High Level Computer Vision

Backpropagation & Convolutional Neural Networks @ April 24, 2019

Bernt Schiele & Mario Fritz

www.mpi-inf.mpg.de/hlcv/

Max Planck Institute for Informatics & Saarland University,
Saarland Informatics Campus Saarbrücken

Overview Today's Lecture

- Backpropagation - Gradient Descent
  - illustrated using computational graphs
  - chain rule - upstream and local gradients
  - modularization simple
- What is Deep Learning
  - intuition why deep learning can help
  - integrated learning of features and classifier
- Convolutional Neural Networks (CNNs)
  - one of the (few) highly successful NNs

Where we are

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

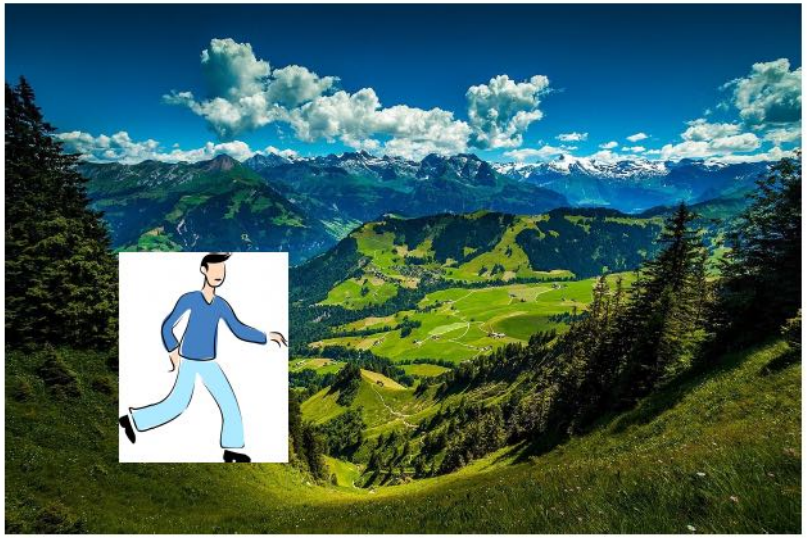
$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$ SVM loss

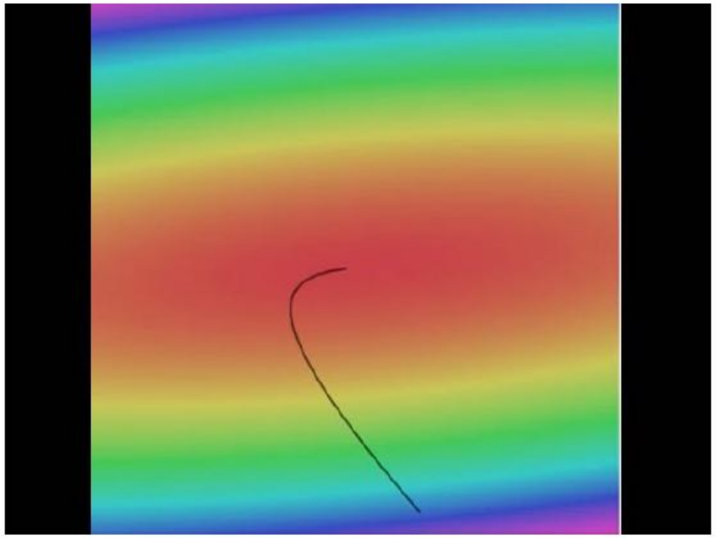
$$L = \frac{1}{N} \sum_{i=1}^{N} L_i + \sum_k W_k^2$$ data loss + regularization

want $\nabla_W L$

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Optimization





```
# Vanilla Gradient Descent

while True:
  weights_grad = evaluate_gradient(loss_fun, data, weights)
  weights += - step_size * weights_grad # perform parameter update
```

Landscape image is CC0 1.0 public domain
Walking man image is CC0 1.0 public domain

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

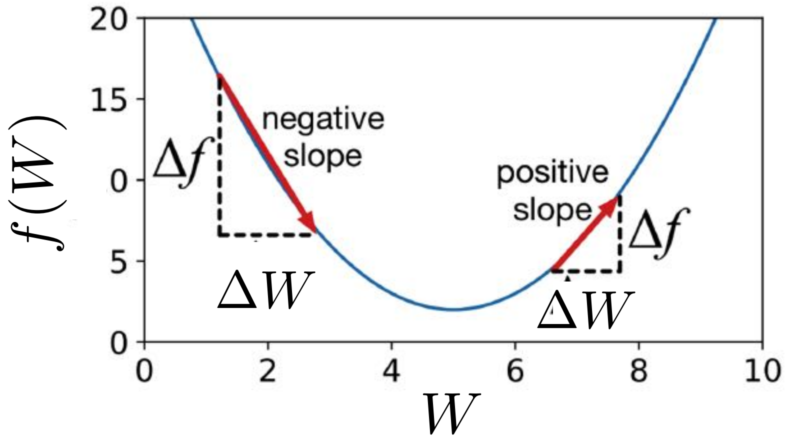
Gradient Descent

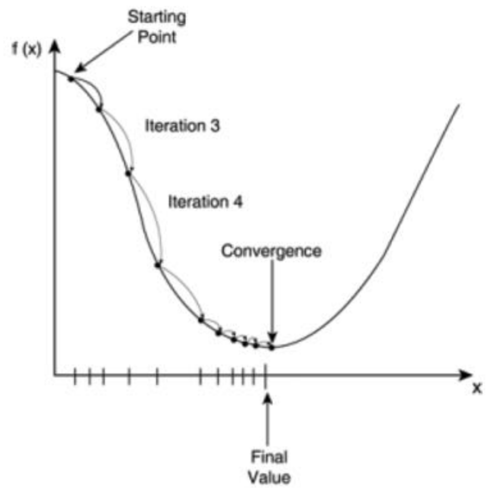
General gradient descent: Start with initial point W_0

$$\text{Sequence:} \quad W_{t+1} = W_t + \alpha_t d_t$$

Steepest Descent:

$d_t = -\nabla f(W_t)$ (we move in the opposite direction of the gradient).





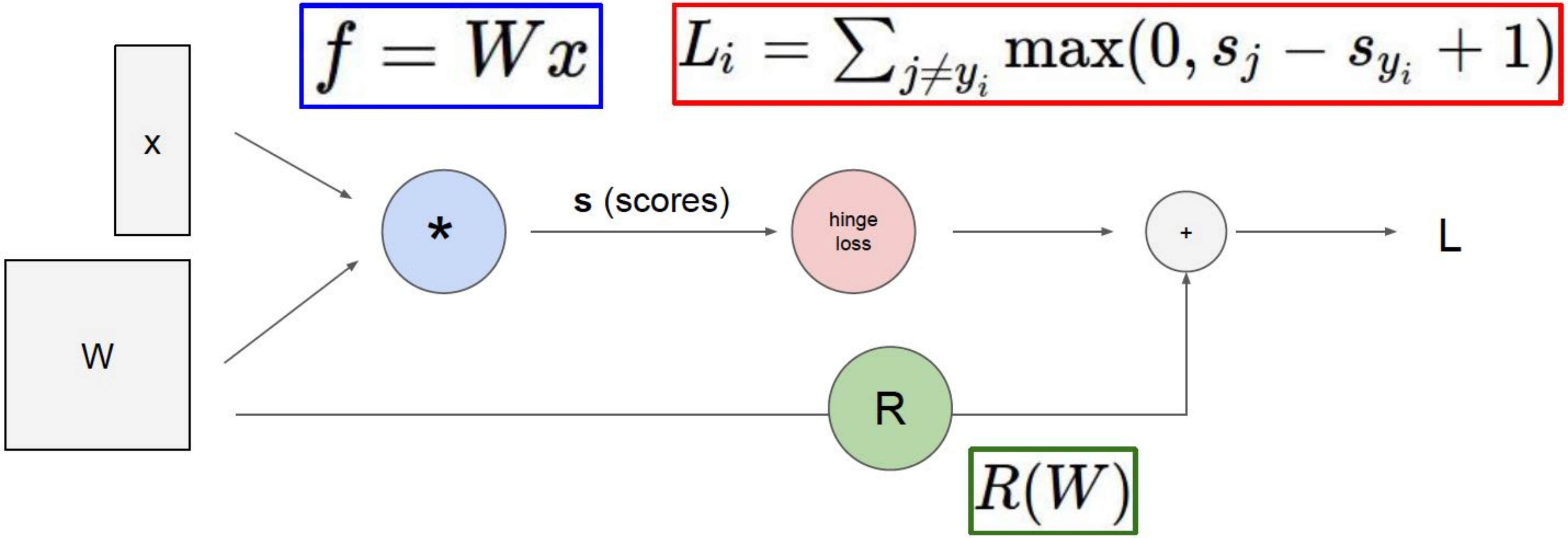

```
# Vanilla Gradient Descent

while True:
  weights_grad = evaluate_gradient(loss_fun, data, weights)
  weights += - step_size * weights_grad # perform parameter update
```

Gradient Descent - Variants…

- Assume Loss to be: $$L(W) = \frac{1}{n}\sum_{i=1}^{n} L_i(W)$$
  - with n the number of training samples
  - L_i the loss for training sample x_i

- **Stochastic Gradient Descent**:
  - randomly choose one training sample x_i
  - update weights based on loss $L_i(W)$
- **Mini-batch training**:
  - process a subset of training samples $M \subset \{1, \ldots, n\}$
  - update weights based on $$L_M(W) = \frac{1}{|M|}\sum_{i \in M} L_i(W)$$
- **Batch training**:
  - process all training samples
  - update weights based on $$L(W) = \frac{1}{n}\sum_{i=1}^{n} L_i(W)$$

Computational Graphs



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Neural Network Example…

Convolutional network (AlexNet)

input image

weights

loss

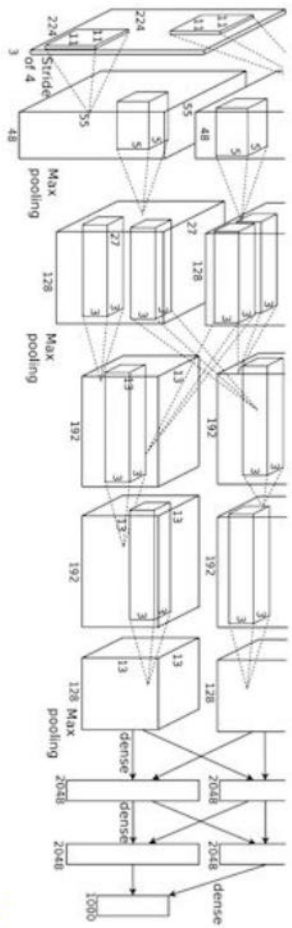


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Neural Network Example…

Neural Turing Machine

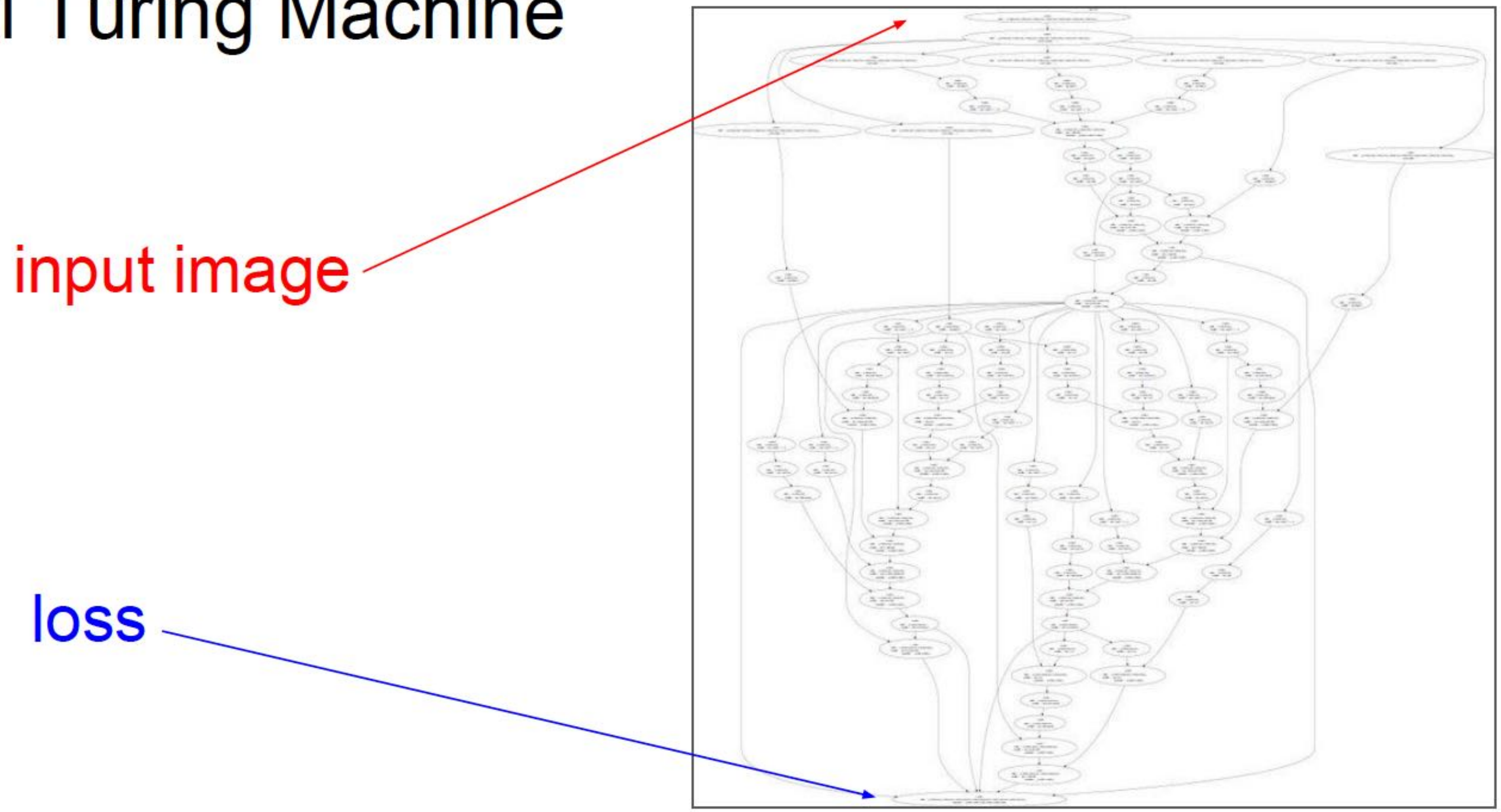


Figure reproduced with permission from a Twitter post by Andrej Karpathy.

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Backpropagation - A Simple Example

Backpropagation: a simple example

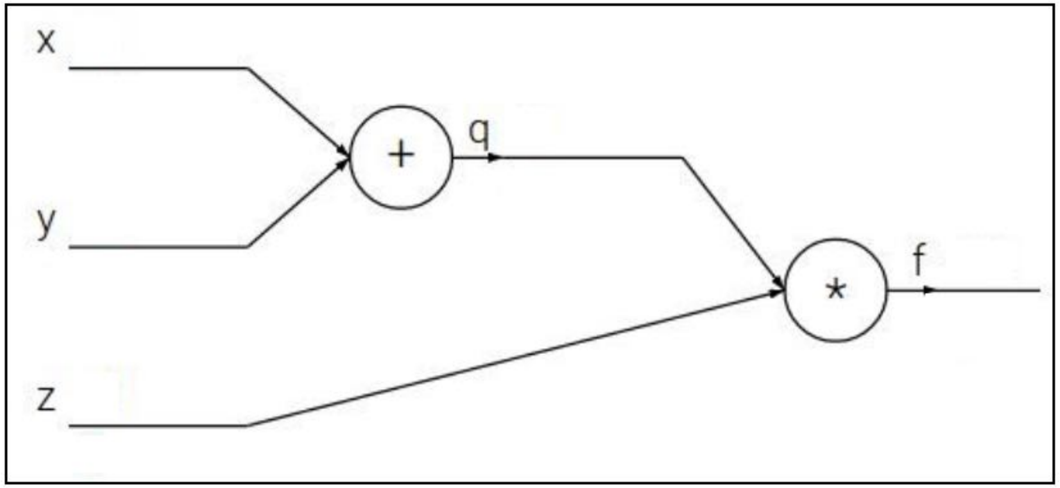
$$f(x, y, z) = (x + y)z$$

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Backpropagation - A Simple Example

Backpropagation: a simple example

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



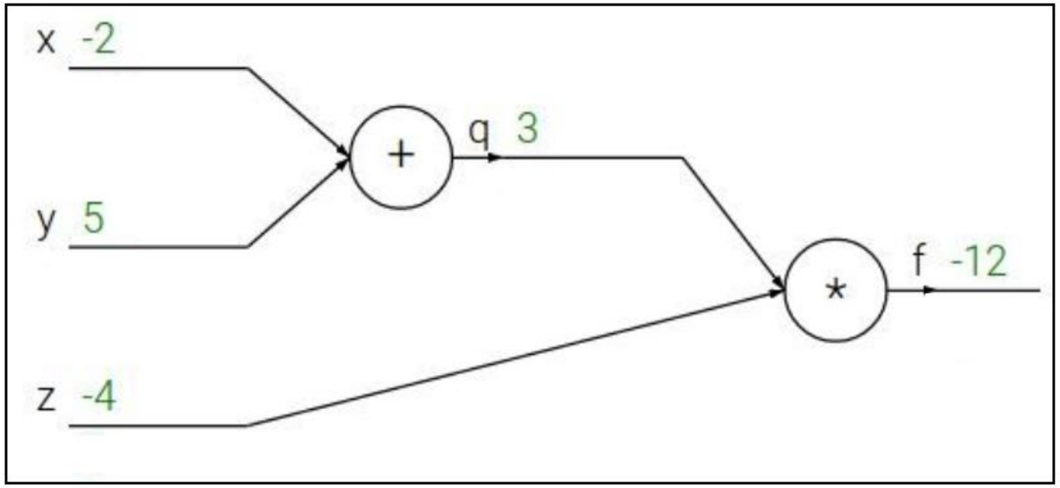
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Backpropagation - A Simple Example

Backpropagation: a simple example

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

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



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Backpropagation - A Simple Example

Backpropagation: a simple example

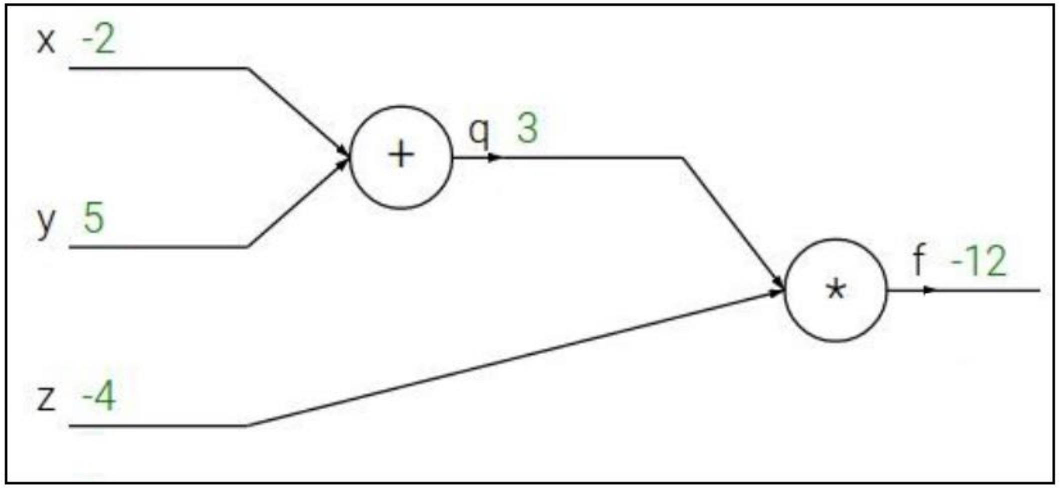
$$f(x, y, z) = (x + y)z$$

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



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Backpropagation - A Simple Example

Backpropagation: a simple example

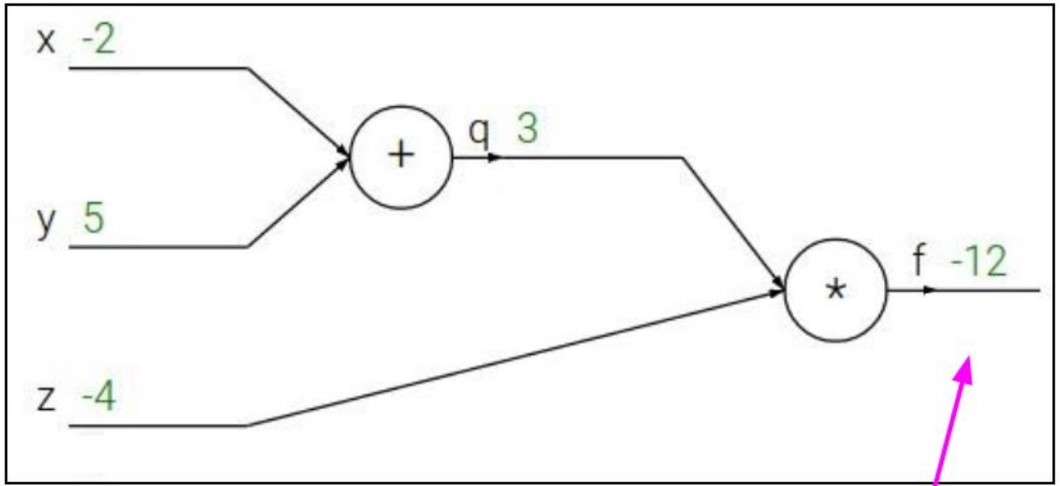
$$f(x, y, z) = (x + y)z$$

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



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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Backpropagation - A Simple Example

Backpropagation: a simple example

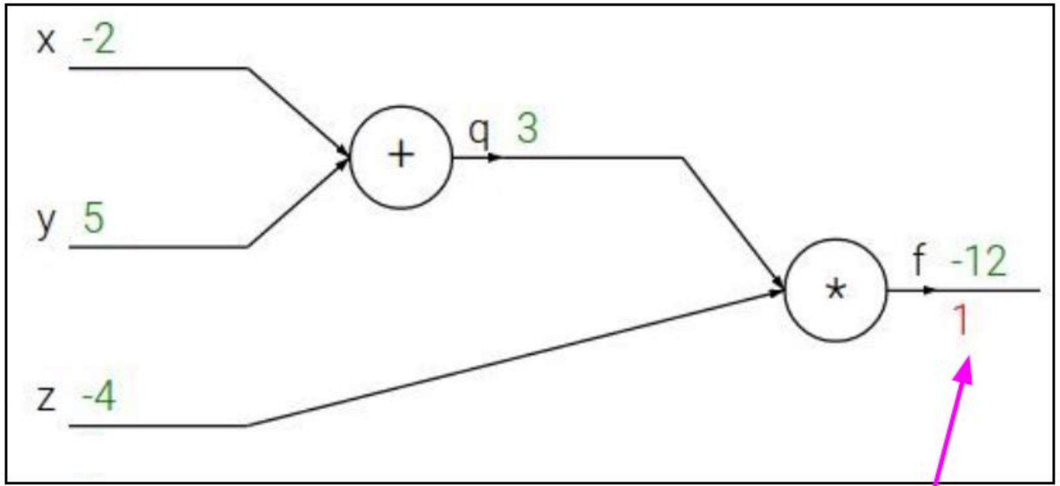
$$f(x, y, z) = (x + y)z$$

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



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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Backpropagation - A Simple Example

Backpropagation: a simple example

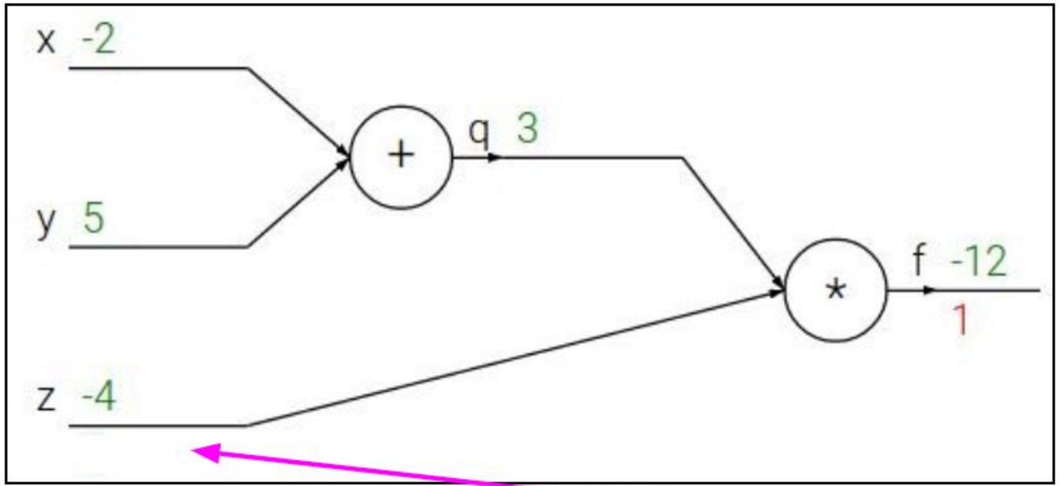
$$f(x, y, z) = (x + y)z$$

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



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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Backpropagation - A Simple Example

Backpropagation: a simple example

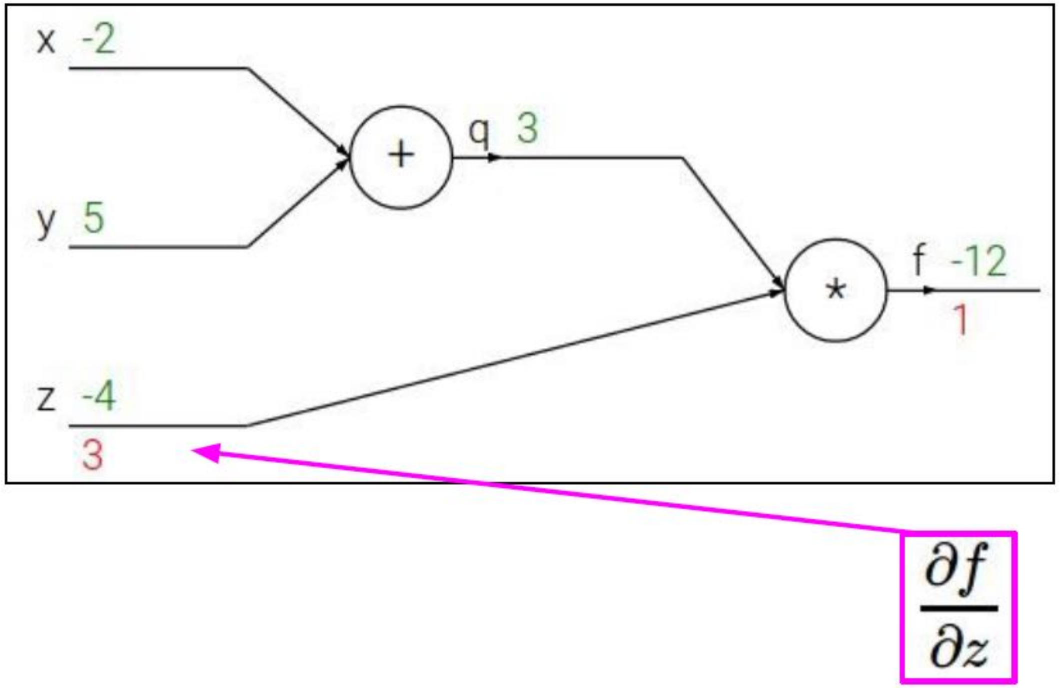
$$f(x, y, z) = (x + y)z$$

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



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Backpropagation - A Simple Example

Backpropagation: a simple example

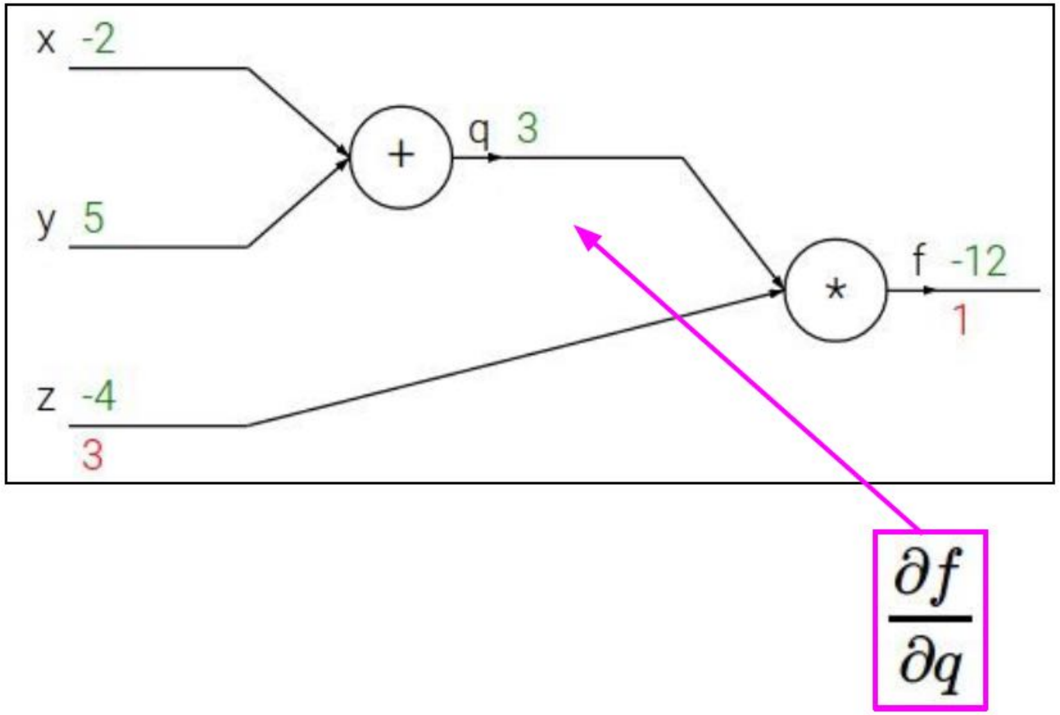
$$f(x, y, z) = (x + y)z$$

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



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Backpropagation - A Simple Example

Backpropagation: a simple example

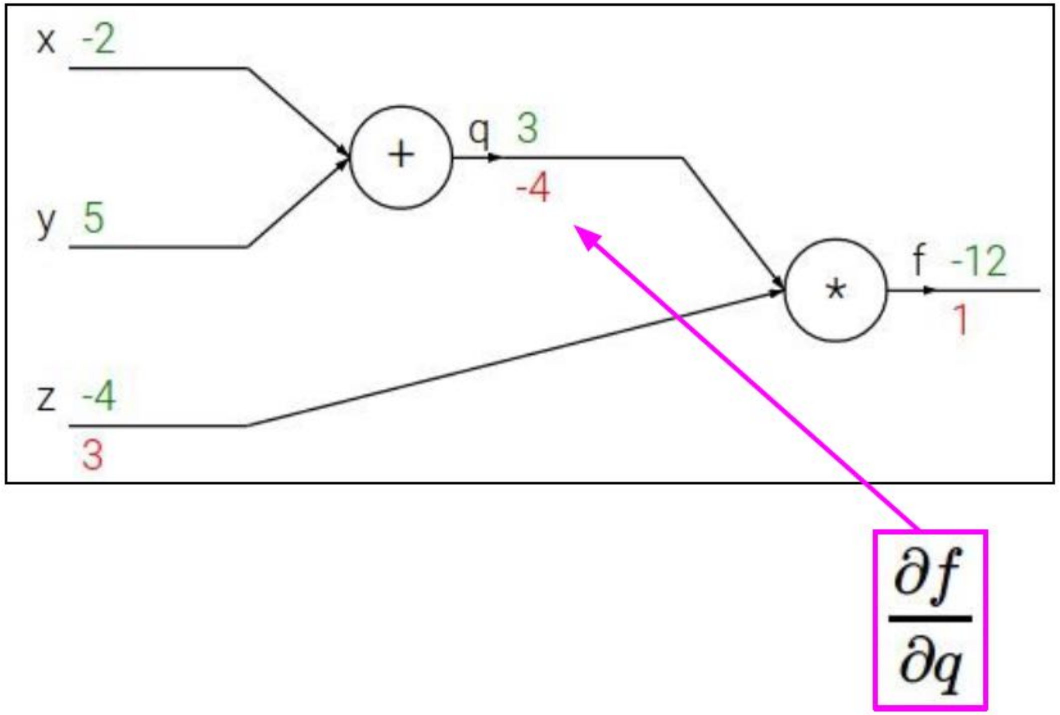
$$f(x, y, z) = (x + y)z$$

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



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Backpropagation - A Simple Example

Backpropagation: a simple example

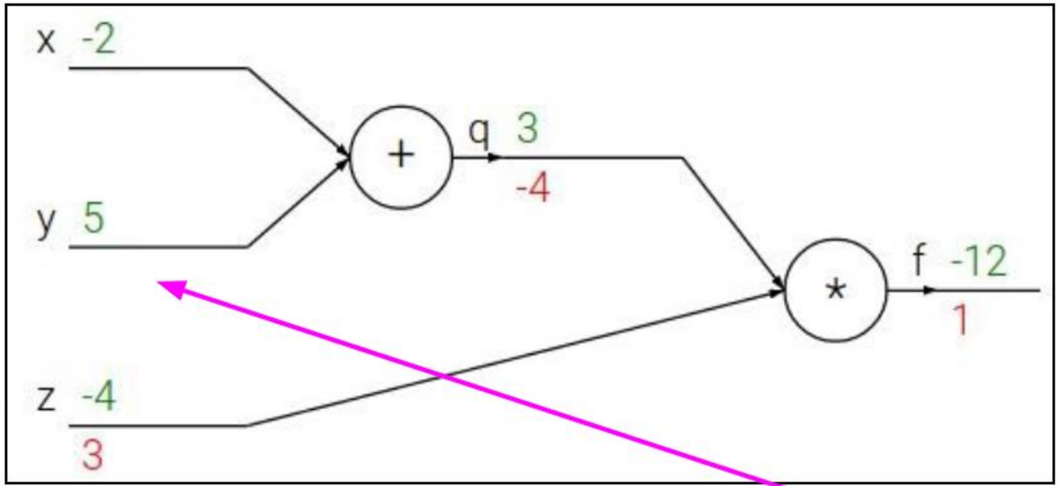
$$f(x, y, z) = (x + y)z$$

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Backpropagation - A Simple Example

