ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

Машинное зрение

Лекция 5. Сверточные сети

Учебные вопросы

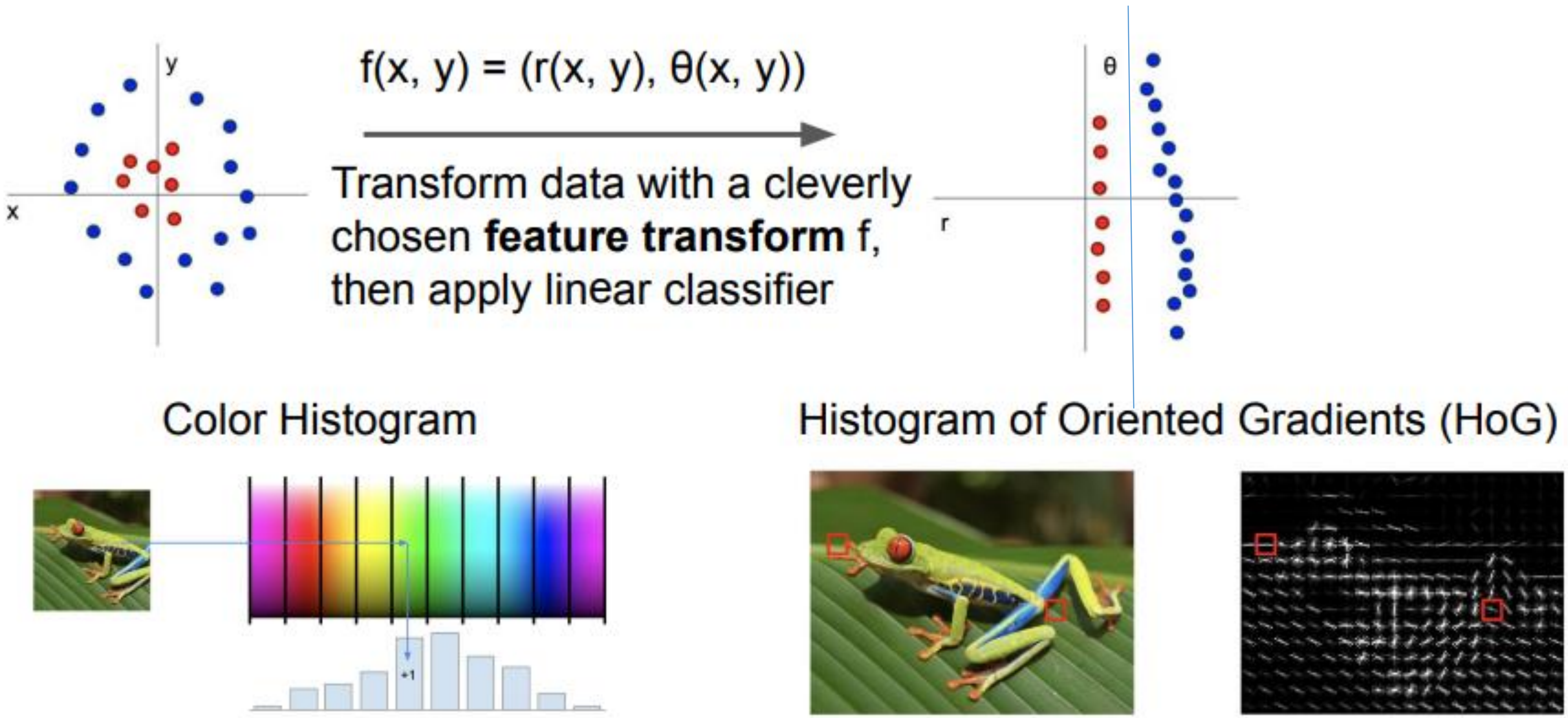
ФИНАНСОВЫЙ УНИВЕРСИТЕТ

1. Многослойные нелинейные классификаторы.
2. Механизм обратного распространения ошибки (Backpropagation).
3. Сверточные сети (Convolution nets).

Материалы для более глубокого изучения:
- **cs231n.stanford.edu (Convolution NN for Visual Recognition)**
- **dlcourse.ai (курс на русском)**

Сроки сдачи CodeLab_2 (всего будет 4, по 10 баллов за каждый):
- **3 декабря 2021 г.**

What`s wrong with linear classification?



Neural networks: also called fully connected network

(**Before**) Linear score function: $f = Wx$

(**Now**) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$

$$x \in \mathbb{R}^D, W_1 \in \mathbb{R}^{H \times D}, W_2 \in \mathbb{R}^{C \times H}$$

"Neural Network" is a very broad term; these are more accurately called "fully-connected networks" or sometimes "multi-layer perceptrons" (MLP)

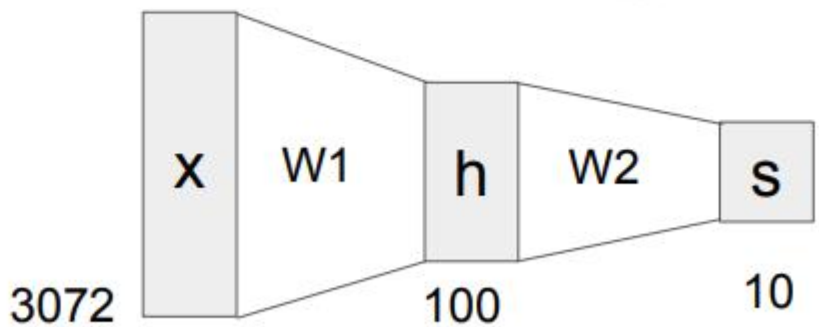
(In practice we will usually add a learnable bias at each layer as well)

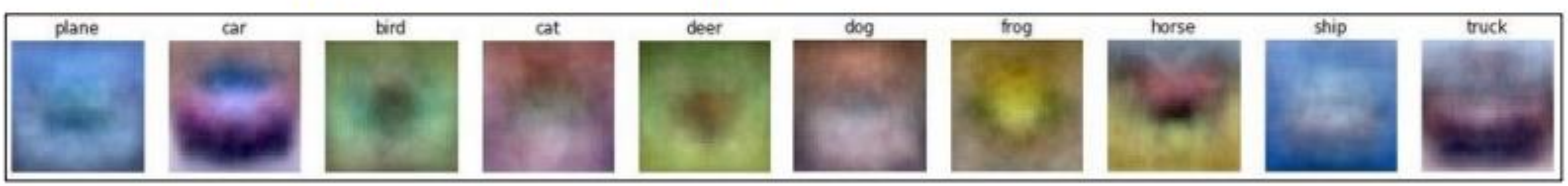
Многослойные нелинейные

Neural networks: learning 100s of templates

(**Before**) Linear score function: $f = Wx$

(**Now**) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$





Learn 100 templates instead of 10.

Share templates between classes

Neural networks: why is max operator important?

(**Before**) Linear score function: $f = Wx$

(**Now**) 2-layer Neural Network $f = W_2 \boxed{\max(0,} W_1 x)$

The function $\max(0, z)$ is called the **activation function.**
Q: What if we try to build a neural network without one?

$$f = W_2 W_1 x \qquad W_3 = W_2 W_1 \in \mathbb{R}^{C \times H}, f = W_3 x$$

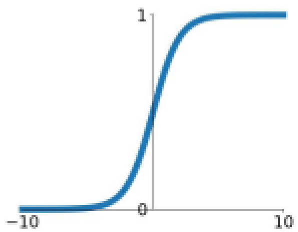
A: We end up with a linear classifier again!

Activation functions

ReLU is a good default choice for most problems

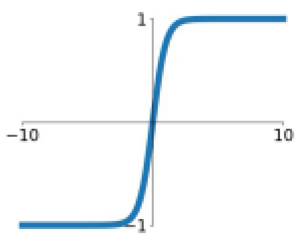
Sigmoid

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



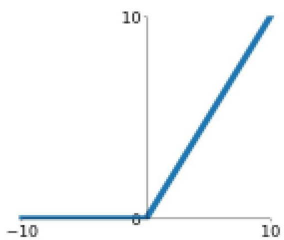
tanh

$$\tanh(x)$$



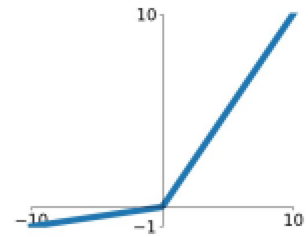
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

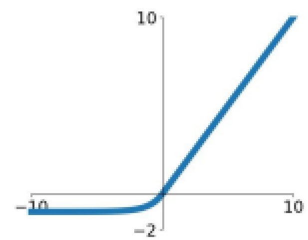


Maxout

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

ELU

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



Full implementation of training a 2-layer Neural Network needs ~20 lines:

```python
import numpy as np
from numpy.random import randn

N, D_in, H, D_out = 64, 1000, 100, 10
x, y = randn(N, D_in), randn(N, D_out)
w1, w2 = randn(D_in, H), randn(H, D_out)

for t in range(2000):
  h = 1 / (1 + np.exp(-x.dot(w1)))
  y_pred = h.dot(w2)
  loss = np.square(y_pred - y).sum()
  print(t, loss)

  grad_y_pred = 2.0 * (y_pred - y)
  grad_w2 = h.T.dot(grad_y_pred)
  grad_h = grad_y_pred.dot(w2.T)
  grad_w1 = x.T.dot(grad_h * h * (1 - h))

  w1 -= 1e-4 * grad_w1
  w2 -= 1e-4 * grad_w2
```

