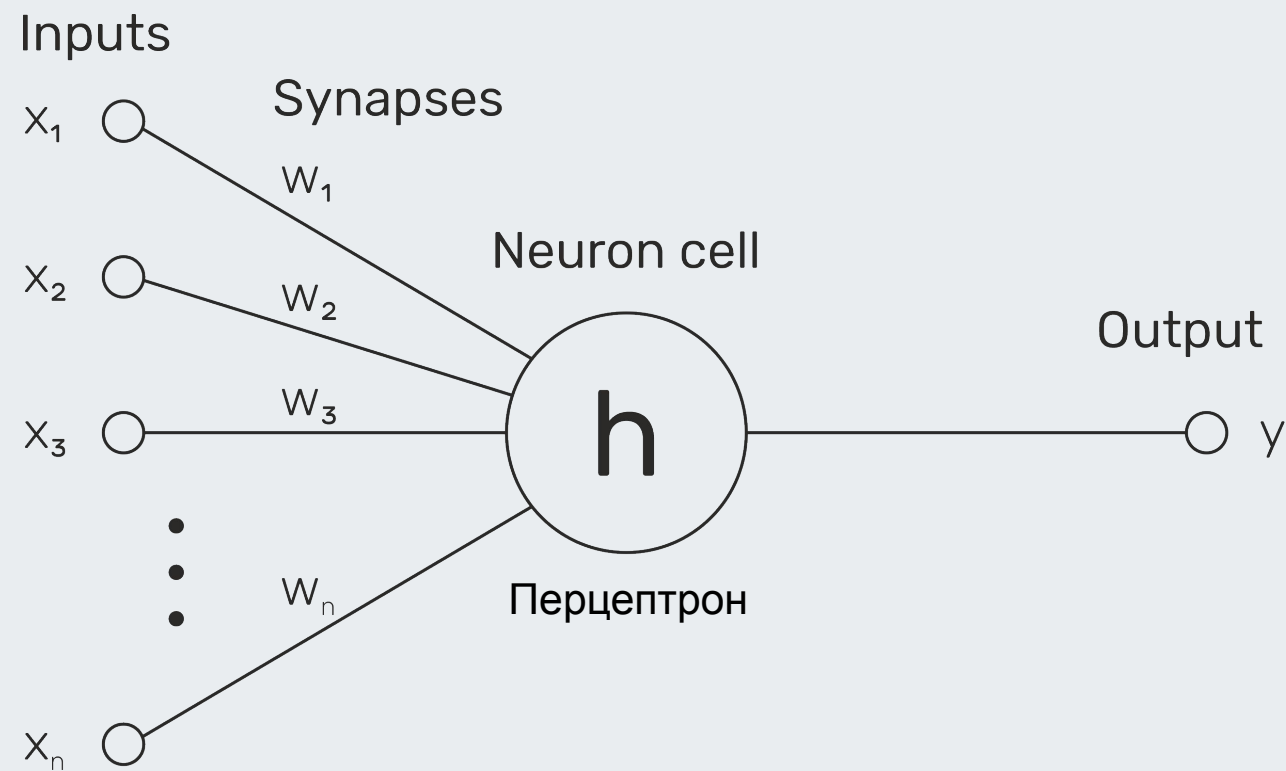




Нейронные сети, обратное распространение ошибки, Свертка

Преподаватель: Герард Костин

Модель Нейрона

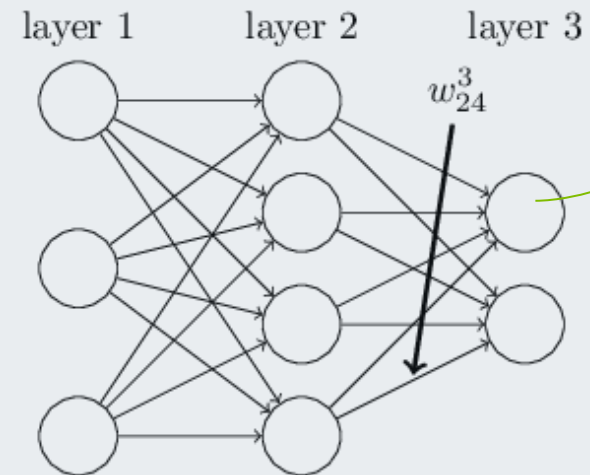
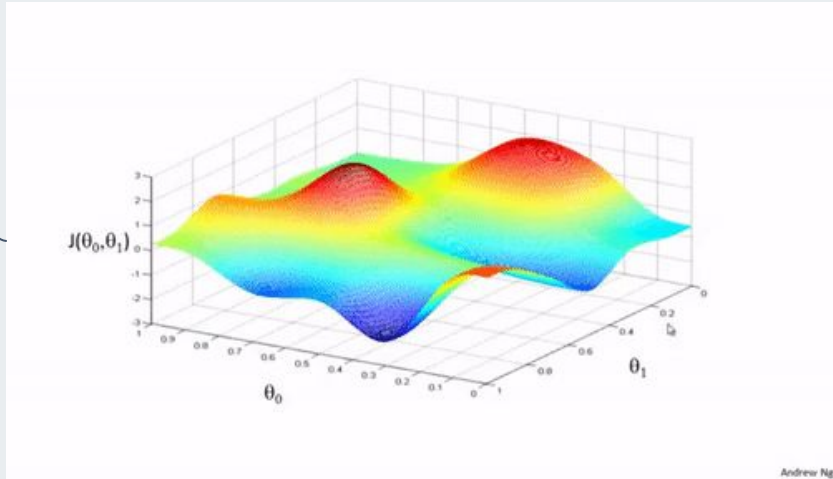


