic gradient descent

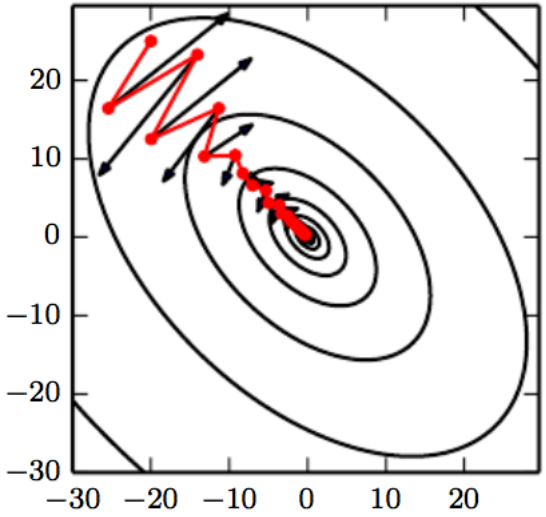
$$\omega \leftarrow \omega - \eta \frac{\partial E}{\partial \omega}$$

gradient of objective function

weight vector

learning rate




**Require:** Learning rate $\epsilon_k$.
**Require:** Initial parameter $\boldsymbol{\theta}$
**while** stopping criterion not met **do**
Sample a minibatch of $m$ examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.
Compute gradient estimate: $\hat{\boldsymbol{g}} \leftarrow +\frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \hat{\boldsymbol{g}}$
**end while**

Goodfellow I., Bengio Y., Courville A., Deep learning

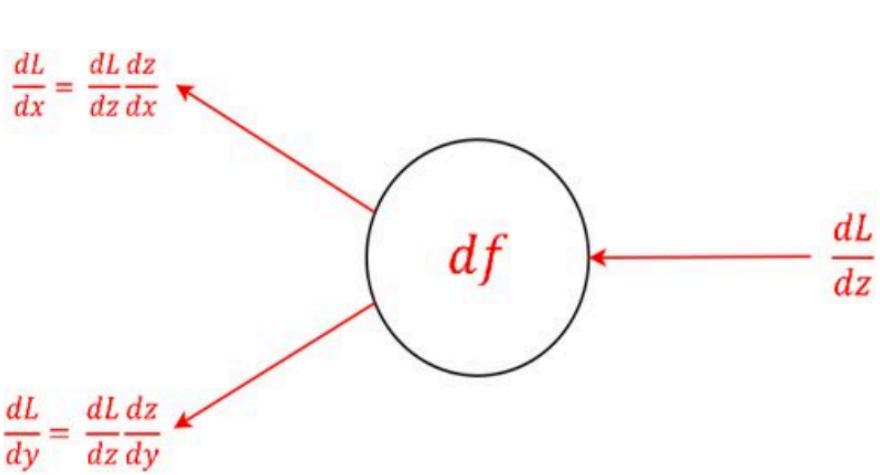
## AdaGrad, AdaDelta, RMSProp, Adam, AdaMax, Nadam,…

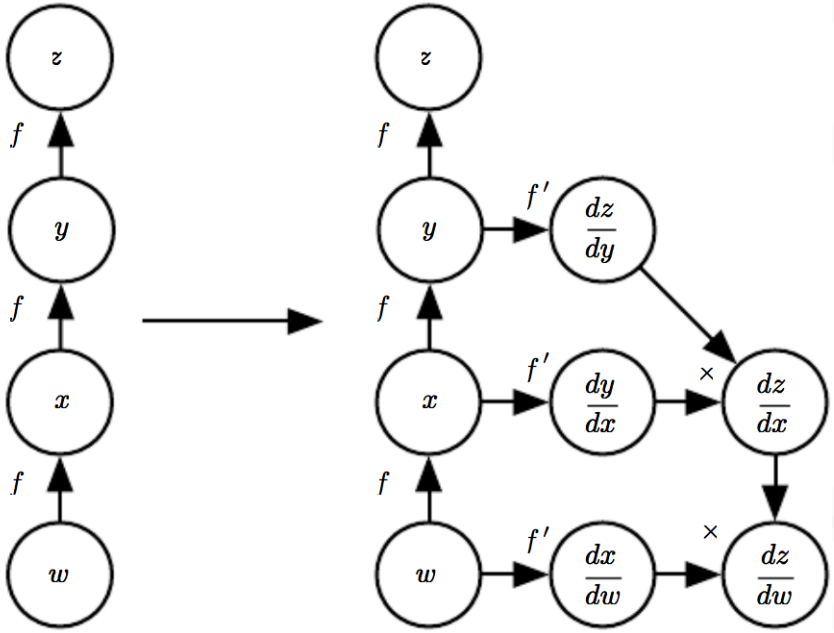
### Пример: RMSProp

**Require:** Global learning rate $\epsilon$, decay rate $\rho$.
**Require:** Initial parameter $\boldsymbol{\theta}$
**Require:** Small constant $\delta$, usually $10^{-6}$, used to stabilize division by small numbers.
Initialize accumulation variables $\boldsymbol{r} = 0$
**while** stopping criterion not met **do**
Sample a minibatch of $m$ examples from the training set $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ with corresponding targets $\boldsymbol{y}^{(i)}$.
Compute gradient: $\boldsymbol{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), \boldsymbol{y}^{(i)})$
Accumulate squared gradient: $\boldsymbol{r} \leftarrow \rho \boldsymbol{r} + (1 - \rho) \boldsymbol{g} \odot \boldsymbol{g}$
Compute parameter update: $\Delta \boldsymbol{\theta} = -\frac{\epsilon}{\sqrt{\delta + \boldsymbol{r}}} \odot \boldsymbol{g}$. ($\frac{1}{\sqrt{\delta + \boldsymbol{r}}}$ applied element-wise)
Apply update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta \boldsymbol{\theta}$
**end while**

Goodfellow I., Bengio Y., Courville A., Deep learning

## Эффективный способ вычисления частных производных




**Может быть реализован для функций, представляемых в виде композиции простых базисных функций – автоматическое дифференцирование (AdaGrad, Theano, Tensorflow,…)**

Goodfellow I., Bengio Y., Courville A., Deep learning

## Прямой проход

**Require:** Network depth, $l$
**Require:** $\boldsymbol{W}^{(i)}, i \in \{1, \ldots, l\}$, the weight matrices of the model
**Require:** $\boldsymbol{b}^{(i)}, i \in \{1, \ldots, l\}$, the bias parameters of the model
**Require:** $\boldsymbol{x}$, the input to process
**Require:** $\boldsymbol{y}$, the target output

$\boldsymbol{h}^{(0)} = \boldsymbol{x}$
**for** $k = 1, \ldots, l$ **do**
  $\boldsymbol{a}^{(k)} = \boldsymbol{b}^{(k)} + \boldsymbol{W}^{(k)} \boldsymbol{h}^{(k-1)}$
  $\boldsymbol{h}^{(k)} = f(\boldsymbol{a}^{(k)})$
**end for**
$\hat{\boldsymbol{y}} = \boldsymbol{h}^{(l)}$
$J = L(\hat{\boldsymbol{y}}, \boldsymbol{y}) + \lambda \Omega(\theta)$

Goodfellow I., Bengio Y., Courville A., Deep learning

## Обратный проход

After the forward computation, compute the gradient on the output layer:

$\boldsymbol{g} \leftarrow \nabla_{\hat{\boldsymbol{y}}} J = \nabla_{\hat{\boldsymbol{y}}} L(\hat{\boldsymbol{y}}, y)$

**for** $k = l, l-1, \ldots, 1$ **do**

Convert the gradient on the layer's output into a gradient into the pre-nonlinearity activation (element-wise multiplication if $f$ is element-wise):

$\boldsymbol{g} \leftarrow \nabla_{\boldsymbol{a}^{(k)}} J = \boldsymbol{g} \odot f'(\boldsymbol{a}^{(k)})$

Compute gradients on weights and biases (including the regularization term, where needed):

$\nabla_{\boldsymbol{b}^{(k)}} J = \boldsymbol{g} + \lambda \nabla_{\boldsymbol{b}^{(k)}} \Omega(\theta)$

$\nabla_{\boldsymbol{W}^{(k)}} J = \boldsymbol{g}\, \boldsymbol{h}^{(k-1)\top} + \lambda \nabla_{\boldsymbol{W}^{(k)}} \Omega(\theta)$

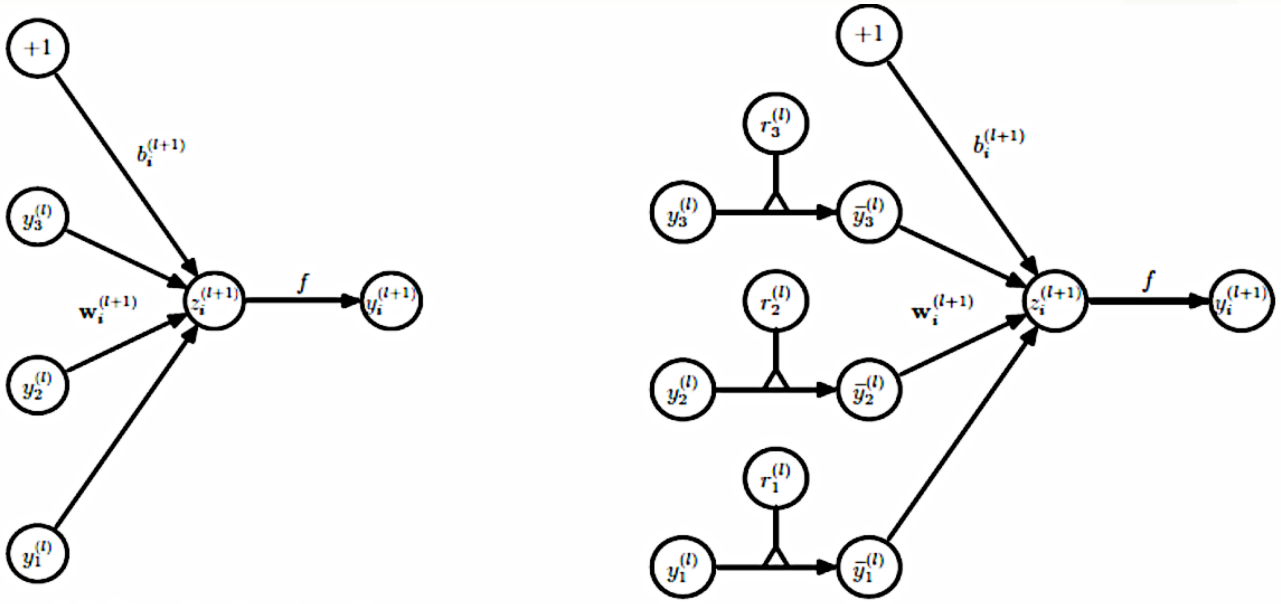
Propagate the gradients w.r.t. the next lower-level hidden layer's activations:

$\boldsymbol{g} \leftarrow \nabla_{\boldsymbol{h}^{(k-1)}} J = \boldsymbol{W}^{(k)\top}\, \boldsymbol{g}$