Define the network

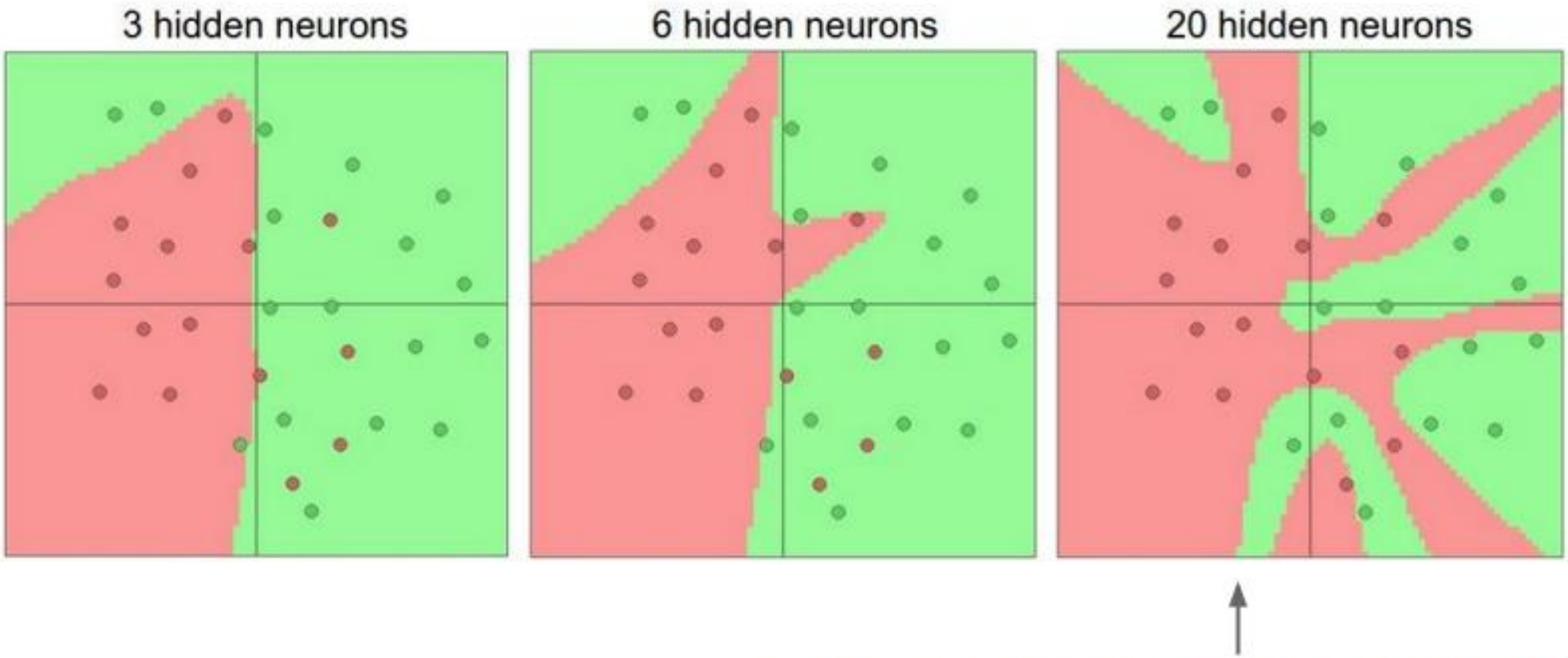
Forward pass

Calculate the analytical gradients

Gradient descent

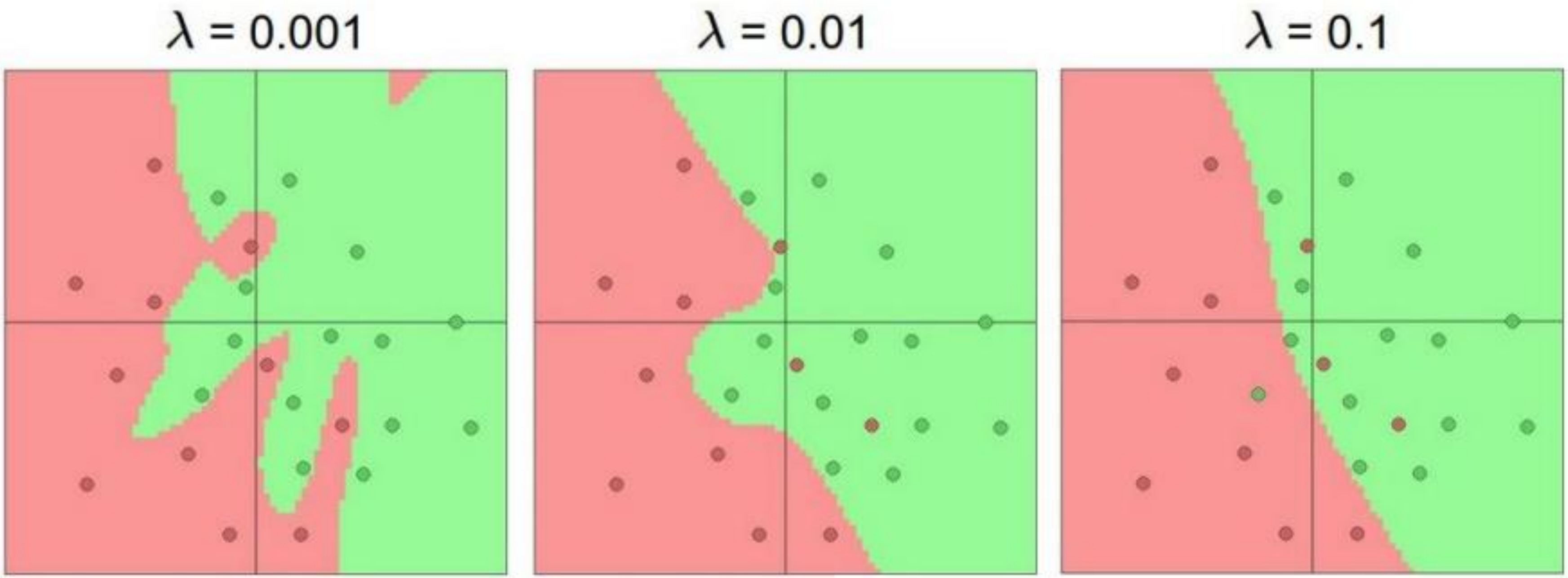
Многослойные нелинейные

Setting the number of layers and their sizes



more neurons = more capacity

Do not use size of neural network as a regularizer. Use stronger regularization instead:



(Web demo with ConvNetJS:

$$L(W) = \frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i) + \lambda R(W)$$

http://cs.stanford.edu/people/karpathy/convnetjs/demo /classify2d.html

Problem: How to compute gradients?

$$s = f(x; W_1, W_2) = W_2 \max(0, W_1 x)$$ Nonlinear score function

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$ SVM Loss on predictions

$$R(W) = \sum_k W_k^2$$ Regularization

$$L = \frac{1}{N} \sum_{i=1}^{N} L_i + \lambda R(W_1) + \lambda R(W_2)$$ Total loss: data loss + regularization

If we can compute $\frac{\partial L}{\partial W_1}, \frac{\partial L}{\partial W_2}$ then we can learn W_1 and W_2

(Bad) Idea: Derive $\nabla_W L$ on paper

$$s = f(x; W) = Wx$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

$$= \sum_{j \neq y_i} \max(0, W_{j,:} \cdot x + W_{y_i,:} \cdot x + 1)$$

$$L = \frac{1}{N} \sum_{i=1}^{N} L_i + \lambda \sum_k W_k^2$$

$$= \frac{1}{N} \sum_{i=1}^{N} \sum_{j \neq y_i} \max(0, W_{j,:} \cdot x + W_{y_i,:} \cdot x + 1) + \lambda \sum_k W_k^2$$

$$\nabla_W L = \nabla_W \left(\frac{1}{N} \sum_{i=1}^{N} \sum_{j \neq y_i} \max(0, W_{j,:} \cdot x + W_{y_i,:} \cdot x + 1) + \lambda \sum_k W_k^2 \right)$$

Problem: Very tedious: Lots of matrix calculus, need lots of paper

Problem: What if we want to change loss? E.g. use softmax instead of SVM? Need to re-derive from scratch =(

Problem: Not feasible for very complex models!