Backpropagation: a simple example

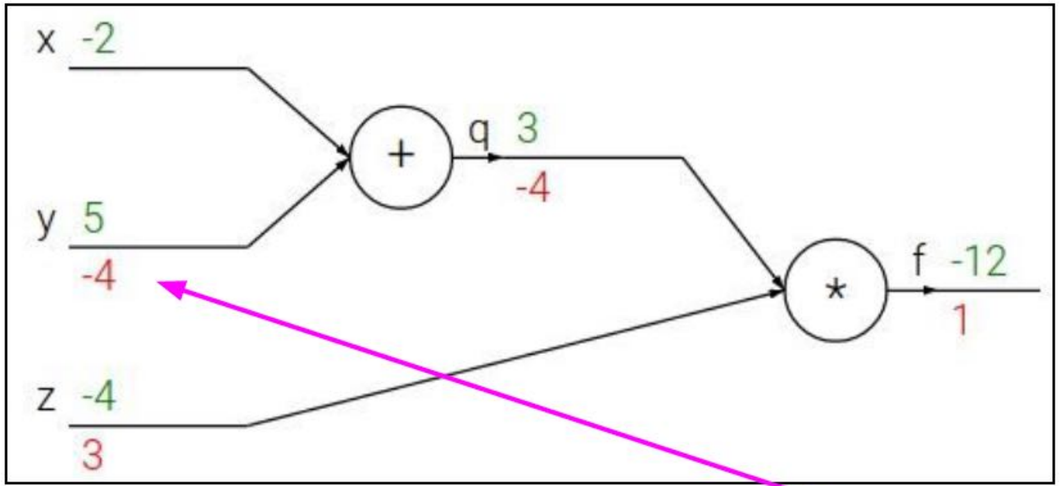
$$f(x, y, z) = (x + y)z$$

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Backpropagation - A Simple Example

Backpropagation: a simple example

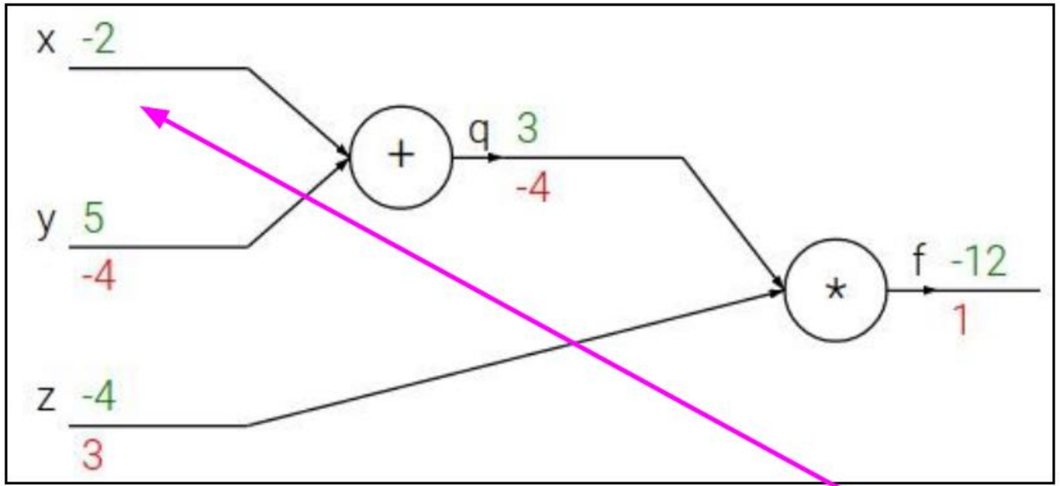
$$f(x, y, z) = (x + y)z$$

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:

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

Upstream gradient — Local gradient

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Backpropagation - A Simple Example

Backpropagation: a simple example

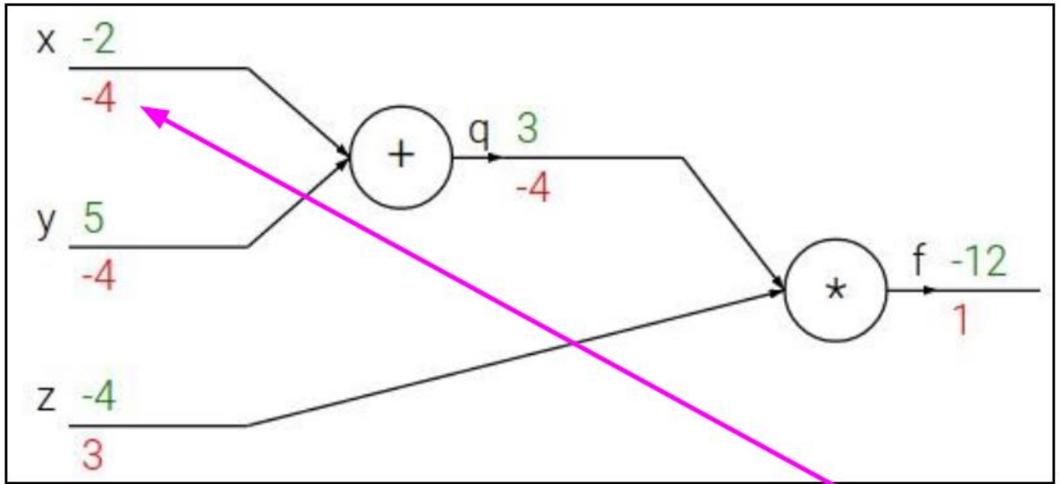
$$f(x, y, z) = (x + y)z$$

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:

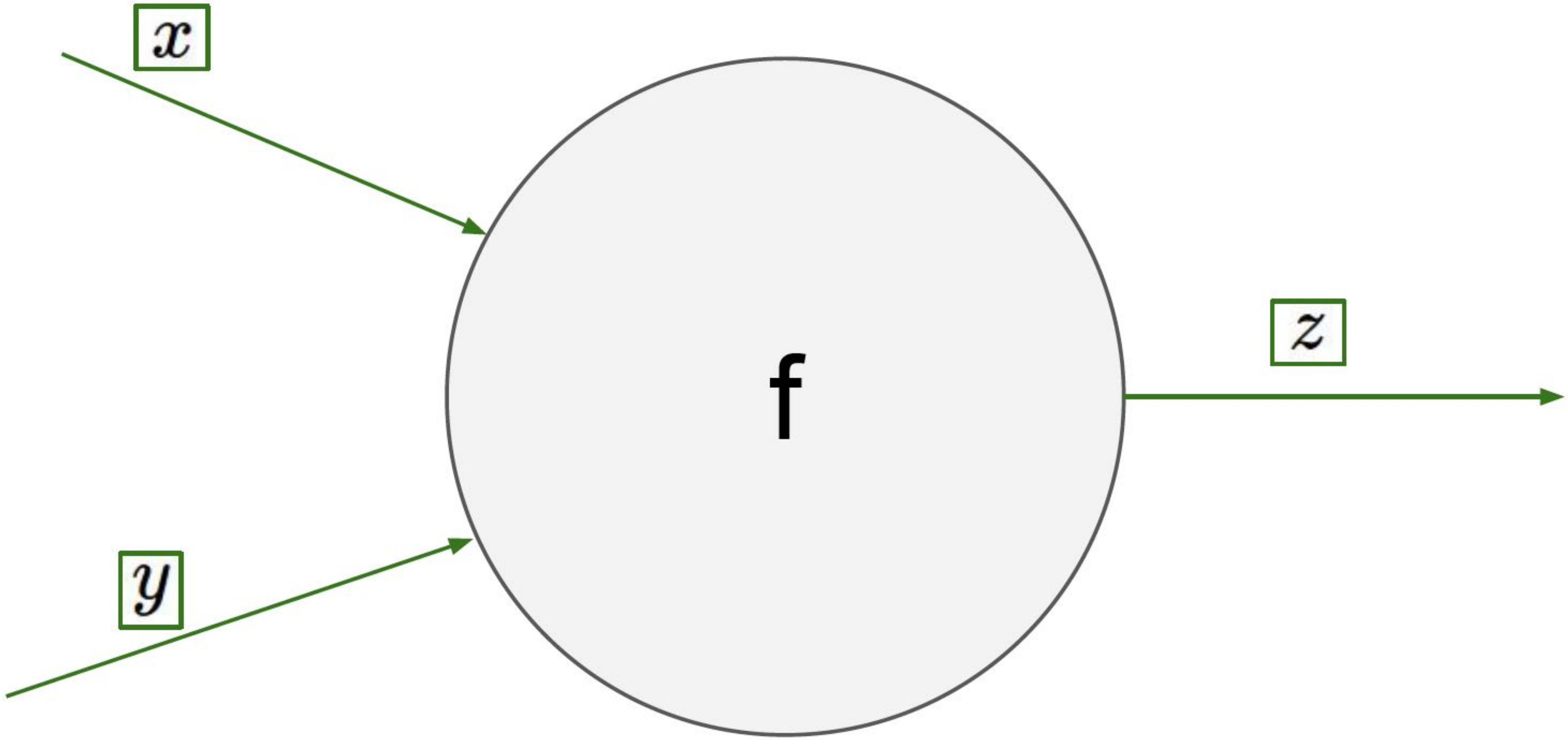
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial x}$$

Upstream gradient

Local gradient

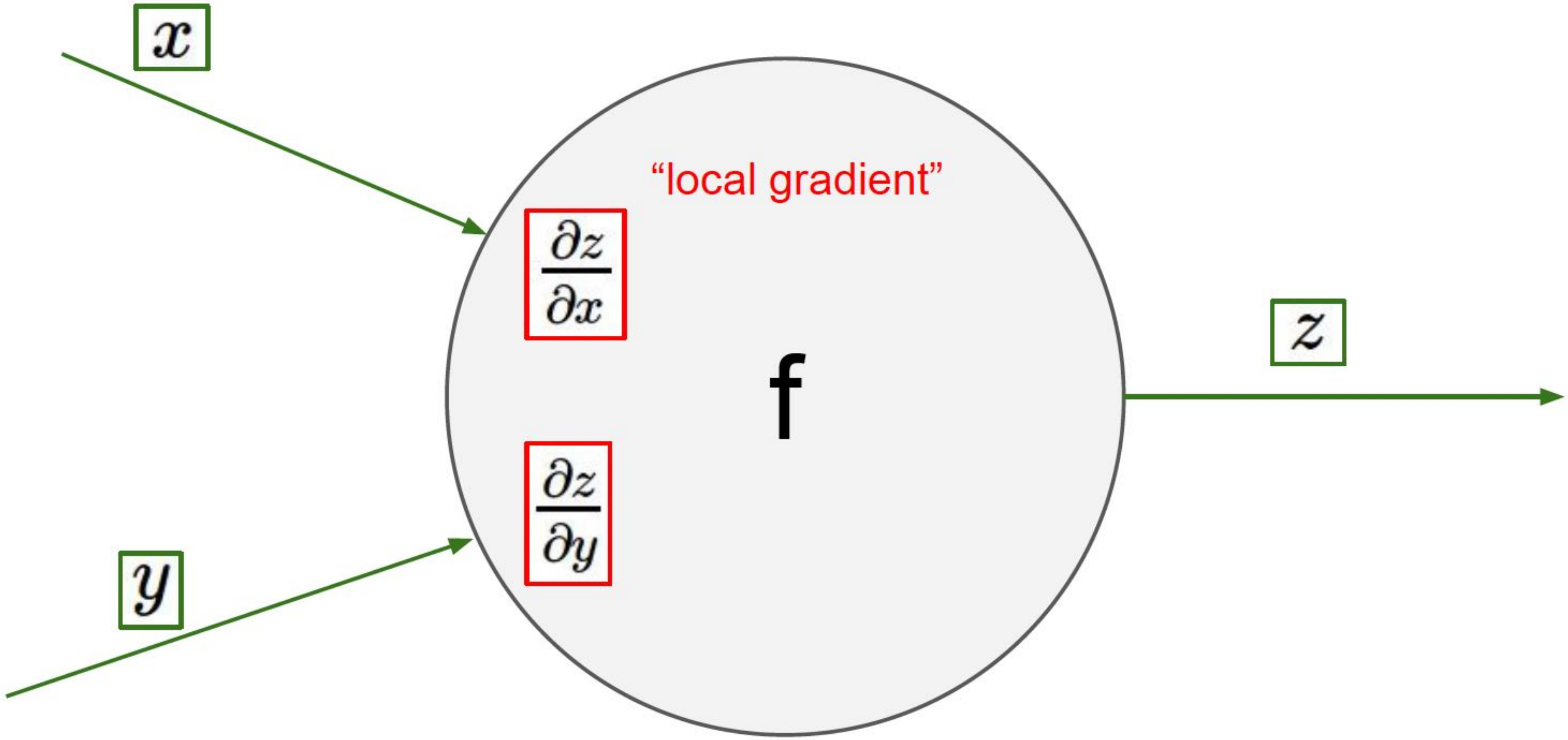
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Backpropagation - Local View



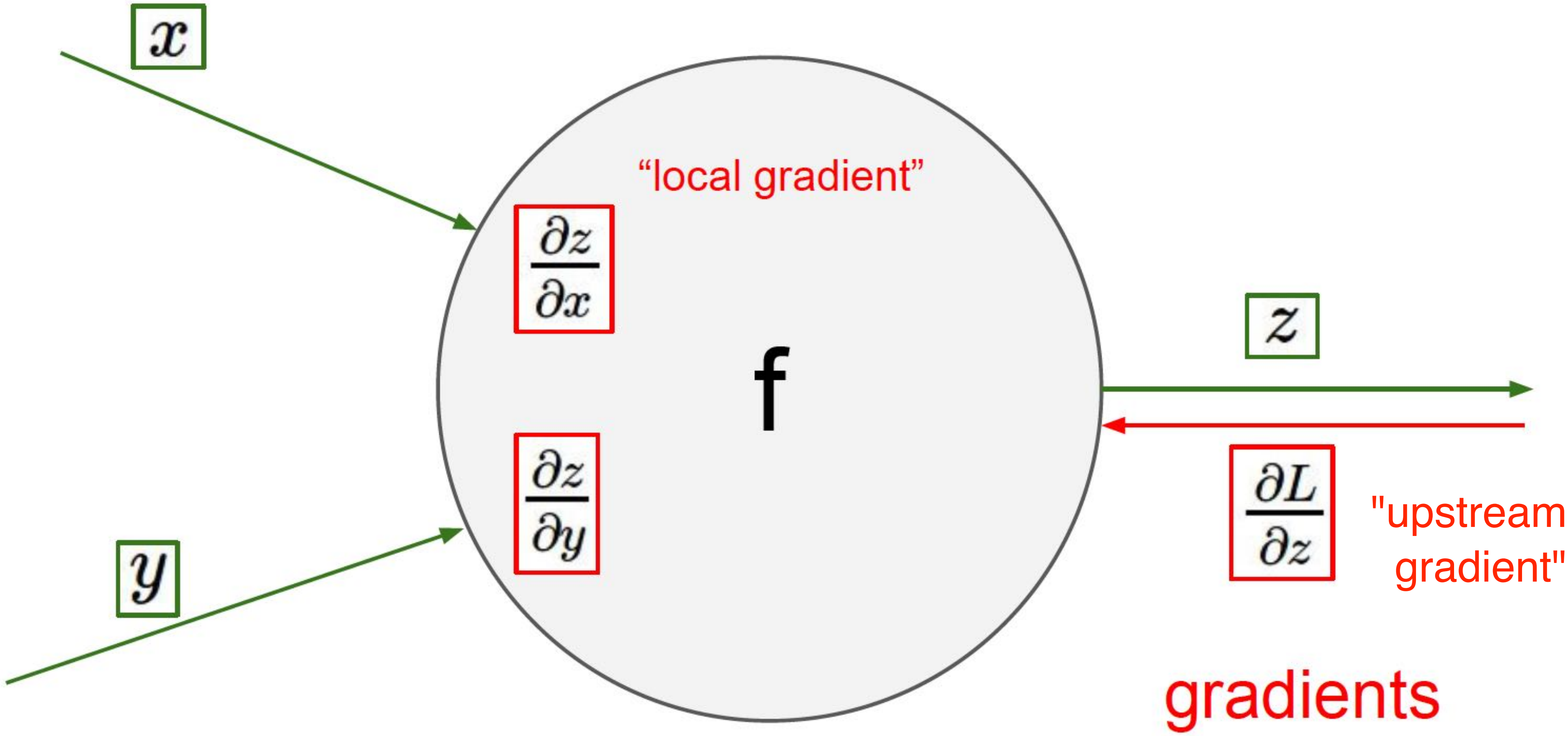
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Backpropagation - Local View



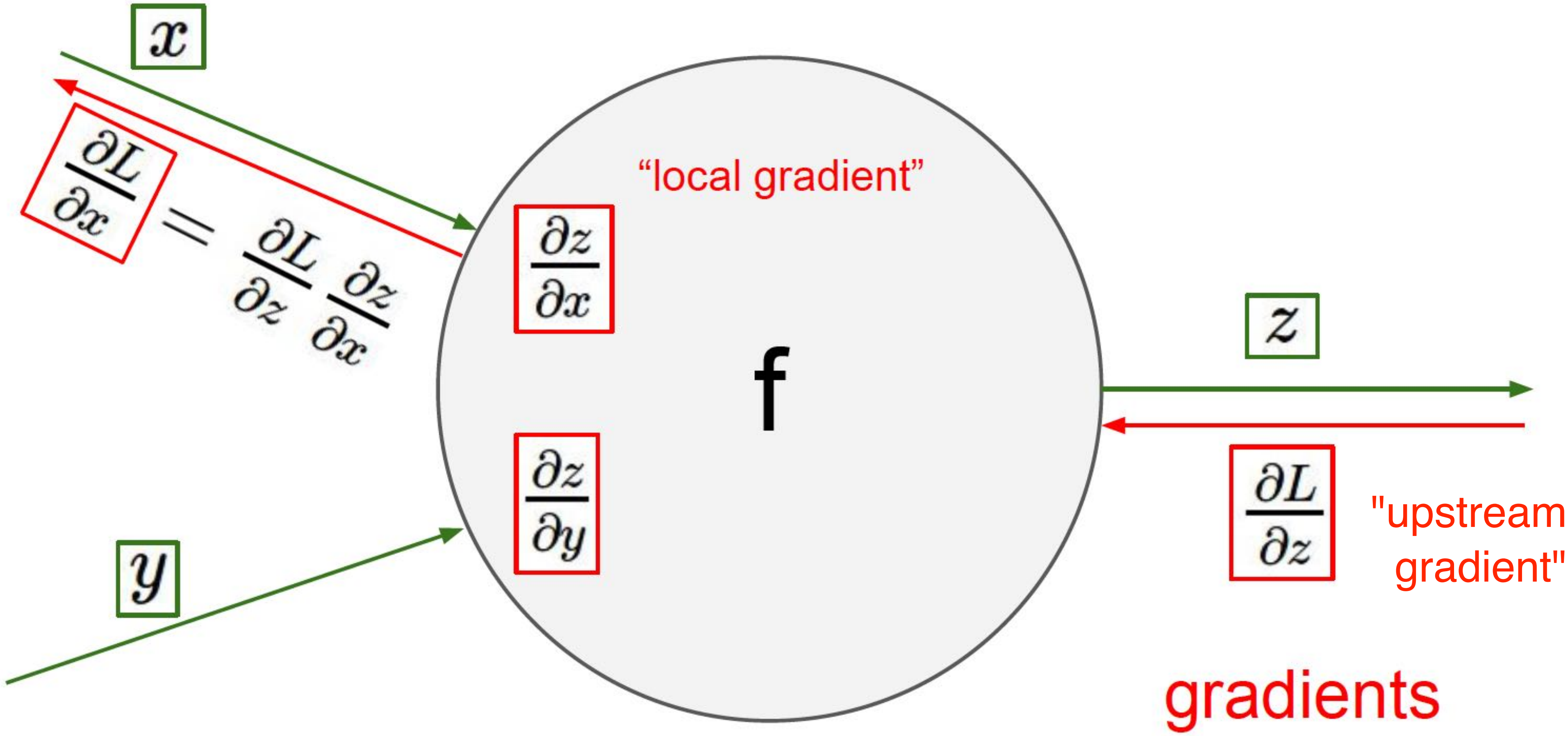
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Backpropagation - Local View



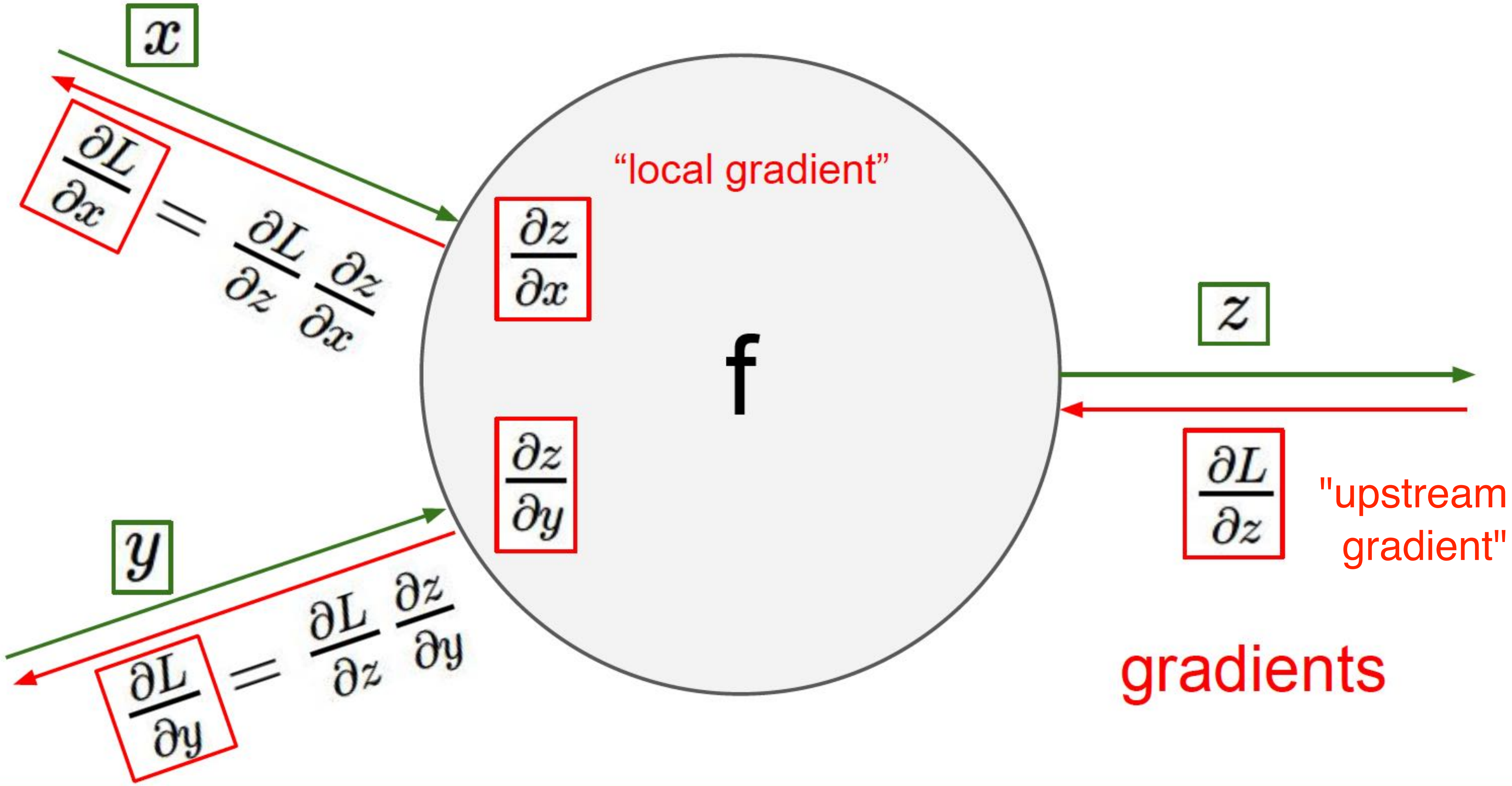
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Backpropagation - Local View



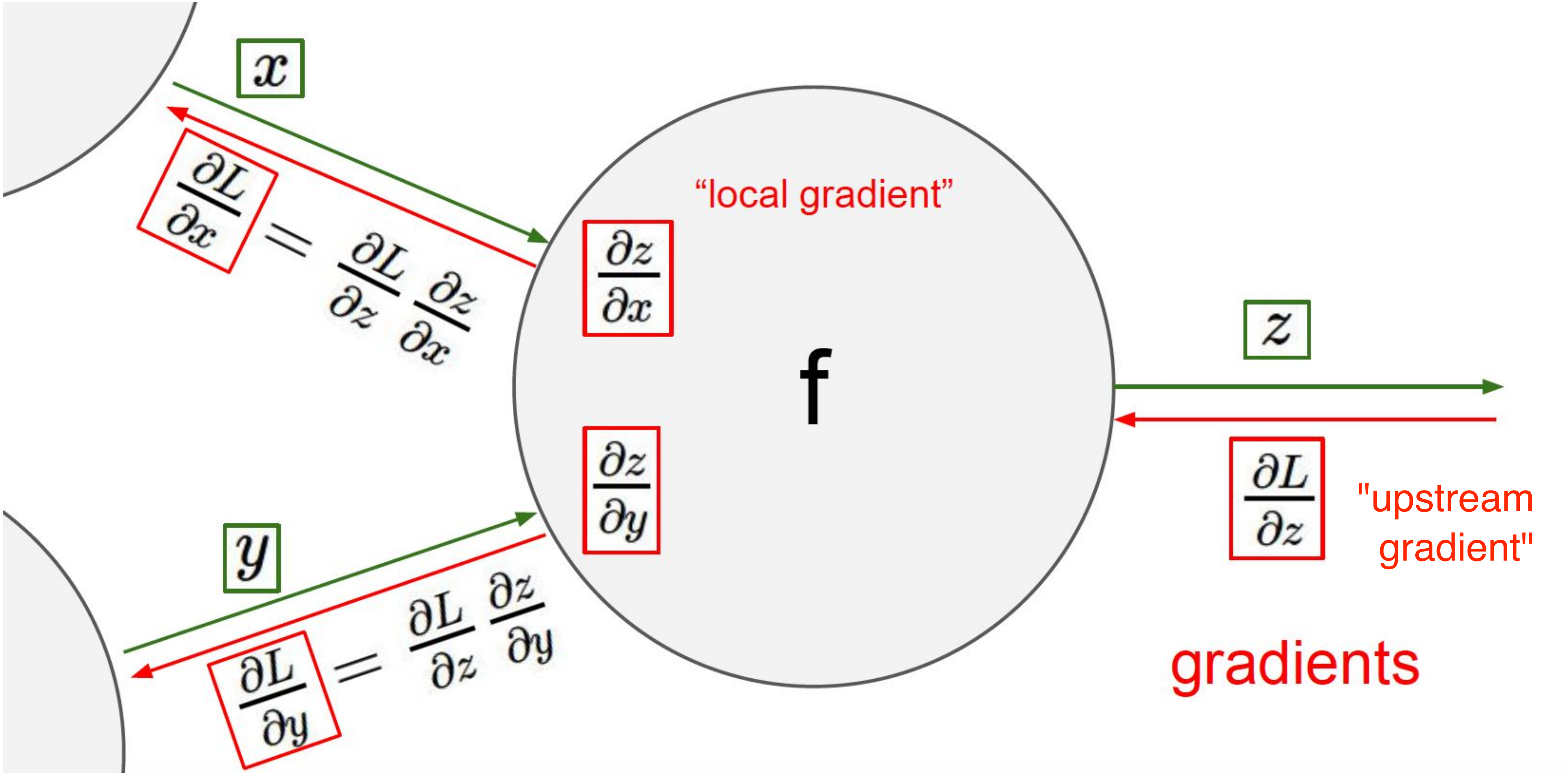
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Backpropagation - Local View



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

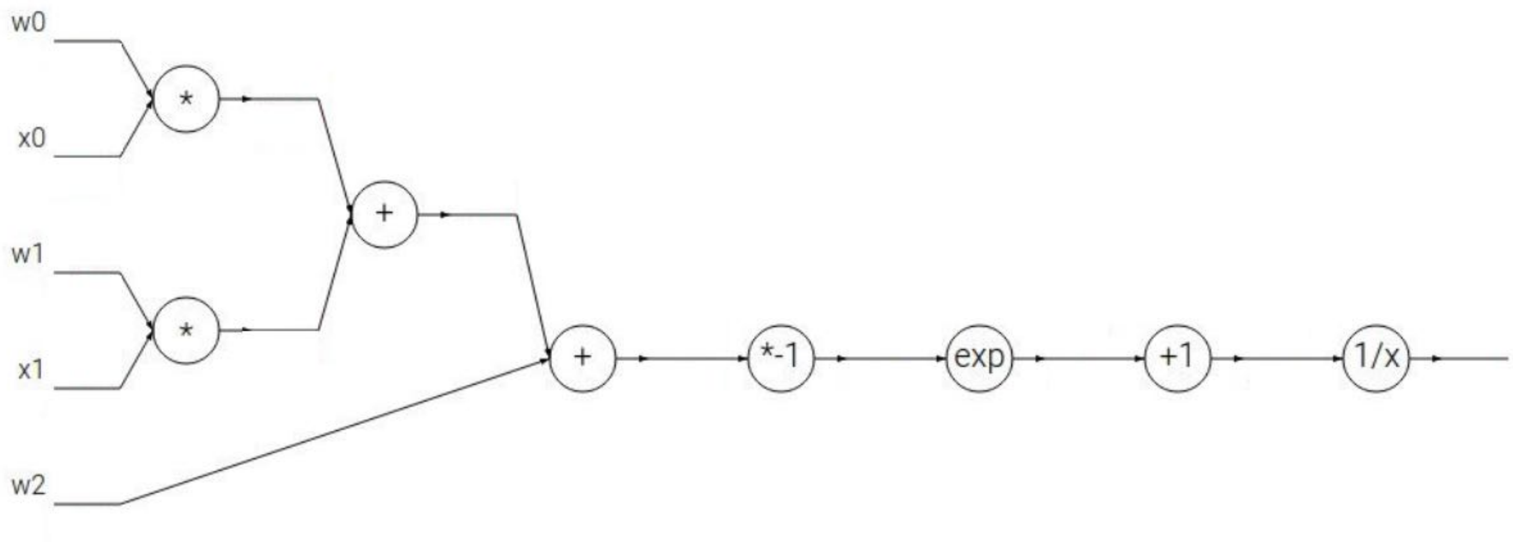
Backpropagation - Local View



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Backpropagation

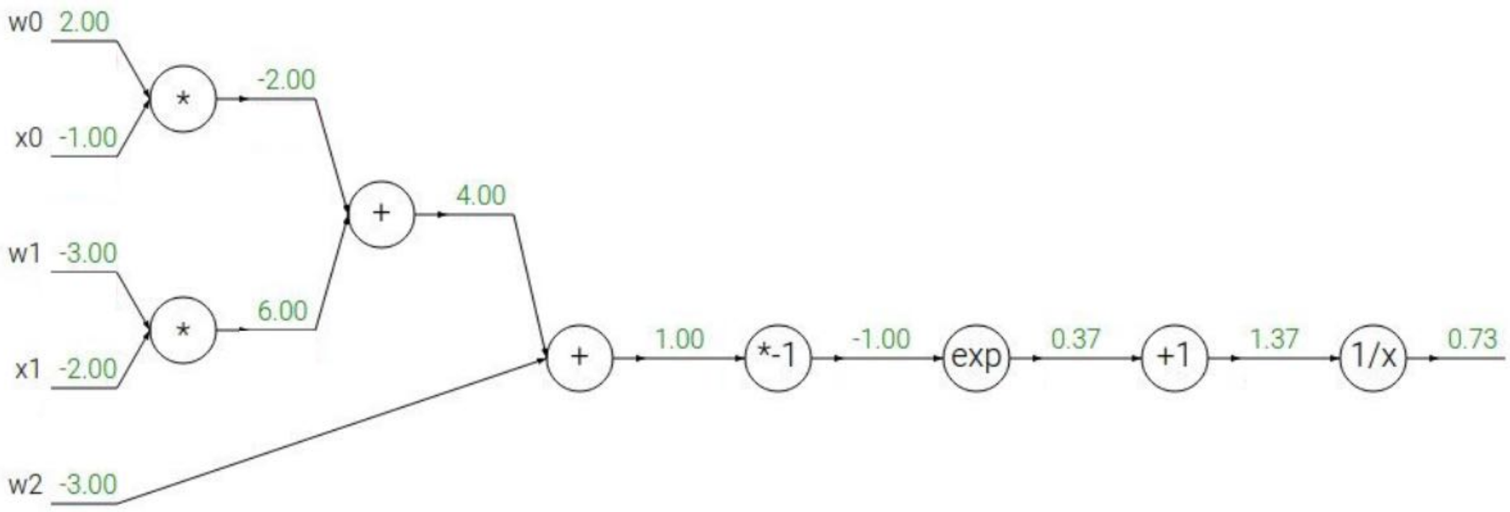
Another example: $$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Backpropagation