**end for**

Goodfellow I., Bengio Y., Courville A., Deep learning

# Регуляризация. Dropout (1)




(a) Standard network

(b) Dropout network

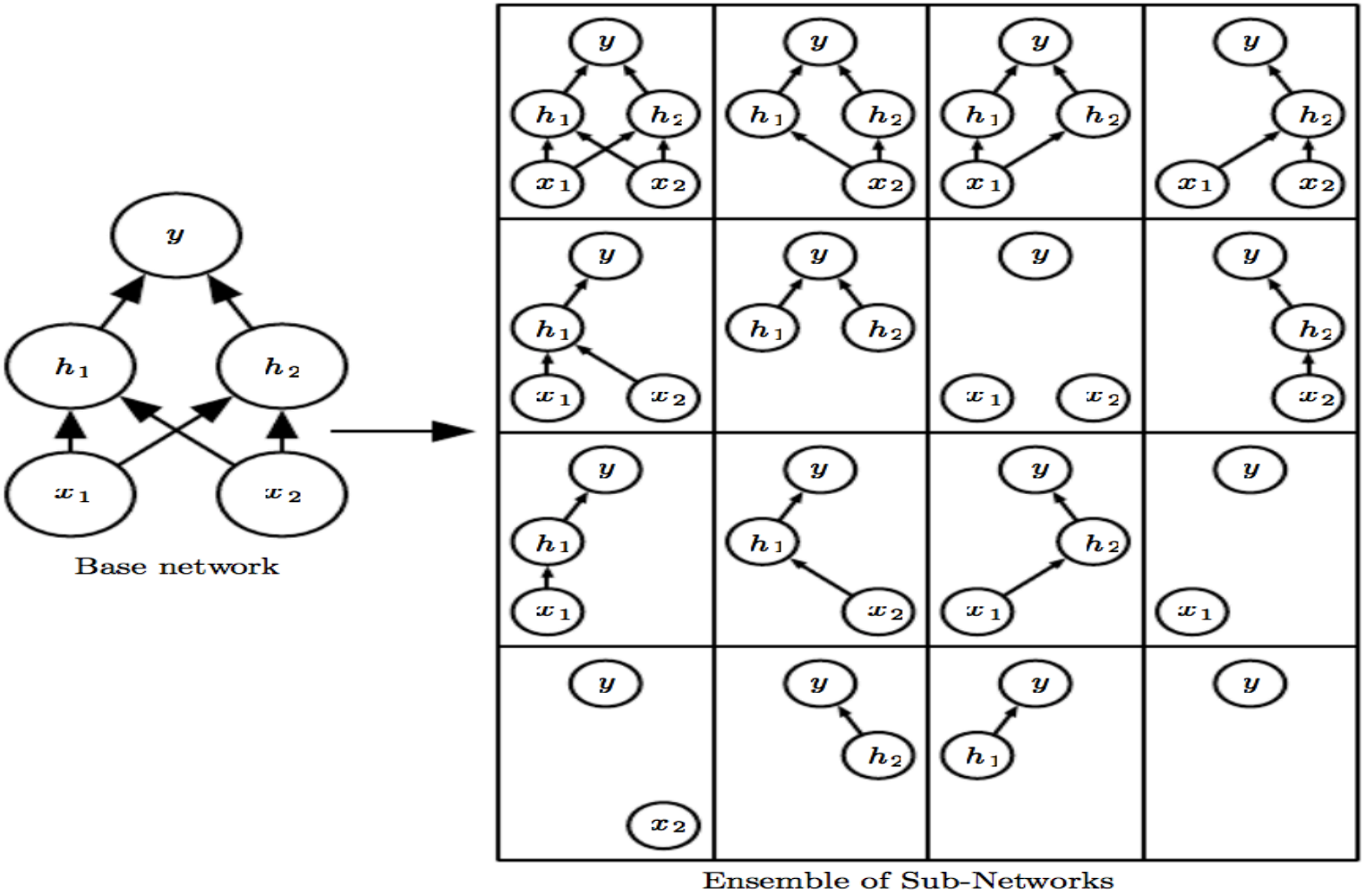
Figure 3: Comparison of the basic operations of a standard and dropout network.

$$z_i^{(l+1)} = \mathbf{w}_i^{(l+1)} \mathbf{y}^l + b_i^{(l+1)},$$
$$y_i^{(l+1)} = f(z_i^{(l+1)}),$$

$$r_j^{(l)} \sim \text{Bernoulli}(p),$$
$$\tilde{\mathbf{y}}^{(l)} = \mathbf{r}^{(l)} * \mathbf{y}^{(l)},$$
$$z_i^{(l+1)} = \mathbf{w}_i^{(l+1)} \tilde{\mathbf{y}}^l + b_i^{(l+1)},$$
$$y_i^{(l+1)} = f(z_i^{(l+1)}).$$

Srivastava et al., Dropout: A Simple Way to Prevent Neural Networks from Overfitting, 2014