Backpropagation

Really complex neural networks!!

input image

loss

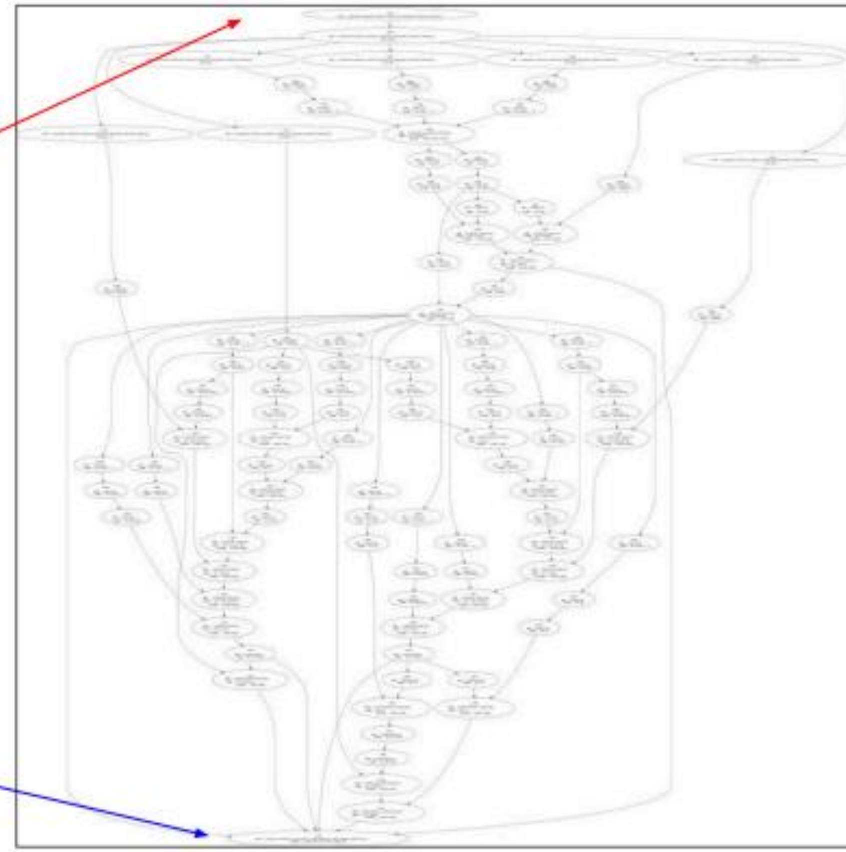
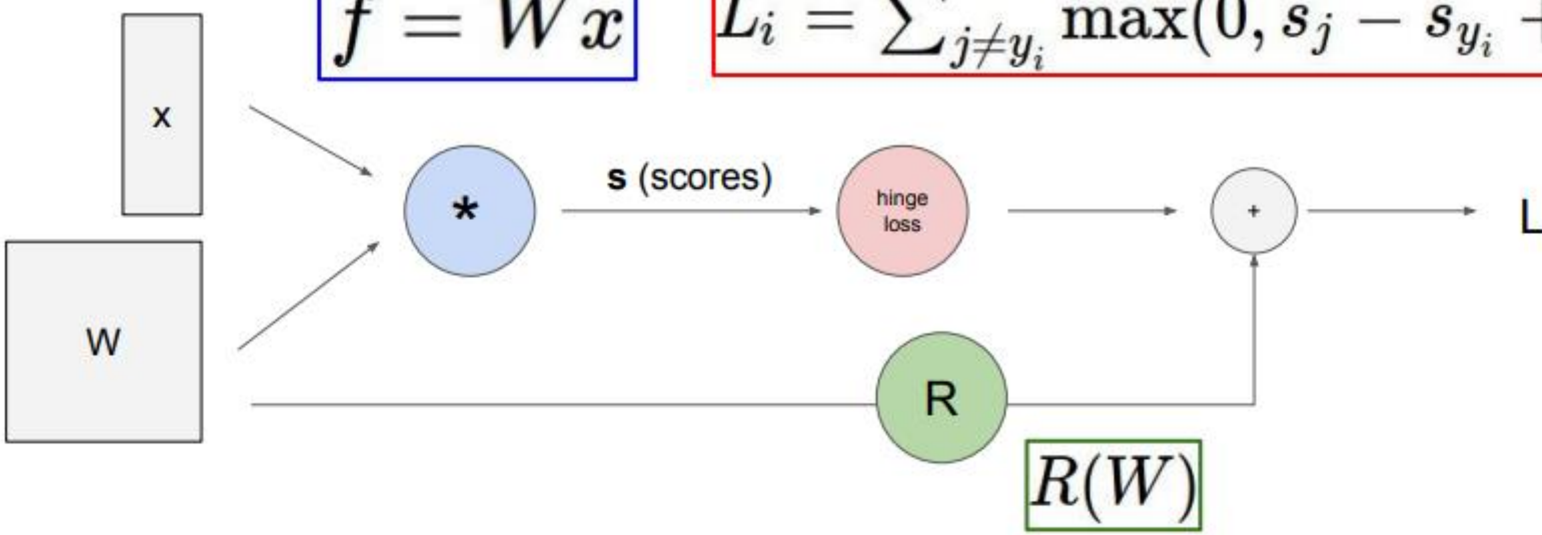


Figure reproduced with permission from a Twitter post by Andrej Karpathy.

Better Idea: Computational graphs + Backpropagation

$$\boxed{f = Wx} \qquad \boxed{L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)}$$

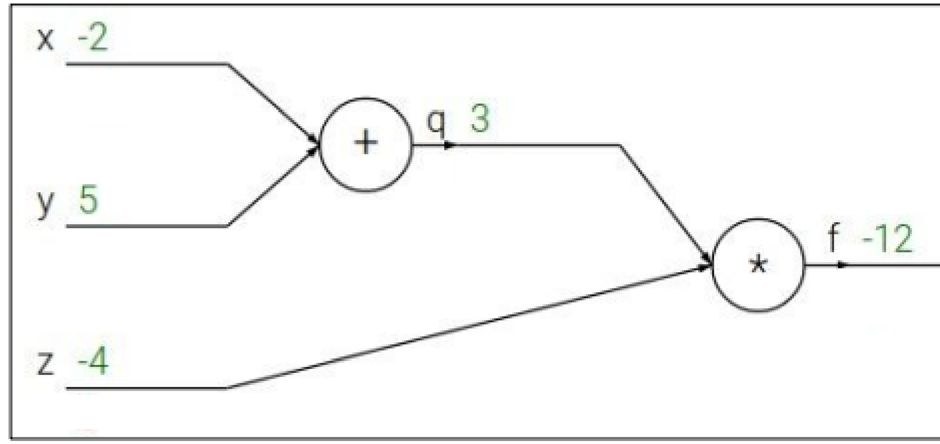


ФИНАНСОВЫЙ УНИВЕРСИТЕТ

Backpropagation: a simple example

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4



ФИНАНСОВЫЙ УНИВЕРСИТЕТ

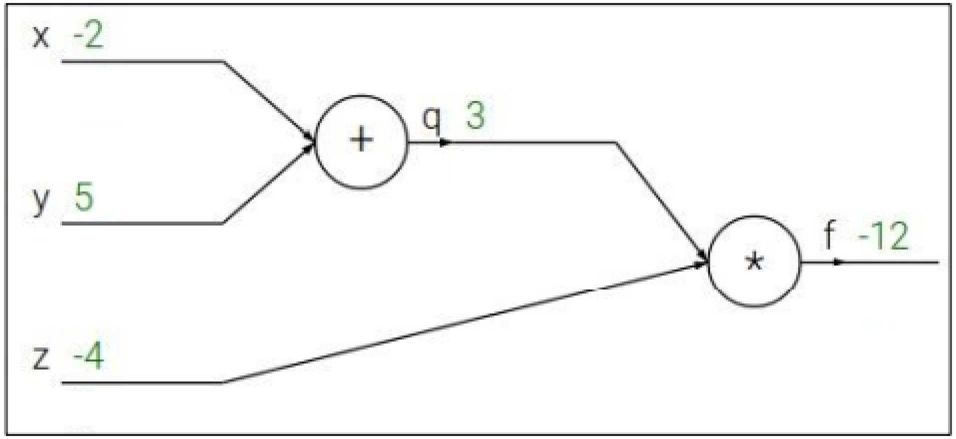
Backpropagation: a simple example

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$



Backpropagation: a simple example

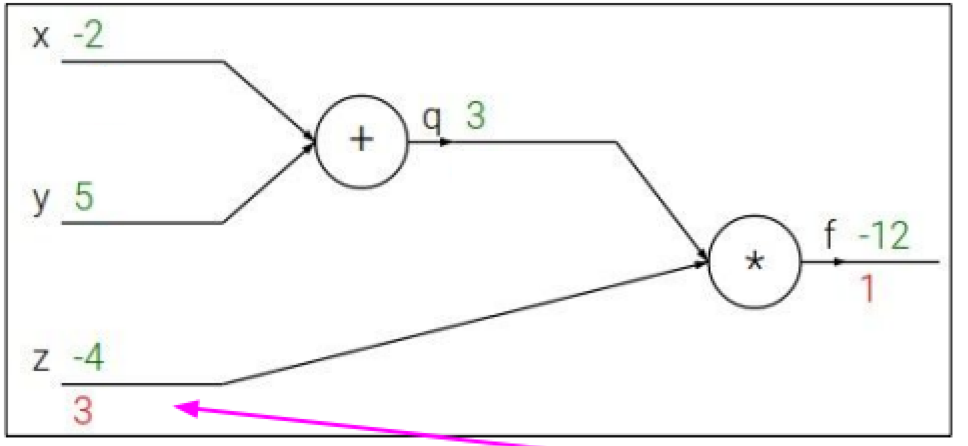
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\frac{\partial f}{\partial z}$$

Backpropagation: a simple example

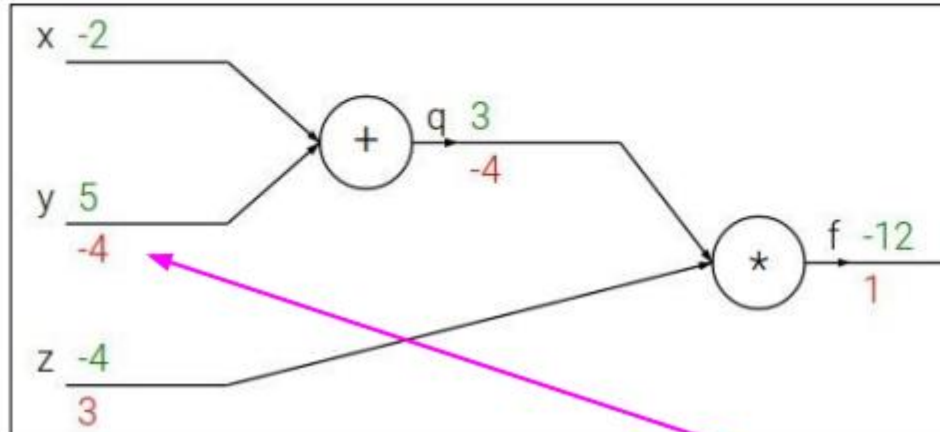
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\frac{\partial f}{\partial y}$$