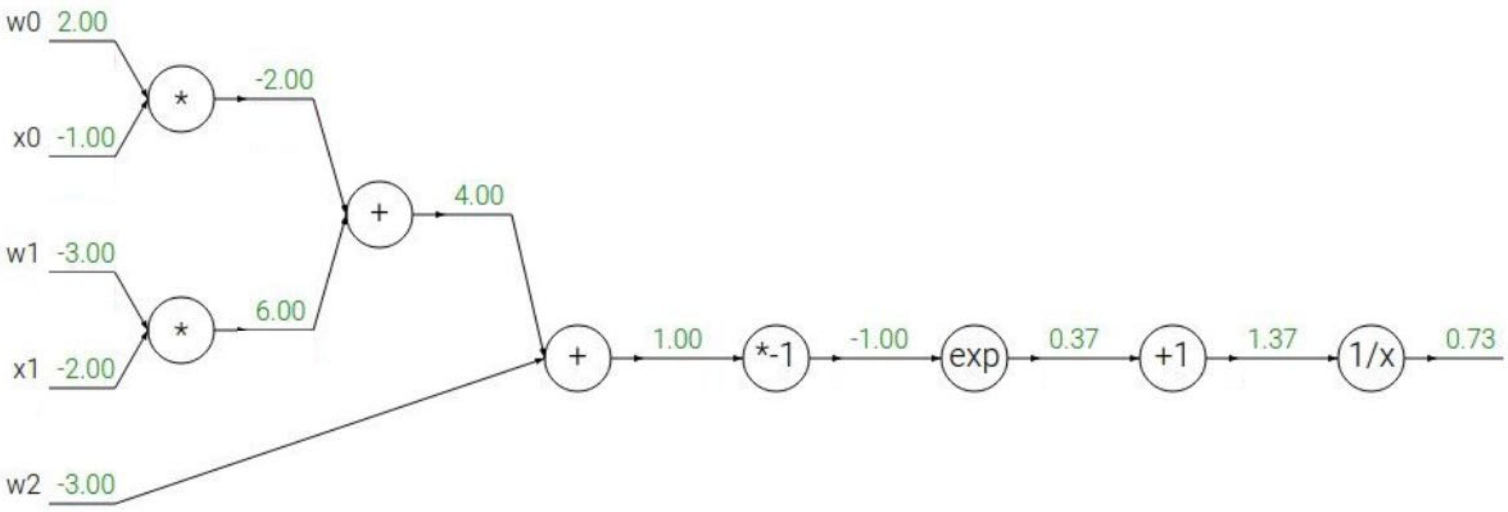
Another example: $$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Backpropagation

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

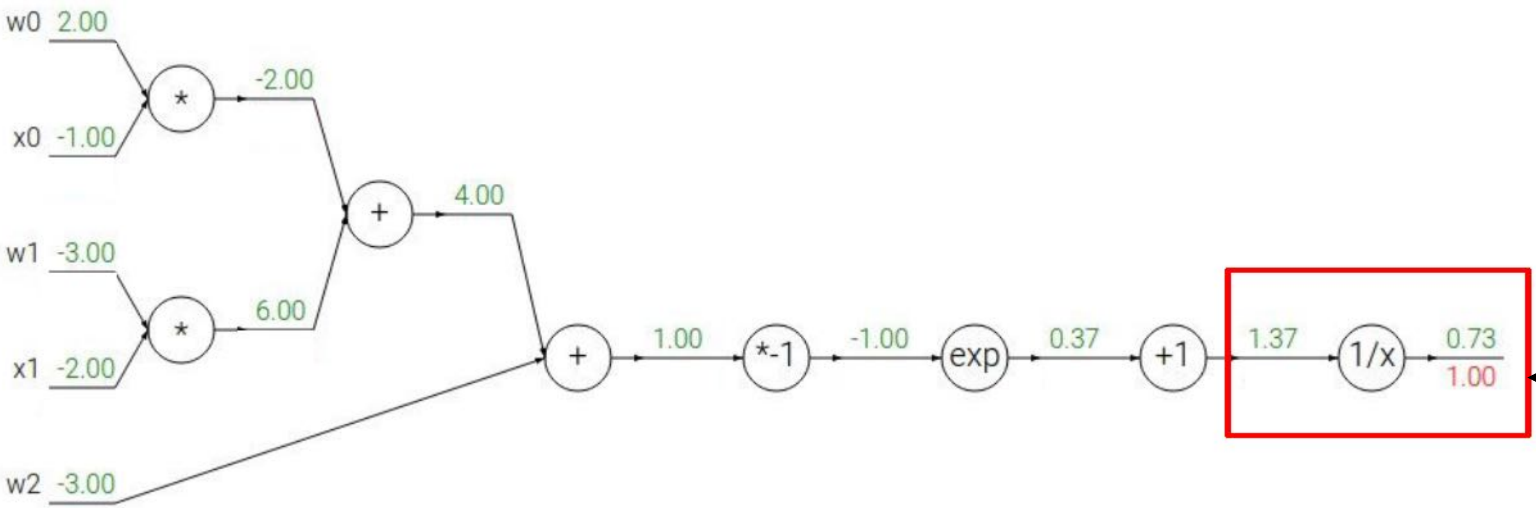


$f(x) = e^x$	$\rightarrow$	$\frac{df}{dx} = e^x$	$f(x) = \frac{1}{x}$	$\rightarrow$	$\frac{df}{dx} = -1/x^2$
$f_a(x) = ax$	$\rightarrow$	$\frac{df}{dx} = a$	$f_c(x) = c + x$	$\rightarrow$	$\frac{df}{dx} = 1$

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Backpropagation

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



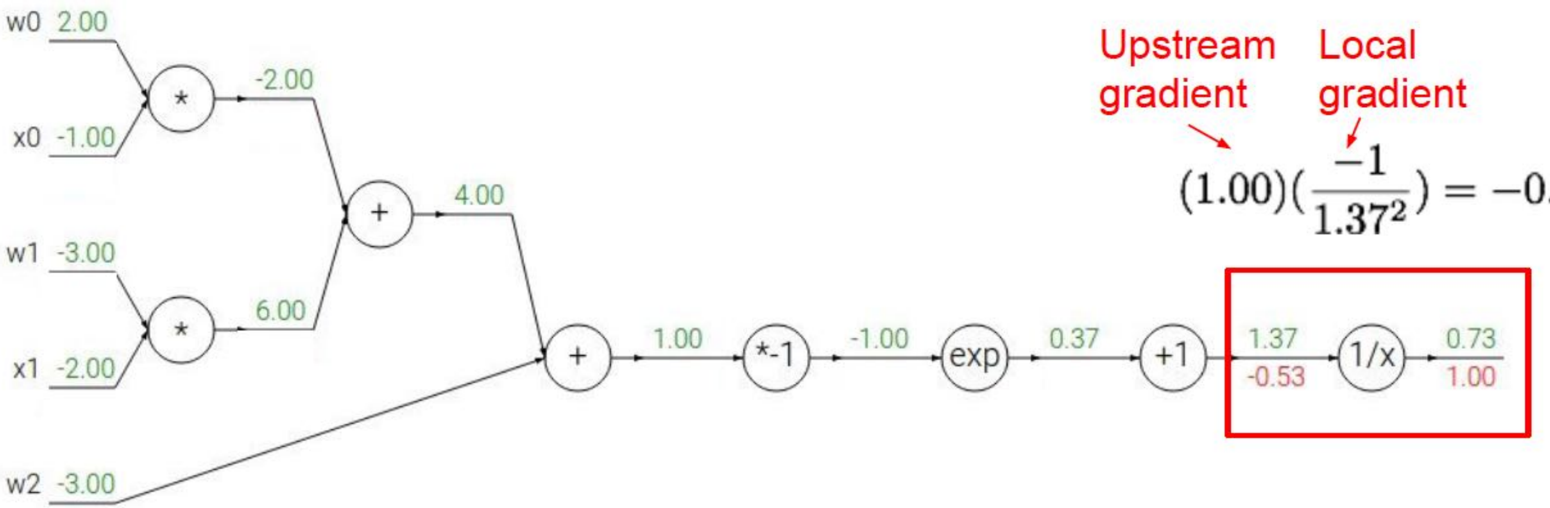
$$\frac{\partial f}{\partial f}$$

$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$	$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$
$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$	$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Backpropagation

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



$(1.00)(\frac{-1}{1.37^2}) = -0.53$

$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$

$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$

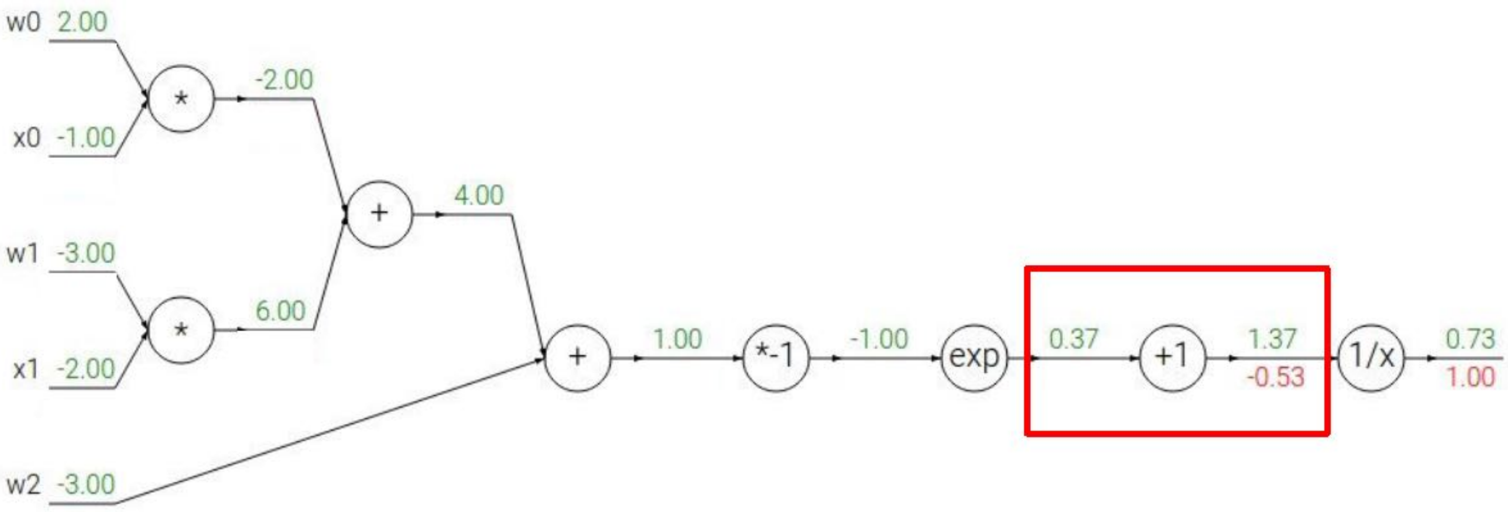
$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$

$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Backpropagation

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



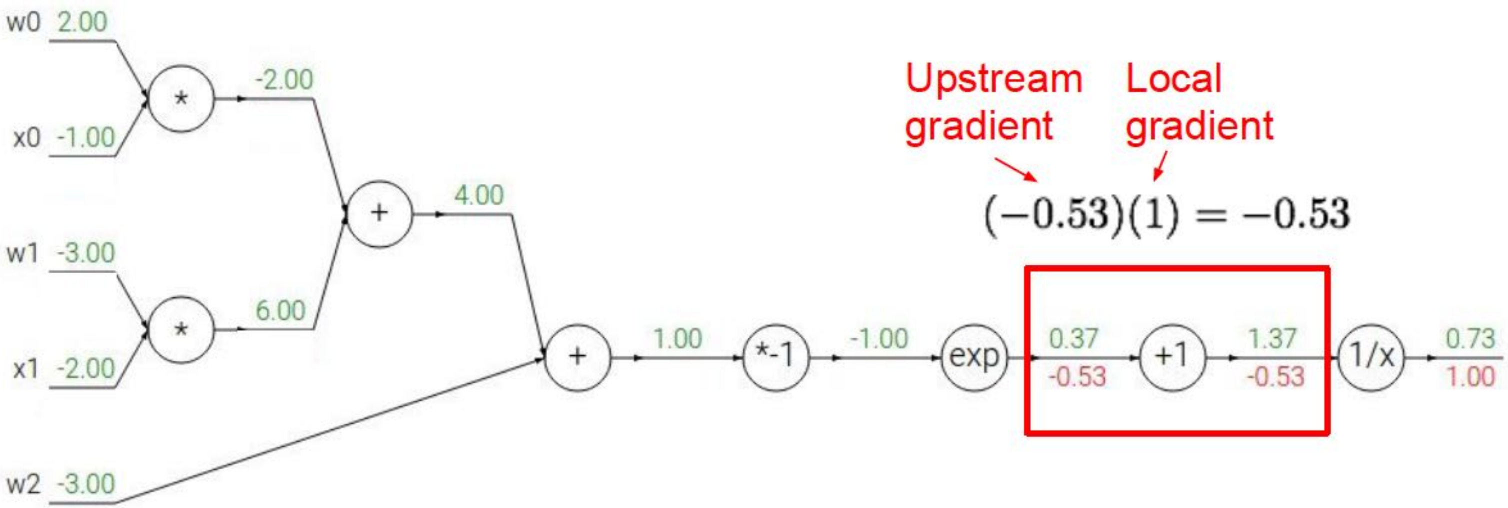
$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$ | $f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$

$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$ | $f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Backpropagation

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



$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$

$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$

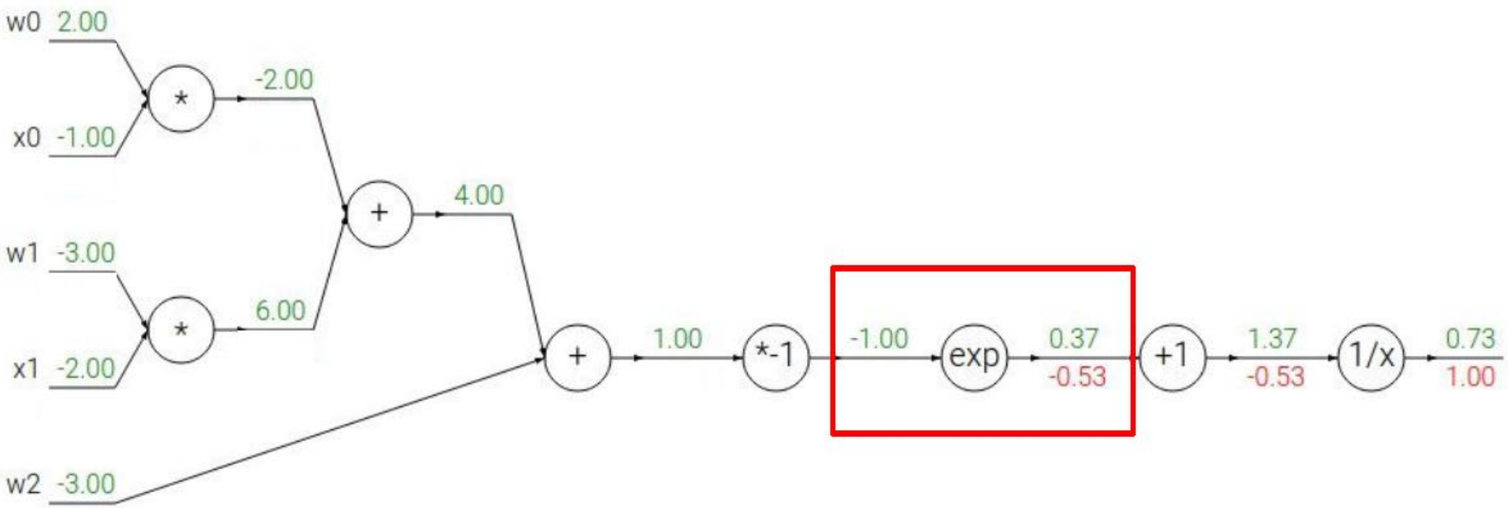
$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$

$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Backpropagation

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



$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$

$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$

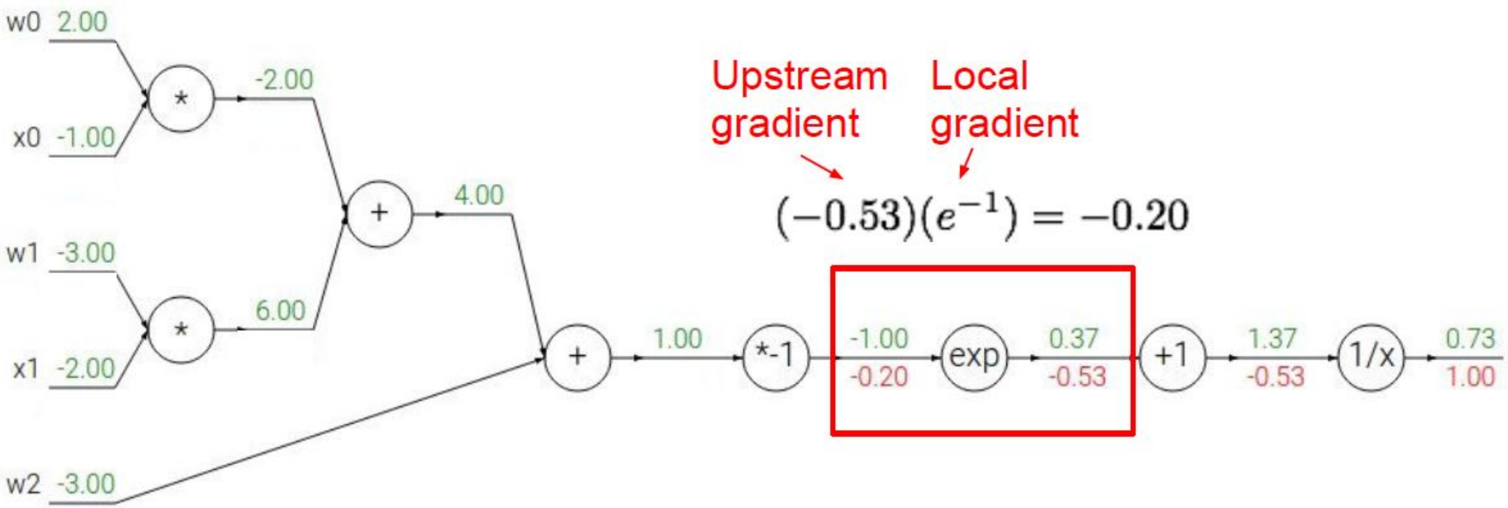
$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$

$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Backpropagation

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



$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$

$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$

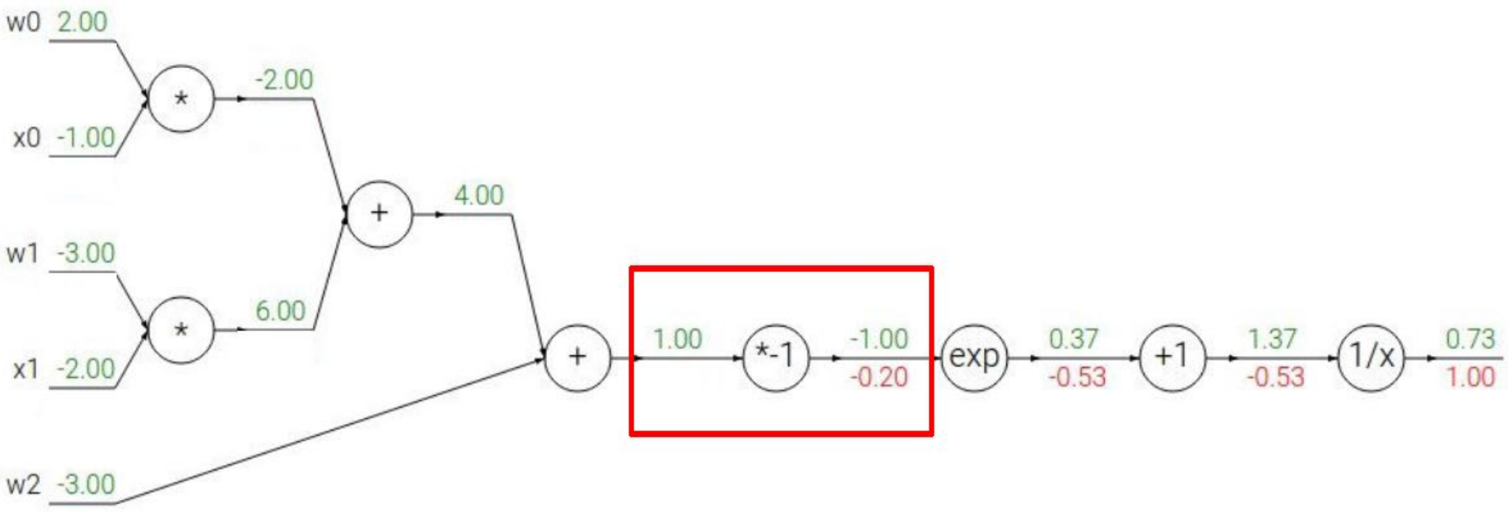
$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$

$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Backpropagation

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



$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$$

$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

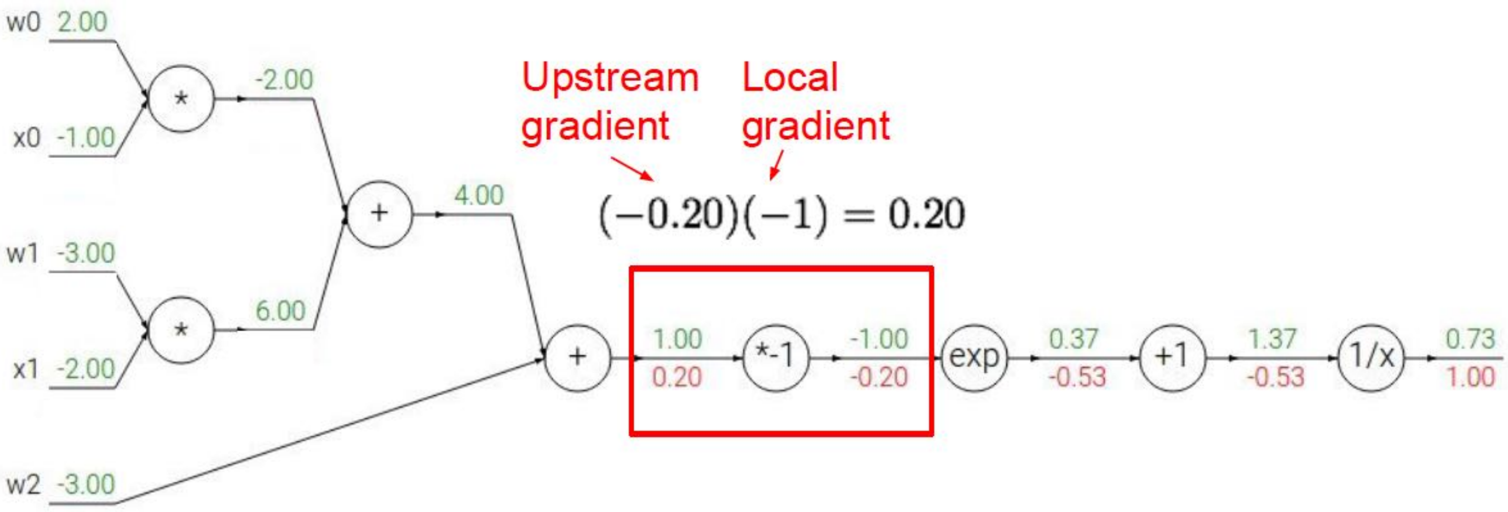
$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Backpropagation

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



$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$

$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$

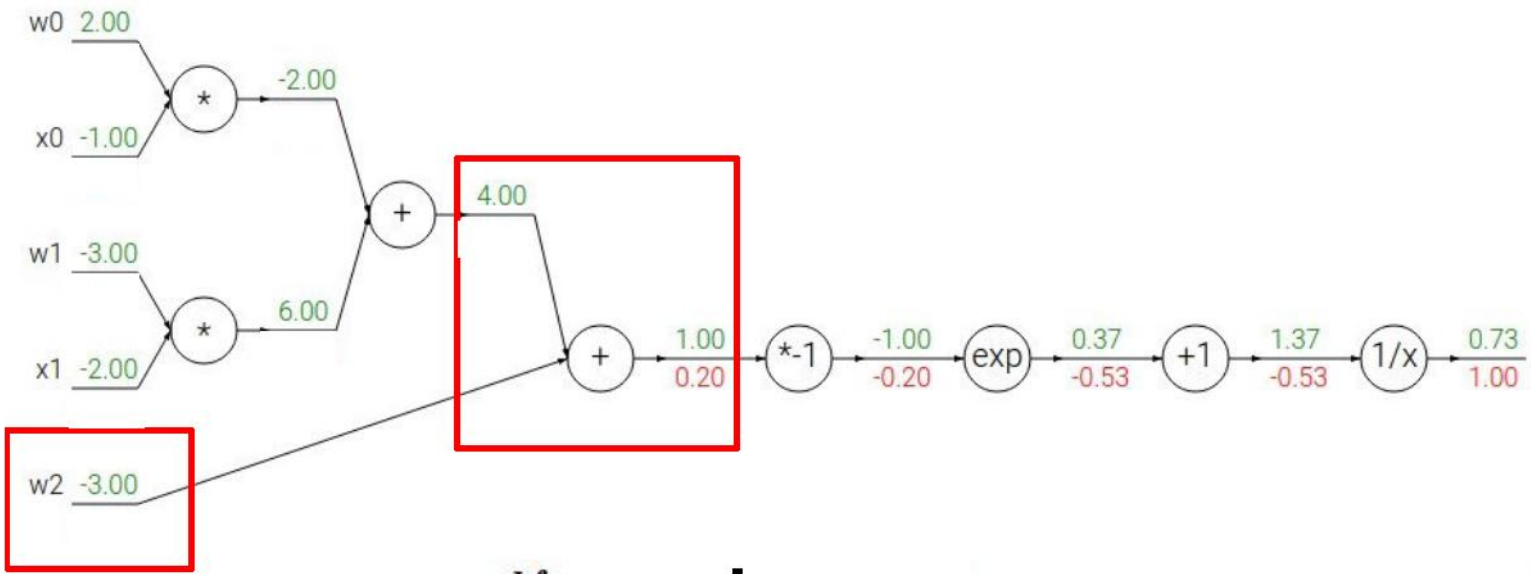
$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$

$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Backpropagation

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



$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$

$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$

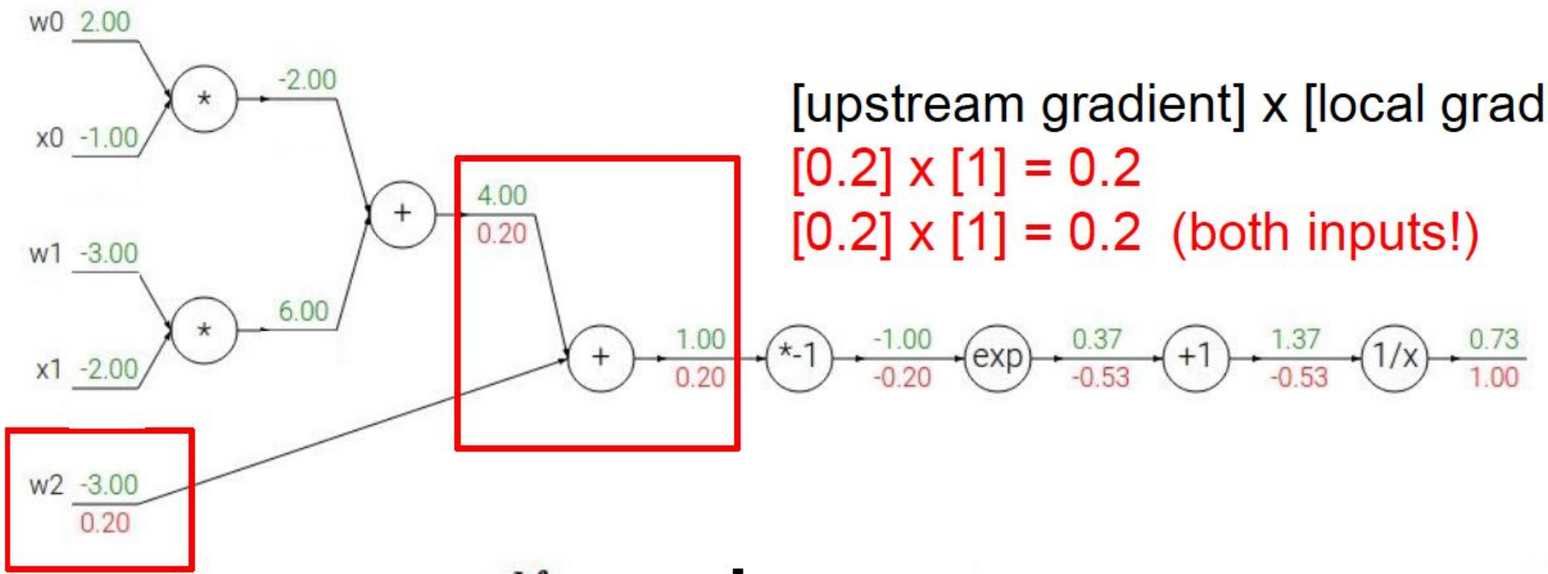
$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$

$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Backpropagation

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



[upstream gradient] x [local gradient]

[0.2] x [1] = 0.2

[0.2] x [1] = 0.2 (both inputs!)