# Регуляризация. Dropout (2)

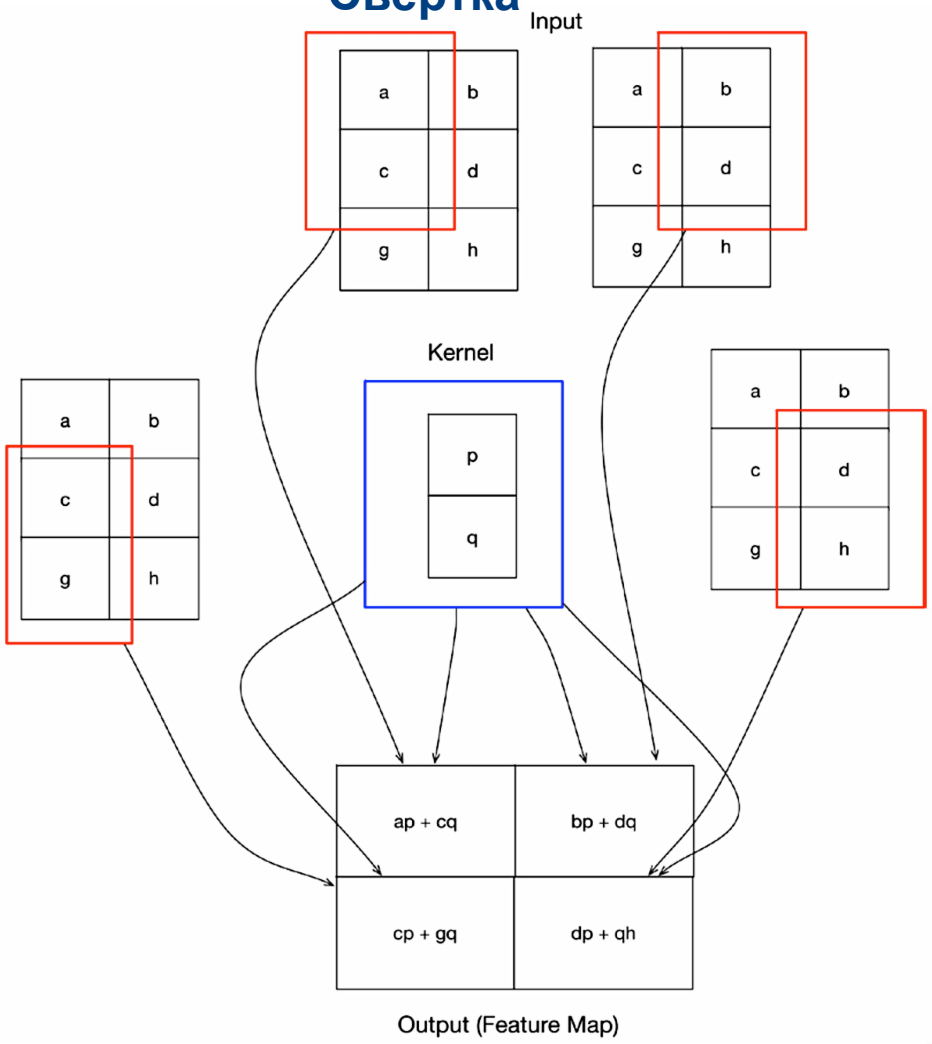



Goodfellow I., Bengio Y., Courville A., Deep learning

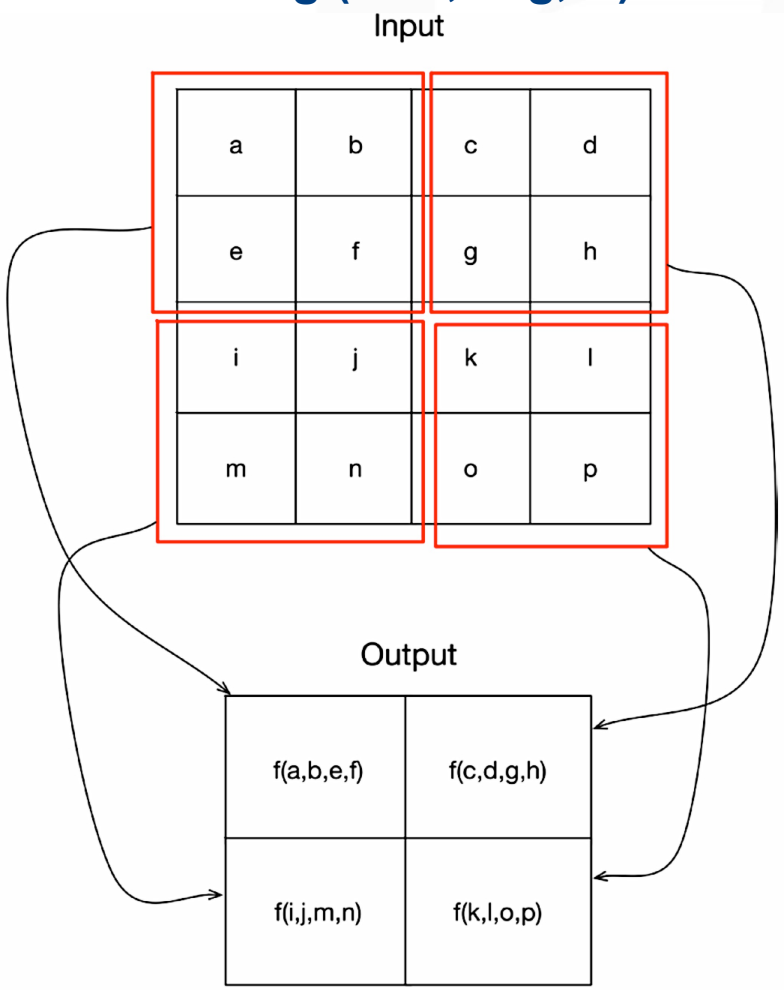
# Сверточные нейронные сети

# Основные элементы сверточной нейронной сети (ConvNet)

## Свертка



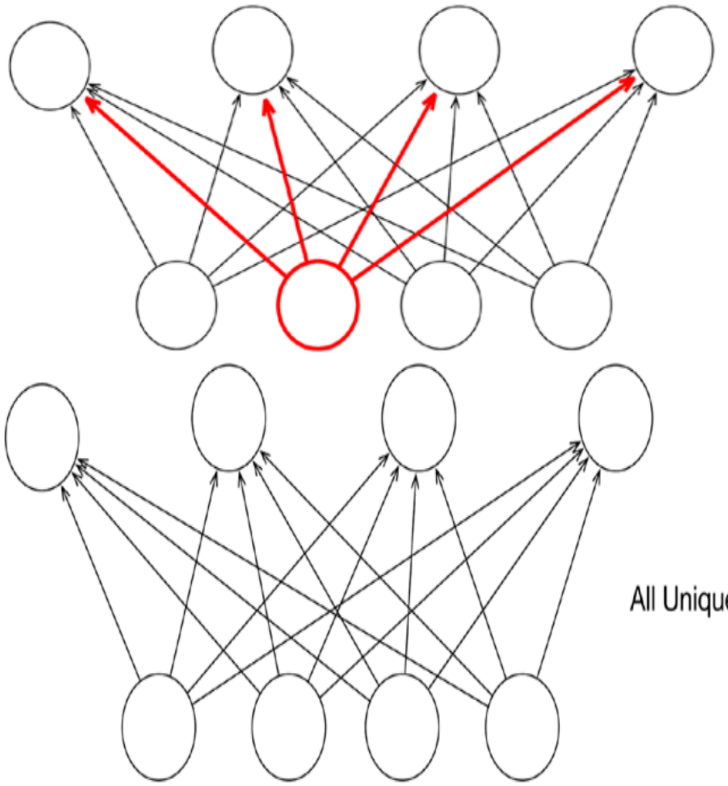

## Pooling (max, avg,…)



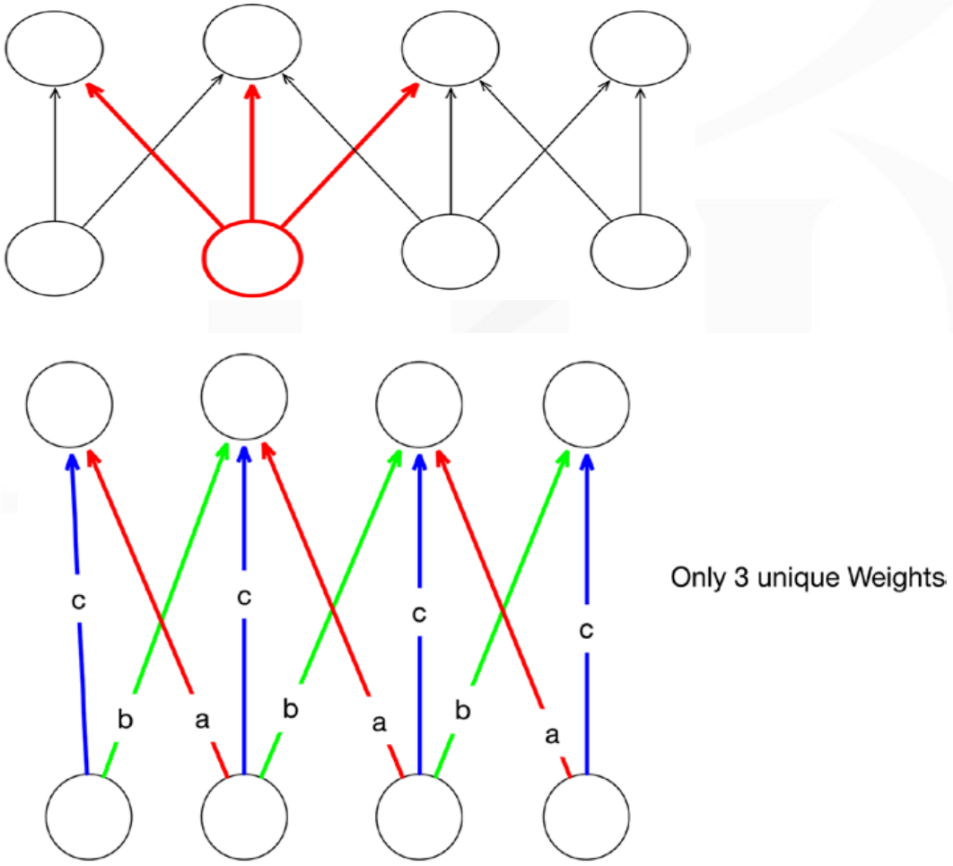

Ketkar “Deep Learning with Python”

# Convolution vs fully connected layer

## Полносвязный слой

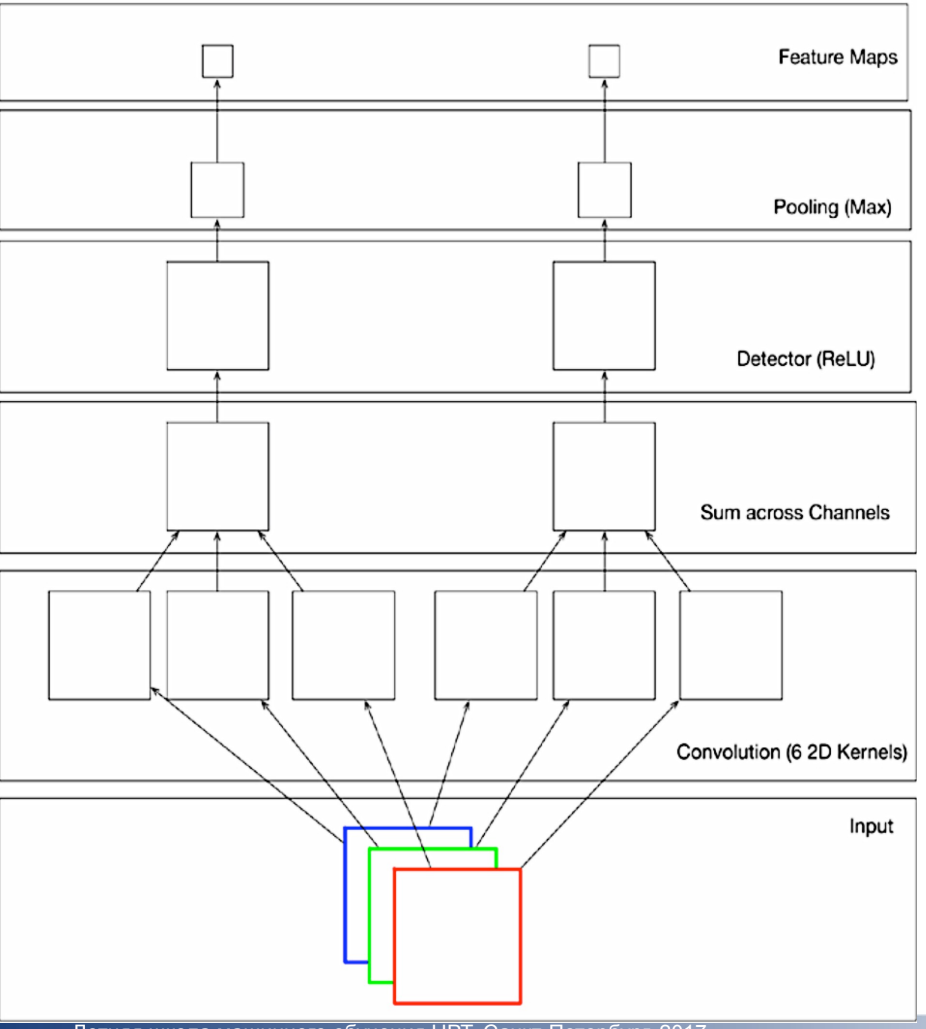



## Сверточный слой

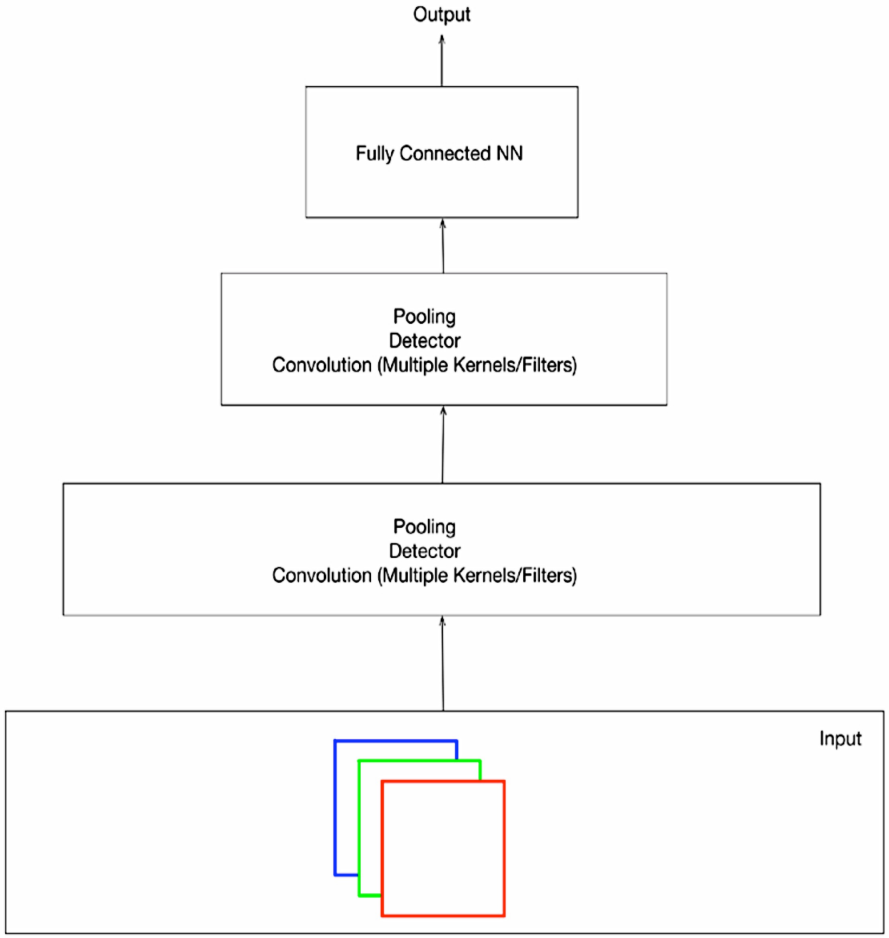



Ketkar “Deep Learning with Python”

## Convolution-detector-pooling block



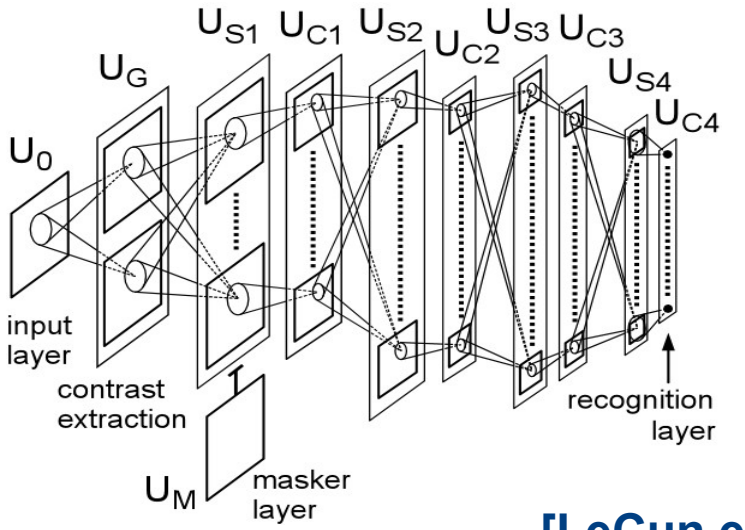

## Complete ConvNet

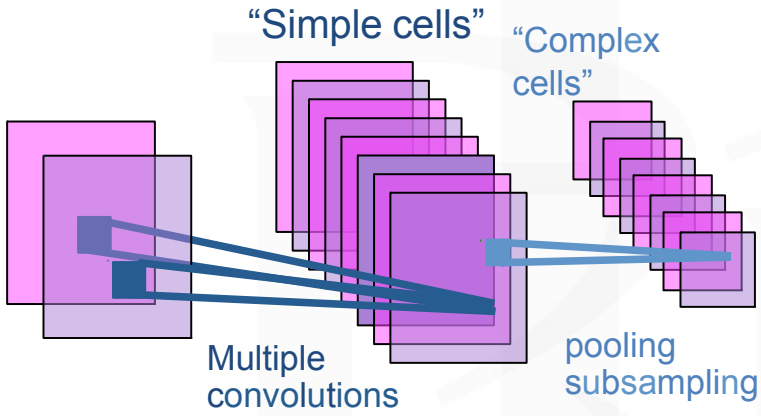



Ketkar "Deep Learning with Python"