Chain rule:

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial y}$$

Upstream gradient

Local gradient

Backpropagation: a simple example

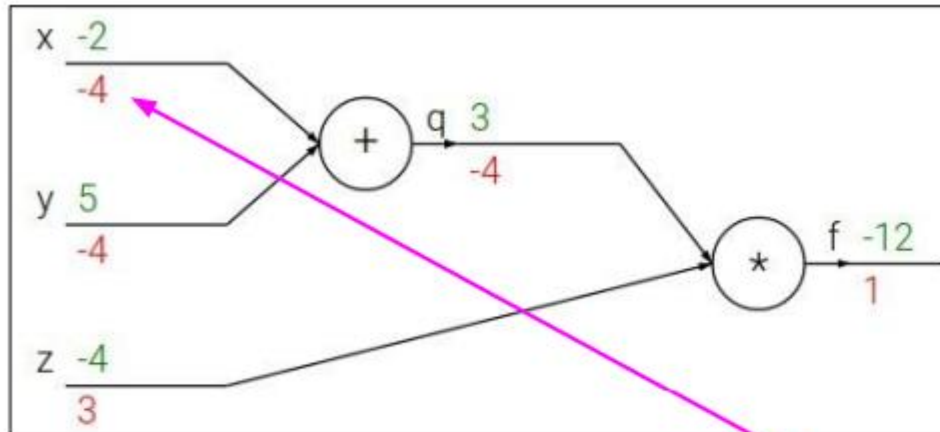
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$\frac{\partial f}{\partial x}$

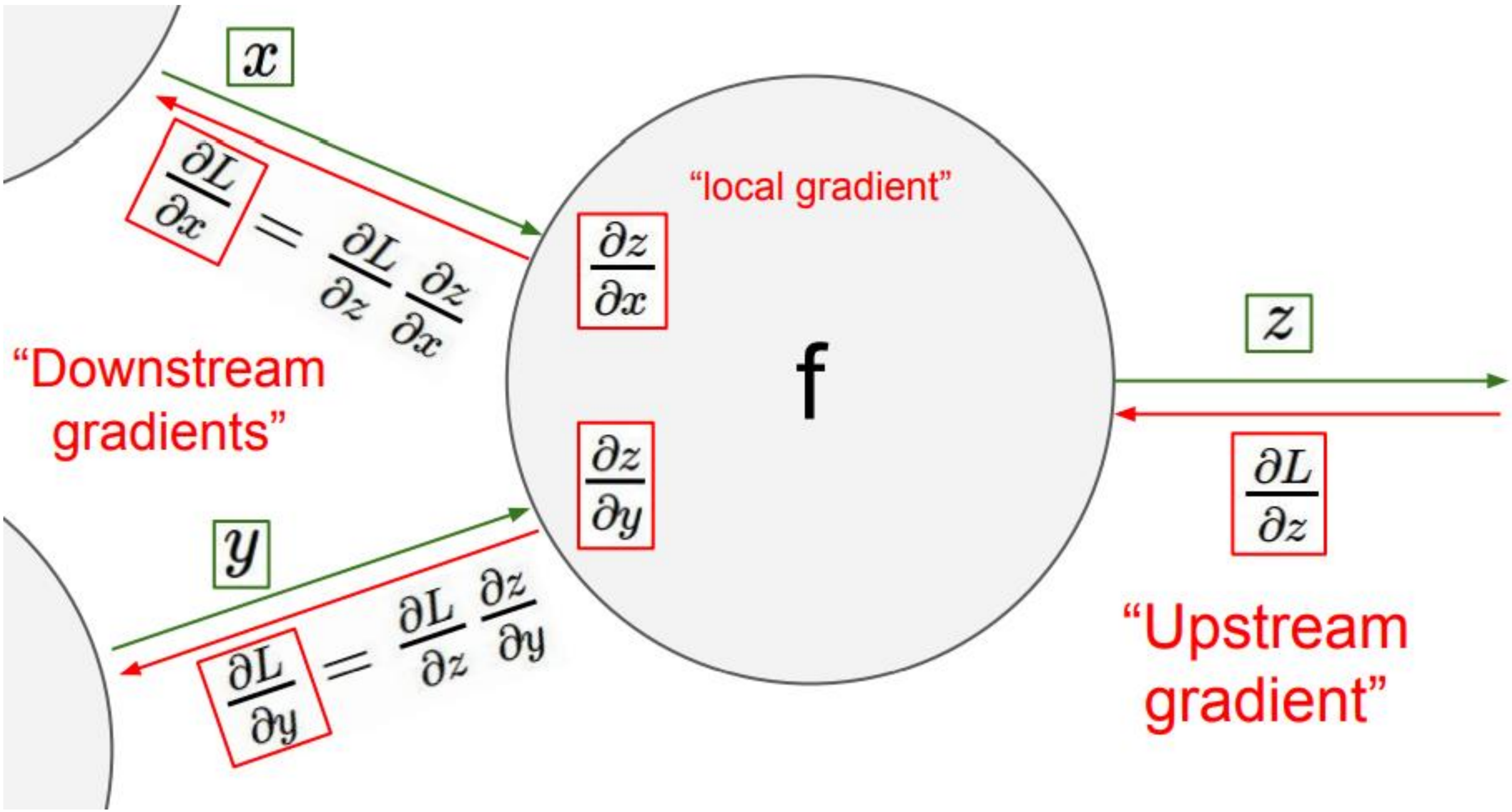
Chain rule:

$$\boxed{\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial x}}$$

Upstream gradient

Local gradient

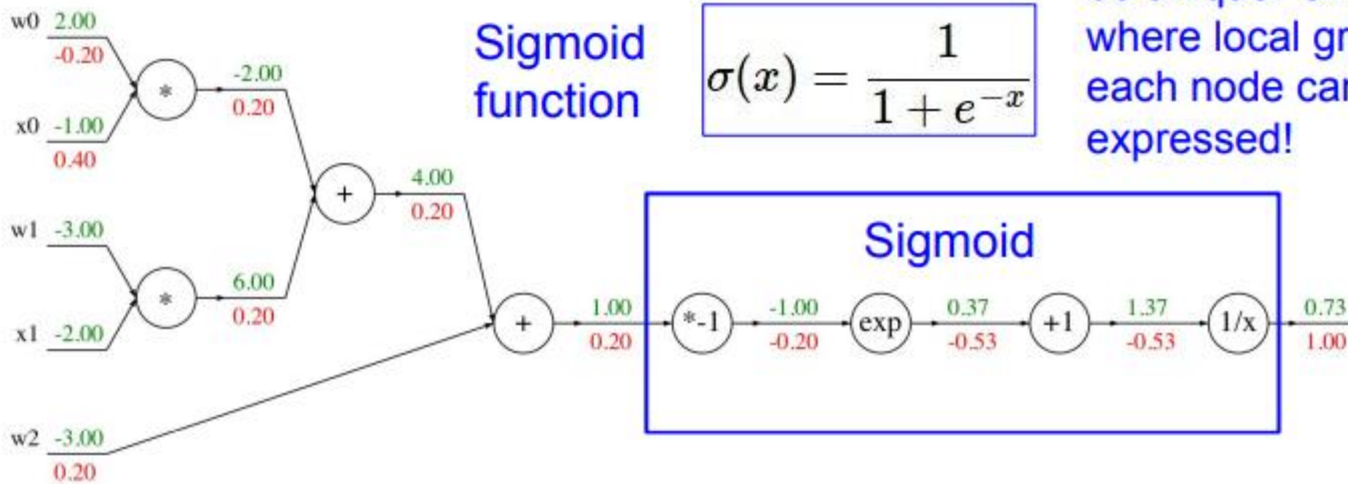
Backpropagation



Another example: $f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$

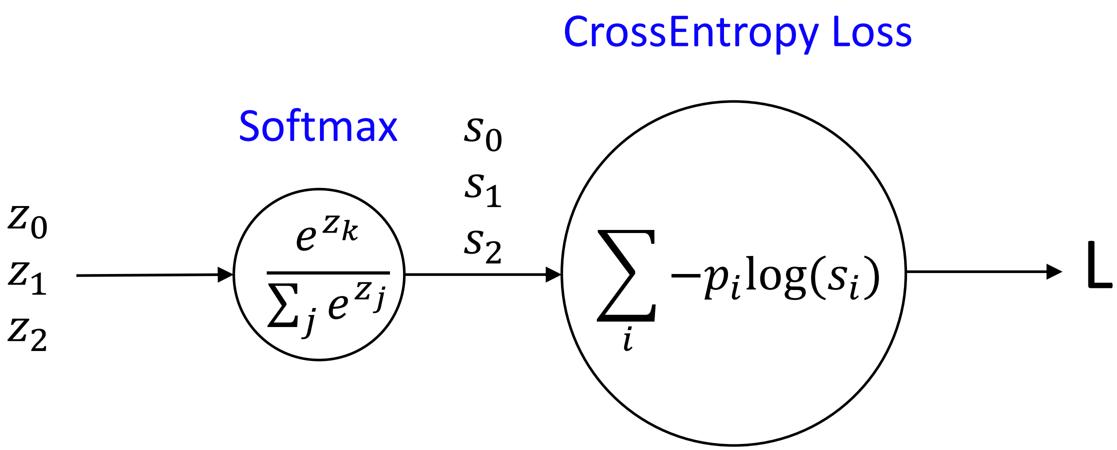
Computational graph representation may not be unique. Choose one where local gradients at each node can be easily expressed!