$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$$

$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

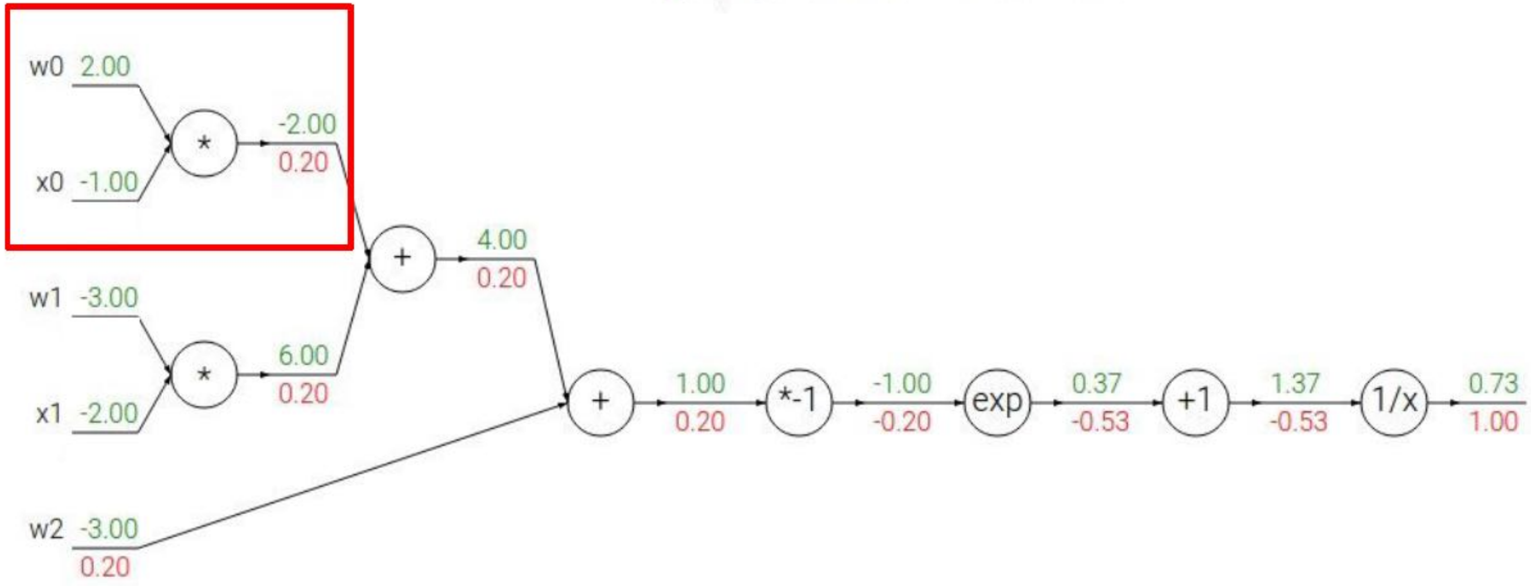
$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Backpropagation

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



$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$$

$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

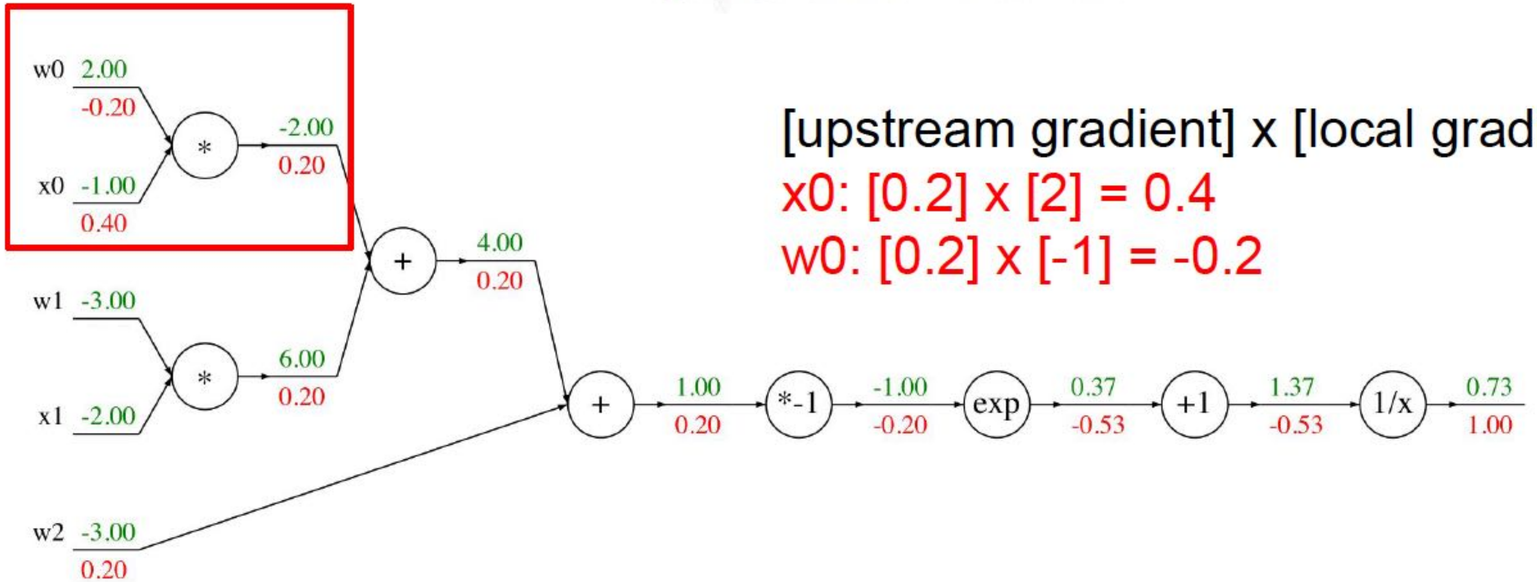
$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Backpropagation

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



[upstream gradient] x [local gradient]

x0: [0.2] x [2] = 0.4

w0: [0.2] x [-1] = -0.2

$$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$$

$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Backpropagation

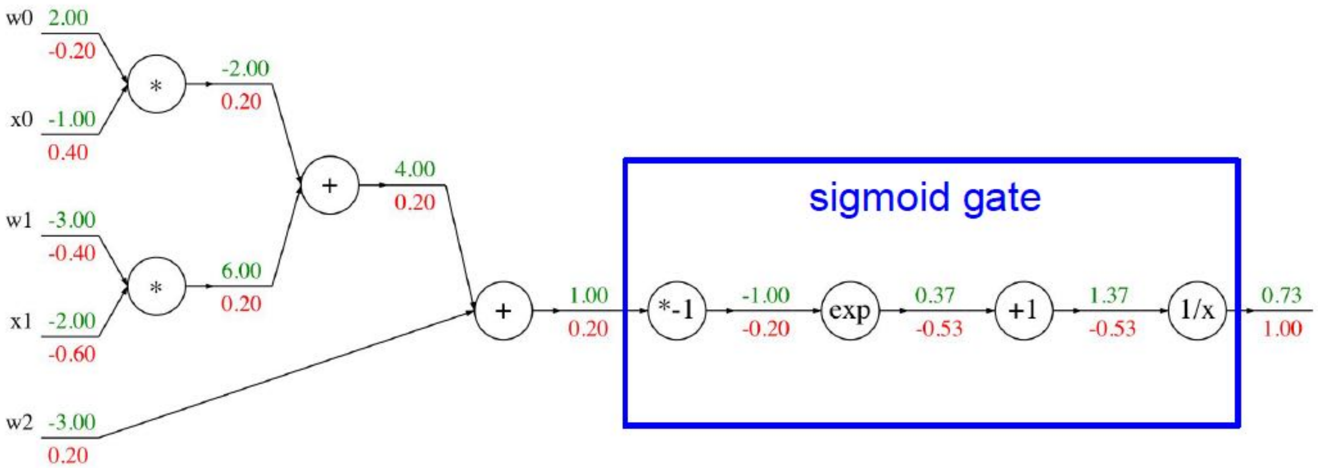
$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$

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

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

sigmoid function

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



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Backpropagation

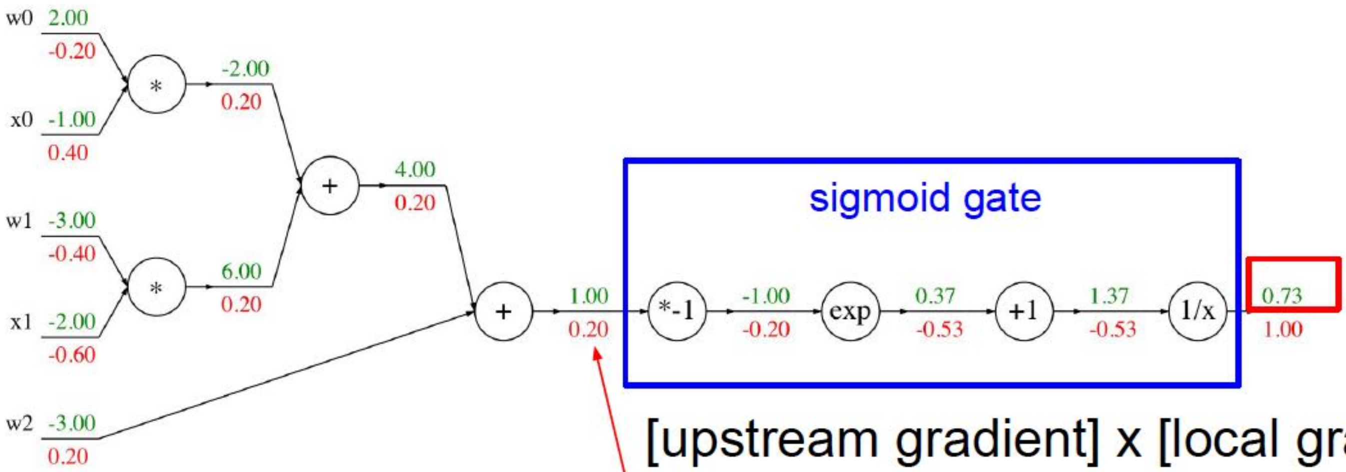
$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$

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

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

sigmoid function

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



[upstream gradient] x [local gradient]

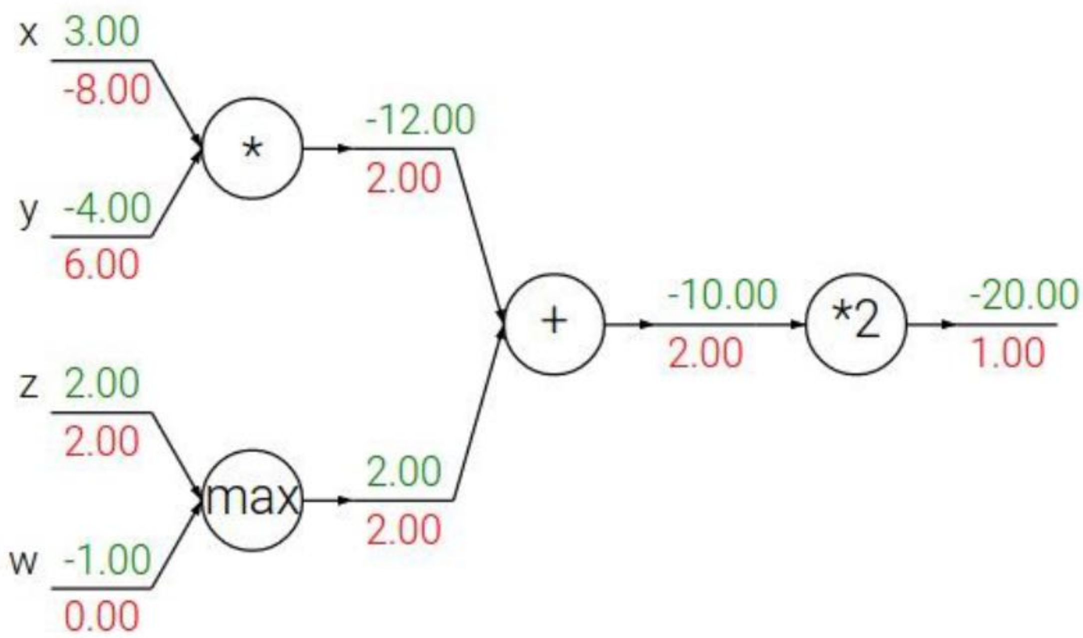
[1.00] x [(1 - 0.73) (0.73)]= 0.2

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Backpropagation

Patterns in backward flow

add gate: gradient distributor



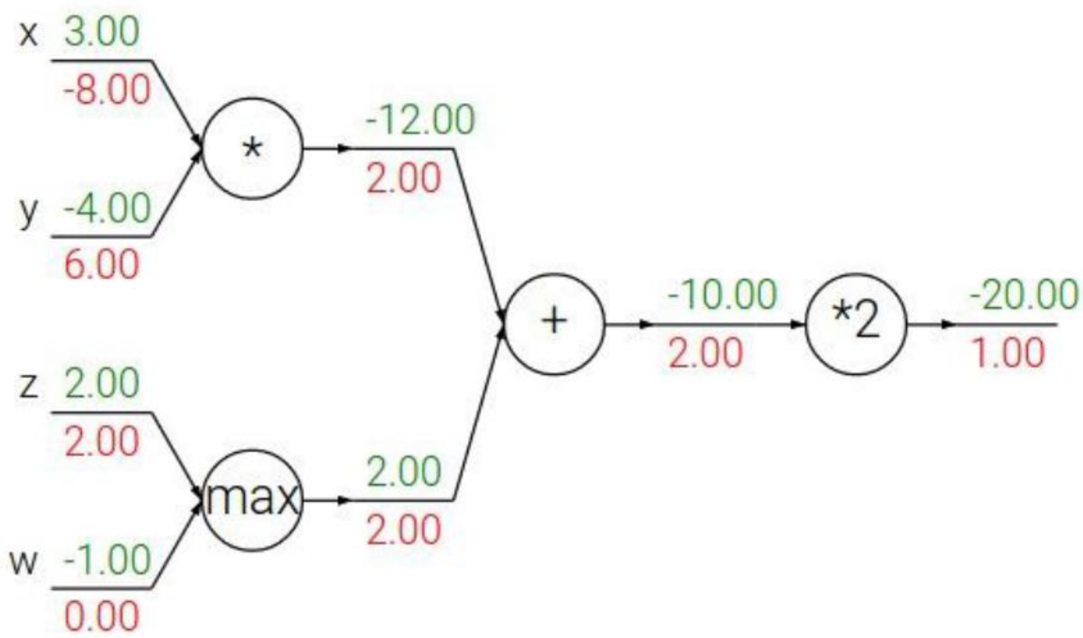
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Backpropagation

Patterns in backward flow

add gate: gradient distributor

Q: What is a **max** gate?



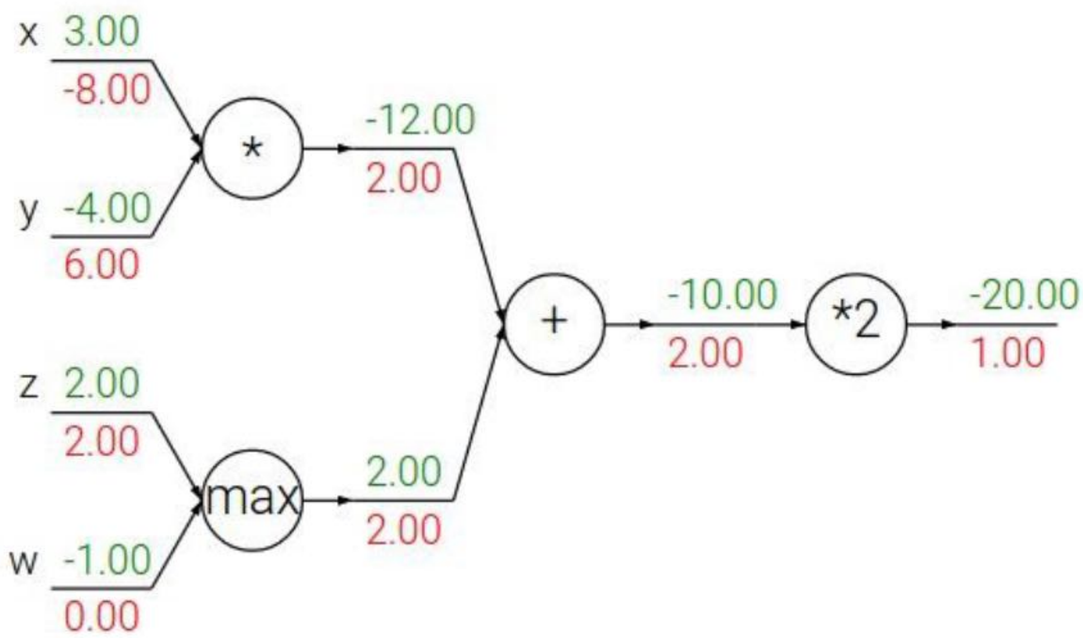
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Backpropagation

Patterns in backward flow

add gate: gradient distributor

max gate: gradient router



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

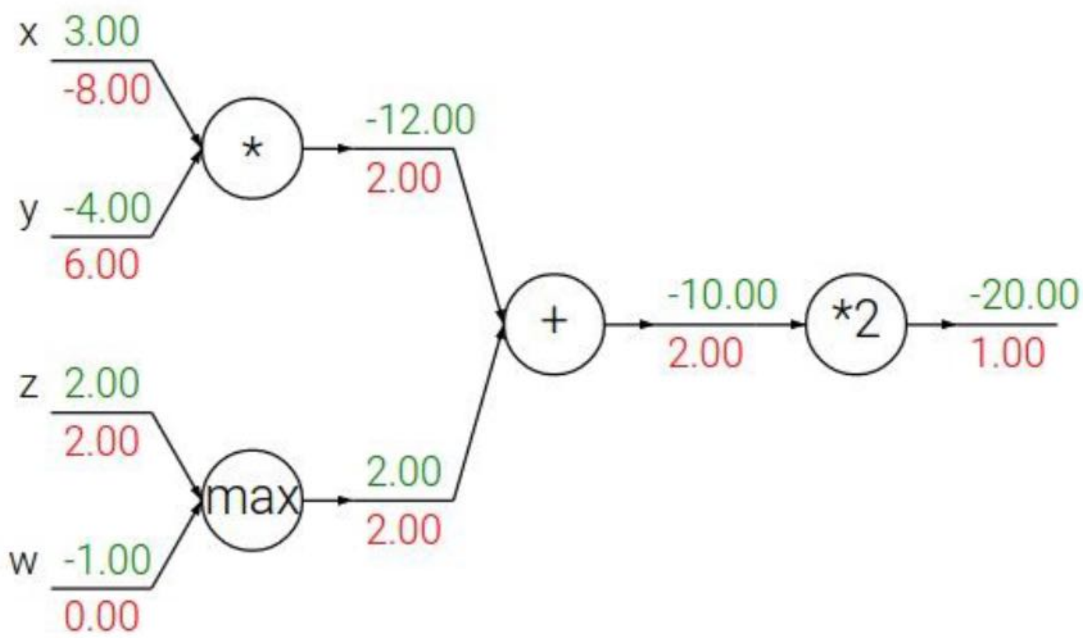
Backpropagation

Patterns in backward flow

add gate: gradient distributor

max gate: gradient router

Q: What is a **mul** gate?



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

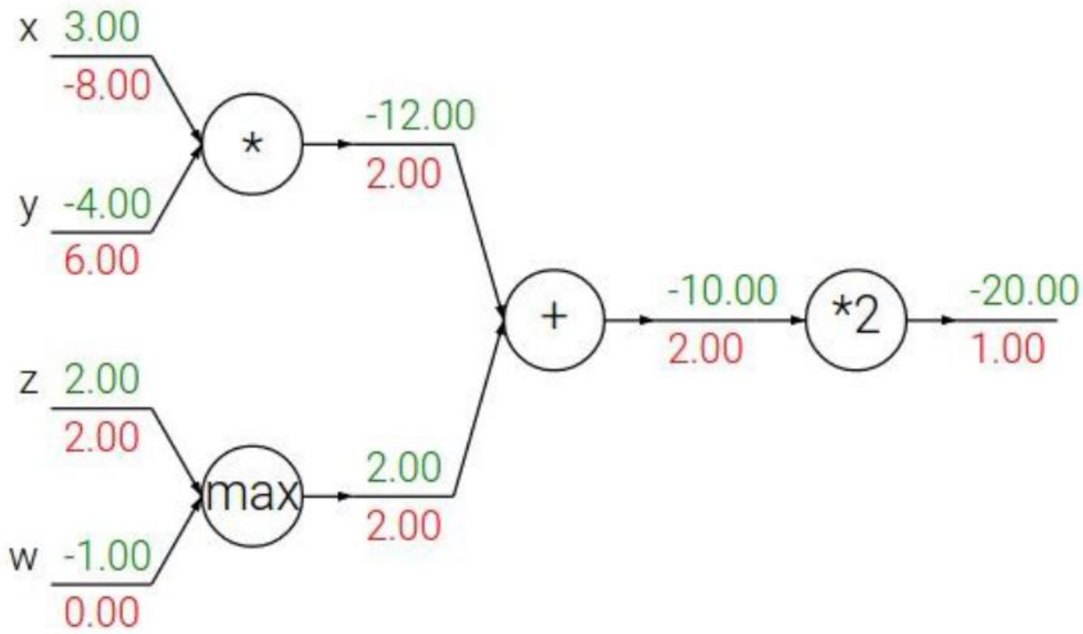
Backpropagation

Patterns in backward flow

add gate: gradient distributor

max gate: gradient router

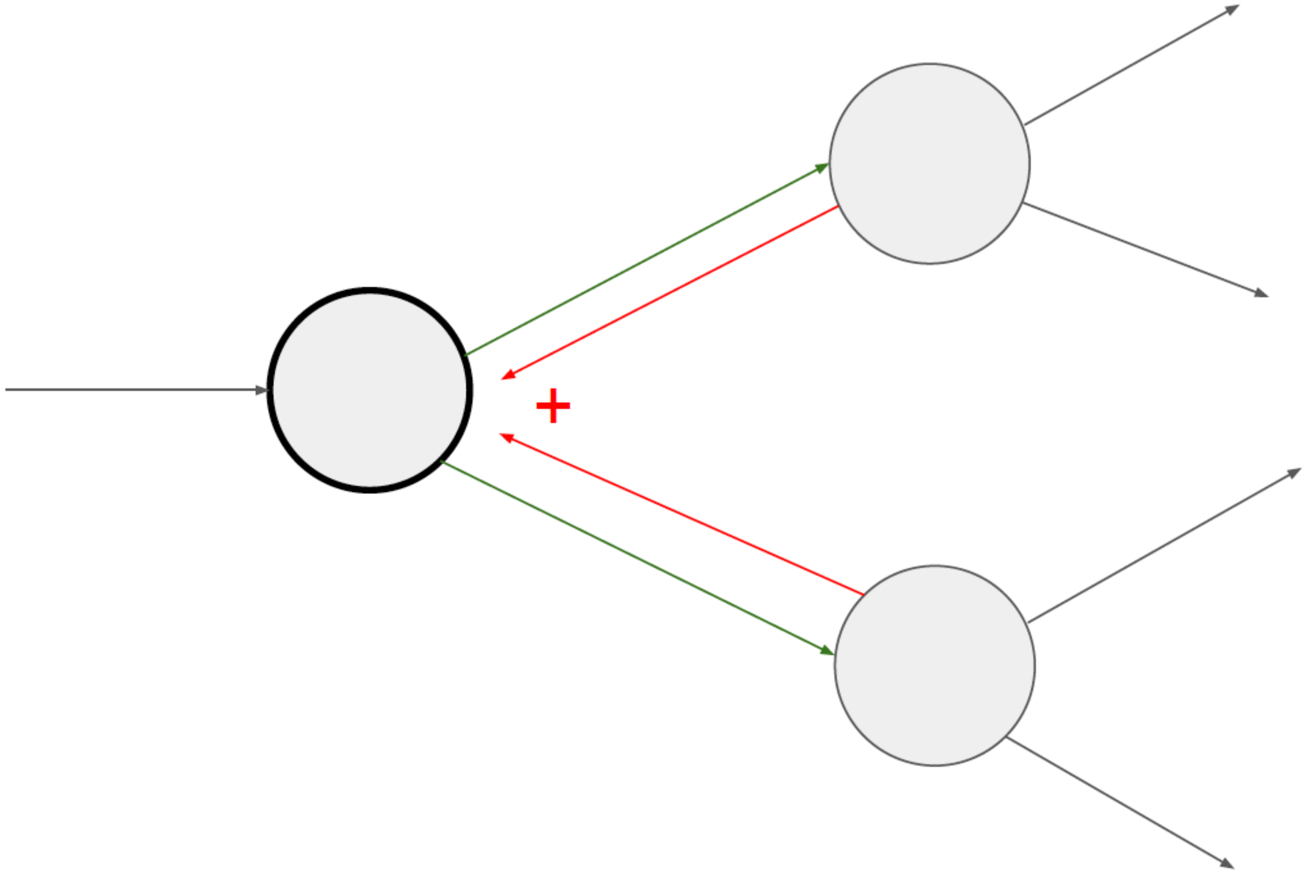
mul gate: gradient switcher



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

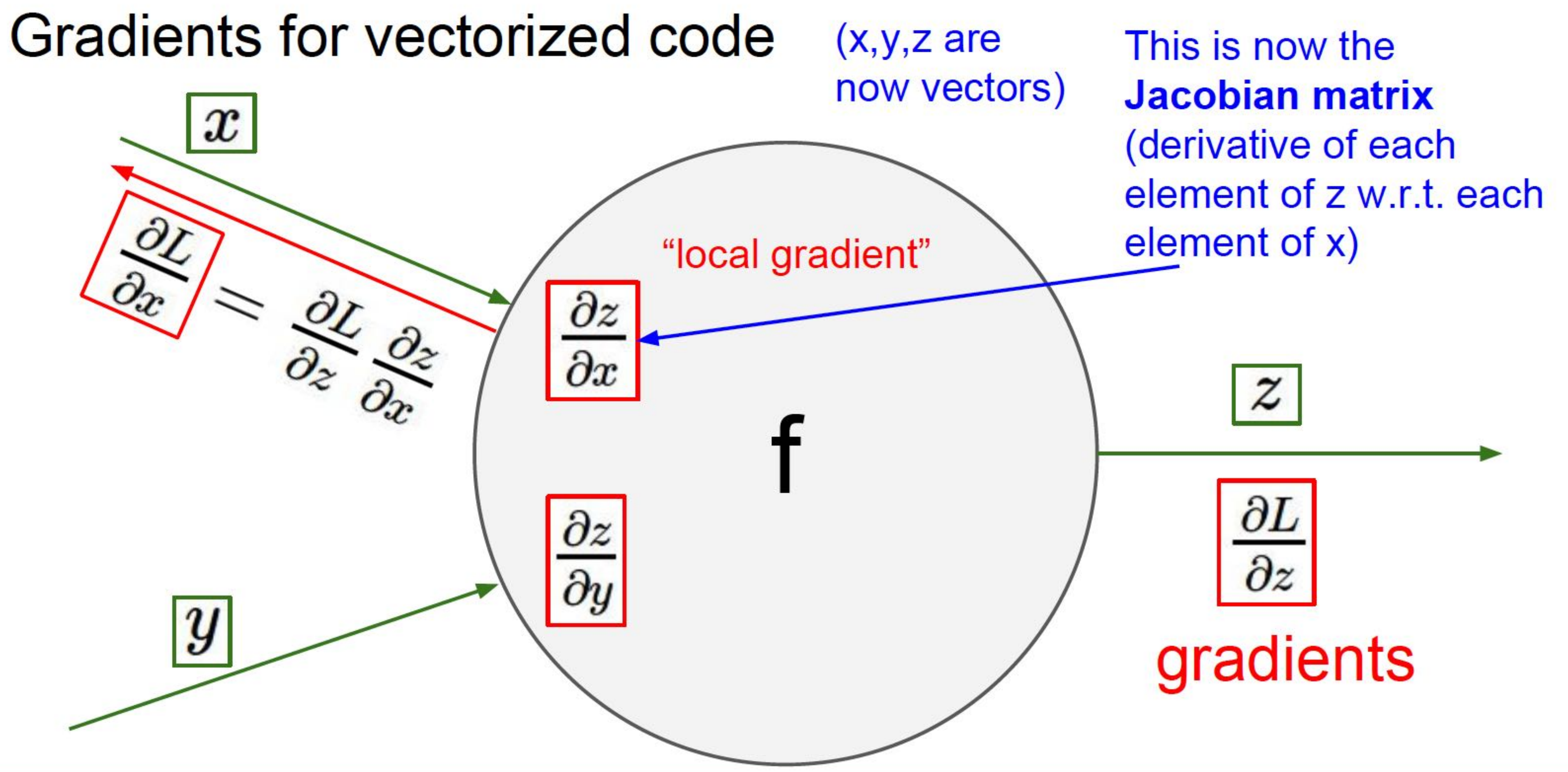
Backpropagation

Gradients add at branches



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

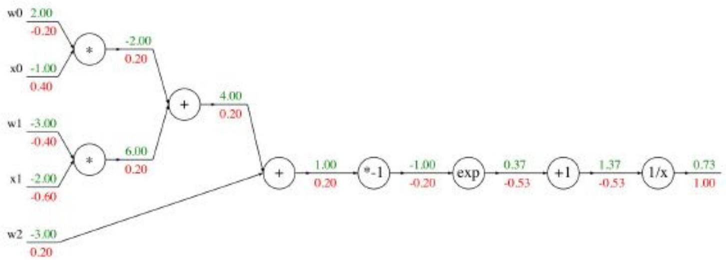
Backpropagation



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Backpropagation

Modularized implementation: forward / backward API



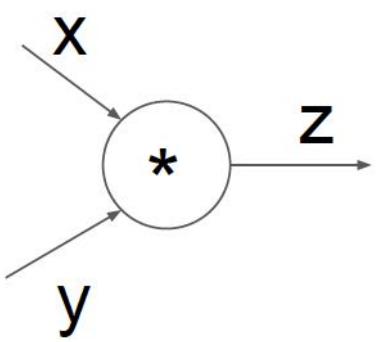
Graph (or Net) object *(rough pseudo code)*

```
class ComputationalGraph(object):
    #...
    def forward(inputs):
        # 1. [pass inputs to input gates...]
        # 2. forward the computational graph:
        for gate in self.graph.nodes_topologically_sorted():
            gate.forward()
        return loss # the final gate in the graph outputs the loss
    def backward():
        for gate in reversed(self.graph.nodes_topologically_sorted()):
            gate.backward() # little piece of backprop (chain rule applied)
        return inputs_gradients
```

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Backpropagation

Modularized implementation: forward / backward API



(x,y,z are scalars)

```
class MultiplyGate(object):
    def forward(x,y):
        z = x*y
        self.x = x # must keep these around!
        self.y = y
        return z
    def backward(dz):
        dx = self.y * dz # [dz/dx * dL/dz]
        dy = self.x * dz # [dz/dy * dL/dz]
        return [dx, dy]
```

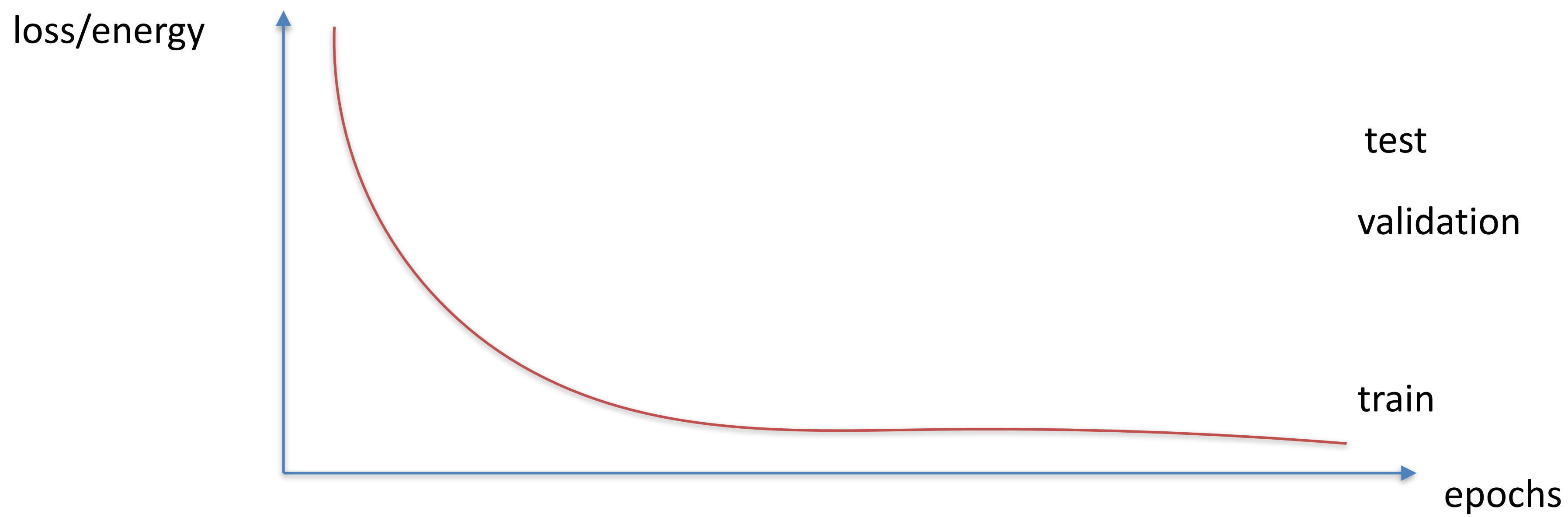
Local gradient   Upstream gradient variable

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Gradient & Training

- Only two things need to be implemented to define new layer:
  - forward pass
  - error back propagation

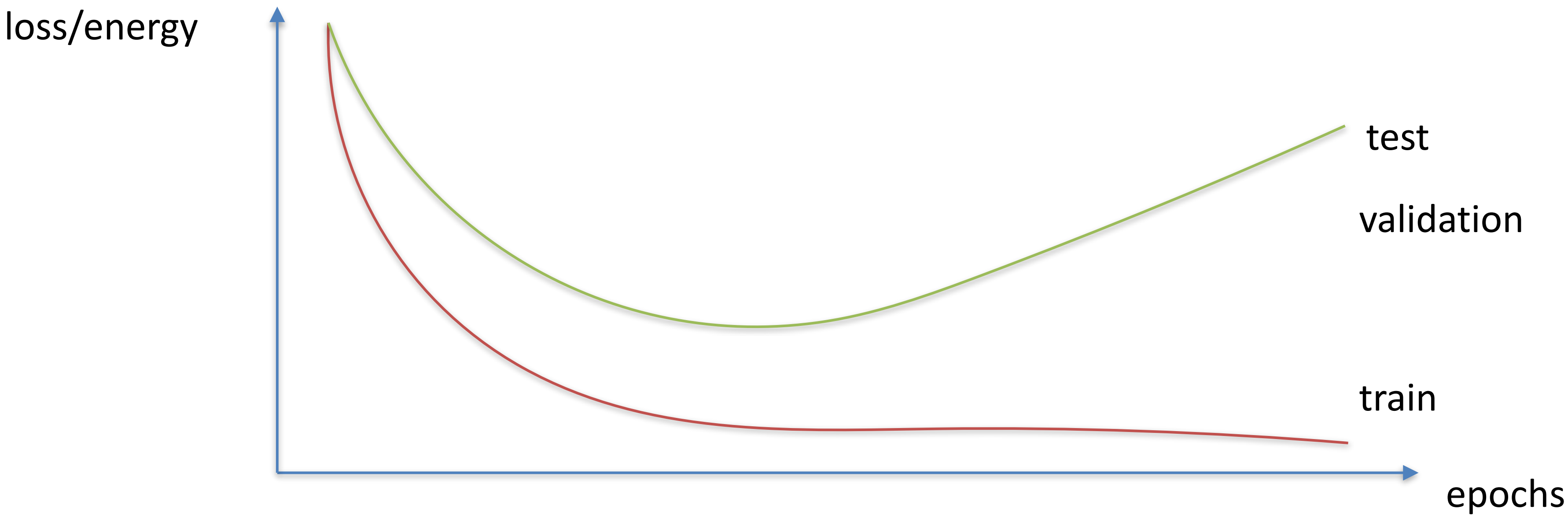
- Watch overfitting



Gradient & Training

- Only two things need to be implemented to define new layer:
  - forward pass
  - error back propagation

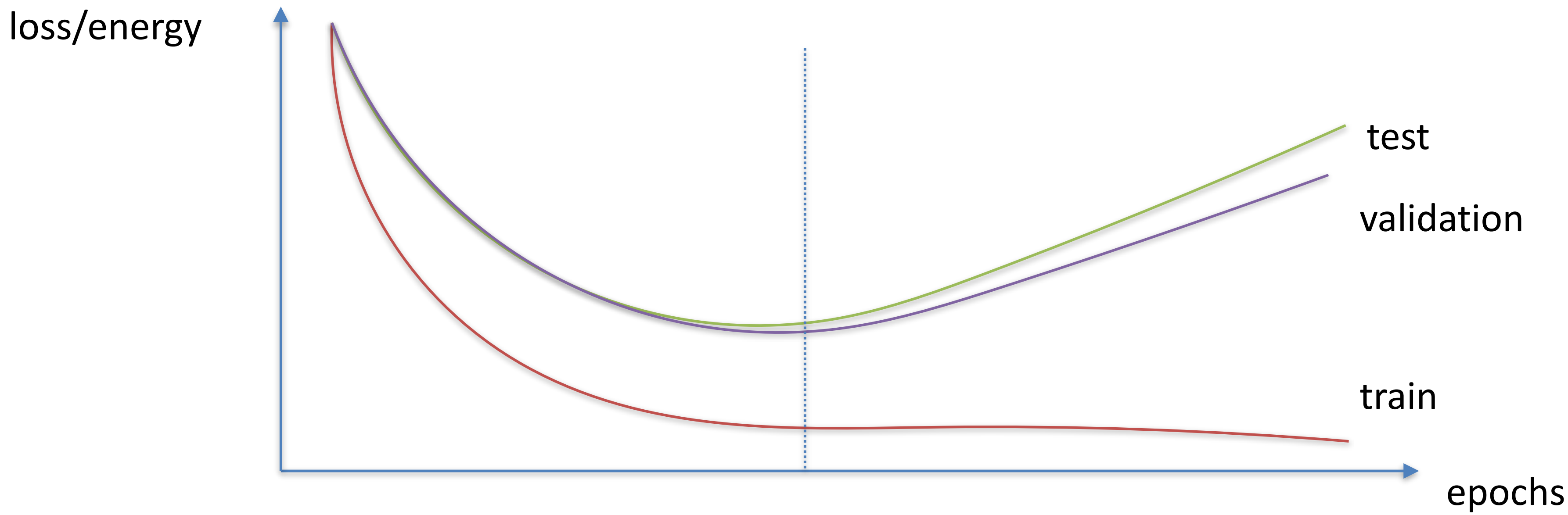
- Watch overfitting



Gradient & Training

- Only two things need to be implemented to define new layer:
  - forward pass
  - error back propagation

- Watch overfitting



Summary so far...