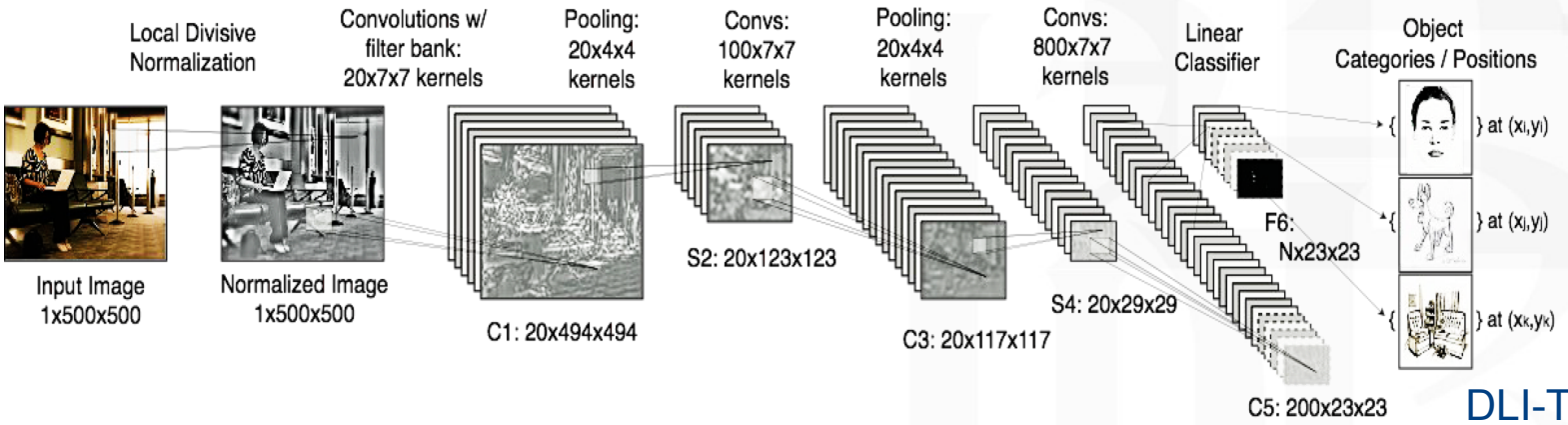
# Немного истории (1)

## [Hubel & Wiesel 1962], Cognitron & Neocognitron [Fukushima 1974-1982]




## [LeCun et al. 89], [LeCun et al. 98]




DLI-Teaching-Kit

**А потом появились:**

- **ImageNet [Fei-Fei et al. 2012]**
  - **1.5 миллиона изображений**
  - **1000 категорий (классов)**
- **Быстрые NVIDIA Graphical Processing Units (GPU)**
  - **1 триллион операций/сек.**


DLI-Teaching-Kit

Matchstick

Sea lion

Flute

Strawberry

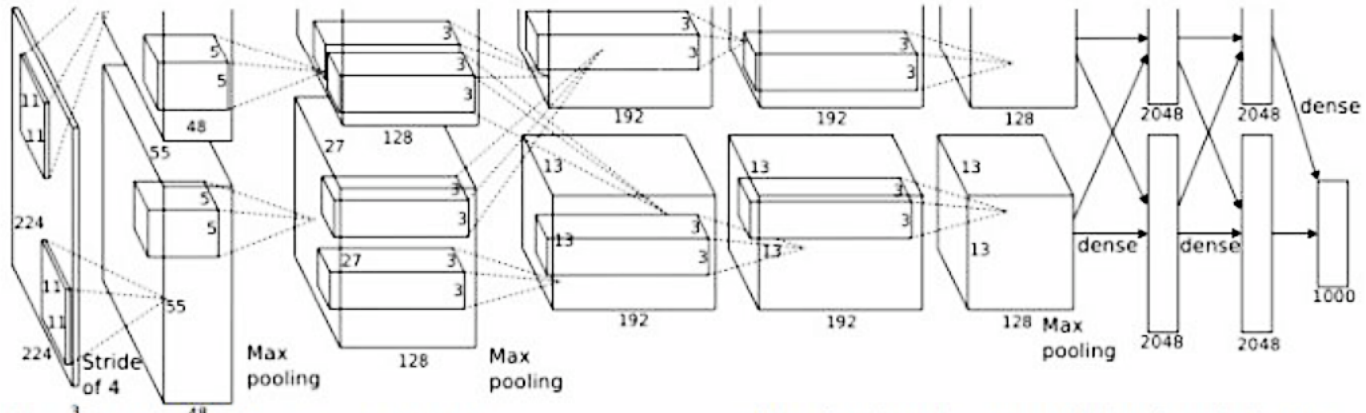
Bathing cap

Backpack

Racket

# Немного истории (3). AlexNet - революция в компьютерном зрении

**[Krizhevsky, Sutskever, Hinton 2012]**




- **Глубокая нейронная сеть, обученная с помощью обратного распространения ошибки на NVIDIA GPU «with all the tricks Yann came up with in the last 20 years, plus dropout" (Hinton, NIPS 2012)»**
- **Top-5 Error rate: 15%. Предыдущий state of the art: 25%**

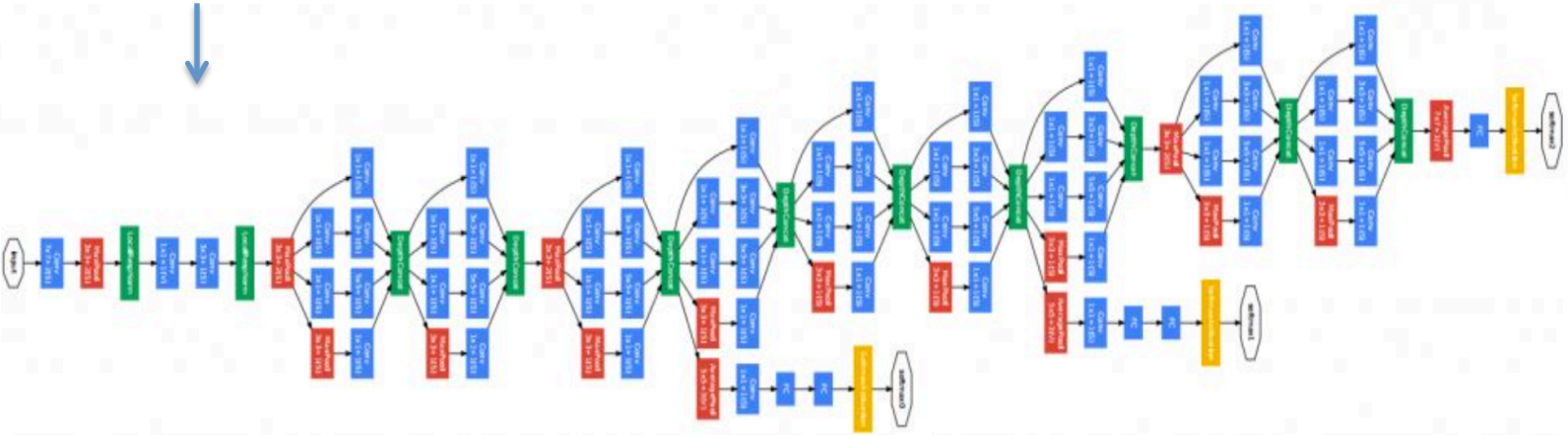
- **Куплен Google в январе 2013, появилась в Google+ Photo Tagging в мае 2013**

# Современные архитектуры сетей (1)

**Сложность сетей растет**

**GoogLeNet**

**VGGNet**

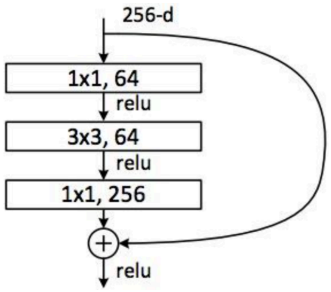


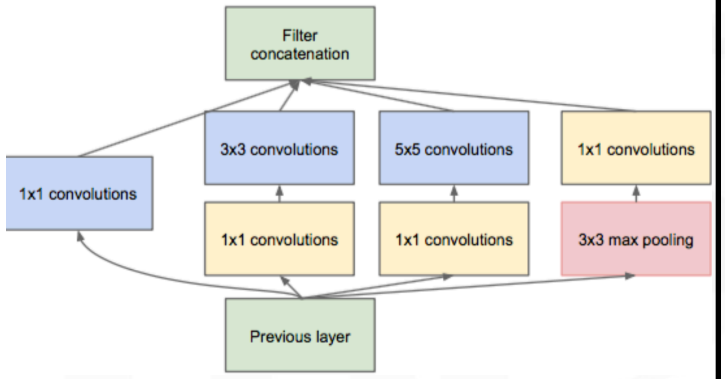
http://cs.unc.edu/~wliu/papers/GoogLeNet.pdf

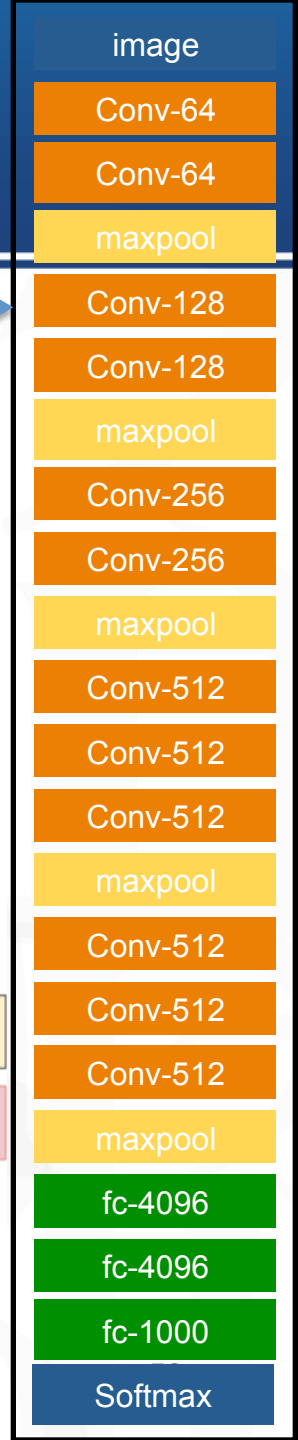
- **VGGNet (16, 19)**
- **GoogLeNet**
- **Inception (v1-3)**
- **ResNet**
- **Wide ResNet**