$$h = \sum_i x_i * w_i \quad y = f(h)$$

Backpropagation

Cost function

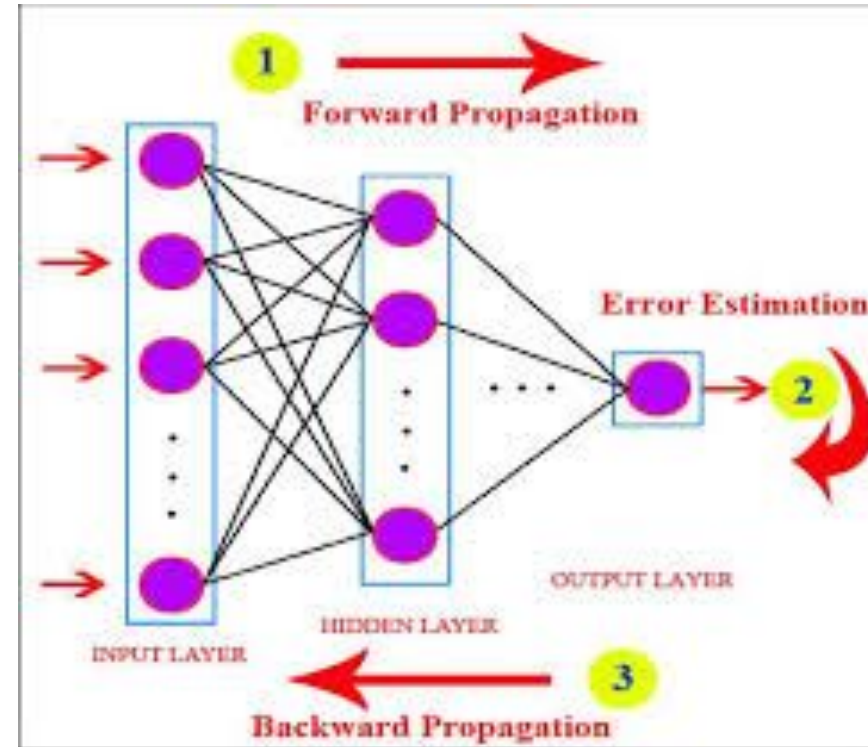
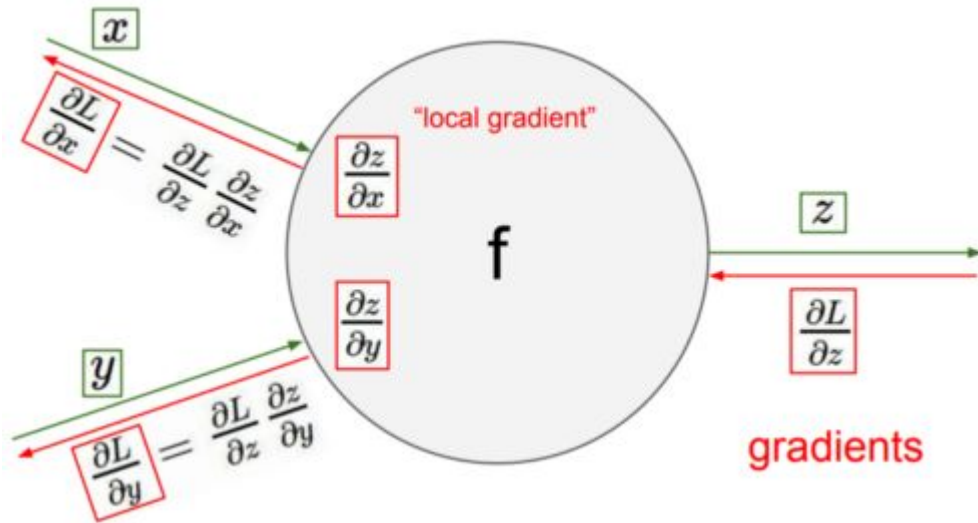
$$C = \frac{1}{2} \|y - a^L\|^2 = \frac{1}{2} \sum_j (y_j - a_j^L)^2$$



w_{jk}^l is the weight from the k^{th} neuron in the $(l-1)^{\text{th}}$ layer to the j^{th} neuron in the l^{th} layer