- neural nets will be very large: impractical to write down gradient formula by hand for all parameters
- **backpropagation** = recursive application of the chain rule along a computational graph to compute the gradients of all inputs/parameters/intermediates
- implementations maintain a graph structure, where the nodes implement the **forward**() / **backward**() API
- **forward**: compute result of an operation and save any intermediates needed for gradient computation in memory
- **backward**: apply the chain rule to compute the gradient of the loss function with respect to the inputs

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Overview Today's Lecture

- Backpropagation - Gradient Descent
  - illustrated using computational graphs
  - chain rule - upstream and local gradients
  - modularization simple
- **What is Deep Learning**
  - intuition why deep learning can help
  - integrated learning of features and classifier
- Convolutional Neural Networks (CNNs)
  - one of the (few) highly successful NNs

Simple Neural Networks

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Simple Neural Networks

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

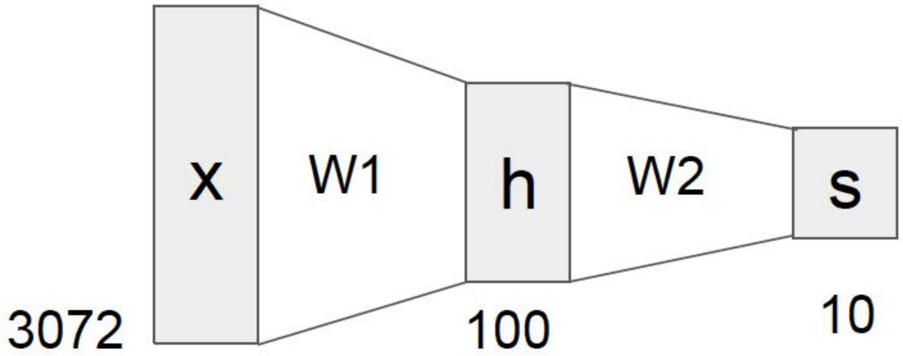
(**Now**) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Simple Neural Networks

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

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

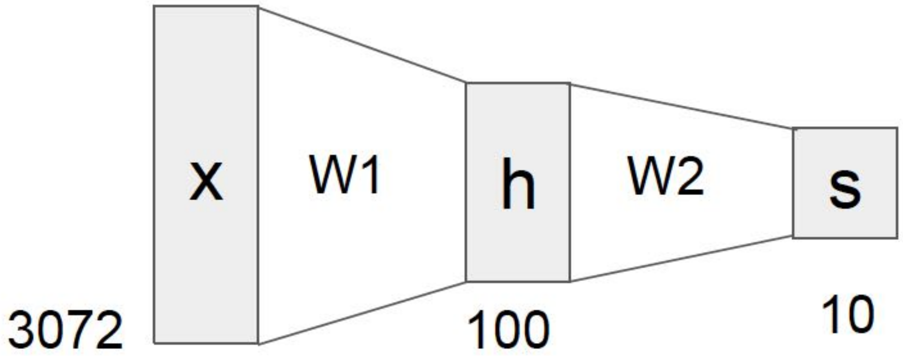


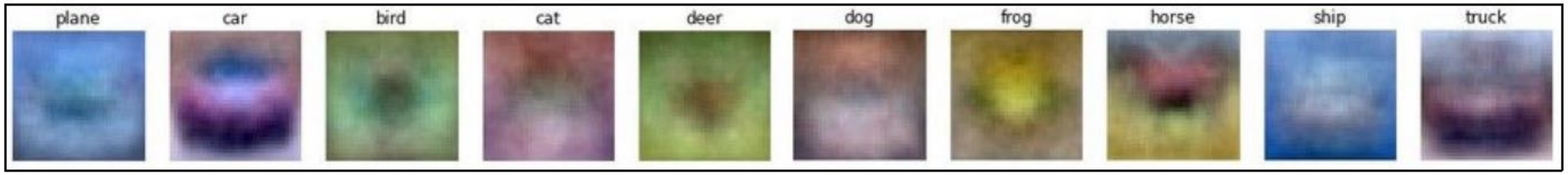
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Simple Neural Networks

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

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





slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Simple Neural Networks

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

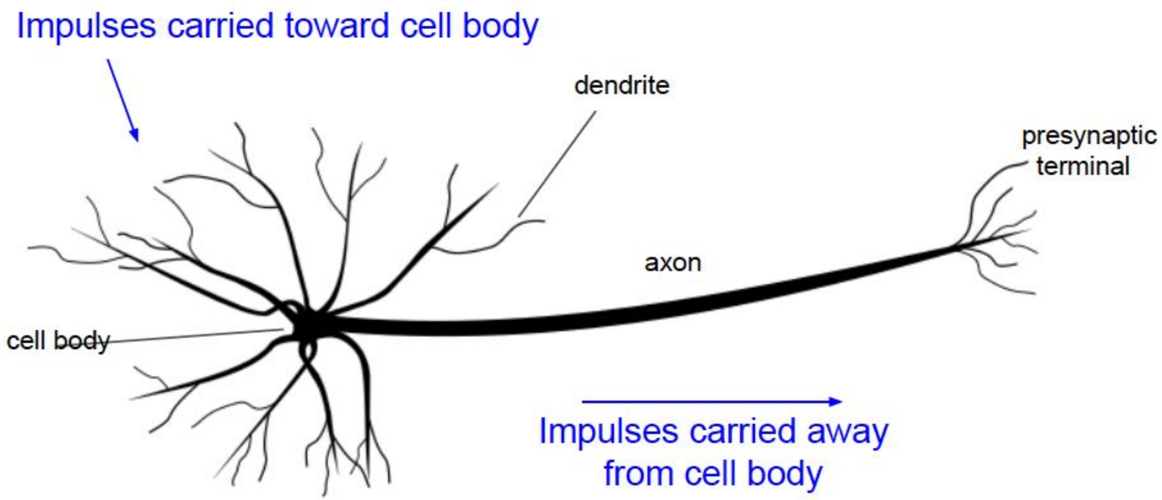
(**Now**) 2-layer Neural Network $f = W_2 \max(0, W_1 x)$

or 3-layer Neural Network

$$f = W_3 \max(0, W_2 \max(0, W_1 x))$$

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

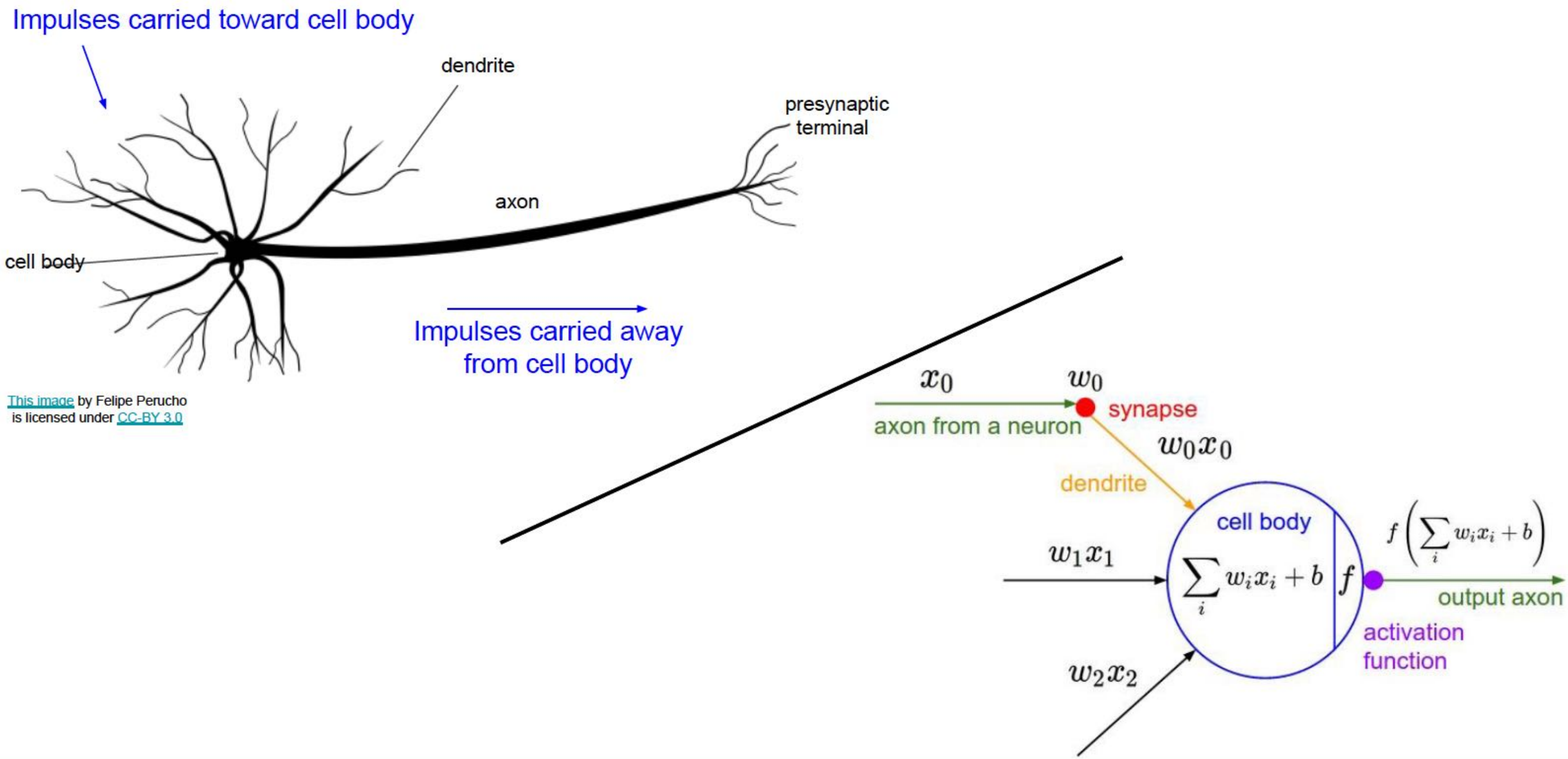
Some Inspiration from the (Human) Brain



This image by Felipe Perucho is licensed under CC-BY 3.0

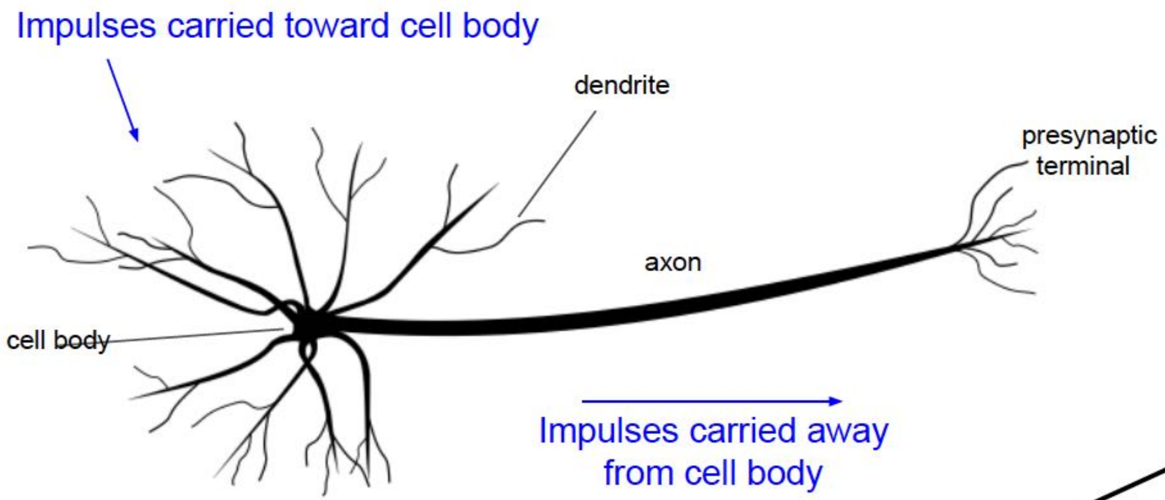
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Some Inspiration from the (Human) Brain

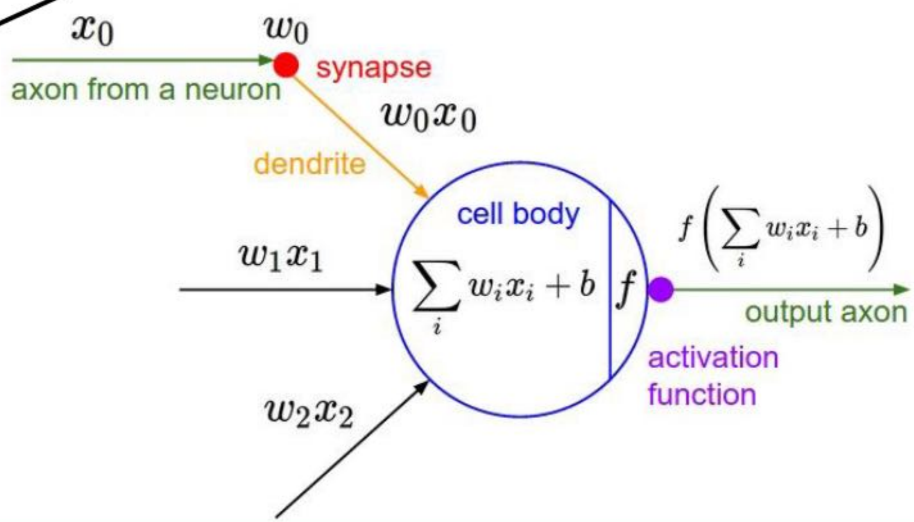


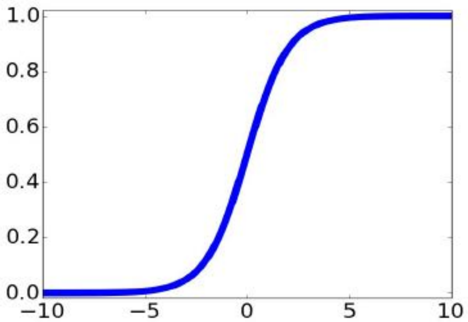
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Some Inspiration from the (Human) Brain



This image by Felipe Perucho is licensed under CC-BY 3.0





sigmoid activation function

$$\frac{1}{1+e^{-x}}$$

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Some Inspiration from the (Human) Brain

Be very careful with your brain analogies!

Biological Neurons:

- Many different types
- Dendrites can perform complex non-linear computations
- Synapses are not a single weight but a complex non-linear dynamical system
- Rate code may not be adequate

[Dendritic Computation. London and Hausser]

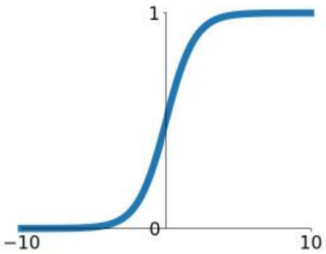
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Activation Functions

Activation functions

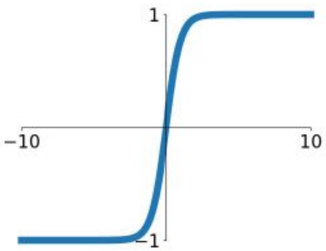
Sigmoid

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



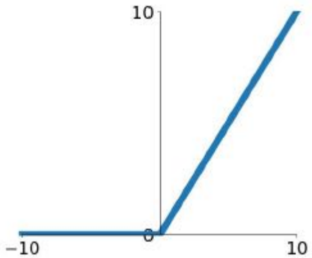
tanh

$\tanh(x)$



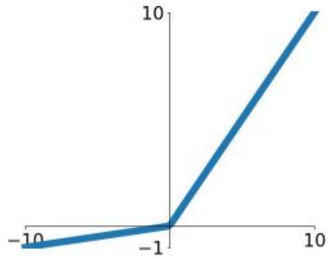
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

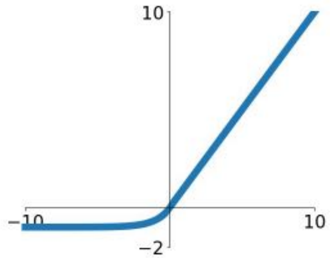


Maxout

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

ELU

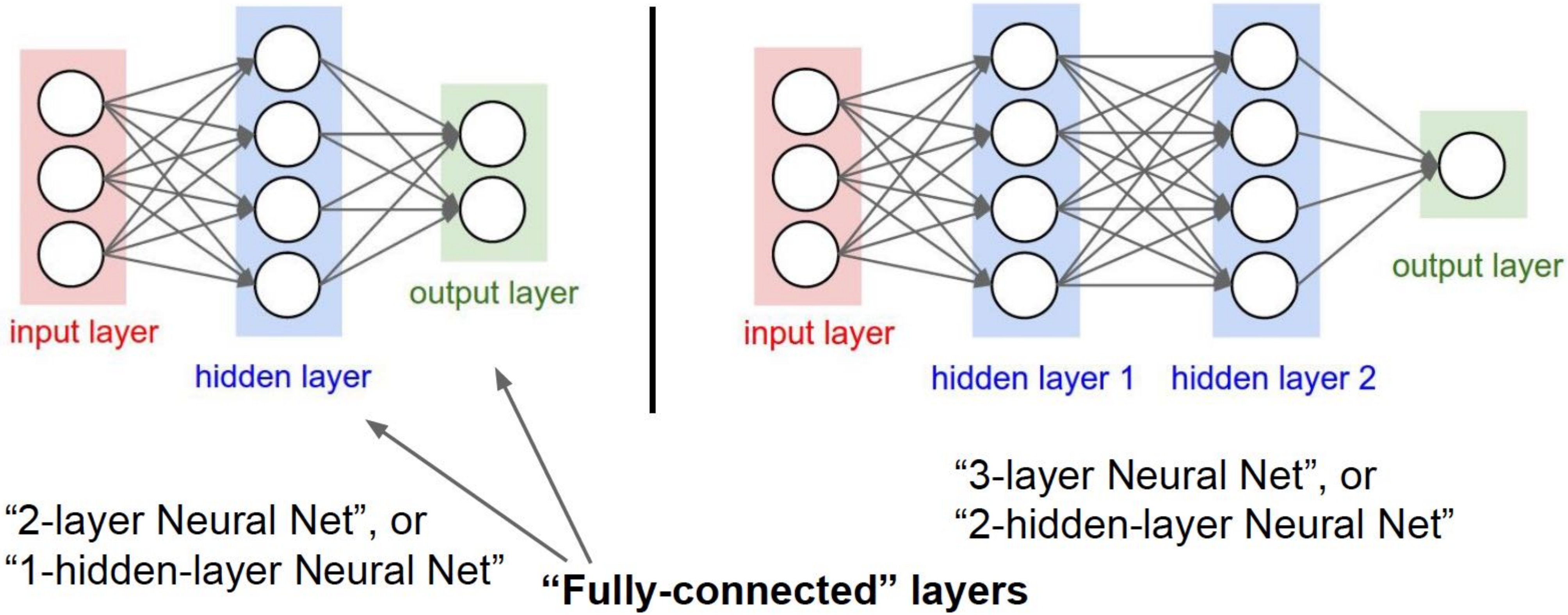
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

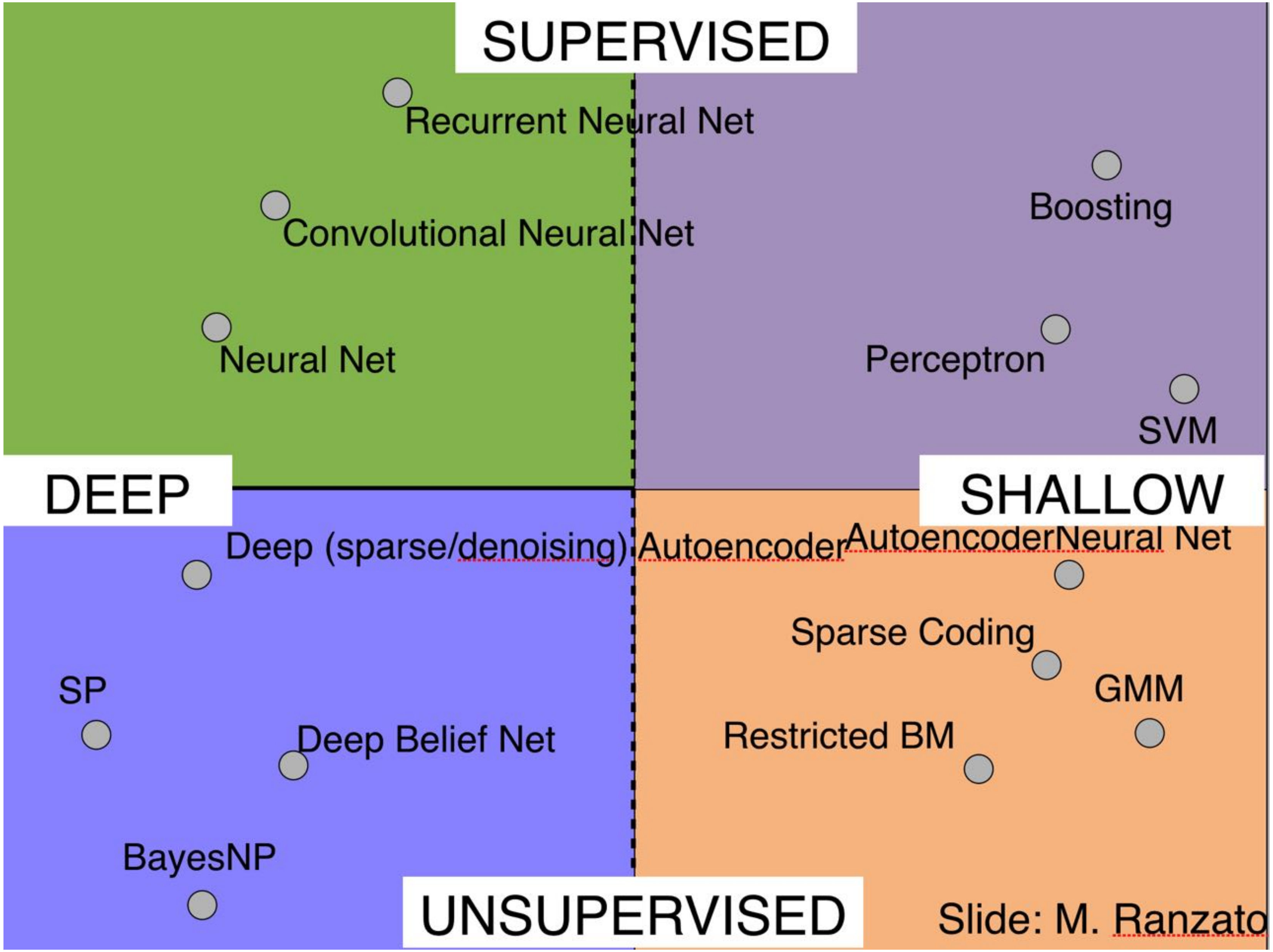
Neural Networks - Classic Architectures

- (Multi-Layer) Perceptron



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Overview of Neural Network Architectures

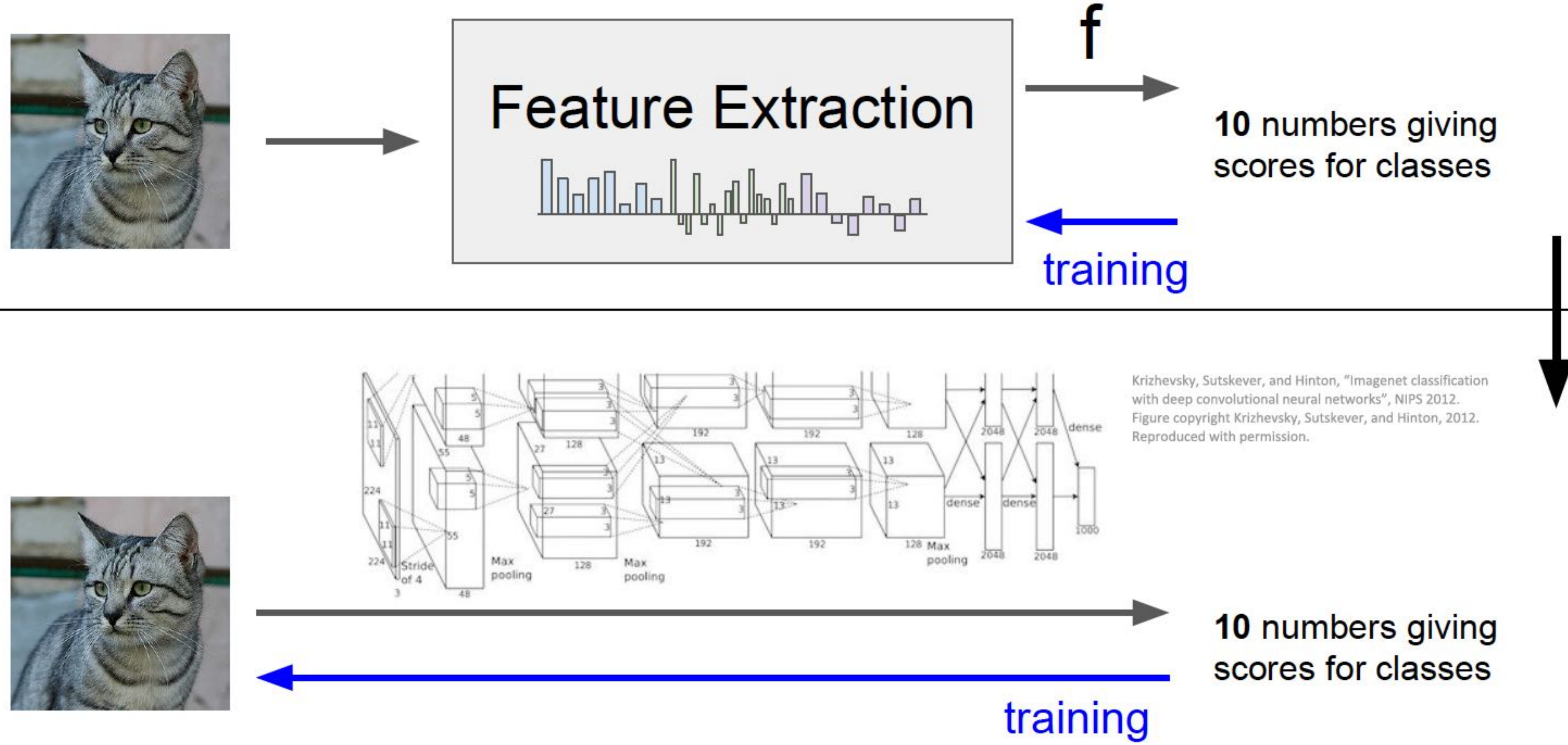


Deep Learning Ingredients

- Deep Learning is based on
  - Availability of large datasets
  - Massive parallel compute power
  - Advances in machine learning over the years

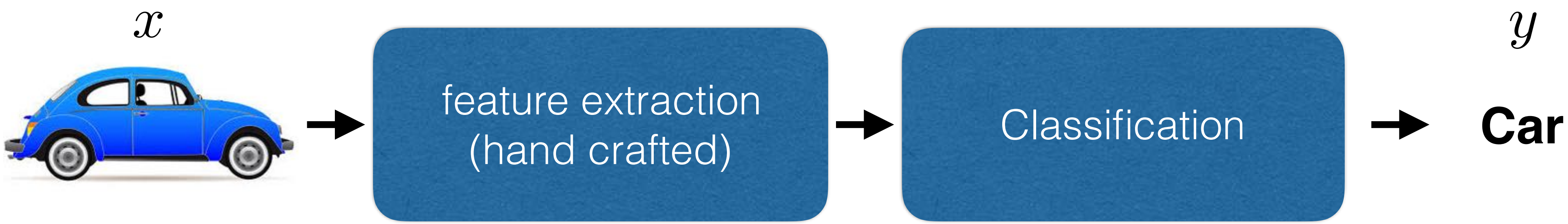
- Strong improvements due to
  - Internet (availability of large-scale data)
  - GPUs (availability of parallel compute power)
  - Deep / hierarchical models with end-to-end learning

Image Features vs. Deep Learning



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

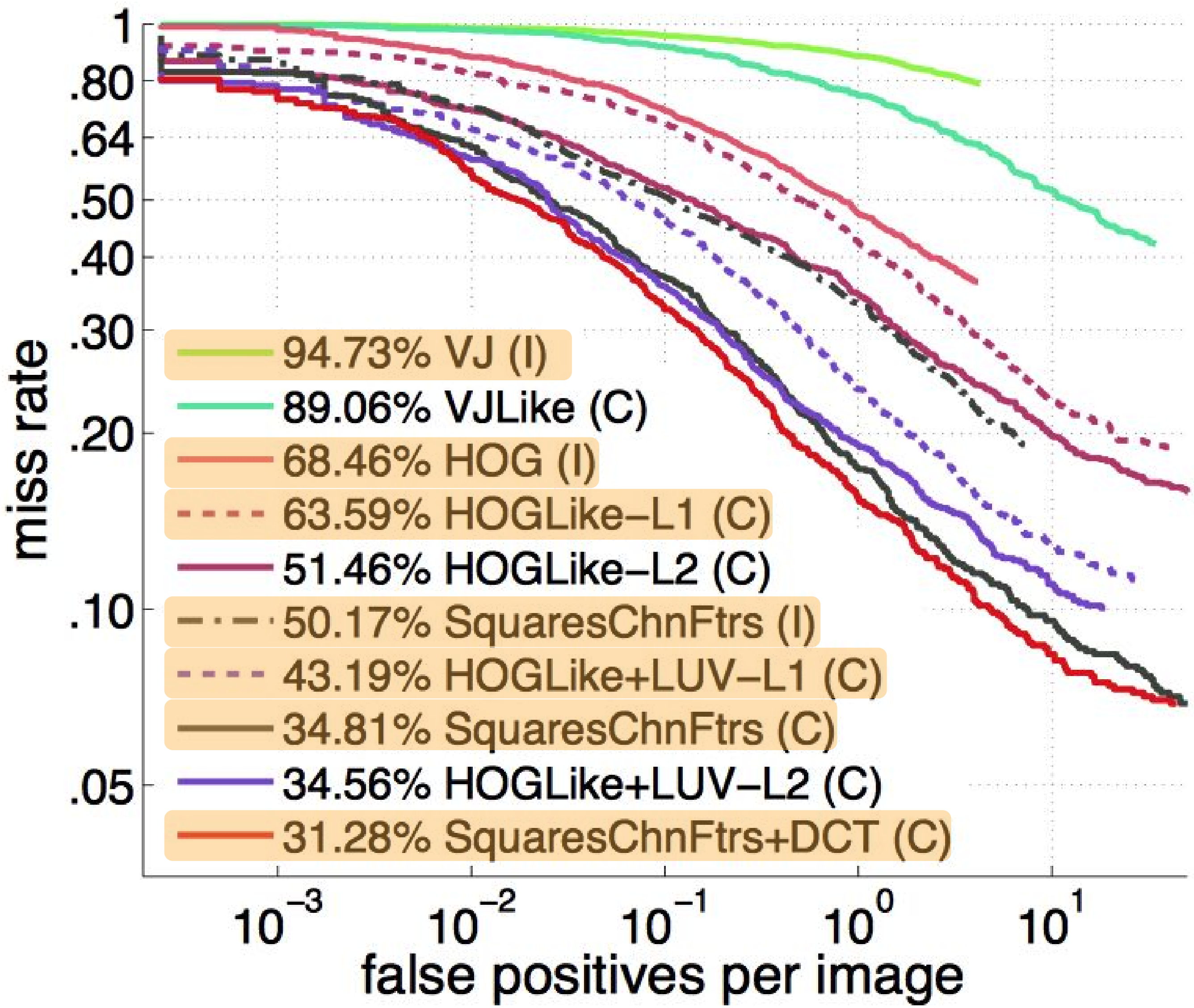
Traditional Approach



- Feature extraction
  - often hand crafted and fixed
  - might be too general (not task-specific enough)
  - might be too specific (does not generalize to other tasks)
- How to achieve best classification performance
  - more complex classifier (e.g. multi-feature, non-linear)?
  - how specialized for the task?

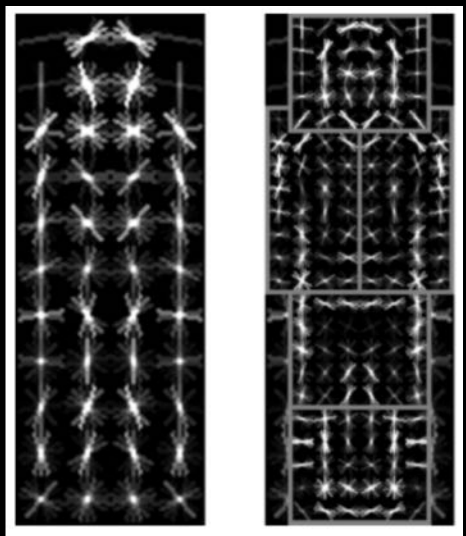
Features Alone Can Explain (Almost) 10 Years of Progress

[Benenson,Omran,Hosang,Schiele@ECCV workshop'14]

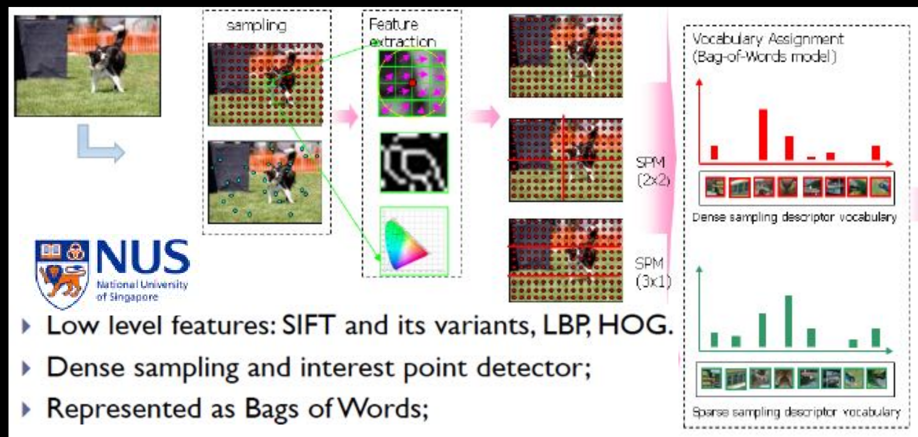


Motivation

- Features are key to recent progress in recognition
- Multitude of hand-designed features currently in use
  - SIFT, HOG, LBP, MSER, Color-SIFT………….
- Where next? Better classifiers? Or keep building more features?