**…**

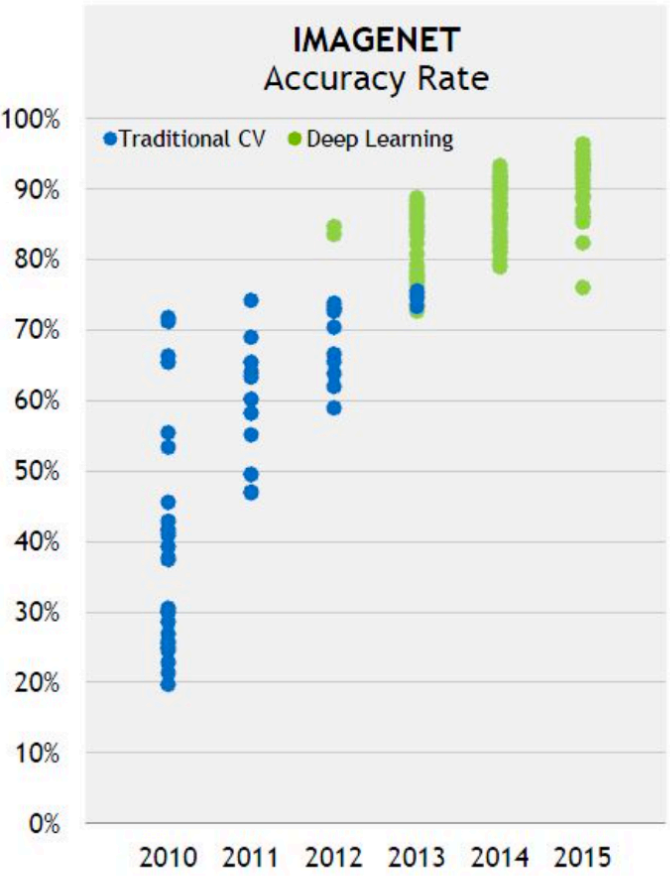
**содержат специальные блоки слоев**

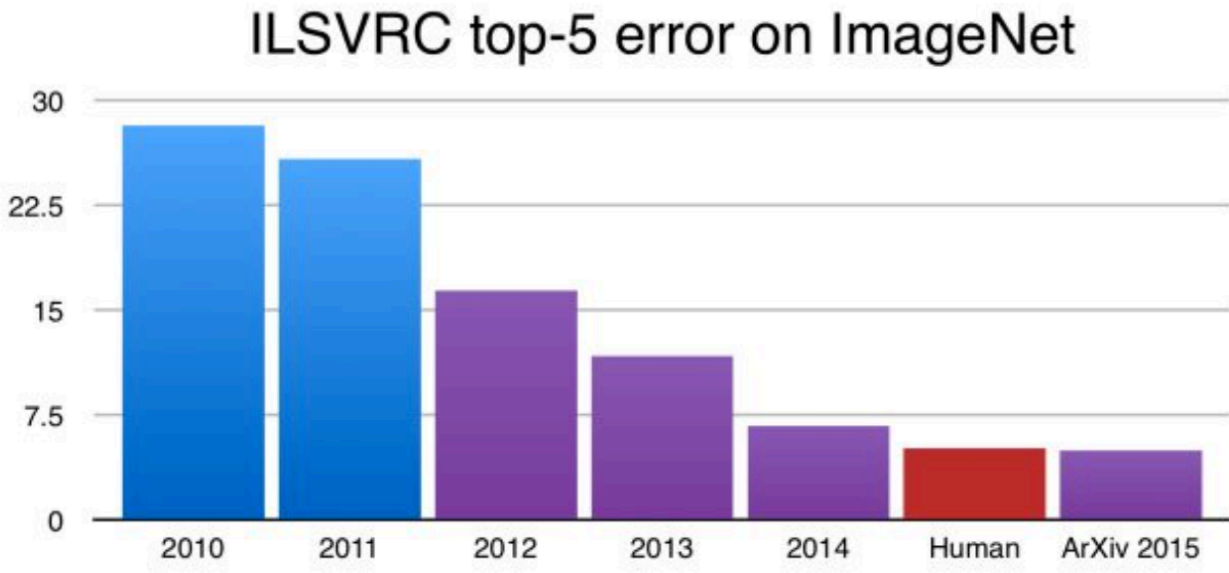


# Современные архитектуры сетей (2)



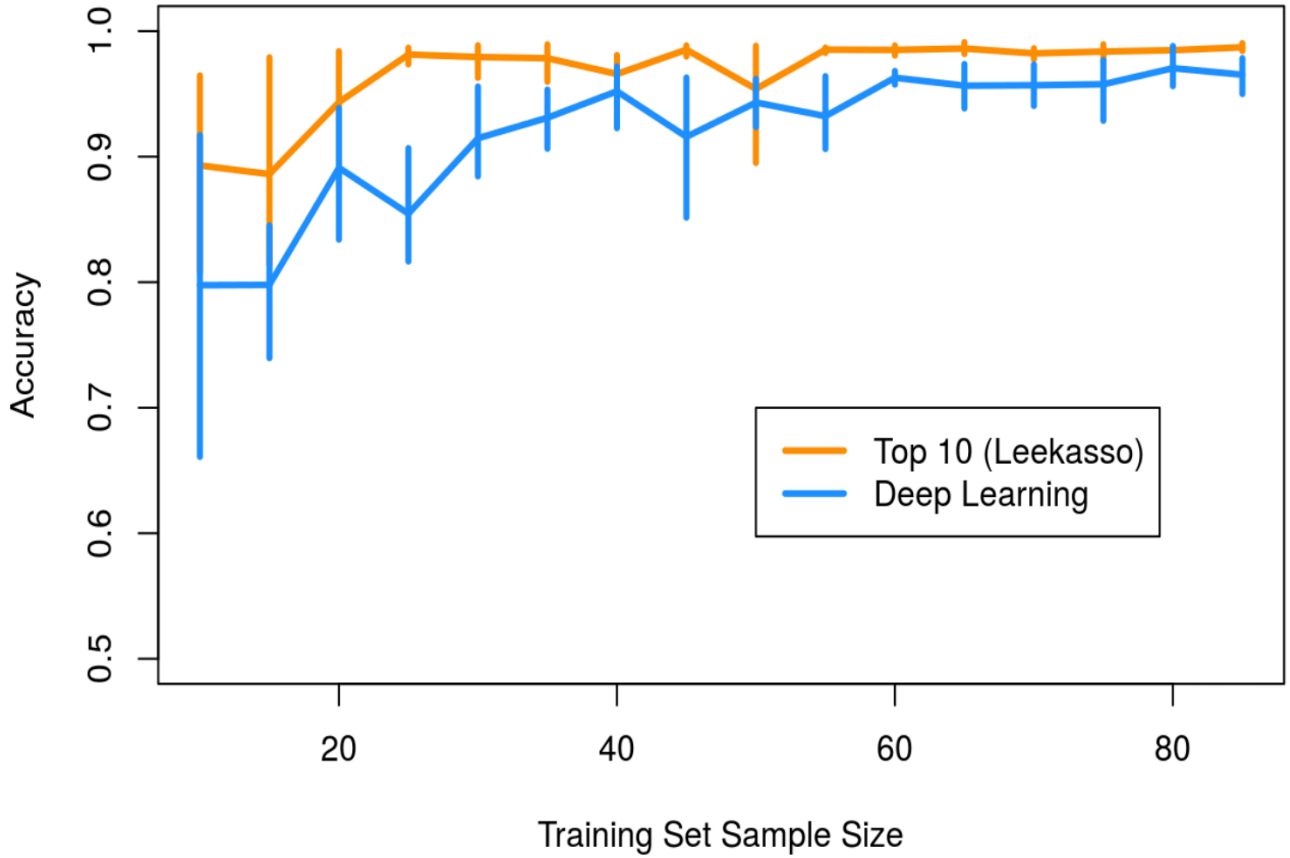

- Blue: Traditional CV
- Purple: Deep Learning
- Red: Human

Sapunov
http://arxiv.org/abs/1502.01852

# Проблемы ConvNet

# Проблема малых обучающих выборок (1)

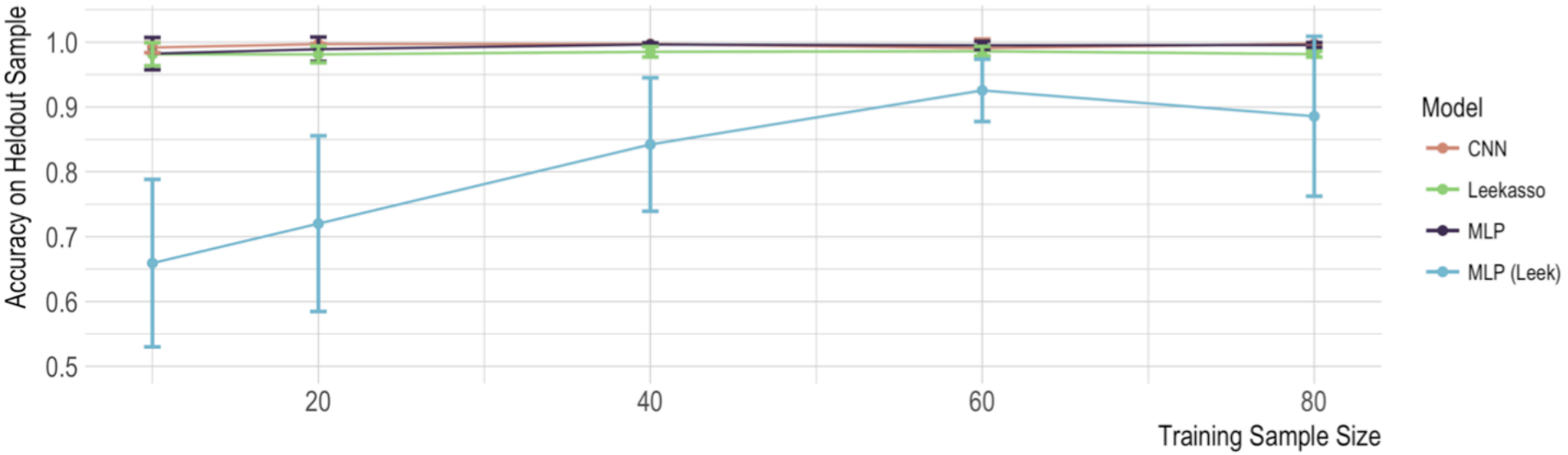
## Пример: Распознавание цифр «0» и «1» (MNIST)




## Bias/variance tradeoff

Leek “Don’t use deep learning your data isn’t that big”

## Обучать сеть нужно правильно




1. **tanh->relu**
2. **SGD должен сойтись (20-50 эпох недостаточно)**
3. **Использовать dropout**
4. **Важен подбор параметров (но параметры по умолчанию в Keras – ok)**

http://beamandrew.github.io/deeplearning/2017/06/04/deep_learning_works.html

# Проблема малых обучающих выборок (3). Leek MLP vs Beam MLP (Keras)

**def get_mlp_leek(n_classes):**

```
model = Sequential()
model.add(Dense(160,
activation='tanh',input_shape=(784,)))
model.add(Dense(160, activation='tanh'))
model.add(Dense(160, activation='tanh'))
model.add(Dense(160, activation='tanh'))
model.add(Dense(n_classes,
activation='softmax'))
opt = SGD(lr=0.005, momentum=0.0,
decay=0.99, nesterov=True)
model.compile(optimizer=opt,
loss='categorical_crossentropy',
                    metrics=[‘accuracy’])
return model
```

**def get_mlp(n_classes):**

```
model = Sequential()
model.add(Dense(128,
activation='relu',input_shape=(784,)))
model.add(Dropout(0.5))
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(n_classes,
activation='softmax'))
model.compile(optimizer='Adam',
                    loss='categorical_crossentropy
                      metrics=['accuracy'])
return model
```

https://github.com/beamandrew/deep_learning_works/blob/master/mnist.py

# Проблема малых обучающих выборок (4). CNN (Keras)

**def get_cnn(n_classes):**

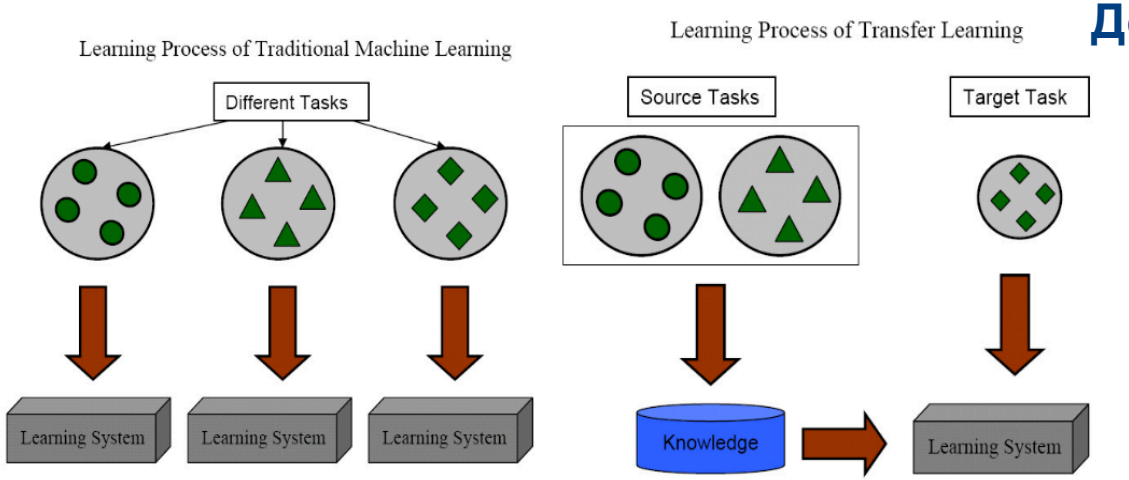
```
model = Sequential()
model.add(Convolution2D(32, 3, 3, activation='relu',
input_shape=(28,28,1)))
model.add(Convolution2D(32, 3, 3, activation='relu'))
model.add(MaxPooling2D())
model.add(Convolution2D(64, 3, 3, activation='relu'))
model.add(Convolution2D(64, 3, 3, activation='relu'))
model.add(MaxPooling2D())
model.add(Flatten())
model.add(Dropout(0.2))
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(n_classes, activation='softmax'))
model.compile(optimizer='Adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
return model
```