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



Sigmoid local gradient: $$\frac{d\sigma(x)}{dx} = \frac{e^{-x}}{(1+e^{-x})^2} = \left(\frac{1+e^{-x}-1}{1+e^{-x}}\right)\left(\frac{1}{1+e^{-x}}\right) = (1-\sigma(x))\sigma(x)$$

Backpropagation



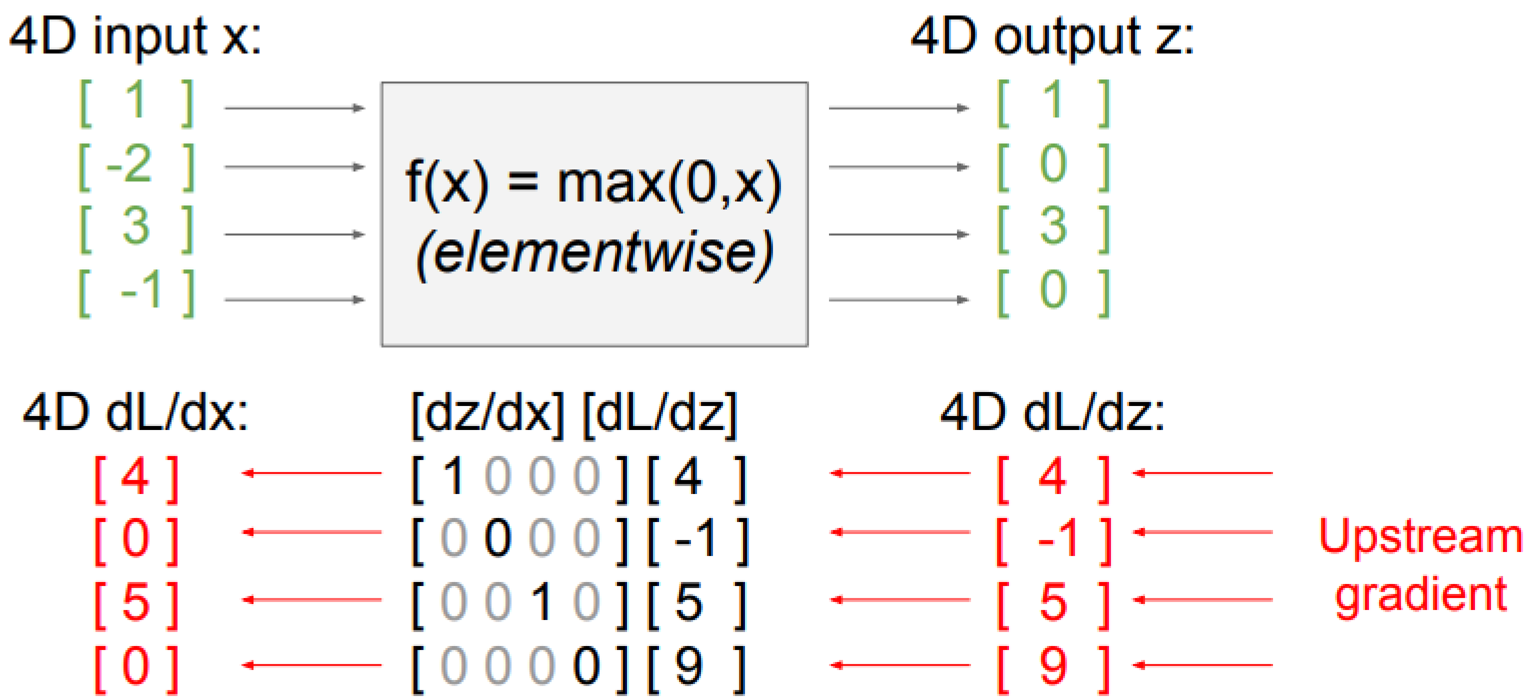
$$\frac{dL}{ds_i} = \begin{bmatrix} -\frac{p_0}{s_0} \\ 0 \\ 0 \end{bmatrix}$$

$$\frac{ds_0}{dz_0} = d\left(\frac{e^{z_0}}{e^{z_0}+e^{z_1}+e^{z_2}}\right)_{z_0} = \frac{e^{z_0}\cdot(e^{z_0}+e^{z_1}+e^{z_2}) - e^{z_0}\cdot e^{z_0}\cdot(e^{z_0}+e^{z_1}+e^{z_2})^2}{(e^{z_0}+e^{z_1}+e^{z_2})^2} = s_0 - s_0^2 = s_0\,(1-s_0)$$

$$\frac{ds_0}{dz_1} = d\left(\frac{e^{z_0}}{e^{z_0}+e^{z_1}+e^{z_2}}\right)_{z_1} = \frac{-e^{z_1}e^{z_0}}{(e^{z_0}+e^{z_1}+e^{z_2})^2} = -s_0 s_1 \qquad \frac{ds_0}{dz_2} = -s_0 s_2$$

$$\frac{dL}{dz} = \frac{dL}{ds} * \frac{ds}{dz} = \begin{matrix} -\frac{1}{s_0} \\ 0 \\ 0 \end{matrix} * \begin{matrix} s_0\,(1-s_0) & -s_0 s_1 & -s_0 s_2 \\ -s_1 s_0 & s_1\,(1-s_1) & -s_1 s_2 \\ -s_2 s_0 & -s_2 s_1 & s_2\,(1-s_2) \end{matrix} = [s_0 - 1\,,\ s_1\,, s_2\,]$$

ReLU gradient flow



A bit of history:

Gradient-based learning applied to document recognition

[LeCun, Bottou, Bengio, Haffner 1998]

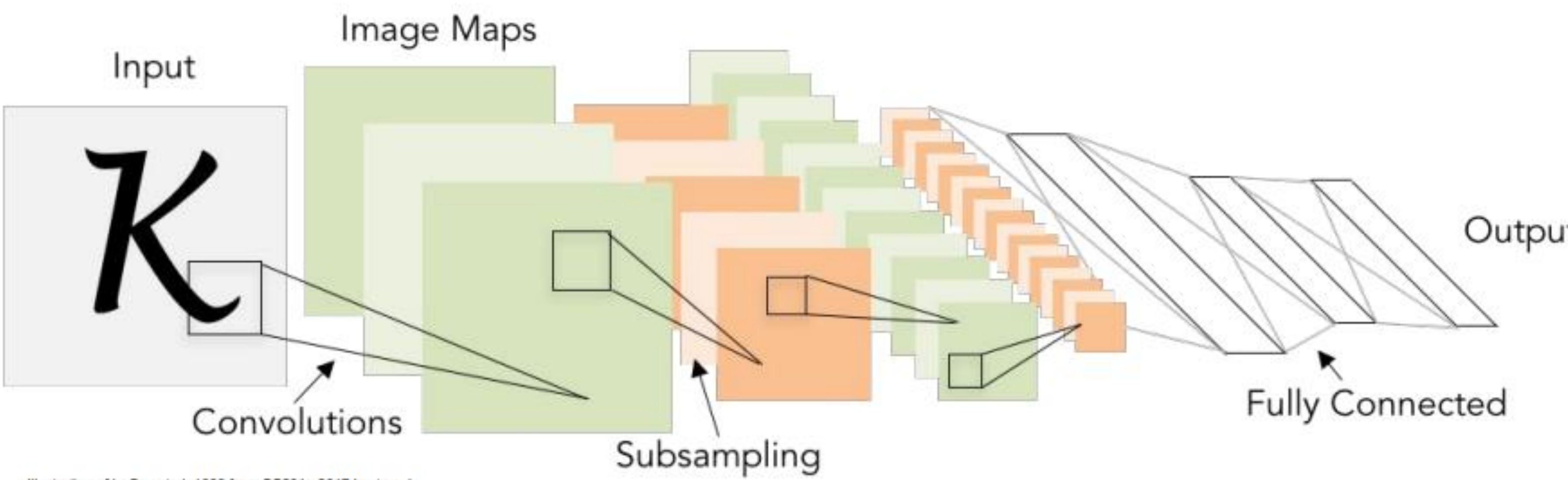
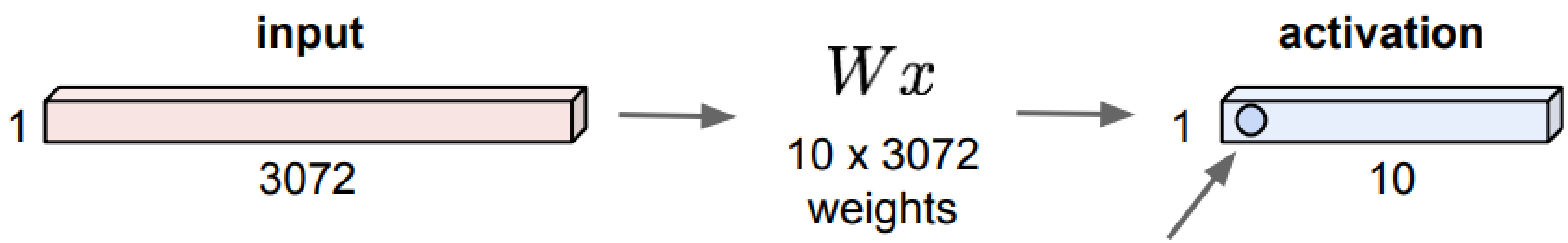