Felzenszwalb, Girshick, McAllester and Ramanan, PAMI 2007



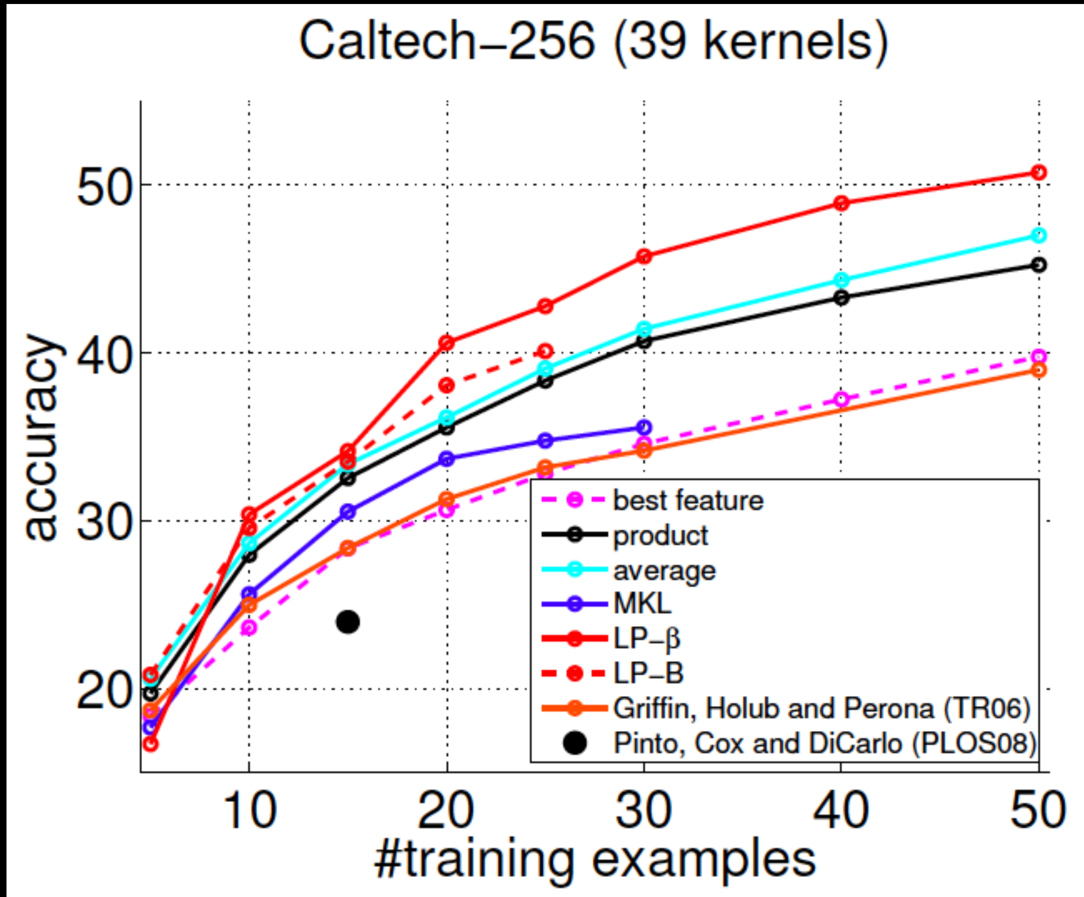
Yan & Huang
(Winner of PASCAL 2010 classification competition)

slide credit: Rob Fergus

Hand-Crafted Features

- LP-β Multiple Kernel Learning (MKL)
  - Gehler and Nowozin, On Feature Combination for Multiclass Object Classification, ICCV'09
- 39 different kernels
  - PHOG, SIFT, V1S+, Region Cov. Etc.
- MKL only gets few % gain over averaging features

→ Features are doing the work



slide credit: Rob Fergus

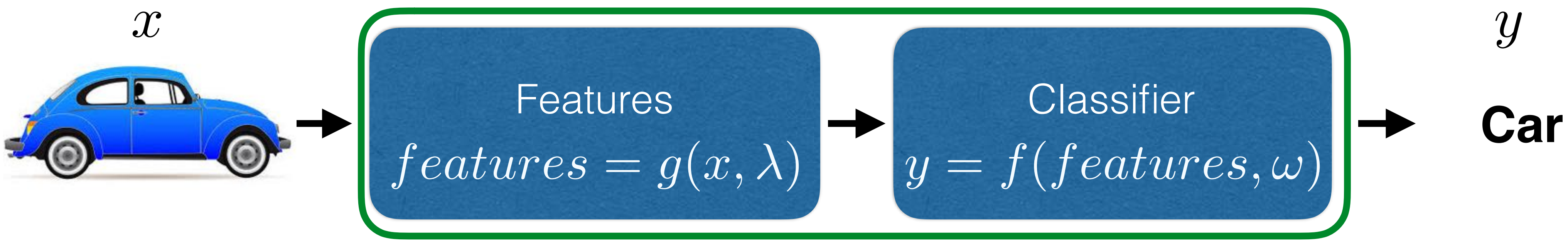
Deep Learning: Trainable features

x

→ **Features**: $features = g(x, \lambda)$ → **Classifier**: $y = f(features, \omega)$ → y **Car**

- Parameterized feature extraction
- Features should be
  - efficient to compute
  - efficient to train (differentiable)

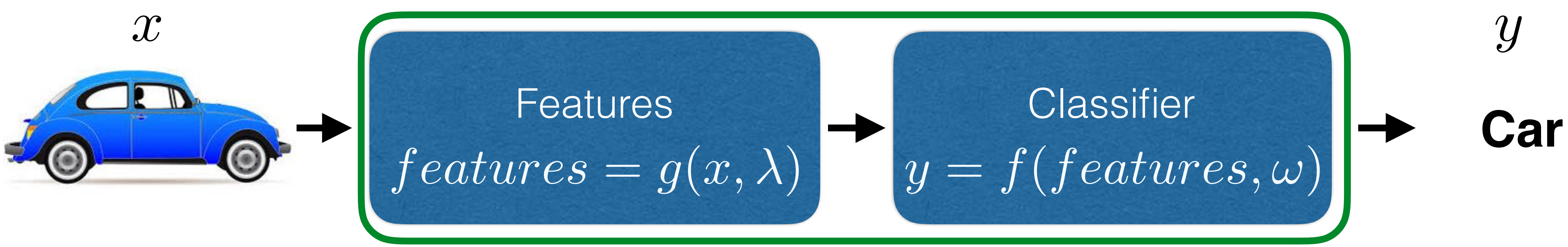
Deep Learning: Joint Training of all Parameters



"End-to-End" System

- Parameterized feature extraction
- Features should be
  - efficient to compute
  - efficient to train (differentiable)
- **Joint** training of **feature** extraction and **classification**
- Feature extraction and classification merge into one pipeline

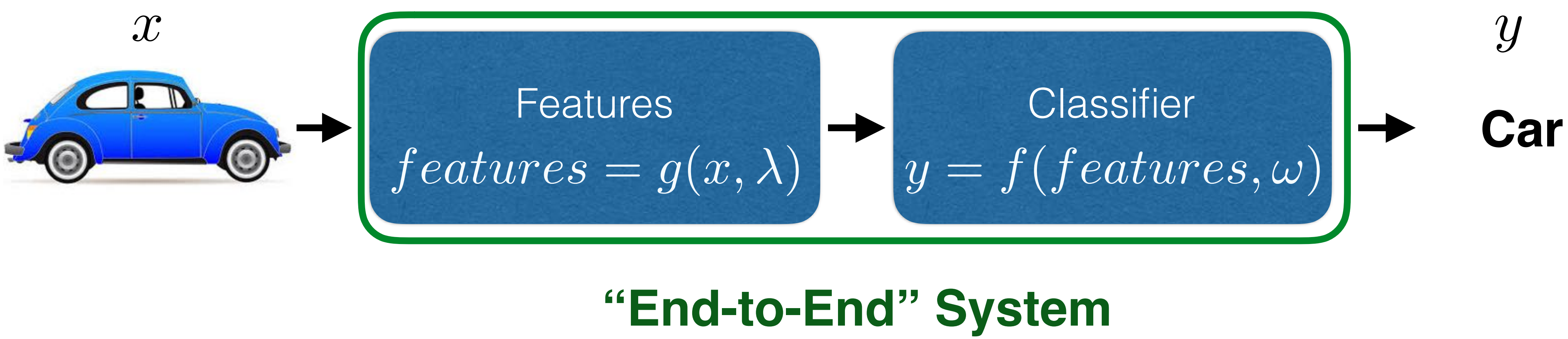
Deep Learning: Joint Training of all Parameters



“End-to-End” System

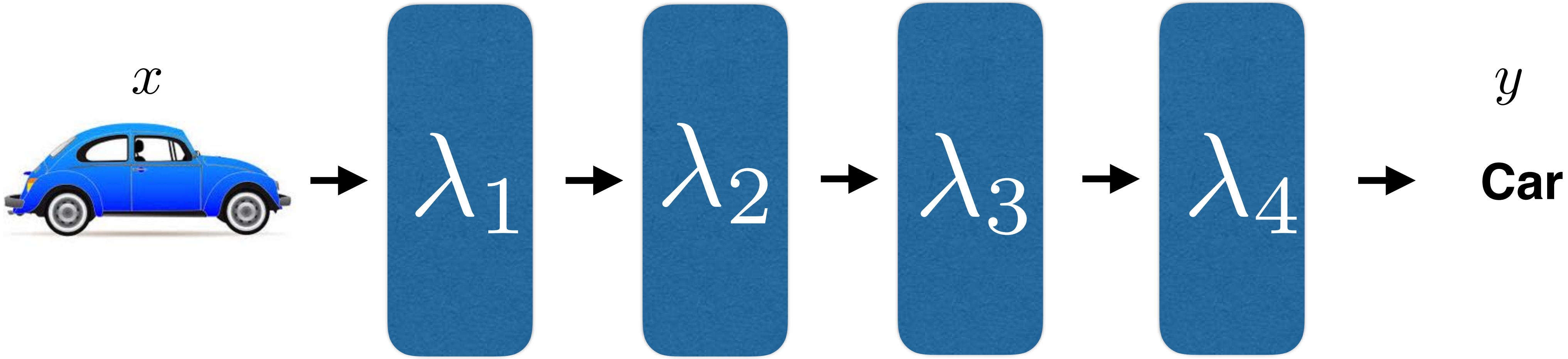
- All parts are adaptive
- No differentiation between feature extraction and classification
- Non linear transformation from input to desired output

Deep Learning: Complex Functions by Composition



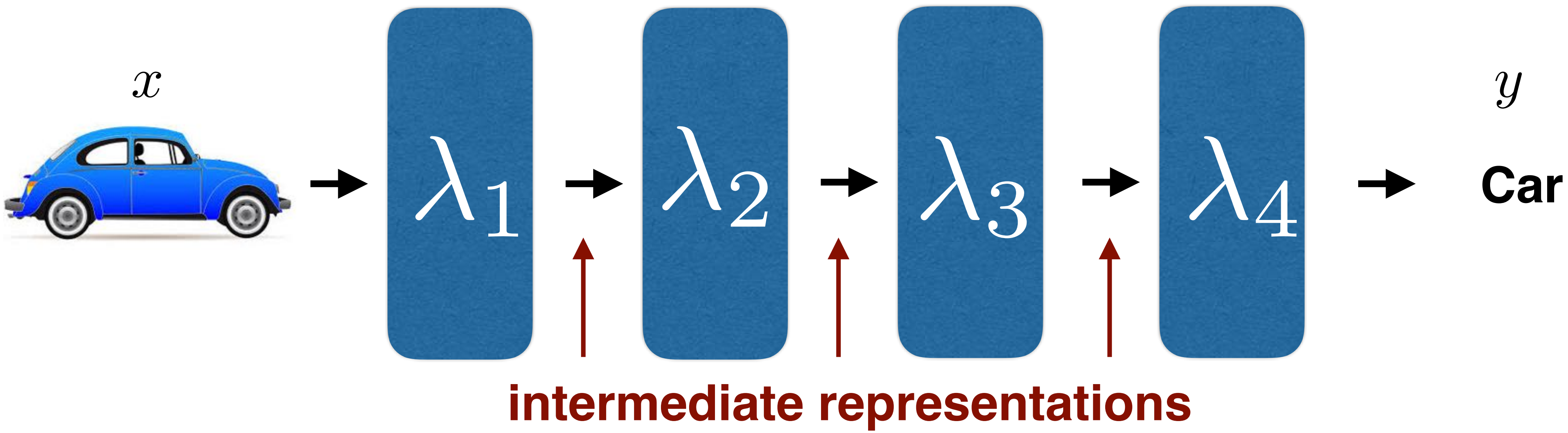
- How can we build such systems?
- What is the parameterization (hypothesis)?
- Composition of simple building blocks can lead to complex systems (e.g. neurons - brain)

Deep Learning: Complex Functions by Composition



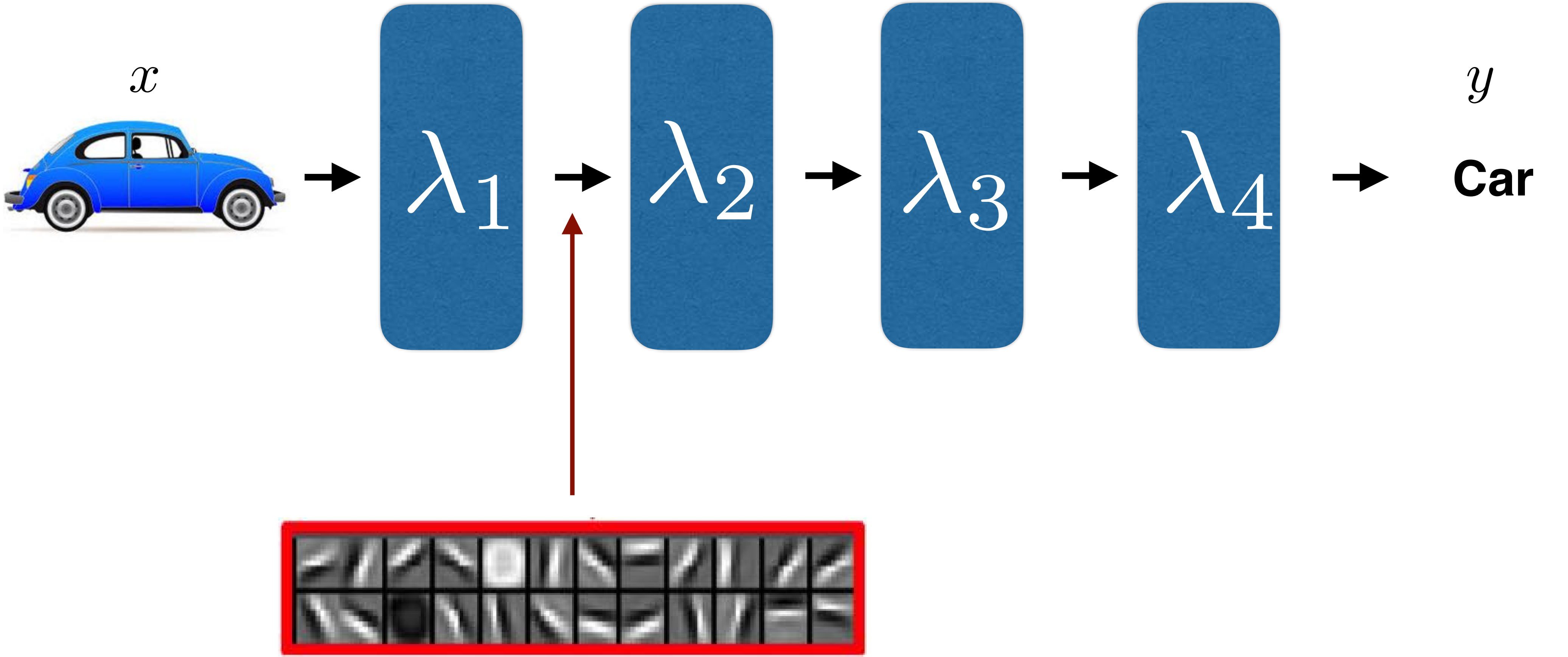
- How can we build such systems?
- What is the parameterization (hypothesis)?
- Composition of simple building blocks can lead to complex systems (e.g. neurons - brain) each block has trainable parameters
- Each block has trainable parameters λ_i

Deep Learning: Complex Functions by Composition



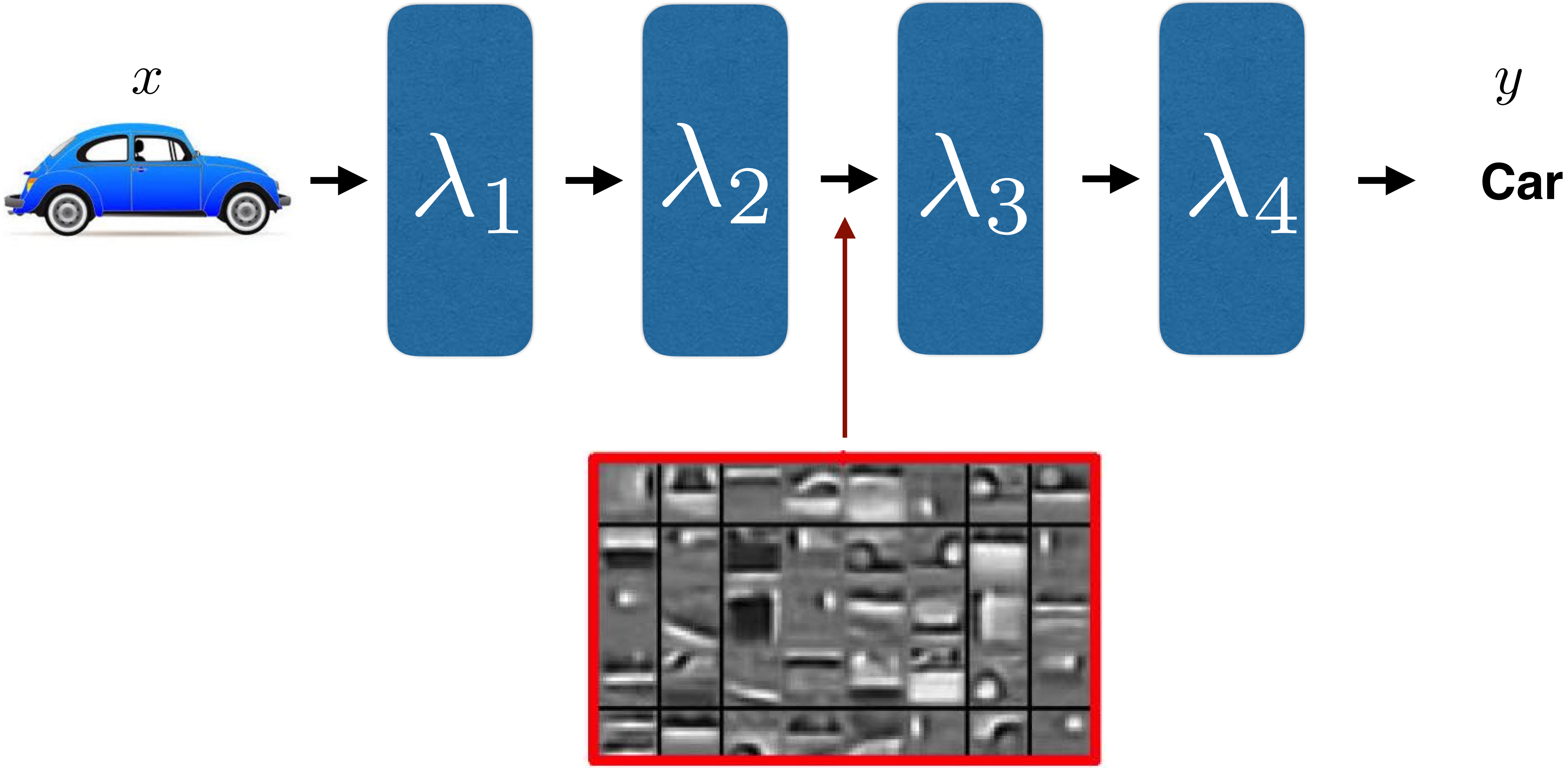
- How can we build such systems?
- What is the parameterization (hypothesis)?
- Composition of simple building blocks can lead to complex systems (e.g. neurons - brain) each block has trainable parameters
- Each block has trainable parameters λ_i

Deep Learning: Complex Functions by Composition



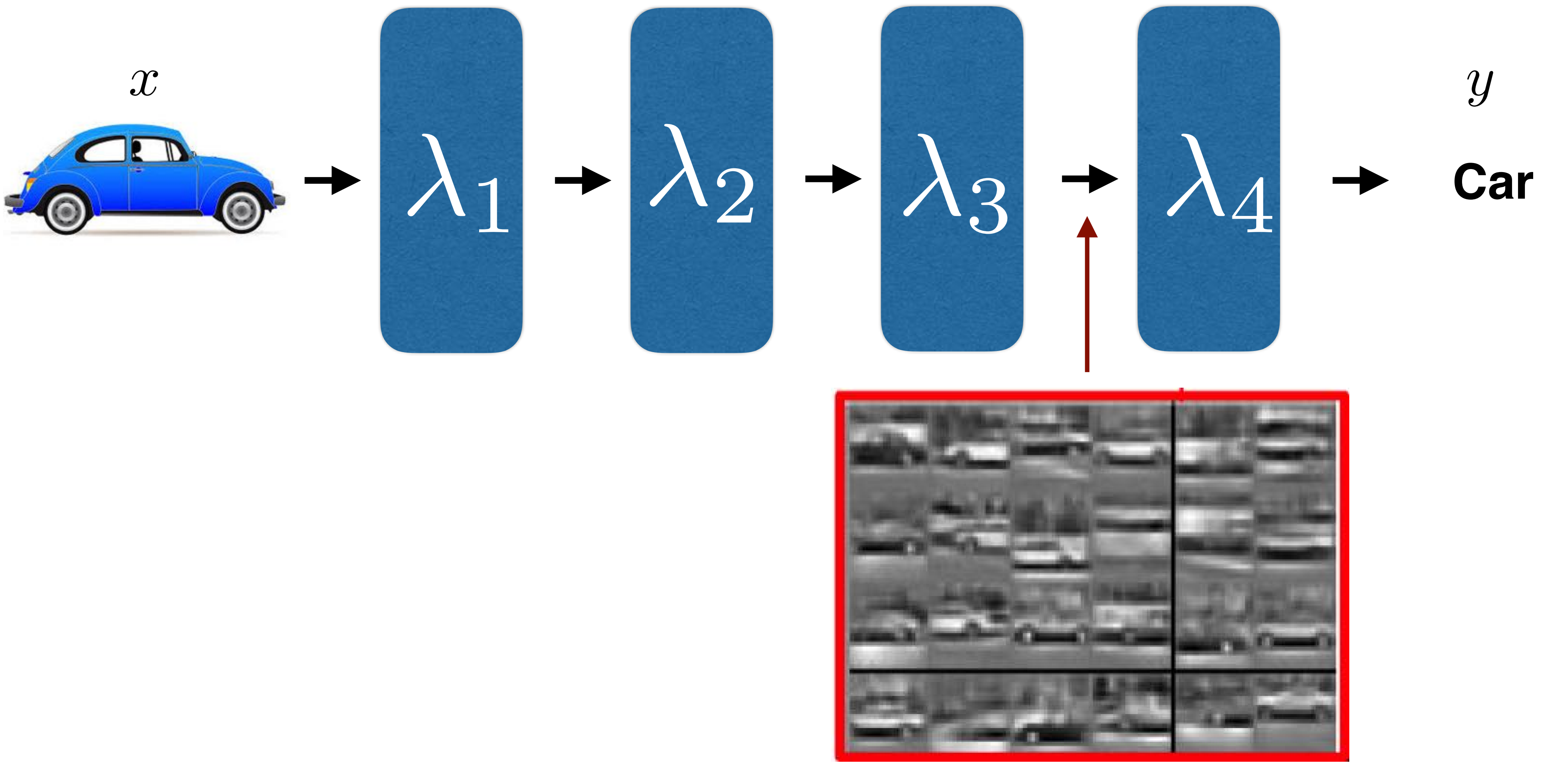
Lee et al. “Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations”

Deep Learning: Complex Functions by Composition



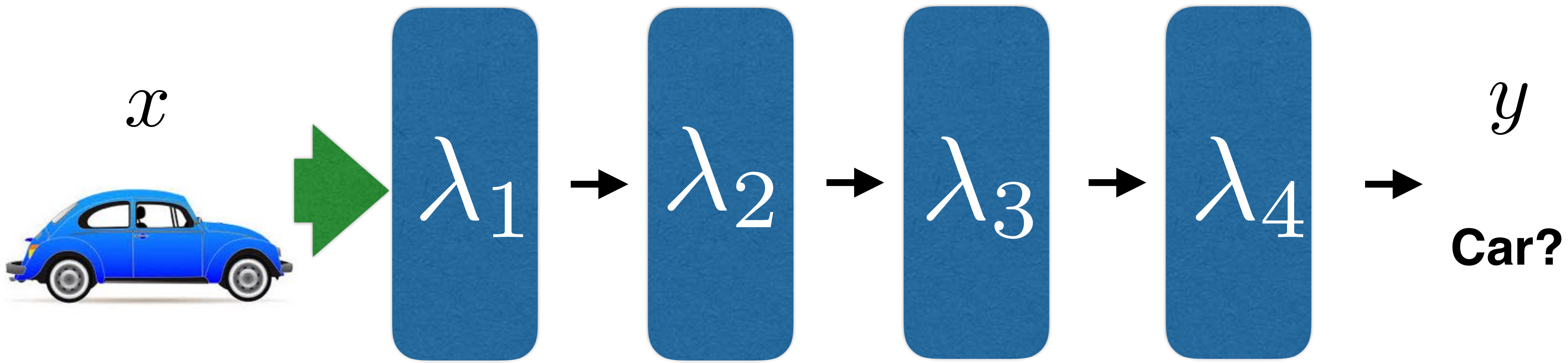
Lee et al. “Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations”

Deep Learning: Complex Functions by Composition



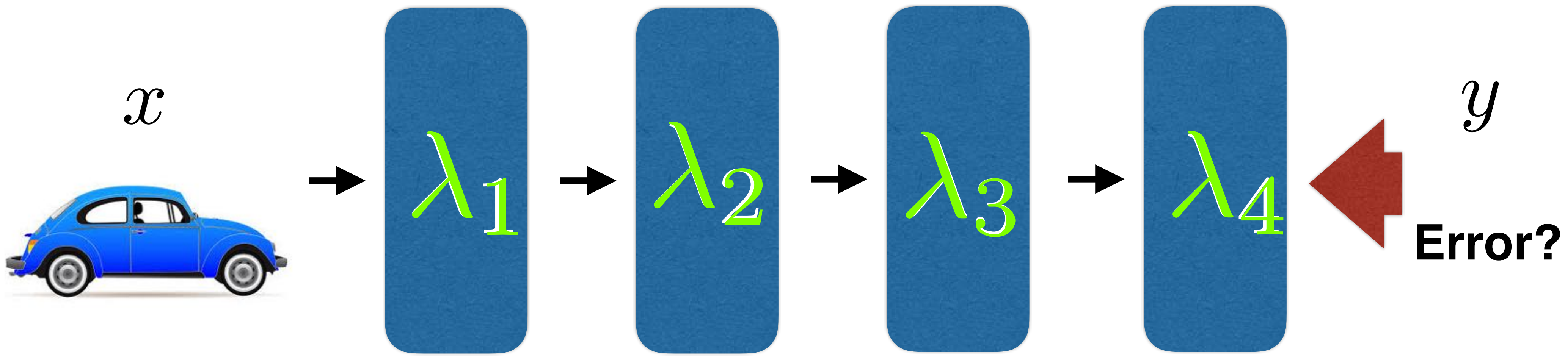
Lee et al. “Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations”

Training: Overview



- Setting
  - generate output y for input x (forward pass)

Training: Overview



- Setting
  - generate output y for input x (forward pass)
  - when there is an error, propagate error backwards to update weights (error back propagation)

Summary of Main Ideas in Deep Learning

1. **Learning of feature extraction** (across many layers)

2. **Efficient** and **trainable** systems by **differentiable** building blocks

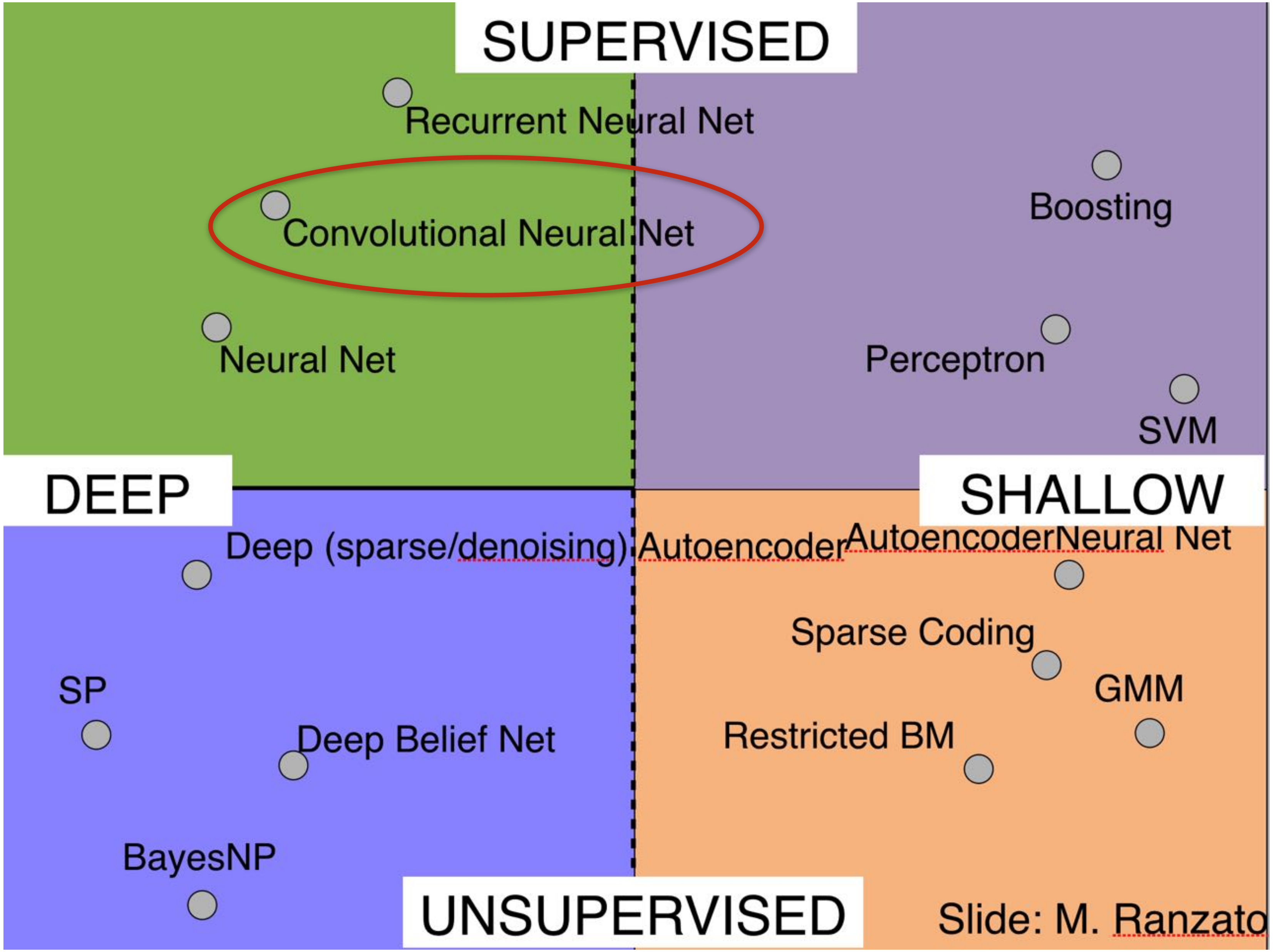
3. Composition of deep architectures via **non-linear modules**

4. “**End-to-End**” training: no differentiation between feature extraction and classification

Overview Today's Lecture

- Backpropagation - Gradient Descent
  - illustrated using computational graphs
  - chain rule - upstream and local gradients
  - modularization simple
- What is Deep Learning
  - intuition why deep learning can help
  - integrated learning of features and classifier
- **Convolutional Neural Networks (CNNs)**
  - one of the (few) highly successful NNs