Backpropagation

Cost function

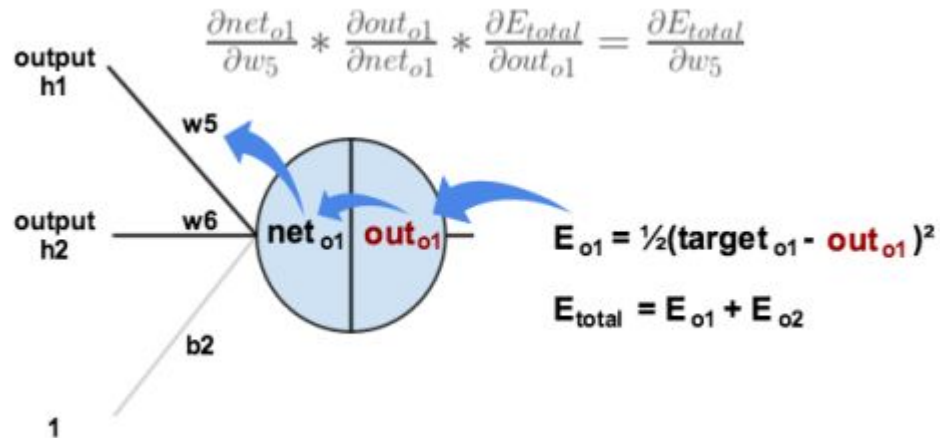


<http://neuralnetworksanddeeplearning.com/chap2.html>

Backpropagation

$$\Delta C \approx \sum_{mnp\dots q} \frac{\partial C}{\partial a_m^L} \frac{\partial a_m^L}{\partial a_n^{L-1}} \frac{\partial a_n^{L-1}}{\partial a_p^{L-2}} \cdots \frac{\partial a_q^{l+1}}{\partial a_j^l} \frac{\partial a_j^l}{\partial w_{jk}^l} \Delta w_{jk}^l$$

$$\frac{\partial C}{\partial w_{jk}^l} = \sum_{mnp\dots q} \frac{\partial C}{\partial a_m^L} \frac{\partial a_m^L}{\partial a_n^{L-1}} \frac{\partial a_n^{L-1}}{\partial a_p^{L-2}} \cdots \frac{\partial a_q^{l+1}}{\partial a_j^l} \frac{\partial a_j^l}{\partial w_{jk}^l}$$



<http://neuralnetworksanddeeplearning.com/chap2.html>

1. **Input a set of training examples**
2. **For each training example x :** Set the corresponding input activation $a^{x,1}$, and perform the following steps:
 - **Feedforward:** For each $l = 2, 3, \dots, L$ compute $z^{x,l} = w^l a^{x,l-1} + b^l$ and $a^{x,l} = \sigma(z^{x,l})$.
 - **Output error $\delta^{x,L}$:** Compute the vector $\delta^{x,L} = \nabla_a C_x \odot \sigma'(z^{x,L})$.
 - **Backpropagate the error:** For each $l = L - 1, L - 2, \dots, 2$ compute $\delta^{x,l} = ((w^{l+1})^T \delta^{x,l+1}) \odot \sigma'(z^{x,l})$.
3. **Gradient descent:** For each $l = L, L - 1, \dots, 2$ update the weights according to the rule $w^l \rightarrow w^l - \frac{\eta}{m} \sum_x \delta^{x,l} (a^{x,l-1})^T$, and the biases according to the rule $b^l \rightarrow b^l - \frac{\eta}{m} \sum_x \delta^{x,l}$.

Функции Активации

- Applying function to the neuron output:

$$N1 = x1 * w1 + x2 * w2 \quad \rightarrow \quad N1 = f(x) = f(x1 * w1 + x2 * w2)$$

- Sigmoid:

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

- ReLu (Rectified linear unit):

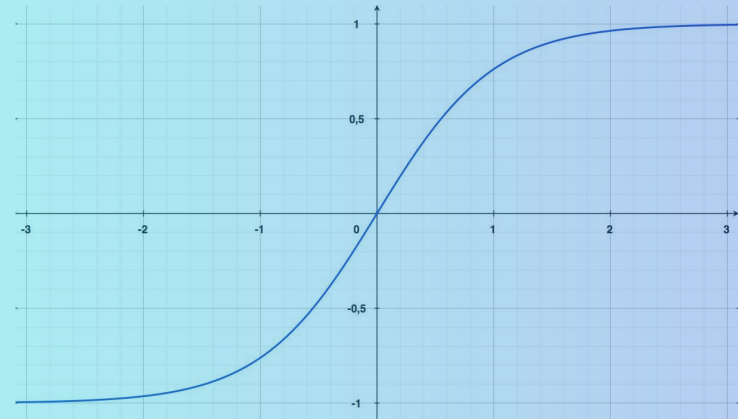
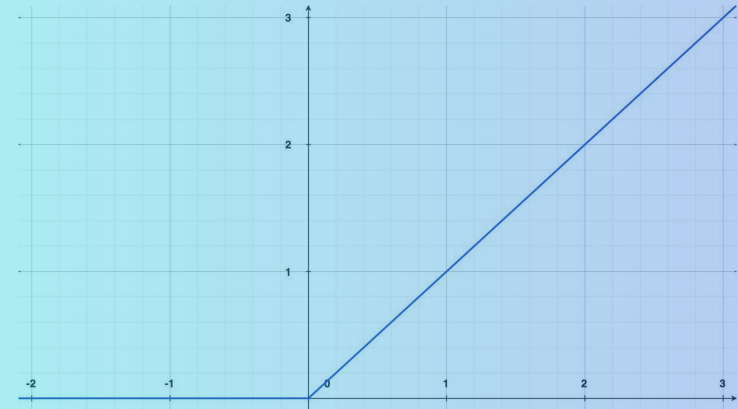
$$f(x) = \begin{cases} 0 & x < 0 \\ x & x \geq 0 \end{cases}$$