Illustration of LeCun et al. 1998 from CS231n 2017 Lecture 1

Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1

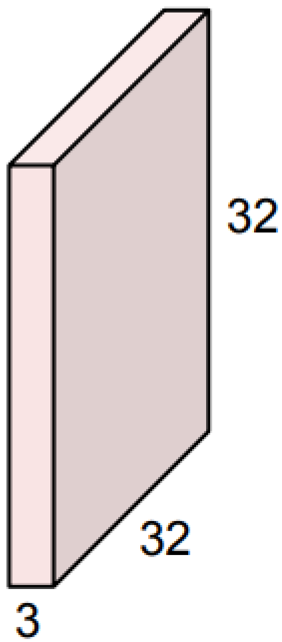


1 number:
the result of taking a dot product between a row of W and the input (a 3072-dimensional dot product)

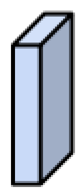
Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image



5x5x3 filter

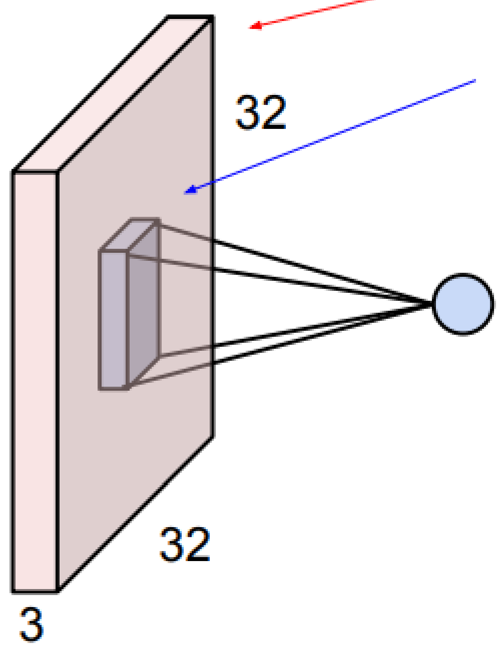


Convolve the filter with the image i.e. "slide over the image spatially, computing dot products"

Convolution Layer

32x32x3 image

5x5x3 filter w

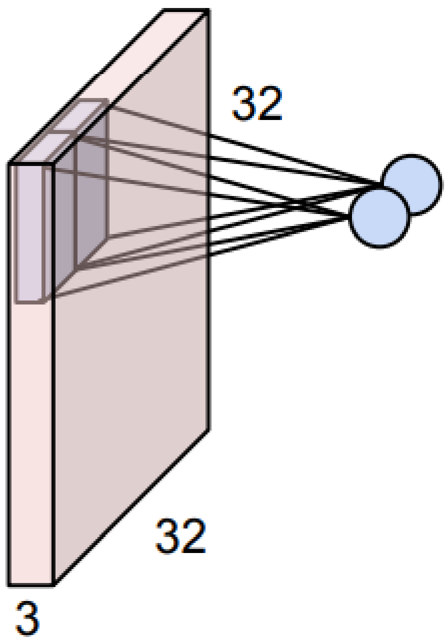


1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image (i.e. 5*5*3 = 75-dimensional dot product + bias)