**Train:**

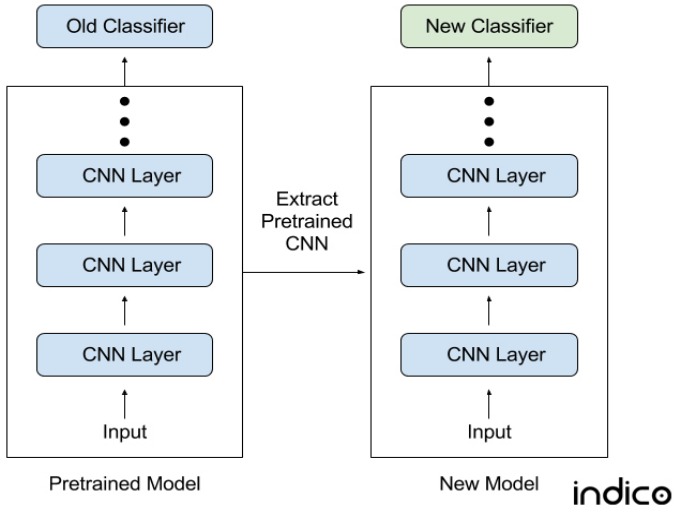
```
model = get_cnn(n_classes)

model.fit(X_train_fold,
one_hot_fold, nb_epoch=200)

score =model.evaluate
(X_final_test, one_hot_final_test,
batch_size=128))
```

https://github.com/beamandrew/deep_learning_works/blob/master/mnist.py

# Transfer learning (1)




## Дообучение сети (fine-tuning)

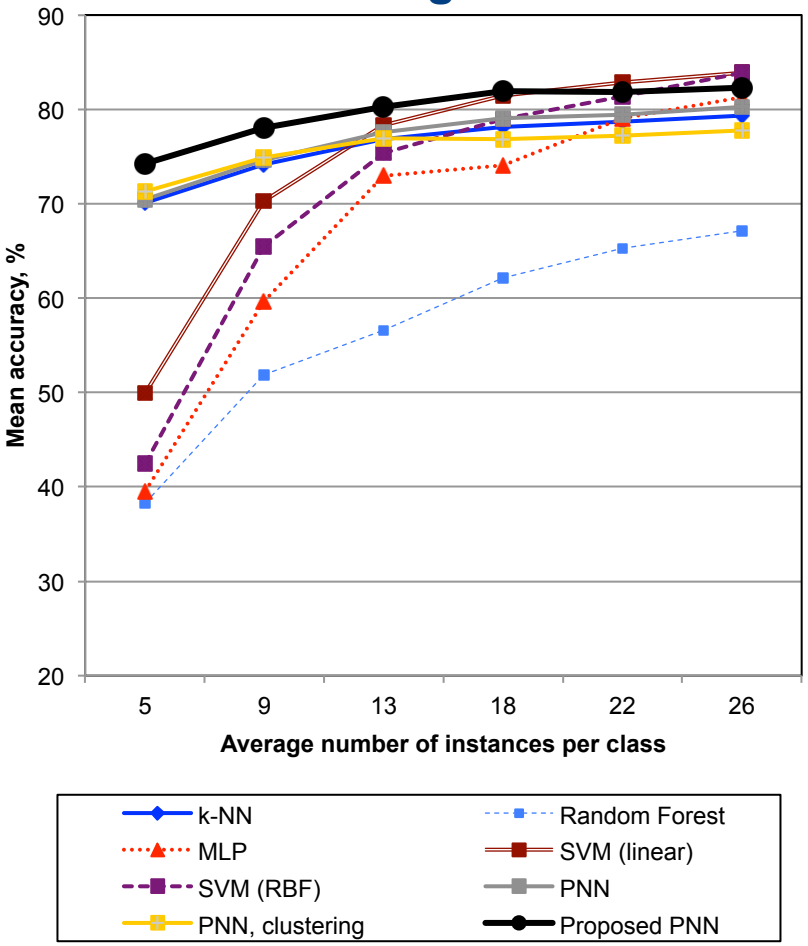



**OverFeat features → Trained classifier on other data sets [Razavian, Azizpour, Sullivan, Carlsson "CNN features off-the-shelf: An astounding baseline for recogniton", CVPR 2014], http://www.csc.kth.se/cvap/cvg/DL/ots/**

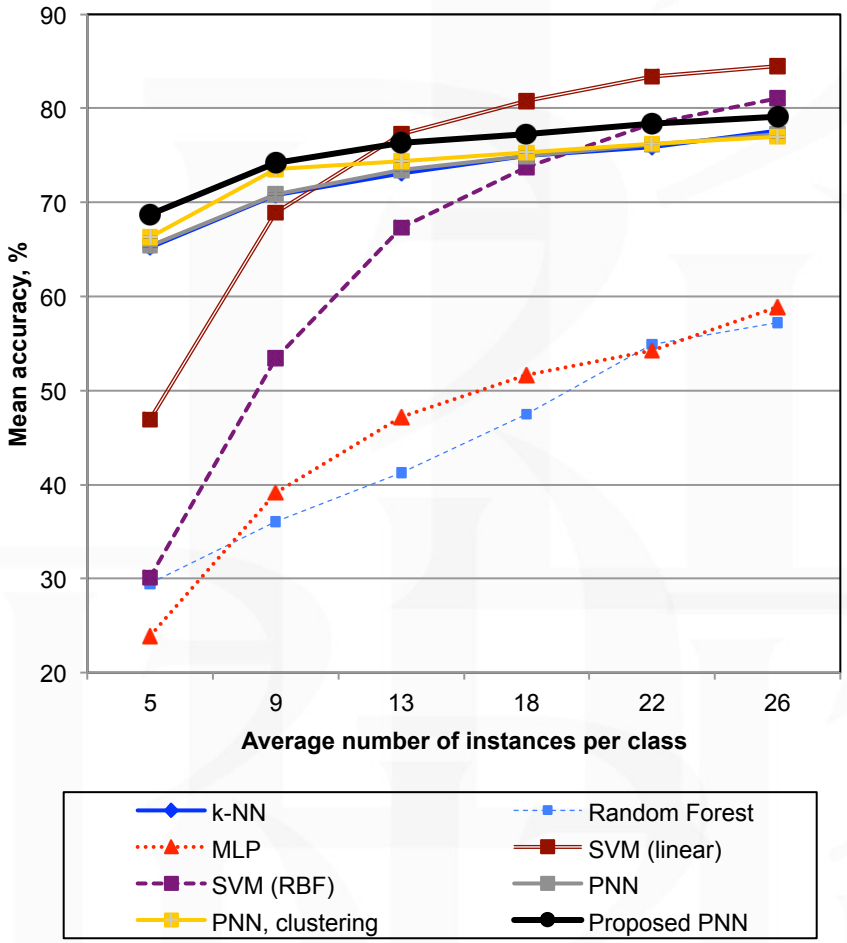
| | | | |
|---|---|---|---|
| • image classification | ImageNet LSVRC 2013 | competitive | 13.6 % error |
| | Dogs vs Cats Kaggle challenge 2014 | state of the art | 98.9% |
| • object localization | ImageNet LSVRC 2013 | state of the art | 29.9% error |
| • object detection | ImageNet LSVRC 2013 | state of the art | 24.3% mAP |

# Transfer learning (2). Пример (Caltech-101)

## GoogLeNet




## VGG-19-Net




https://arxiv.org/abs/1708.02733

# Сжатие сверточных сетей (1)

[Han et al. 2016] Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding – **ICLR16 Best paper**

- **Pruning**
  [Han et al. 2016], [Molchanov et al. 2016]
- **Distillation The Knowledge (FitNet)**
  [Hinton et al. 2014], [Romero et al. 2014]
- **Weights Hashing / Quantization**
  [Chen et al. 2015], [Han et al. 2016]
- **Tensor Decompositions**
  [Lebedev et al. 2015], [Kim et al. 2015], [Novikov et al. 2015], [Garipov et al. 2016]
- **Binarization**
  [Courbariaux / Hubara et al. 2016], [Rastegari et al. 2016], [Merolla et al. 2016], [Hou et al. 2016]
- **Architectural tricks**
  [Hong et al. 2016], [Iandola et al. 2016], [Teerapittayanon et al. 2016]

[Rassadin, Savchenko, 2017]

# Сжатие сверточных сетей (2).
# Распознавание эмоций (Radboud Faces Database, RaFD)

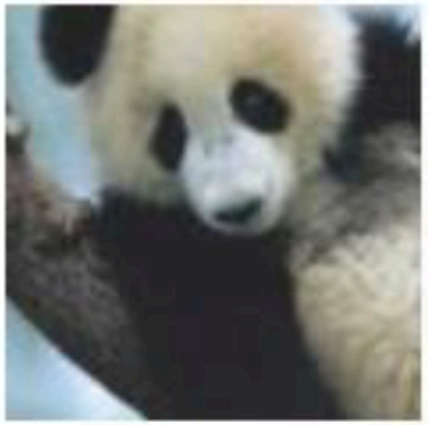
| | Training time per epoch, ms | Inference time, ms | Model size, MB | Accuracy, % |
|---|---|---|---|---|
| VGG-S (baseline) | 43.7 | 33.4 | 372.2 | 97.13 |
| SqueezeNet-1.1 (baseline) | **22.94** | **4.94** | 2.8 | 89.14 |
| SqueezeNet-1.1, CP-Decomposition | **22.94** | 7.74 | **2.1** | 87.5 |
| HashedNets | 294.8 | 158.2 | 68.3 | 96.31 |
| Binary-Weight-Network (BWN) | 83.8 | 33.5 | 11.6 | **98.57** |
| XNOR-Net | 84.3 | 34.2 | 11.6 | 58.81 |
| XNOR-Net w/o weights activation | 43.4 | 34.1 | 11.6 | 88.32 |

[Rassadin, Savchenko, 2017]