- tanh(Hyperbolic tangent):

$$f(x) = \text{th}(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$$

- **Variants:** elu, softmax, selu, softplus, softsign, hard_sigmoid, exponential, LeakyReLU, PReLU, ELU, ThresholdedReLU, Softmax, ReLU

<https://keras.io/activations/>



Функция ошибки - loss

- mean absolute error (MAE):

$$\text{MAE} = \frac{\sum_{i=1}^n |y_i - x_i|}{n} = \frac{\sum_{i=1}^n |e_i|}{n}.$$

- mean squared error (MSE):

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2.$$

- binary cross entropy:

$$H_p(q) = -\frac{1}{N} \sum_{i=1}^N y_i \cdot \log(p(y_i)) + (1 - y_i) \cdot \log(1 - p(y_i))$$

- **Варианты:** mean_absolute_percentage_error, hinge, mean_squared_logarithmic_error, squared_hinge, kullback_leibler_divergence, poisson, categorical_hinge, huber_loss, logcosh, sparse_categorical_crossentropy, cosine_proximity, is_categorical_crossentropy

<https://keras.io/losses/>



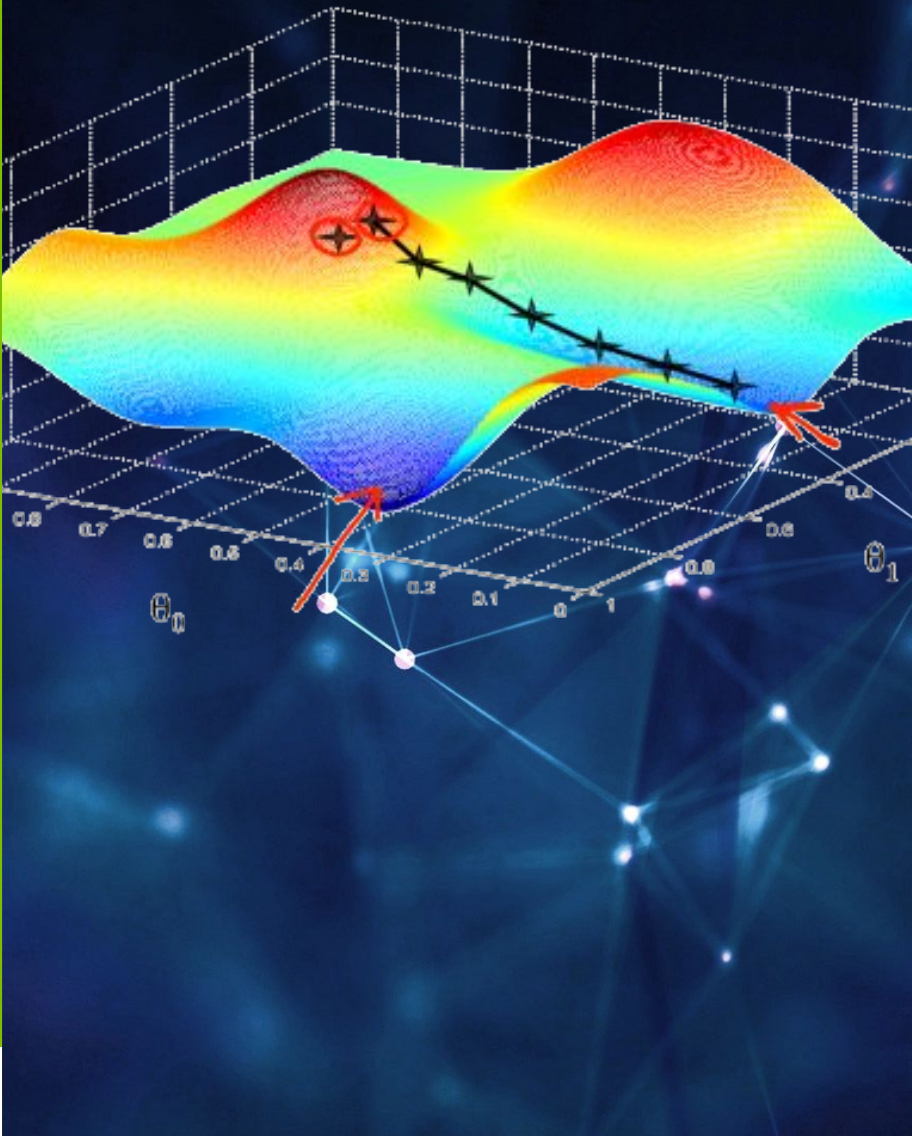
Оптимизаторы

- Adam
- RMSprop
- Adadelata

learning_rate - learning rate