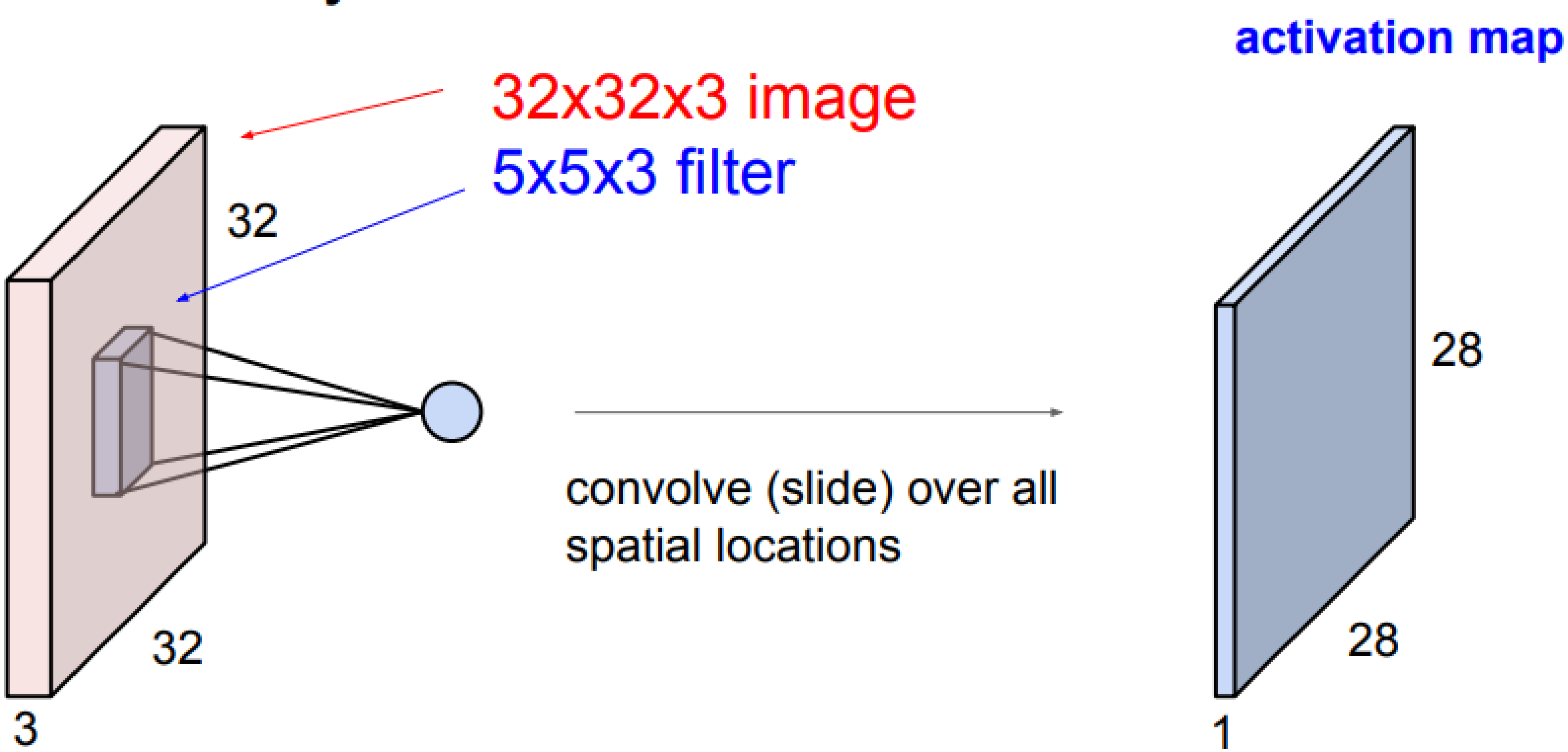
$$w^T x + b$$

ФИНАНСОВЫЙ УНИВЕРСИТЕТ

Convolution Layer

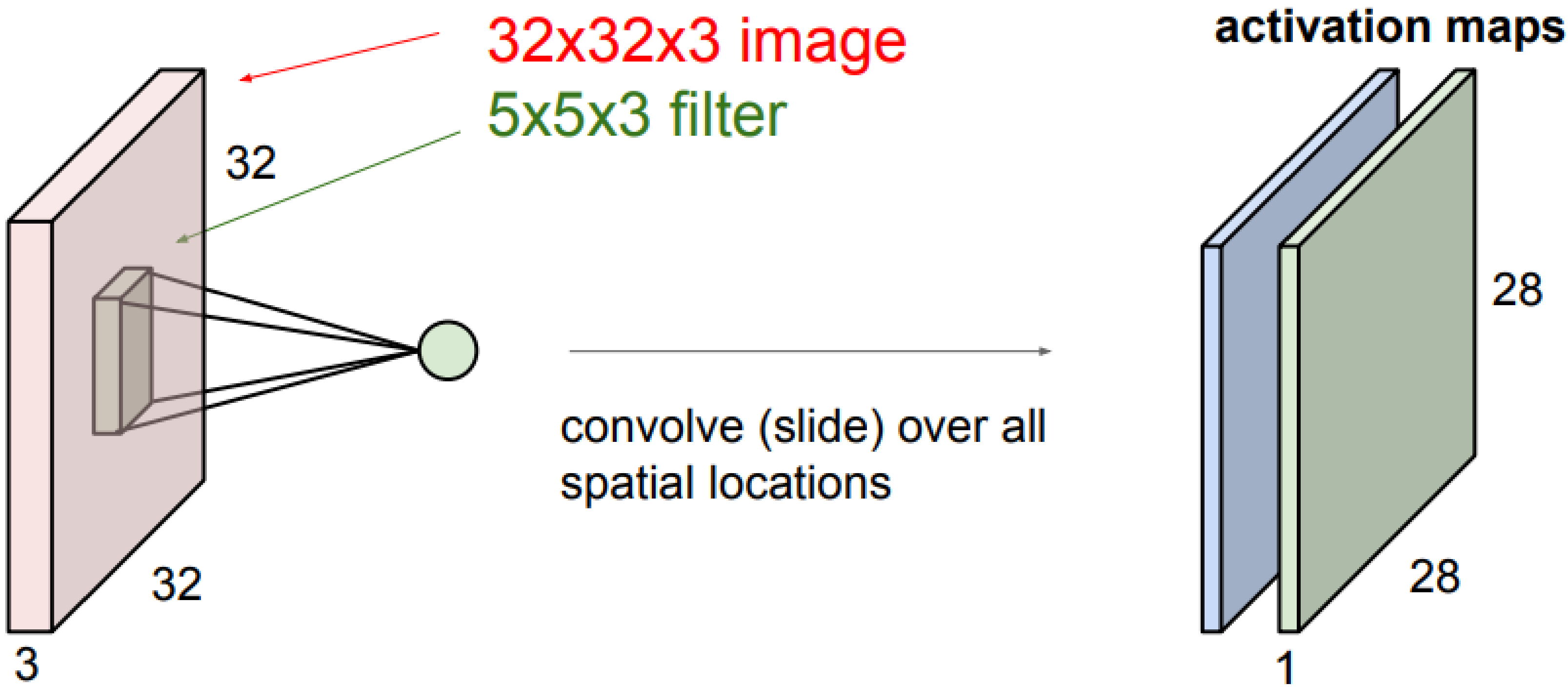


Convolution Layer

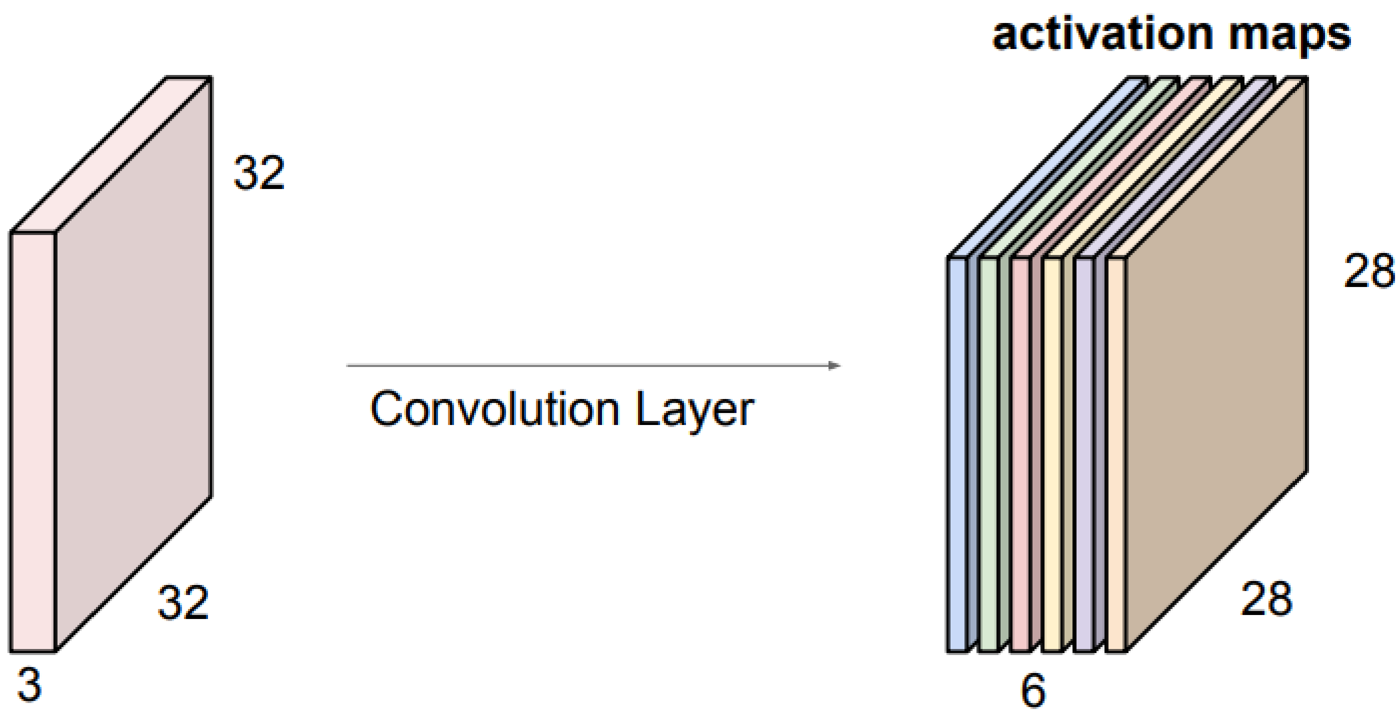


Convolution Layer

consider a second, green filter



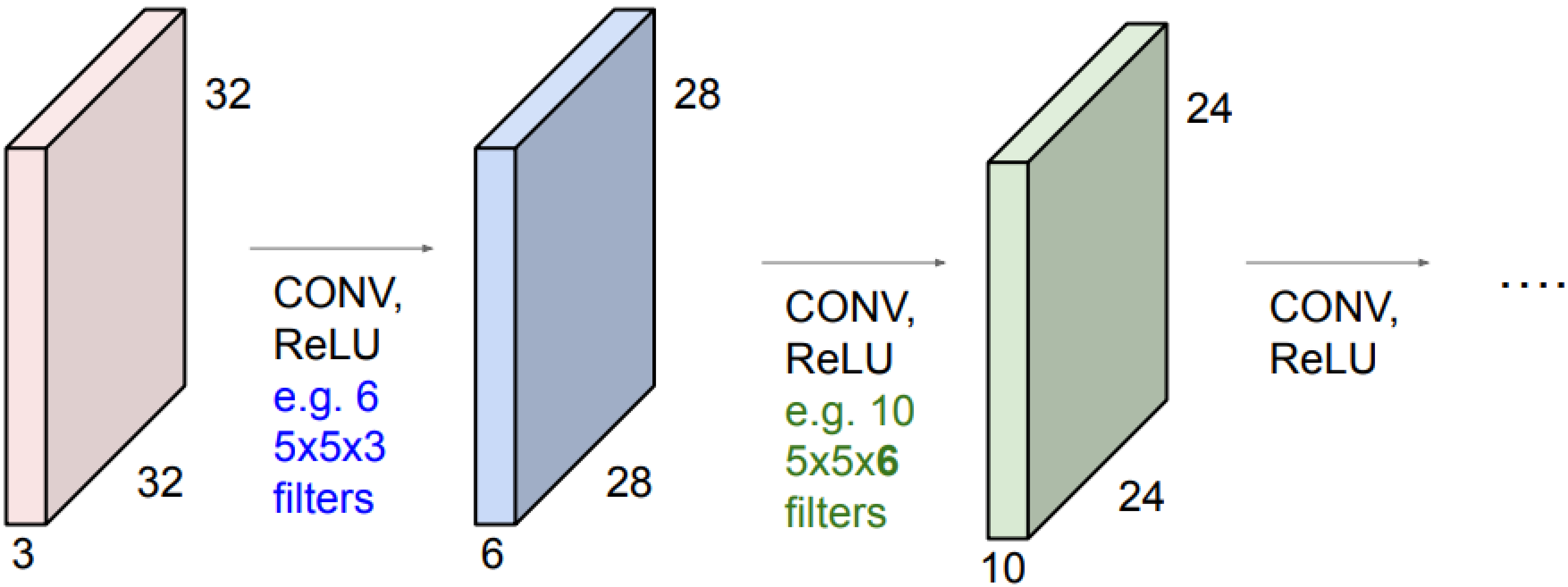
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



We stack these up to get a "new image" of size 28x28x6!