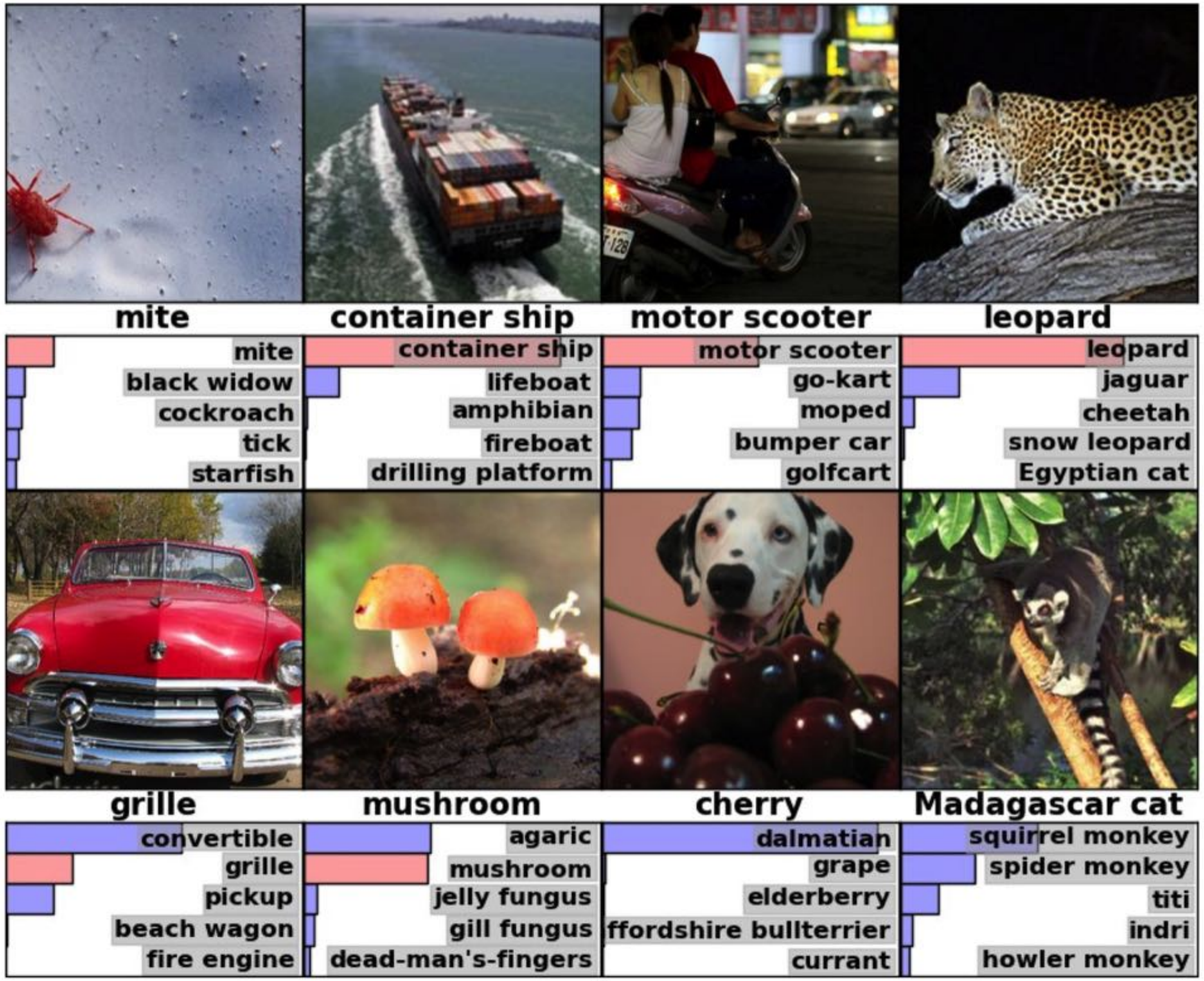
Overview of Deep Learning



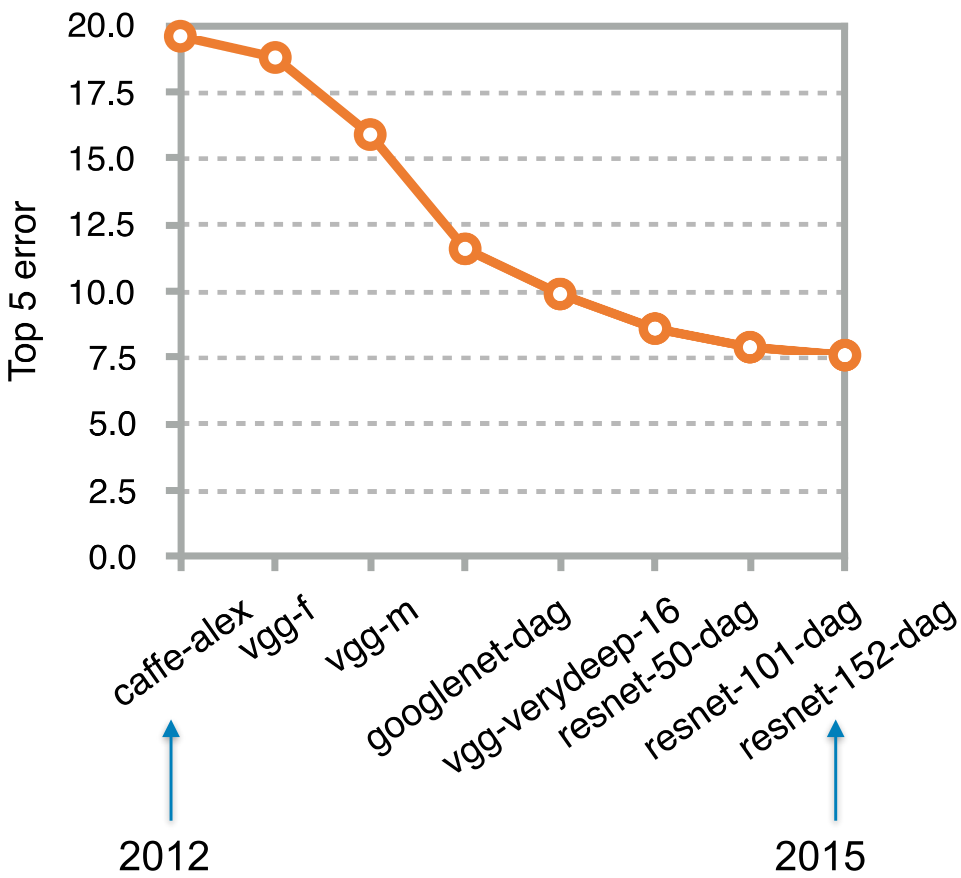
Computer Vision: Image Classification

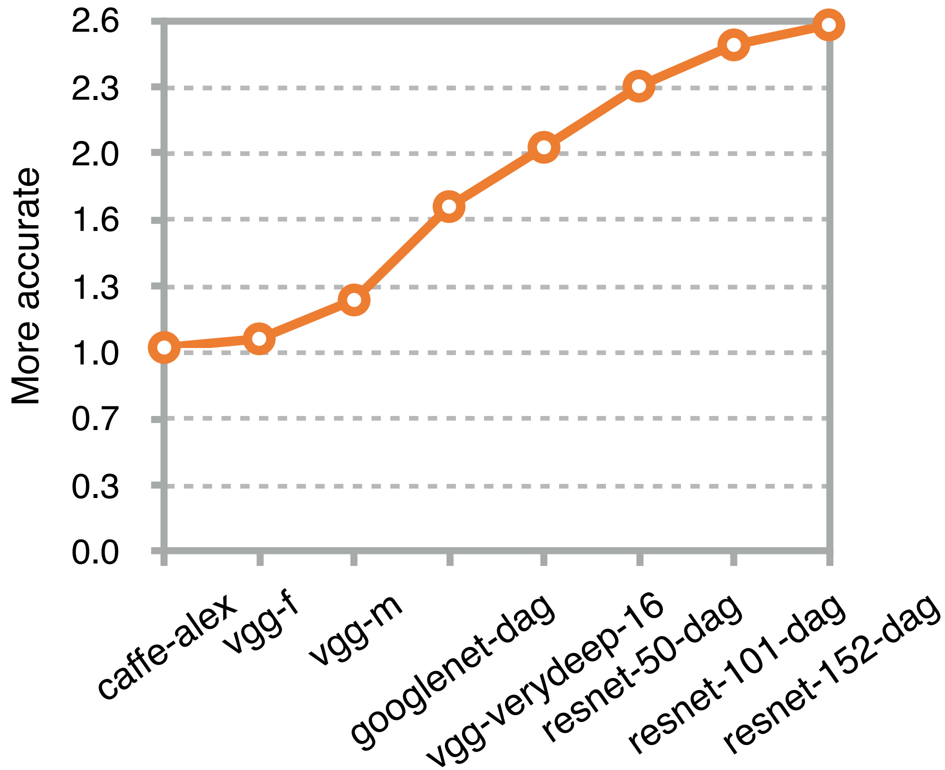
1000 classes; 1ms for an image; 92.5% recognition rate

http://demo.caffe.berkeleyvision.org



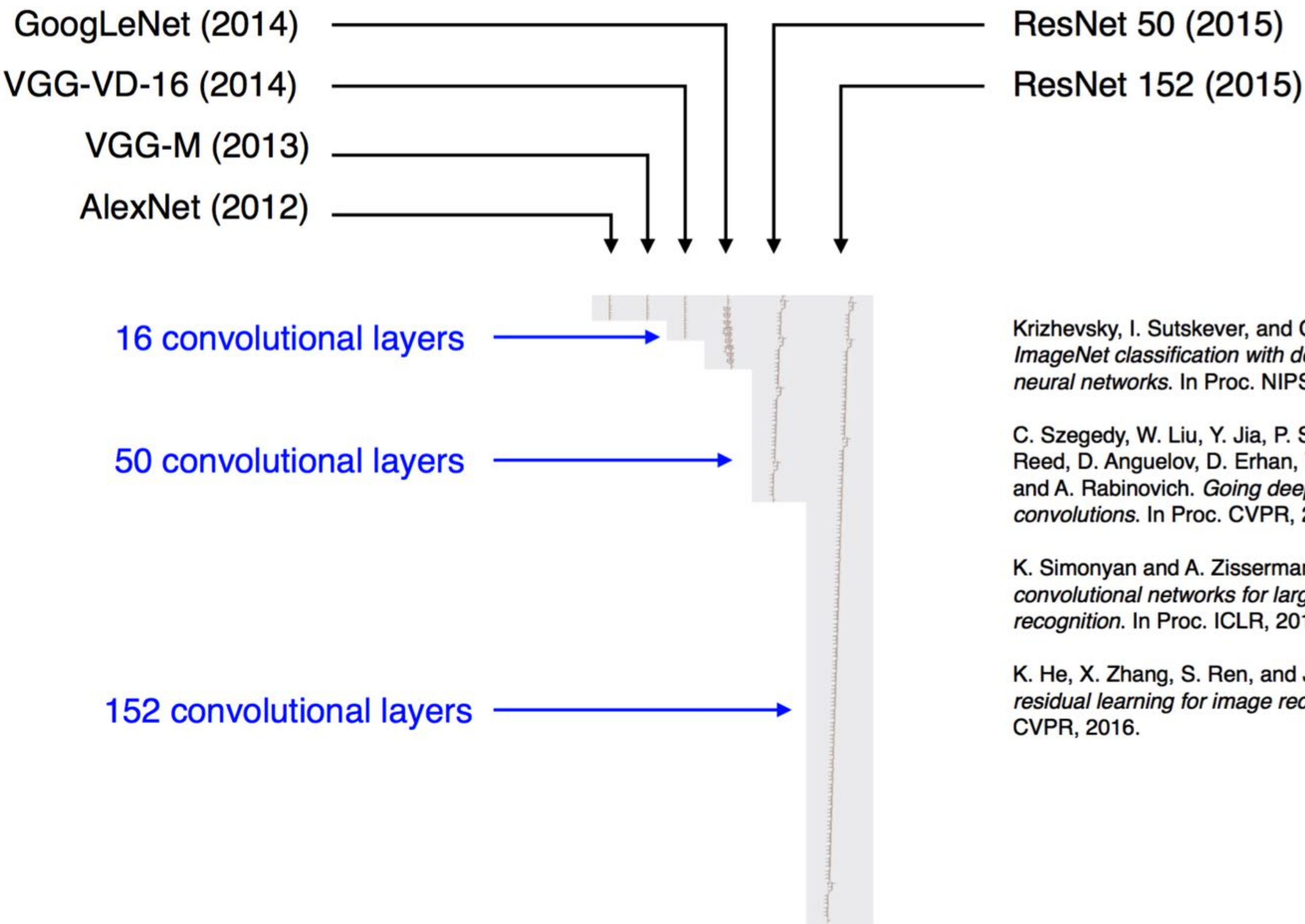
Architectures get deeper





~ 2.6x improvement in 3 years

Architectures get deeper



Krizhevsky, I. Sutskever, and G. E. Hinton. *ImageNet classification with deep convolutional neural networks*. In Proc. NIPS, 2012.

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. *Going deeper with convolutions*. In Proc. CVPR, 2015.

K. Simonyan and A. Zisserman. *Very deep convolutional networks for large-scale image recognition*. In Proc. ICLR, 2015.

K. He, X. Zhang, S. Ren, and J. Sun. *Deep residual learning for image recognition*. In Proc. CVPR, 2016.

Computer Vision: Semantic Segmentation

Badrinarayanan et al. 2015
SegNet: A Deep Convolutional Encoder-Decoder Architecture for Robust Semantic Pixel-Wise Labelling

Computer Vision: Semantic Segmentation

Input Groundtruth Output

Yang He; Wei-Chen Chiu; Margret Keuper; Mario Fritz
STD2P: RGBD Semantic Segmentation Using Spatio-Temporal Data-Driven Pooling
IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017

max planck institut
informatik

SIC Saarland Informatics Campus

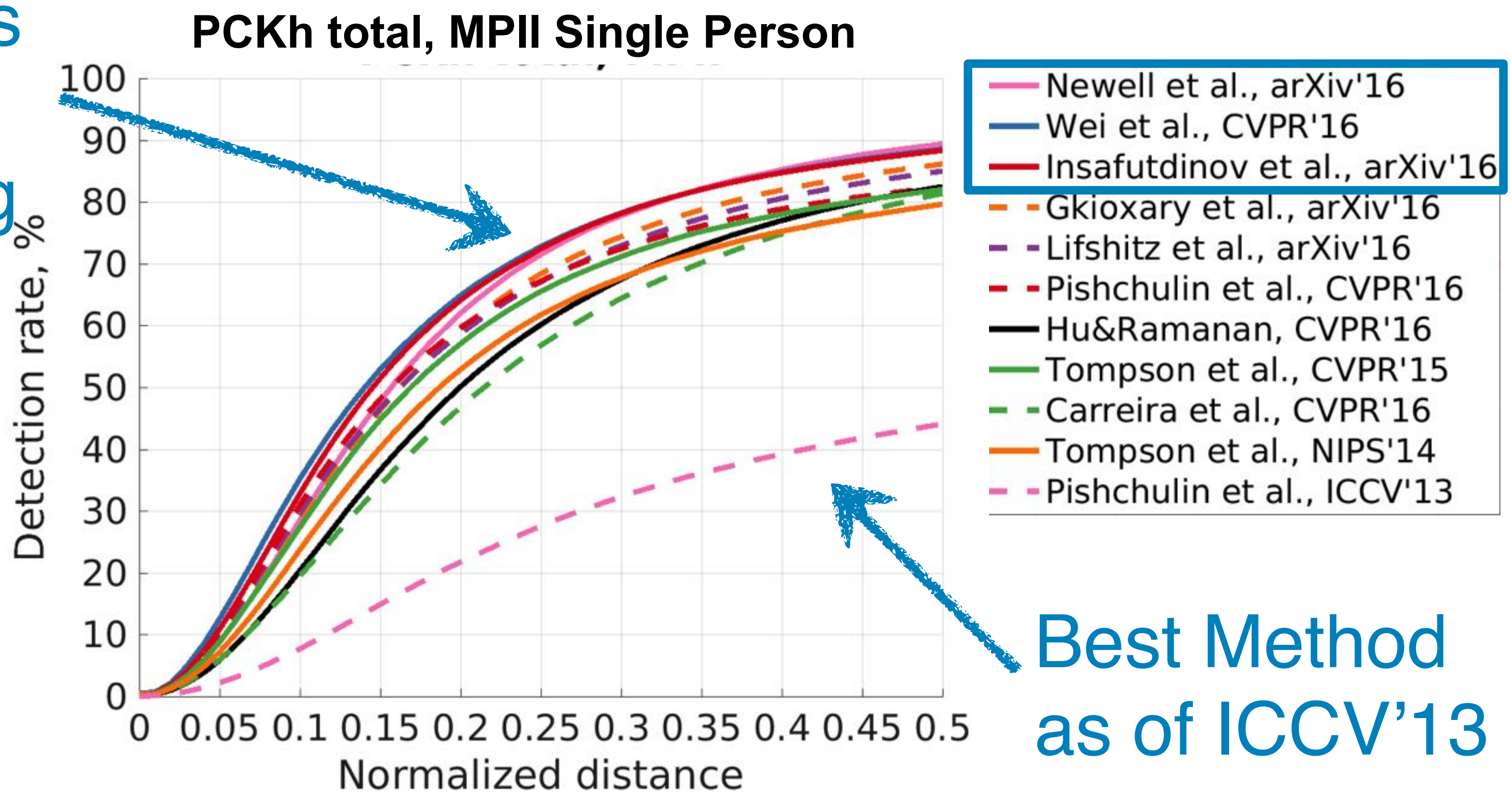
Deep(er)Cut: Joint Subset Partition & Labeling for Multi Person Pose Estimation

DeepCut: Joint Subset Partition and Labeling for Multi-Person Pose Estimation
L. Pishchulin, E. Insafutdinov, S. Tang, B. Andres, M. Andriluka, P. Gehler, and B. Schiele, CVPR'16

DeeperCut: A Deeper, Stronger, and Faster Multi-Person Pose Estimation Model
E. Insafutdinov, L. Pishchulin, B. Andres, M. Andriluka, and B. Schiele, ECCV'16

Analysis - overall performance

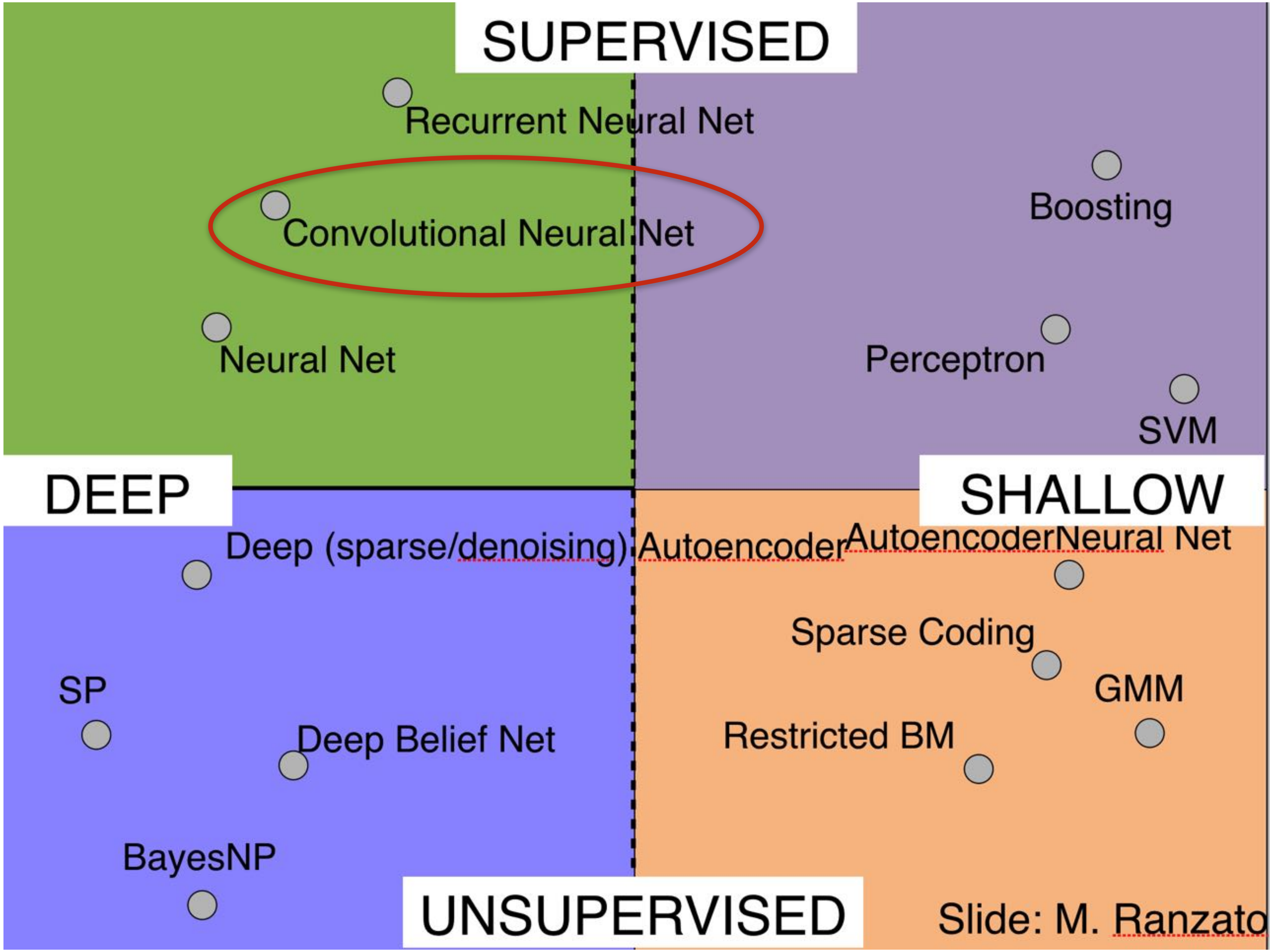
Best Methods now: deep learning "takes" over



Best Method as of ICCV'13

- ✓ since CVPR'14, dataset has become **de-facto standard benchmark**
- ✓ **large training set** facilitated development of **deep learning methods**

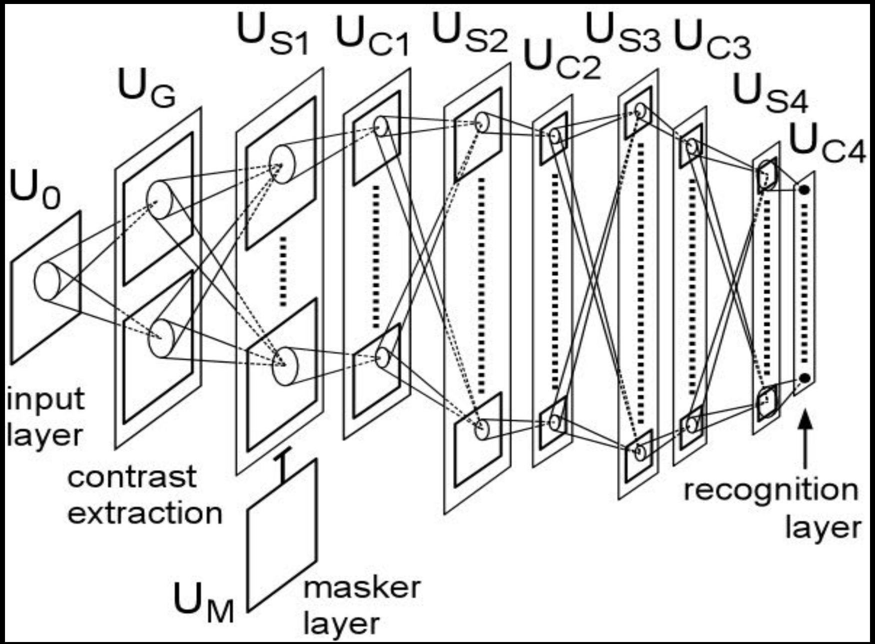
Overview of Deep Learning



Multistage Hubel&Wiesel Architecture

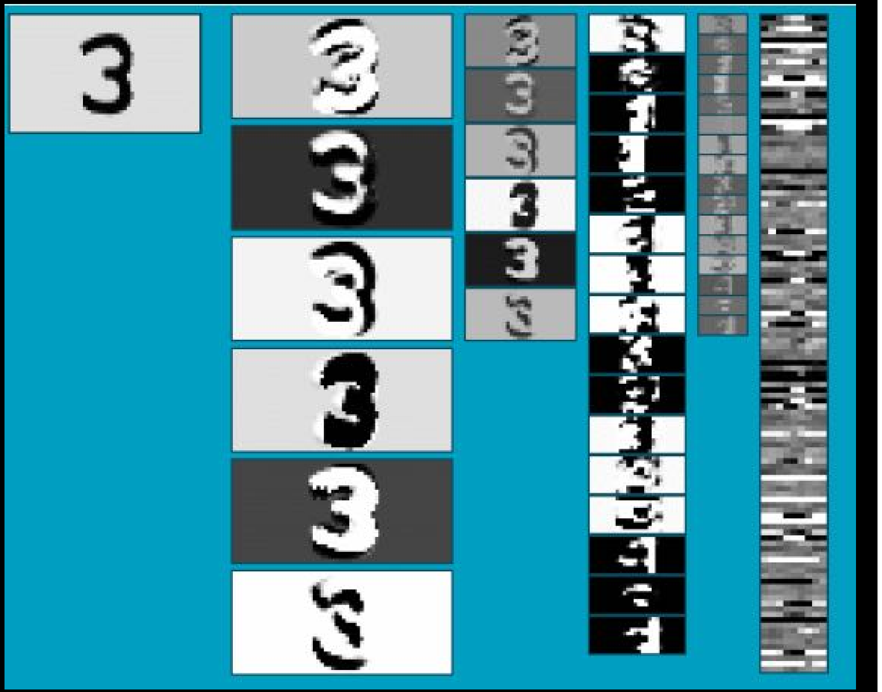
Slide: Y.LeCun

- [Hubel & Wiesel 1962]
  - simple cells detect local features
  - complex cells “pool” the outputs of simple cells within a retinotopic neighborhood.



Cognitron / Neocognitron [Fukushima 1971-1982]

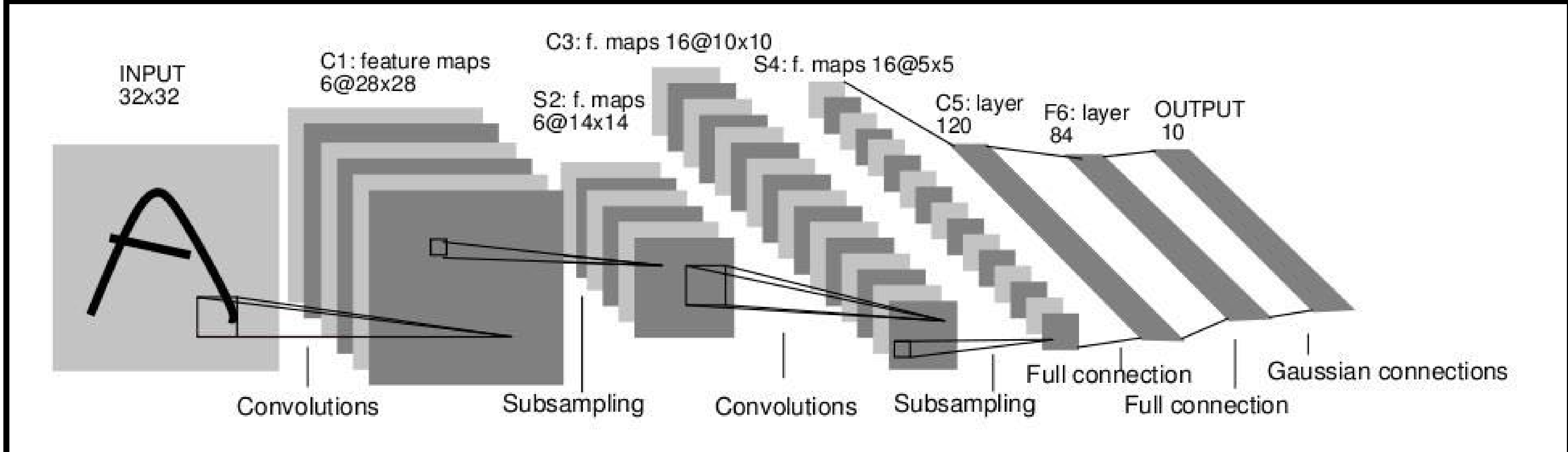
- Also HMAX [Poggio 2002-2006]



Convolutional Networks [LeCun 1988-present]

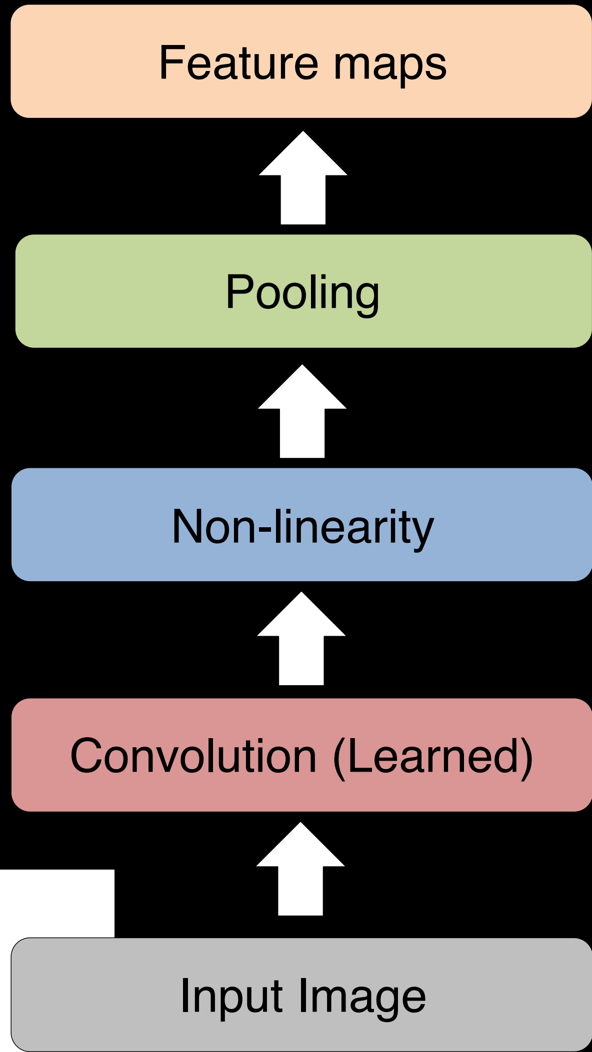
Convolutional Neural Networks

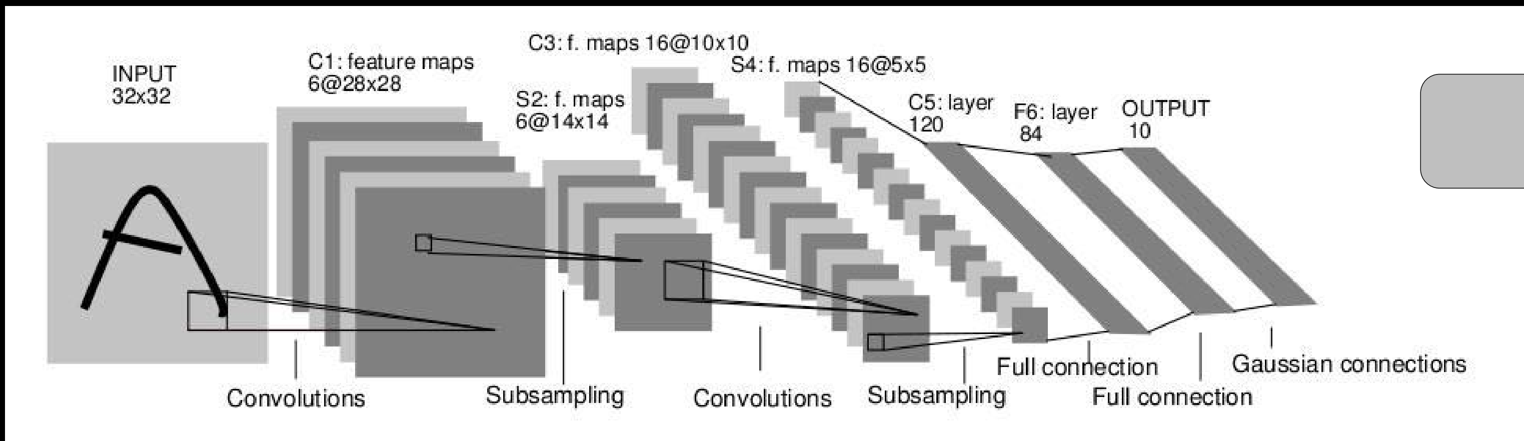
- LeCun et al. 1989
- Neural network with specialized connectivity structure



Characteristics of Convnets

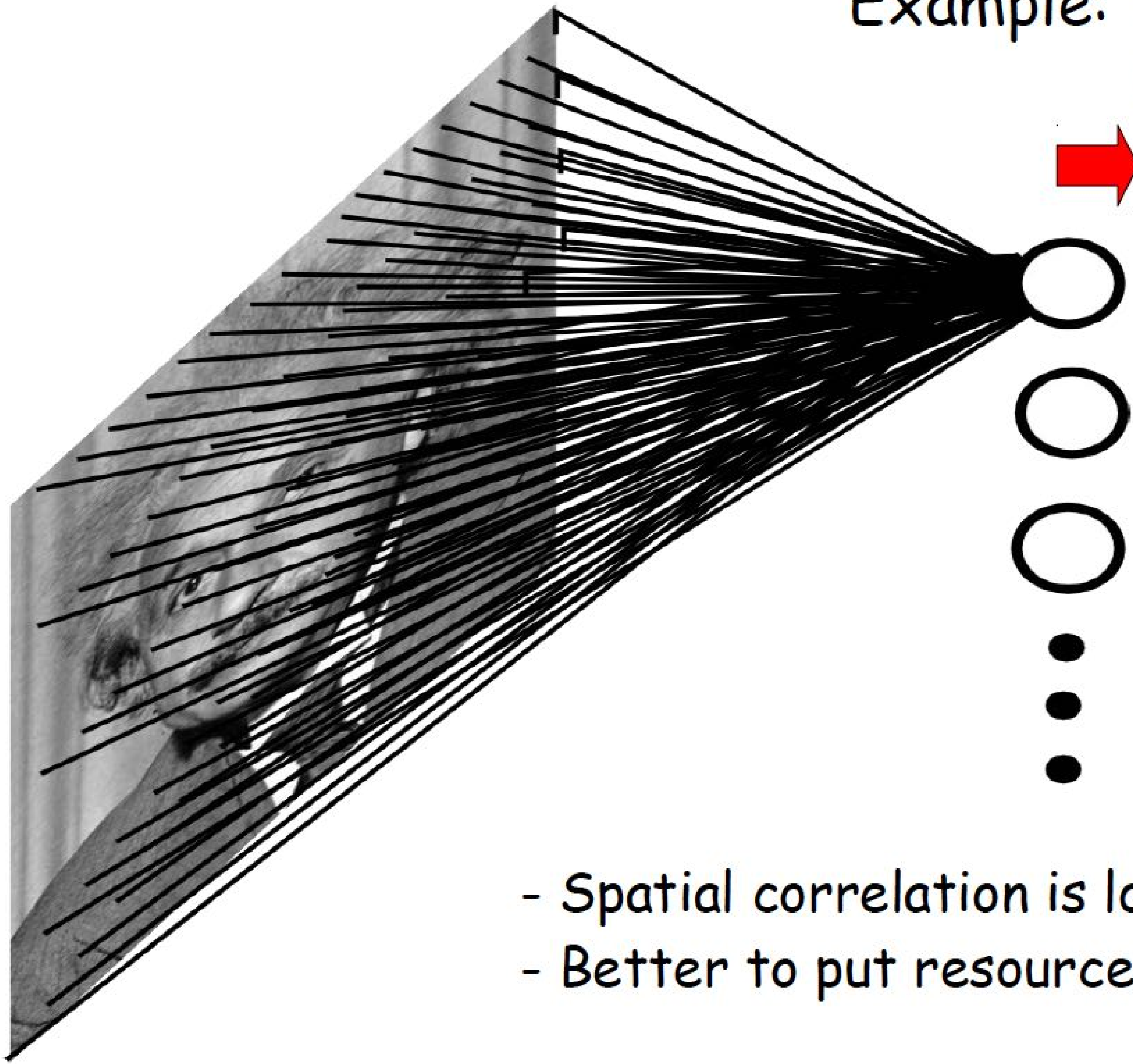
- Feed-forward:
  - Convolve input
  - Non-linearity (rectified linear)
  - Pooling (local max) / (=subsampling)
- Supervised
- Train convolutional filters by back-propagating classification error