# Уязвимости сверточных сетей (1). Adversarial examples

$+ .007 \times$

$=$



$\boldsymbol{x}$

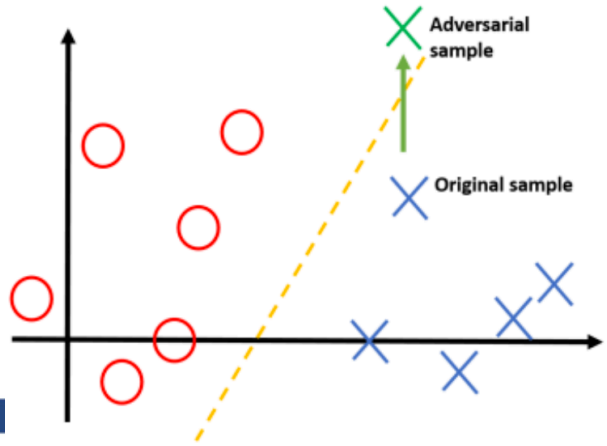
"panda"
57.7% confidence

$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"nematode"
8.2% confidence

$\boldsymbol{x} + \epsilon \text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"gibbon"
99.3 % confidence




Goodfellow I., Bengio Y., Courville A., Deep learning,
[Szegedy et al., 2014]

# Уязвимости сверточных сетей (2)

## ImageNet

| Attack Method | VGG-16 | AlexNet | ResNet-50 | Inception v3 |
|---|---|---|---|---|
| FGS (Fast gradient sign) | 0.85/0.56/0.61 | 0.56/0.22/0.33 | 0.88/0.53/0.62 | **0.92**/0.01/0.07 |
| Black-Box | 0.85/0.61 | 0.56/0.43 | 0.88/0.69 | **0.92**/0.19 |

## CIFAR-10

| Attack Method | VGG-16 | ResNet-50 |
|---|---|---|
| FGS | 0.79/0.03/0.36 | **0.81**/0.05/0.43 |
| Black-Box | 0.79/0.44 | **0.81**/0.51 |

## CIFAR-100

| Attack Method | VGG-16 | ResNet-50 |
|---|---|---|
| FGS | 0.64/0.15/0.36 | **0.67**/0.29/0.43 |
| Black-Box | 0.64/0.41 | **0.67**/0.38 |

[Груздев, 2017]

# Распознавание объектов на видео

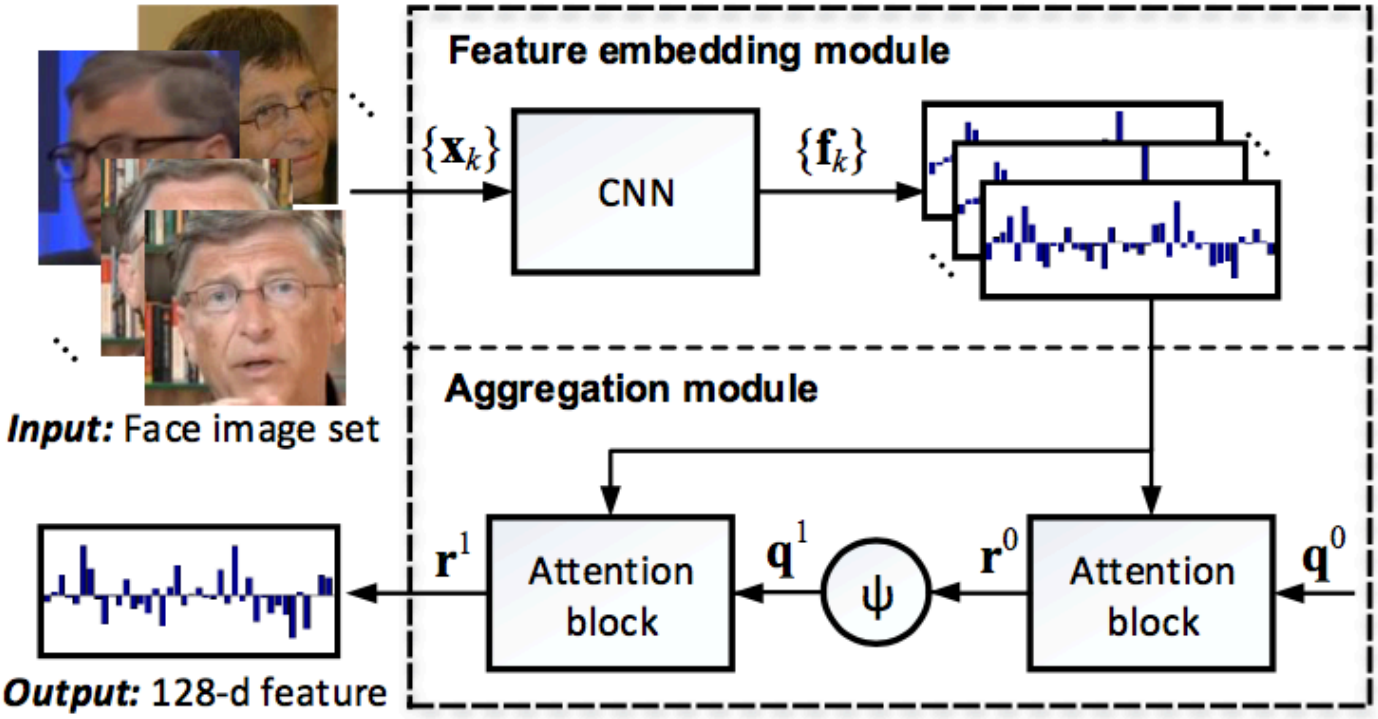
## Attention block

$$e_k = \mathbf{q}^T \mathbf{f}_k$$

$$a_k = \frac{\exp(e_k)}{\sum_j \exp(e_j)}$$

## Агрегация признаков кадров

$$\mathbf{r} = \sum_k a_k \mathbf{f}_k$$




## Несколько последовательных блоков attention

$$\mathbf{q}^1 = \tanh(\mathbf{W}\mathbf{r}^0 + \mathbf{b})$$

https://arxiv.org/abs/1603.05474

# Распознавание тегов на видео (1). Youtube-8M

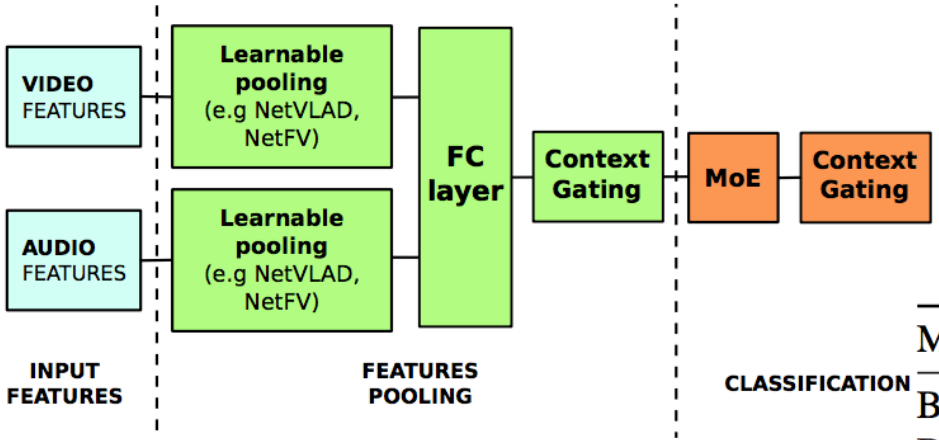
## 8M видео, 4716 тегов

**Groundtruth:** Car - Vehicle - Sport Utility Vehicle - Dacia Duster - Renault - 4-Wheel Drive
**Top 6 scores:** Car - Vehicle - Sport Utility Vehicle - Dacia Duster - Fiat Automobiles Volkswagen Beetles

**Groundtruth:** Tree- Christmas Tree - Christmas Decoration - Christmas
**Top 6 scores:** Christmas - Christmas decoration - Origami - Paper - Tree Christmas Tree

https://arxiv.org/abs/1706.06905

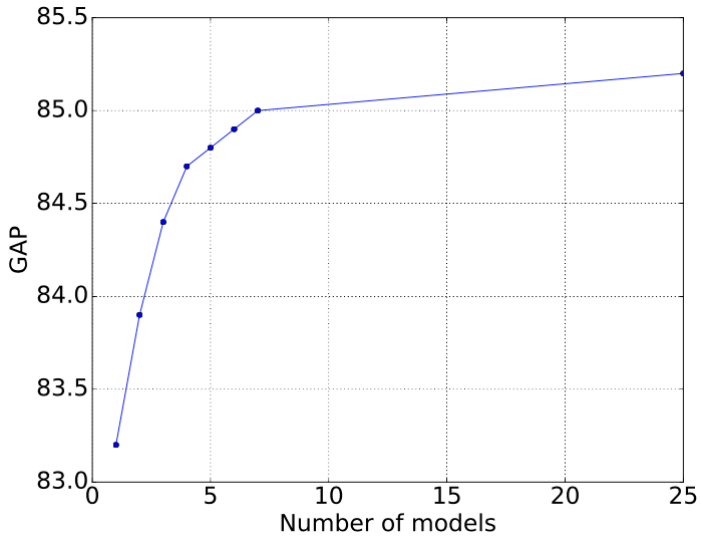
# Распознавание тегов на видео (2). Победитель




## Context Gating

$$Y = \sigma(WX + b) \circ X$$

## Ансамбль моделей

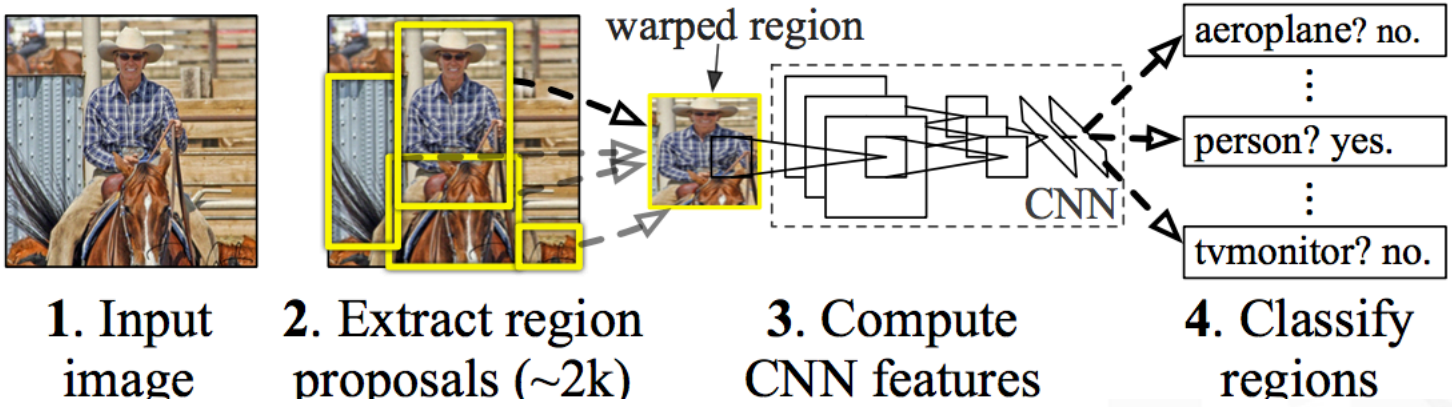



| Method | GAP |
|---|---|
| Baseline 1 (Average pooling + Logistic Regression) | 71.4% |
| Baseline 2 (Average pooling + MoE + CG) | 74.1% |
| LSTM (2 Layers) | 81.7% |
| GRU (2 Layers) | 82.0% |
| Soft-DBoW (4096 Clusters) | 81.6% |
| NetFV (128 Clusters) | 82.2% |
| NetVLAD (256 Clusters) | 82.4% |
| Gated Soft-DBoW (4096 Clusters) | 82.0% |
| Gated NetFV (128 Clusters) | 83.0% |
| Gated NetRVLAD (256 Clusters) | 83.1% |
| Gated NetVLAD (256 Clusters) | **83.2%** |