Convolution Nets

Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions

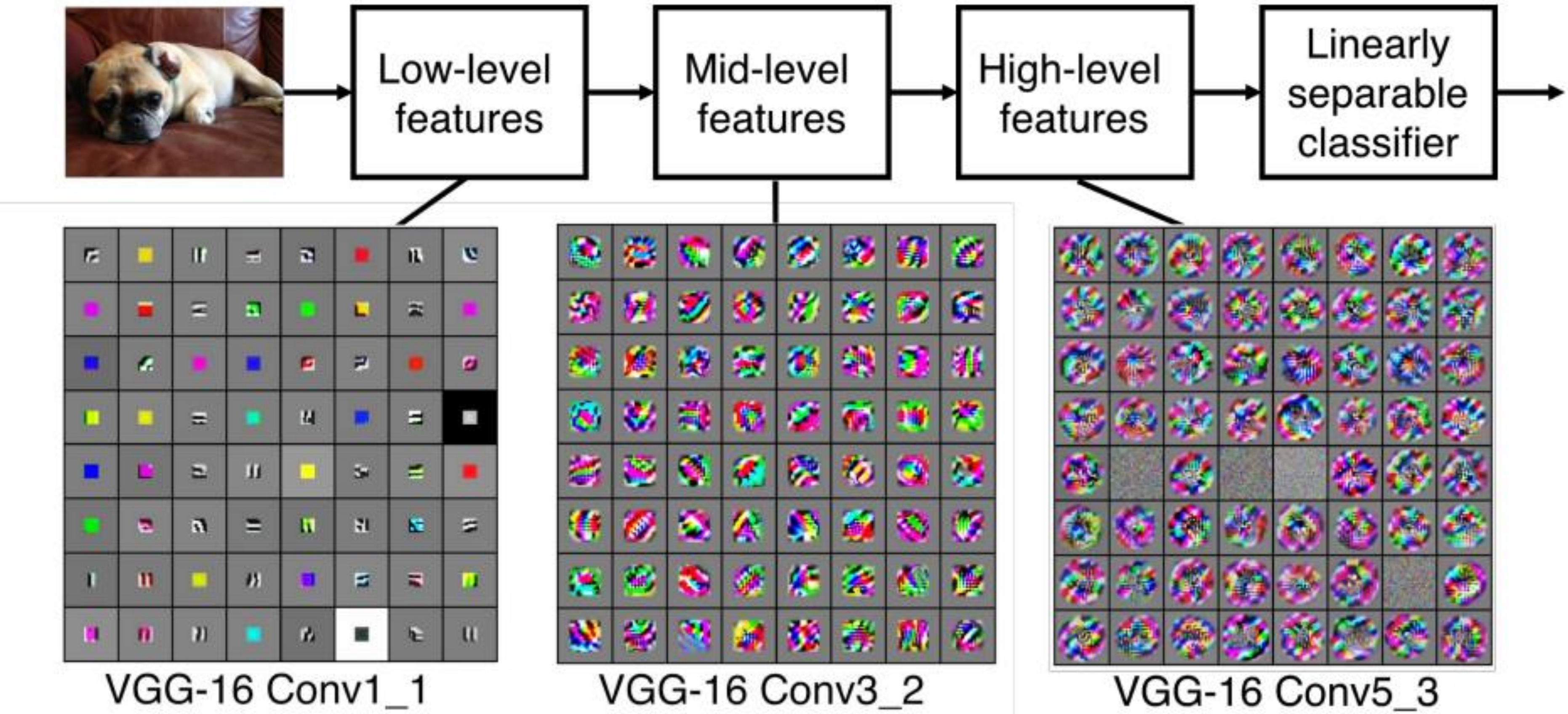


Convolution Nets

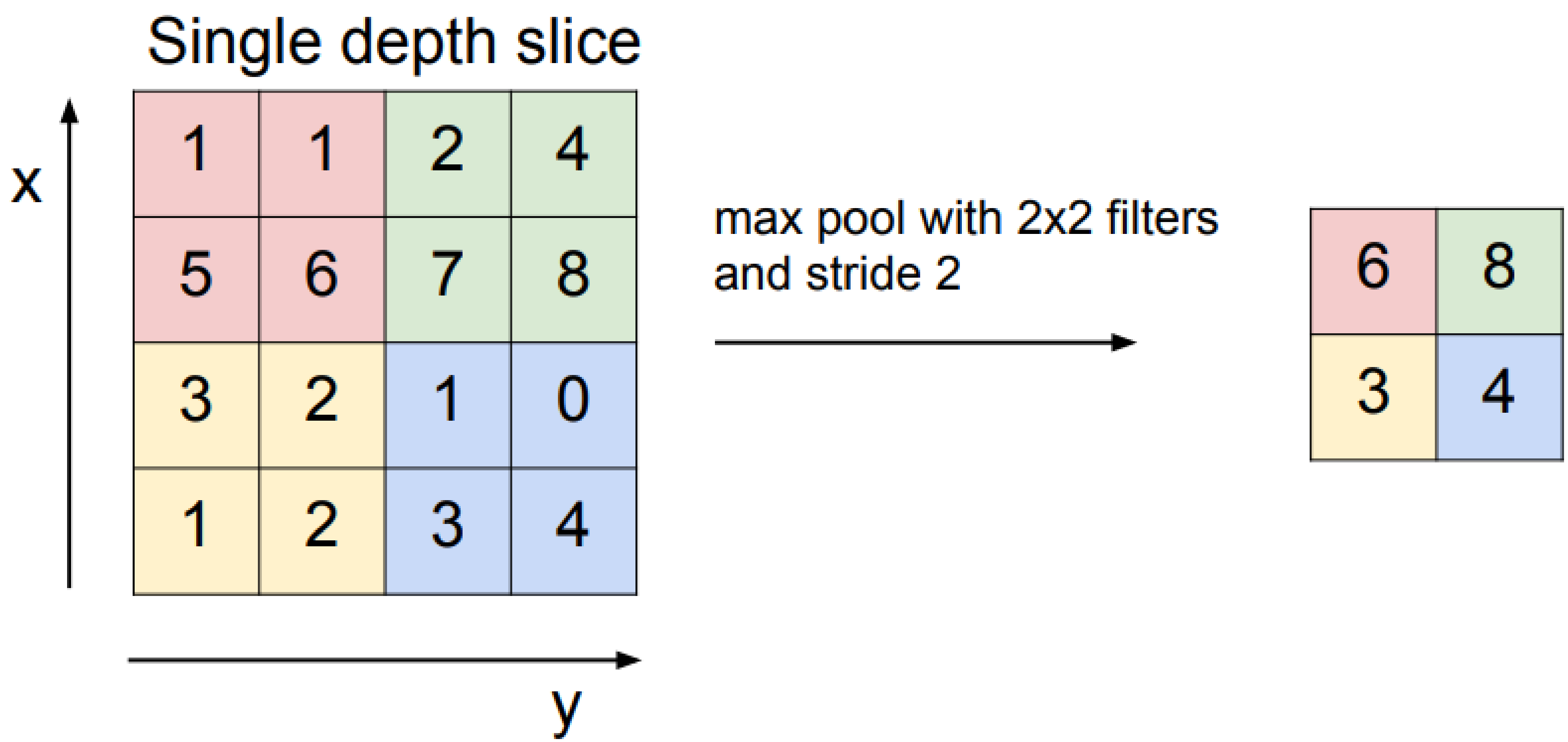
Preview

[Zeiler and Fergus 2013]

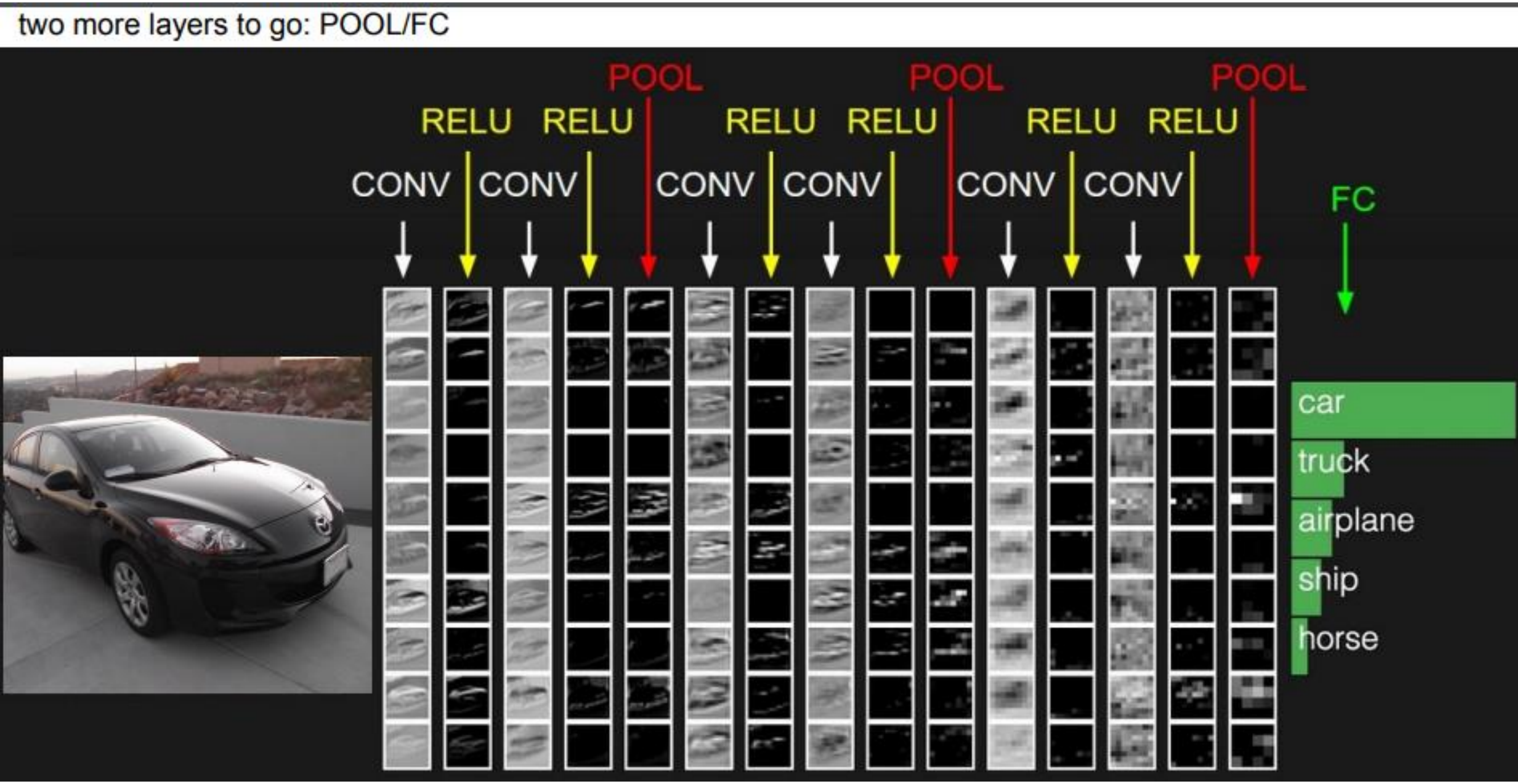
Visualization of VGG-16 by Lane McIntosh. VGG-16 architecture from [Simonyan and Zisserman 2014].



MAX POOLING


Single depth slice
x
y
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
max pool with 2x2 filters
and stride 2
6 8
3 4

Convolution Nets



Выводы

1.Многослойные нелинейные классификаторы позволяют строить сколь угодно сложные разделяющие гиперповерхности.

2. Любую сложную функцию можно представить как суперпозицию более простых. Механизм обратного распространения ошибки позволяет вычислить градиенты для сколь угодно сложных вычислительных графов.

3. С помощью сверточных фильтров можно находить на изображении локальные шаблоны (признаки). По их комбинации и взаимному расположению можно многое сказать об изображении.