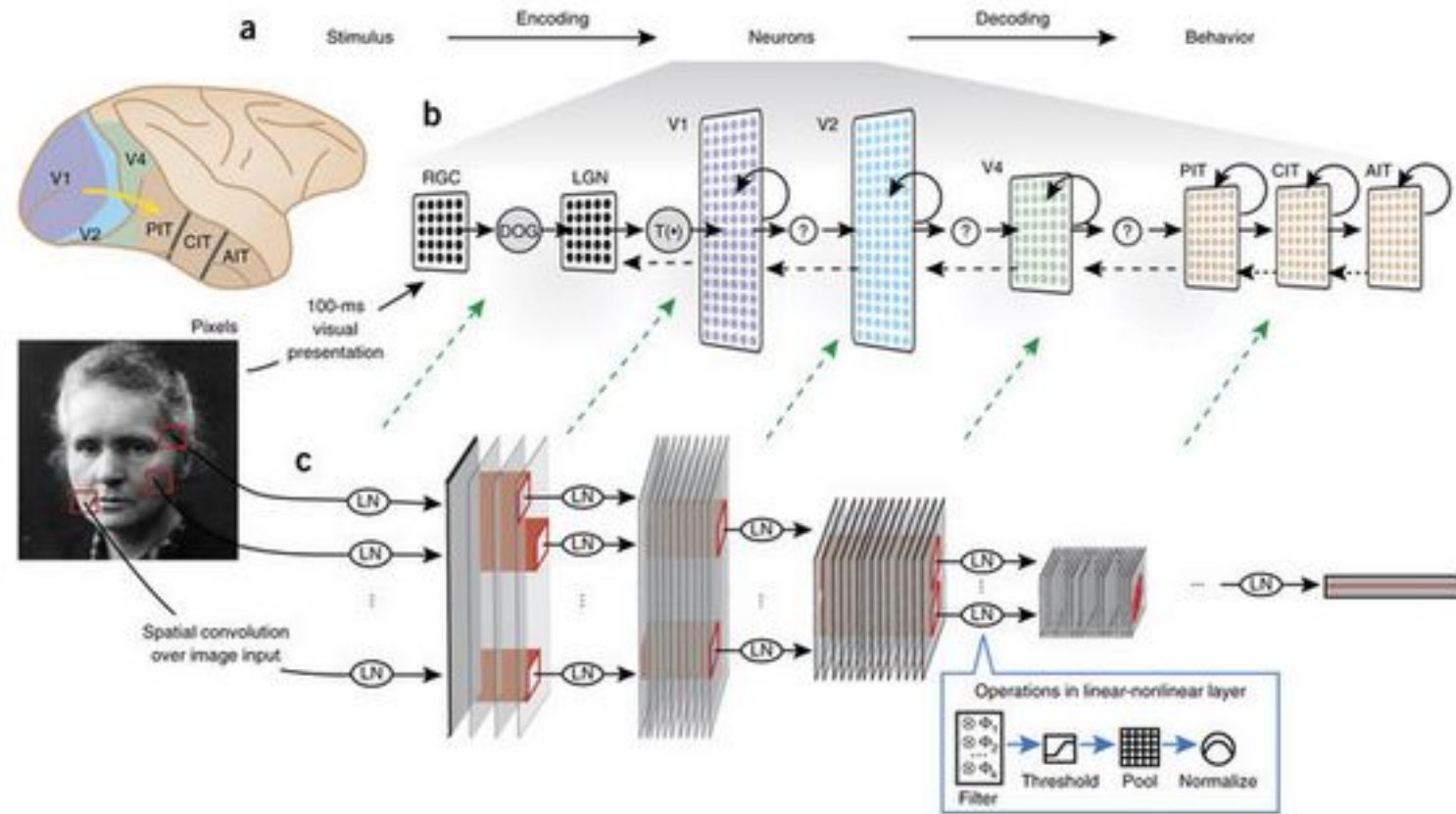
Варианты: SGD, Adagrad, Adamax, Nadam

<https://keras.io/optimizers/>



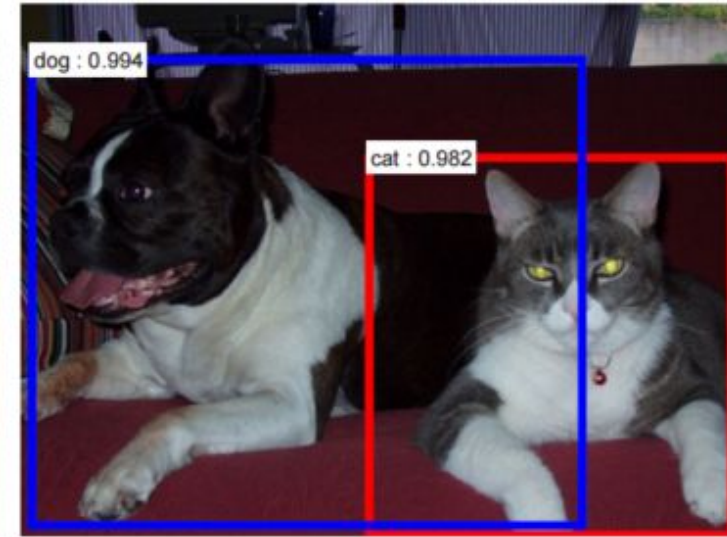
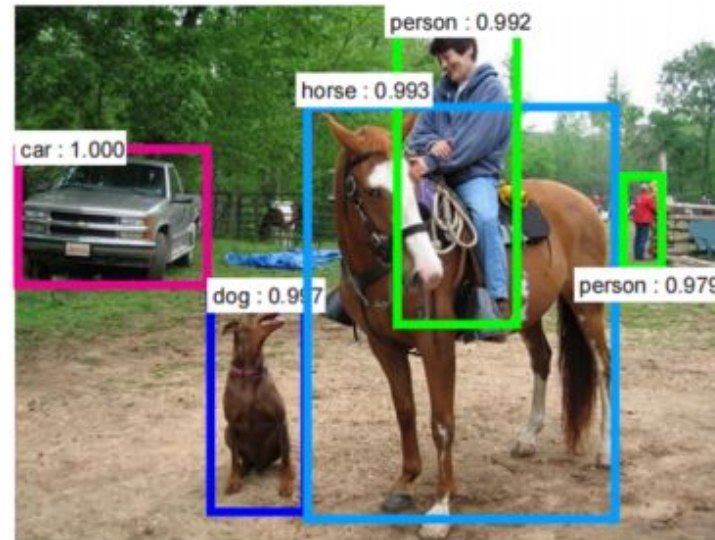
Сверточные нейронные сети

Вдохновлены тем, как мозг
обрабатывает изображения



Сверточные нейронные сети

Примеряются к
изображениям



a soccer player is kicking a soccer ball



a street sign on a pole in front of a building



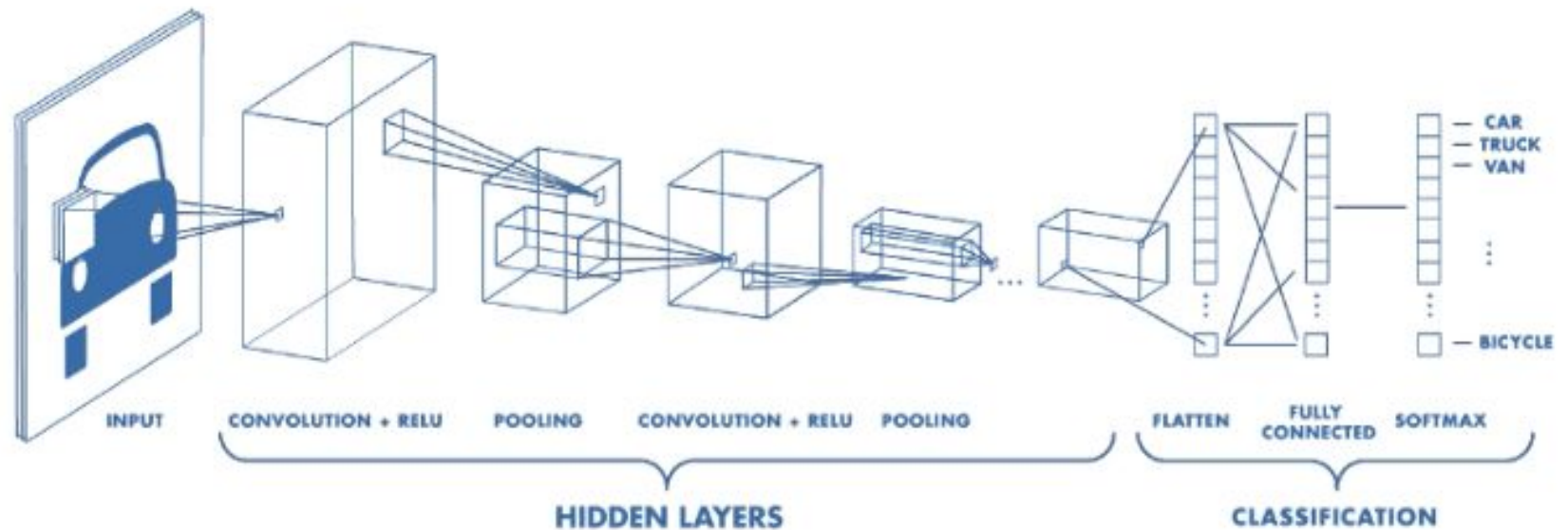
a couple of giraffe standing next to each other