https://arxiv.org/abs/1706.06905
https://github.com/antoine77340/LOUPE

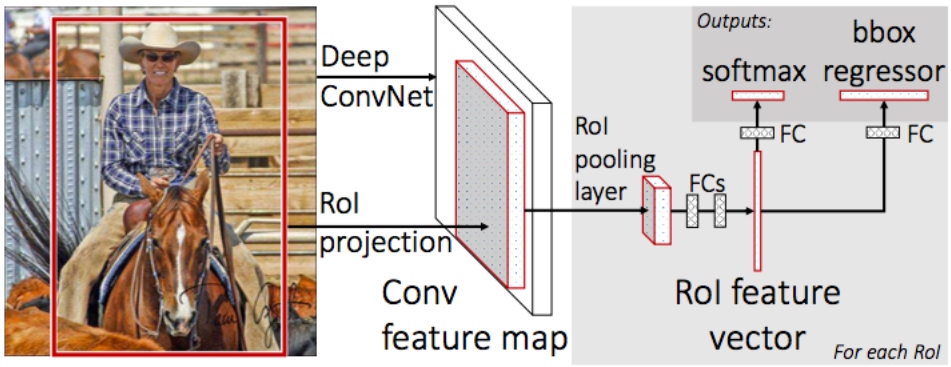
# Детектирование объектов

## R-CNN

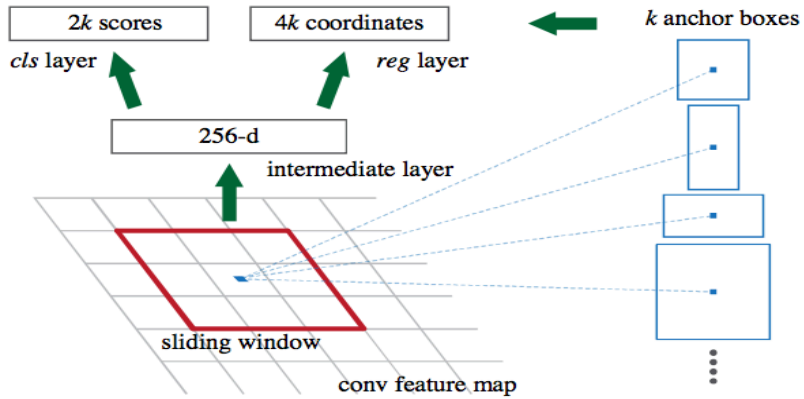



3 – признаки из AlexNet
4 – SVM
В конце – non-maximum suppression (для каждого класса)

## Fast R-CNN



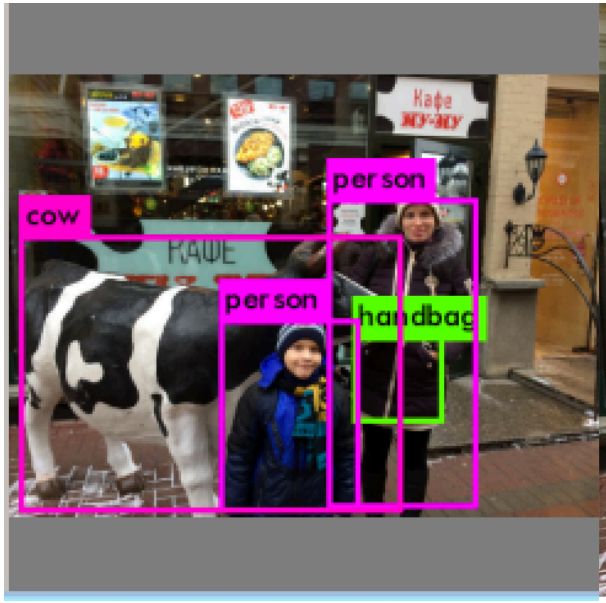

## Faster R-CNN with region proposal network

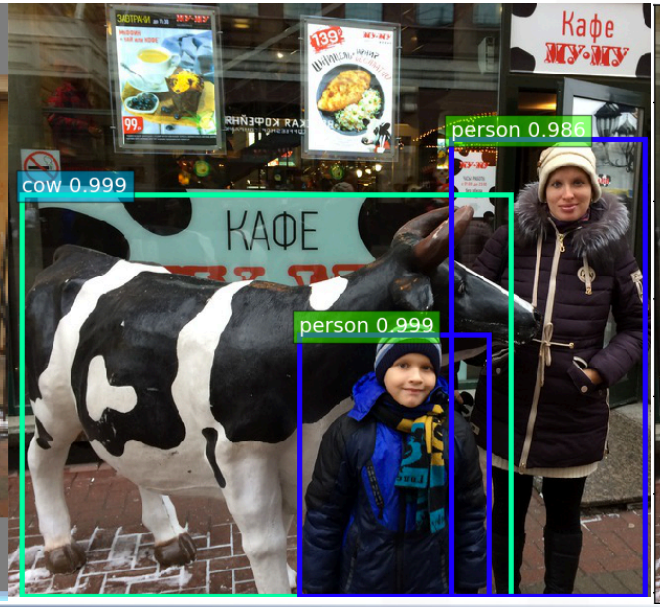


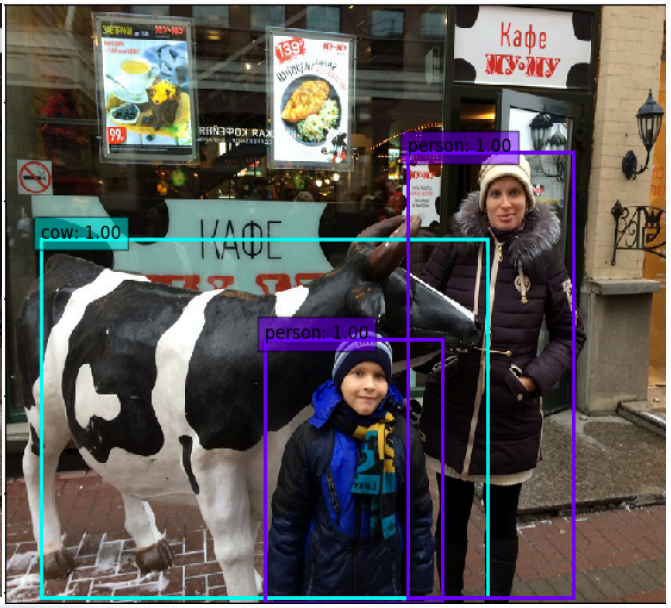

[Girshick et al, 2013], [Girshick et al, 2015], [Ren et al, 2016]

# R-CNN (Region Convolutional Neural Network)

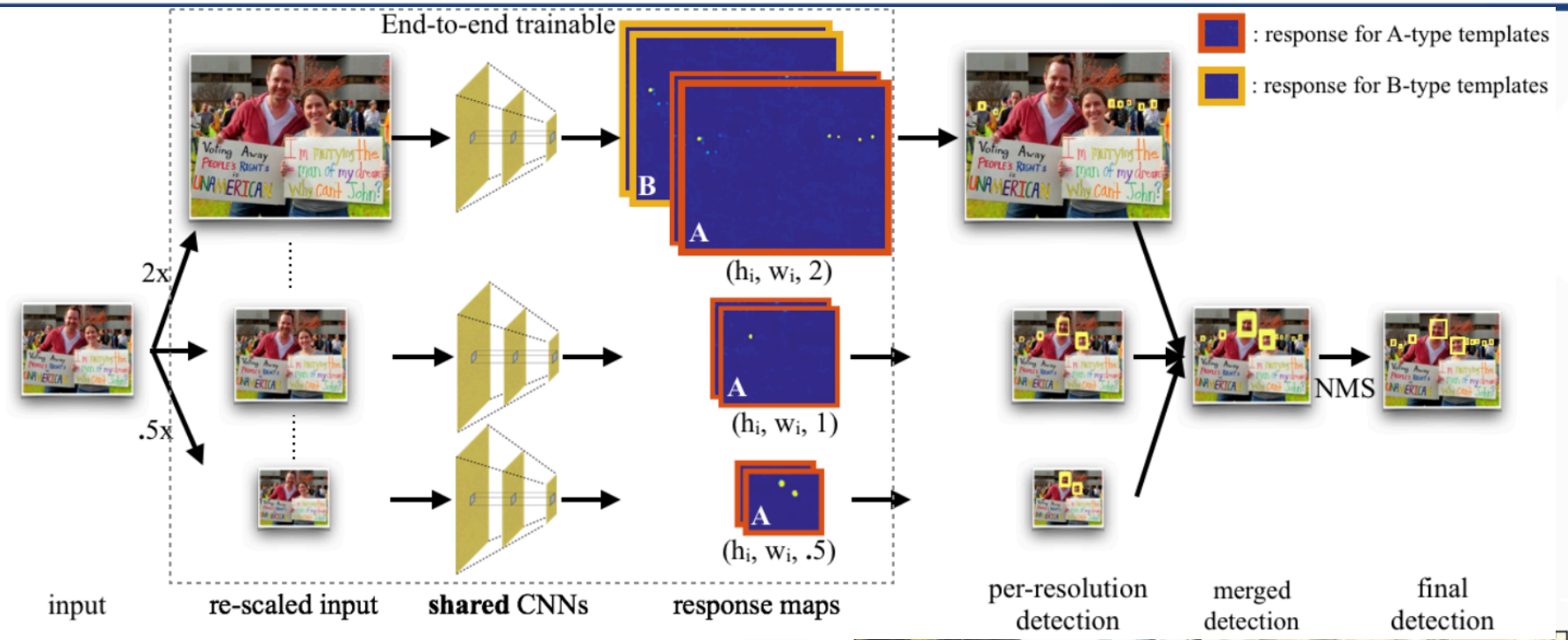
| YOLO (You only look once)+Darknet | PVANet | SSD: Single shot multibox detector |
|---|---|---|
| 5.93 s.<br>handbag: 29%, person: 75%, person: 84%, cow: 87% | 1.279s, 200 object proposals<br>('cow', 0.99929786)<br>('person', 0.99882239)<br>('person', 0.98628533) | 6.330s, 200 object proposals<br>cow: 1.00, person: 1.00, person: 1.00 |
| arxiv.org/abs/1612.08242 | arxiv.org/abs/1611.08588 | arxiv.org/abs/1611.08588 |



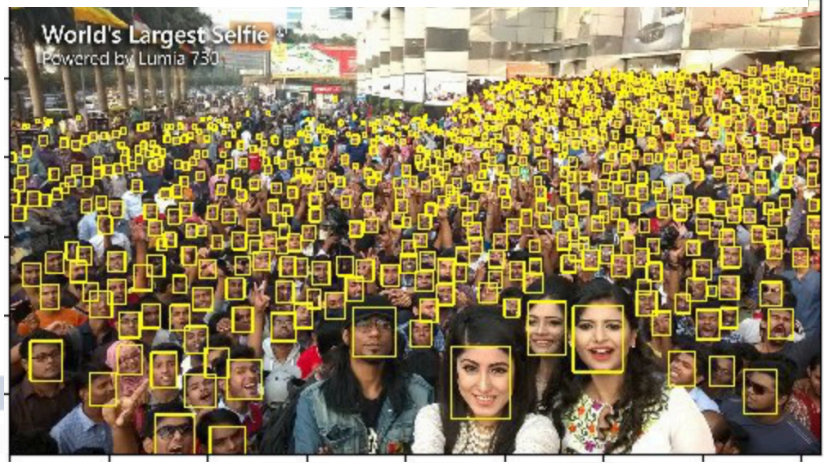



# Нейросетевое детектирование лиц




Tiny Face Detector, CVPR 2017, https://www.cs.cmu.edu/~peiyunh/tiny/



НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

# Спасибо за внимание!