[LeCun et al. 1989]

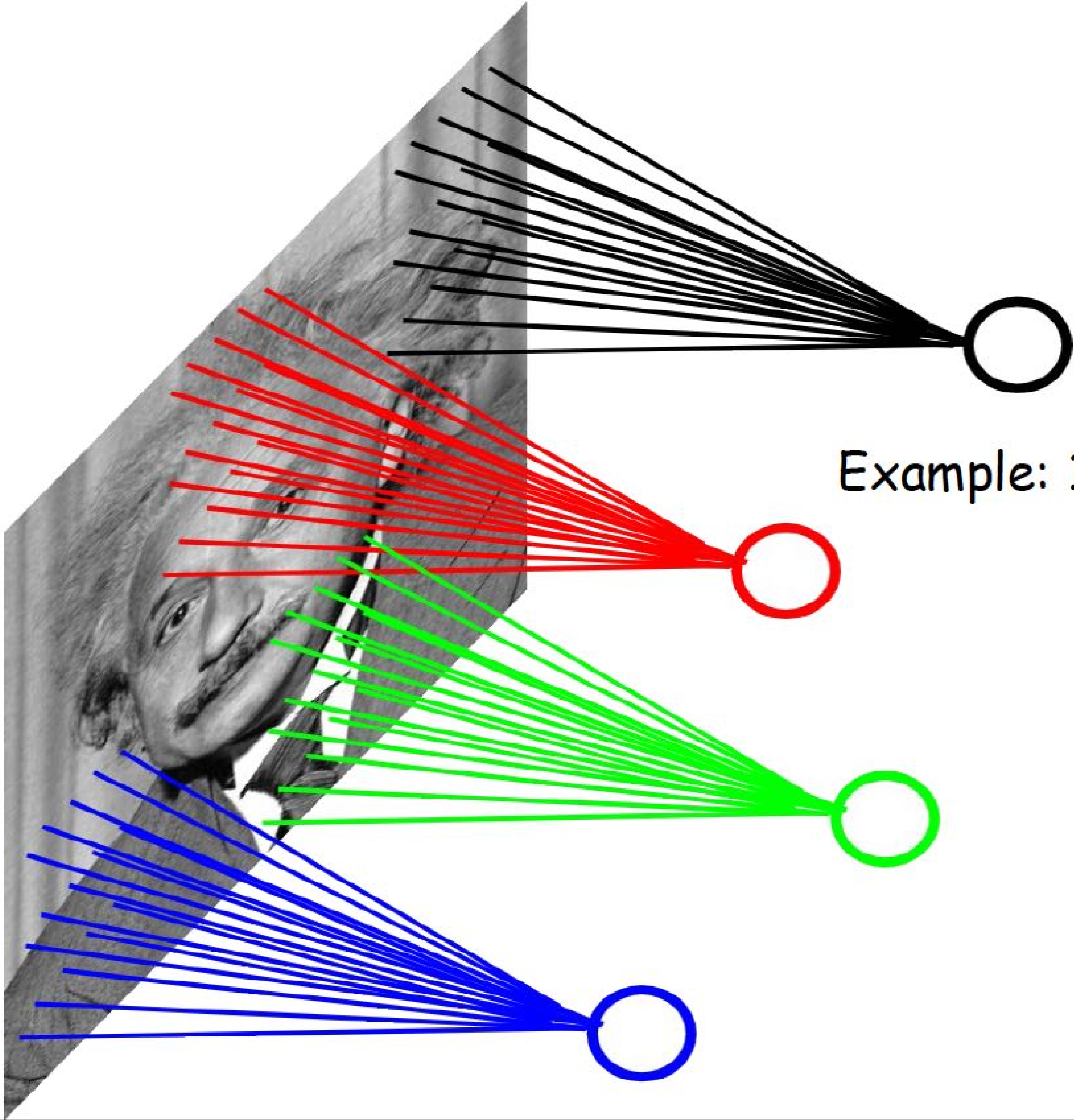
FULLY CONNECTED NEURAL NET



Example: 1000x1000 image
1M hidden units
→ 10^12 parameters!!!

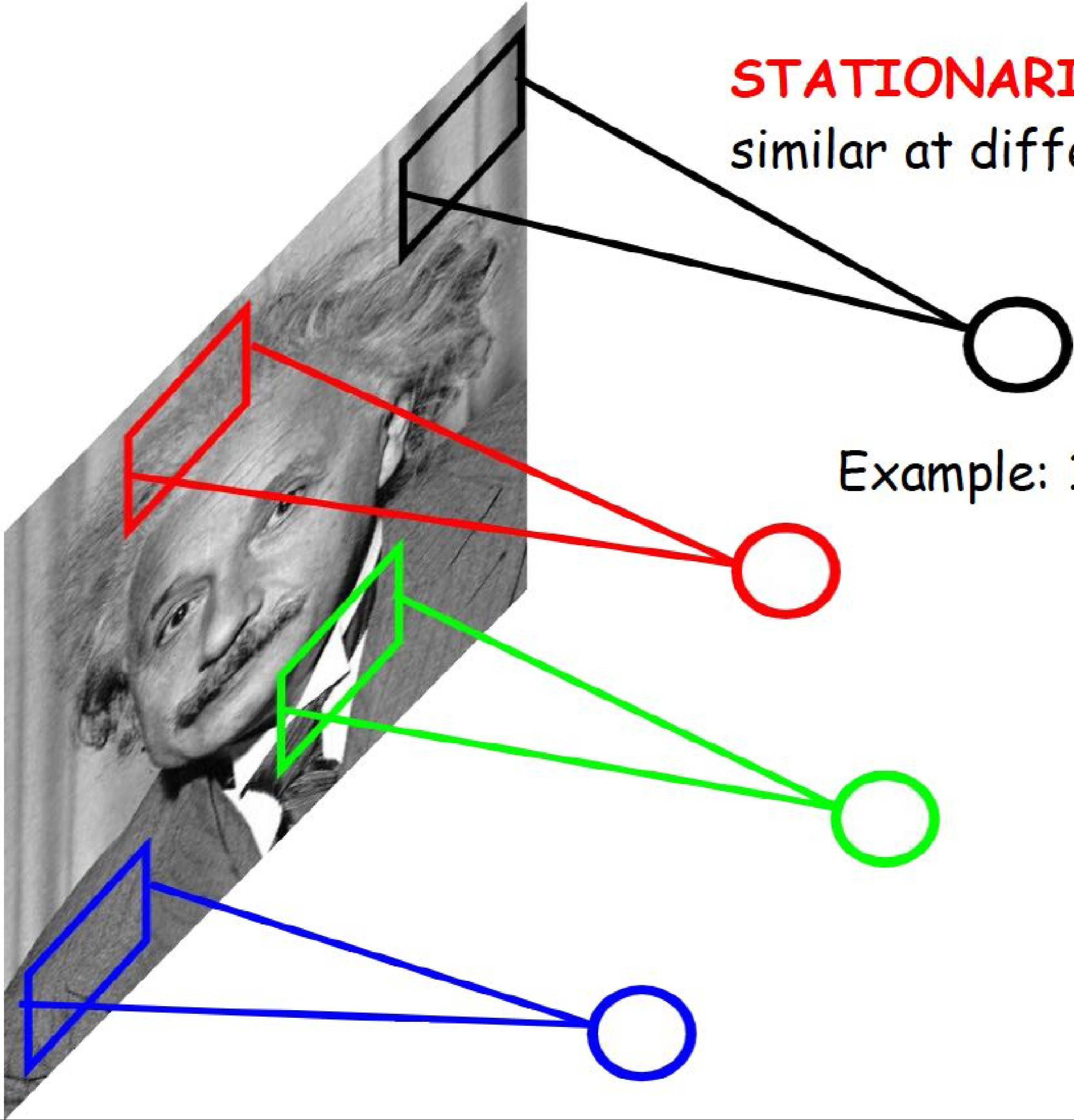
- Spatial correlation is local
- Better to put resources elsewhere!

LOCALLY CONNECTED NEURAL NET



Example: 1000x1000 image
1M hidden units
Filter size: 10x10
100M parameters

LOCALLY CONNECTED NEURAL NET

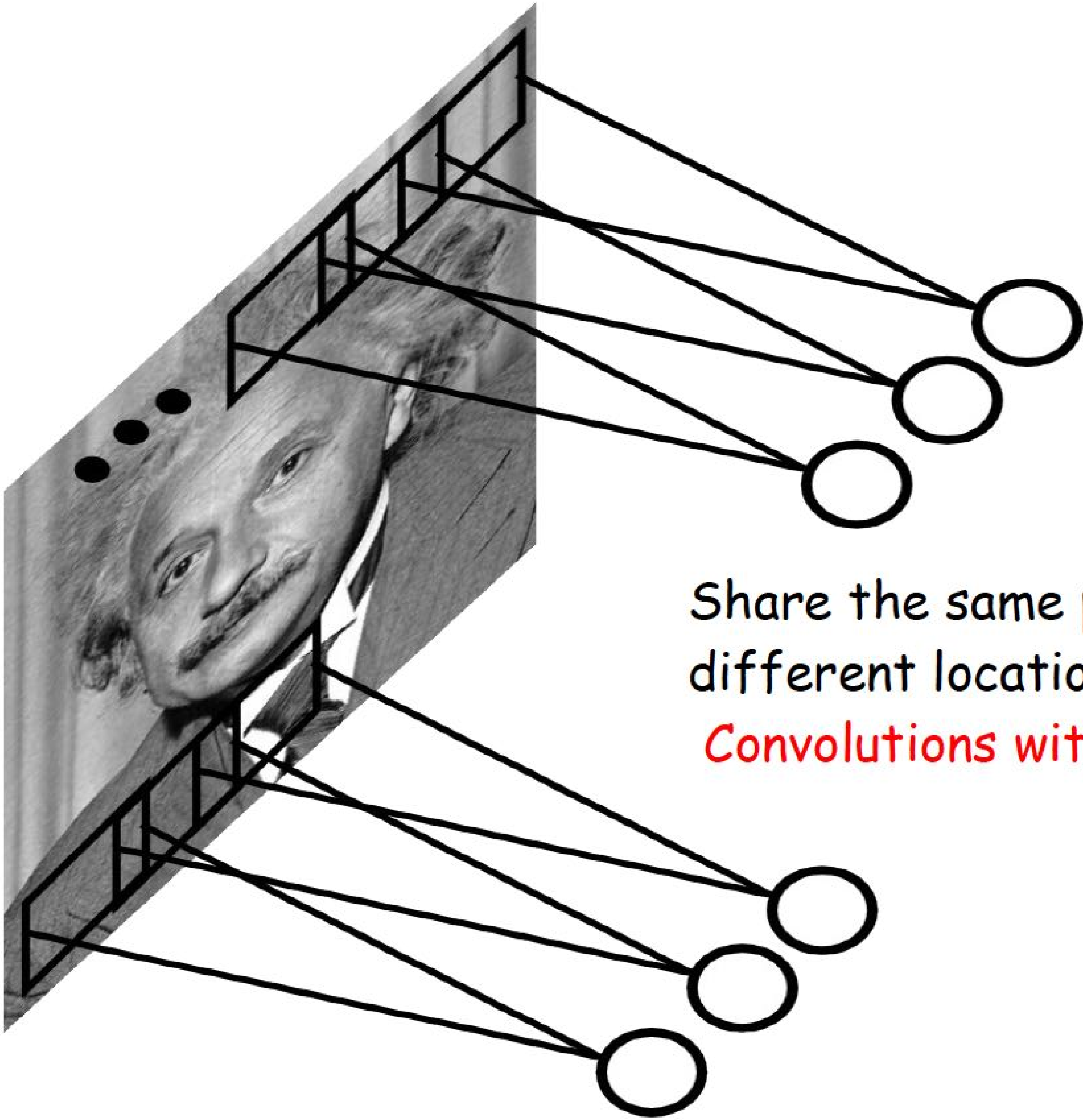


STATIONARITY? Statistics is similar at different locations

Example: 1000x1000 image
1M hidden units
Filter size: 10x10
100M parameters

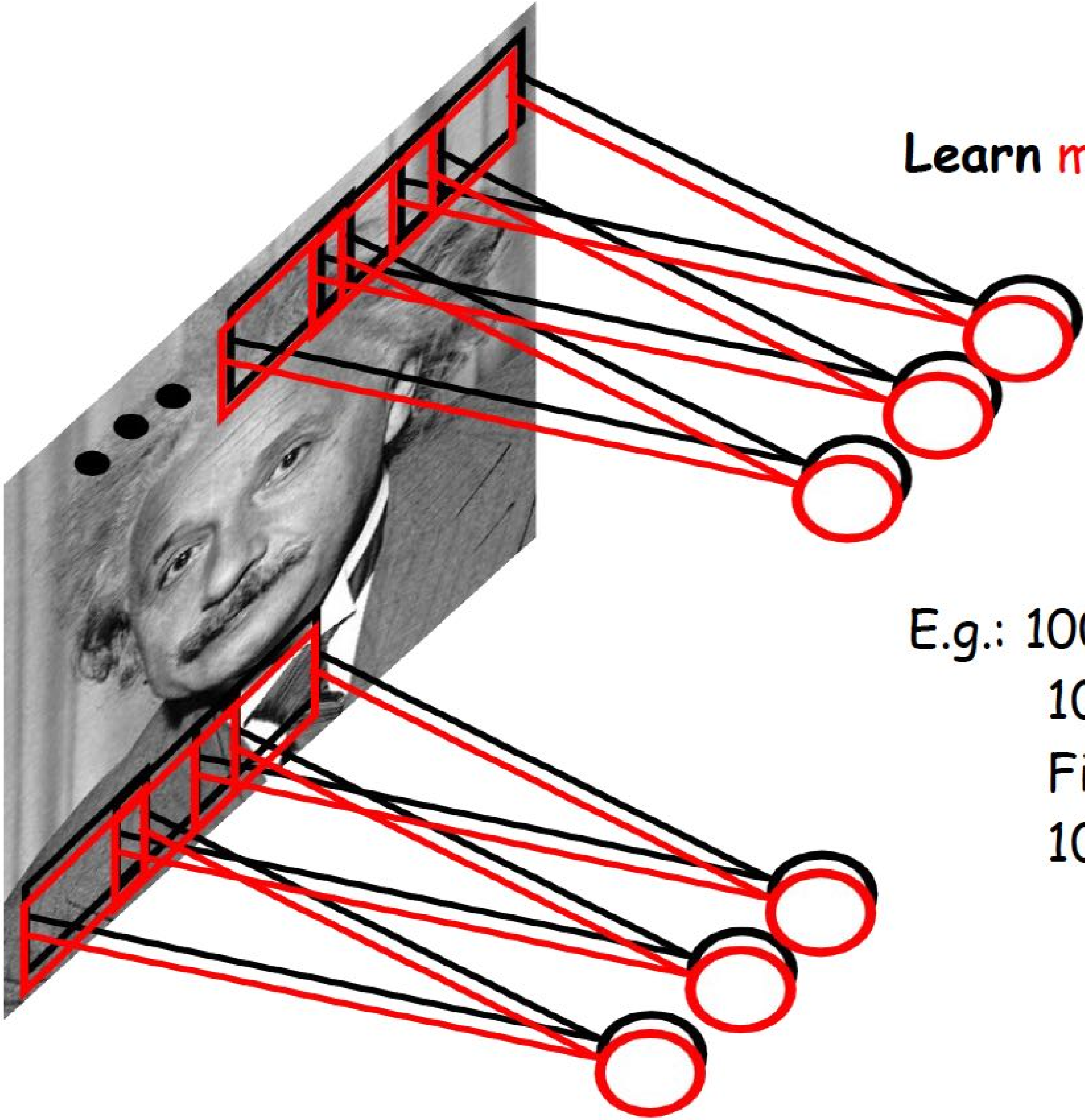
Ranzato

CONVOLUTIONAL NET



Share the same parameters across different locations:
Convolutions with learned kernels

CONVOLUTIONAL NET



Learn multiple filters.

E.g.: 1000x1000 image
100 Filters
Filter size: 10x10
10K parameters

NEURAL NETS FOR VISION

A standard neural net applied to images:

- scales quadratically with the size of the input
- does not leverage stationarity

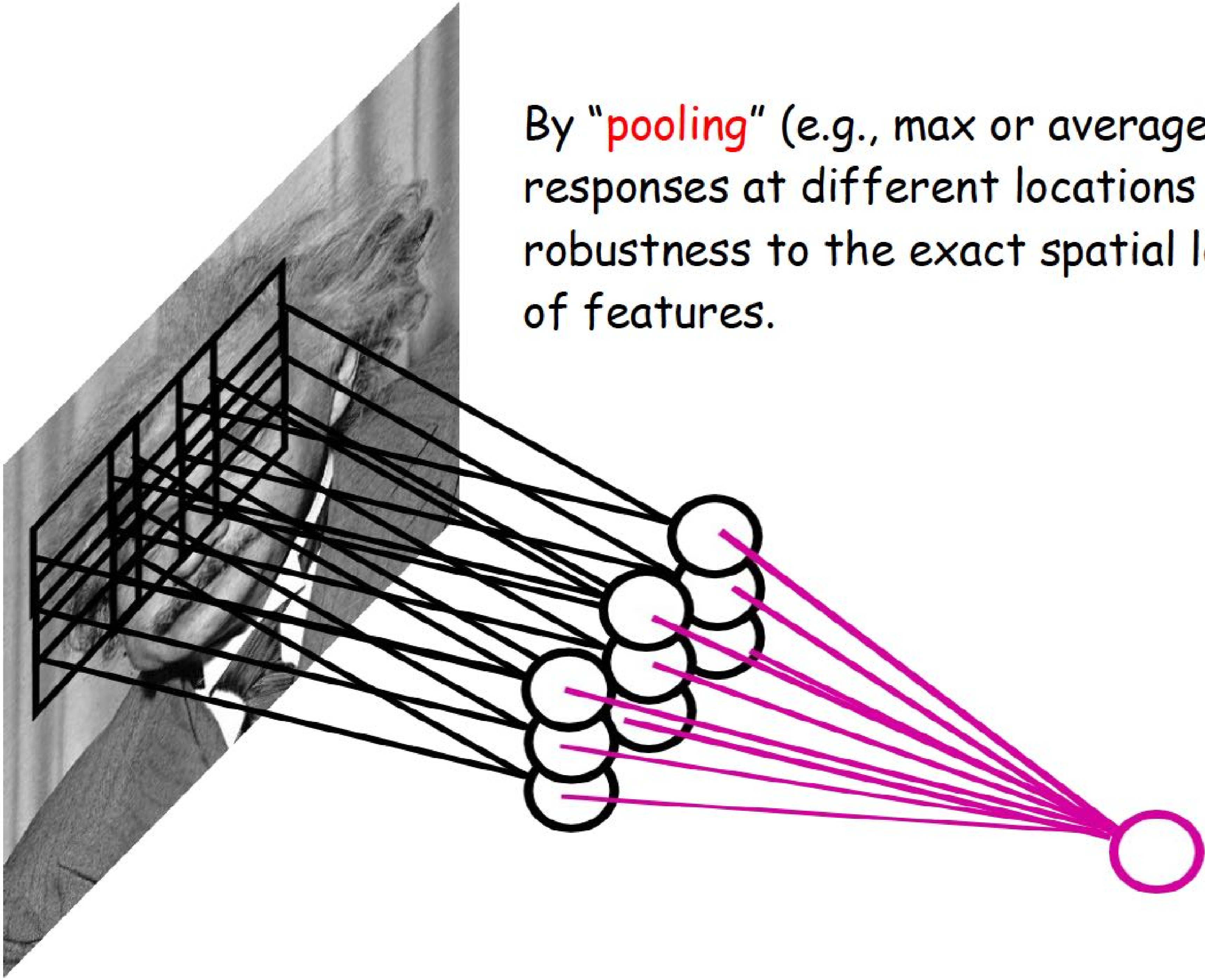
Solution:

- connect each hidden unit to a small patch of the input
- share the weight across hidden units

This is called: **convolutional network.**

LeCun et al. "Gradient-based learning applied to document recognition" IEEE 1998

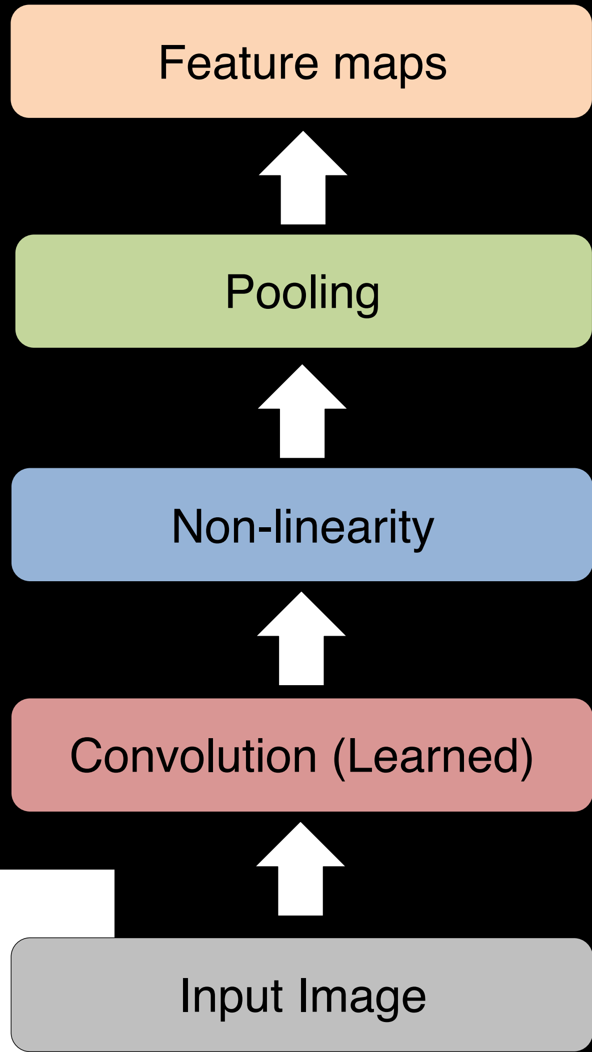
CONVOLUTIONAL NET

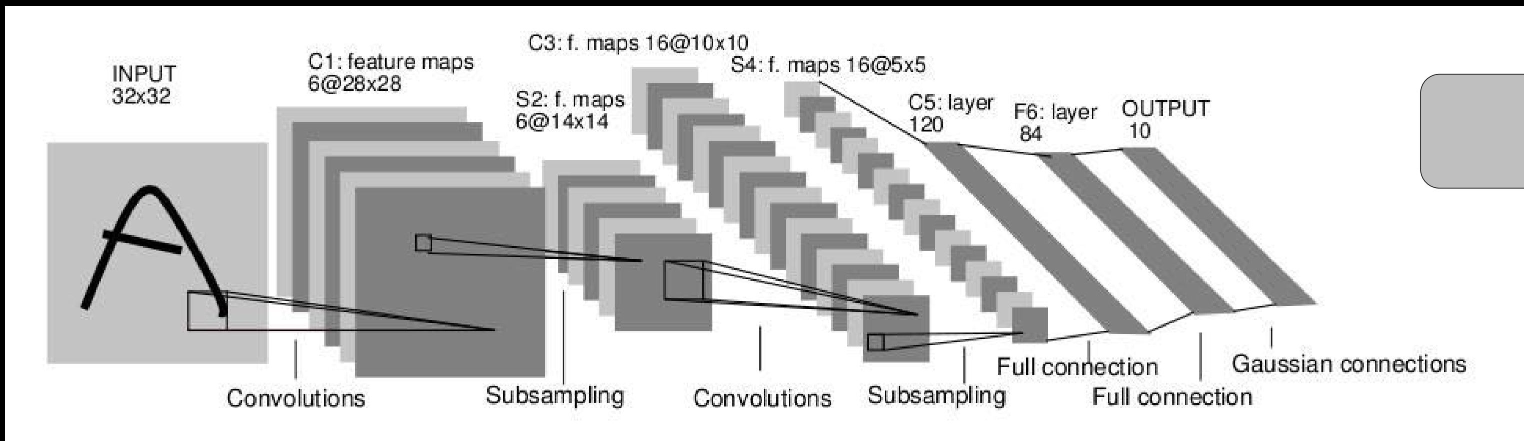


By "pooling" (e.g., max or average) filter responses at different locations we gain robustness to the exact spatial location of features.

Characteristics of Convnets

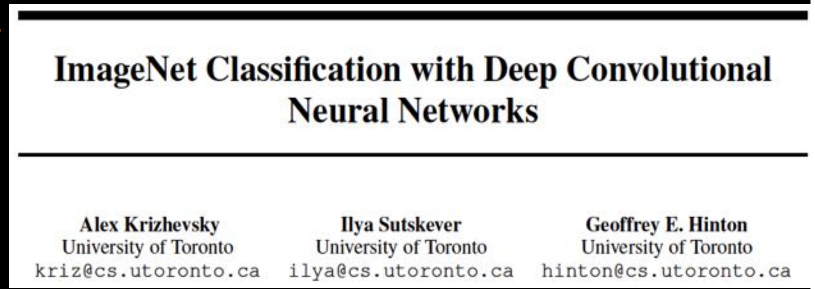
- Feed-forward:
  - Convolve input
  - Non-linearity (rectified linear)
  - Pooling (local max) / (=subsampling)
- Supervised
- Train convolutional filters by back-propagating classification error





[LeCun et al. 1989]

Krizhevsky et al. [NIPS2012]

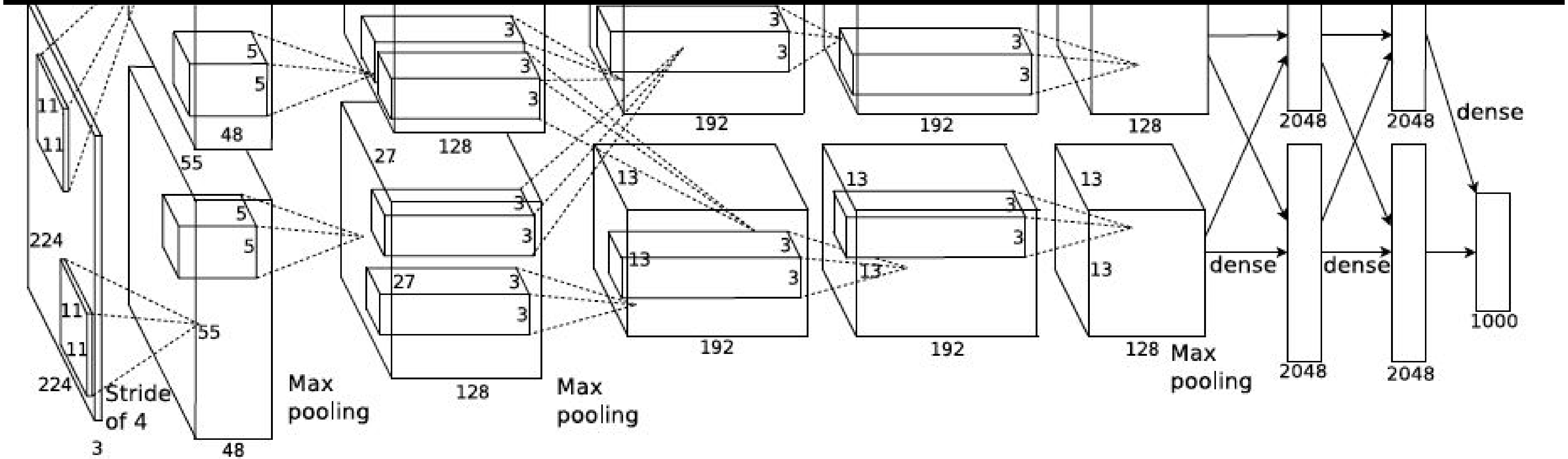

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

- Same model as LeCun'98 but:
  - Bigger model (8 layers)
    - More data (10^6 vs 10^3 images)
    - GPU implementation (50x speedup over CPU)
    - Better regularization (DropOut)



- 7 hidden layers, 650,000 neurons, 60,000,000 parameters
- Trained on 2 GPUs for a week

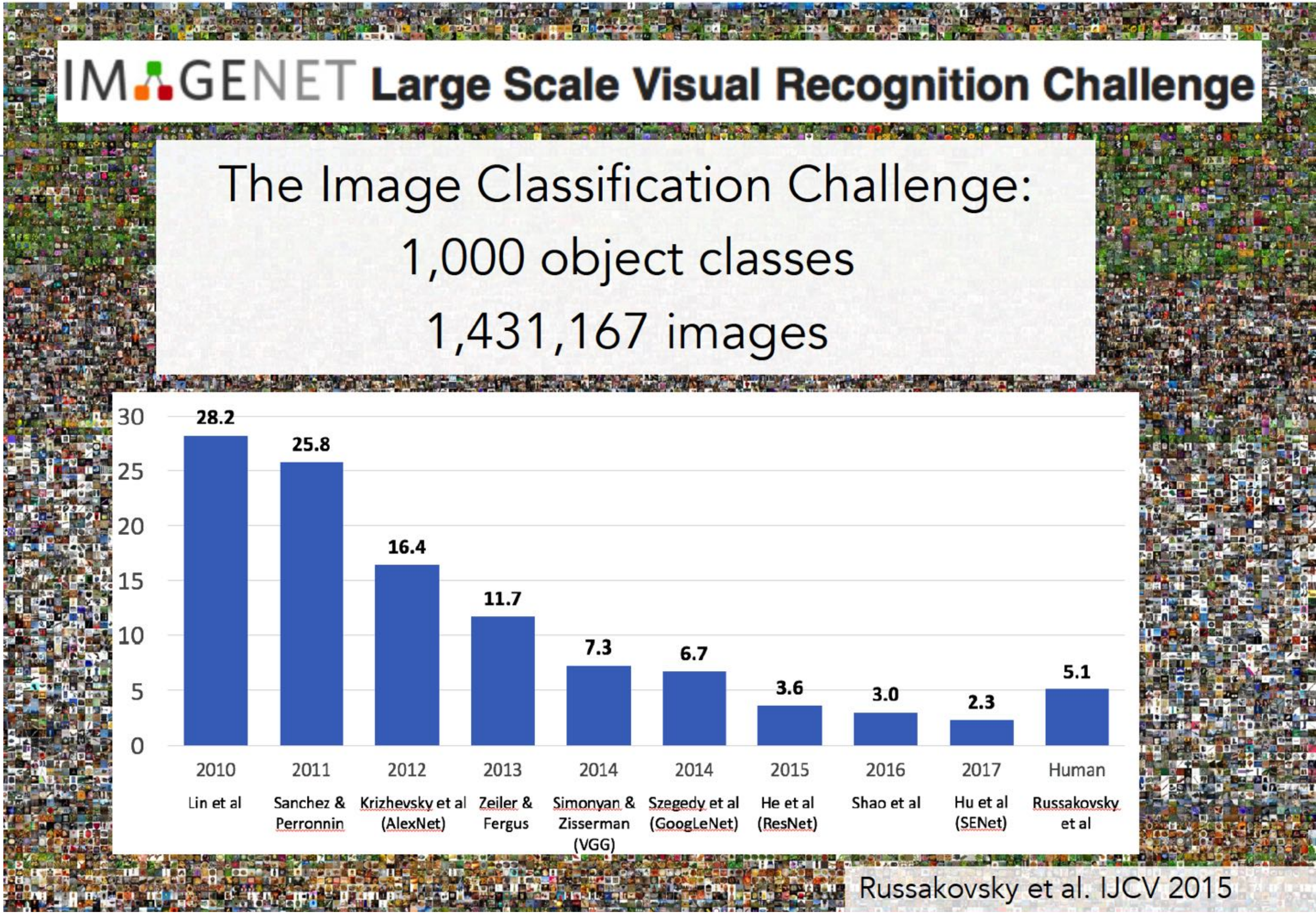
IMAGENET Large Scale Visual Recognition Challenge

The Image Classification Challenge:
1,000 object classes
1,431,167 images

Output:		Output:	
Scale T-shirt Steel drum Drumstick Mud turtle	✔	Scale T-shirt Giant panda Drumstick Mud turtle	✗

Russakovsky et al. IJCV 2015

slide credit: Fei-Fei, Justin Johnson, Serena Yeung



slide credit: Fei-Fei, Justin Johnson, Serena Yeung