Как нейронные сети видят изображение

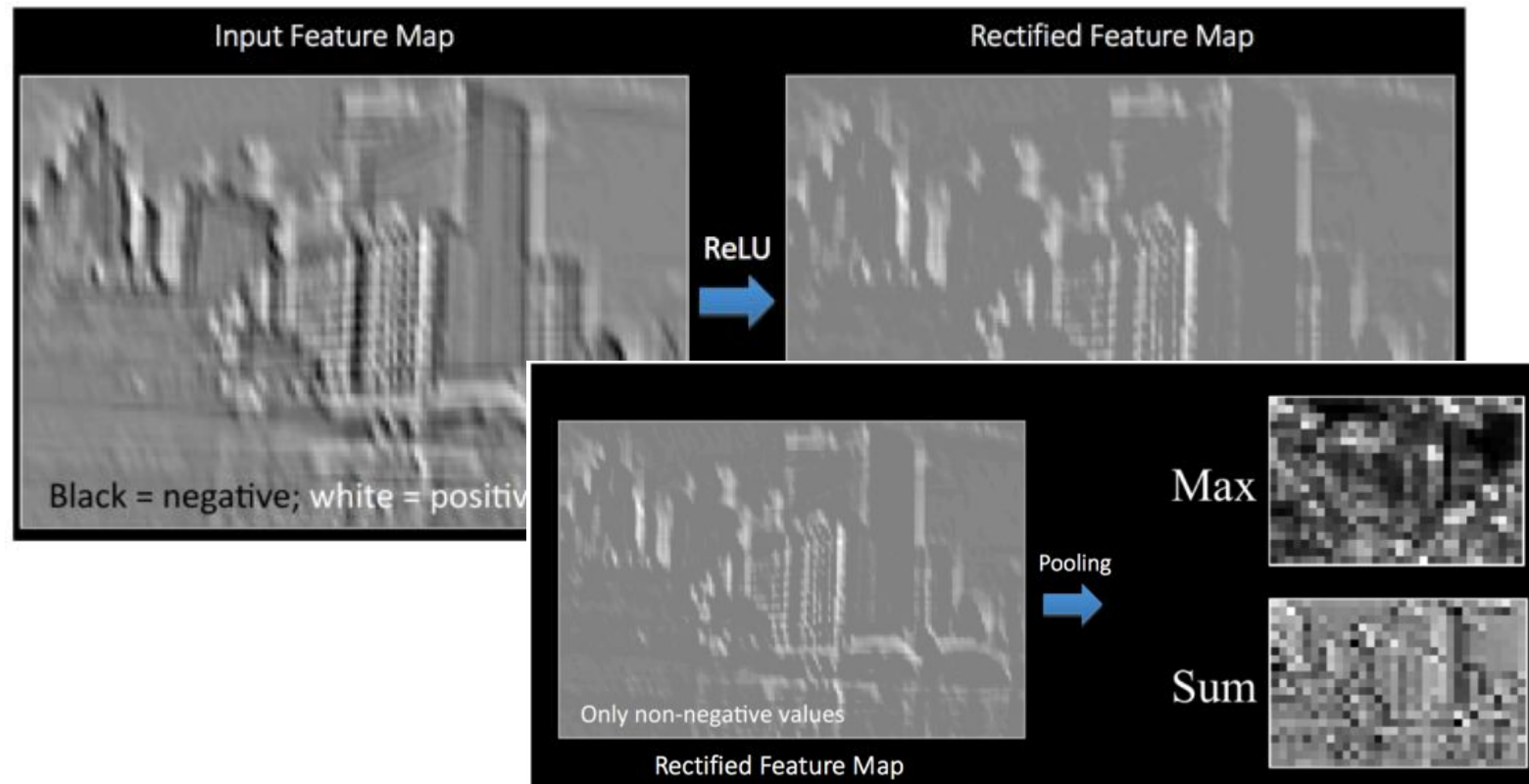
1.0	1.0	1.0	0.9	0.6	0.6	0.6	1.0	1.0	1.0	1.0	1.0
1.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.5	1.0	1.0	1.0
1.0	0.2	0.2	0.5	0.6	0.6	0.5	0.0	0.0	0.5	1.0	1.0
1.0	0.9	1.0	1.0	1.0	1.0	1.0	0.9	0.0	0.0	0.9	1.0
1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0
1.0	1.0	1.0	0.5	0.5	0.5	0.5	0.5	0.4	0.0	0.5	1.0
1.0	0.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	1.0
0.9	0.0	0.0	0.6	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0
0.5	0.0	0.6	1.0	1.0	1.0	1.0	1.0	0.5	0.0	0.5	1.0
0.5	0.0	0.7	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.5	1.0
0.6	0.0	0.6	1.0	1.0	1.0	1.0	0.5	0.0	0.0	0.5	1.0
0.9	0.1	0.0	0.6	0.7	0.7	0.5	0.0	0.5	0.0	0.5	1.0
1.0	0.7	0.1	0.0	0.0	0.0	0.1	0.9	0.8	0.0	0.5	1.0
1.0	1.0	1.0	0.8	0.8	0.9	1.0	1.0	1.0	1.0	1.0	1.0








Сверточные нейронные сети

Как нейронные сети
работают с
изображением

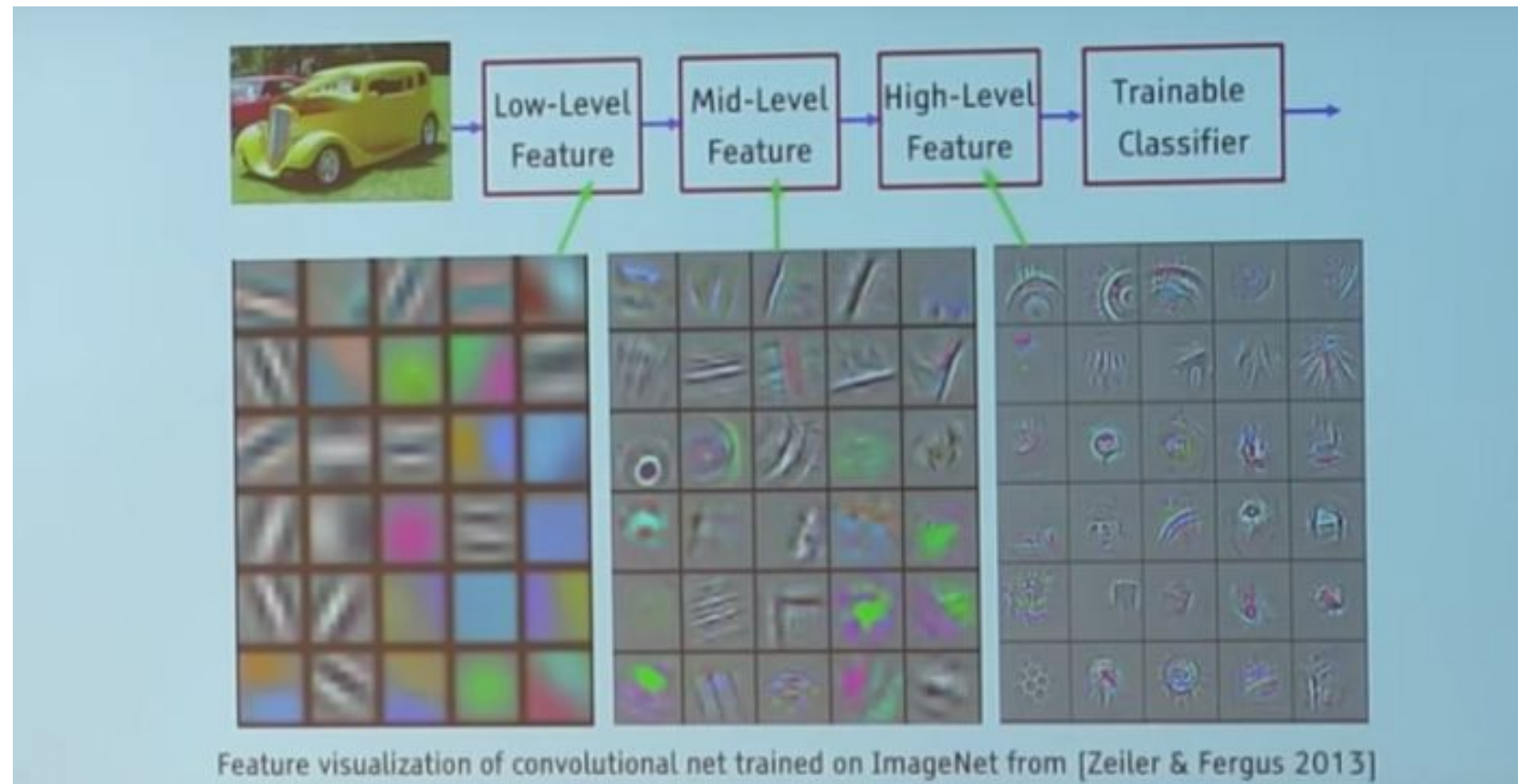


Сверточные нейронные сети



Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Сверточные нейронные сети



Популярные библиотеки

Top 8 Deep Learning Frameworks

